WEB BASED AUTOMATED E-INVOICING SYSTEM

ELISABETH LEE MEI JIN

UNIVERSITI TUNKU ABDUL RAHMAN

WEB BASED AUTOMATED E-INVOICING SYSTEM

ELISABETH LEE MEI JIN

A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Software Engineering with Honours

Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

September 2025

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : 

Name : Elisabeth Lee Mei Jin

ID No. : 2104817

Date : 19/9/2025

COPYRIGHT STATEMENT

This final year project report is submitted in partial fulfilment of the requirements for the degree of Bachelor of Software Engineering (Honours) at Universiti Tunku Abdul Rahman (UTAR). This final year project report represents the work of the author, except where due acknowledgement has been made in the text. No part of this final year project report may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

ACKNOWLEDGEMENTS

I would like to sincerely thank my supervisor, Ts Dr Sugumaran a/l Nallusamy, which had been my supervisor for the past two trimesters. He had undoubtedly provided invaluable guidance, encouragement, and patience throughout the development of this project. His insights and feedback have played an instrumental part in shaping both the direction and quality of my work.

Lastly, I would like to also express my heartfelt gratitude to my family, friends, and peers for their encouragement, understanding, and moral support throughout this journey. Their presence and assistance had been a constant source of motivation and had contributed tremendously to the success of the development of this project.

ABSTRACT

This project focuses on the development of an e-Invoice system designed to modernize and automate invoice generation, submission, and verification in compliance with regulatory standards. The purpose of this project is to address the challenges in manual invoicing, including manual errors, lack of transparency in manual invoicing, and the increasing number of fraud risks that had been a pervasive problem within the domain of financial technology and digital tax compliance. The existing problems motivating this project stems from the need for a secure, auditable, and standardized process for generating and submitting invoices, which in turn, ensures data integrity and legal compliance. To address this, a modular approach was adopted which involves mapping invoice data to the UBL 2.1 JSON standard, performing canonicalization for deterministic representation, and cryptographically signing it before submission. Besides, the system integrates with LHDN tax authority APIs for asynchronous validation, generates QR codes for verification, supports cancellation of invoices, notifications, and maintains a ledger for reporting and reconciliation. The preliminary results demonstrate that the platform reliably produces compliant e-invoices, which also accurately detect data mismatches, and enables taxpayers to track invoice status in real time, while QR code verification enhances transparency for end users. The significance of this research lies in its ability to streamline the invoicing process, reduce human error, and provide an auditable digital record, contributing to greater efficiency and regulatory compliance. Therefore, this project has demonstrated a scalable framework, which includes document signing, submission automation, and real-time verification in financial technology applications.

Keywords: e-Invoice, LHDN MyInvois Integration, RESTful API, Web Application Development, API Integration, OAuth2 Authentication, Digital Signature, Tax Compliance.

Subject Area: QA76.75-76.765 Computer software

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

ERP	Enterprise Resource Planning
LHDN	Lembaga Hasil Dalam Negeri / Inland Revenue Board
API	Application Programming Interface

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 General Introduction

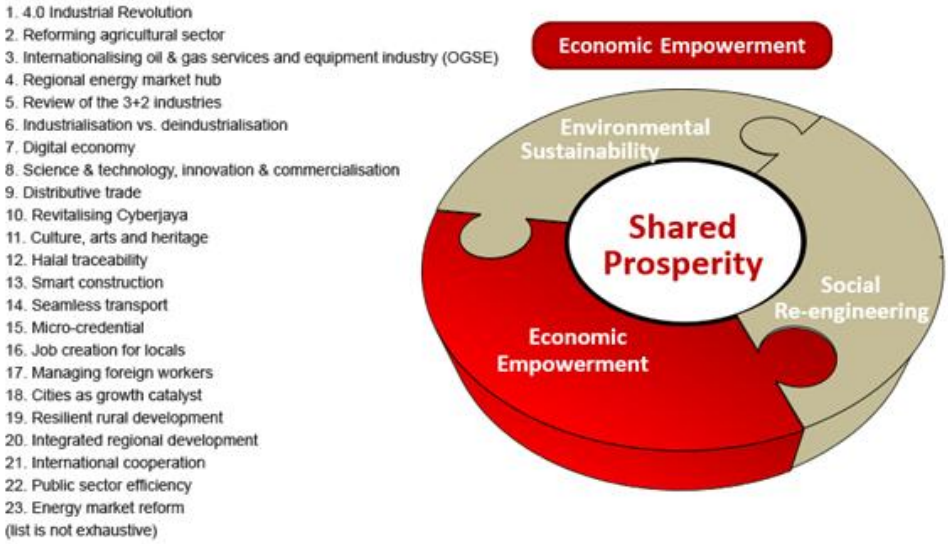


Figure 1.1 Twelfth Malaysia Plan, 2016-2020 (MyGOV - The Government of Malaysia's Official Portal, 2025)

Malaysia is one of the countries that is rapidly adopting to the digital transformation landscape in several areas such as finance, commerce and taxation. Specifically in the taxation field, the Malaysian government plans to undertake the implementation of e-invoices, which aims to streamline business transactions, improve tax compliance, as well as improve taxation operations. This initiative aligns with the Twelfth Malaysia Plan, which supports the implementation of e-Invoicing in Malaysia in order to facilitate the growth of the digital economy by improving the digital services infrastructure and modernizing tax administration through digitalization (MyGov, 2025). Therefore, the Inland Revenue Board of Malaysia (LHDN) has gradually rolled out the implementation of e-invoice in stages to taxpayers involved in businesses-to-business (B2B), business-to-consumers (B2C), and business-to-government (B2G).

The taxation process has previously been traditionally done manually, which mainly involves medium-sized businesses to record invoices in the accounting software. An alternative to this is the paper-based invoices, particularly used by small-sized taxpayers for record-keeping of the taxes. However, these methods are still considered manually done as it requires the taxpayers to manually fill in each of the fields. This in turn, raises the complexity of filing for taxes, causing frustration among taxpayers. Therefore, the implementation of e-invoice aims to combat these complexities.

An e-invoice can be defined as a structured digital document that documents a transaction between a buyer and a supplier. It contains essential information such as the supplier's and buyer's details, the description of the item, quantity, price, tax, and total amount, which are essential information in documenting transaction data for routine business activities. In contrast to conventional paper invoices, e-invoices need to be created according to LHDN's specified format, which is in machine-readable format, like JSON or XML, in order for LHDN to validate them. With this implementation, it eliminates the necessity for paper-based invoices by enabling businesses to generate and store digitized version of their invoices. Thus, streamlining the entire process for billing and payments.

The outcome of the project will strengthens the tax-compliance and reduce the complexities dealing with the recordance of e-invoices which was previously done through manual methods. With the completion of this project, it has the potential to significantly improve the process of submitting and reporting taxes to the Inland Revenue Board of Malaysia (LHDN) with the integration of API into the ERP system.

1.2 Problem Statement

The implementation and integration of e-invoice into the enterprise resource planning system (ERP) has numerous challenges stemming from the absence of certain technologies and constraints. Without a doubt, these constraints will eventually be an obstacle in the development process, and could potentially cause a cascading effect, leading to other potential problems. Hence, it is

important to address the root cause of the existing problem related to the traditional approach of invoicing itself.

1.2.1 Manual process in recording invoices

Traditional paper-based invoices, which are done manually, have been the norm for many years in generating an invoice. This norm typically involves a paper-based procedure, which includes manually filling up invoices and providing printed copies to the customer via mail or fax (Gustafson, 2023). This manual method of generating an invoice is plagued with inefficiencies as it is time-consuming, uses high usage of resources, and increases the likelihood of human error. The physical invoicing process also diverts financial staff's time away from potentially higher-value work, as the invoicing process itself involves the laborious process of creating, processing, and mailing invoices, which may be a multi-step or day-long process for firms or businesses. Furthermore, any attempt at invoice personalisation or customisation takes an immense amount of time to implement when employing a manual, physical invoicing method. As a result, this manual handling of invoice generation can potentially disrupt cash flow management and increase operational costs for businesses.

1.2.2 Lack of transparency and visibility

Traditional invoicing systems have a key disadvantage in that they are unable to monitor business activities in real-time. This is due to the manual invoice processing procedures that occur offline, making it tough for the provider or a third party, such as a finance institution, to figure out the actual condition of the instrument or the status of the transaction, as they are unable to follow and oversee the entire process (InvoiceMate, 2021). This lack of visibility makes the process of monitoring the status of tax filings and auditing immensely difficult due to the lack of real-time validation and the absence of a centralised system (B2BE, n.d.). Hence, traditional invoicing inadvertently makes it significantly difficult for tax authorities to enforce compliance and detect fraudulent activities promptly.

1.2.3 Fraud and tax evasion risk

Aside from compliance difficulties stemming from the old approach to manual invoicing, this particular approach has also increased the possibility of fraud and tax evasion. In this context, invoice fraud can be defined as a deceitful practice in which criminals send false or altered invoices to unwary receivers (Wejke, 2023). These invoices are typically designed to look and simulate real payment demands in order to mislead the receiver into paying for products or services they never received. As a result, this causes businesses to pay for these fraudulent activities, which causes the company itself to lose money. Furthermore, a manual invoicing system also makes it easier for fraudulent activities such as tax evasion to take place. The modus operandi for tax evasion typically involves fraudulently tampering with the tax amount to be declared by underreporting the tax amount or to use it as a means to mask illegal operations. As a result, the invoice fraud and tax evasion result in significant loss of tax revenue for the government, making it a major contributing factor in the implementation of e-invoicing in Malaysia.

1.3 Project Objectives

This project's goal is to ease businesses in navigating through the complexities in implementing e-invoicing in an ERP system. This in turn, will assist businesses in the digitalization of their business transactions. Hence, contributing to the digital economy.

Objectives:

1. To establish a centralized database for standardized invoice management and seamless integration with Malaysian tax authorities' systems.
2. To develop an automated web-based e-invoicing system that is capable in generating, validating and tracking invoices in real time.
3. To design a secure e-invoicing system using encryption and digital signature technologies to protect sensitive financial data from unwanted access.
4. To verify the system's productivity and efficiency by automating invoice and payment reconciliation to reduce manual mistakes and improve financial accuracy.

1.4 Proposed Solution

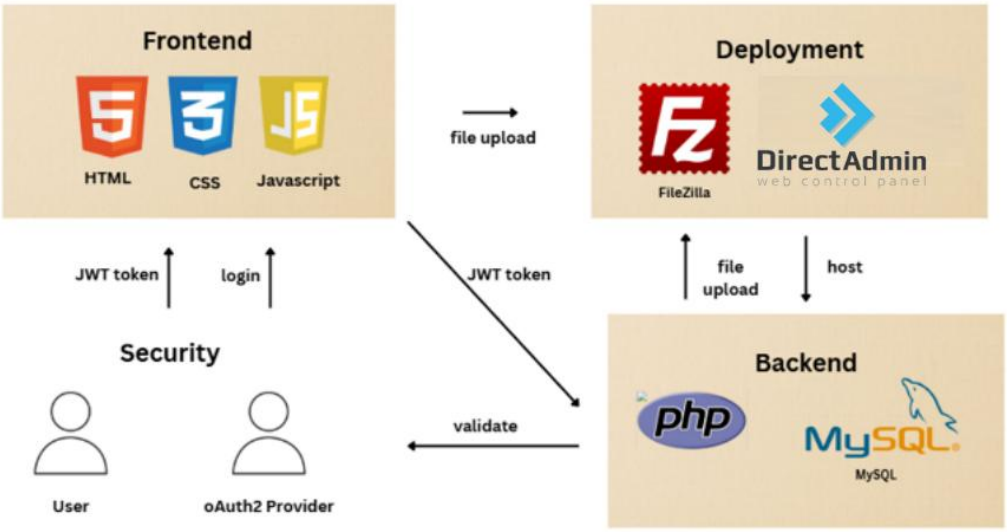


Figure 1.2 Overview of the Proposed Solution

The proposed solution involves the development of a web-based invoicing system which entirely operates with LHDN's MyInvois system in order to allow easy submission, validation, and processing of invoices in accordance with Malaysia's digital tax regulations. The platform was deployed on a Direct Admin web hosting platform, which provided a secure and easy-to-manage environment to process invoicing activities. Besides that, the hosting platform also serves primarily for domain management, SSL certificate integration, security configurations design and server jobs automation (cPanel, n.d.).

For back-end development, the system will be developed using the PHP programming language. The reason behind using this particular server-side script language is due to its ability to handle Application Programming Interface (API) interactions, database integration, and security. Furthermore, PHP is also suitable for facilitating communication with LHDN's MyInvois API, whereby invoices are encoded in JSON format and validated conveniently. Aside from that, the MySQL database was used to store and keep track of invoice history, taxpayer data, business transactional logs and financial data.

On the other hand, the front-end development uses HTML primarily for providing the structure and defining the content layout of the webpages, whereas CSS and JavaScript were used to create a responsive web interface (GoldenOwl, 2025). CSS was primarily used to design the web interface by defining the styling for the web page. Next, JavaScript was utilized to handle validation feedback and dynamic interaction. The system communicated with LHDN's MyInvois system through JSON, a lightweight data format that offers structured data exchange between the web-based system and LHDN's API. Moreover, JSON was used in the invoice submission, validation response, and error handling in order to smoothen the overall data flow process.

In order to ensure an effective and safe file transfer, FileZilla was used as the FTP client for the transference of system files between the local development environment and the Direct Admin host server. With this, easy updates and secure file management are accessible without the necessity of accessing the server directly. Besides that, OAuth2 Authentication is also used in the user login process in order to generate the JWT token for authentication purposes.

1.5 Proposed Approach

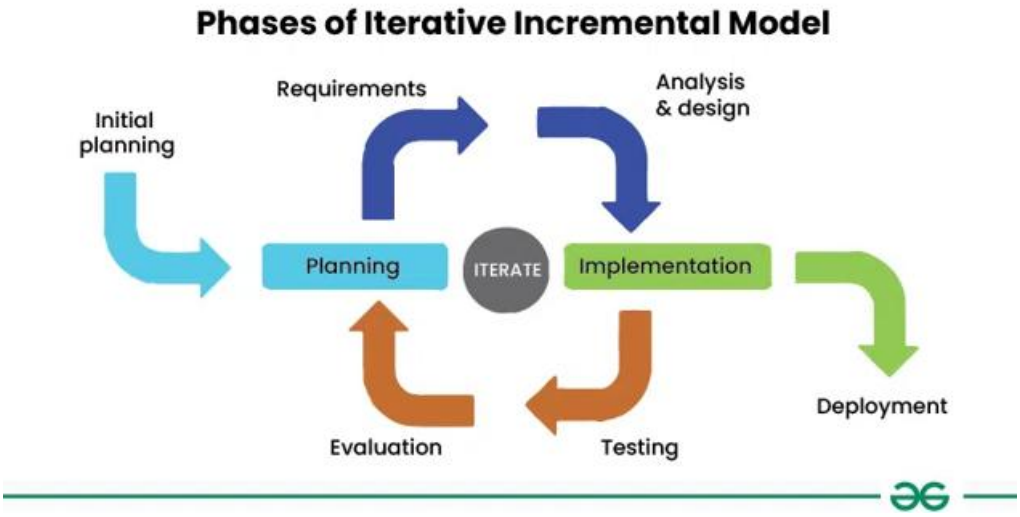


Figure 1.3 Phases of the Iterative Incremental Model (GeeksForGeeks, 2024)

The e-invoicing system adopted the Iterative and Incremental Development (IID) methodology as its foundational approach. The IID methodology in particular places strong emphasis on adaptation and continuous updates, which is suitable considering the regulatory aspects of the system, which is constantly needing to comply with LHDN. This also implies that the IID is flexible and enables regular improvements against changing tax laws, meaning IID can make incremental improvements through periodic updates without incurring any disruption.

Apart from that, the IID methodology has been characterized by decomposition of the system into incremental, manageable components that are incrementally developed through iteration by progressively refining these components, an approach best suited to fitting ever-changing regulatory needs of the environment. When initial requirements analysis discloses flaws in underlying assumptions, these flexible implementation methodologies allow teams to adjust their priorities and make improvements that eventually improve the system's effectiveness (Fagarasan, Popa, Pisla and Crtistea, 2025). This allows easier management and better effectiveness of each individual component.

Next, the iterative nature of IID also allows problem detection early on in the earlier iterations, and encourages revision on a timely basis, given changes in LHDN guidelines. Continuous stakeholder feedback from compliance officers and LHDN liaisons, among others, is built into each iteration, thereby ensuring legal and technical consistency. Hence, risk is greatly reduced through delivering working increments that are verified and validated at each SDLC phase, ensuring ongoing regulatory compliance and system integrity.

1.5.1 Integration Approach

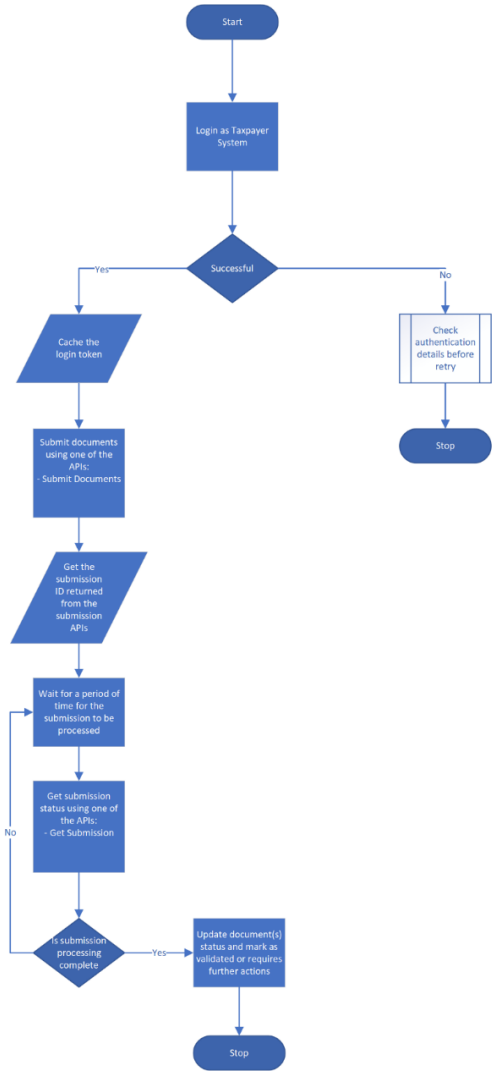


Figure 1.4 Integration Practices: Polling Approach (Lembaga Hasil Dalam Negeri Malaysia, 2025)

For this project, the integration approach used is the polling approach (LHDN, 2025). The polling approach is applied in iterations in order to ensure seamless communication of the e-invoice system with external regulatory interfaces. This approach is particularly used to communicate with the external API such as LHDN's API endpoints.

One of the key advantages for using polling is its support for asynchronous operations. When businesses submit documents such as invoices,

the system performs validation, processing, and verification processes, which can be time-consuming. Thus, the polling approach enables such tasks to occur behind the scenes, without tying up system resources and introducing bottlenecks. Hence, businesses are able to continuously check the status of their submissions periodically rather than waiting for real-time feedback, thereby reducing the chances of timeouts or delays.

Besides that, polling also facilitates the scalability of the system as it allows the processing of several requests simultaneously. Such concurrent processing decreases the server load, which indicates that the system is scalable to accept any increase in transactional levels or operations without a decline in performance.

Another advantage of polling is its ability to reduce errors and increase reliability. By decoupling the submission and validation processes, polling reduces the chances of failure due to network disruption or processing delays. For instance, in the event of an unsuccessful status check, businesses can retry getting the results again by just providing the submission id without having to resubmit invoices. In contrast to a synchronous system, such issues would result in the complete failure of the entire transaction, and the business would have to resubmit the invoice. The system is therefore ensured to remain reliable even if faced with temporary issues.

Last but not least, another reason why polling is chosen is because it aligns with the principles of system design, particularly for high-volume transactions. It also guarantees outcomes are realized only after intensive processing and verification, thereby ensuring data integrity and security. By using the polling approach, the e-invoicing system is able to achieve a balance between efficiency, scalability, and reliability, making it highly capable in meeting the demands of a high-transaction environment.

1.6 Scope of the Project

1.6.1 System Scope

This project's scope includes the development of an e-invoicing system that aims to help businesses and enterprises to submit, validate, and manage invoices. The e-invoicing system have provided a secure and user-friendly platform for businesses to generate invoices, validate them against LHDN's regulatory requirements, and track their status efficiently. The key features are secure user authentication, invoice generation with digital signatures, asynchronous invoice validation, QR code generation for approved invoices, and easy integration with the user interface for real-time updates. Besides that, the e-invoicing system will also have security measures, such as OAuth2 authentication, to maintain data integrity and block unauthorized access. Furthermore, the system will support integration with existing business systems, such as the ERP or accounting software, to streamline invoicing processes.

1.6.2 Target User

The target users or the end users of the e-Invoicing system include businesses of all sizes, tax authorities, and individual taxpayers. Businesses will use the system for generating, submitting, and tracking invoices in order to be compliant with the regulatory requirements. On the other hand, the tax authorities will benefit from the system's ability to validate invoices in real-time and create QR codes for easy verification of invoices. Individual taxpayers, such as small business owners, will also use the system to automate their invoicing processes.

1.6.3 Application Features

1.6.3.1 Login As Taxpayer

This functionality allows taxpayers to securely log in to the e-Invoicing system using OAuth2 authentication. It generates a JWT access token for secure and

authorized use of the system. Furthermore, the LoginAsTaxpayer API facilitates this process, which can be used for authenticating a user. The login process involves verifying and authenticating user credentials with a secure database and generating a token that is used for subsequent API calls. This will allow only authenticated users to access sensitive invoicing information and take actions like submitting invoices and checking the status of their invoice.

1.6.3.2 Invoice Generation

This functionality allows businesses to create invoices in the stipulated format, which includes security mechanisms such as digital signatures for authenticity and compliance. Moreover, the Submit Documents API allows users to submit the created invoices to the system for validation and processing. Invoice creation involves validating the invoice information against predefined business rules and regulatory requirements prior to submission. This ensures that all submitted invoices are compliant and ready for further processing. The digital signature guarantees that the invoice has not been altered and is authentic. With this, the digital signature ensures the validity of e-invoice and is able to proceed to the next step.

1.6.3.3 Invoice Submission

This functionality allows businesses to electronically submit invoices through a standardized API called Submit Documents that functions in checking and validating document formatting before approval. Furthermore, this feature supports both individual submissions and batch processing of numerous invoices. All submissions are digitally signed and timestamped to ensure non-repudiation and data integrity.

1.6.3.4 Invoice Validation

This functionality enables businesses to verify the status of submitted invoices using the GetSubmission API. With this functionality, it provides updates on the validation process, including whether the invoice has been successfully validated or if there is more action to be taken. The invoices are validated

asynchronously to enable easy processing of large quantities. The validation verifies the invoice data against the tax regulations, checks the digital signature, and if all the mandatory fields have been filled in correctly. The status of the validation process is also updated in real time, and companies can monitor the status of their invoices.

1.6.3.4 QR Code Generation

The QR code functionality serves to validate the existence and status of the e-invoice using the MyInvois Portal. Upon scanning, the buyer would be able to view the validated e-invoice details as well as, verify its authenticity via the generated QR code. This feature will be embedded within the e-invoice itself, as a way for the buyer to track their e-invoices details in a convenient and seamless manner.

1.6.3.5 Cancel Documents

This feature enables the supplier, who is the issuer of the invoice to cancel previously issued documents or invoice, as a means to address submission problems that were discovered immediately. This cancellation can be achieved either by self-induced cancellation or by accepting a buyer rejection request. The potential reasons that may prompt the invoice cancellation could be due to issues related to the document or the supplier accepts the buyer's rejection request.

1.6.3.6 Notification

This feature enables the ERP system to search for previously sent notifications. The notifications are issued through email and then saved in the notification history. This procedure uses the Get Notification API to query the notification history in order to retrieve the notifications.

1.6.3.7 Ledger

This feature provides businesses with detailed reporting on their financial activities. Through this feature, it serves to help businesses to keep track and

closely monitor their financial health. Hence, this feature provides a centralized, organized and comprehensive view to manage all business transactions.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The digital age presents a wide range of digitalization activities to be prevalent in many business activities. Without a doubt, the taxation sector has also been taking initiatives in digitalization the taxation process. One of their initiatives is through e-invoicing. An e-invoice can be defined as a digital record of a transaction between a supplier and a purchaser (LHDN, 2025). In terms of the information that an e-invoice carries, an e-invoice has the same critical information as a traditional document, such as supplier and buyer identification, item description, quantity, price excluding tax, tax, and total amount, and it also captures transaction data for everyday business activities. Besides that, e-invoices also serve as a substitute for physical or electronic documents like invoices, credit notes, and debit notes.

2.2 History and Evolution of E-invoicing

The origins of e-invoicing can be traced back to the creation of EDI, which is an abbreviation for Electronic Data Interchange. EDI is a set of technical standards for e-documents, and has been used to facilitate corporate transactions for decades. In 1965, the first electronic document was communicated between companies using electronic data interchange (EDI) due to a shipment manifest (Agile Dynamics Solutions, 2024). This implementation had laid the benchmark for what would be a revolution in business communication.

Fast forward to 10 years later, File Transfer Protocol was introduced, which allows businesses to transfer various types of files over the internet. Hence, increasing the speed and the reliability of electronic transactions over the internet. During the 1980s, EDI began gaining traction in numerous retail and car manufacturing companies, and this led to many suppliers began the adoption of EDI. The 1990s also experienced further development and utilization of EDI as large corporations began adopting it. As more

organisations gained internet access in the 1990s, EDI systems were increasingly superseded by web-based alternatives. The emphasis then shifted away from just sending papers and towards digital transaction processing, which paved the path to the arrival of modern e-invoicing.

The early 2000s is when the term “Electronic Invoicing” became more popularized due to the adoption and development of dedicated e-invoicing solutions by companies. The aim to digitize the entire invoicing process has led to the globalization and involvement of governments from different countries, as governments throughout the world saw the benefits of e-invoicing and began encouraging businesses to implement it. For instance, the European Commission was one of the first to integrate e-invoicing into its legal frameworks after recognizing the immense cost savings through e-invoicing. With this, many other countries, such as Mexico and Brazil, followed suit by mandating e-invoicing in all of the business transactions (Darash, 2022).

As technology began evolving rampantly in the 2010s, the usage of e-invoicing has also undoubtedly increased significantly, owing mostly to the Covid-19 worldwide pandemic, which had pushed digitalization across a wide range of businesses. In consequence, cloud-based solutions began gaining popularity as it has the capabilities of enabling real-time invoicing, worldwide accessibility, as well as, increasing collaboration between industries.

2.3 E-Invoice model in Malaysia

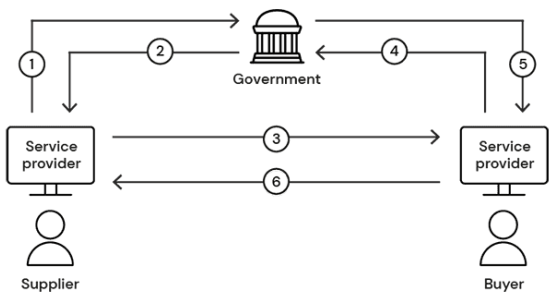


Figure 2.1 Clearance Model (Pagero, 2025)

As time goes on, the e-invoicing initiative has begun to be adopted by businesses in Malaysia. With this implementation spreading rampantly, Malaysia has adopted the Continuous Transaction Control (CTC) Model, which enables a high level of control through the validation of e-invoices received by LHDN. Specifically, Malaysia adopts the Clearance Model, which is a type of the CTC model (Edicom, 2024). Under this model, businesses to validate and approve invoices in real time before they can be lawfully sent to purchasers (Cleartax, 2024). Businesses must first submit their invoice data to LHDN's MyInvois system, where it is verified for regulatory compliance, including appropriate formatting, tax computations, and digital signature validation. Once validated, the system then provides the invoice with a unique number and a QR code, indicating that it has been formally approved for usage (Cleartax, 2024). Therefore, this approach enhances tax compliance, reduces the risk of fraud, and provides greater transparency in business transactions. By adopting the CTC clearance model, LHDN ensures that all invoices are processed through a centralized system, which then allows tax authorities to monitor transactions in real-time and maintain a secure digital tax infrastructure.

2.4 Importance and Advantages of E-invoicing

The primary benefits of e-invoicing to businesses is that it is able to minimize manual effort and human errors (LHDN, 2025). This can be achieved through a unified invoicing procedure of e-invoicing that allows for electronic input of transaction documents and data. Traditional invoicing requires manual data entry, which may increase the likelihood of mistakes such as incorrect data entry, duplicate data, or even missing details. With e-invoicing, it is able to streamline the entire billing process as the invoices are generated in one standardized format, which then facilitates data accuracy and consistency. Besides that, automated validation included as part of the e-invoice feature also prevents incorrect data from being sent, thereby reducing the risks of errors.

Next, e-invoicing also improves tax filing efficiency. This is said so as the traditional approach to tax filing may take days, if not weeks, due to

multiple reasons such as human processing, delays in mail, or even, slow internal approval. With e-invoicing, the aforementioned bottlenecks can be eliminated by enabling invoices to be transmitted electronically. For instance, this can be achieved by the integration of API into the e-invoicing system, which allows fast submissions and automates data exchange between businesses and tax authorities. For instance, LHDN provide a multitude of seamless and easy-to-use APIs connection points, such as the SubmitDocuments API, which functions to submit the document electronically before an invoice is issued. Therefore, this ensures high accuracy in the tax filing process.

Apart from that, the integration of e-invoices into the system is able to streamline operational efficiency, leading to considerable time and cost savings. The implementation of e-invoicing is able to reduce the expenses used in printing and paper expenses, postage fees, storage fees and labour costs for manual handling of data entry. Hence, this results in considerable cost savings. This can be further validated by experts who foresees that the switch to digital invoicing can save up to 80% on invoicing costs (Chan and Alias, 2023).

Last but not least, digitalization through the implementation of e-invoices and comprehensive financial reporting can ensure that the tax filing procedures adhere to the industry standards. In conventional reporting, businesses frequently face delays in invoice processing, data reconciliation, and periodic financial updates. Thus, the digitization of financial reporting through e-invoicing allows for real-time access to financial activities.

2.5 Current Similar Applications of E-invoicing

2.5.1 Financio

Figure 2.2 Logo of Financio

Financio is a cloud-based accounting software which is specifically designed to cater to small and medium-sized enterprises (SMEs) in Malaysia. Furthermore, Financio have been a pioneering leader in Asian business solutions for over 20 years, by utilizing their extensive financial knowledge and advanced technology in their services. Their core offerings include Financio Accounting and Financio Payroll, which seamlessly work together as a comprehensive and workable financial management platform. Through their services, Financio aims to streamline its financial operations, ensure compliance, and boost overall business efficiency (Financio, 2025). To achieve this, Financio aims to automate key financial workflows to save up crucial time needed for strategic growth initiatives in order to boost efficiency. Besides that, they also aim to provide real-time insights through their services by allowing access to comprehensive reports, making it clear for customers to understand the state of their financial health. Not only that, they also have a visionary approach in ensuring compliance by providing businesses with the necessary tools in order to maintain adherence while reducing operational risks.

As a trusted Small Business Enterprise (SME) Accounting solution provider that has considerable experience implementing e-invoicing in Singapore, Financio has begun extending its product coverage to the Malaysian market. Hence, this allows Financio to provide solutions that perfectly match the requirements of the e-invoice implementation in Malaysia due to their extensive experience and knowledge in delivering e-invoicing solutions.

The e-invoicing solution developed by Financio is one of the most sought-after solutions. In terms of the features, the solution has the feature of integrating with other accounting software and ERP systems, allowing businesses to link their invoicing system with a variety of financial and operational modules. With this seamless integration taking place, this allows all financial activities, including invoicing, to be synchronised amongst departments, thereby reducing data duplication and increasing operational efficiency.

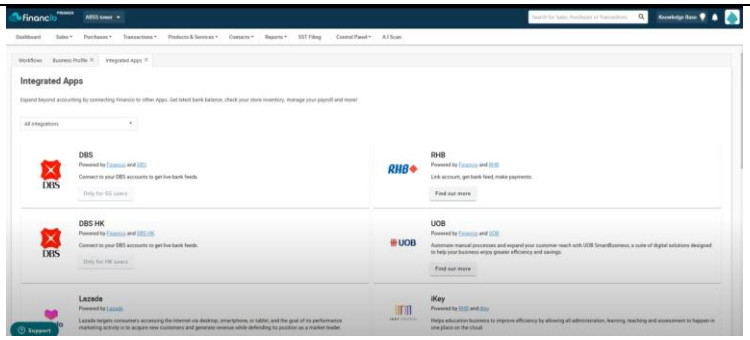
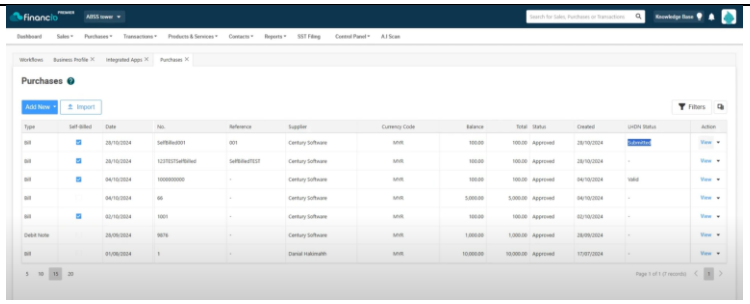
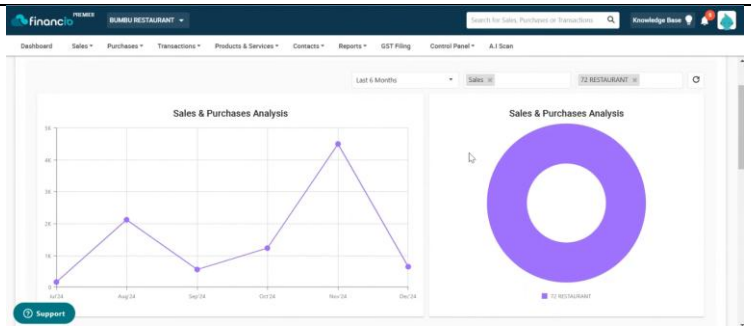
Furthermore, one of the most notable characteristics of Financio's e-invoicing system is it allows real-time submission to necessary authorities and customers. Once an invoice is created, it is immediately registered in the system, allowing businesses to monitor its status in real time. This allows organisations to better track whether an invoice has been submitted, validated, rejected or cancelled. Therefore, this feature promotes transparency and visibility in business operations and allows businesses to take rapid action in the event of any anomalies or rejections, thereby reducing delays in any financial operations and lowering the chance of any compliance issues.

Next, real-time tracking guarantees that firms may discover any delays in invoice processing and take appropriate steps to avoid cash flow problems. Not only that, the e-invoicing solution also provides the flexibility to make cancellations on the e-invoice within 72 hours. If an invoice has mistakes or inconsistencies, Financio allows firms to cancel and reprint e-invoices to ensure government compliance. Hence, this improves financial reporting accuracy and lowers the risk of errors.

Last but not least, another distinguishing feature of Financio's e-invoicing system is its integrated reporting and dashboard capability, which gives businesses a clear and structured perspective of their invoicing operations. For instance, the system will organizes financial data into a user-friendly dashboard that provides comprehensive information and actionable insights about trends over time and comparisons between categories in a graphical format. Through this data-driven strategy, this functionality enables businesses to make more informed financial decisions by spotting patterns and possible bottlenecks in their invoicing processes.

Table 2.1 Feature and Interface of Financio

Feature	Interface

Integration with ERP and Accounting Software	 Figure 2.3 Feature of ERP and Accounting Software Integration
Tracking of Invoices	 Figure 2.4 Feature of Tracking of E-invoices
Seamless Reporting	 Figure 2.5 Feature of Reporting through Dashboard

2.5.2 Bukku

Figure 2.6 Logo of Bukku

Bukku is a cloud accounting software that originated from Malaysia in the early 2020. The platform itself was started by a serial entrepreneur with a deep fascination with technology (Bukku, 2024). Bukku was founded to embark on

a mission to change the way small and medium-sized enterprises (SMEs) perform accounting tasks. Noting that more than a million micro and small businesses in Malaysia were still doing traditional accounting, such as manual bookkeeping and spreadsheets, Bukku was founded to assist companies in switching to a fully digitalized accounting system. As automation and artificial intelligence are introduced, Bukku offers a user-friendly platform that makes accounting tasks simpler while keeping up with the ever-changing tax laws.

The primary aim of Bukku is to empower accountants and business owners with its efficient, precise, and cost-effective cloud accounting platform. Bukku aims to eliminate inefficiency, manual labour, and non-compliance and achieve greater financial visibility for small, medium, or large enterprises. Bukku enables individuals to maintain accurate financial records through automated financial reporting, intelligent invoicing, and live tracking so they can remain compliant with tax regulations.

Within the Malaysian government's e-invoicing compliance mandate under the MyInvois regime, Bukku has led the way in the industry in enabling seamless adoption of e-invoicing. The platform aims to simplify e-invoice submission, tracking, and verification while facilitating compliance with the law. In contrast to conventional invoicing platforms, Bukku offers a range of innovative e-invoicing functionalities specific to the requirements of Malaysian businesses.

One of the best features of Bukku is the E-Invoice Rejection feature. The recipient does this by scanning the QR code available in the invoice, and subsequently, the recipient may request a document rejection. This enables companies to correct mistakes very promptly, thereby reducing disputes and bringing transparency to financial transactions. In addition, Bukku permits companies to reject e-invoices up to 72 hours after submission to MyInvois so that any errors may be rectified before becoming permanent records. Such control over invoice processing makes Bukku an extremely flexible e-invoicing solution.

Another key benefit of Bukku is that it supports real-time status notifications for submission. In contrast to standard invoicing software, where companies have to check the status of invoices manually, Bukku offers automated status updates on the state of every submission. The users receive real-time feedback on whether an invoice has been successfully processed, rejected, or needs some other action. This aspect greatly improves workflow efficiency since companies is able to act on any problems instantaneously.

Apart from submission tracking, Bukku also provides comprehensive reporting and a dashboard that gives businesses complete financial visibility. The system has a user-friendly interface with over 10 interactive charts, which allow businesses to track sales performance, expenses, and cash flow trends in real time. With over 50 financial reports, businesses can perform in-depth financial analysis, enhancing decision-making and long-term financial viability. Unlike manual accounting systems that take data input and report generation, Bukku automates the process, saving time and minimizing errors.

In addition, Bukku offers a unique MyInvois Readiness Verification function that allows businesses to determine if their contacts are MyInvois compliant. As the Malaysian tax authority moves towards complete integration of e-invoicing, it becomes obligatory for businesses to have their buyers and sellers duly registered under the MyInvois network. Bukku makes it easy by offering automated verification tools that minimize administrative hassle and ensure seamless exchanges of invoices between businesses.

Table 2.2 Feature and Interface of Bukku

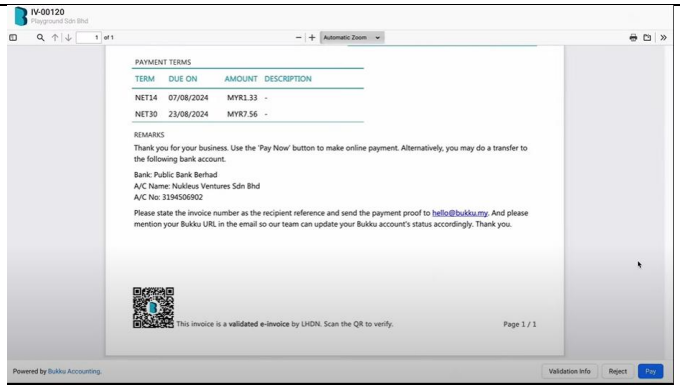
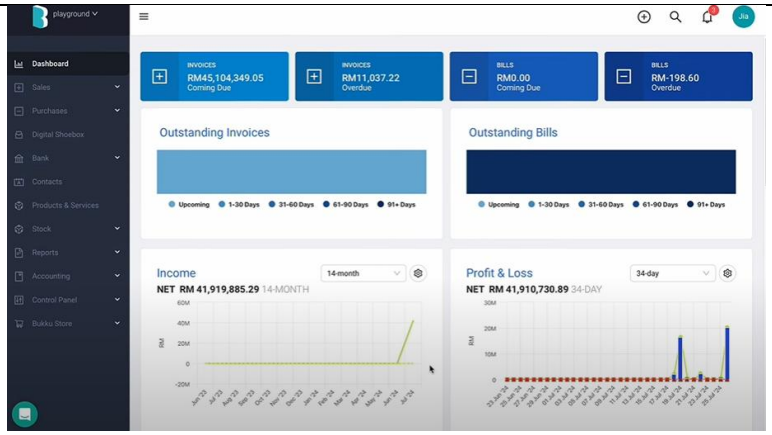
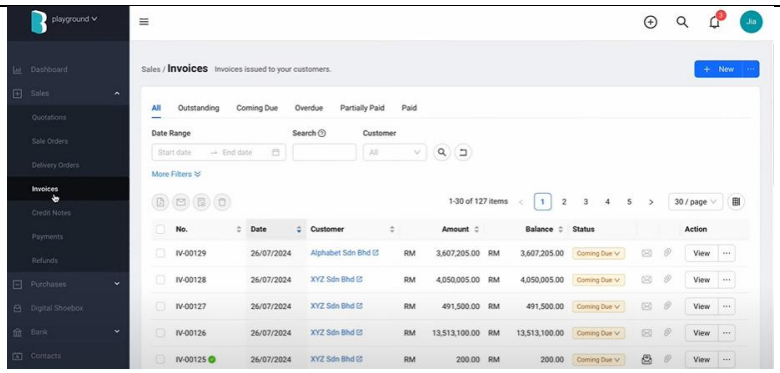
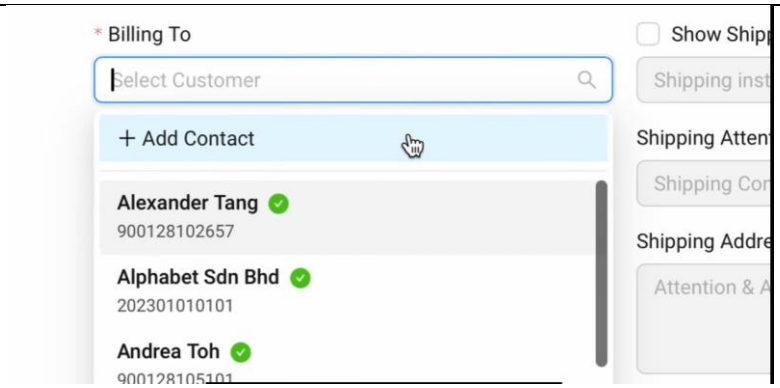
Feature	Interface
Rejection of e-invoice	
Notification	
Seamless Reporting	

Figure 2.7 Feature of Rejection of E-Invoice

Figure 2.8 Notification Feature

Figure 2.9 Feature of Reporting Dashboard

Tracking of e-invoices	 Figure 2.10 Feature of Tracking of E-invoices
Verification for MyInvois Readiness	 Figure 2.11 Feature of Verification for MyInvois Readiness

2.5.3 JomeInvoice

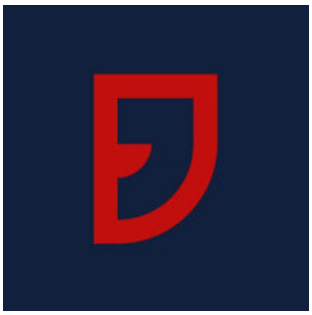


Figure 2.12 Logo of JomeInvoice

JomeInvoice is a comprehensive e-invoicing system developed to meet the needs of businesses of all kinds, from small and medium-sized enterprises (SMEs) to major multinational businesses. JomeInvoice particularly stands out because of its smooth ERP integration, emphasis on security and strict compliance with Malaysian regulatory bodies. Besides that, businesses may use JomeInvoice as it includes a comprehensive middleware platform that is simple to combine or integrate with the current company systems, as well as

an easy-to-use, yet highly customizable solution for e-invoicing (JomeInvoice, 2025).

The major goal of JomeInvoice is to streamline the invoicing process while simultaneously attaining high security and compliance standards. Due to the kickstart of the implementation of e-invoicing in Malaysia, businesses now must have a compliant e-invoicing system in place. Therefore, JomeInvoice guarantees that firms meet regulatory obligations while maintaining a simplified workflow. With real-time invoice monitoring, security compliance, and extensive ERP integrations with other systems, the software is one of the most integrated in Malaysia's e-invoicing industry.

First of all, one of JomeInvoice's primary features is its seamless integration with leading ERP systems. Unlike typical e-invoicing software, JomeInvoice integrates with SAP ECC6, S4 Hana, Sage300, Microsoft 365, Netsuite, Odoo, Xero, and other corporate applications. With the large-scale interoperability, businesses can just integrate JomeInvoice solutions into their existing system, thereby reducing disruptions and also, eliminating the need to shift to another platform just for the sake of using performing e-invoicing operations. Therefore, this flexibility enables businesses to continue utilizing their preferred ERP systems while being compliant with Malaysia's e-invoicing regulations.

Besides that, JomeInvoice offers extensive reporting and analytics. The system contains a dashboard that gives real-time feedback on invoicing activities, which enables businesses to track invoice status, cash flow, and payment patterns. Through the detailed analytics, it allows businesses to make data-driven decisions, analyze revenue performance, and automate their invoicing process.

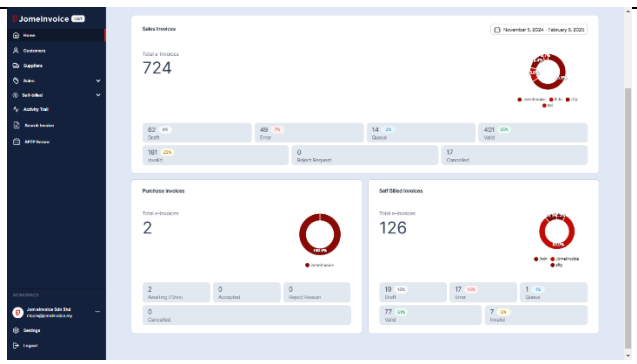
Next, JomeInvoice also prioritizes security. The firm is PDPA compliant and holds ISO certifications (9001, 20001, and 27000), proving its strong emphasis on data privacy and cybersecurity (Jomeinvoice, 2025). These certifications show that JomeInvoice follows worldwide best practices for

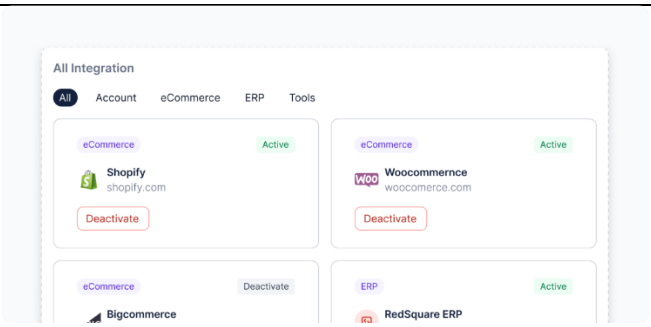
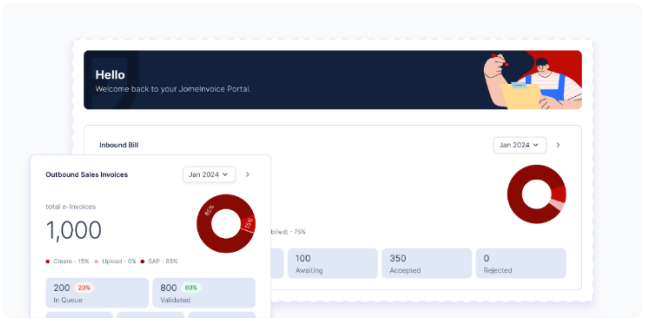
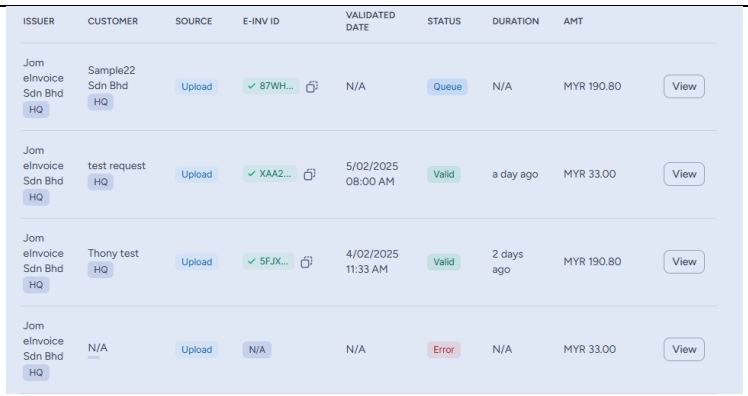
information security management, IT service management, and quality assurance. The platform safeguards financial data and businesses against fraudulent activities, illegal access, and cyber assaults by utilizing strong encryption techniques and secure authentication mechanisms.

Another unique feature of JomeInvoice is real-time invoice status tracking, which allows businesses to monitor each stage of the invoicing process. Businesses may track the progress of their invoices from submission to verification, rejection, or approval in real time. This functionality is especially useful for businesses dealing with a high volume of transactions since it reduces processing errors and improves overall efficiency. With this, JomeInvoice accelerates decision-making process and eliminates any financial barriers by enabling businesses to be instantly alerted of the status of their bills.

Last but not least, JomeInvoice is also known for its user-friendly and adaptable e-invoicing system, which was created utilizing a no-code method. Unlike certain tools, which demand a high level of technical understanding, JomeInvoice streamlines the process of integrating and deploying the system without requiring programming experience. This makes it a great alternative for SMEs who do not have in-house IT staff but need an effective, compliant e-invoicing system. The website's user-friendly layout and simple design ensure that users will find it easy to adopt and use its features.

Table 2.3 Feature and Interface of JomeInvoice

Feature	Interface
User-Friendly Interface	 Figure 2.13 JomeInvoice's Interface

ERP and other Software Integration	 Figure 2.14 Feature of ERP and other Software Integration
Seamless Reporting	 Figure 2.15 Feature of Reporting Dashboard
Tracking of e-invoices	 Figure 2.16 Feature of Tracking of E-Invoices

2.5.4 Autocount

autocount®
Beyond accounting software

Figure 2.17 Logo of Autocount

AutoCount is a well-known Malaysian business software company that was founded in 1996. Initially, Autocount was recognized as BCE Software Sdn Bhd, but was later then rebranded to AutoCount Sdn Bhd in 2008 (Autocount, 2024). Over the years, it has evolved into a pioneering software development company in Malaysia, with a main focus in accounting and business applications. Besides being a leader in software development in Malaysia, it is also a solution provider behind a top accounting, point-of-sale (POS), and human resource management system (HRMS), which facilitates a one-stop financial management platform for businesses. In 2024, Autocount launched its e-invoicing platform, which automates the process of processing invoices, and at the same time, ensures compliance with regulatory requirements.

The primary goal of AutoCount's e-invoicing system is to provide businesses with an integrated system that automates invoicing without sacrificing accuracy and compliance. By allowing companies to track invoices in coordination with accounting, payroll, inventory, and sales information, AutoCount avoids duplicate data entry and minimizes errors. This integration enables accurate financial transactions in all modules, thereby allowing businesses to achieve real-time financial visibility and transparency in order to streamline their day-to-day operations.

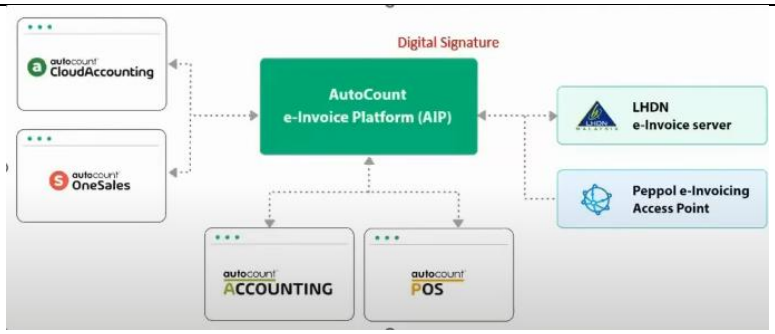
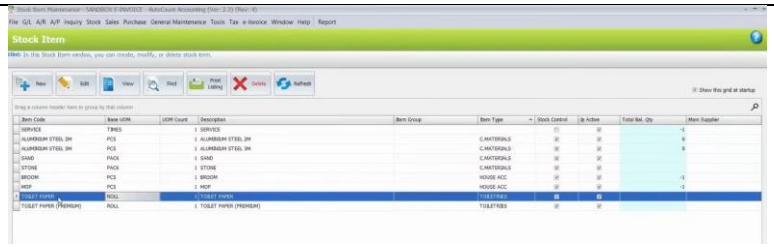
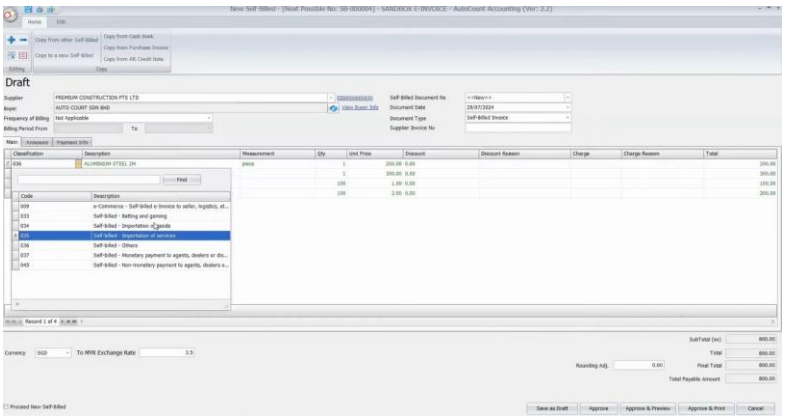
One of AutoCount's best features is its e-invoicing solution, which allows total integration with the other AutoCount modules, such as accounting, POS, and HRMS. Unlike standalone e-invoicing software, AutoCount allows companies to manage their financial, sales, and human resources activities into a single system. The unified process gives maximum workflow efficiency since invoices can be easily linked automatically with payroll, sales transactions, and accounting records. For example, an AutoCount POS-using retail business can send e-invoices directly from the POS system to guarantee that transactions are well-documented without any manual intervention needed.

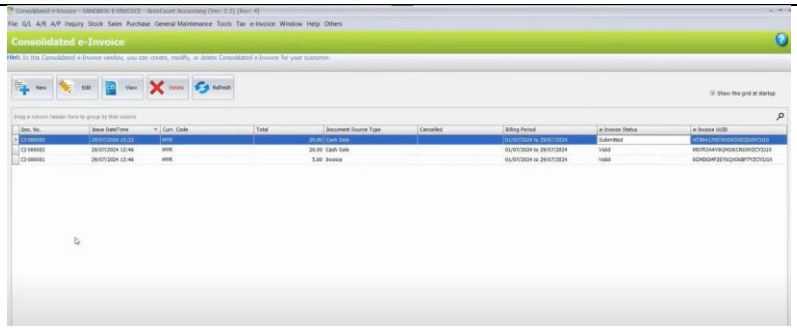
Apart from that, another significant feature of AutoCount's e-invoicing solution is the inclusion of its inventory management system (IMS) with stock entry facilities. With this feature, businesses that have large-scale inventories can link their e-invoicing system with their stock levels such that creation of invoices takes into consideration the actual availability of products at real time. This is particularly valuable for businesses involved in retail, wholesale, and distribution, where real-time stock inventory needs to be tracked. Therefore, this integration allows for automatic and instant updates of stock records, thereby reducing stock mismatch between inventory data and invoicing data.

Besides that, AutoCount has data entry automation whereby companies can easily search for item codes when issuing invoices. Instead of manually typing in product details, users can access information instantaneously from a pre-defined database of items in hand. With this, there is no possibility of pricing errors and wrong entries. For instance, this aspect is specifically useful for companies with high volumes of invoices to process on a daily basis because it quickens the invoicing process and improves accuracy.

Last but not least, real-time tracking of e-invoices is also a significant feature of AutoCount's system. Businesses can monitor the status of their invoices, ranging from submission to verification and authorization, to make sure that all transactions are being processed in an efficient and transparent manner. With an interface which presents the whole invoices issued, businesses can readily identify pending, rejected, or successfully processed invoices, which ultimately helps them to take any immediate corrective action if necessary. Therefore, this transparency on the invoice status is able to assist businesses in preventing delayed payments and compliance issues.

Table 2.4 Feature and Interface of Autocount

Feature	Interface
ERP and Accounting Software Integration	 Figure 2.18 Feature of ERP and Accounting Software Integration
Inventory management system with stock entry facilities	 Figure 2.19 Feature of Inventory Management System with an Overview of the Stock Entries
Automation of Data Entry	 Figure 2.20 Feature of Automation of Data Entry

Tracking of e-invoices	 Figure 2.21 Feature of Tracking of e-invoices

2.6 Comparative Analysis of Existing Applications

2.6.1 Financio

2.6.1.1 Advantages of Financio

Financio has been widely recognized for having a user-friendly interface that is leading towards small businesses with limited technical skills (Outsized, 2024). Their interface design is particularly advantageous in enhancing usability as it allows small-scale businesses to navigate easily, ultimately eliminating the need for interface training. This particular aspect is of great significance as micro, small and medium enterprises occupy more than 97.4% or more than 1.2 million of the entire business environment (SME Corp, 2024). The user-friendliness of the application provides Financio with a great competitive advantage against other competitors.

Next, Financio also offers real-time invoice submission ensures compliance with regulations. This is particularly useful in ensuring tax compliance in a consistent manner, while simultaneously minimizing any invoice risks such as invoice tampering, fraud or late reporting of invoices. Subsequently, this benefits both the government and businesses as it gives governments real-time or near-real-time access to business transaction data as well as improves transparency in the tax reporting (RTC Suite, 2024).

Besides that, Financio also allows extensive integration with Enterprise Resource Planning (ERP) software and accounting software. This allows streamlined workflows and synchronization of financial data to take place as it eliminates the need for manual data entry. Furthermore, the

integration would then be able to provide a centralized, administrative hub that operates all in one location, thereby significantly improving business efficiency.

2.6.1.2 Disadvantages of Financio

Although Financio has many advantages that outweigh its disadvantages, Financio has one major aspect, with it being the absence of a ledger. The ledger is particularly important as it functions not only as a record-keeping tool, but also as a basis for financial planning, reporting, and auditing (Tantawy, 2024). Without the presence of a ledger, businesses must rely on manual tracking, which could potentially lead to data fragmentation or even inaccuracies or inconsistencies in financial reporting. Hence, the absence of a ledger is one of the setbacks of Financio's application.

2.6.2 Bukku

2.6.2.1 Advantages of Bukku

Bukku has several advantages when it comes to its application, one of it being the ability to cancel submitted invoices. This feature is particularly useful and provides users with some flexibility in making changes to their submitted document.

Next, Bukku also offers real-time invoice submission, which allows high transparency in its invoice submission process. This allows both businesses and LHDN to monitor for any discrepancies in the tax reporting process. Therefore, this particular feature allows for better tax compliance in the overall tax filing process.

Apart from that, Bukku also has great automation features which help reduce human errors. This particular feature is intended to reduce manual data entry while simultaneously increasing accuracy and efficiency in the tax reporting process.

In addition, Bukku also supports system notification features. This feature is particularly useful in notifying taxpayers of their e-invoicing process.

For instance, the notification feature serves to bring attention to the invoice processing status as well as any modifications made in the profile. Therefore, this guides taxpayers in navigating through their entire e-invoicing process, thereby providing clarity and increasing the usability of the system.

2.6.2.2 Disadvantages of Bukku

Despite the advantages presented by Bukku, there are also a few limitations that are present in Bukku's e-invoicing application. One of it being the absence of any ERP integration. This aspect is a major setback for large growing businesses as it does not have a centralized hub, thereby limiting their operational scalability. Hence, this makes the application less comprehensive for larger businesses due to its simplicity.

Lastly, Bukku also does not have a built-in ledger functionality. This is a setback for Bukku as the application itself targets in selling their solution to small, medium enterprises. Without the presence of a ledger, this then heightens the administrative burden but also the risk of inconsistency in data and delayed accounting information.

2.6.3 JomeInvoice

2.6.3.1 Advantages of JomeInvoice

The key advantages of JomeInvoice's application is the e-invoice cancellation feature, which gives taxpayers the flexibility to modify and correct invoicing errors. This particular feature is extremely important in ensuring compliance, as compliance in tax filing is handled with utmost regard. For instance, non-compliance penalties vary from RM200 to RM20,000, or 6 months imprisonment and both penalties for those with severe cases (Cleartax, 2024). Therefore, this feature allows businesses to correct mistakes in the invoice, thereby ensuring compliance and avoiding potential regulatory penalties.

Besides that, JomeInvoice also excels in tracking e-invoices efficiently, such that businesses can maintain accurate audit trails and retrieve historical data for compliance, reporting, or internal audit purposes.

Another fundamental advantage is its automatic inputting of data, which reduces human errors in data input and enhances operational efficiency. It is particularly helpful to businesses with high volumes of transactions, as it can reduce time taken for manual entry as well as, raises productivity.

Moreover, JomeInvoice supports in-depth integration with accounting and ERP systems, such that users can remain flexible while manipulating financial data in various systems.

2.6.3.2 Disadvantages of JomeInvoice

Although JomeInvoice is great in many aspects, it is still lacking in terms of having a ledger. A ledger is significant in the consolidation of financial activities as it takes into account non-invoice transactions and invoice transactions into one single record. Hence, the absence of this feature implies that JomeInvoice users are not able to perform full accounting activities and that they are required to export invoice information and integrate it manually into external accounting applications.

2.6.4 Autocount

2.6.4.1 Advantages of Autocount

One of Autocount's primary advantages is its ability in its integrated ERP system. This allows Autocount to offer an extensive experience for their customer base as their solutions allow the integration with other applications. With this, it makes it especially appealing to medium-sized to large-sized businesses as they require a proper and centralized system which allows them to integrate with other finance, inventory, and customer relationship management (CRM) systems.

Next, Autocount also supports document cancellation. This feature is a valuable feature as it is especially required in cases of errors or modifications in the documents. Hence, this allows businesses to keep up-to-date with their records and at the same time, ensure tax compliance within their business practice.

Autocount is also known to provide a great mechanism in providing real-time updates for invoices and inventory. Their real-time updates includes real-time reconcillations in the levels of inventory once an e-invoice is submitted. Therefore, this ensures businesses always keep up-to-date records in various aspects of operations.

2.6.4.2 Disadvantages of Autocount

Despite the advantages of Autocount, there are a number of weaknesses in Autocount that hinder the usability of their application. Most critically, their lack of intuitive and user-friendliness in the interface is one of the setbacks of Autocount. Some of Autocount's features are deemed too difficult for non-technical users to understand. For instance, there are instances where users are required to navigate through multiple menus in order to perform a basic task. Due to this, this specific aspect makes it even more difficult for non-technical users to understand and navigate through the application.

Similar to Financio, Bukku and JomeInvoice, Autocount also lacks in terms of having a ledger. Although Autocount is advanced in terms of its accounting capabilities, certain businesses with unique requirements may find the lack of a standard and dedicated ledger restrictive.

Moreover, Autocount lacks in its notification feature. Although Autocount provides users with error messages as a response to alert users of the action's status, it is still lacking in producing a notification feature which is capable of providing users with real-time updates about important e-invoicing activities.

Table 2.5 Comparison between Similar Applications and the Proposed Project

	Financio	Bukku	JomeInvoice	Autocount	Proposed project
User-Friendly	Yes	Yes	Yes	No	Yes

Interface					
ERP Integration	Yes	No	Yes	Yes	Yes
Generate Compliant Invoices	Yes	Yes	Yes	Yes	Yes
Real-Time Invoice Submission	Yes	Yes	Yes	Yes	Yes
Automation of data entry	No	Yes	Yes	Yes	Yes
Digital Validation of E-Invoices	Yes	Yes	Yes	Yes	Yes
QR Code Generation	Yes	Yes	Yes	Yes	Yes
Authorized Digital Certificate & Signature	Yes	Yes	Yes	Yes	Yes
E-Invoice Cancellation	No	Yes	Yes	Yes	Yes
Notifications	Yes	Yes	Yes	No	Yes
Tracking of e-invoices	Yes	Yes	Yes	Yes	Yes
Ledger	No	No	No	No	Yes
Seamless Reporting	Yes	Yes	Yes	No	No

Based on the compiled analysis and findings in Table 2.5, it is evident that there are certain unique key features that can be optimized or added to increase the functionalities of the system. Therefore, the proposed project will

incorporate and integrate the best features available to reduce the disadvantages.

Firstly, the proposed project will be created in an easy-to-use interface. As seen in the example provided by Autocount, their interface is deemed too technical, which may discourage adoption from non-technical users and small businesses. Hence, the proposed project will emphasize intuitiveness and user-friendliness of the interface in order to reduce the technical complexity of the system.

Secondly, the proposed project will also implement a notification feature which aims to alert users of important actions such as status updates, document updates, or even a reminder on the invoice due date.

Besides that, one of the major drawbacks of the current e-invoicing applications is that all 4 of the applications that were previously compared do not have an in-built ledger feature. Hence, the proposed project will be a significant advancement in this aspect, as it will include a ledger in the application.

As for the ERP integration feature, 3 of the applications, which are Financio, JomeInvoice, and Autocount, have this feature, while Bukku does not, making it less appealing to mid-sized to large-sized companies to adopt this application. Hence, the proposed project is built as part of an ERP framework, but for the purposes of this project, it will be solely focusing on the e-invoicing activities. However, this system itself is designed so that it could be connected to other ERP modules in the future.

Other than that, Bukku, JomeInvoice, and Autocount all provide support for the data automation entry feature, whereas Financio does not. This may eventually cause extra manual and redundant work for users. Hence, the proposed project will include automation for data entry, thereby reducing human error and increasing accuracy.

In regard to other common features being implemented in all 4 of the applications, the proposed project will also implement features such as generating compliant invoices, real-time invoice submission, digital validation of e-invoices, authorized digital certificate and signature, e-Invoice cancellation, and tracking of e-invoices, as these common features represent the fundamental features needed to be implemented in an e-invoicing system.

2.7 Summary

The need for e-invoicing is driven by the government's mandate that all businesses implement electronic invoicing solutions. As such, several software companies have developed e-invoicing software, each with its own set of functionalities and features. Following a close examination of available systems, it becomes evident that though the solutions do offer the necessary e-invoicing function, some of the essential features are either missing or understated.

The suggested project contains several innovative features that are less prevalent in other systems, including enhanced notification functionality and the implementation of a ledger. Through this implementation, companies not only gain the potential for automating invoice procedures but also better overall financial control and process efficiency. This approach not only seeks to ensure compliance, but also to simplify and automate the invoicing process, thereby streamlining the invoicing process for businesses of any size.

CHAPTER 3

METHODOLOGY AND WORKPLAN

3.1 Introduction

This section outlines the chosen methodology that was used throughout the project. This section in particular provides a comprehensive overall workflow of the project with the provisioning of the work breakdown structure (WBS) as well as the Gantt Chart. Apart from that, the development tools and technologies will be discussed in this section.

3.2 Software Development Methodology

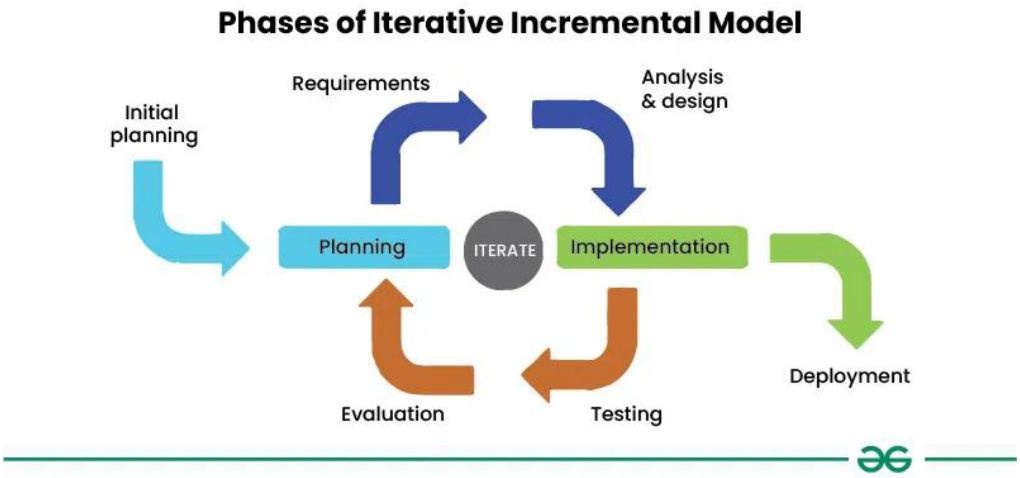


Figure 3.1 Phases of the Iterative Incremental Model (GeeksForGeeks, 2024)

The methodology selected for the e-Invoice system is based on the principles of Agile Development, which uses the Iterative and Incremental Development (IDD) approach. The Iterative and Incremental approach was particularly chosen as it is a subset of the Agile Approach and it places high emphasis on the adaptable nature, continuous improvement, and collaboration with stakeholders. These aspects are particularly important as the development of the system requires strict compliance with evolving laws and guidelines proposed by LHDN.

As illustrated in Figure 3.1, the development process under this methodology is typically carried out in short iterations, which are often referred to as sprints. Each round of sprints will then focus on finishing a specific set of features or tasks inside the system, such as invoice creation, invoice validation, or any new needs in the e-invoicing system. The iterative nature of the technique enables the development team to continually monitor project progress and make necessary adjustments in response to stakeholder feedback or changes in the compliance landscape.

One of the key strengths of the IID approach and the Agile approach in general comes from its ability to include mechanisms for feedback after every iteration. This aspect is of particular significance to the e-Invoice system that needs to regularly comply with the requirements proposed by LHDN. Besides that, this further permitted adaptation to changing requirements while simultaneously ensuring that higher priority functionalities are addressed first, followed by lower priority functionalities.

In addition to that, the iterative and incremental nature of the methodology allows the deployment of system updates with minimal disturbance. Each increment will result in a portion of the system that is completely workable, functioning and tested. With that, as each increment comes to a conclusion, the entire system then gets further refined and customized to better match the particular demands of LHDN and the system's users. Hence, this methodology thus eases the addition of new features or the modification of existing capabilities in accordance with LHDN standards and regulations without the need for a complete system re-design and re-work.

Furthermore, this particular approach provides risk management as a core aspect throughout the design and development process. In each cycle, the potential risks are identified, examined and mitigated at the earliest stages, thereby reducing the likelihood of severe issues in later stages of the project life cycle. This approach is particularly crucial to the development of the e-invoicing system, whereby early identification and correction of risk

determines the success of the project, and non-compliance to the regulations is legally and financially expensive.

Furthermore, the IID approach also enables early delivery of values to stakeholders (GeeksForGeeks, 2024). This approach ensures close interaction with the stakeholders after every iteration in order to ensure that the system not only performs the required functional aspects but also remains in compliance with the ever-changing tax laws. Unlike conventional models with the entire system being delivered up-front, the scheduled nature of IID approach delivers functional system increments on a scheduled basis. This allows stakeholders and users to gain benefits from beta software versions, which might already satisfy initial requirements. Hence, this indicates that the fundamental LHDN-compliant features are able to be deployed ahead of time so that users can begin operating specific operations while other features are still in development.

3.2.1 Phase 1 : Initial Planning

3.2.1.1 Requirement Gathering

The web-based e-invoicing system requirements were gathered from the business and regulatory angles. During this phase, gathering of preliminary needs, identification of key stakeholders, and goals for the system were identified. This initiative was conducted in order to fully understand the unique needs of the users and refine the details of the system structure. Furthermore, in-depth discussion with supervisor and informal meeting with company representatives were conducted during this identification stage.

3.2.1.2 Internship with Industry Linked Company

The developer had undergone industrial training with the industry-linked company Hybrid Infinity Tech Sdn Bhd for 3 months and had been involved with the company's project called E-Invoicing Project. This final year project is a direct continuation of the previous project initiated during the internship, with the goal of enhancing, optimizing and extending the work done before to achieve an even more

comprehensive and fully functional system. The developer had also attended meetings with the supervisor of the project on a bi-weekly basis in order to fully understand and gain feedback of the developing system.

3.2.1.3 Review on Existing Systems

During this phase, a review of the existing systems was conducted in order to analyze and compare existing systems, aiming to come up with a novel solution that transcends past constraints and offers greater functionality. With this, four existing e-invoicing systems were compared which were Autocount, Bukku, JomeInvoice, and Financio.

3.2.1.4 Schedule Project

A detailed and thorough Work Breakdown Structure (WBS) was produced with the intention of identifying the major tasks, milestones and deliverables in every phase of the project. Besides that, a Gantt chart was also created using the WBS to provide a visual representation to track the start dates, completion dates as well as dependencies of the tasks. Therefore, these planning tools offered a dynamic environment for tracking progress and rescheduling as needed to guarantee the effective implementation of the E-Invoice System.

3.2.2 Phase 2 : Analysis and Design

In the analysis and design phase, the analysis of the project scope was one of the main objectives of this phase. The project scope was analyzed using a Use Case Diagram and User Requirements Specifications to provide a better understand of the system's flow, architecture and structure before proceeding to the implementation and testing phase. Moreover, simple and easy-to-understand prototypes for the web-based e-invoicing system were also produced to provide better visualization of the system design.

3.2.3 Phase 3 : Implementation and testing

3.2.3.1 Identification of High Priority Features

The implementation process starts by having the project's development team determine and prioritize features that are high in priority in the e-invoicing system. This was achieved through a prioritization matrix that balances the importance of the features against the development complexity in order to ensure that high value and low-complexity features could be addressed first. This ultimately ensures that users would have a functional system that met their immediate invoicing needs as soon as possible, with room for incremental feature additions in subsequent iterations.

3.2.3.2 Iteration 1: Initial Iteration

The first phase of development mainly places emphasis on the creation of the fundamental modules of the system. This includes modules or features such as invoice generation, invoice submission, invoice validation and QR code generation. Furthermore, unit testing and integration testing were also done throughout the development process in order to ensure that the behaviour of the functionality works as intended.

3.2.3.3 Evaluation through User Feedback

After the completion of the first iteration, a review session with the stakeholders is conducted. In this session, a low-fidelity prototype of the application is presented to obtain feedback. This is to ensure that the feedback collected will be a tool to enhance the major functionalities of the system.

3.2.3.4 Iteration 2: Functionalities Enhancement

Using the feedback collected in the previous iteration, this particular iteration focuses on introducing an enhanced version of the new features and system functionalities.

3.2.3.5 Iteration 3: Advanced Functionalities

In this iteration, advanced functionalities were developed in order to enrich user experience and interaction with the system. This iteration in particular focuses on developing non-major functionalities such as notification alert feature and ledger.

3.2.3.6 Iteration 4: Final Iteration and Testing

After implementing all the functionalities, there will be a final iteration and testing to ensure that all the functioning components integrate and work together. The type of testing that will be included are unit testing, integration testing as well as system testing in order to ensure the overall system's functionality.

3.2.4 Phase 4 : Deployment

Following the successful completion of testing and refinements, the e-invoicing system will be rolled out and deployed to the web and made accessible to the users. Furthermore, documentation about the system will be prepared and briefed to the programmer of the company in order to ensure understanding of the implemented features as well as, to increase clarity and clarify any doubts related to the system. This includes explanation of the entire architecture, the functionalities and the overall workflow of the system processes.

3.2.4.1 Work Break-down Structure (WBS)

0.0 Web-based Automated E-Invoicing System
1.0 Project Initiation
1.1 Preliminary Planning
1.1.1 Understand Project Background, LHDN's Documentation and Guidelines
1.1.2 Identify Problems in the Current Traditional Invoicing Practices
1.1.3 Determine Project Objectives
1.1.4 Define Proposed Solution
1.1.4.1 Research Existing E-Invoicing Applications
1.1.4.2 Study the Differences between Applications

1.1.4.3 Select Appropriate Technology Stack

1.1.4.4 Conclude Project Solution

1.1.5 Define Software Development Methodology

1.1.5.1 Research on the Appropriate Methodology to be Applied

1.1.5.2 Conclude the Selected Methodology

1.1.6 Define Project Scope

1.1.6.1 Identify Target Users of the E-Invoicing Application

1.1.6.2 Define System Scope

1.1.6.3 Identify Application Features to be Implemented

1.2 Literature Review

1.2.1 Review Credible Papers and Articles on E-Invoicing Trends

1.2.2 Define History and Evolution of E-Invoicing

1.2.3 Define the Type of E-Invoice Model used in Malaysia

1.2.4 Define the Advantages of E-Invoicing

1.2.5 Review Existing E-Invoicing Systems

1.2.5.1 Identify Common Features

1.2.5.2 Identify Differences in Features

1.3 Methodology and Work Plan

1.3.1 Select the Iterative and Incremental Development methodology

1.3.2 Identify development tools

1.3.3 Develop a detailed Work Breakdown Structure (WBS)

1.3.3.1 Define Main Activities for Each Phases

1.3.4 Develop Gantt Chart and timeline

1.3.4.1 Determine the Timeline of the Project

1.3.4.2 Determine the Task Dependencies

1.3.4.3 Determine the Duration of the Project

1.3.4.4 Produce Gantt Chart

1.4 Requirement Identification

1.4.1 Requirement Specification

1.4.1.1 Collect Functional Requirements

1.4.1.2 Collect Non-functional Requirements

1.4.1.3 Draft Requirements

1.4.2 UML Modelling

1.4.2.1 Create use case diagram

1.4.2.2 Create use case descriptions

2.0 System Development

2.1 Design Interfaces

2.1.1 Develop a low-fidelity prototype

2.2 System Design

2.2.1 Set up the Cloud-based Environment of the E-Invoicing System

2.2.2 Database table design

2.2.3 Create data dictionary

2.2.4 Create data flow diagram

2.2.5 Develop mid-fidelity prototype

3.0 Iteration 1 (Development of Major Functionalities)

3.1 Project Initiation and Planning

3.2 Design and Assess Major Functionalities Architecture

3.3 Implement Major Functionalities

3.3.1 Setup Database

3.3.2 Implement Basic Frontend Interface

3.3.3 Implement User Login Functionality

3.3.3.1 Implement OAuth2 Authentication

3.3.4 Implement Invoice Generation Functionality

3.3.4.1 Implement Digital Signature Encryption Mechanism for Invoices

3.3.5 Implement Invoice Submission with Functionality

3.3.6 Implement Invoice Validation Functionality

3.3.7 Implement Generation of QR Code for Validated Invoices Functionality

3.4 Testing and Quality Assurance

3.4.1 Unit Testing for Major Functionalities

3.4.1.1 Unit Testing for User Login Functionality

3.4.1.2 Unit Testing for Invoice Generation Functionality

3.4.1.3 Unit Testing for Invoice Submission Functionality

3.4.1.4 Unit Testing for Invoice Validation Functionality

3.4.1.5 Unit Testing for Generation of QR Code for Validated Invoices Functionality

3.4.2 Integration Testing for Major Functionalities

3.4.3 Verify Invoice Generation, Submission, Validation and Generation of QR Code Workflow

3.5 Report and Documentation

3.5.1 Document Major Functionalities

3.6 Iteration 1 User Review and Feedback

3.6.1 Demonstrate Major Functionalities to Project Supervisor and Stakeholder

3.6.2 Collect Feedback and Suggestions for Improvement

3.6.3 Analyze Feedback and Incorporate Feedback for Iteration 2

4.0 Iteration 2 (Development in Enhancing Functionalities)

4.1 Review of Supervisor and Stakeholders Feedback

4.1.1 Sort Functionalities According to Priority

4.2 Implement Non-major Functionalities (Enhancing Functionalities)

4.2.1 Implement Automated Form Checks prior to Invoice Submission

4.2.1.1 Implement Autofill Tax Codes

4.2.1.2 Implement Duplicate Entries Detection

4.2.2 Implement PDF Export of E-Invoice

4.3 Interface Improvement from Iteration 1

4.4 Usability Testing

4.4.1 Determine Group Involved in Usability Testing

4.4.2 Conduct Usability Testing

4.4.3 Evaluate Usability Testing Results

4.4.4 Collect Feedback and Suggestions for Improvement

4.4.5 Identify Improvements in the UI/UX Areas

4.4.6 Implement Recommended Enhancements

4.5 Report and Documentation

4.5.1 Document the Development Process in Enhancing Functionalities

4.6 Iteration 2 User Review and Feedback

4.6.1 Demonstrate Enhanced Functionalities to Project Supervisor and Stakeholder

4.6.2 Collect Feedback and Suggestions for Improvement

4.6.3 Analyze Feedback and Incorporate Feedback for Iteration 3

5.0 Iteration 3 (Development in Advanced Functionalities)

5.1 Implement Cancel Invoice Functionality

5.1.1 Implement Invoice Cancellation to LHDN's Cloud

5.1.2 Integrate Invoice Cancellation Functionality into Web Application

5.2 Implement Notification Functionality

5.2.1 Implement Notification and Reminder Alert Mechanisms

5.2.2 Integrate Notification Functionality into Web Application

5.3 Implement Ledger Functionality

5.3.1 Implement Tracking for the Invoices

5.3.2 Integrate Ledger Functionality into Web Application

5.4 Testing and Quality Assurance

5.4.1 Plan User Testing Scenarios

5.4.2 Address any Issues Pertaining to the Application

5.4.3 Improve System following the Data Collected from User Testing Results

5.5 Report and Documentation

5.5.1 Document the Development Process in Advanced Functionalities

5.6 Iteration 3 User Review and Feedback

5.6.1 Demonstrate Advanced Functionalities to Project Supervisor and Stakeholder

5.6.2 Collect Feedback and Suggestions for Improvement

5.6.3 Analyze Feedback and Incorporate Feedback for Iteration 4

6.0 Iteration 4 (Final Iteration and Testing)

6.1 Incorporate User Testing Feedback

6.1.1 Review User Testing Results and Feedback

6.1.2 Identify Areas for Improvement

6.1.3 Integrate Improved Functionalities and UI

6.2 Addition of User-Suggested Improvements

6.2.1 Collect and Assess User Suggestions for Improvement

6.2.2 Select User-Suggested Functionalities based on Priority

6.3 Final Testing of Overall Application

6.3.1 Conduct testing on the Finalized Version of the E-Invoicing System

6.4 Final System Deployment

6.4.1 Use FileZilla for deployment to Direct Admin

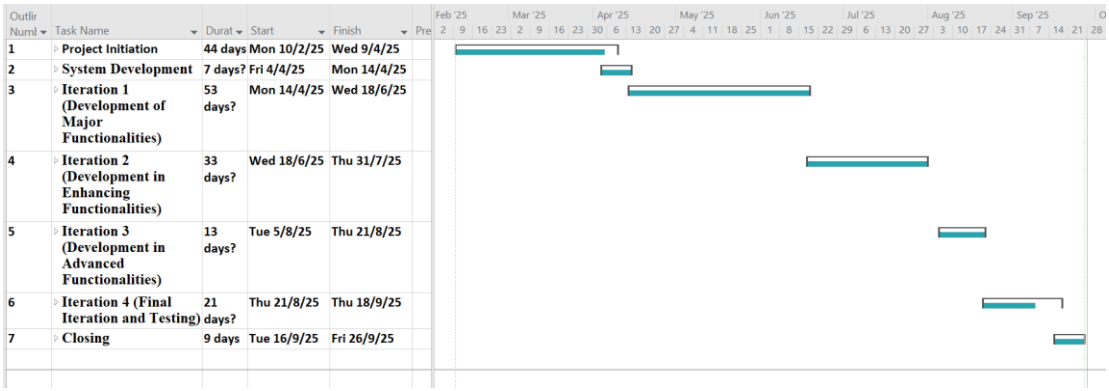
6.4.2 Test hosting and live connectivity

7.0 Closing

7.1 Perform System Usability Scale (SUS) Evaluation.

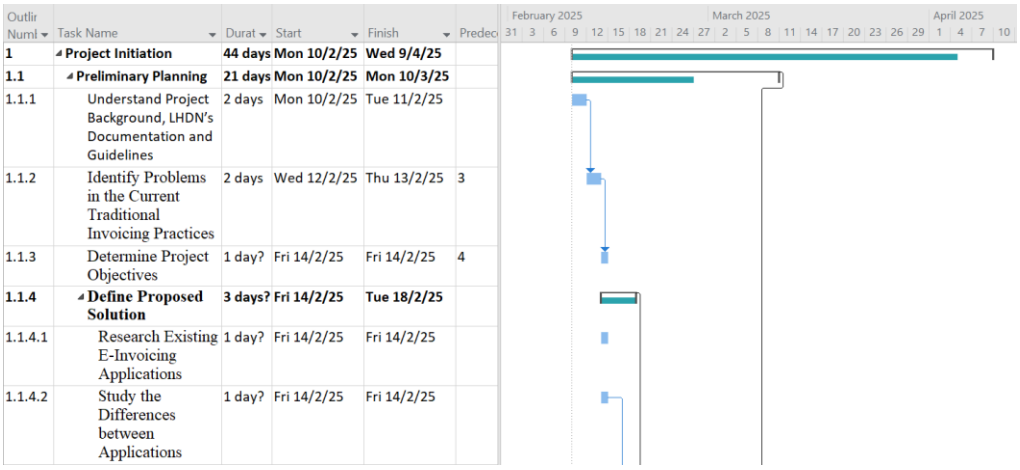
7.2 Complete the final project documentation

3.2.4.2 Gantt Chart



Outlir Numl	Task Name	Durat	Start	Finish
1	Project Initiation	44 days	Mon 10/2/25	Wed 9/4/25
2	System Development	7 days?	Fri 4/4/25	Mon 14/4/25
3	Iteration 1 (Development of Major Functionalities)	53 days?	Mon 14/4/25	Wed 18/6/25
4	Iteration 2 (Development in Enhancing Functionalities)	33 days?	Wed 18/6/25	Thu 31/7/25
5	Iteration 3 (Development in Advanced Functionalities)	13 days?	Tue 5/8/25	Thu 21/8/25
6	Iteration 4 (Final Iteration and Testing)	21 days?	Thu 21/8/25	Thu 18/9/25
7	Closing	9 days	Tue 16/9/25	Fri 26/9/25

Figure 3.2 Overview of the WBS using Gantt Chart



Outlir Numl	Task Name	Durat	Start	Finish	Predec
1	Project Initiation	44 days	Mon 10/2/25	Wed 9/4/25	
1.1	Preliminary Planning	21 days	Mon 10/2/25	Mon 10/3/25	
1.1.1	Understand Project Background, LHDN's Documentation and Guidelines	2 days	Mon 10/2/25	Tue 11/2/25	
1.1.2	Identify Problems in the Current Traditional Invoicing Practices	2 days	Wed 12/2/25	Thu 13/2/25	3
1.1.3	Determine Project Objectives	1 day?	Fri 14/2/25	Fri 14/2/25	4
1.1.4	Define Proposed Solution	3 days?	Fri 14/2/25	Tue 18/2/25	
1.1.4.1	Research Existing E-Invoicing Applications	1 day?	Fri 14/2/25	Fri 14/2/25	
1.1.4.2	Study the Differences between Applications	1 day?	Fri 14/2/25	Fri 14/2/25	

Figure 3.3 Project Initiation Phase

1.1.4.3	Select Appropriate Technology Stack	1 day?	Mon 17/2/25	Mon 17/2/25	8
1.1.4.4	Conclude Project Solution	1 day?	Tue 18/2/25	Tue 18/2/25	9
1.1.5	**Define Software Development Methodology**	**4 days?**	**Wed 19/2/25**	**Mon 24/2/25**	**6**
1.1.5.1	Research on the Appropriate Methodology to be Applied	2 days	Wed 19/2/25	Thu 20/2/25	9
1.1.5.2	Conclude the Selected Methodology	2 days	Fri 21/2/25	Mon 24/2/25	12
1.1.6	**Define Project Scope**	**2 days?**	**Tue 25/2/25**	**Wed 26/2/25**	**11**
1.2	**Literature Review**	**13 days**	**Mon 10/3/25**	**Tue 25/3/25**	**2**
1.3	**Methodology and Work Plan**	**3 days?**	**Tue 25/3/25**	**Thu 27/3/25**	**18**

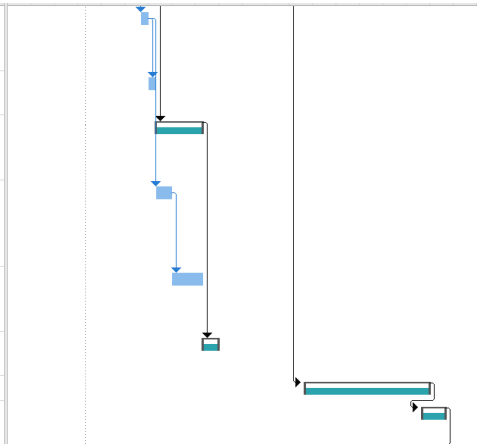


Figure 3.4 Project Initiation Phase (Continued)

Outlir Numl	Task Name	Durat	Start	Finish	Predec
1.3.1	Select the Iterative and Incremental Development methodology	1 day?	Tue 25/3/25	Tue 25/3/25	
1.3.2	Identify development tools	1 day?	Tue 25/3/25	Tue 25/3/25	
1.3.3	**Develop a detailed Work Breakdown Structure (WBS)**	**2 days?**	**Tue 25/3/25**	**Wed 26/3/25**	**27**
1.3.3.1	Define Main Activities for Each Phases	2 days	Tue 25/3/25	Wed 26/3/25	
1.3.4	**Develop Gantt Chart and timeline**	**2 days?**	**Wed 26/3/25**	**Thu 27/3/25**	**29**
1.3.4.1	Determine the Timeline of the Project	1 day?	Wed 26/3/25	Wed 26/3/25	
1.3.4.2	Determine the Task Dependencies	1 day?	Wed 26/3/25	Wed 26/3/25	

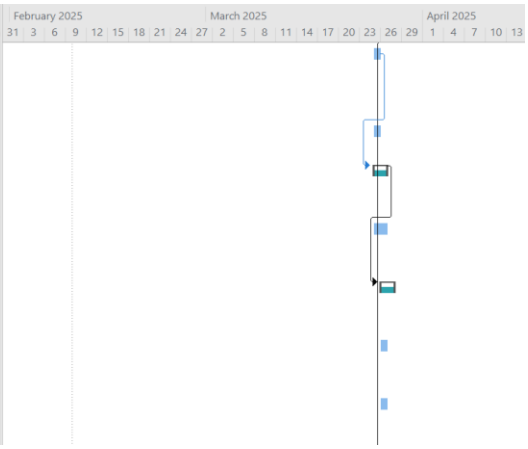


Figure 3.5 Project Initiation Phase (Continued)

Outli Numl	Task Name	Durat	Start	Finish	Predec
1.3.4.3	Determine the Duration of the Project	1 day	Wed 26/3/25	Wed 26/3/25	
1.3.4.4	Produce Gantt Chart	2 days	Wed 26/3/25	Thu 27/3/25	
1.4	**Requirement Identification**	**7 days?**	**Thu 27/3/25**	**Fri 4/4/25**	**26**
1.4.1	**Requirement Specification**	**4 days?**	**Thu 27/3/25**	**Tue 1/4/25**	
1.4.1.1	Collect Functional	1 day?	Thu 27/3/25	Thu 27/3/25	
1.4.1.2	Collect Non-functional Requirements	1 day?	Fri 28/3/25	Fri 28/3/25	38
1.4.1.3	Draft Requirements	1 day?	Mon 31/3/25	Mon 31/3/25	39
1.4.2	**UML Modelling**	**2 days?**	**Mon 31/3/25**	**Tue 1/4/25**	**37**
1.4.2.1	Create use case diagram	1 day?	Mon 31/3/25	Mon 31/3/25	
1.4.2.2	Create use case descriptions	1 day?	Tue 1/4/25	Tue 1/4/25	42

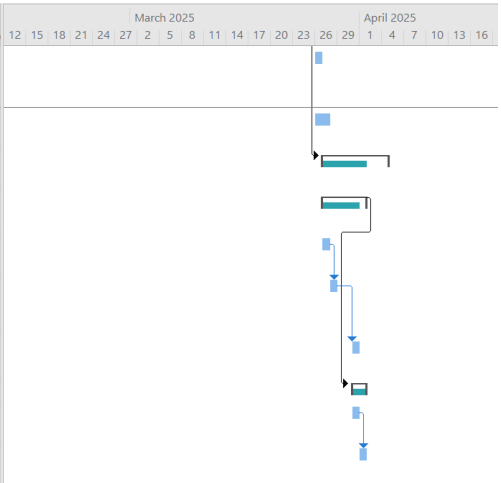
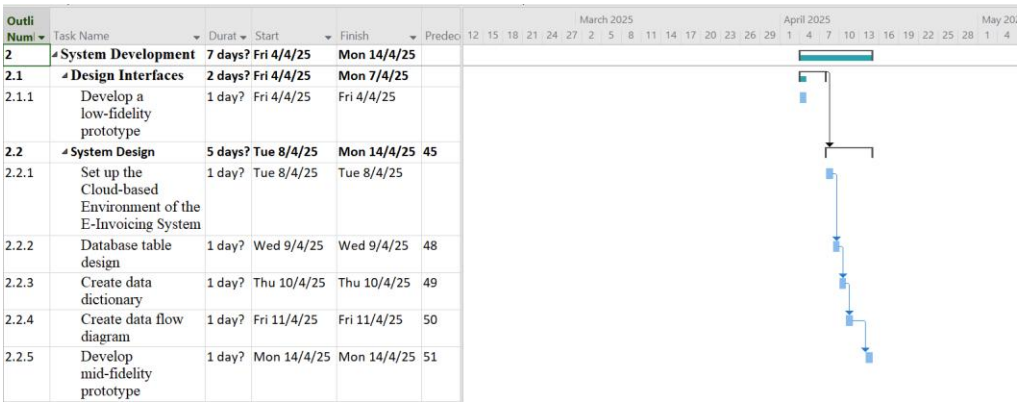
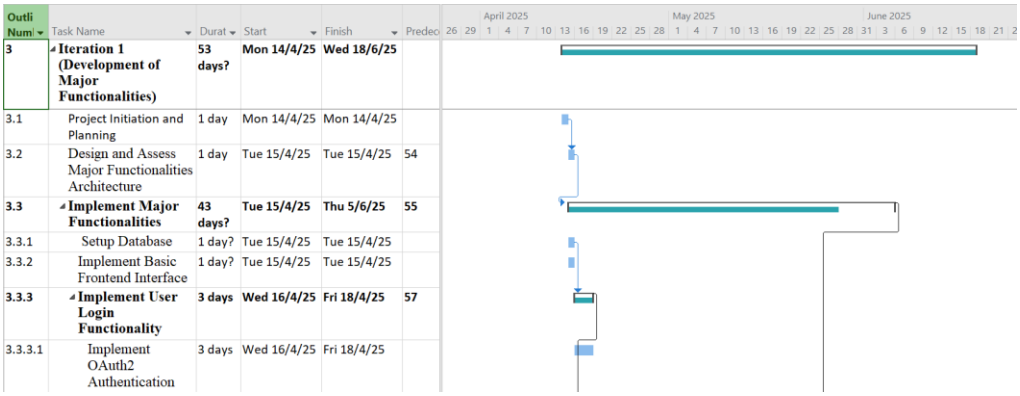


Figure 3.6 Project Initiation Phase (Continued)



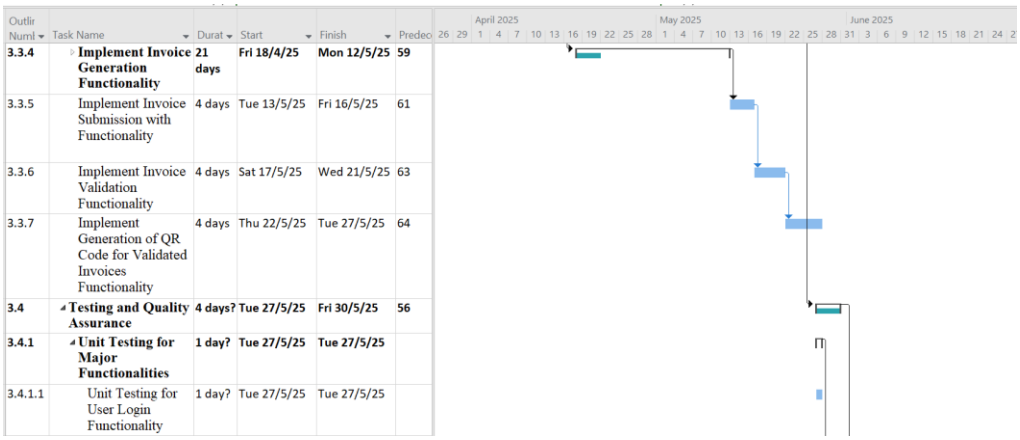
Outli Num	Task Name	Durat	Start	Finish	Predec
2	**System Development**	**7 days?**	**Fri 4/4/25**	**Mon 14/4/25**	
2.1	**Design Interfaces**	**2 days?**	**Fri 4/4/25**	**Mon 7/4/25**	
2.1.1	Develop a low-fidelity prototype	1 day?	Fri 4/4/25	Fri 4/4/25	
2.2	**System Design**	**5 days?**	**Tue 8/4/25**	**Mon 14/4/25**	**45**
2.2.1	Set up the Cloud-based Environment of the E-Invoicing System	1 day?	Tue 8/4/25	Tue 8/4/25	
2.2.2	Database table design	1 day?	Wed 9/4/25	Wed 9/4/25	48
2.2.3	Create data dictionary	1 day?	Thu 10/4/25	Thu 10/4/25	49
2.2.4	Create data flow diagram	1 day?	Fri 11/4/25	Fri 11/4/25	50
2.2.5	Develop mid-fidelity prototype	1 day?	Mon 14/4/25	Mon 14/4/25	51

Figure 3.7 System Development Phase



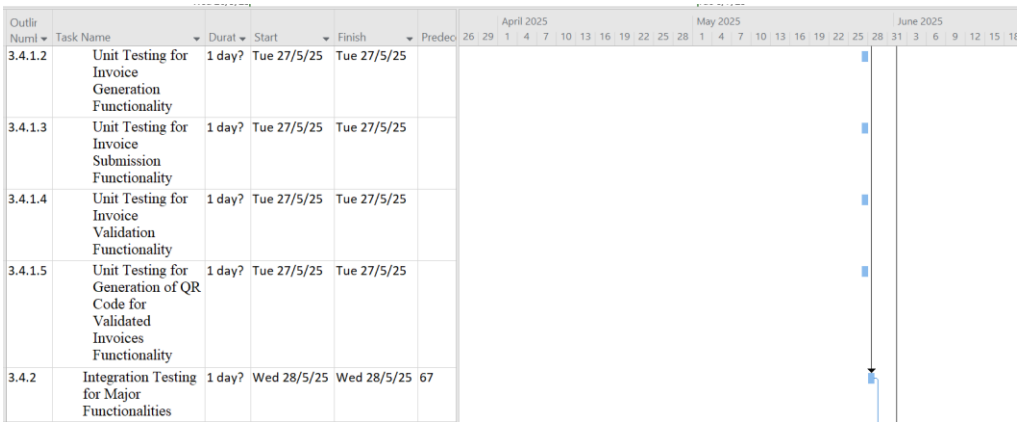
Outli Num	Task Name	Durat	Start	Finish	Predec
3	**Iteration 1 (Development of Major Functionalities)**	**53 days?**	**Mon 14/4/25**	**Wed 18/6/25**	
3.1	Project Initiation and Planning	1 day	Mon 14/4/25	Mon 14/4/25	
3.2	Design and Assess Major Functionalities Architecture	1 day	Tue 15/4/25	Tue 15/4/25	54
3.3	**Implement Major Functionalities**	**43 days?**	**Tue 15/4/25**	**Thu 5/6/25**	**55**
3.3.1	Setup Database	1 day?	Tue 15/4/25	Tue 15/4/25	
3.3.2	Implement Basic Frontend Interface	1 day?	Tue 15/4/25	Tue 15/4/25	
3.3.3	**Implement User Login Functionality**	**3 days**	**Wed 16/4/25**	**Fri 18/4/25**	**57**
3.3.3.1	Implement OAuth2 Authentication	3 days	Wed 16/4/25	Fri 18/4/25	

Figure 3.8 Iteration 1



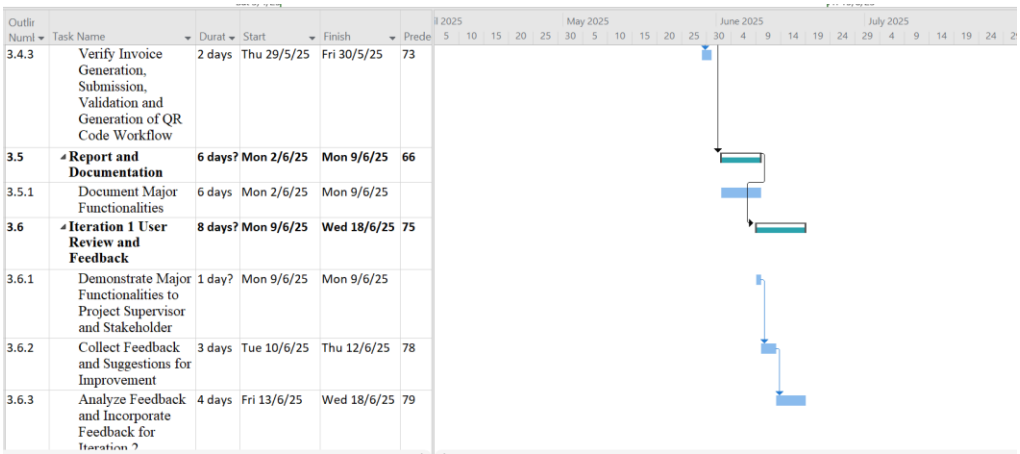
Outli Num	Task Name	Durat	Start	Finish	Predec
3.3.4	**Implement Invoice Generation Functionality**	**21 days**	**Fri 18/4/25**	**Mon 12/5/25**	**59**
3.3.5	Implement Invoice Submission with Functionality	4 days	Tue 13/5/25	Fri 16/5/25	61
3.3.6	Implement Invoice Validation Functionality	4 days	Sat 17/5/25	Wed 21/5/25	63
3.3.7	Implement Generation of QR Code for Validated Invoices Functionality	4 days	Thu 22/5/25	Tue 27/5/25	64
3.4	**Testing and Quality Assurance**	**4 days?**	**Tue 27/5/25**	**Fri 30/5/25**	**56**
3.4.1	**Unit Testing for Major Functionalities**	**1 day?**	**Tue 27/5/25**	**Tue 27/5/25**	
3.4.1.1	Unit Testing for User Login Functionality	1 day?	Tue 27/5/25	Tue 27/5/25	

Figure 3.9 Iteration 1 (Continued)



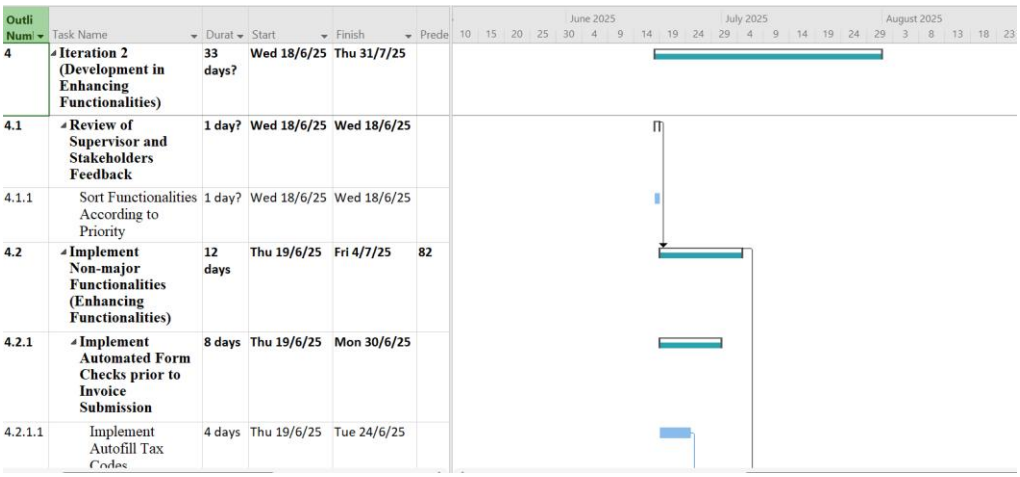
Outlir Numl	Task Name	Durat	Start	Finish	Predec
3.4.1.2	Unit Testing for Invoice Generation Functionality	1 day?	Tue 27/5/25	Tue 27/5/25	
3.4.1.3	Unit Testing for Invoice Submission Functionality	1 day?	Tue 27/5/25	Tue 27/5/25	
3.4.1.4	Unit Testing for Invoice Validation Functionality	1 day?	Tue 27/5/25	Tue 27/5/25	
3.4.1.5	Unit Testing for Generation of QR Code for Validated Invoices Functionality	1 day?	Tue 27/5/25	Tue 27/5/25	
3.4.2	Integration Testing for Major Functionalities	1 day?	Wed 28/5/25	Wed 28/5/25	67

Figure 3.10 Iteration 1 (Continued)



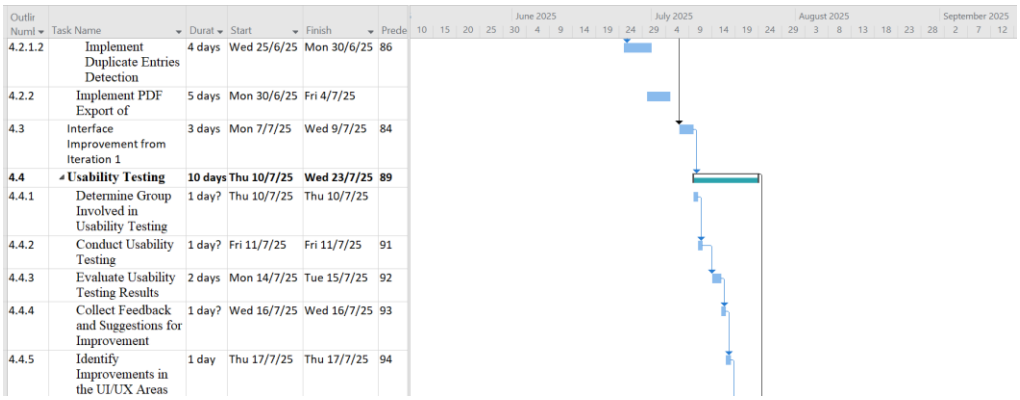
Outlir Numl	Task Name	Durat	Start	Finish	Prede
3.4.3	Verify Invoice Generation, Submission, Validation and Generation of QR Code Workflow	2 days	Thu 29/5/25	Fri 30/5/25	73
3.5	**Report and Documentation**	**6 days?**	**Mon 2/6/25**	**Mon 9/6/25**	**66**
3.5.1	Document Major Functionalities	6 days	Mon 2/6/25	Mon 9/6/25	
3.6	**Iteration 1 User Review and Feedback**	**8 days?**	**Mon 9/6/25**	**Wed 18/6/25**	**75**
3.6.1	Demonstrate Major Functionalities to Project Supervisor and Stakeholder	1 day?	Mon 9/6/25	Mon 9/6/25	
3.6.2	Collect Feedback and Suggestions for Improvement	3 days	Tue 10/6/25	Thu 12/6/25	78
3.6.3	Analyze Feedback and Incorporate Feedback for Iteration 2	4 days	Fri 13/6/25	Wed 18/6/25	79

Figure 3.11 Iteration 1 (Continued)



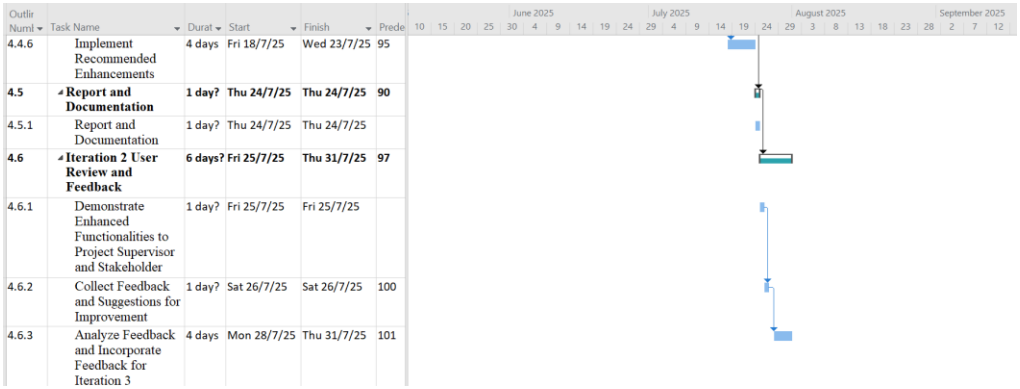
Outli Numl	Task Name	Durat	Start	Finish	Prede
4	**Iteration 2 (Development in Enhancing Functionalities)**	**33 days?**	**Wed 18/6/25**	**Thu 31/7/25**	
4.1	**Review of Supervisor and Stakeholders Feedback**	**1 day?**	**Wed 18/6/25**	**Wed 18/6/25**	
4.1.1	Sort Functionalities According to Priority	1 day?	Wed 18/6/25	Wed 18/6/25	
4.2	**Implement Non-major Functionalities (Enhancing Functionalities)**	**12 days**	**Thu 19/6/25**	**Fri 4/7/25**	**82**
4.2.1	**Implement Automated Form Checks prior to Invoice Submission**	**8 days**	**Thu 19/6/25**	**Mon 30/6/25**	
4.2.1.1	Implement Autofill Tax Codes	4 days	Thu 19/6/25	Tue 24/6/25	

Figure 3.12 Iteration 2



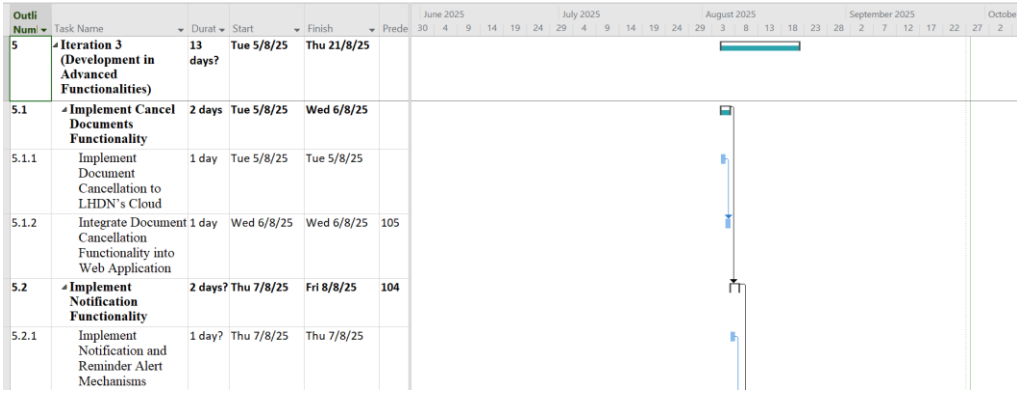
Outlir Numl	Task Name	Durat	Start	Finish	Prede
4.2.1.2	Implement Duplicate Entries Detection	4 days	Wed 25/6/25	Mon 30/6/25	86
4.2.2	Implement PDF Export of	5 days	Mon 30/6/25	Fri 4/7/25	
4.3	Interface Improvement from Iteration 1	3 days	Mon 7/7/25	Wed 9/7/25	84
4.4	**Usability Testing**	**10 days**	**Thu 10/7/25**	**Wed 23/7/25**	**89**
4.4.1	Determine Group Involved in Usability Testing	1 day?	Thu 10/7/25	Thu 10/7/25	
4.4.2	Conduct Usability Testing	1 day?	Fri 11/7/25	Fri 11/7/25	91
4.4.3	Evaluate Usability Testing Results	2 days	Mon 14/7/25	Tue 15/7/25	92
4.4.4	Collect Feedback and Suggestions for Improvement	1 day?	Wed 16/7/25	Wed 16/7/25	93
4.4.5	Identify Improvements in the UI/UX Areas	1 day	Thu 17/7/25	Thu 17/7/25	94

Figure 3.13 Iteration 2 (Continued)



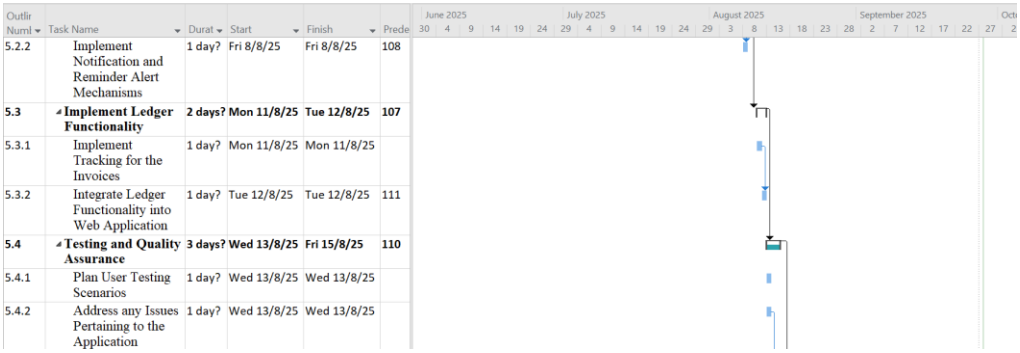
Outlir Numl	Task Name	Durat	Start	Finish	Prede
4.4.6	Implement Recommended Enhancements	4 days	Fri 18/7/25	Wed 23/7/25	95
4.5	**Report and Documentation**	**1 day?**	**Thu 24/7/25**	**Thu 24/7/25**	**90**
4.5.1	Report and Documentation	1 day?	Thu 24/7/25	Thu 24/7/25	
4.6	**Iteration 2 User Review and Feedback**	**6 days?**	**Fri 25/7/25**	**Thu 31/7/25**	**97**
4.6.1	Demonstrate Enhanced Functionalities to Project Supervisor and Stakeholder	1 day?	Fri 25/7/25	Fri 25/7/25	
4.6.2	Collect Feedback and Suggestions for Improvement	1 day?	Sat 26/7/25	Sat 26/7/25	100
4.6.3	Analyze Feedback and Incorporate Feedback for Iteration 3	4 days	Mon 28/7/25	Thu 31/7/25	101

Figure 3.14 Iteration 2 (Continued)



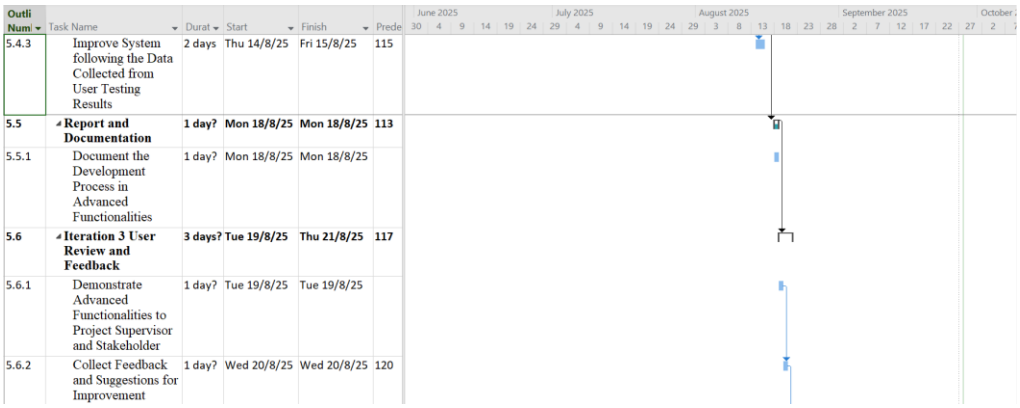
Outli Num	Task Name	Durat	Start	Finish	Prede
5	**Iteration 3 (Development in Advanced Functionalities)**	**13 days?**	**Tue 5/8/25**	**Thu 21/8/25**	
5.1	**Implement Cancel Documents Functionality**	**2 days**	**Tue 5/8/25**	**Wed 6/8/25**	
5.1.1	Implement Document Cancellation to LHDN's Cloud	1 day	Tue 5/8/25	Tue 5/8/25	
5.1.2	Integrate Document Cancellation Functionality into Web Application	1 day	Wed 6/8/25	Wed 6/8/25	105
5.2	**Implement Notification Functionality**	**2 days?**	**Thu 7/8/25**	**Fri 8/8/25**	**104**
5.2.1	Implement Notification and Reminder Alert Mechanisms	1 day?	Thu 7/8/25	Thu 7/8/25	

Figure 3.15 Iteration 3



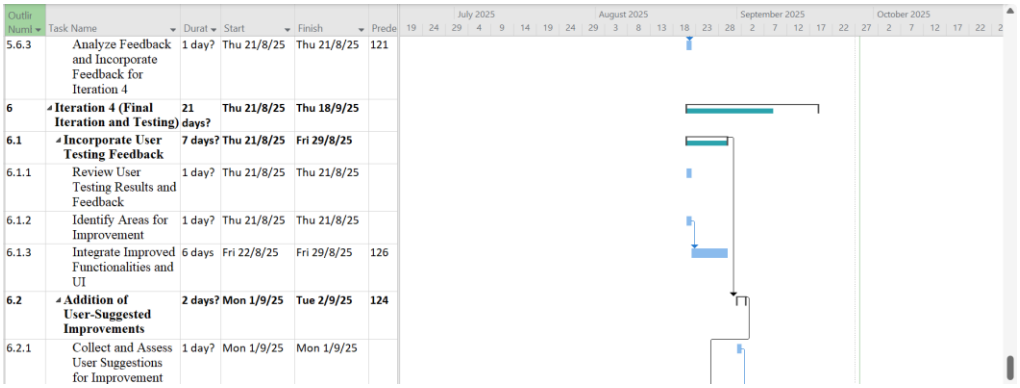
Outlir Numl	Task Name	Durat	Start	Finish	Prede
5.2.2	Implement Notification and Reminder Alert Mechanisms	1 day?	Fri 8/8/25	Fri 8/8/25	108
5.3	**Implement Ledger Functionality**	**2 days?**	**Mon 11/8/25**	**Tue 12/8/25**	**107**
5.3.1	Implement Tracking for the Invoices	1 day?	Mon 11/8/25	Mon 11/8/25	
5.3.2	Integrate Ledger Functionality into Web Application	1 day?	Tue 12/8/25	Tue 12/8/25	111
5.4	**Testing and Quality Assurance**	**3 days?**	**Wed 13/8/25**	**Fri 15/8/25**	**110**
5.4.1	Plan User Testing Scenarios	1 day?	Wed 13/8/25	Wed 13/8/25	
5.4.2	Address any Issues Pertaining to the Application	1 day?	Wed 13/8/25	Wed 13/8/25	

Figure 3.16 Iteration 3 (Continued)



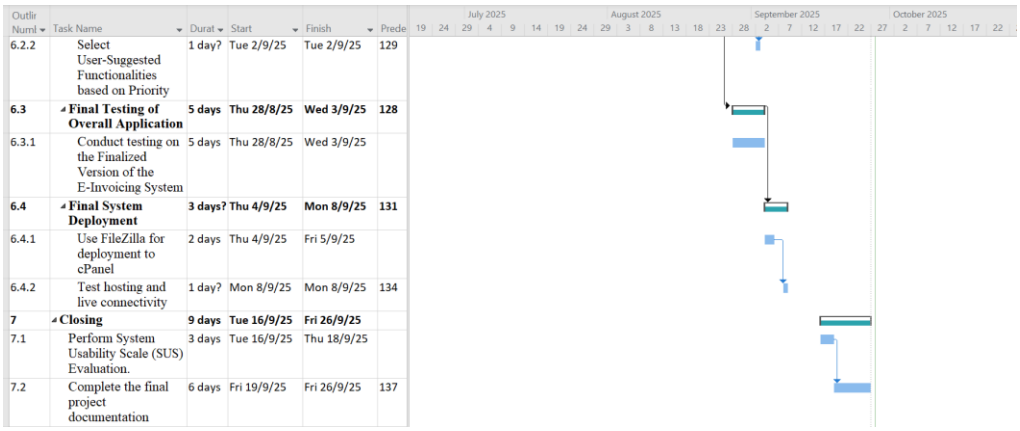
Outli Numl	Task Name	Durat	Start	Finish	Prede
5.4.3	Improve System following the Data Collected from User Testing Results	2 days	Thu 14/8/25	Fri 15/8/25	115
5.5	**Report and Documentation**	**1 day?**	**Mon 18/8/25**	**Mon 18/8/25**	**113**
5.5.1	Document the Development Process in Advanced Functionalities	1 day?	Mon 18/8/25	Mon 18/8/25	
5.6	**Iteration 3 User Review and Feedback**	**3 days?**	**Tue 19/8/25**	**Thu 21/8/25**	**117**
5.6.1	Demonstrate Advanced Functionalities to Project Supervisor and Stakeholder	1 day?	Tue 19/8/25	Tue 19/8/25	
5.6.2	Collect Feedback and Suggestions for Improvement	1 day?	Wed 20/8/25	Wed 20/8/25	120

Figure 3.17 Iteration 3 (Continued)



Outlir Numl	Task Name	Durat	Start	Finish	Prede
5.6.3	Analyze Feedback and Incorporate Feedback for Iteration 4	1 day?	Thu 21/8/25	Thu 21/8/25	121
6	**Iteration 4 (Final Iteration and Testing)**	**21 days?**	**Thu 21/8/25**	**Thu 18/9/25**	
6.1	**Incorporate User Testing Feedback**	**7 days?**	**Thu 21/8/25**	**Fri 29/8/25**	
6.1.1	Review User Testing Results and Feedback	1 day?	Thu 21/8/25	Thu 21/8/25	
6.1.2	Identify Areas for Improvement	1 day?	Thu 21/8/25	Thu 21/8/25	
6.1.3	Integrate Improved Functionalities and UI	6 days	Fri 22/8/25	Fri 29/8/25	126
6.2	**Addition of User-Suggested Improvements**	**2 days?**	**Mon 1/9/25**	**Tue 2/9/25**	**124**
6.2.1	Collect and Assess User Suggestions for Improvement	1 day?	Mon 1/9/25	Mon 1/9/25	

Figure 3.18 Iteration 3 (Continued) and Iteration 4



Outlir Numl	Task Name	Durat	Start	Finish	Prede
6.2.2	Select User-Suggested Functionalities based on Priority	1 day?	Tue 2/9/25	Tue 2/9/25	129
6.3	**Final Testing of Overall Application**	**5 days**	**Thu 28/8/25**	**Wed 3/9/25**	**128**
6.3.1	Conduct testing on the Finalized Version of the E-Invoicing System	5 days	Thu 28/8/25	Wed 3/9/25	
6.4	**Final System Deployment**	**3 days?**	**Thu 4/9/25**	**Mon 8/9/25**	**131**
6.4.1	Use FileZilla for deployment to cPanel	2 days	Thu 4/9/25	Fri 5/9/25	
6.4.2	Test hosting and live connectivity	1 day?	Mon 8/9/25	Mon 8/9/25	134
7	**Closing**	**9 days**	**Tue 16/9/25**	**Fri 26/9/25**	
7.1	Perform System Usability Scale (SUS) Evaluation.	3 days	Tue 16/9/25	Thu 18/9/25	
7.2	Complete the final project documentation	6 days	Fri 19/9/25	Fri 26/9/25	137

Figure 3.19 Iteration 4 (Continued) and Closing Phase

3.3 Project Tools

The development of the e-Invoicing system requires a multitude of tools and technologies in order to ensure smooth development, ease of updates and deployment. The array of tools were used to support multiple aspects of the system, which includes from backend development and database management to frontend development, testing, as well as, in project management. The following section entails the tools and technologies that were used and its relevance in the development of the e-Invoice system.

3.3.1 Programming and Markup Languages

3.3.1.1 PHP

PHP is a popular open-source server-side scripting language that is well-suited for web development and is integrated into HTML. PHP is especially essential for constructing dynamic web applications such as the e-Invoice system. It is used primarily to handle a variety of backend tasks within web applications.

3.3.1.2 HTML

HTML, which is synonymously known as Hypertext Markup Language, is primarily known as the standard markup language for constructing web pages. HTML is also known as a client-side language. It is typically used throughout the application to provide a visual interface as well as to define the structure and layout of the application that users see in their browsers.

3.3.1.3 CSS

CSS, which is synonymously known as Cascading Style Sheets, is primarily used to provide styling such as layout, colours, fonts, spacing, and responsiveness to the application. It is responsible for controlling the appearance and design of the HTML components. In addition, CSS also functions to ensure that the user interface is responsive and visually appealing to users.

3.3.1.4 JavaScript

JavaScript (JS) is a client-side scripting language in web development that adds interactivity and dynamic behaviour on web pages. Besides that, it is also in-charge of improving the front-end user experience by making the web pages responsive, interactive and engaging to the users of the e-invoicing system.

3.3.2 Development Tools and IDE

3.3.2.1 Visual Studio Code

Visual Studio Code (VSCode) is the primary integrated development environment (IDE) used for both the frontend and backend of the development. VSCode in particular, is also a capable code editor that supports a multitude of programming languages such as PHP, JavaScript, HTML, CSS and many more. Besides that, VSCode has many useful features which includes debugging tools and a multitude of extensions which serve to improve the functionalities of the system as well as to speed up development efficiency.

3.3.2.2 Microsoft Project

Microsoft Project is a project management software application primarily used to serve the purpose of planning, scheduling, and managing projects. Microsoft Project is used in the e-invoicing system to manage the project's timeline, resources, and milestones. By utilising this program, the development team may create a comprehensive project plan that covers all tasks, deliverables, and deadlines. Apart from that, this software also helps in tracking progress, assigning resources, and ensuring that the project is completed on time. Furthermore, Microsoft Project also has an additional feature called the Gantt chart tool, which provides a comprehensive visual representation of the project's progress. This in return, eases the development process as it serves to identify potential bottlenecks.

3.3.2.3 Enterprise Architect

Enterprise Architect is a comprehensive modelling tool used to develop detailed diagrams and models of the e-Invoice system's design and architecture. Apart from that, this particular tool is a useful tool which aids in developing complex systems as it supports a variety of modelling approaches such as the UML (Unified Modelling Language), BPMN (Business Process Model and Notation), and SysML (Systems Modelling Language). Finally, this tool facilitates collaboration among team members by offering a unified view of the system's architecture.

3.3.3 Database

3.3.3.1 MySQL

MySQL is an open-source relational database management system (RDBMS) that primarily manages data within the application using structured query language. Besides that, MySQL also serves as the central database for the e-invoice system, which includes storing and managing data and important information about the web application. Apart from that, the relational structure of the database also enables efficient querying via queries, thereby ensuring that the data is kept in an ordered way and is easily accessible when needed.

3.3.4 API Testing Tool

3.3.4.1 Postman

POSTMAN is an API (Application Programming Interface) development tool that lets developers test and interact with a variety of APIs through an easy-to-understand interface. This particular tool allows developers to easily test RESTful APIs through request creation, response validation and error troubleshooting. Besides that, Postman is important for testing the connection between the system's frontend and backend, particularly when integrating with external systems such as LHDN, in ensuring tax compliance and invoice validation.

3.3.5 Deployment and Hosting Tools

3.3.5.1 FileZilla

FileZilla is an open-source application that utilizes FTP (File Transfer Protocol) to transfer data securely between a local computer and a remote server. FileZilla is utilized by the e-Invoice project to upload and save files onto the web server. FileZilla also simplifies handling the server, whether it is uploading PHP scripts, images, configuration files, or logs, making the e-Invoice system easily accessible online.

3.3.5.2 DirectAdmin Control Panel

DirectAdminPanel was used for web hosting management. This hosting tool had a graphical user interface and automated capabilities which aid in tasks like domain management, database initialization, server usage monitoring, and SSL certificate configuration. With DirectAdmin Control Panel, it eases and quickens the deployment from the development phase, as well as assists in maintaining operational stability after deployment.

3.4 Summary

The proposed development methodology used by the system is iterative and incremental. Besides that, the Work Breakdown Structure (WBS) and Gantt Chart provide a comprehensive breakdown of the main processes involved in each stages. Hence, all the major processes involved in each phase of the development of the project were identified. Furthermore, the development tools used in the project were explained in detail in order to provide a comprehensive explanation of the usage of the tools in bringing this project to completion.

CHAPTER 4

PROJECT SPECIFICATIONS

4.1 Introduction

This chapter explains the requirements gathering and analysis. Requirement specification, use case diagram, use case descriptions, requirements traceability matrix and user requirement specifications (URS) are created and explained in an in-depth manner in order to provide full visualization of the system's overall architecture, workflows, as well as user interactions with the system's functionality.

4.2 Requirement Specification

4.2.1 User Requirements Specification

ID	Requirements
USR001	Users must be able to securely log into the e-Invoicing web application with their account.
USR002	The system shall be able to generate and update the form before submitting it for e-invoice generation.
USR003	The system must be able to encrypt the invoice before submitting it for e-invoice generation.
USR004	The system must be able to validate the submitted invoice's data against the schema provided by LHDN.
USR005	The system should be able to generate a QR code of the validated e-invoice.

USR006	The system should be able to cancel the submitted e-invoices.
USR007	The system should be able to receive notifications related to their e-invoicing activities.
USR008	The system shall provide users with a ledger for traceability purposes.
USR009	Users must be able to view details of their status of invoice submission.

4.2.2 Functional Requirements

ID	Requirements
FR001	The system shall allow users to log into the e-invoicing web application securely using their username, password and with attached OAuth2-generated JWT tokens.
FR002	The system shall allow users to input, create and update the required fields in the form before generating an invoice.
FR003	The system must be able to encrypt the invoice using digital signature mechanisms such as SHA-256 hashing and Base64 encoding before submitting it for e-invoice generation.
FR004	The system shall allow users to send created invoices via the Submit Documents API to LHDN's cloud.

FR005	The system must be able to validate the submitted invoice's data against the schema provided by LHDN.
FR006	The system should be able to cancel the previously submitted invoice from the LHDN's cloud within 72 hours of submission.
FR007	The system should be able to generate and send real-time notifications and status updates to the user about any e-invoicing activities.
FR008	The system shall provide users with a ledger which provides an overview of all financial activities for traceability purposes.
FR009	The system shall be able to retrieve and output the validation feedback and status updates from LHDN.
FR010	The system shall allow to generate the QR code based on the longId retrieved from the Get Submissions API.

4.2.3 Non-Functional Requirements

ID	Requirement	Category of NFR
NFR001	The system shall ensure that the invoice generation and submission have a response time of 3 seconds or less under normal load.	Performance
NFR002	The system should ensure that the e-invoice utilizes strong encryption that leverages SHA-256 hashing and Base64 encoding.	Security
NFR003	The system shall authenticate the user with the username, password and OAuth2-generated JWT tokens.	Security

NFR004	The system shall have a clear and intuitive user interface with a user-friendly and responsive UI design.	Usability
NFR005	The system must be available to all users with an uptime of 99.99%.	Availability

4.2.4 Use Case Modelling

4.2.4.1 Use Case Diagram

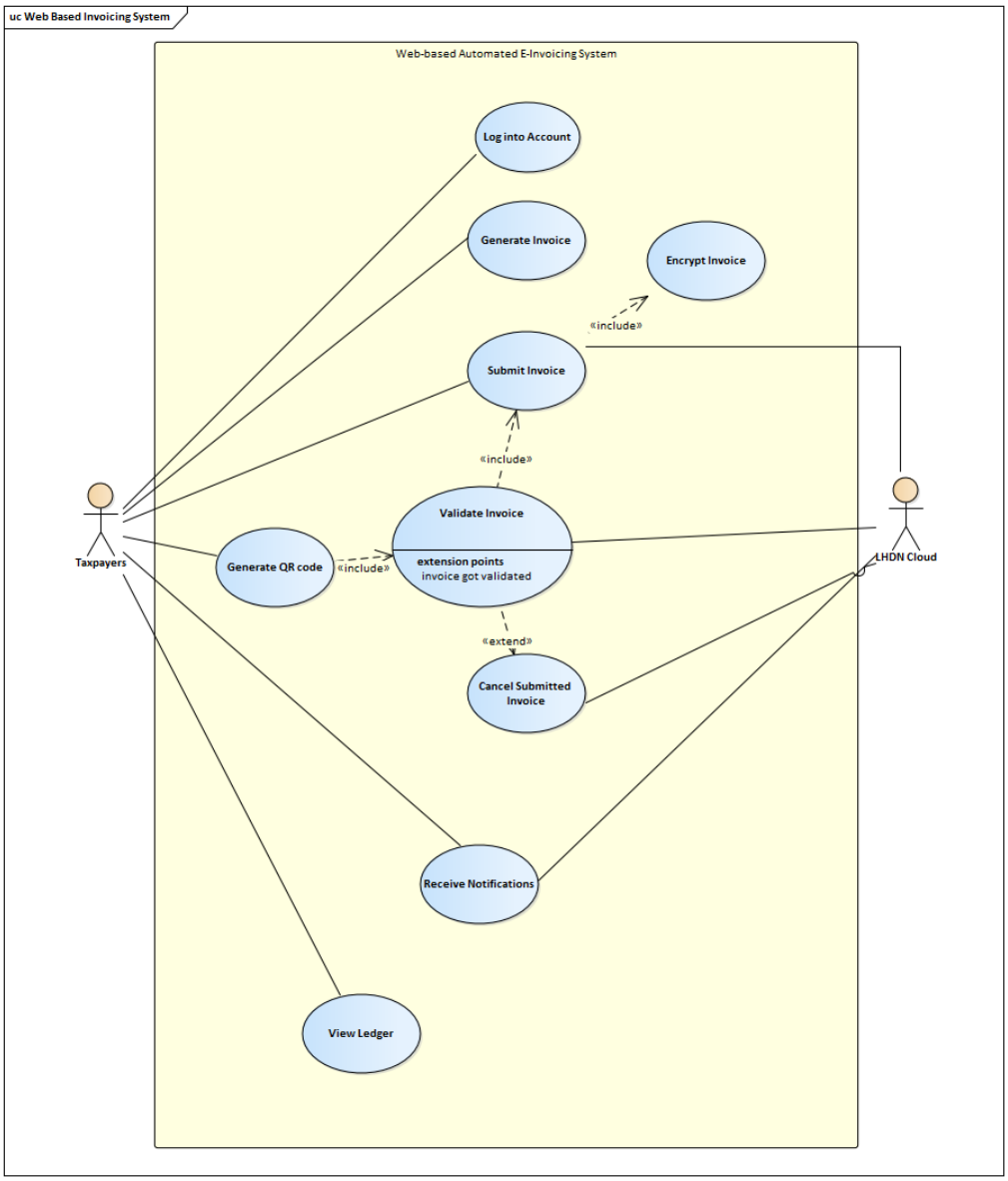


Figure 4.1 Web Based Invoicing System Use Case Diagram

4.2.4.2 Use Case Description

<table>
<tr><td colspan="2">Use Case Name: Log Into Account</td><td>ID: FR001</td><td>Importance Level: High</td></tr>
<tr><td colspan="2">Primary Actor: Taxpayers</td><td colspan="2">Use Case Type: Detail, Real</td></tr>
<tr><td colspan="4">Stakeholders and Interests:

Taxpayers – User who wants to log into their account in the E-Invoicing System.</td></tr>
<tr><td colspan="4">Brief Description: This use case describes how taxpayers logs into the e-invoicing system and utilize its features.</td></tr>
<tr><td colspan="4">Trigger: The taxpayer navigates to the system login page.</td></tr>
<tr><td colspan="4">Relationships:

Association : Taxpayers

Include : N/A

Extend : N/A

Generalization : N/A</td></tr>
</table>

Normal Flow of Events**:**

1. The taxpayer launches the Web Based E-Invoicing System.
2. The system requests the taxpayer to provide account login information which includes the taxpayer's username and password.
3. The taxpayer enters the necessary account login information.
4. The system checks the account validity by the details entered by the taxpayer. *Perform 4.1 or 4.2.*
5. The system then generates an OAuth2-based JWT Token and stores it into the database and attaches it to the user session.
6. The system redirects the taxpayer to the home page after successful login.

Sub-flows**:**

4.1 If the account is invalid,

4.1.1 The system prompts the taxpayer to re-enter different information for account login.

4.1.2 Return to flow no.3.

4.2 If the account is valid,

4.2.1 Continue to flow no.5.

Alternate/Exceptional Flows:

3a. The taxpayer must enter the correct username and password.

Use Case Name: Generate Invoice	ID: FR002	Importance Level: High
Primary Actor: Taxpayer	Use Case Type: Detail, Real	
Stakeholders and Interests**:** Taxpayer: interested in generating e-invoices as proof of digital transaction.		
Brief Description: This use case describes the process of generating an invoice from the invoice form.		
Trigger**:** A taxpayer navigates to the "Create Invoice" page to generate invoices.		
Relationships: Association : Taxpayer Include : Encrypt Invoice Extend : N/A Generalization : N/A		

Normal Flow of Events**:**

1. The taxpayer who has recently logged in navigates to the "Create Invoice" page.
2. The system displays a form that includes multiple text-fields of items and their quantity and prices to be filled in.
3. The taxpayer inputs in information such as customer information, item details, amount and many more.
4. The system processes the input with client-side validation.
5. The system checks for invalid, inaccurate or missing fields. *Perform 5.1 or 5.2*
6. The taxpayer then clicks on the "Sync" button to sync all the inputs in the text fields to an invoice.

Sub-flows**:**

5.1 If the taxpayer inputs an invalid, inaccurate or missing value,

5.1.1 The system will display an invalid message.

5.1.2 Return to flow no.3.

5.2 If the taxpayer inputs an accurate value,

5.2.1 Continue to flow no.6.

Alternate/Exceptional Flows:

3a. The taxpayer must enter a valid and complete input for the text fields.

Use Case Name: Encrypt Invoice	ID: FR003	Importance Level: High
Primary Actor: E-Invoicing System	Use Case Type: Detail, Essential	
Stakeholders and Interests**:** Taxpayer: wants reassurance that their sensitive data contained in the invoice is safeguarded against unwanted access. LHDN: requires that all submitted invoices are encrypted to ensure data security and privacy.		
Brief Description: This use case describes the process of how the system encrypts the data contained in the invoice before submitting it to LHDN's cloud.		
Trigger: A taxpayer creates the invoice, and the digital signing of the invoice is immediately triggered.		
Relationships: Association : System Include : N/A Extend : N/A Generalization : N/A		

Normal Flow of Events**:**

1. The system retrieves the invoice data entered previously by the taxpayer during the input of data into the text fields.
2. The system selects the appropriate digital signature mechanisms, such as SHA-256 for hashing and RSA-256 for signing the data.
3. The system then extracts the certificate to retrieve the personal data contained in the certificate.
4. The invoice data is then encrypted using the selected algorithm, key and certificate.
5. The successfully encrypted JSON invoice is then saved as a digitally signed invoice before proceeding to the invoice submission process.

Sub-flows: -

Alternate/Exceptional Flows:

3a. The system must ensure that the certificate is not expired.

4a. The system must ensure that the process of digitally signing the invoice data is in compliance with the SDK documentation provided by LHDN.

A. The system must ensure that the digitally signed invoice is either in JSON or XML format.

B. The system must ensure that the digitally signed signature follows the UBL 2.1 schema.

Use Case Name: Submit Invoice	ID: FR004	Importance Level: High
Primary Actor**:** Taxpayers	Use Case Type: Detail, Real	
Stakeholders and Interests**:** Taxpayer: wants the submitted digitally signed invoice to be accepted. LHDN: Ensures the submitted digitally signed invoice is authentic and valid according to the schema.		
Brief Description**:** This use case describes the process of submitting a complete, valid and authentic digitally signed invoice.		
Trigger**:** Taxpayer clicks on the "Submit to LHDN" button after successfully syncing. The digitally signed invoice is then sent over as payload to the Submit Documents API.		
Relationships: Association : Taxpayer, LHDN Include : Encrypt Invoice Extend : N/A Generalization : N/A		

Normal Flow of Events**:**

1. The taxpayer clicks on the “Submit to LHDN” button after successfully syncing.
2. The system retrieves the digitally signed invoice.
3. The system then packages the invoice into a payload specifying the format of the invoice, the hash value, code number, and the base64 of the JSON invoice before submission.
4. The system submits the digitally signed invoice to LHDN cloud.
5. The system will then retrieve the status code of the response payload. *Perform 5.1 or 5.2.*
6. The system will then store the submission UID and uuid of the invoice in the session and database.
7. The system will then alert taxpayers with a notification of successful submission.

Sub-flows**:**

5.1 If the response payload returns a response message that is badly structured, incorrect or a duplicate submission,

5.1.1 The taxpayer modifies the details of the invoice.

5.1.2 Return to flow no.4.

5.2 If the response payload returns a successful response message,

5.2.1 Continue to flow no.6.

Alternate/Exceptional Flows:

3a. The taxpayer must ensure the payload components, such as hash value and the base64 of the digitally signed invoice in JSON format, is correctly

done and in accordance with the LHDN's digital signature schema.

Use Case Name: Validate Invoice	ID: FR005	Importance Level: High
Primary Actor: LHDN Cloud	Use Case Type: Detail, Real	
Stakeholders and Interests: Taxpayer: is interested in their submitted invoices being checked as soon as possible to ensure compliance and to avoid rejection. LHDN: wants to check that the e-Invoice complies with all the applicable tax legislations, guidelines, and digital signature validity.		
Brief Description: This use case describes the process of the system validating a submitted invoice. The validation process comprises of reviewing the invoice schema, invoice data, tax computations, and the legitimacy of the digital signature.		
Trigger: The invoice details such as the Submission UID and the UUID is sent over to the Get Submissions API.		
Relationships: Association : N/A Include : Submit Invoice Extend : Cancel Submitted Invoice Generalization : N/A		
Normal Flow of Events: 1. The system retrieves the Submission UUID or UUID stored in the session. 2. The system sends an API call with the Submission UID or UUID to the Get Submissions API. 3. The systems then validate the invoice and check for any errors that occurred during submission. 4. The system receives a response after validation. *Perform 4.1 or 4.2.*		

5. The system then retrieves the longId based on successful validation check. 6. The system then stores the longId into the session.
Sub-flows: 4.1 If all validation checks are successful, 4.1.1 The system declares that the invoice is valid by returning a response message. 4.1.2 Continue with flow 5. 4.2 If certain validation checks are unsuccessful, 4.2.1 The system declares that the invoice is invalid and returns a response message. 4.2.2 The taxpayer then has to perform modifications and resubmit a new digitally signed invoice. 4.2.3 Continue to flow 3.
Alternate/Exceptional Flows: A. The digitally signed invoice must only be in either XML of JSON file format.

<table>
<tr><td>Use Case Name: Cancel Submitted Invoice</td><td>ID:
FR006</td><td>Importance
Level: High</td></tr>
<tr><td colspan="2">Primary Actor: Taxpayer</td><td>Use Case Type: Detail, Real</td></tr>
<tr><td colspan="3">Stakeholders and Interests:
Taxpayer: wants to cancel a previously submitted invoice.
LHDN: wants a reason for the cancellation of the invoice for logging and documentation purposes.</td></tr>
<tr><td colspan="3">Brief Description: This use case describes the process of how a taxpayer cancels a previously submitted invoice.</td></tr>
<tr><td colspan="3">Trigger: The taxpayer clicks on the “Cancel Invoice” button. This cancellation process can only be done within 72 hours of submission.</td></tr>
<tr><td colspan="3">Relationships:
Association : Taxpayer
Include : N/A
Extend : N/A
Generalization : N/A</td></tr>
<tr><td colspan="3">Normal Flow of Events:
1. The system outputs an overview of the list of previously submitted invoices.
2. The taxpayer views a list of previously submitted invoices in their personal invoices database.
3. The taxpayer clicks on the “Cancel Invoice” on a selected e-invoice.
4. The taxpayer then states their reason for cancellation.
5. The system then retrieves the uuid stored in the session and the reason for cancellation.
6. The system submits the uuid and reason for cancellation to the Cancel Document API.
7. The system then returns a response message. Perform 7.1 or 7.2.
8. The system outputs the successful status message received from the cancellation response.
9. The system then cancels the invoice.</td></tr>
</table>

Sub-flows: 7.1 If the response message returns an IncorrectState or Forbidden, 7.1.1 The system declares that the invoice cancellation process is invalid by returning a response message. 7.1.2 Continue with flow 5. 7.2 If it returns a successful response message, 7.2.1 Continue to flow 8.
Alternate/Exceptional Flows: A. The cancellation must be made within 72 hours of submitting the invoice.

Use Case Name: Receive Notifications	ID: FR007	Importance Level: High
Primary Actor: Taxpayer	Use Case Type: Detail, Real	

Stakeholders and Interests:

Taxpayer: needs timely updates and immediate information about invoicing activities.

LHDN: needs to inform taxpayers of important invoice statuses and the processing of e-invoicing activities.

Brief Description: This use case describes the process of how taxpayers receive system notifications via the Get Notifications API.

Trigger: The taxpayer is interested in receiving notifications on any e-invoicing activities that happen within the web application.

Relationships:

Association	: Taxpayer, LHDN
Include	: N/A
Extend	: N/A
Generalization	: N/A

Normal Flow of Events:

1. The system authenticates the taxpayer before initiating the request.
2. The system initiates an API call to the Get Notifications API.
3. The system retrieves the notification history of the particular taxpayer. *Perform 3.1 or 3.2.*
4. The system then outputs the particular notification type and its details based on the taxpayer's location in the web application.

Sub-flows:

3.1 If there are no notifications stored in the notification history,

3.1.1 The notification history presents an empty array of notification objects.

3.2 If there are notifications stored in the notification history,

3.2.1 The notification history presents an array of notification objects and its metadata.

3.2.2 Continue with flow no.4.

Alternate/Exceptional Flows: 1a. The taxpayer must be an authenticated user before proceeding to retrieve notification history.

Use Case Name: View Ledger	ID: FR008	Importance Level: High
Primary Actor: Taxpayer	Use Case Type: Detail, Real	
Stakeholders and Interests: Taxpayer: is interested in viewing the list of invoicing activities.		
Brief Description: This use case describes the process of viewing the overview of invoicing activities in the form of a ledger.		
Trigger: The taxpayer clicks on the "Ledger" hyperlink at the header.		
Relationships: Association : Taxpayer Include : N/A Extend : N/A Generalization : N/A		
Normal Flow of Events: 1. The taxpayer selects the "Ledger" at the top of the navigation menu. 2. The system acquires the particular taxpayer's e-invoicing history. *Perform 2.1 or 2.2.* 3. The system presents a table, which provides an overview of all the e-invoices that have been issued or received. 4. The taxpayer can then export the ledger as a PDF report.		
Sub-flows: 2.1 If there are no invoice information stored in the ledger, 2.1.1 The ledger presents an empty result. 2.2 If there are information stored in the ledger, 2.2.1 Continue with flow no.3.		
Alternate/Exceptional Flows: -		

Use Case Name: Generate QR code	ID: FR010	Importance Level: High
Primary Actor: Taxpayer	Use Case Type: Detail, Real	
Stakeholders and Interests: Taxpayer: is interested in generating a QR code for the particular invoice.		
Brief Description: This use case describes the process of generating a QR code that is linked to the invoice in order to verify the existence of the e-invoice.		
Trigger: The taxpayer successfully validates the invoice.		
Relationships: Association : Taxpayer Include : Validate Invoice Extend : N/A Generalization : N/A		
Normal Flow of Events: 1. The taxpayer validates the submission of the invoice. 2. The system then retrieves the envBaseUrl, uuid and longId values from the response of successful submission. 3. The system constructs a validation link for the QR code. 4. The system generates the QR code using the Endroid QR code library using the validation link. 5. The system then encodes the QR code string to base 64. 6. The system outputs the QR code that was previously encoded to the form. 7. The taxpayer can view and use the QR code.		
Sub-flows:		
Alternate/Exceptional Flows: - 2a The system must have access to data such as envBaseUrl, uuid and longId as well as the Endroid QR code library in order to generate the QR code.		

4.3 Low-Fidelity Prototype Interface

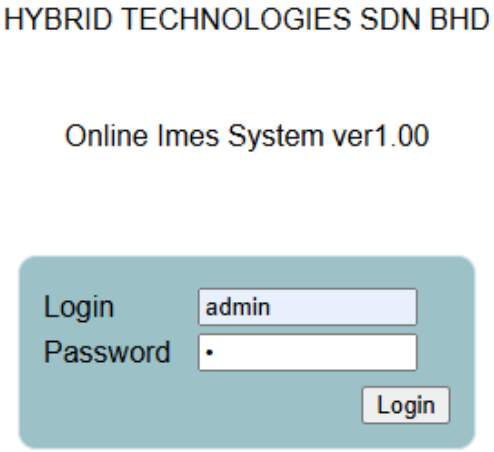


Figure 4.2 User Login

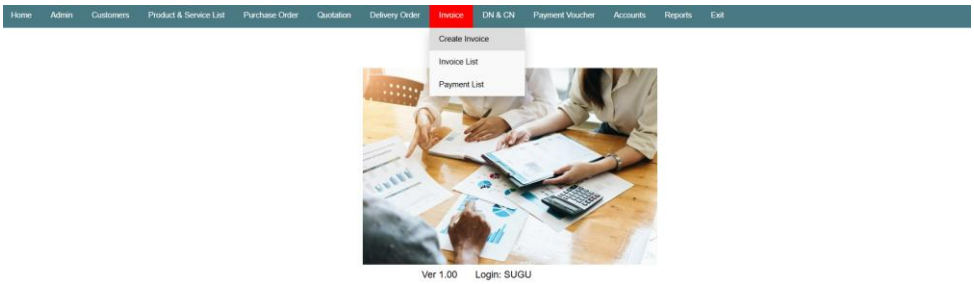


Figure 4.3 Main Page of the E-Invoicing System

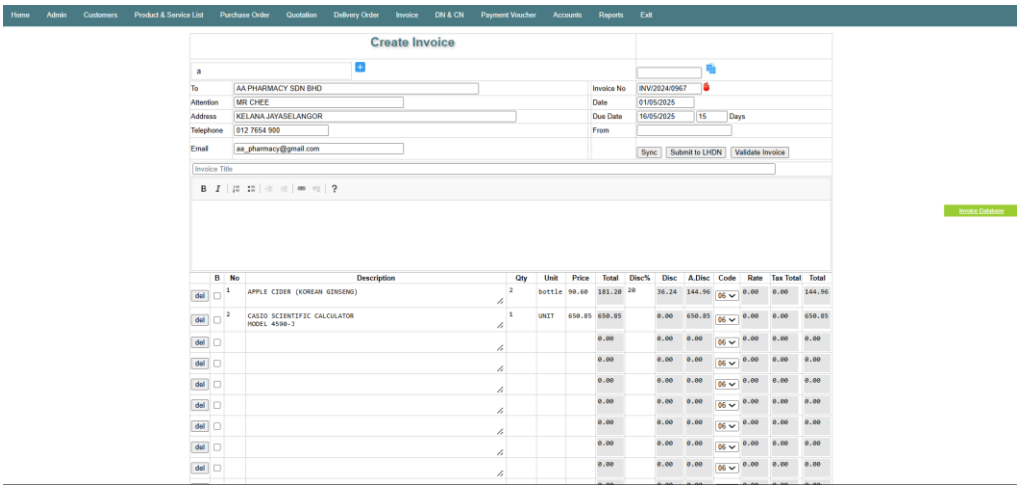


Figure 4.4 Generate, Submit, Validate Invoice

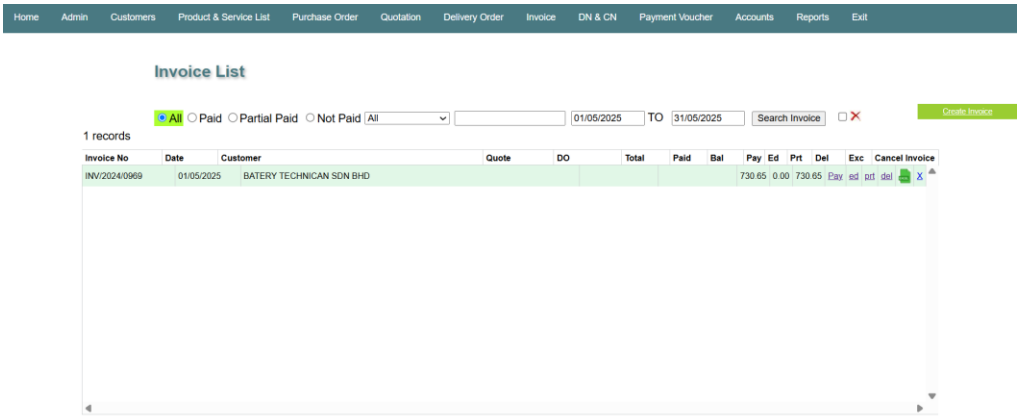


Figure 4.5 Cancel Invoice

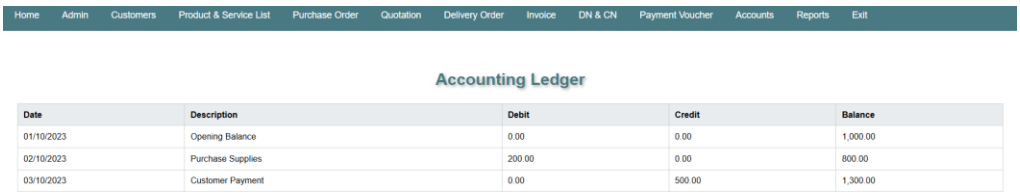


Figure 4.6 Ledger

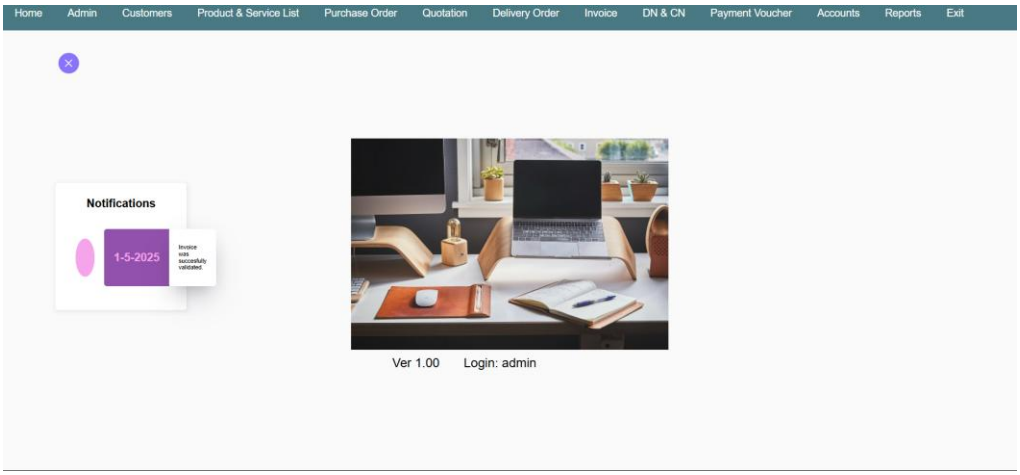


Figure 4.7 Notification Feature

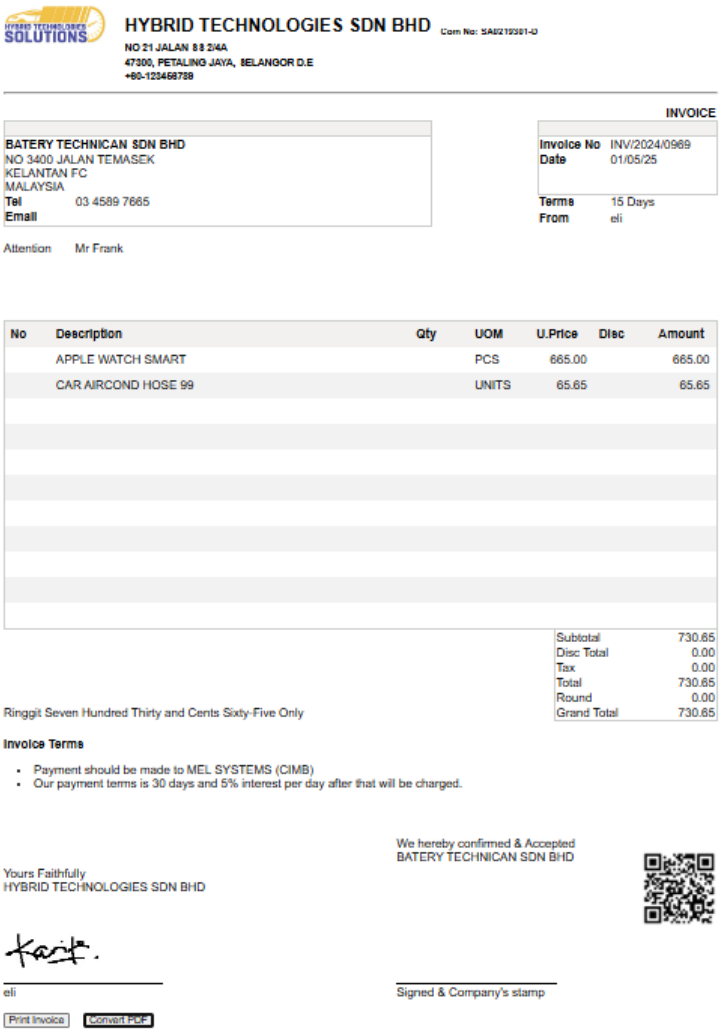


Figure 4.8 E-Invoice with QR Code Attached

4.4 Summary

In essence, this chapter had summarized and analyzed the requirements gathering findings. With that, the data collected supported the problem statement and features of the e-invoicing web application. Besides that, this chapter also highlighted the User Requirements Specifications in order to get the overall and high-level requirements that the stakeholders expect of the system. Besides, Functional Requirements and Non-Functional Requirements were also listed in detail and evaluated. The requirements findings have then contributed to the creation of the Use Case Diagram, Use Case Descriptions, as well as the Low Fidelity Prototypes to fully provide detailed insights and functionalities of the e-invoicing web applications.

CHAPTER 5

SYSTEM DESIGN

5.1 Introduction

This chapter mainly discusses the system architecture design by using diagrams such as Entity Relationship Diagram, Analysis Class Diagram and Data Flow Diagram, which includes the Context Diagram to visualize the system from a high-level standpoint. Moreover, there is a separate section which shows the user interface design of the system.

5.2 System Architecture Design

The entire interaction of the system architecture components from a high-level standpoint can be split into a three-tier architecture comprising a Web User Interface (UI), a Database Layer and an Application Server. These tiers are further supported by external components such as digital signing, authentication, and QR code generation library modules, which are essential as part of LHDN's requirements. Furthermore, deployment and file management are supported by tools such as DirectAdmin and FileZilla for smooth transfer and web hosting of the system. The overall architecture is built to enable invoices to be generated, digitally signed, transmitted to the LHDN e-Invois API, and subsequently retrieved for verification and record-keeping.

Firstly, the Web User Interface acts as the end-user interface and is an access point between the system and its end users. Furthermore, the web user interface is delivered through a standard web browser and was developed using HTML, CSS and JavaScript to provide forms, tables and printable views that are responsive and intuitive. Besides that, the user interface enables users to create, edit, review, print QR codes that are embedded in each of the e-invoices, and passes all input data to the backend for further data processing. This layer primarily concentrates on making sure user input is captured accurately and passed to the underlying logic of the backend for further processing.

Secondly, the second layer, which is the Database Layer or synonymously known as the persistence layer, is implemented using MySQL relational database. It serves as the central repository for business records, primarily used to manage all system data by enabling storing, easy retrieval, reporting and auditing of all invoice data, including information such as header information, line items, and customer details, in a structured and reliable manner. With this, data that are stored in tables are often frequently updated and fetched many times by different scripts on the application, thereby ensuring the workflow proceeds smoothly all the time.

On the other hand, the third layer, which is the Application Server, is the central part of the system and represents the backend logic of the system that governs the functionality of e-invoicing. Within this layer, PHP scripts are used as controllers and mappers that manage the construction of JWT authentication tokens using OAuth2 protocol, invoice payloads, creation of a digital signature and invoking the LHDN MyInvois APIs. Furthermore, cryptographic signing is also performed using the SHA-256 hashing algorithm along with a private key stored in a PKCS#12 certificate container to ensure authentication and non-repudiation of all invoices sent. Therefore, this layer thereby ensures the backend processing logic and ensures that submissions are technically accurate as well as compliant with LHDN's regulation standards.

As for deployment and operational management, DirectAdmin and FileZilla are responsible for handling these tasks. DirectAdmin serves as the control panel for cloud hosting, where the PHP application resides. It provides services such as domain configuration and MySQL database hosting. This tool ensures that the web application being deployed on the cloud is constantly available and in a state of stability. On the other hand, FileZilla is used during the development and deployment phase for the transfer of source code and configuration files from the local development environment to the cloud server. It is used to perform secure FTP transfers, ensuring development updates are synced appropriately with the hosted environment.

5.3 Analysis Class Diagram

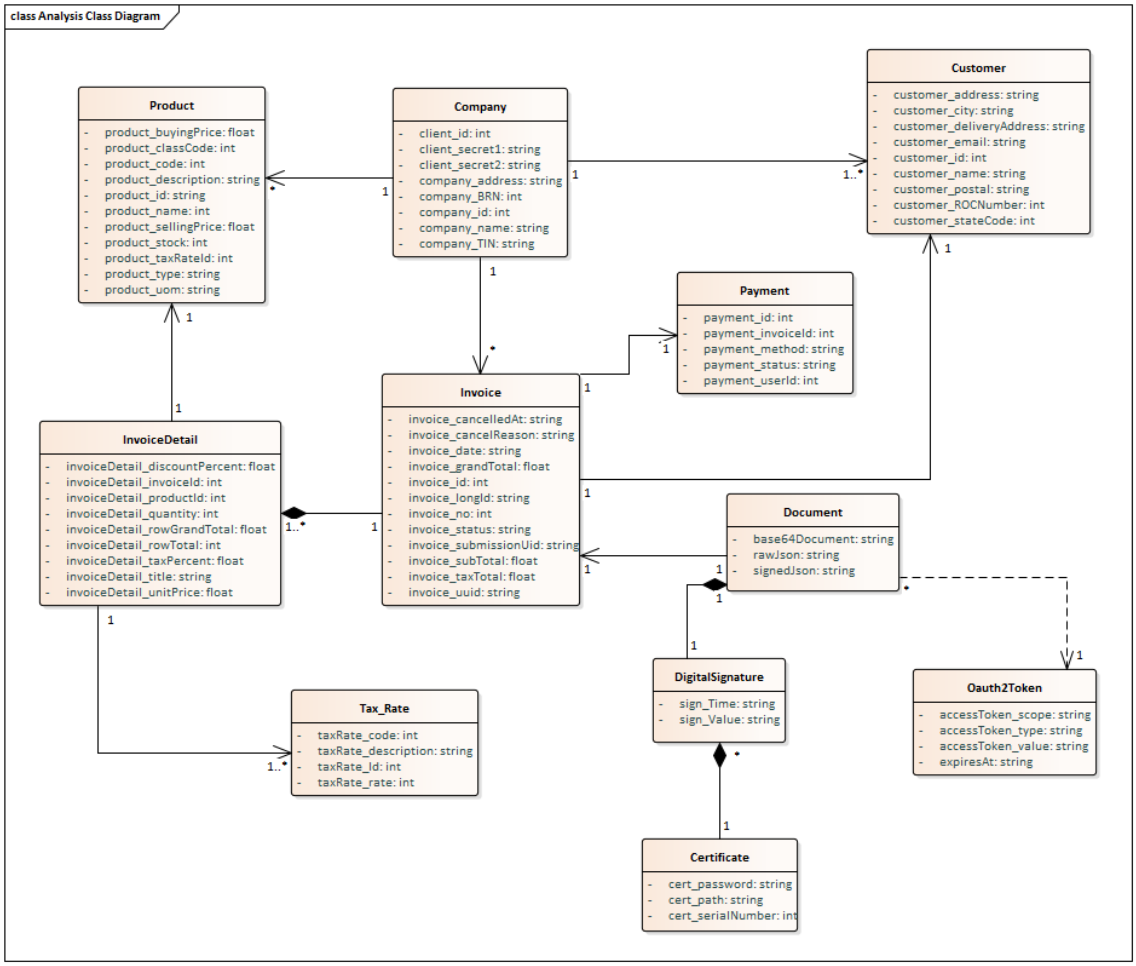


Figure 5.1 Analysis Class Diagram

The analysis class diagram above illustrates a conceptual representation of the system's main classes, attributes, and relationships that define how data typically flows between components. Besides, it also serves as a foundation for understanding how various entities within the invoicing process interact to meet the system's functional requirements.

Located in the center of the diagram is the Invoice class, which is an entity that aggregates fields such as invoice IDs, total charges, status, and timestamps of creation, submission and cancellation. Each invoice is associated with one Company and one Customer. As for the type of

relationship between classes, there is the one-to-many relationship of Invoice to Company that implies that a single company can issue multiple invoices, and the one-to-many relationship of Invoice to Customer, signifying that a single customer can receive numerous invoices over time.

Next, each invoice is further detailed by one or more InvoiceDetail objects, which define the individual items or services billed in the invoice. This relationship is a composition of one-to-many, whereby an invoice cannot be meaningfully described without its details, and when an invoice is deleted, its corresponding details are deleted as well. Furthermore, the InvoiceDetail class contains information such as the product description, quantity, unit price, discount, and tax amounts. Besides that, the composition relationship has high dependency as it emphasizes invoice details as part of an invoice.

Moreover, the InvoiceDetail class also bears a relationship to the Product Class through an association, indicating that every invoice detail is a reference to a particular product being sold. This reference allows the system to retrieve standardized product information like code, description, and price directly from the database. Similarly, the Tax_Rate class is closely related to the InvoiceDetail class through a one-to-many association because several items are subjected to various tax rates.

Besides that, the Payment class has a one-to-one association with the Invoice class, indicating that every Invoice will have a corresponding payment record. This association mainly contains payment details like method, status, and reference IDs to allow the system to monitor and provide clarity on whether the invoice has been settled or remains outstanding.

To allow for digital document handling, the Document class is linked to the Invoice class, which is the electronic invoice sent to the tax authority. Every document stores the encoded and signed JSON properties of the invoice, such as the base64Document, rawJson, and signedJson. The Document class is connected to the DigitalSignature class through a composition relationship,

indicating that each of the document must have one related digital signature as part of the validation process, thereby guaranteeing data authenticity and non-repudiation. Besides that, the Certificate class is closely related to the DigitalSignature class, as the DigitalSignature class serves to represent the cryptographic credentials utilized in the digital signing process. This composition relationship thereby ensures secure and verifiable digital transactions based on LHDN's e-invoicing standards.

Lastly, the Oauth2Token class is associated with the Document class through a dependency relationship, as the system requires authentication tokens in order to securely transmit documents to LHDN's protected APIs. The authentication token's main purpose is to cache values such as scope, type, and expiry time to provide authorized access between the system and LHDN's external API endpoints.

5.4 Entity Relationship Diagram

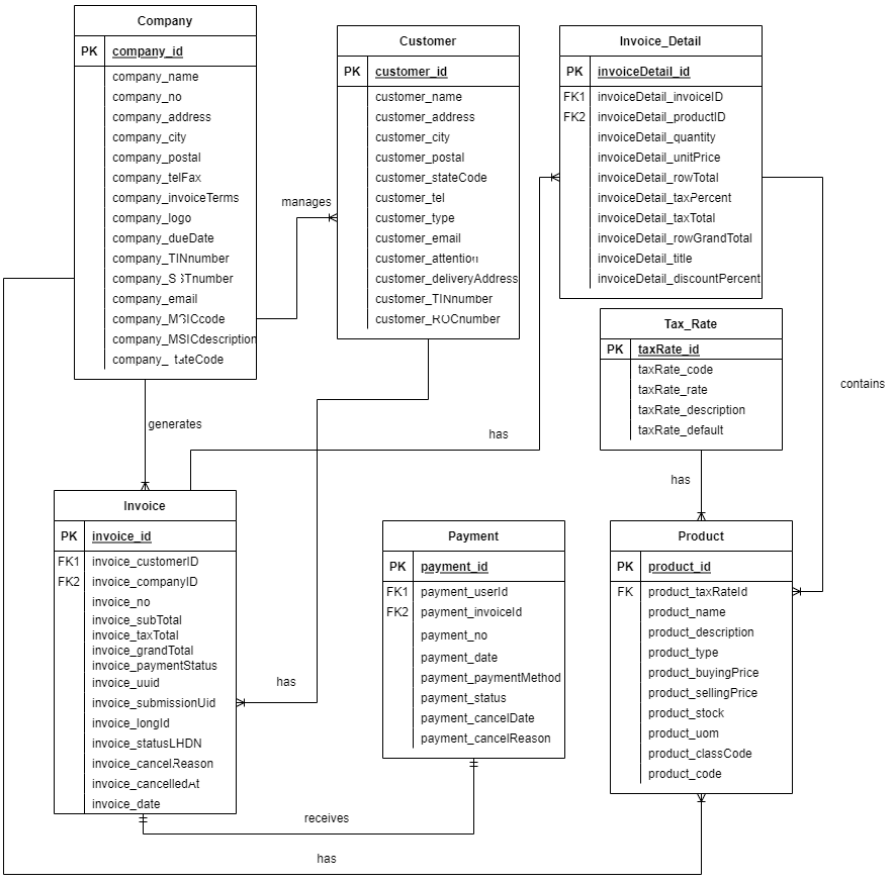


Figure 5.2 Entity Relationship Diagram

The Entity Relationship Diagram above demonstrates the logical structure of the system's database. It primarily describes how data is stored and related to one another through tables, fields and relationships. In this system, the company entity, which is also known as the taxpayer or supplier, is responsible for issuing the invoices. There can be numerous customers per company, who are the buyers or recipients of goods and services. The relationship between the company entity and the customer entity mainly stems from the one-to-many relationship. This indicates that one company can have many customers, but one customer can be associated with only one company entity. Similarly, a company entity may also have a range of products that it retails to its clients or customers. With this relationship, at least one product relates to one company, though a company may have many products in its catalogue.

Besides, a company entity is also responsible for the issuance of multiple invoices over time. An invoice records a transaction, such as a financial transaction between a company and its customer. This also establishes a one-to-many relationship as each customer is able to receive multiple invoices from the company.

In every invoice, there can be numerous invoice details that are individual line items, like the sold products, quantities, unit prices, discounts or even taxes. A product can be in numerous invoice line items, making the Product and Invoice Detail entities to have a one-to-many relationship. Similarly, the tax rates are linked to the invoice line items in order to ensure proper and accurate calculation of tax. Thus, having a one-to-many relationship is also present here.

Last but not least, payments are also tracked by the system through the payment entity. This entity has a one-to-one relationship with the invoice entity as the payment is only issued once for every invoice.

5.5 Context Diagram

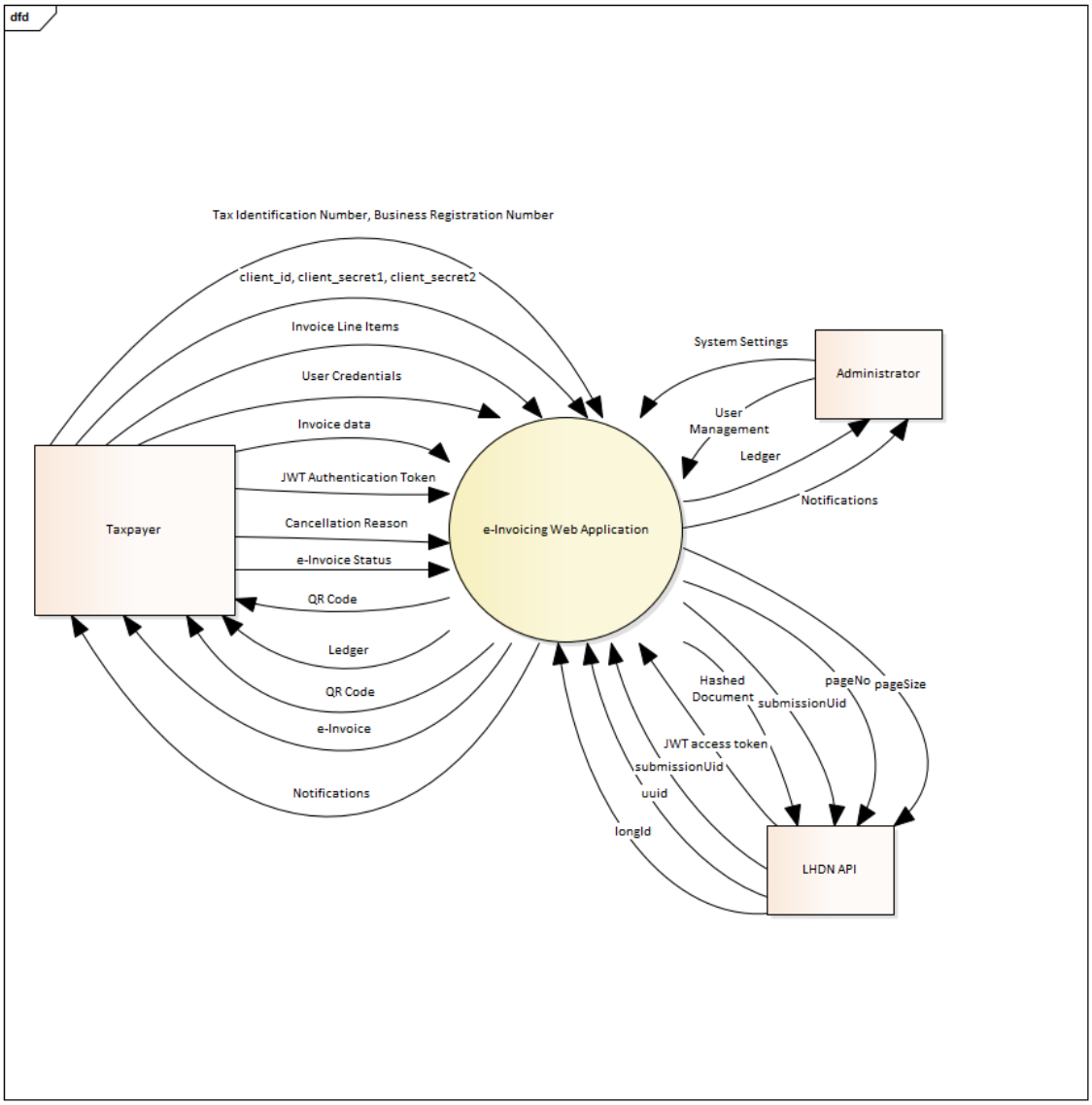


Figure 5.3 Context Diagram

The diagram above illustrates the Context Diagram, which represents the highest-level view of the Data Flow Diagram. This diagram, in particular, serves to provide a conceptual overview of the logical flow of information that involves major external entities interacting with the e-invoicing system and the types of data that are being passed among them. The purpose of this context diagram is mainly to show the system boundaries, where information entering and leaving the system is shown, without detailing the internal data processes.

In the middle of the diagram is the e-Invoicing Web Application, which is represented as a single process that encapsulates all core functionalities such as user authentication, invoice generation and submission, and external data communication with external APIs. Surrounding the entity are three important core external entities that are interacting with the web application, which are the Taxpayer, Administrator and the LHDN APIs. These external entities perform its own distinctive role in the e-Invoicing system.

The taxpayer is the registered business organization, also known as the end-user, that works with the system to produce, maintain and send e-invoices. The inputs of data between the taxpayer and the e-Invoicing Web Application are the invoice details, line items, tax identification numbers, such as Tax Identification Number and Business Registration Numbers, user credentials, and cancellation reasons. The system, in return, provides outputs of the status of the e-Invoice, QR code, ledgers and notifications. This two-way exchange of information facilitates taxpayers to complete performing invoicing processes while simultaneously ensuring adherence to data and tax regulations.

On the other hand, the administrator entity interacts with the system, mainly in managing user accounts, configuring options, and overseeing ledger and notification activities. The administrator entity also involves in sending system management inputs, while the system responds with updates and reports to assist in the monitoring process. Hence, this ensures a secure and controlled use of the application.

Besides that, the LHDN API entity represents the external service of LHDN. The e-Invoicing Web Application communicates with the APIs for invoice data authentication, document submission, and verification of invoice data. The system will then send the hashed document payloads, submission UIDs, JWT access tokens, and other related identifiers to the LHDN APIs. As a result, the APIs will then respond with validation responses, official

validation responses, and identifiers involved in QR code creation. Such interaction guarantees that all e-invoices adhere to the national tax regulations and are recorded in the LHDN system.

5.6 Design of the User Interface (UI)

This section contains the user interface design and the overall flow when using the web application.

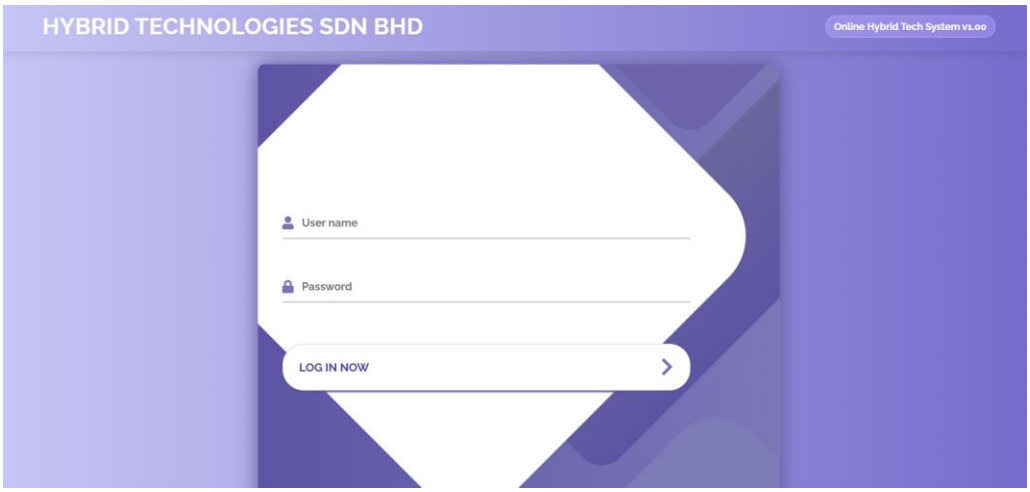


Figure 5.4 Login Page

The picture above shows the login page, which is the system's authentication interface, which only allows registered users to securely access the e-Invoicing system. These users will input their username and password, which are then further validated against stored credentials in the database. Successful authentication makes the system generate a session token and redirects the user to the home page. If there are invalid credentials being inputted, it will prompt an error message asking the user to try again.

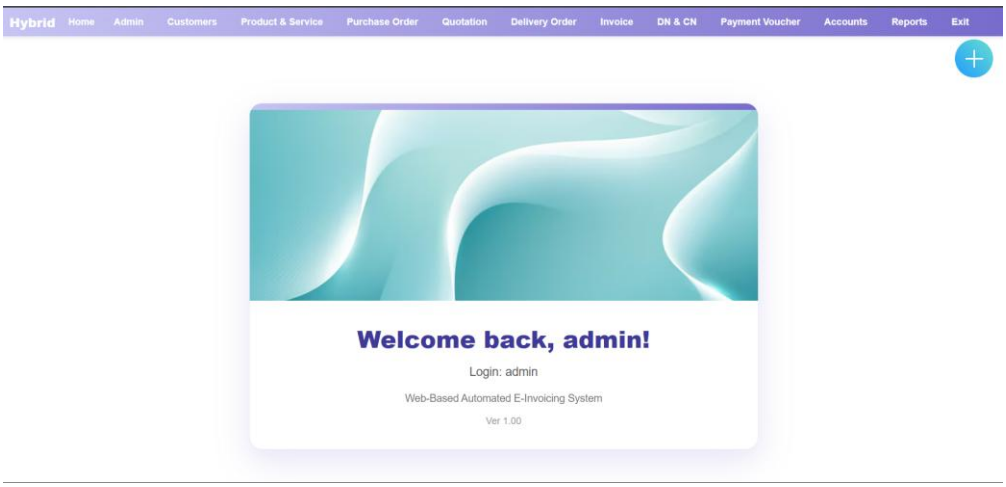


Figure 5.5 Home Page

The image above shows the home page of the system, which is responsible for displaying a welcome message to the user. Furthermore, the home page also displays all the notifications related to the e-invoicing activities that are being performed in the web application.

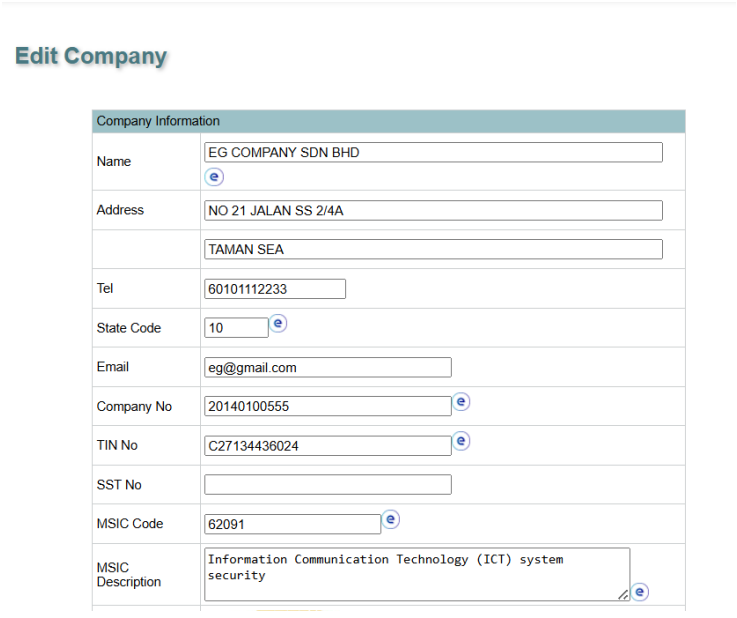


Figure 5.6 Edit Company Details

The image shown above is a page that serves to give users the flexibility to edit their company details. For instance, information such as company name, business registration number, tax identification number, company address and contact information can be edited if there are changes to the company's

credentials. Once edited, the new information will then be updated to the company database table. Hence, this page serves to ensure that the e-invoices generated contain reliable and up-to-date information about the company, thereby ensuring data accuracy.

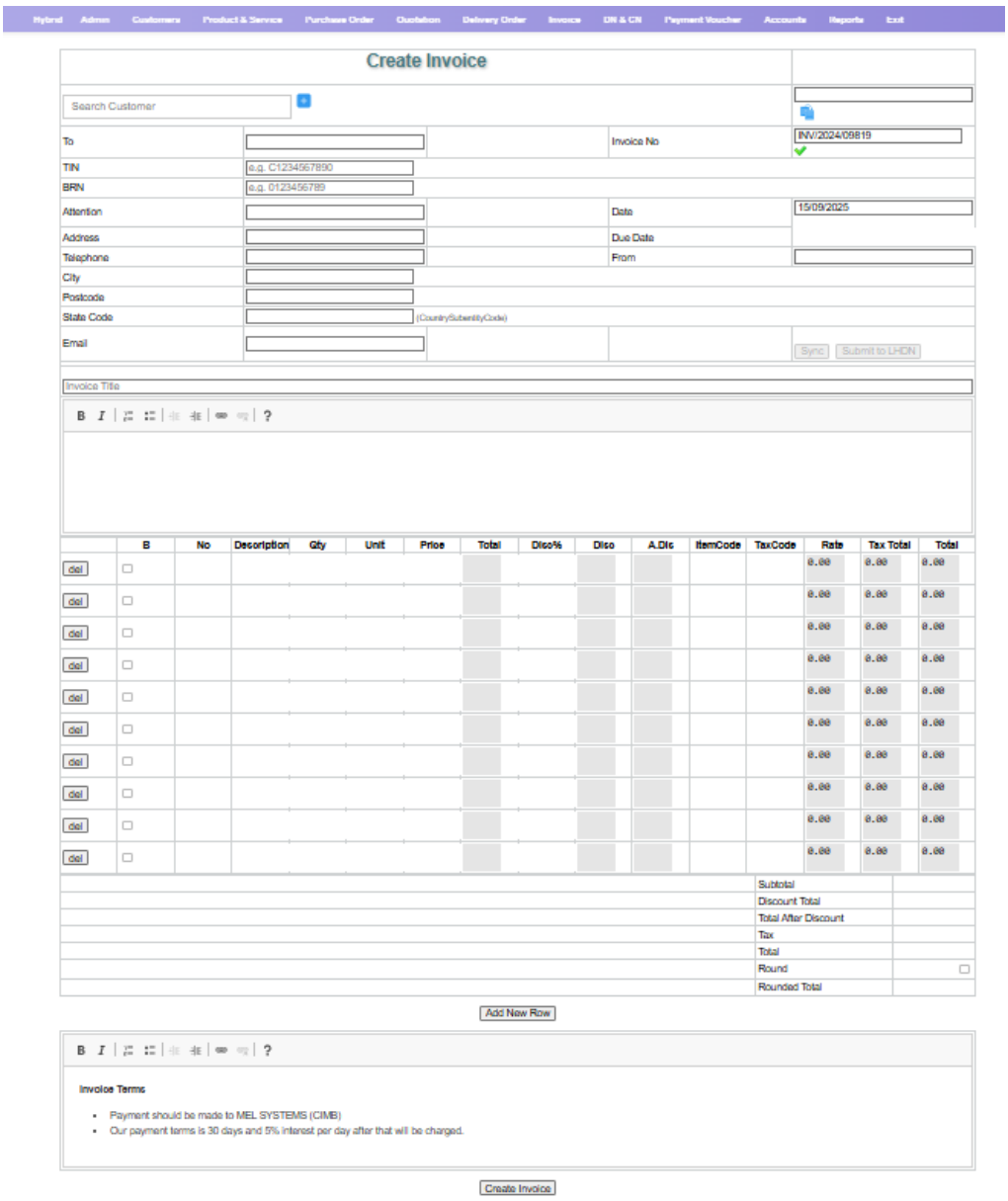


Figure 5.7 Create Invoice Page

The Create Invoice page in particular enables users to generate new invoices or e-invoices by entering relevant details into the text fields, such as customer information, product line Items, quantity, unit price, and tax rate information.

The subtotals, tax amounts and grand totals are automatically computed by the system prior to approving submission. On completion, the invoice can be either saved as a regular invoice or further issued to LHDN's APIs, such as the Submit Documents API and Get Submissions API for validation.

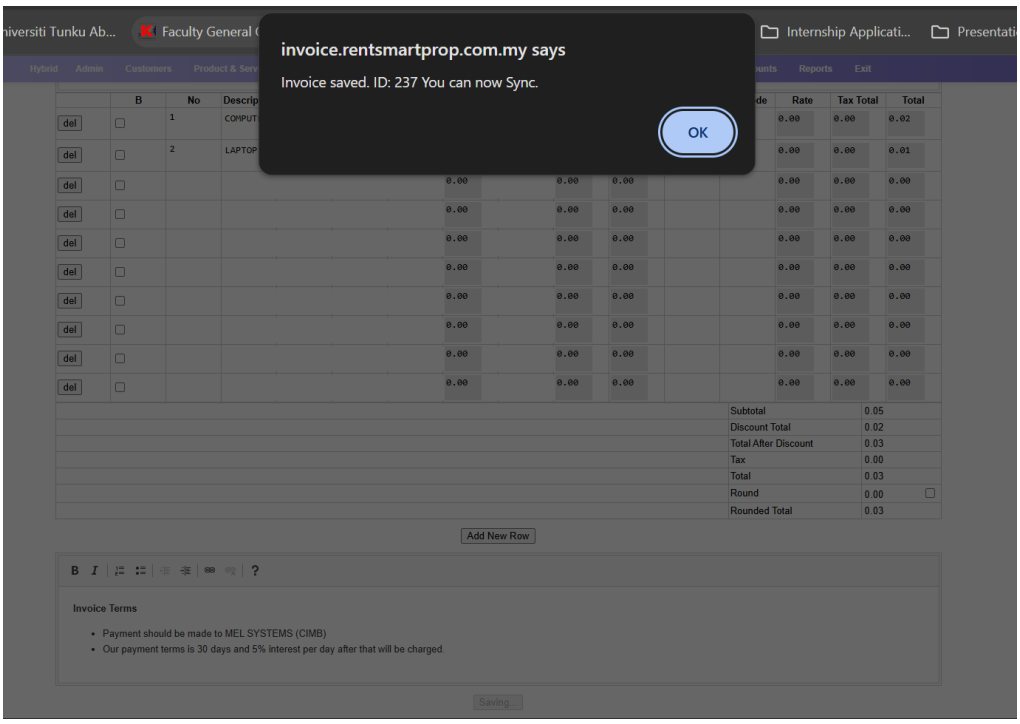


Figure 5.8 Creating Invoice

The image above shows the success alert after successfully saving the regular invoice to the database. The success message further details the next instruction, which is to sync by mapping the data stored in the text fields into the JSON document. This syncing process is only performed if the user intends to generate an e-invoice.

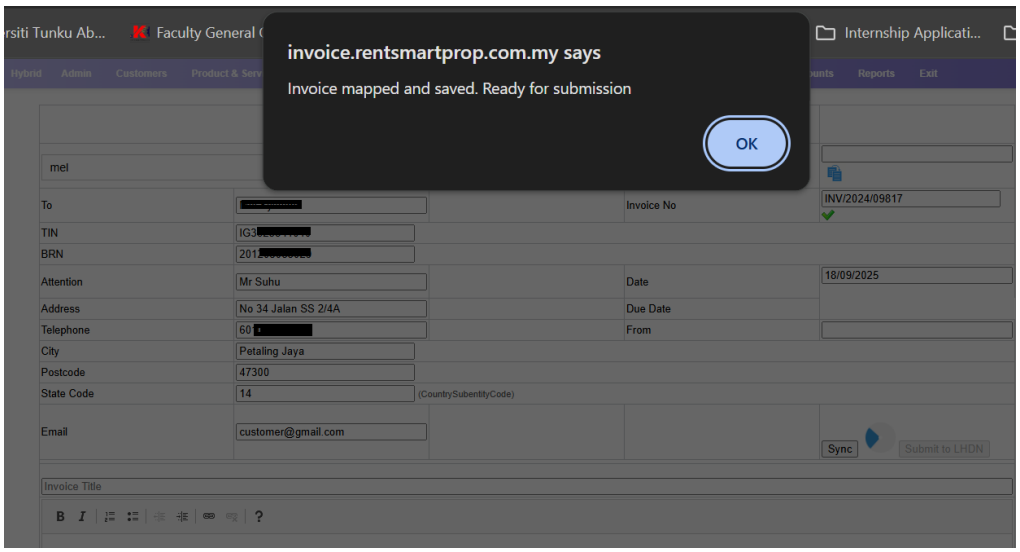


Figure 5.9 Invoice Mapped to UBL 2.1 JSON format

The image above shows the success alert that is shown when the user successfully maps the data stored from the text fields to the JSON document in UBL 2.1 format. The message further details the next instruction, which is to submit the mapped JSON document to LHDN's APIs.

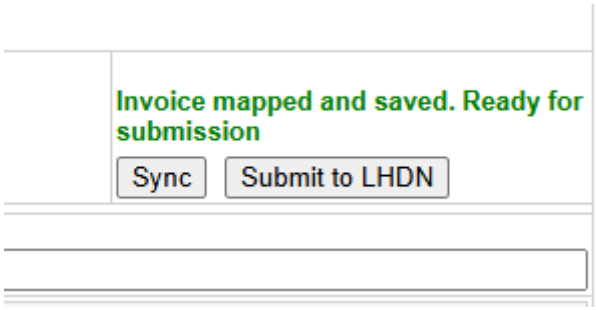


Figure 5.10 Invoice Mapping Success Message

The image above shows the success message after successful mapping of the data from the text field to the JSON document in UBL 2.1 format.

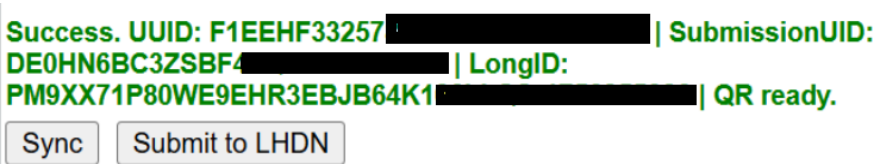
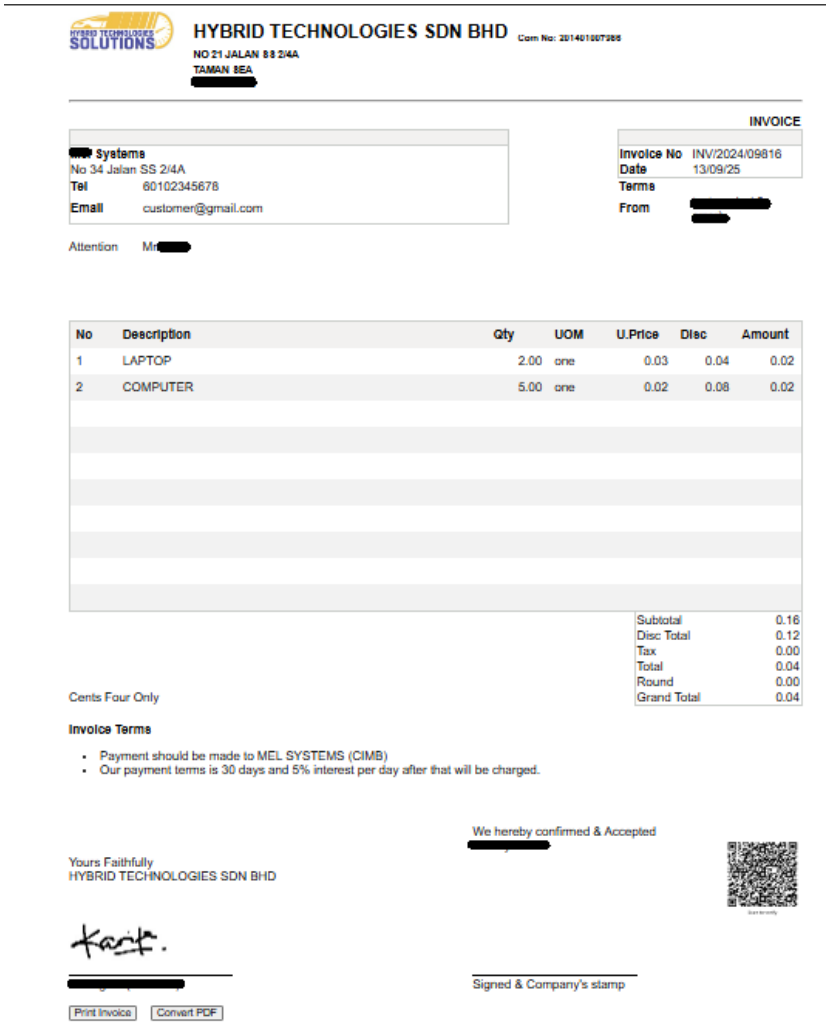


Figure 5.11 Successful API call to Submit Documents API

The image above illustrates the success message after successfully sending the invoice for further validation. This process is required as part of the requirements to generate e-invoices. The validation process includes two API calls to the Submit Documents API and the Get Submissions API. Upon successful validation, the response message will include the UUID and the SubmissionUID received from the Submit Documents API. The UUID or SubmissionsUID is then used as an input to call upon the Get Submissions API, in order to retrieve the longId, which is a fundamental data used in generating the QR code of the e-invoice.



HYBRID TECHNOLOGIES SDN BHD

NO 21 JALAN 88 2/4A

TAMAN SEA

INVOICE

[redacted] Systems
No 34 Jalan SS 2/4A
Tel 60102345678
Email customer@gmail.com

Invoice No INV/2024/09816
Date 13/09/25
Terms
From [redacted]

Attention Mr[redacted]

No	Description	Qty	UOM	U.Price	Disc	Amount
1	LAPTOP	2.00	one	0.03	0.04	0.02
2	COMPUTER	5.00	one	0.02	0.08	0.02

Subtotal	0.16
Disc Total	0.12
Tax	0.00
Total	0.04
Round	0.00
Grand Total	0.04

Cents Four Only

Invoice Terms

- Payment should be made to MEL SYSTEMS (CIMB)
- Our payment terms is 30 days and 5% interest per day after that will be charged.

We hereby confirmed & Accepted

Yours Faithfully
HYBRID TECHNOLOGIES SDN BHD

Signed & Company's stamp

Print Invoice Convert PDF

Figure 5.12 E-Invoice with Embedded QR Code

The image above shows the successful outcome of generating an e-invoice. The QR code embedded into the e-invoice serves as a unique identifier for the

e-invoice and verifies that the financial transaction has been successfully validated and recorded by LHDN. Upon scanning the QR code, it will redirect the user to the official record of the invoice stored in the LHDN's database. Thus, guaranteeing authentication and compliance.

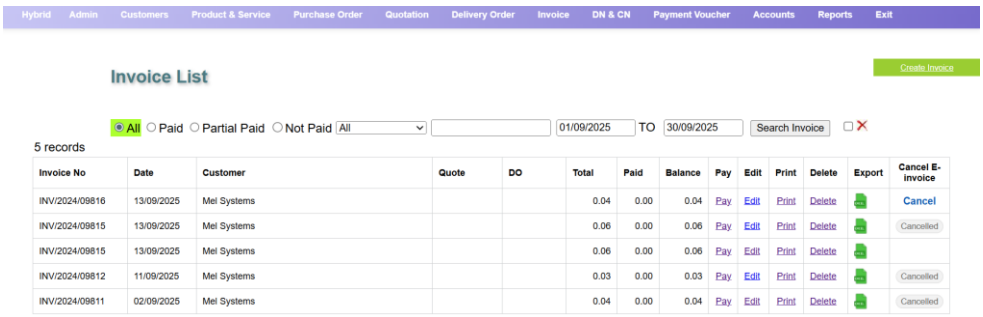


Figure 5.13 Invoice List Page for Cancellation of E-Invoices

The image above displays all previously created invoices and e-invoices, including their current cancellation status. In this page, user can press the "Cancel" button to proceed with their e-invoice cancellation.

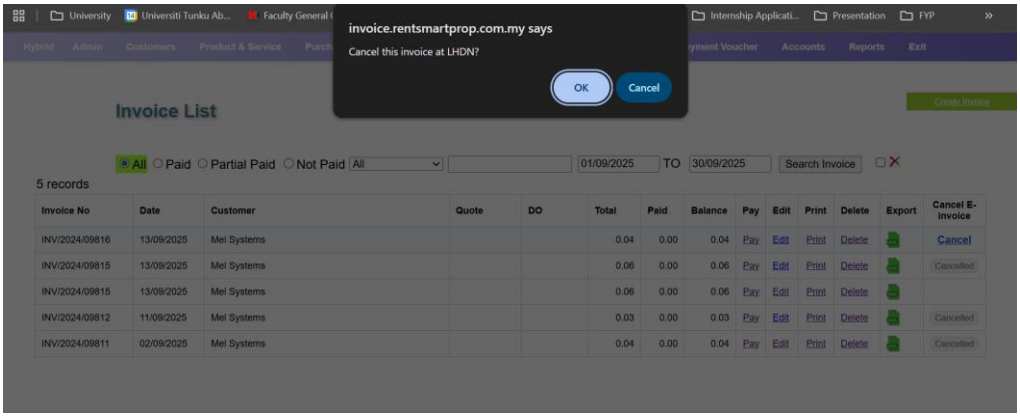


Figure 5.14 Submitting API call to Cancel Documents API

The image above shows the alert message, which prompts the user to proceed or not to proceed with the cancellation. Upon clicking on "Ok", the cancellation process will proceed.

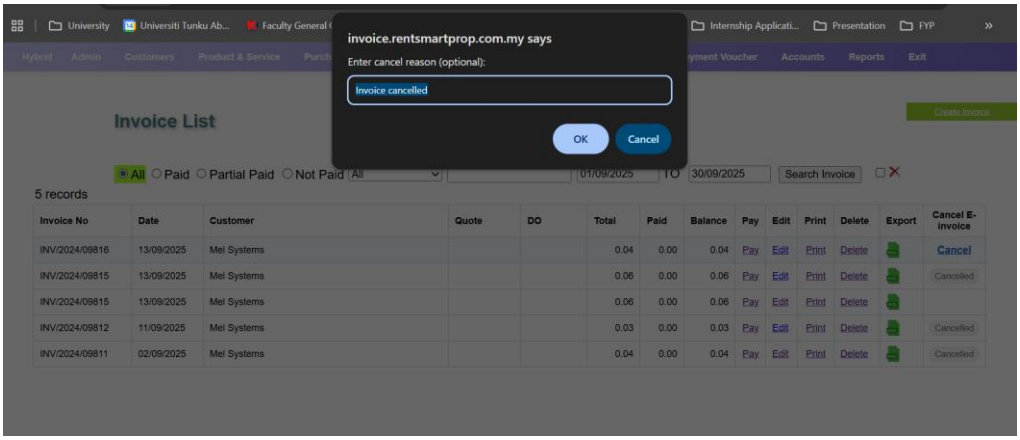


Figure 5.15 Cancellation Reason Prompt

The image above shows the alert for the cancellation reason of the particular e-invoice. The cancellation reason will then be used as an input to the Cancel Documents API.

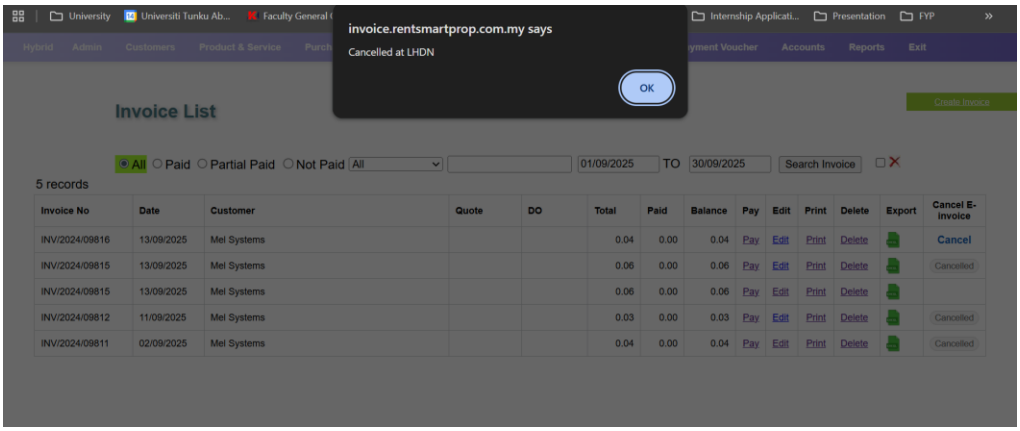


Figure 5.16 Successful Cancellation Message

The image above demonstrates the successful alert displayed upon cancellation of an e-invoice. The alert message further details that the e-invoice has been cancelled at LHDN's cloud.

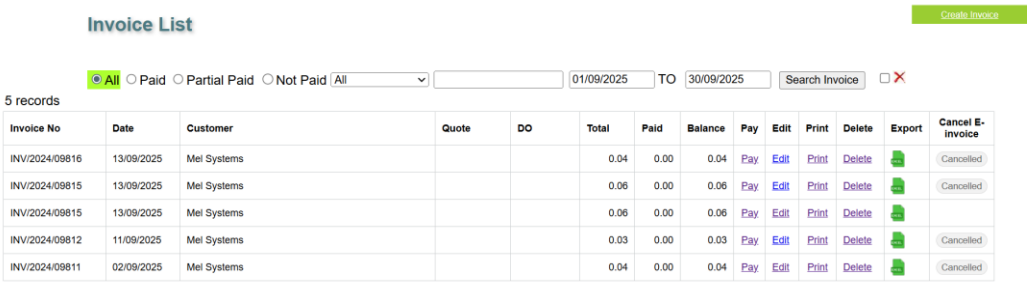


Figure 5.17 Updated "Cancelled" Status of E-Invoice

The image above shows that once an e-invoice has been cancelled, the system will automatically reflect the change by updating the status of the e-invoice cancellation state in the invoice list. Furthermore, the corresponding record in the database is also updated.

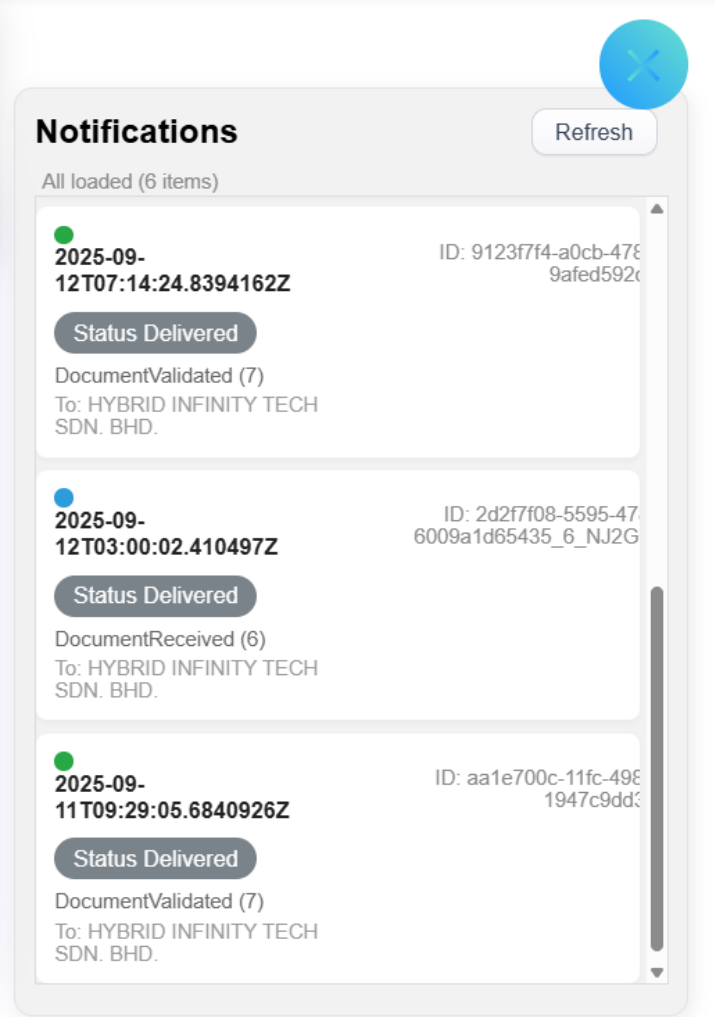


Figure 5.18 Notifications Feature via invoking Get Notifications API

The image above shows the notification feature of the system. This feature provides real-time alerts about e-invoicing activities being performed that were retrieved from the Get Notifications API.

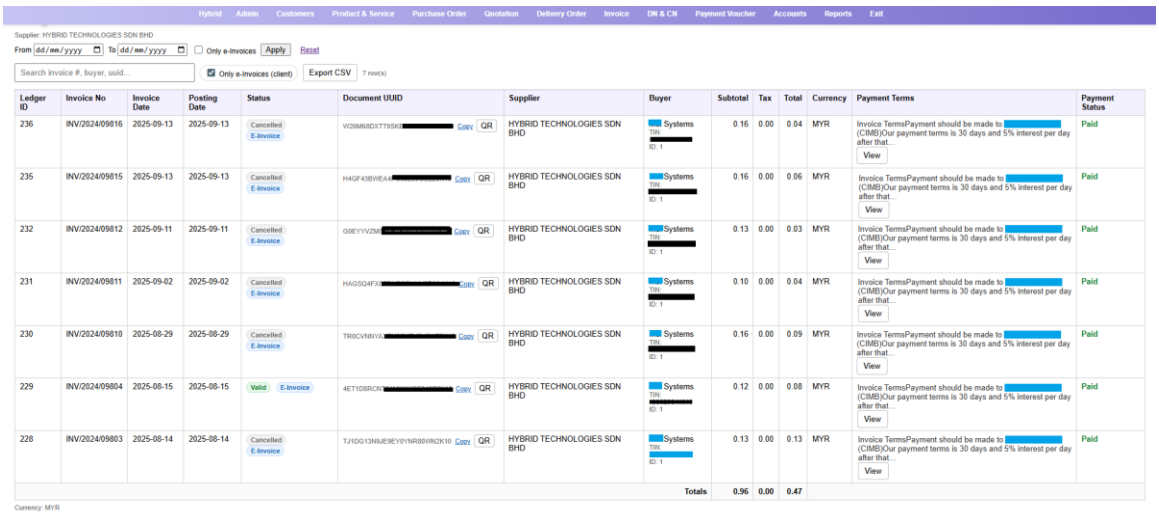


Figure 5.19 Ledger Feature for Invoice and E-Invoice Tracking

The image above shows the ledger page, which provides a summarized financial data which were derived from submitted invoices and e-invoices. This includes total taxable amounts, total tax collected and grand totals across specific periods. Furthermore, the ledger feature provides additional features which allow users to filter results by date or type of customer, as well as export reports in Excel format.

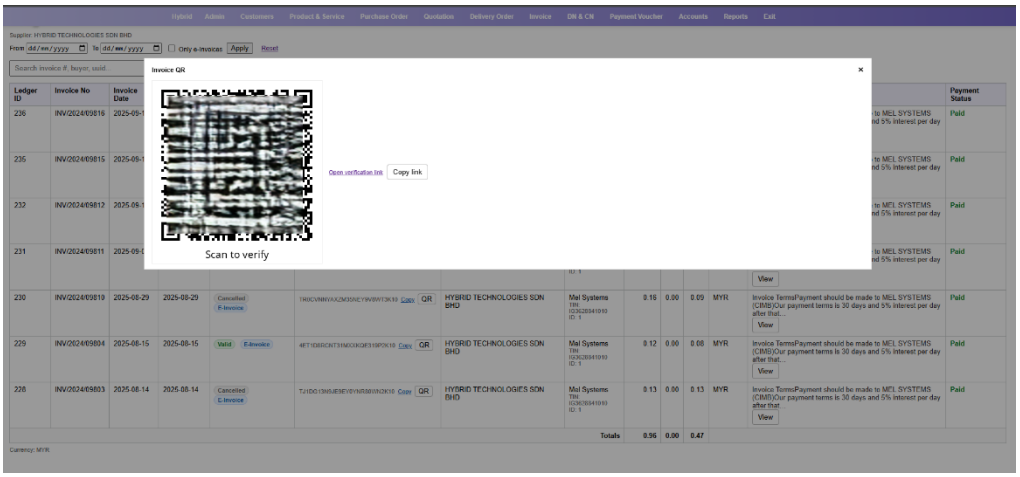


Figure 5.20 View QR Code through Ledger Modal

The image above further showcases the extension of the ledger page. Furthermore, the ledger feature can also access the QR code of the specific e-invoice. This provides the system with additional flexibility, thereby reducing the time taken for users to access stored records.

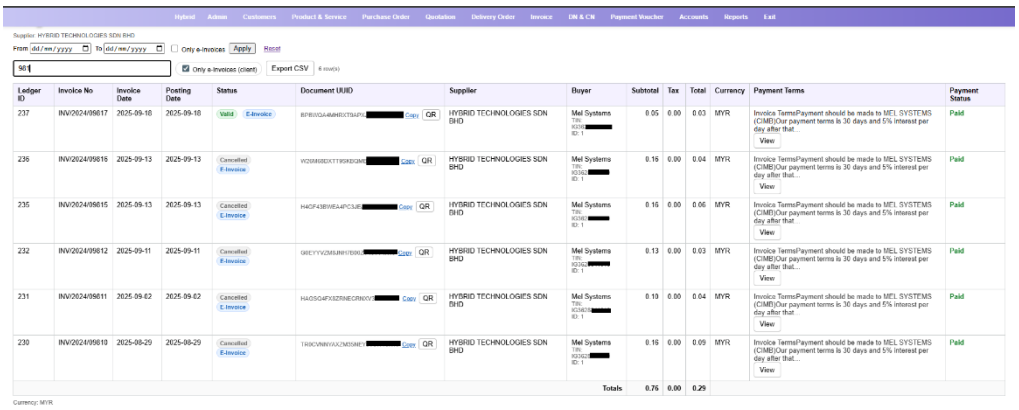


Figure 5.21 Filter based on Date, Keywords, or Type of Invoice

The image above demonstrates the additional feature of the ledger page, which is to filter based on date, keywords or type of invoice. In this example, “961” is part of the filtration keyword. As a result, only invoices or e-invoices with “961” are shown.

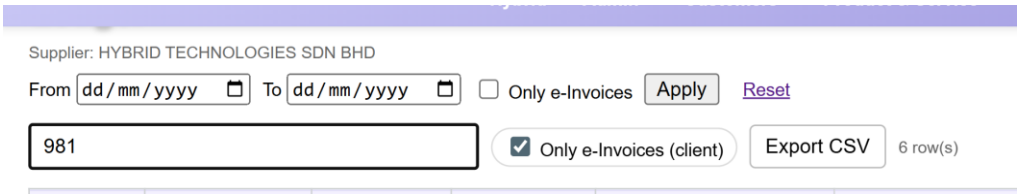


Figure 5.22 Export Invoices or E-Invoices to Excel

Furthermore, the ledger page also supports exporting invoices or e-invoice records to Excel format. This allows users to perform further analysis, reporting, or auditing outside the system. Hence, by supporting data export, the reviewing and analysis of financial transactions become more efficient, as users can easily evaluate records without manually copying data from the system.

CHAPTER 6

SYSTEM IMPLEMENTATION

6.1 Introduction

This chapter details the intricate workflow of the backend architecture process of integrating LHDN API to the web application. It covers the introduction of the setup of the subdomain in the cloud environment by utilizing Direct Control Admin Panel, until the integration of the entire e-invoicing API process.

6.2 Connection Setup

6.2.1 Create a Subdomain at Direct Admin Control Panel

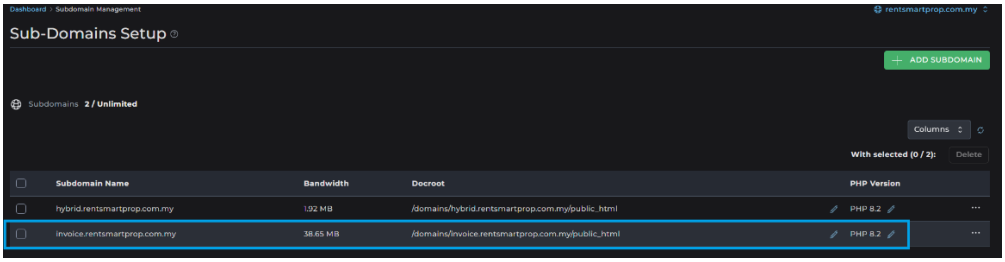


Figure 6.1 Subdomain Setup

The initial process in setting up the environment for the web application is to first establish a subdomain using the DirectAdmin Control Panel. Subdomains can be easily created by logging into the DirectAdmin Panel, and navigate to the Subdomain Management section. To create the subdomain, one can just simply type their preferred prefix to create the subdomain. For instance, naming the prefix to be “invoice” will then automatically create the subdomain “invoice.rentsmartprop.com.my”. With this, Direct Admin then establishes the appropriate folder structure in the hosting directory and automatically configures the necessary components. Hence, this setup establishes a platform where the web application is able to exist independently from the main domain, but yet utilizes the same web infrastructure.

6.2.2 Setup MySQL Database in Direct Admin Control Panel

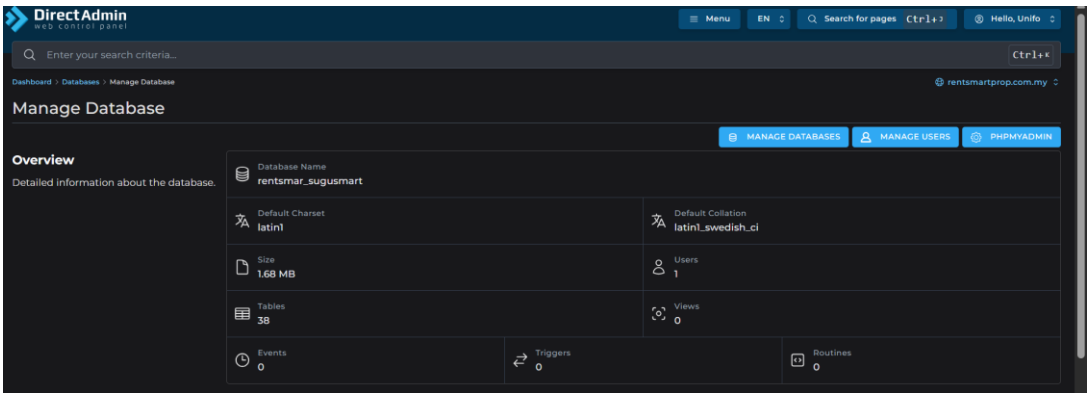


Figure 6.2 Setup MySQL Database

Once the subdomain is configured and running, the next step is to configure the MySQL database by utilizing the DirectAdmin Control Panel. By using the "Manage Database" section, the database for the web application can then be created by specifying the database name, collation and charset. As for assigning privileges, DirectAdmin will automatically assign the suitable privileges so that the current user account is able to manage and manipulate data within the database. With this setup, the database will be the central repository to hold invoices, e-invoices, payments and customer information, which will all be required for the generation of e-invoices using LHDN's protected APIs.

6.2.3 Configure Script that handles connection to Database

```
connectionImes.php > ...
1   <?php
2
3       // use the SQL operation for all the tables
4       //default online engine is MyISAM
5       // code ALTER TABLE my_table ENGINE = MyISAM;
6
7       //set database variables
8       $mysql_host = "";
9       $mysql_user = "rentsmar_sugusmart";
10      $mysql_password = "                    ";
11      $mysql_database = "rentsmar_sugusmart";
12
```

Figure 6.3 Configure Script to Connect to Database

Finally, the last step of setting up the web application via web hosting is through the configuration of the database connection script. This script is used to manage the connection to the newly established database. In this script, the database host, database user, database password and database name are specified. Once properly configured, the script is then transferred via FTP using FileZilla. With this, the script then establishes a secure connection to the MySQL database.

6.3 Registering as a Business Entity

Before the e-Invoice is able to communicate with the LHDN's protected APIs, it is mandatory to register the company as a business entity by registering it as an Enterprise Resource Planning (ERP) system within the MyInvois Portal. This step is important to ensure the access token can be generated by utilizing the client_id and client_secret, which are retrieved from this process.

6.3.1 Log In to the MyTax Portal

Figure 6.4 Screenshot of Logging Into MyTax Portal (SQL Account HQ, 2024)

Firstly, the taxpayer must first log in into the official MyTax Portal using their credentials such as their identification number and business registration number.

6.3.2 Click on “MyInvois” to be redirected to the MyInvois Portal

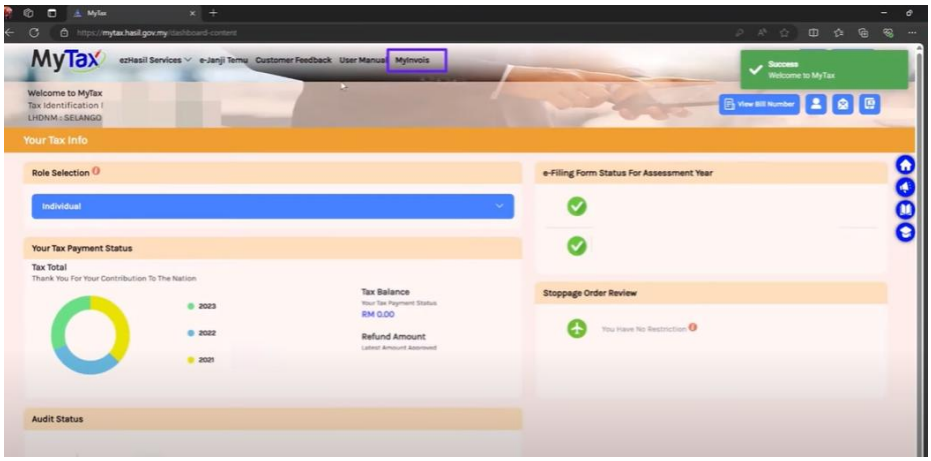


Figure 6.5 Screenshot of MyInvois Portal (SQL Account HQ, 2024)

Upon successful login, the taxpayer selects the “MyInvois” from the navigation header. This will redirect the taxpayer to the dedicated MyInvois Portal, which is designed in a way to handle e-invoicing processes. This portal in particular provides functionalities such as taxpayer profile management, manual invoice submissions, and also registering as an ERP system.

6.3.3 Click on “View Taxpayer Profile”

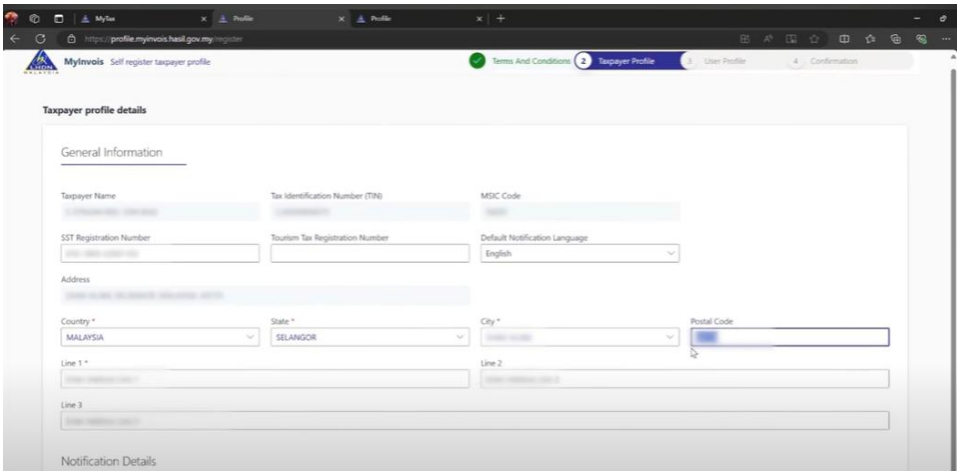


Figure 6.6 Screenshot of Taxpayer Profile (SQL Account HQ, 2024)

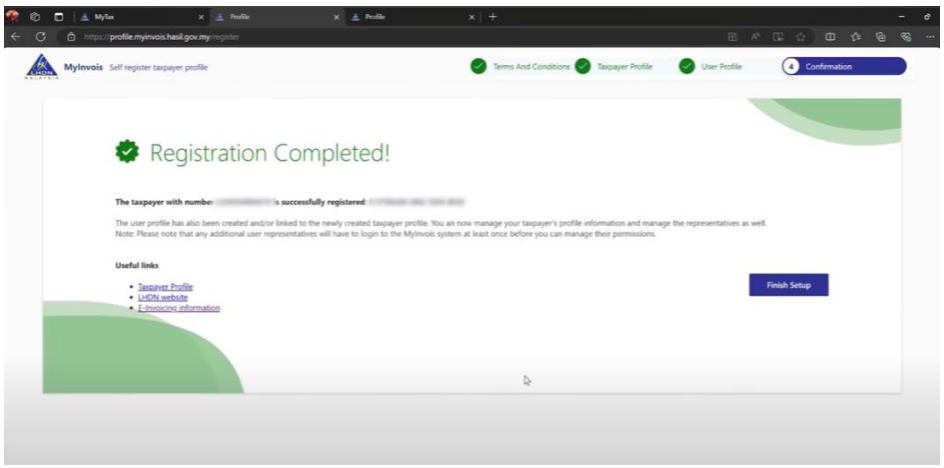


Figure 6.7 Screenshot of Complete Registration (SQL Account HQ, 2024)

Within the MyInvois Portal, the taxpayer will then navigate to their profile section by clicking on the “View Taxpayer Profile” in the navigation header.

6.3.4 Click on “Register ERP” under Representatives

Figure 6.8 Screenshot of Registering as an ERP (SQL Account HQ, 2024)

In the profile view, the taxpayer can locate and select the “Register ERP” option to proceed in initiating the process of registering their system as an ERP system.

6.3.5 Enter Required Details

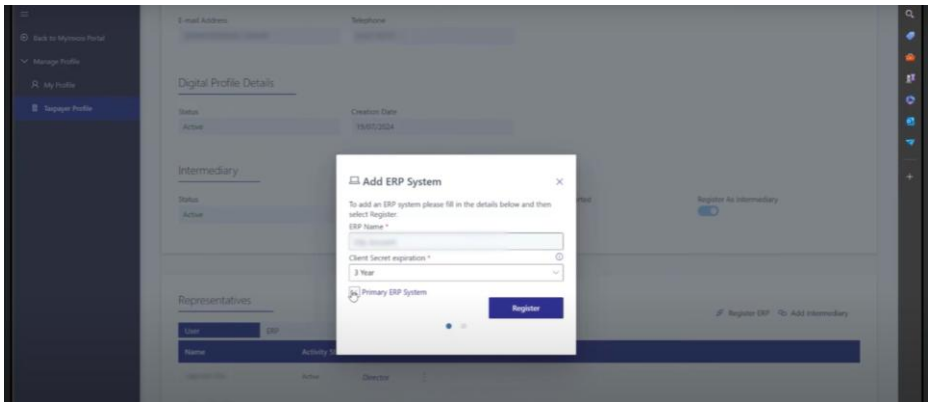


Figure 6.9 Screenshot of Entering Required Details (SQL Account HQ, 2024)

In this section, the user is required to provide important information such as ERP system name, contact information and other sensitive details. Providing accurate details in this section is particularly important as these inputs serve as a way for LHDN to uniquely identify the ERP system, verifies its legitimacy and prevent unauthorized access.

6.3.6 Click Register

After filling all the required information, the taxpayer submits the registration details by clicking on the “Register” button. This action finalizes the ERP registration and generates the required credentials to generate the access token, which are later used to access protected LHDN APIs.

6.3.7 Copy “Client Id”, “Client Secret 1” and “Client Secret 2”

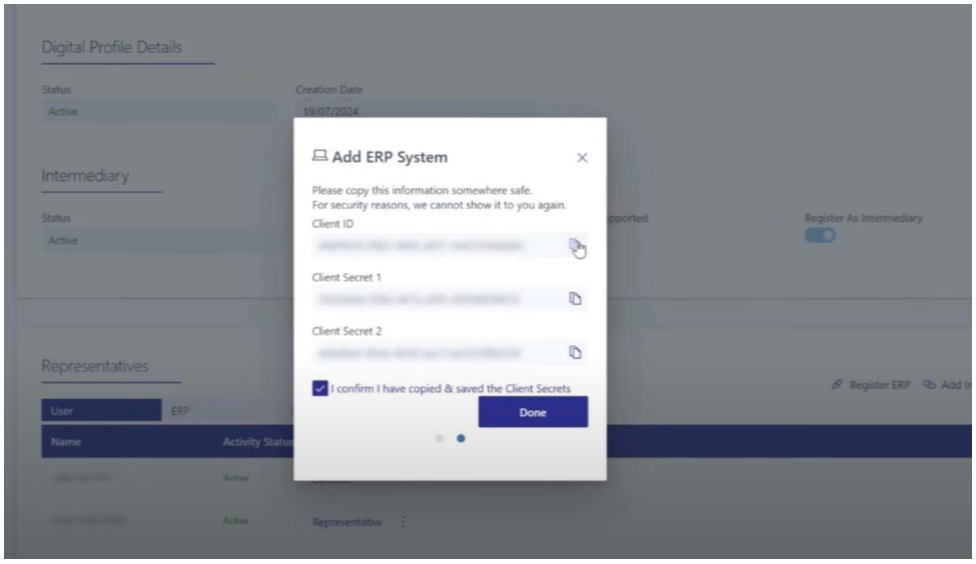


Figure 6.10 Screenshot of the Generated Client Id and Client Secret (SQL Account HQ, 2024)

The generated “Client Id”, “Client Secret 1” and “Client Secret 2” are vital and be used by the ERP in the upcoming sections to authenticate via the OAuth 2.0 provider and obtain the access tokens from the LHDN’s servers. Hence, it is of utmost importance for taxpayers to store these sensitive credentials securely.

6.4 Screenshot of Application’s User Interface (UI) and Code Snippet

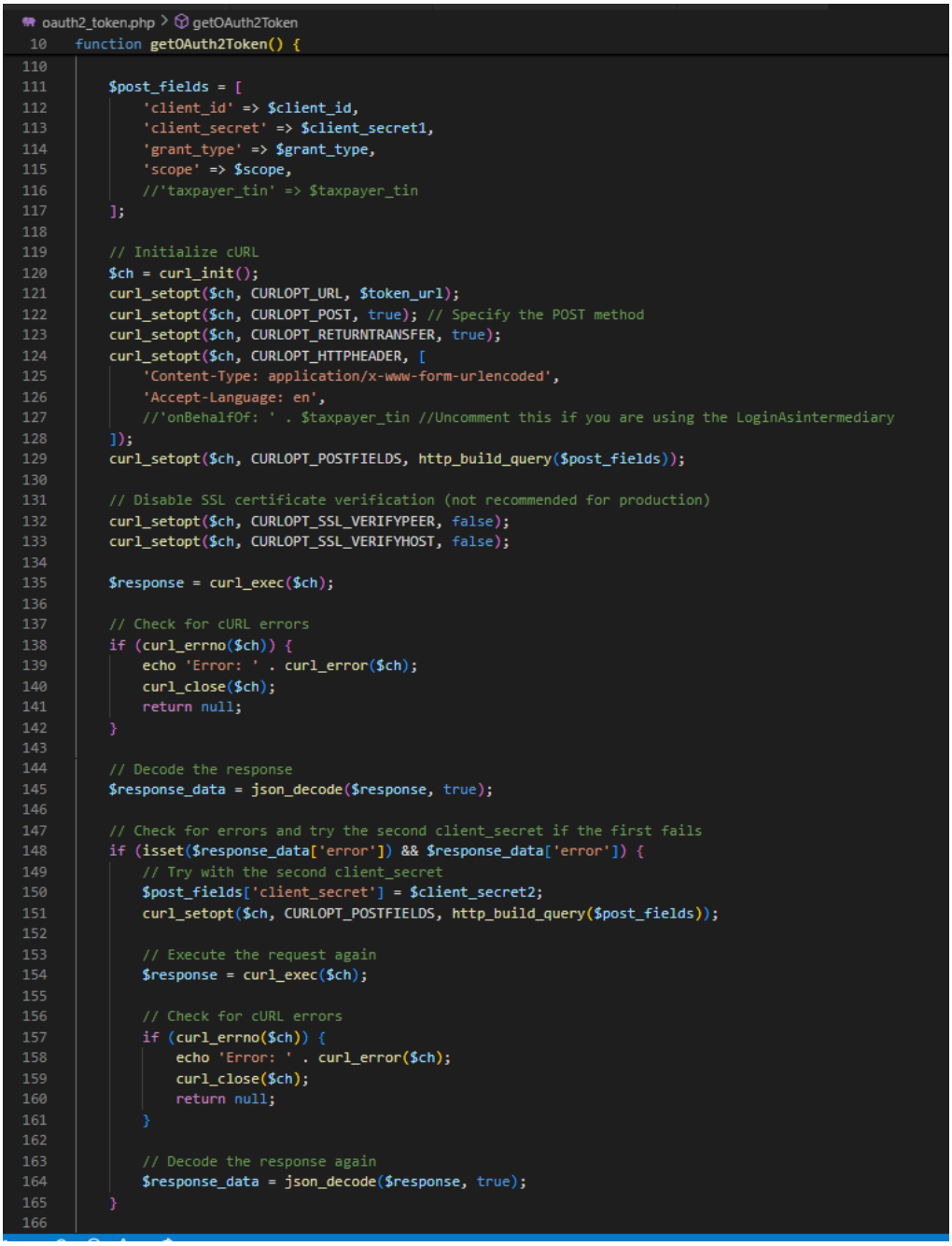
6.4.1 Creating the Login Page

The login page of the web-based e-Invoice application represents the access point in which, taxpayers can authenticate, log in and use the e-invoicing system.

6.4.1.1 Create Access Token to Access LHDN Protected APIs

```php
function getOAuth2Token() {
    // Define the token endpoint URL for the OAuth2.0 identity service

    /* This is the URL for the production environment*/
    $token_url = 'https://api.myinvois.hasil.gov.my/connect/token';

    /* This is the API endpoint for the pre-production (sandbox) environment*/
    //$token_url = 'https://preprod-api.myinvois.hasil.gov.my/connect/token';

    $client_id = null;
    $client_secret1 = null;
    $client_secret2 = null;

    // Try DB lookup
    $dbLoaded = false;
    try {
        require_once 'connectionImes.php';
        $conn = @mysqli_connect($mysql_host, $mysql_user, $mysql_password, $mysql_database);
        if ($conn) {
            $dbLoaded = true;
            $sql = "SELECT client_id, client_secret1, client_secret2 FROM tbsecret WHERE secret_status=1 LIMIT 1";
            $res = mysqli_query($conn, $sql);
            if ($res && mysqli_num_rows($res) > 0) {
                $row = mysqli_fetch_assoc($res);
                $client_id = isset($row['client_id']) ? trim($row['client_id']) : null;
                $client_secret1 = isset($row['client_secret1']) ? trim($row['client_secret1']) : null;
                $client_secret2 = isset($row['client_secret2']) ? trim($row['client_secret2']) : null;
            } else {
                $sql2 = "SELECT client_id, client_secret1, client_secret2 FROM tbsecret LIMIT 1";
                $res2 = mysqli_query($conn, $sql2);
                if ($res2 && mysqli_num_rows($res2) > 0) {
                    $row = mysqli_fetch_assoc($res2);
                    $client_id = isset($row['client_id']) ? trim($row['client_id']) : null;
                    $client_secret1 = isset($row['client_secret1']) ? trim($row['client_secret1']) : null;
                    $client_secret2 = isset($row['client_secret2']) ? trim($row['client_secret2']) : null;
                }
            }
            @mysqli_close($conn);
        }
    } catch (Exception $e) {
    }

    if (empty($client_id)) {
        $client_id = getenv('LHDN_CLIENT_ID') ?: null;
    }
    if (empty($client_secret1)) {
        $client_secret1 = getenv('LHDN_CLIENT_SECRET1') ?: null;
    }
    if (empty($client_secret2)) {
        $client_secret2 = getenv('LHDN_CLIENT_SECRET2') ?: null;
    }
```

Figure 6.11 Retrieve Client Id and Client Secret from Database



```php
oauth2_token.php > getOAuth2Token
function getOAuth2Token() {

    $post_fields = [
        'client_id' => $client_id,
        'client_secret' => $client_secret1,
        'grant_type' => $grant_type,
        'scope' => $scope,
        //'taxpayer_tin' => $taxpayer_tin
    ];

    // Initialize cURL
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $token_url);
    curl_setopt($ch, CURLOPT_POST, true); // Specify the POST method
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_HTTPHEADER, [
        'Content-Type: application/x-www-form-urlencoded',
        'Accept-Language: en',
        //'onBehalfOf: ' . $taxpayer_tin //Uncomment this if you are using the LoginAsintermediary
    ]);
    curl_setopt($ch, CURLOPT_POSTFIELDS, http_build_query($post_fields));

    // Disable SSL certificate verification (not recommended for production)
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
    curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, false);

    $response = curl_exec($ch);

    // Check for cURL errors
    if (curl_errno($ch)) {
        echo 'Error: ' . curl_error($ch);
        curl_close($ch);
        return null;
    }

    // Decode the response
    $response_data = json_decode($response, true);

    // Check for errors and try the second client_secret if the first fails
    if (isset($response_data['error']) && $response_data['error']) {
        // Try with the second client_secret
        $post_fields['client_secret'] = $client_secret2;
        curl_setopt($ch, CURLOPT_POSTFIELDS, http_build_query($post_fields));

        // Execute the request again
        $response = curl_exec($ch);

        // Check for cURL errors
        if (curl_errno($ch)) {
            echo 'Error: ' . curl_error($ch);
            curl_close($ch);
            return null;
        }

        // Decode the response again
        $response_data = json_decode($response, true);
    }
```

Figure 6.12 Initiate API call to the Login As Taxpayer API

```
{} token_cache.json > ...
1  {"access_token":"eyJhbGciOiJSUzI1NiIsImtpZCI6Ijk2RjNBNjU2OEFEQzY0MzZDNjVBNDg1MUQ5REM0NTlFQTlCM0I1NTRSUzI1NiIsIng1dCI6Imx2T21Wb3JjWkRiR1draFIyZHhGbnFtenR6WUSIsInR5cCI6ImF0K
2  "expires_in":3600,
3  "token_type":"Bearer",
4  "scope":"InvoicingAPI"}
```

Figure 6.13 Store the access token in JSON file

As mentioned in the previous section, the taxpayer must first register as an Enterprise Resource Planning (ERP) in order to get the necessary credentials such as the client_id, client_secret1 and client_secret2. These credentials

serves as unique identifiers that establish trust between the LHDN's secure API and the ERP system.

Upon obtaining the credentials, the client_id, client_secret1, and client_secret2 can be saved in the admin settings page, and it will automatically store those credentials in the database under the tb_company fields. This is to allow easy retrieval of the credentials to compute the access token. Hence, when the taxpayer logs in, the access token will be generated by automatically sending a request to the LoginAsTaxPayer API. The LoginAsTaxPayer API is an API that strictly adheres to the OAuth 2.0 authentication standards such that the ERP system exchanges its credentials with an access token. This token is required as it enables the application to safely communicate with LHDN's secured endpoints without frequently exposing the client's login credentials. The screenshot of the code snippet in this section shows how the request is initiated and sent, making clear how the client_id and client_secret parameters are applied in the token request process.

Upon successful authentication, the LHDN system gives back an OAuth2 access token. This token is then cached to a JSON file within the web application in such a way that subsequent API calls such as submitting document, retrieve document details can be executed without requiring repeated logins on the part of the taxpayer.

6.4.2 Creating the Home Page

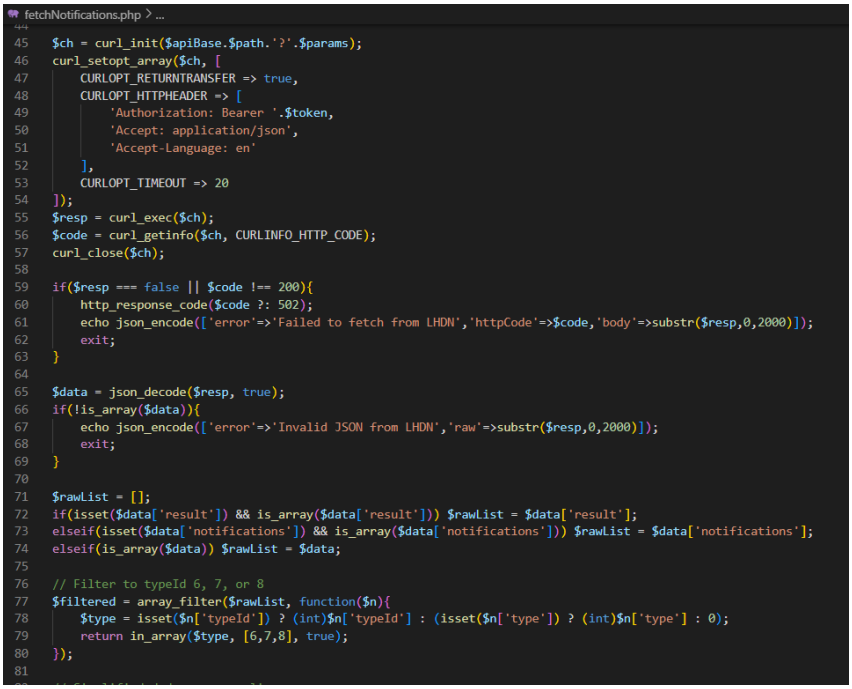
The home page of the e-Invoice application is the main dashboard for the taxpayer after successful login. It is designed to give a good overview of the functionality offered and serve as the main navigation point to main modules such as invoice generation, invoice management, profile management, and reporting through the implementation of a ledger.

6.4.2.1 Implementing the Notification Feature

fetchNotifications.php > ...

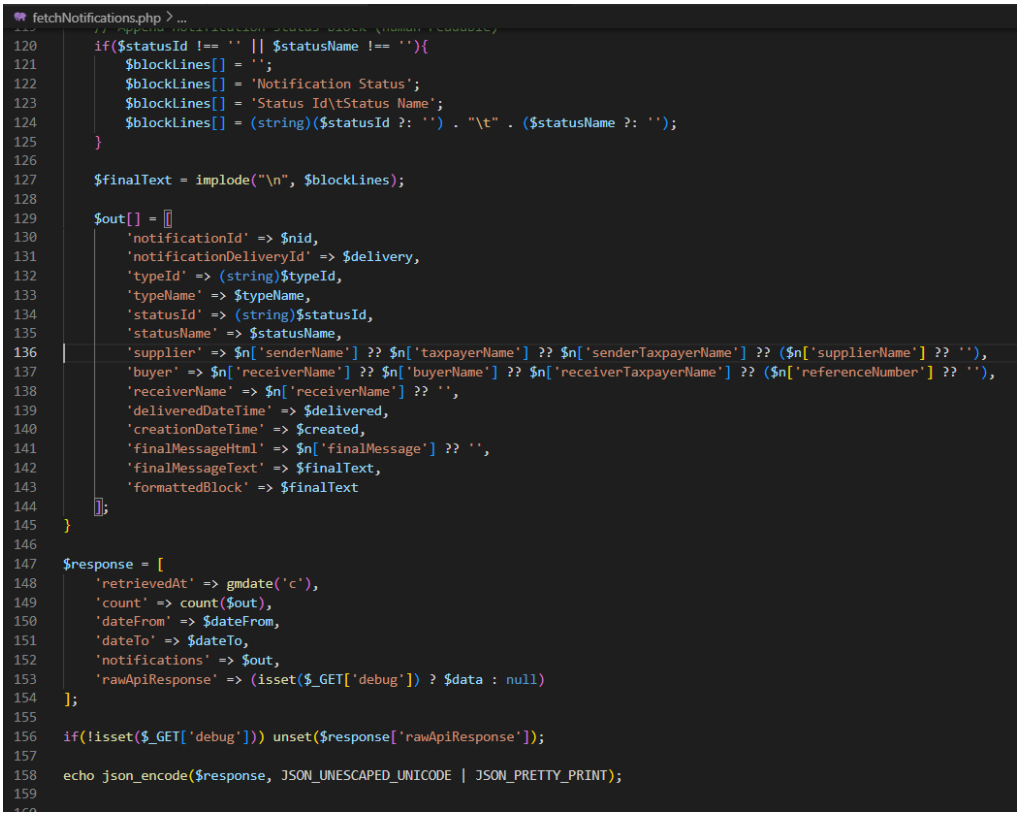
```php
header('Content-Type: application/json; charset=utf-8');
session_start();

if(empty($_SESSION['userID'])){
    http_response_code(401);
    echo json_encode(['error'=>'Unauthorized']);
    exit;
}

$token = !empty($_SESSION['lhdn_access_token']) ? trim($_SESSION['lhdn_access_token']) : '';
if($token === ''){
    http_response_code(500);
    echo json_encode(['error'=>'Missing access token in session (lhdn_access_token)']);
    exit;
}

$apiBase = 'https://api.myinvois.hasil.gov.my';
$path = '/api/v1.0/notifications/taxpayer';

$since = isset($_GET['since']) ? trim($_GET['since']) : '';
if($since !== '' && preg_match('/^\\d+$/', $since)){
    $mins = (int)$since;
    if($mins > 43200) $mins = 43200; // cap
    $dateFrom = gmdate('Y-m-d\\TH:i:s\\Z', time() - $mins*60);
    $dateTo = gmdate('Y-m-d\\TH:i:s\\Z');
} else {
    // default: last 120 hours
    $dateTo = gmdate('Y-m-d\\TH:i:s\\Z');
    $dateFrom = gmdate('Y-m-d\\TH:i:s\\Z', time() - 120*3600);
}

$params = http_build_query([
    'dateFrom' => $dateFrom,
    'dateTo'   => $dateTo,
    'pageNo'   => 1,
    'pageSize' => 100
]);
```

Figure 6.14 Filter the Notifications before Retrieval



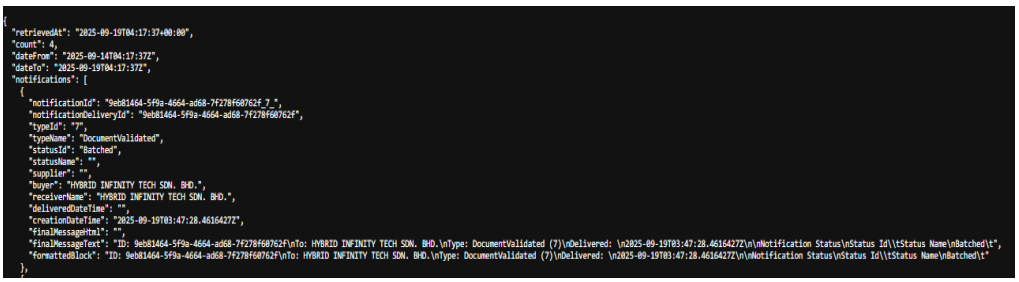
```php
fetchNotifications.php > ...
$ch = curl_init($apiBase.$path.'?'.$params);
curl_setopt_array($ch, [
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_HTTPHEADER => [
        'Authorization: Bearer '.$token,
        'Accept: application/json',
        'Accept-Language: en'
    ],
    CURLOPT_TIMEOUT => 20
]);
$resp = curl_exec($ch);
$code = curl_getinfo($ch, CURLINFO_HTTP_CODE);
curl_close($ch);

if($resp === false || $code !== 200){
    http_response_code($code ?: 502);
    echo json_encode(['error'=>'Failed to fetch from LHDN','httpCode'=>$code,'body'=>substr($resp,0,2000)]);
    exit;
}

$data = json_decode($resp, true);
if(!is_array($data)){
    echo json_encode(['error'=>'Invalid JSON from LHDN','raw'=>substr($resp,0,2000)]);
    exit;
}

$rawList = [];
if(isset($data['result']) && is_array($data['result'])) $rawList = $data['result'];
elseif(isset($data['notifications']) && is_array($data['notifications'])) $rawList = $data['notifications'];
elseif(is_array($data)) $rawList = $data;

// Filter to typeId 6, 7, or 8
$filtered = array_filter($rawList, function($n){
    $type = isset($n['typeId']) ? (int)$n['typeId'] : (isset($n['type']) ? (int)$n['type'] : 0);
    return in_array($type, [6,7,8], true);
});
```

Figure 6.15 Construct the payload to send to the Get Notifications API



```php
fetchNotifications.php > ...
    if($statusId !== '' || $statusName !== ''){
        $blockLines[] = '';
        $blockLines[] = 'Notification Status';
        $blockLines[] = 'Status Id\tStatus Name';
        $blockLines[] = (string)($statusId ?: '') . "\t" . ($statusName ?: '');
    }

    $finalText = implode("\n", $blockLines);

    $out[] = [
        'notificationId' => $nid,
        'notificationDeliveryId' => $delivery,
        'typeId' => (string)$typeId,
        'typeName' => $typeName,
        'statusId' => (string)$statusId,
        'statusName' => $statusName,
        'supplier' => $n['senderName'] ?? $n['taxpayerName'] ?? $n['senderTaxpayerName'] ?? ($n['supplierName'] ?? ''),
        'buyer' => $n['receiverName'] ?? $n['buyerName'] ?? $n['receiverTaxpayerName'] ?? ($n['referenceNumber'] ?? ''),
        'receiverName' => $n['receiverName'] ?? '',
        'deliveredDateTime' => $delivered,
        'creationDateTime' => $created,
        'finalMessageHtml' => $n['finalMessage'] ?? '',
        'finalMessageText' => $finalText,
        'formattedBlock' => $finalText
    ];
}

$response = [
    'retrievedAt' => gmdate('c'),
    'count' => count($out),
    'dateFrom' => $dateFrom,
    'dateTo' => $dateTo,
    'notifications' => $out,
    'rawApiResponse' => (isset($_GET['debug']) ? $data : null)
];

if(!isset($_GET['debug'])) unset($response['rawApiResponse']);

echo json_encode($response, JSON_UNESCAPED_UNICODE | JSON_PRETTY_PRINT);
```

Figure 6.16 Sanitize the Notification Output into Readable Format



```
{
  "retrievedAt": "2025-09-19T04:17:37+00:00",
  "count": 4,
  "dateFrom": "2025-09-14T04:17:37Z",
  "dateTo": "2025-09-19T04:17:37Z",
  "notifications": [
    {
      "notificationId": "9eb81464-5f9a-4664-ad68-7f278f60762f_7_",
      "notificationDeliveryId": "9eb81464-5f9a-4664-ad68-7f278f60762f",
      "typeId": "7",
      "typeName": "DocumentValidated",
      "statusId": "Batched",
      "statusName": "",
      "supplier": "",
      "buyer": "HYBRID INFINITY TECH SDN. BHD.",
      "receiverName": "HYBRID INFINITY TECH SDN. BHD.",
      "deliveredDateTime": "",
      "creationDateTime": "2025-09-19T03:47:28.4616427Z",
      "finalMessageHtml": "",
      "finalMessageText": "ID: 9eb81464-5f9a-4664-ad68-7f278f60762f\nTo: HYBRID INFINITY TECH SDN. BHD.\nType: DocumentValidated (7)\nDelivered: \n2025-09-19T03:47:28.4616427Z\n\nNotification Status\nStatus Id\\tStatus Name\nBatched\t",
      "formattedBlock": "ID: 9eb81464-5f9a-4664-ad68-7f278f60762f\nTo: HYBRID INFINITY TECH SDN. BHD.\nType: DocumentValidated (7)\nDelivered: \n2025-09-19T03:47:28.4616427Z\n\nNotification Status\nStatus Id\\tStatus Name\nBatched\t"
    },
```


Figure 6.17 Output from Get Notifications API

One of the main features on the home page is its integration with the LHDN Notification API, which allows for the real-time retrieval of important notifications from the MyInvois system. These may be status updates on invoice submissions, error messages, or system notifications related to the taxpayer. By displaying these notifications directly on the home page, the system ensures that taxpayers are immediately aware of any issues or updates, improving responsiveness and compliance.

6.4.3 Process of Creating e-Invoice

6.4.3.1 Invoice Generation

Invoice generation in the e-Invoice system is a formal procedure to transform internal invoice data into signed, canonicalized UBL 2.1 JSON documents that can be posted to the LHDN API. The steps to achieve this include mapping, transformation, canonicalization, digest creation, signing, certificate hash, signed-properties population, and signed-properties hash. In the current implemention, the mapping operations are handled by invoice_mapper.php and createInvoice.php, which both transform the invoice data into normalized JSON form, according to LHDN's standards. The entire process is then orchestrated via the UI, with just a click on the "Sync" button. This gives operators a degree of control over the submission process and allow them to validate invoice data before digitally signing the document.

6.4.3.1.1 Generate JSON Document According to UBL 2.1 Format

This step requires producing one complete, well-formed UBL 2.1 invoice document which could be in either XML or the UBL-JSON format, with no

signature elements included yet. For this project, the chosen format is the JSON format which also follows the UBL 2.1 format.

6.4.3.1.2 Apply Transformations to Document

The second step involves structural transformations that are applied to the document to meet the exact UBL JSON standards required by the signing process and LHDN. The first requirement is to ensure that the document is encoded in UTF-8 so that the character bytes remain consistent across different environments. The transformation also involves the removal of elements that are not required at this stage, specifically the <UBLExtensions>, <UBLExtension>, and <Signature> nodes. Once removed, the document should be validated to ensure that all mandatory UBL 2.1 structures remain intact. The end result of this transformation process is a clean, UTF-8 encoded UBL document without unnecessary extension or signature components.

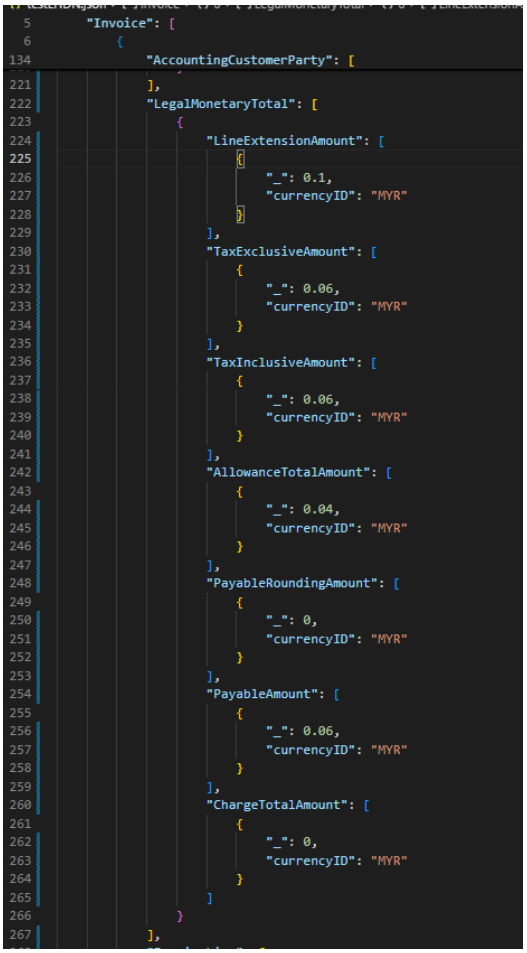
6.4.3.1.3 Canonicalize the Document

```php
$g = static function (string $k, $d = null) use ($post) { return isset($post[$k]) && $post[$k] !== '' ? $post[$k] : $d; };
$num = static function ($v) { return $v === null ? 0.0 : (float)str_replace([',',' '], '', (string)$v); };

$currency = 'MYR';
$invoiceNo = (string)$g('invoiceNo', '');
$issueDate = gmdate('Y-m-d', time()-120);

$lineDesc = $post['noCol2'] ?? [];
$lineQty = $post['noCol3'] ?? [];
$linePrice = $post['noCol5'] ?? [];
$lineTaxRate = $post['noCol8'] ?? [];

$invoiceLines = [];
$taxByCode = [];
$max = max(count($lineDesc), count($lineQty), count($linePrice), count($lineTaxRate));
for ($i = 0; $i < $max; $i++) {
    $desc = trim((string)($lineDesc[$i] ?? '')); if ($desc === '') continue;
    $qty = $num($lineQty[$i] ?? 0);
    $price = $num($linePrice[$i] ?? 0);
    $taxRate = $num($lineTaxRate[$i] ?? 0);

    $lineNet = round(max(0.0, $qty * $price), 2);
    $lineTaxAmt = round($lineNet * ($taxRate / 100.0), 2);

    $lineArr = [
        "ID" => [[ "_" => (string)($i+1) ]],
        "InvoicedQuantity" => [[ "_" => round($qty,4), "unitCode" => "C62" ]],
        "LineExtensionAmount" => [[ "_" => $lineNet, "currencyID" => $currency ]],
        "Item" => [[ "Description" => [[ "_" => $desc ]] ]],
        "Price" => [[ "PriceAmount" => [[ "_" => round($price,2), "currencyID" => $currency ]]]],
        "TaxTotal" => [[ "TaxAmount" => [[ "_" => $lineTaxAmt, "currencyID" => $currency ]],
                         "TaxSubtotal" => [[ "TaxableAmount" => [[ "_" => $lineNet, "currencyID" => $currency ]],
                                             "TaxAmount" => [[ "_" => $lineTaxAmt, "currencyID" => $currency ]],
                                             "TaxCategory" => [[ "Percent" => [[ "_" => round($taxRate,2) ]]]]]]]
    ];

    $invoiceLines[] = $lineArr;
    $codeKey = $taxRate > 0 ? (string)$taxRate : 'ZERO';
    $taxByCode[$codeKey]['amount'] = ($taxByCode[$codeKey]['amount'] ?? 0.0) + $lineTaxAmt;
    $taxByCode[$codeKey]['taxable'] = ($taxByCode[$codeKey]['taxable'] ?? 0.0) + $lineNet;
}

// Build header TaxTotal and LegalMonetaryTotal (simplified)
$headerTaxAmount = 0.0;
$taxSubtotals = [];
foreach ($taxByCode as $code => $agg) {
    $headerTaxAmount += $agg['amount'];
    $taxSubtotals[] = ["TaxableAmount"=> [[ "_" => round($agg['taxable'],2), "currencyID"=>$currency ]],
                       "TaxAmount"=> [[ "_" => round($agg['amount'],2), "currencyID"=>$currency ]],
                       "TaxCategory"=> [ ["Percent" => [[ "_" => (float)$code ]]] ] ];
}
$invoice = [
    "ID" => [[ "_" => $invoiceNo ]],
    "IssueDate" => [[ "_" => $issueDate ]],
    "DocumentCurrencyCode" => [[ "_" => $currency ]],
    "InvoiceLine" => $invoiceLines,
    "TaxTotal" => [[ "TaxAmount" => [[ "_" => round($headerTaxAmount,2), "currencyID"=>$currency ]], "TaxSubtotal" => $taxSubtotals ]],
    "LegalMonetaryTotal" => [[ "LineExtensionAmount" => [[ "_" => array_sum(array_column($invoiceLines, 0)) ]] ]]
];
```

Figure 6.18 Code Snippet of invoice_mapper.php

```
{
    "_D": "urn:oasis:names:specification:ubl:schema:xsd:Invoice-2",
    "_A": "urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateComponents-2",
    "_B": "urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2",
    "Invoice": [
        {
            "ID": [
                {
                    "_": "INV/2024/09802"
                }
            ],
            "IssueDate": [
                {
                    "_": "2025-08-15"
                }
            ],
            "IssueTime": [
                {
                    "_": "02:59:44Z"
                }
            ],
            "InvoiceTypeCode": [
                {
                    "_": "01",
                    "listVersionID": "1.1"
                }
            ],
            "DocumentCurrencyCode": [
                {
                    "_": "MYR"
                }
            ],
            "TaxCurrencyCode": [
                {
                    "_": "MYR"
                }
            ],
            "AccountingSupplierParty": [
                {
                    "Party": [
                        {
                            "IndustryClassificationCode": [
                                {
                                    "_": "62091",
                                    "name": "Information Communication Technology (ICT) system security"
                                }
                            ],
                            "PartyIdentification": [
                                {
                                    "ID": [
```


Figure 6.19 Mapped JSON UBL 2.1 Template 1



```
    "Invoice": [
        {
            "AccountingCustomerParty": [
            ],
            "LegalMonetaryTotal": [
                {
                    "LineExtensionAmount": [
                        {
                            "_": 0.1,
                            "currencyID": "MYR"
                        }
                    ],
                    "TaxExclusiveAmount": [
                        {
                            "_": 0.06,
                            "currencyID": "MYR"
                        }
                    ],
                    "TaxInclusiveAmount": [
                        {
                            "_": 0.06,
                            "currencyID": "MYR"
                        }
                    ],
                    "AllowanceTotalAmount": [
                        {
                            "_": 0.04,
                            "currencyID": "MYR"
                        }
                    ],
                    "PayableRoundingAmount": [
                        {
                            "_": 0,
                            "currencyID": "MYR"
                        }
                    ],
                    "PayableAmount": [
                        {
                            "_": 0.06,
                            "currencyID": "MYR"
                        }
                    ],
                    "ChargeTotalAmount": [
                        {
                            "_": 0,
                            "currencyID": "MYR"
                        }
                    ]
                }
            ],
```


Figure 6.20 Mapped JSON UBL 2.1 Template 2

Figure 6.21 Mapped JSON UBL 2.1 Template 3

The first step of the generation of the invoice is application-level invoice data mapping to the UBL 2.1 JSON format as mandated by LHDN. The createInvoice.php file collects all application-level invoice header fields such as Invoice Number, Issue Date, Customer and Supplier references, and terms of payment, and line and input tax data from the database tables which originated from the tbinvoice and tbinvoicedetail tables. These outputs are then passed through mapping processes in invoice_mapper.php to convert each line on the invoice into a normalized InvoiceLine record. Each record would then include an ID, quantity, unit code, LineExtensionAmount, item description, price, and total tax. Aggregated monetary totals are also calculated to populate LegalMonetaryTotal fields, including LineExtensionAmount, TaxExclusiveAmount, TaxInclusiveAmount, and PayableAmount. The invoice_mapper.php will then map those fields and the calculated fields to a JSON UBL 2.1 template provided by LHDN.

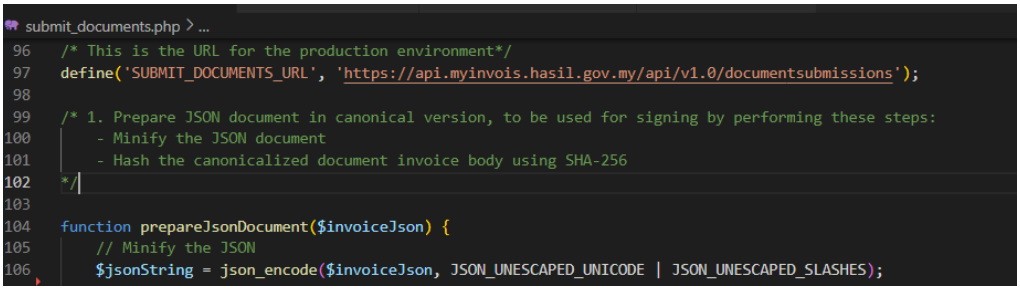


Figure 6.22 Canonicalization of JSON document

Canonicalization of document is also performed through minification to remove unnecessary whitespace, line breaks, and comments. This standardizes the JSON document to a uniform format. This canonical form serves as the foundation for calculating the document digest and must be identical in all environments in order to check signatures.

6.4.3.1.4 Generate Document Hash (Digest)

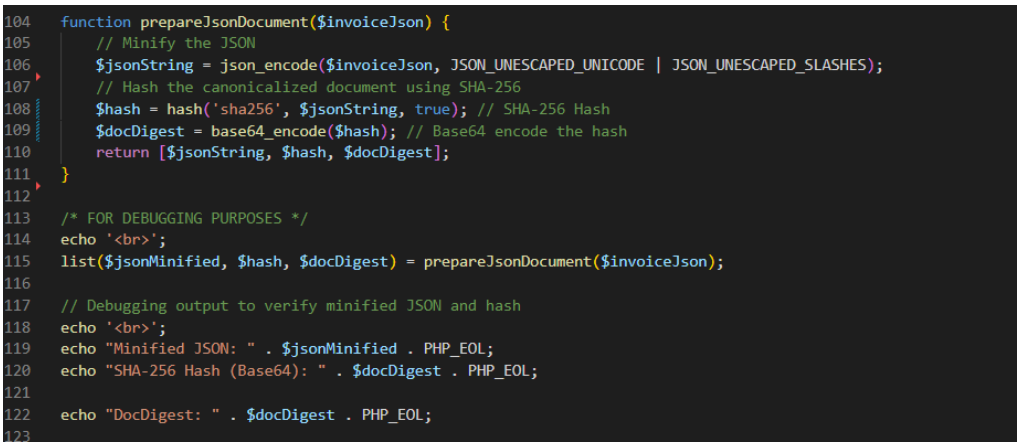


Figure 6.23 Generate DocDigest

With the canonicalized JSON bytes, a cryptographic digest is computed which utilizes SHA-256 to produce a SHA-256 hash, and then subsequently base64-encoded. The resultant digest will be set as the value of the property DocDigest in the upcoming section. Thus, the naming convention for the base64 encoded hash will remained as “DocDigest”. The DocDigest is the document's integrity and is also conveyed in the SignedProperties and in the submission payload.

6.4.3.1.5 Sign the Document Digest

```php
<?php
/* For Debugging Purposes */
ini_set('display_errors', 1);
error_reporting(E_ALL);
echo OPENSSL_VERSION_TEXT;

/* Overall Process :
-- This file is mainly for generating and signing the document digest and certificate hash. --

1.4 LOAD THE PRIVATE CERTIFICATE > EXTRACT THE PRIVATE KEY > CREATE RSA SIGNATURE FORMATTER > SIGN THE HASH >CONVERT HASH TO BASE64
1.5. LOAD THE PRIVATE CERTIFICATE > COMPUTE THE HASH OF THE CERTIFICATE > CONVERT THE HASH TO BASE64
*/

// Get the current script path
$currentPath = dirname(__FILE__);

// Output the current script path
echo 'Current script path: ' . $currentPath . PHP_EOL;
echo "\n";

// Path to the .p12 file
$p12FilePath = $currentPath . DIRECTORY_SEPARATOR . "certificate" . DIRECTORY_SEPARATOR . "CERT_1[redacted].p12";
$p12Password = "[redacted]"; //softcert pin

echo $p12FilePath ;
echo "\n";

// Check if the file exists
if (!file_exists($p12FilePath)) {
    die('The .p12 file does  not exist: ' . $p12FilePath);
}
if (!is_readable($p12FilePath)) {
    die('The .p12 file is not readable: ' . $p12FilePath);
}

// Load the .p12 file
try {
        $p12Content = file_get_contents($p12FilePath);
        if ($p12Content === false) {
            die('Failed to read the .p12 file');
        }
}
catch(Exception $e) {
  echo 'Message: ' .$e->getMessage();
}
```

Figure 6.24 Access the Production Certificate Directory

```php
/* 2. Generate Digital Signature: Sign the generated invoice hash (DocDigest) with RSA-SHA256 using the signing certificate private key
   Please refer to the code in generateSig.php#id-xades-signed-props
*/
echo '<br>';
$cleanedJson = json_encode($invoiceJson, JSON_UNESCAPED_UNICODE | JSON_UNESCAPED_SLASHES);
$signature = ''; // $signature passed by reference
$signatureValid = openssl_sign($cleanedJson, $signature, $privateKey, OPENSSL_ALGO_SHA256);
$sig = base64_encode($signature);
echo "Signature (Sig): " . $sig . PHP_EOL;
```

Figure 6.25 Sign the generated invoice hash with RSA-SHA256 using the signing certificate private key

The signing process generates a signature over the document digest or SignedProperties structure referencing the digest. With a private key extracted from a PKCS#12 (.p12) file which was executed from generateSig.php, the application performs an RSA-SHA256 signature, particularly represented by “openssl_sign”. The output of the signature will then be referenced to the property “Sig”.

6.4.3.1.6 Generate the Certificate Hash

generateSig.php > ...

```php
// Function to load the certificate and private key
function loadCertificate($p12Content, $p12Password) {
    $certs = [];
    if (!openssl_pkcs12_read($p12Content, $certs, $p12Password)) {
        // Detailed error message
        while ($error = openssl_error_string()) {
            echo "OpenSSL Error: $error\n";
        }
        die('Failed to parse the .p12 file');
    }

    // Validate keys and certificate
    if (!isset($certs['pkey']) || !isset($certs['cert'])) {
        die('The .p12 file does not contain the expected keys or certificate.');
    }

    return $certs;
}

// Load the certificate and private key
$certs = loadCertificate($p12Content, $p12Password);
if (!$certs) {
    die('Failed to load the certificate and private key.');
}
```

Figure 6.26 Load Production Certificate

```php
//1.5 Step 5: Generate the certificate hash
// Function to compute the hash of the certificate
function computeCertHash($cert) {

    // Convert certificate to binary format [! important]
    $certBin = base64_decode(preg_replace('/\-{5}.*?\-{5}/', '', $cert));

    // Compute SHA-256 hash
    $certHash = hash('sha256', $certBin, true);
    return base64_encode($certHash);
}

// Output certificate details
echo "\n";
echo 'Certificate: ' . $certs['cert'] . PHP_EOL;

// Print certificate details
$certDetails = openssl_x509_parse($certs['cert']);
if ($certDetails) {
    print_r($certDetails);
} else {
    echo "Failed to parse certificate details.\n";
}
?>
```

Figure 6.27 Generate the CertHash

To facilitate certificate identification, a hash of the signing certificate is also computed. The extracted certificate from the .p12 certificate format is hashed using SHA-256 hashing algorithm and subsequently base64-encoded. The certificate hash is then inserted into the signing block under SigningCertificate.CertDigest.

6.4.3.1.7 Populate the Signed Properties Section

```php
/* 4. Populate the signed properties section by performing these:
    4.1 - Calculate the DigestValue(3), SigningTime, X509SerialNumber and X509IssuerName
    4.2 - Obtain the qualifyingProperties properties in string format > minify the string > compute the hashed of the signed properties > covert to base64
    4.3 - Assign it to PropsDigest
*/

/*
4.1a - Calculate the DigestValues
    1st DigestValue is referring to the DocDigest (SHA-256 hash of the canonicalized (minified) JSON document)
    2nd DigestValue is referring to the CertDigest (CertHash)
    3rd DigestValue is referring to the PropsDigest (SHA-256 hash of the minified string of the signed properties)
*/

//4.1b - Calculate the SigningTime (Note: make sure the SigningTime in the chunk is later than document submission time)

// Set the SigningTime to a time slightly later than the current time
echo '<br>';
$signingTime = gmdate('Y-m-d\TH:i:s\Z');
echo "Signing Time: " . $signingTime . PHP_EOL;

//4.1c - Calculate the X509SerialNumber
// Note: Use the SerialNumber of the private certificate (.p12), convert the hexadecimal string to a BigInteger

function getCertSerialNumber($cert) {
    $certDetails = openssl_x509_parse($cert);
    $serialNumberHex = $certDetails['serialNumberHex'];
    // Remove any leading "0x" or spaces
    $serialNumberHex = preg_replace('/^0x/i', '', $serialNumberHex);
    $serialNumberHex = preg_replace('/\s+/', '', $serialNumberHex);

    // Use BCMath to convert hex to decimal for large numbers
    if (function_exists('bcadd')) {
        $serialNumber = '0';
        $len = strlen($serialNumberHex);
        for ($i = 0; $i < $len; $i++) {
            $serialNumber = bcmul($serialNumber, '16');
            $serialNumber = bcadd($serialNumber, hexdec($serialNumberHex[$i]));
        }
    } else {
        // Fallback for small numbers
        $serialNumber = (string)hexdec($serialNumberHex);
    }
    return $serialNumber;
}
```

Figure 6.28 Calculate X509SerialNumber

```php
// Get the certificate serial number (BigInteger)
$serialNumber = getCertSerialNumber($certs['cert']);
echo $serialNumber . PHP_EOL;

// Function to get the certificate data in Base64 string
function getX509Certificate($cert) {
    return base64_encode($cert);
}

// Extract certificate data
$base64CertData = getX509Certificate($certs['cert']);
echo "Certificate Data (Base64): " . $base64CertData . PHP_EOL;

// Parse cert to extract in the form of associative array
$certData = openssl_x509_parse($certs['cert']);

// Extract and format the X509IssuerName
$issuerArray = $certData['issuer'];
$issuerName = "CN={$issuerArray['CN']}, OU={$issuerArray['OU']}, O={$issuerArray['O']}, C={$issuerArray['C']}";
echo "X509IssuerName: " . $issuerName . PHP_EOL;

// Get the certificate issuer name
echo "Issuer Name: " . $issuerName . PHP_EOL;
```

Figure 6.29 Calculate the X509IssuerName

```php
// Extract the subject array (X509SubjectName) and format it
$subjectArray = isset($certData['subject']) ? $certData['subject'] : [];
$order = [
    'E' => 'emailAddress', // emailAddress → E
    'SERIALNUMBER' => 'serialNumber', // serialNumber → SERIALNUMBER
    'CN' => 'CN',
    'OU' => 'OU',
    'OID.2.5.4.97' => 'organizationIdentifier', // organizationIdentifier → OID.2.5.4.97
    'O' => 'O',
    'C' => 'C'
];

$subjectParts = [];
foreach ($order as $label => $key) {
    if (isset($subjectArray[$key])) {
        $subjectParts[] = "$label={$subjectArray[$key]}";
    }
}
$subjectName = implode(', ', $subjectParts);
echo "X509SubjectName: " . $subjectName . PHP_EOL;
```

Figure 6.30 Calculate the X509SubjectName

```php
//4.1b - Calculate the SigningTime (Note: make sure the SigningTime in the chunk is later than document submission time)

// Set the SigningTime to a time slightly later than the current time
echo '<br>';
$signingTime = gmdate('Y-m-d\TH:i:s\Z');
echo "Signing Time: " . $signingTime . PHP_EOL;
```

Figure 6.31 Calculate SigningTime

```
//4.2 - Obtain the qualifyingProperties properties in json string format > minify the string > compute the hashed of the signed properties > covert to base64
// Create an array and populate the qualifying properties with the calculated values
$qualifyingProperties = [
    "QualifyingProperties" => [
    [
    "Target" => "signature",
    "SignedProperties" => [
        [
            "Id" => "id-xades-signed-props",
            "SignedSignatureProperties" => [
                [
                    "SigningTime" => [
                        ["_" => $signingTime]
                    ],
                    "SigningCertificate" => [
                        [
                            "Cert" => [
                                [
                                    "CertDigest" => [
                                        [
                                            "DigestMethod" => [
                                                ["_" => "", "Algorithm" => "http://www.w3.org/2001/04/xmlenc#sha256"]
                                            ],
                                            "DigestValue" => [
                                                ["_" => $certHashBase64]
                                            ]
                                        ]
                                    ],
                                    "IssuerSerial" => [
                                        [
                                            "X509IssuerName" => [
                                                ["_" => $issuerName]
                                            ],
                                            "X509SerialNumber" => [
                                                ["_" => $serialNumber]
                                                ]
                                                ]
                                            ]
                                        ]
                                    ]
                                ]
                            ]
                        ]
                    ]
                ]
            ]
        ]
    ]
];
```

Figure 6.32 Populate Properties into QualifyingProperties

The SignedProperties element contains structured claims for the signature, including SigningTime, CertDigest, X509IssueName and X509SerialNumber references. The function will then constructs a JSON code block called the QualifyingProperties containing the computed document properties. This code block is also then canonicalized and hashed separately to produce the PropsDigest.

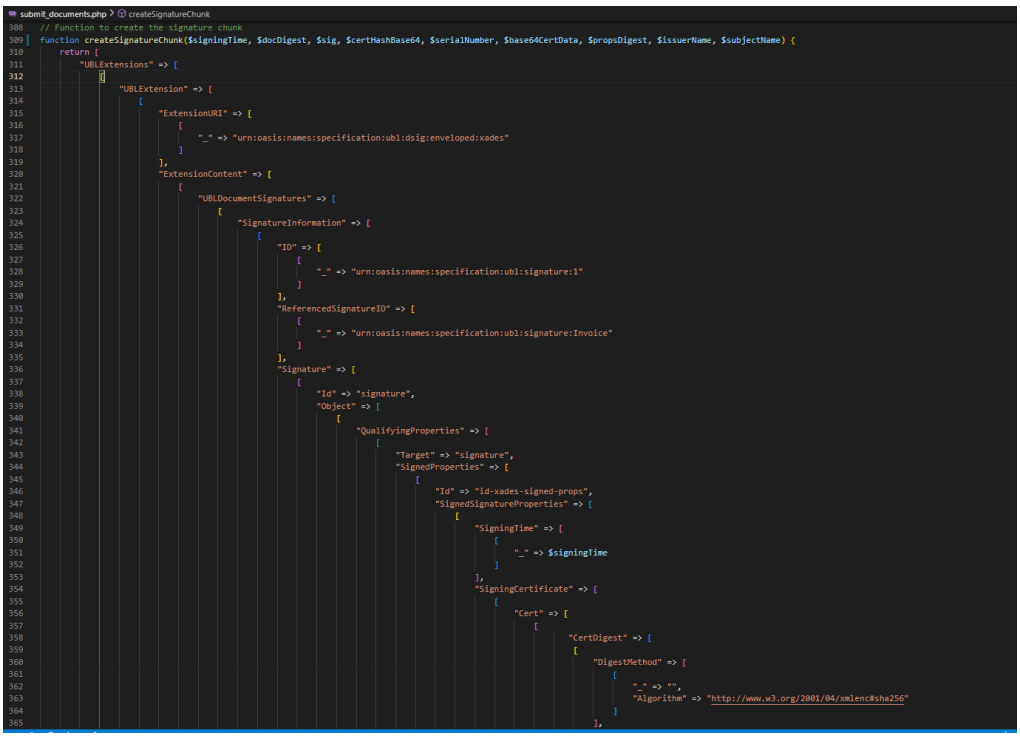
6.4.3.1.8 Generate Signed Properties Hash

```php
//Extract the SignedProperties Block Only
$signedProperties = $qualifyingProperties["QualifyingProperties"][0];
echo "Signed Properties: " . json_encode($signedProperties, JSON_PRETTY_PRINT) . PHP_EOL;
//Convert to Compact JSON (Linearization)
$linearizedJson = json_encode($signedProperties, JSON_UNESCAPED_SLASHES);
echo "Linearized JSON: " . $linearizedJson . PHP_EOL;
//Generate the sha256 hash
$linearJsonHash = hash("sha256", $linearizedJson, true);  // Output in raw binary
echo "Hash: " . $linearJsonHash . PHP_EOL;
//Encode the hash in base64
$propsDigest = base64_encode($linearJsonHash);

// Debugging Output
echo "Signed Properties: " . json_encode($signedProperties, JSON_PRETTY_PRINT) . PHP_EOL;
echo "Linearized JSON: " . $linearizedJson . PHP_EOL;
echo "PropsDigest: " . $propsDigest . PHP_EOL;
```

Figure 6.33 Generate propsDigest

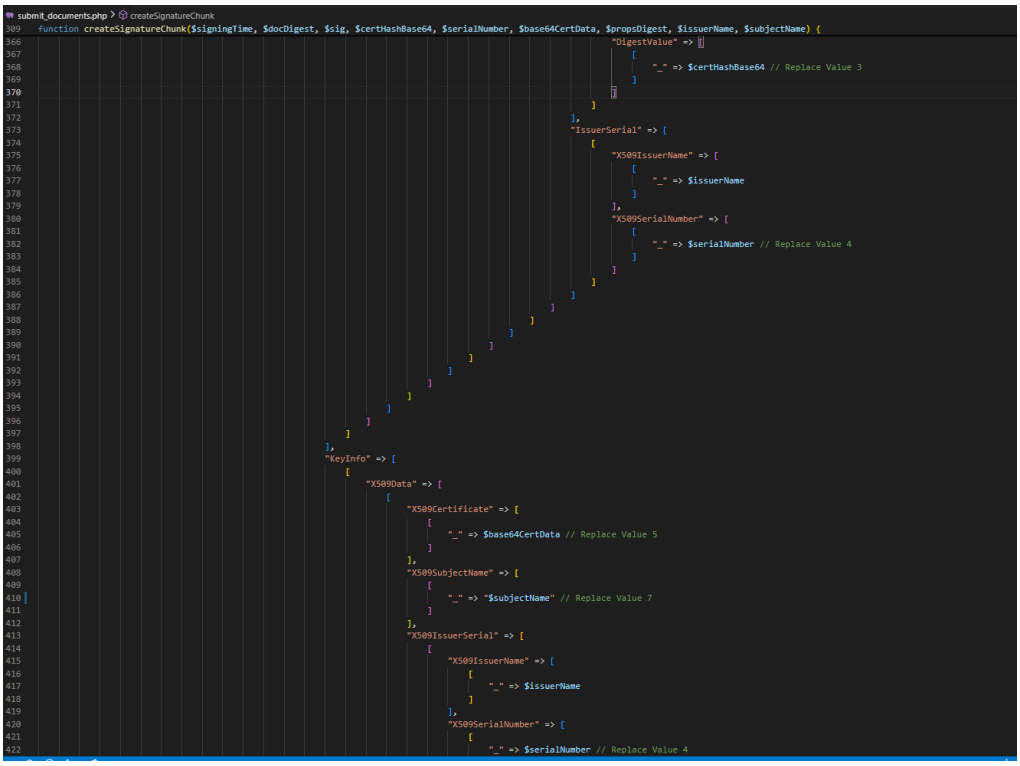
For this step, the system extracts <SignedProperties> element from the JSON document. Once the properties tag is obtained, the JSON document is linearized to ensure a consistent structure and all unnecessary spaces are removed. The linearized block is then hashed using the SHA-256 algorithm to produce a fixed-length digest. This digest is subsequently encoded using a HEX-to-Base64 encoder, and the resulting value is stored as the property named PropsDigest, which serves as a critical input for the signing process.

6.4.3.1.9 Populate the Information in Document to Create Signed Document.



```
submit_documents.php > createSignatureChunk
// Function to create the signature chunk
function createSignatureChunk($signingTime, $docDigest, $sig, $certHashBase64, $serialNumber, $base64CertData, $propsDigest, $issuerName, $subjectName) {
    return [
        "UBLExtensions" => [
            [
                "UBLExtension" => [
                    [
                        "ExtensionURI" => [
                            [
                                "_" => "urn:oasis:names:specification:ubl:dsig:enveloped:xades"
                            ]
                        ],
                        "ExtensionContent" => [
                            [
                                "UBLDocumentSignatures" => [
                                    [
                                        "SignatureInformation" => [
                                            [
                                                "ID" => [
                                                    [
                                                        "_" => "urn:oasis:names:specification:ubl:signature:1"
                                                    ]
                                                ],
                                                "ReferencedSignatureID" => [
                                                    [
                                                        "_" => "urn:oasis:names:specification:ubl:signature:Invoice"
                                                    ]
                                                ],
                                                "Signature" => [
                                                    [
                                                        "Id" => "signature",
                                                        "Object" => [
                                                            [
                                                                "QualifyingProperties" => [
                                                                    [
                                                                        "Target" => "signature",
                                                                        "SignedProperties" => [
                                                                            [
                                                                                "Id" => "id-xades-signed-props",
                                                                                "SignedSignatureProperties" => [
                                                                                    [
                                                                                        "SigningTime" => [
                                                                                            [
                                                                                                "_" => $signingTime
                                                                                            ]
                                                                                        ],
                                                                                        "SigningCertificate" => [
                                                                                            [
                                                                                                "Cert" => [
                                                                                                    [
                                                                                                        "CertDigest" => [
                                                                                                            [
                                                                                                                "DigestMethod" => [
                                                                                                                    [
                                                                                                                        "_" => "",
                                                                                                                        "Algorithm" => "http://www.w3.org/2001/04/xmlenc#sha256"
                                                                                                                    ]
                                                                                                                ],
```

Figure 6.34 Populate Properties into SignatureChunk 1



```
submit_documents.php > createSignatureChunk
function createSignatureChunk($signingTime, $docDigest, $sig, $certHashBase64, $serialNumber, $base64CertData, $propsDigest, $issuerName, $subjectName) {
                                                                                "DigestValue" => [
                                                                                    [
                                                                                        "_" => $certHashBase64 // Replace Value 3
                                                                                    ]
                                                                                ]
                                                                            ]
                                                                        ],
                                                                        "IssuerSerial" => [
                                                                            [
                                                                                "X509IssuerName" => [
                                                                                    [
                                                                                        "_" => $issuerName
                                                                                    ]
                                                                                ],
                                                                                "X509SerialNumber" => [
                                                                                    [
                                                                                        "_" => $serialNumber // Replace Value 4
                                                                                    ]
                                                                                ]
                                                                            ]
                                                                        ]
                                                                    ]
                                                                ]
                                                            ]
                                                        ],
                                                        "KeyInfo" => [
                                                            [
                                                                "X509Data" => [
                                                                    [
                                                                        "X509Certificate" => [
                                                                            [
                                                                                "_" => $base64CertData // Replace Value 5
                                                                            ]
                                                                        ],
                                                                        "X509SubjectName" => [
                                                                            [
                                                                                "_" => "$subjectName" // Replace Value 7
                                                                            ]
                                                                        ],
                                                                        "X509IssuerSerial" => [
                                                                            [
                                                                                "X509IssuerName" => [
                                                                                    [
                                                                                        "_" => $issuerName
                                                                                    ]
                                                                                ],
                                                                                "X509SerialNumber" => [
                                                                                    [
                                                                                        "_" => $serialNumber // Replace Value 4
```

Figure 6.35 Figure 6.29 Populate Properties into SignatureChunk 2

Figure 6.36 Populate Properties into SignatureChunk 3

Figure 6.37 Populate Properties into SignatureChunk 4

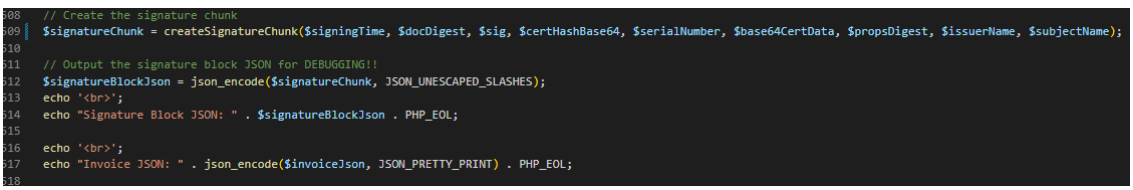
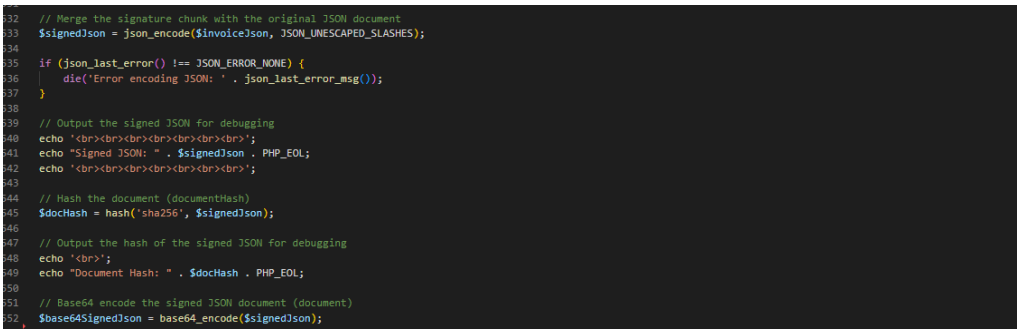


Figure 6.38 Create Signature Chunk



```
// Merge the signature chunk with the original JSON document
$signedJson = json_encode($invoiceJson, JSON_UNESCAPED_SLASHES);

if (json_last_error() !== JSON_ERROR_NONE) {
    die('Error encoding JSON: ' . json_last_error_msg());
}

// Output the signed JSON for debugging
echo '<br><br><br><br><br><br><br>';
echo "Signed JSON: " . $signedJson . PHP_EOL;
echo '<br><br><br><br><br><br><br>';

// Hash the document (documentHash)
$docHash = hash('sha256', $signedJson);

// Output the hash of the signed JSON for debugging
echo '<br>';
echo "Document Hash: " . $docHash . PHP_EOL;

// Base64 encode the signed JSON document (document)
$base64SignedJson = base64_encode($signedJson);
```

Figure 6.39 Generate Signed Document

Next, populate properties such as Sig, signingTime, certHashBase64, base64CertData, propsDigest, docDigest, serialNumber, issuerName and subjectName into signatureChunk. Then, merge the signatureChunk with the original document. After that, use the SHA-256 algorithm to hash the document to get docHash. Lastly, base-64 encode the document to get the completed signed JSON document.

6.4.3.2 Invoice Submission

submit_documents.php M

submit_documents.php > ...

```php
// Create the submission payload
$payload = [
    "documents" => [
        [
            "format" => "JSON",
            "documentHash" => $docHash,  //The hash value of the document being submitted.
            "codeNumber" => "JSON-INV12345",
            "document" => $base64SignedJson // The base64 of the document JSON or XML
        ]
    ]
];

// Echo the payload for debugging
echo '<br>';
echo "Payload: " . json_encode($payload, JSON_PRETTY_PRINT) . PHP_EOL;

// Function to submit the document
function submitDocument($payload) {
    $accessToken = getOAuth2Token();
    if (!$accessToken) {
        die('Access token not found.');
    }

    $ch = curl_init(SUBMIT_DOCUMENTS_URL);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_POST, true);
    curl_setopt($ch, CURLOPT_HTTPHEADER, [
        'Content-Type: application/json',
        'Authorization: Bearer ' . $accessToken,
        'Accept-Language: en',
        'Accept: application/json'
    ]);
    curl_setopt($ch, CURLOPT_POSTFIELDS, json_encode($payload));

    $response = curl_exec($ch);

    if (curl_errno($ch)) {
        die('Request Error: ' . curl_error($ch));
    }

    curl_close($ch);

    return $response;
}

// Submit the signed document to the API endpoint
echo '<br>';
$response = submitDocument($payload);

// Output Response
echo '<br>';
echo "API Response: " . $response;

// Store the uuid and submissionUID in the session
$responseData = json_decode($response, true);
if (isset($responseData['submissionUid']) && isset($responseData['acceptedDocuments'][0]['uuid'])) {
    $_SESSION['uuid'] = $responseData['acceptedDocuments'][0]['uuid'];
    $_SESSION['submissionUid'] = $responseData['submissionUid'];
    echo '<br>';
    echo "UUID: " . $responseData['acceptedDocuments'][0]['uuid'] . PHP_EOL;
    echo "SubmissionUID: " . $responseData['submissionUid'] . PHP_EOL;
} else {
    echo '<br>';
    die('UUID or SubmissionUID not found in the response.');
}
?>
```

Figure 6.40 Invoke Submit Document API

The completed JSON file is now ready for submission. These are coordinated by a file named submit_documents.php, which builds the final payload made of base64Document, documentHash, and other submission metadata. Upon completion, a call to the Submit Documents API will be executed to submit

the hashed document with the digital signature through the construction of a payload. The payload generally has a base64-encoded signed JSON document and a documentHash. Upon successful implementation, the expected output will be the uuid and submission uuid. It will be stored in the tbInvoice table in the database for future retrieval.

6.4.3.3 Invoice Validation

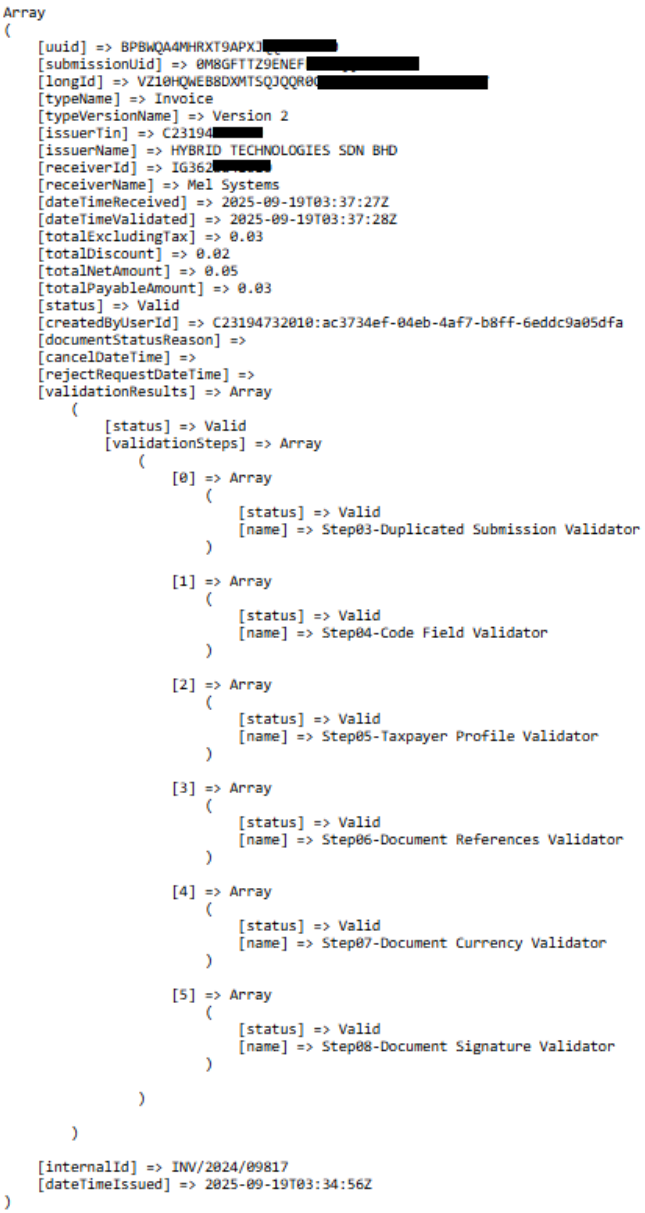
getSubmission.php > getDocumentDetails

```php
<?php

/*  This php file is created in order to validate the document details before proceeding in generating the QR code.

    For further details, do refer to the following links:
    1. https://sdk.myinvois.hasil.gov.my/einvoicingapi/08-get-document-details/
    2. https://sdk.myinvois.hasil.gov.my/standard-header-parameters/
*/

session_start(); // start the session
include 'oauth2_token.php';

// Function to get the cached access token from token_cache.json
function getAccessToken() {
    $tokenCachePath = __DIR__ . DIRECTORY_SEPARATOR . 'token_cache.json';
    if (file_exists($tokenCachePath)) {
        $tokenData = json_decode(file_get_contents($tokenCachePath), true);
        if (isset($tokenData['access_token'])) {
            return $tokenData['access_token'];
        }
    }
    return null;
}

function getDocumentDetails($uuid, $suppressOutput = false) {
    $accessToken = getAccessToken();
    if (!$accessToken) {
        die('Access token not found.');
    }
    // Make sure you are using the correct environment for the UUID you submitted
    // If you submitted to production, use production endpoint; if sandbox, use sandbox endpoint

    // For production:
    $url = "https://api.myinvois.hasil.gov.my/api/v1.0/documents/{$uuid}/details";
    // For sandbox:
    // $url = "https://preprod-api.myinvois.hasil.gov.my/api/v1.0/documents/{$uuid}/details";

    if (!$suppressOutput) {
        echo "DEBUG: GetDocumentDetails URL: $url\n";
        echo "DEBUG: Access Token: $accessToken\n";
    }

    $ch = curl_init($url);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_HTTPHEADER, [
        'Content-Type: application/json',
        'Authorization: Bearer ' . $accessToken
    ]);

    $response = curl_exec($ch);

    if (curl_errno($ch)) {
        die('Request Error: ' . curl_error($ch));
    }

    curl_close($ch);

    $responseData = json_decode($response, true);

```

Figure 6.41 Invoke Get Submissions API

```php
if (empty($uuid) && isset($_GET['invoiceID']) && is_numeric($_GET['invoiceID'])) {
    $invoiceID = (int)$_GET['invoiceID'];
    // include DB connection and attempt to read tbinvoice.invoice_uuid
    @include 'connectionImes.php';
    if (isset($mysql_host)) {
        $conn = @mysqli_connect($mysql_host, $mysql_user, $mysql_password, $mysql_database);
        if ($conn) {
            // Check that invoice_uuid column exists
            $resCols = @mysqli_query($conn, "SHOW COLUMNS FROM tbinvoice LIKE 'invoice_uuid'");
            if ($resCols && mysqli_num_rows($resCols) > 0) {
                $sql = "SELECT invoice_uuid FROM tbinvoice WHERE invoice_id=" . $invoiceID . " LIMIT 1";
                $r = @mysqli_query($conn, $sql);
                if ($r && mysqli_num_rows($r) > 0) {
                    $row = mysqli_fetch_assoc($r);
                    if (!empty($row['invoice_uuid'])) {
                        $uuid = $row['invoice_uuid'];
                        // store in session for downstream use
                        $_SESSION['uuid'] = $uuid;
                    }
                }
            }
            @mysqli_close($conn);
        }
    }
}

if (!empty($uuid)) {
    echo "UUID: " . $uuid . "<br>";
    $documentDetails = getDocumentDetails($uuid, false);
    echo '<pre>';
    print_r($documentDetails);
    echo '</pre>';

    // Upon successful retrieval of doc details, can retrieve the longId
    if (isset($documentDetails['longId'])) {
        $_SESSION['longId'] = $documentDetails['longId'];
        echo "longId saved in session: " . $_SESSION['longId'] . "<br>";
    } else {
        echo "longId not found in response.<br>";
    }
} else {
    die('UUID not found (GET uuid, session, or invoiceID lookup).');
}
```

Figure 6.42 Retrieve longId



```
Array
(
    [uuid] => BPBWQA4MHRXT9APXJ[redacted]
    [submissionUid] => 0M8GFTTZ9ENEF[redacted]
    [longId] => VZ10HQWEB8DXMTSQJQQR0[redacted]
    [typeName] => Invoice
    [typeVersionName] => Version 2
    [issuerTin] => C23194[redacted]
    [issuerName] => HYBRID TECHNOLOGIES SDN BHD
    [receiverId] => IG362[redacted]
    [receiverName] => Mel Systems
    [dateTimeReceived] => 2025-09-19T03:37:27Z
    [dateTimeValidated] => 2025-09-19T03:37:28Z
    [totalExcludingTax] => 0.03
    [totalDiscount] => 0.02
    [totalNetAmount] => 0.05
    [totalPayableAmount] => 0.03
    [status] => Valid
    [createdByUserId] => C23194732010:ac3734ef-04eb-4af7-b8ff-6eddc9a05dfa
    [documentStatusReason] =>
    [cancelDateTime] =>
    [rejectRequestDateTime] =>
    [validationResults] => Array
        (
            [status] => Valid
            [validationSteps] => Array
                (
                    [0] => Array
                        (
                            [status] => Valid
                            [name] => Step03-Duplicated Submission Validator
                        )

                    [1] => Array
                        (
                            [status] => Valid
                            [name] => Step04-Code Field Validator
                        )

                    [2] => Array
                        (
                            [status] => Valid
                            [name] => Step05-Taxpayer Profile Validator
                        )

                    [3] => Array
                        (
                            [status] => Valid
                            [name] => Step06-Document References Validator
                        )

                    [4] => Array
                        (
                            [status] => Valid
                            [name] => Step07-Document Currency Validator
                        )

                    [5] => Array
                        (
                            [status] => Valid
                            [name] => Step08-Document Signature Validator
                        )

                )

        )

    [internalId] => INV/2024/09817
    [dateTimeIssued] => 2025-09-19T03:34:56Z
)
```

Figure 6.43 Get Submissions API Output

Validation is performed by invoking LHDN's Get Submissions API. After submission, asynchronous polling is performed using getSubmission.php with submissionUid or uuid to verify if the document is accepted or rejected. Then, the current status of the e-invoice will be updated to the local invoice record, and the available longId is used for generating the QR code.

6.4.4 Generate QR Code

generateQR_raw.php > ...

```php
<?php
// Return QR code (base64 PNG) for a specific invoice (by id)
if (session_status() !== PHP_SESSION_ACTIVE) { session_start(); }
header('Content-Type: application/json');
require 'vendor/autoload.php';
require_once 'connectionImes.php';
use Endroid\QrCode\Builder\Builder;
use Endroid\QrCode\Encoding\Encoding;
use Endroid\QrCode\ErrorCorrectionLevel;
use Endroid\QrCode\Label\LabelAlignment;

$envBaseUrl = 'https://myinvois.hasil.gov.my';

// Inputs
$invoiceID = isset($_GET['invoiceID']) ? (int)$_GET['invoiceID'] : 0;
$noSession = isset($_GET['noSession']) && $_GET['noSession'] == '1';

// Initialize
$uuid = '';
$longId = '';
$debug = [
    'source' => null,
    'invoiceID' => $invoiceID,
];

// prefer DB query when invoiceID provided
if ($invoiceID > 0) {
    $debug['source'] = 'db';
    $conn = @mysqli_connect($mysql_host,$mysql_user,$mysql_password,$mysql_database);
    if($conn){
        $hasUuid = @mysqli_query($conn,"SHOW COLUMNS FROM tbinvoice LIKE 'invoice_uuid'");
        $hasLong = @mysqli_query($conn,"SHOW COLUMNS FROM tbinvoice LIKE 'invoice_longId'");
        if($hasUuid && mysqli_num_rows($hasUuid)>0 && $hasLong && mysqli_num_rows($hasLong)>0){
            $res = @mysqli_query($conn, "SELECT invoice_uuid, invoice_longId FROM tbinvoice WHERE invoice_id=".$invoiceID." LIMIT 1");
            if($res && mysqli_num_rows($res)>0){
                $row = mysqli_fetch_assoc($res);
                $uuid = trim((string)($row['invoice_uuid'] ?? ''));
                $longId = trim((string)($row['invoice_longId'] ?? ''));
                $debug['db_row'] = $row;
            } else {
                $debug['db_note'] = 'No tbinvoice row for id';
            }
        } else {
            $debug['db_note'] = 'invoice_uuid or invoice_longId column not found';
        }
        @mysqli_close($conn);
    } else {
        $debug['db_error'] = 'DB connection failed';
    }
    if(($uuid === '' || $longId === '') && !$noSession){
        $debug['fallback'] = 'session';
        $uuid = $_SESSION['uuid'] ?? '';
        $longId = $_SESSION['longId'] ?? '';
    }
} else {
    // Session path
    $debug['source'] = 'session';
    $uuid = $_SESSION['uuid'] ?? '';
    $longId = $_SESSION['longId'] ?? '';
}

if($uuid === '' || $longId === '') {
    echo json_encode(['status'=>'error','message'=>'Missing uuid or longId','uuid'=>$uuid,'longId'=>$longId,'debug'=>$debug]);
    exit;
}

// Build QR URL (base env + /{uuid}/share/{longId})
$link = rtrim($envBaseUrl, '/').'/'.rawurlencode($uuid).'/share/'.rawurlencode($longId);
```

Figure 6.44 Build the Link to the QR Code

```php
// Build QR
$builder = new Builder(
    writer: new \Endroid\QrCode\Writer\PngWriter(),
    data: $link,
    encoding: new Encoding('UTF-8'),
    errorCorrectionLevel: ErrorCorrectionLevel::High,
    size: 300,
    margin: 10,
    labelText: 'Scan to verify',
    labelFont: new \Endroid\QrCode\Label\Font\Font(__DIR__.'/vendor/endroid/qr-code/assets/open_sans.ttf', 14),
    labelAlignment: LabelAlignment::Center
);
$result = $builder->build();
$base64 = base64_encode($result->getString());
echo json_encode([
    'status'=>'success',
    'uuid'=>$uuid,
    'longId'=>$longId,
    'qrBase64'=>$base64,
    'url'=>$link,
    'debug'=>$debug,
]);
?>
```

Figure 6.45 Build the QR Code

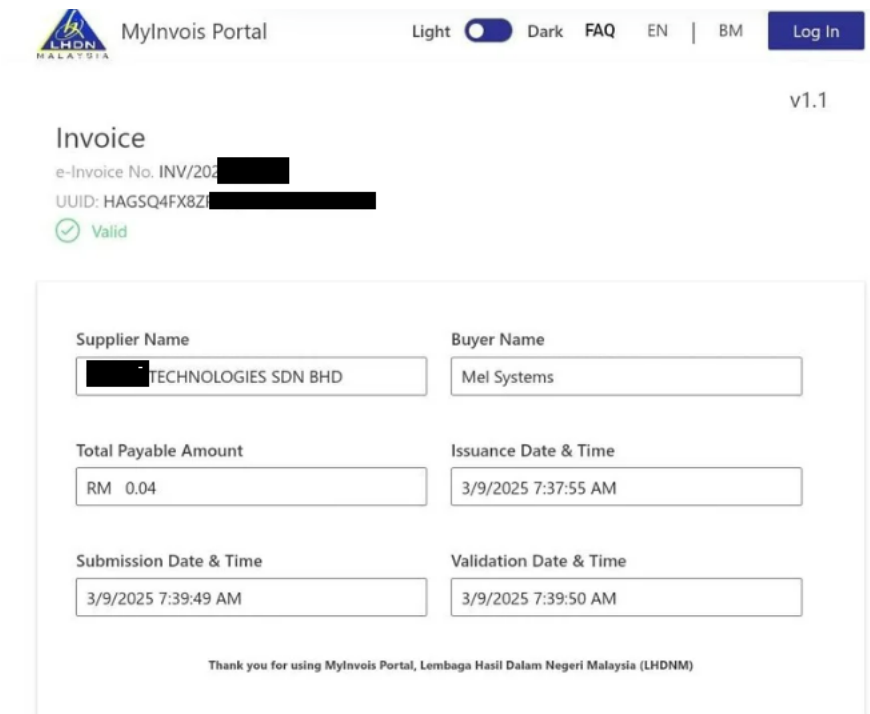


Figure 6.46 Validated e-Invoice

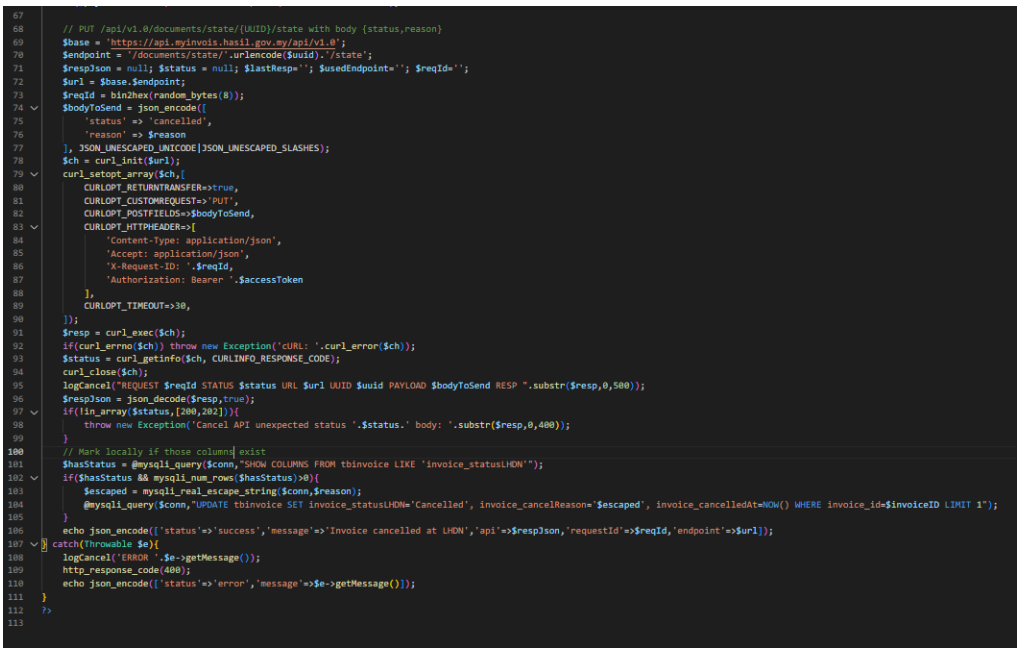
The generateQR_raw.php file creates a verification URL that is constructed by appending the base_url with the uuid and longId that were previously retrieved as an output from the Get Submissions API. Besides that, this script also utilizes third-party libraries such as the Endroid QR library in facilitating QR generation. As a result, the successfully generated QR code will then be appended to the invoice in PDF format. With this, the e-invoice has been

successfully generated. As seen in the figure above, the e-invoice transaction status is fully validated by LHDN upon scanning the QR code.

6.4.5 Cancel Document

```php
header('Content-Type: application/json');
if(session_status()===PHP_SESSION_NONE){ session_start(); }
include 'connectionImes.php';
ini_set('display_errors','0');

$logFile = __DIR__.'/cancel_error.log';
function logCancel($msg){ global $logFile; @file_put_contents($logFile, date('c')." | ".$msg."\n", FILE_APPEND); }

set_error_handler(function($no,$str,$file,$line){
    logCancel("PHPERROR $str @ $file:$line");
});

function db(){
    global $mysql_host,$mysql_user,$mysql_password,$mysql_database;
    return @mysqli_connect($mysql_host,$mysql_user,$mysql_password,$mysql_database);
}

try {
    if($_SERVER['REQUEST_METHOD']!=='POST') throw new Exception('Invalid method');
    $invoiceID = isset($_POST['invoiceID'])?(int)$_POST['invoiceID']:0;
    if($invoiceID<=0) throw new Exception('Missing invoiceID');
    $reason = trim($_POST['reason'] ?? 'Invoice cancelled');
    $conn = db();
    if(!$conn) throw new Exception('DB connect fail');
    // If already cancelled
    $already = @mysqli_query($conn,"SHOW COLUMNS FROM tbinvoice LIKE 'invoice_statusLHDN'");
    if($already && mysqli_num_rows($already)>0){
        $stRes = @mysqli_query($conn,"SELECT invoice_statusLHDN FROM tbinvoice WHERE invoice_id=$invoiceID");
        if($stRes && ($stRow=mysqli_fetch_row($stRes)) && strcasecmp($stRow[0],'Cancelled')===0){
            echo json_encode(['status'=>'success','message'=>'Already Cancelled']);
            return;
        }
    }
    // Load uuid if not supplied
    $uuid = trim($_POST['uuid'] ?? '');
    if($uuid===''){
        $res = @mysqli_query($conn,"SHOW COLUMNS FROM tbinvoice LIKE 'invoice_uuid'");
        if($res && mysqli_num_rows($res)>0){
            $r2 = @mysqli_query($conn,"SELECT invoice_uuid FROM tbinvoice WHERE invoice_id=$invoiceID LIMIT 1");
            if($r2 && ($rw=mysqli_fetch_row($r2))){ $uuid = $rw[0]; }
        }
    }
    if($uuid==='') throw new Exception('UUID not found (ensure invoice submitted)');
    // Include token
    if(file_exists(__DIR__.'/oauth2_token.php')){
        ob_start();
        include_once __DIR__.'/oauth2_token.php';
        ob_end_clean();
    }
    $tokenCachePath = __DIR__.'/token_cache.json';
    $accessToken = null;
    if(file_exists($tokenCachePath)){
        $td = json_decode(file_get_contents($tokenCachePath),true);
        if(isset($td['access_token']) && isset($td['expires_at']) && $td['expires_at']>time()) $accessToken=$td['access_token'];
    }
    if(!$accessToken && function_exists('getOAuth2Token')){ ob_start(); $accessToken = getOAuth2Token(); ob_end_clean(); }
    if(!$accessToken) throw new Exception('Access token unavailable');
    // Build cancel payload
    $payloadArr = [ 'uuid'=>$uuid, 'reason'=>$reason ];
    // Common alternative field some specs use
    if(isset($_POST['reasonCode'])) $payloadArr['reasonCode'] = $_POST['reasonCode'];
    $payload = json_encode($payloadArr, JSON_UNESCAPED_UNICODE|JSON_UNESCAPED_SLASHES);
    if($payload===false) throw new Exception('JSON encode failed');
```

Figure 6.47 Get Details of the Specific E-Invoice to be Cancelled



```
    // PUT /api/v1.0/documents/state/{UUID}/state with body {status,reason}
    $base = 'https://api.myinvois.hasil.gov.my/api/v1.0';
    $endpoint = '/documents/state/'.urlencode($uuid).'/state';
    $respJson = null; $status = null; $lastResp=''; $usedEndpoint=''; $reqId='';
    $url = $base.$endpoint;
    $reqId = bin2hex(random_bytes(8));
    $bodyToSend = json_encode([
        'status' => 'cancelled',
        'reason' => $reason
    ], JSON_UNESCAPED_UNICODE|JSON_UNESCAPED_SLASHES);
    $ch = curl_init($url);
    curl_setopt_array($ch,[
        CURLOPT_RETURNTRANSFER=>true,
        CURLOPT_CUSTOMREQUEST=>'PUT',
        CURLOPT_POSTFIELDS=>$bodyToSend,
        CURLOPT_HTTPHEADER=>[
            'Content-Type: application/json',
            'Accept: application/json',
            'X-Request-ID: '.$reqId,
            'Authorization: Bearer '.$accessToken
        ],
        CURLOPT_TIMEOUT=>30,
    ]);
    $resp = curl_exec($ch);
    if(curl_errno($ch)) throw new Exception('cURL: '.curl_error($ch));
    $status = curl_getinfo($ch, CURLINFO_RESPONSE_CODE);
    curl_close($ch);
    logCancel("REQUEST $reqId STATUS $status URL $url UUID $uuid PAYLOAD $bodyToSend RESP ".substr($resp,0,500));
    $respJson = json_decode($resp,true);
    if(!in_array($status,[200,202])){
        throw new Exception('Cancel API unexpected status '.$status.' body: '.substr($resp,0,400));
    }
    // Mark locally if those columns exist
    $hasStatus = @mysqli_query($conn,"SHOW COLUMNS FROM tbinvoice LIKE 'invoice_statusLHDN'");
    if($hasStatus && mysqli_num_rows($hasStatus)>0){
        $escaped = mysqli_real_escape_string($conn,$reason);
        @mysqli_query($conn,"UPDATE tbinvoice SET invoice_statusLHDN='Cancelled', invoice_cancelReason='$escaped', invoice_cancelledAt=NOW() WHERE invoice_id=$invoiceID LIMIT 1");
    }
    echo json_encode(['status'=>'success','message'=>'Invoice cancelled at LHDN','api'=>$respJson,'requestId'=>$reqId,'endpoint'=>$url]);
} catch(Throwable $e){
    logCancel('ERROR '.$e->getMessage());
    http_response_code(400);
    echo json_encode(['status'=>'error','message'=>$e->getMessage()]);
}
?>
```

Figure 6.48 Invoke Cancel Documents API

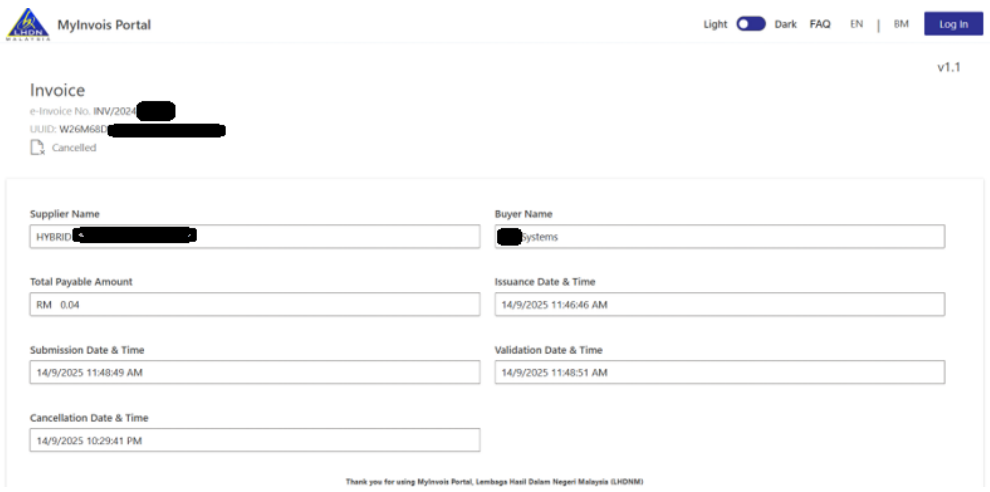


Figure 6.49 Successful Output of E-Invoice Cancellation

The cancellation feature allows suppliers to cancel previously submitted invoices within the permitted time, which is 72 hours. The cancel_invoice_trigger.php script first verifies invoice eligibility to cancel. Then, if it is legible, it will build a cancellation request payload consisting of uuid or submissionUid, cancelReason, and supplier info, and call the cancel endpoint. Upon successful cancellation, the invoice status changes to “Cancelled”. Then, upon scanning the QR code again, the e-invoice

transaction status had also been changed to “Cancelled”, indicating a successful cancellation request.

6.4.6 Implementing Ledger

ledger.php > ...

```php
<?php
/** ~ Ledger ! **/
if (session_status() !== PHP_SESSION_ACTIVE) { session_start(); }
if (!isset($_SESSION['userID']) || $_SESSION['userID'] === '') {
    ob_start(); header('Location: index.php'); ob_end_flush(); exit;
}

require_once 'connectionImes.php';
$conn = @mysqli_connect($mysql_host, $mysql_user, $mysql_password, $mysql_database) or die('unable to connect!');

// Helpers
function col_exists($conn, $table, $col){
    $t = mysqli_real_escape_string($conn,$table);
    $c = mysqli_real_escape_string($conn,$col);
    $res = @mysqli_query($conn, "SHOW COLUMNS FROM `$t` LIKE '$c'");
    return ($res && mysqli_num_rows($res) > 0);
}
function esc($s){ return htmlspecialchars((string)$s ?? '', ENT_QUOTES, 'UTF-8'); }

// to detect if column exists or not
$has_uuid = col_exists($conn,'tbinvoice','invoice_uuid');
$has_status = col_exists($conn,'tbinvoice','invoice_statusLHDN');
$has_local_status = col_exists($conn,'tbinvoice','invoice_status');
$has_submission = col_exists($conn,'tbinvoice','invoice_submissionUid');
$has_long = col_exists($conn,'tbinvoice','invoice_longId');
$has_isEinvoice = col_exists($conn,'tbinvoice','invoice_isEinvoice');
$has_dueDate = col_exists($conn,'tbinvoice','invoice_dueDate');
$has_terms = col_exists($conn,'tbinvoice','invoice_terms');
$has_grandRound = col_exists($conn,'tbinvoice','invoice_grandTotalRound');

// Customer potential TIN/BRN columns
$custTinCols = [];
foreach(['customer_TIN','customer_TINnumber','customer_tin','customerTin'] as $cand){ if(col_exists($conn,'tbcustomer',$cand)) {$custTinCols[]=$cand; break;} }
$custBrnCols = [];
foreach(['customer_BRN','customer_ROCnumber','customer_brn','customerBrn'] as $cand){ if(col_exists($conn,'tbcustomer',$cand)) {$custBrnCols[]=$cand; break;} }

// Company (supplier) fields
$company = ['id'=>1,'name'=>'','tin'=>''];
$supplierTinCol = null;
foreach(['company_TIN','company_tin','company_taxpayerId','company_taxNo'] as $cand){ if(col_exists($conn,'tbcompany',$cand)) {$supplierTinCol=$cand; break;} }
$sqlComp = "SELECT company_name" . ($supplierTinCol? ", `$supplierTinCol`" : "") . " FROM tbcompany WHERE company_id=1 LIMIT 1";
$resComp = @mysqli_query($conn,$sqlComp);
if($resComp && mysqli_num_rows($resComp)>0){
    $row = mysqli_fetch_row($resComp);
    $company['name'] = $row[0];
    $company['tin'] = $supplierTinCol && isset($row[1]) ? $row[1] : '';
}

// Date filters
$from = isset($_GET['from']) ? $_GET['from'] : '';
$to = isset($_GET['to']) ? $_GET['to'] : '';
$eonly = isset($_GET['eonly']) && $_GET['eonly'] === '1';
// Expect format YYYY-MM-DD; sanitize
if($from && !preg_match('/^\d{4}-\d{2}-\d{2}$/',$from)) $from='';
if($to && !preg_match('/^\d{4}-\d{2}-\d{2}$/',$to)) $to='';

$where = [];
if($from){ $where[] = "inv.invoice_date >= '".mysqli_real_escape_string($conn,$from)."'"; }
if($to){ $where[] = "inv.invoice_date <= '".mysqli_real_escape_string($conn,$to)."'"; }
// Exclude locally voided invoices (invoice_status='c')
if($has_local_status){ $where[] = "(inv.invoice_status <> 'c' OR inv.invoice_status IS NULL)"; }
$eInvConds = [];
```

Figure 6.50 Ledger Feature 1

```php
ledger.php > ...

63
64  if($eonly){
65      if($has_isEinvoice){
66          $eInvConds[] = 'inv.invoice_isEinvoice = 1';
67      } else {
68          // consider as e-Invoice if UUID/status/longId present (e-invoice characteristics as compared to regular invoice)
69          $parts = [];
70          if($has_uuid){ $parts[] = "inv.invoice_uuid IS NOT NULL AND inv.invoice_uuid<>''"; }
71          if($has_status){ $parts[] = "inv.invoice_statusLHDN IS NOT NULL AND inv.invoice_statusLHDN<>''"; }
72          if($has_long){ $parts[] = "inv.invoice_longId IS NOT NULL AND inv.invoice_longId<>''"; }
73          if(!empty($parts)) $eInvConds[] = '('.implode(' OR ',$parts).')';
74      }
75  }
76  $allConds = array_merge($where, $eInvConds);
77  $whereSql = empty($allConds) ? '' : (' WHERE '.implode(' AND ',$allConds));
78
79  $sel = [
80      'inv.invoice_id','inv.invoice_no','inv.invoice_date','inv.invoice_customerID',
81      'inv.invoice_subTotal','inv.invoice_taxTotal','inv.invoice_grandTotal'
82  ];
83  if($has_grandRound){ $sel[]='inv.invoice_grandTotalRound'; }
84  if($has_dueDate){ $sel[]='inv.invoice_dueDate'; }
85  if($has_terms){ $sel[]='inv.invoice_terms'; }
86  if($has_uuid){ $sel[]='inv.invoice_uuid'; }
87  if($has_status){ $sel[]='inv.invoice_statusLHDN'; }
88  if($has_local_status){ $sel[]='inv.invoice_status AS invoice_status_local'; }
89  if($has_submission){ $sel[]='inv.invoice_submissionUid'; }
90  if($has_long){ $sel[]='inv.invoice_longId'; }
91
92  // Customer columns
93  $selCust = ['c.customer_id','c.customer_name'];
94  if(!empty($custTinCols)){ $selCust[] = 'c.`'.$custTinCols[0].'` AS customer_tin'; }
95  if(!empty($custBrnCols)){ $selCust[] = 'c.`'.$custBrnCols[0].'` AS customer_brn'; }
96
97  // SUM
98  $arAgg = "SELECT account4_documentTypeID, SUM(account4_debit) AS sum_debit, SUM(account4_credit) AS sum_credit
99            FROM tbaccount4 WHERE account4_documentType='INV' GROUP BY account4_documentTypeID";
100
101 $sql = "SELECT ".implode(',', $sel).", ".implode(',', $selCust).", ar.sum_debit, ar.sum_credit
102         FROM tbinvoice inv
103         LEFT JOIN tbcustomer c ON c.customer_id = inv.invoice_customerID
104         LEFT JOIN ($arAgg) ar ON ar.account4_documentTypeID = inv.invoice_id".
105         $whereSql.
106         " ORDER BY inv.invoice_date DESC, inv.invoice_id DESC";
107
108 $res = @mysqli_query($conn,$sql);
109
110 // Totals
111 $totals = ['subtotal'=>0,'tax'=>0,'total'=>0];
```

Figure 6.51 Ledger Feature 2

The ledger feature collects invoice and e-invoice data for reporting, tracking, and export. It extracts invoices from tbinvoice with status, date of submission of invoice, customer, or uuid and longId filter. It also allows for the calculation of the total taxable amounts, total tax paid, and grand totals. Besides that, CSV or Excel export is also supported to allow for future tracking or record-keeping purposes.

CHAPTER 7

SYSTEM TESTING AND EVALUATION

7.1 Introduction

In this chapter, numerous testing methodologies have been carried out, which include Unit Testing and Integration Testing. Aside from validating the system quality prior to its deployment to the end users, there will be a System Usability Scale (SUS) test. The objective of this test is to formally gather feedback from the users regarding their experience of using the application. With the combination of technical testing and usability testing, this not only ensures that the system works as planned, but also that it is user-friendly and functional to use in practice.

7.2 Unit Testing

The unit tests for this project are conducted manually. Particularly, the type of method to conduct this unit test is Black Box testing, which involves testing the system externally. Black Box Testing is chosen as the project involves integration of API's, and is best validated by observing the outputs and the side-effects of the system.

Name of Module	Web-based Automated E-Invoicing System
Created By	Elisabeth Lee Mei Jin
Created On	10/9/2025
Date of Review	15/9/2025

7.2.1 Test Case 1: Verification of Invoice Mapping

Table 7.1 Unit Test Case for Verification of Invoice Mapping

Test Case Id: TC001	
Scenario	Verify Invoice mapping from the text fields in the Create Invoice page properly maps to the document, which is a

	JSON document with UBL 2.1 format.
Preconditions	The system must be configured to handle valid schema mappings. This includes the invoice_mapper function.
Test Data	InvoiceNo: INV-2025-001 IssueDate: 2025-09-01 SupplierTIN: 1234567890 CustomerTIN: 0987654321 Lines: [{desc: "Service A", qty: 1, price: 100.00, taxRate: 6}]
Test Steps	1. Open the “Create Invoice” page. 2. Fill in the invoice header fields such as InvoiceNo, IssueDate, SupplierTin, and CustomerTin. 3. Add one invoice line item with its description, unit price, quantity, tax and discounts. 4. Save invoice in the database. 5. Run the mapping function in the invoice_mapper script to generate the JSON file in UBL 2.1 format. 6. Retrieve the generated JSON file.
Expected Result	All the line items and header fields are mapped to the JSON file accurately.
Actual Result	All the line items and header fields are mapped to the JSON file accurately.
Status	Pass

7.2.2 Test Case 2: Verification of Invoice Mapping with Multiple Line Items and Different Tax Rates

Table 7.2 Unit Test Case for Verification of Invoice Mapping with Multiple Line Items and Different Tax Rates

Test Case Id: TC002	
Scenario	Verify Invoice mapping handles multiple line items and different tax rates
Preconditions	The system must be configured to handle valid schema mappings. This includes the invoice_mapper function.

Test Data	Lines: - Item1: qty=2, price=50, taxRate=0 - Item2: qty=1, price=100, taxRate=6 - Item3: qty=3, price=30, taxRate=12
Test Steps	1. Create an invoice with 3 line items. 2. Ensure each item has different tax rates (0%, 6%, 12%). 3. Save invoice. 4. Run mapping function (invoice_mapper). 5. Open the generated JSON. 6. Verify each line's tax calculation and totals.
Expected Result	All invoice lines contain the correct LineExtensionAmount and TaxAmount. TaxTotal and PayableAmount are computed correctly.
Actual Result	All invoice lines contain the correct LineExtensionAmount and TaxAmount. TaxTotal and PayableAmount are computed correctly.
Status	Pass

7.2.3 Test Case 3: Verification of Mismatch in Calculation in Totals

Table 7.3 Unit Test Case for Verification of Mismatch in Calculation in Totals

Test Case Id: TC003	
Scenario	Verify Invoice mapping handles multiple line items and different tax rates
Preconditions	An invoice in JSON structure must be made available.
Test Data	Lines total = 200.00 TaxTotal = 12.00 PayableAmount = 250.00
Test Steps	1. Create invoice JSON with header and line items. 2. Ensure that PayableAmount in LegalMonetaryTotal does not equal (LineTotal + TaxTotal). 3. Submit JSON to the validation routine.

	4. Observe and capture validation results.
Expected Result	- Validator calculates expected payable = 212.00. - Validator flags mismatch: “Totals do not reconcile.” - Error message specifies the difference.
Actual Result	- Validator calculates expected payable = 212.00. - Validator flags mismatch: “Totals do not reconcile.” - Error message specifies the difference.
Status	Pass

7.2.4 Test Case 4: Verification of Canonicalization Process to Produce Deterministic Canonical JSON and Correct Digest.

Table 7.4 Unit Test Case for Verification of Canonicalization Process to Produce Deterministic Canonical JSON and Correct Digest

Test Case Id: TC004	
Scenario	Verify that the canonicalization process produces deterministic canonical JSON and a correct digest.
Preconditions	Ensure the Canonicalization function and digest calculation are made available.
Test Data	testLHDN.json
Test Steps	1. Load sample JSON file (JsonWithOutSignature.json). 2. Run the canonicalization function once. 3. Run the same function again on the same input. 4. Compare both outputs. 5. Compute the SHA-256 digest of canonicalized output. 6. Compare to the precomputed expected digest.
Expected Result	- Canonical output is identical across multiple runs. - SHA-256 digest matches expected value.
Actual Result	- Canonical output is identical across multiple runs. - SHA-256 digest matches expected value.
Status	Pass

7.2.5 Test Case 5: Verification of Handling Wrong Certificate Details

Table 7.5 Unit Test Case for Verification of Handling Wrong Certificate Details

Test Case Id: TC005	
Scenario	Certificate loader function handles invalid .p12 or wrong password associated with the password.
Preconditions	Expired .p12 file or wrong password.
Test Data	ExpiredTrialCert.p12 file or wrong password.
Test Steps	1. Provide an invalid .p12 file or the wrong password. 2. Run certificate loader. 3. Capture response and exception.
Expected Result	- Function outputs an error or returns a failure code. - No private key extracted. - No certificate details returned.
Actual Result	- Function outputs an error or returns a failure code. - No private key extracted. - No certificate details returned.
Status	Pass

7.2.6 Test Case 6: Verification of Login Process with Valid Credentials

Table 7.6 Unit Test Case for Verification of Login Process with Valid Credentials

Test Case Id: TC006	
Scenario	User logs in with Valid Credentials.
Preconditions	User account exists in database with the correct username and password.
Test Data	Username = testUser Password = Password123
Test Steps	1. Open the application login page. 2. Enter a valid username. 3. Enter the correct password.
Expected	- User is successfully authenticated.

Result	- System redirects to the home page. - Access token is generated and stored securely.
Actual Result	- User is successfully authenticated. - System redirects to the home page. - Access token is generated and stored securely.
Status	Pass

7.2.7 Test Case 7: Verification of Login Process with Invalid Credentials

Table 7.7 Unit Test Case for Verification of Login Process with Invalid Credentials

Test Case Id: TC007	
Scenario	User logs in with invalid username and password
Preconditions	User account exists in database with correct username and password.
Test Data	Username = testUser Password = wrongPassword
Test Steps	1. Open the application login page. 2. Enter a valid username. 3. Enter the incorrect password.
Expected Result	- System rejects the login attempt. - Error message “Invalid username or password” - User remains on login page.
Actual Result	- System rejects the login attempt. - Error message “Invalid username or password” - User remains on login page.
Status	Pass

7.2.8 Test Case 8: Verification of Login Process with Missing Credentials

Table 7.8 Unit Test Case for Login Process with Missing Credentials

Test Case Id: TC008	
Scenario	User logs in without entering any credentials
Preconditions	Login page is loaded
Test Data	Username = Password =
Test Steps	1. Open the application login page. 2. Leave the username and password fields blank. 3. Click the Login button.
Expected Result	System displays validation messages: “Username is required.” and “Password is required.”
Actual Result	System displays validation messages: “Username is required” and “Password is required.”
Status	Pass

7.2.9 Test Case 9: Verification of User Logout Page

Table 7.9 Unit Test Case for User Logout Page

Test Case Id: TC009	
Scenario	User logs out successfully by exiting the system
Preconditions	User is logged into the web application.
Test Data	N/A
Test Steps	1. Click on the “Exit” button 2. User is redirected to the login page.
Expected Result	1. System logs the user out 2. User session is terminated 3. Application redirects to the login page.
Actual Result	1. System logs the user out 2. User session is terminated 3. Application redirects to the login page.
Status	Pass

7.2.10 Test Case 10: Verification of Viewing QR Code for Issued e-Invoice

Table 7.10 Unit Test Case for Verification of Viewing QR Code for issued e-Invoice

Test Case Id: TC0010	
Scenario	View QR Code for a generated e-invoice
Preconditions	1. Invoice has been successfully submitted and validated. 2. uuid and longId are available for the invoice. 3. User is logged in and has permission to view invoices.
Test Data	InvoiceNo = INV-2025-001 UUID = W26M68DXTT9SKBQMB4ZD535K11 LongId=C0TVRV4CTRV36TKNB4ZD535K102ygBSP1757821729
Test Steps	1. Open the invoice list and search for a specific invoice by InvoiceNo. 2. Click on the chosen invoice to open the detail view. 3. Locate the QR code section. 4. Verify that the QR code corresponds to the correct invoice by scanning it with a QR scanner.
Expected Result	QR code for the specific e-invoice is displayed correctly on the invoice page.
Actual Result	QR code for the specific e-invoice is displayed correctly on the invoice page.
Status	Pass

7.3 Integration Testing

The Integration Testing for all the test cases are also conducted manually. Each of the integration test cases was carried out by simulating real user actions such as creating e-invoices, generating a JSON file and submitting them to the API. Then, the output will involve observing whether the expected interactions between the modules are correctly executed.

7.3.1 Integration Test Case 1: Verification of Submission of Digitally Signed invoice in JSON format to LHDN API

Table 7.11 Integration Test Case for Verification of Submission of Digitally Signed Invoice in JSON format to Submit Documents API

Test Case Id: TC001	
Scenario	Verify that a digitally signed invoice JSON can be successfully submitted to the Submit Documents API.
Preconditions	- Valid PKCS#12 certificate installed - OAuth2 access token available - Invoice JSON constructed according to LHDN schema
Test Data	Invoice: Invoice No: INV00123 Supplier Tin, Supplier BRN, email address, phone number, and address Invoice with Populated Line Items: • Line 1: qty=2, price=100, tax=6% • Line 2: qty=1, price=200, tax=0%
Test Steps	1. Create invoice JSON via invoice_mapper. 2. Sign JSON with generateSig.php. 3. Obtain a token with oauth2_token.php. 4. Submit signed JSON with submit_documents.php. 5. Poll submission status with getSubmission.php.
Expected Result	- Submission accepted - UUID and SubmissionUid generated and stored in database
Actual Result	- Submission accepted - UUID and SubmissionUid generated and stored in database
Status	Pass

7.3.2 Integration Test Case 2: Verification of Mismatch Totals Triggers Validation Errors from LHDN

Table 7.12 Integration Test Case for Mismatch Totals Triggers Validation Errors from LHDN

Test Case Id: TC002	
Scenario	Verify that mismatched totals in invoice trigger validation errors from LHDN
Preconditions	- Invoice JSON exists in tbinvoice - Schema correct, but totals mismatched
Test Data	Invoice totals: • Line items = 300.00 • TaxTotal = 18.00
Test Steps	1. Prepare invoice JSON with incorrect totals. 2. Sign JSON with generateSig.php. 3. Submit to LHDN with submit_documents.php. 4. Poll status with getSubmission.php. 5. Capture validation response.
Expected Result	- Validation fails - Error: “Totals do not reconcile.” - PayableAmount = 350.00
Actual Result	- Error received: “Totals do not reconcile.”
Status	Pass

7.3.3 Integration Test Case 3: Verification of System Behavior when Submitting Invoice after OAuth2 Token Expiry

Table 7.13 Integration Test Case for Verification of System Behavior when Submitting Invoice after OAuth2 Token Expiry

Test Case Id: TC003	
Scenario	Verify system behavior when submitting invoice after OAuth2 token expiry
Preconditions	- Valid invoice JSON available - Expired OAuth2 token cached in token_cache.json
Test Data	Invoice: • Invoice No: INV00234 • Expired access token in cache
Test Steps	1. Attempt invoice submission with expired token. 2. Observe LHDN response. 3. Refresh token using oauth2_token.php. 4. Resubmit invoice. 5. Poll submission status.
Expected Result	- First attempt fails with “401 Unauthorized/Token expired.” - System retrieves new token. - Resubmission succeeds - Status updated as “Submitted”
Actual Result	- Token expired detected - New token acquired - Invoice submitted successfully
Status	Pass

7.4 System Usability Scale (SUS)

The System Usability Scale is an evaluation tool with its main objective being, to derive understanding and feedback from users regarding their overall experience of using the web-based automated e-invoicing application. To determine the usability of the system, a structured evaluation approach is applied. The aim is to achieve usability readiness beyond the industry threshold score of 68 by analyzing feedback scores (Thomas, 2020). For this project, the system usability scale (SUS) was conducted through handing out forms. The results of the forms can be viewed in the Appendix. Hence, this usability scale serves as a great indicator as a measurement of the usability and functionality of the system.

Table 7.14 System Usability Scale (SUS) Questionnaire

		Strongly Disagree	Somewhat Agree	Neutral	Somewhat Agree	Strongly Agree
1	I think I would use this system frequently to submit invoices for e-invoice generation.					
2	I found the process of submitting invoices to generate e-invoices unnecessarily complex.					
3	I thought the system was easy					

	to use.					
4	I think I would need the support of a technical person to onboard with the e-invoicing system effectively.					
5	I found the various features werewell integrated into the system.					
6	I thought there was too much inconsistency in how the system handled invoice tasks.					
7	I would imagine that most peoplewould learn to use this e-invoicing system very quickly.					
8	I found the system to be cumbersome when performing invoice-related tasks.					

9	I felt very confident using the system to submit invoices and generate e-invoices.					
10	I needed to equip myself with sufficient knowledge to use the e-invoice system effectively.					

The collected responses are then converted and combined to evaluate the usability of the project using the calculation method below:

1. The calculation begins with calculating the odd-numbered items. For the odd-numbered items, the user's score will be reduced by 1, while for even-numbered items, subtract the score from 5.
2. Sum all the adjusted values to obtain the total score.
3. Lastly, multiply the total by 2.5 to scale the results within the range of 0-100.

7.4.1 Result of the System Usability Scale (SUS)

Table 7.15 System Usability Scale (SUS) Results

Question Number	Users					Average Score
	User 1	User 2	User 3	User 4	User 5	
1.	3	4	4	3	3	3.4/4.0
2.	4	4	4	4	4	4.0/4.0
3.	4	4	4	4	4	4.0/4.0

4.	2	3	1	1	2	1.8/4.0
5.	4	4	4	2	4	3.6/4.0
6.	3	3	3	3	3	3.0/4.0
7.	4	4	4	4	4	4.0./4.0
8.	3	4	4	4	3	3.6/4.0
9.	4	4	3	2	4	3.4/4.0
10.	4	2	3	4	4	3.4/4.0
Total	35	36	34	31	35	34.2/40
SUS Score	87.5	90	85	77.5	87.5	85.5

As seen in the table above, the calculated System Usability Score (SUS) is 85.5, which is higher than the industry standard of 68. This demonstrates that the system is satisfactory for regular users and satisfies the objective of the project. The individual results will be appended in the appendix section.

CHAPTER 8

CONCLUSION AND RECOMMENDATIONS

8.1 Conclusion

The expected outcome of this project is mainly on integrating the LHDN API's into the automated e-invoicing web application. This would eventually allow taxpayers to submit documents, validate, cancel documents, and perform reporting analysis through the ledger feature. The implementation of an e-invoicing system integrated with the LHDN API has been an insightful and filled with complexities as it simulates the real-world circumstances of building systems under regulatory constraints. During the course of this project, there were many technical and operational problems that occurred, which significantly affected both the development timeline and the methods employed to achieve the goals of implementing the project.

8.2 Project Challenges and Solutions

One of the key findings is that the complexity of API integration into the system is very high, as the system places major emphasis on compliance with LHDN's standards and regulations, particularly due to the fact that it involves sensitive data such as financial transactions. Therefore, the requirement for digital signatures, which ensures not only authenticity and integrity of e-invoices, is implemented as a way to safeguard data. This in turn, demanded not only advanced programming but also solid fundamentals of cybersecurity and cryptographic principles. Unlike standard application development, such integration without in-depth knowledge is not possible, since security implementation errors can potentially cause deviation from compliance standards. Hence, I had equipped myself with fundamentals in cybersecurity principles and cryptographic operations such as SHA-256 hashing, private key extraction and canonicalization from forums and websites before starting this project.

Another key takeaway is that cost considerations are unavoidable when moving from local prototypes to a production-ready web application. Specifically, the web-based e-invoicing system requires the use of a production certification in order to perform digital signing of invoices. While the project did utilize sandbox certificates during the early stage of development, transitioning from a local environment to a production environment still incurs financial cost due to the need to purchase a production certificate, which is not always feasible for students or even, small enterprises. To address this, I primarily relied on the free trial certificates issued by the certificate provider. This in turn, allowed me to complete major parts of the system functionalities in the sandbox environment before switching over to the production environment. As a result, this approach reduces the need to use a production certificate from the start, and ensured that it was only used at the deployment stage, thereby reducing any unnecessary incurred expenses.

Moreover, this project also revealed that the software development process is change-oriented in nature with compliance to regulations. This is because of the continuous updates that LHDN has implemented in order to ensure its API guidelines, coding practice, and compliance are up-to-date. During the implementation phase, there were a lot of changes observed, which required the system design and part of the codebase to be changed. One of them includes adding mandatory fields such as email and telephone number to validate taxpayer. Hence, this further solidifies the need to adopt the Iterative and Incremental Development and Agile approaches as approaches that are rigid would not be able to accommodate the frequent regulatory changes.

Furthermore, the Software Development Kit documentation provided by LHDN was found to be lacking in details throughout the development period. While there are high-level explanations of the API endpoints and data format that were made available, most of the intricate, complex technical details were unclear or not even documented at all. To resolve this, I relied heavily on external sources such as GitHub and E-Invoice forums and trial-and-error testing in order to come up with the correct implementation.

Last but not least, the lack of available and responsive customer support made these problems even greater. Although LDHN provides a helpdesk to assist developers, most technical issues are eventually redirected to email communication with their outsourced vendors. This email-based process sometimes took days to get a response and constantly involved repeated back-and-forth clarification. Thus, the lack of real-time support has slowed down troubleshooting and prolonged the process of solving major integration issues. To mitigate this issue, I had made it a necessity to track and document every error, maintain formal testing logs, and verify those errors with community forums and available developer documentations. Therefore, this approach reduces the over-reliance on the delayed email responses, thereby ensuring that the project is able to progress while waiting for a response.

8.3 Recommendations

Although the project has fulfilled its primary objective of integrating with the LHDN MyInvois API and being capable of successfully generating e-invoices, there are still multiple limitations to this project that can be enhanced further in the future.

Firstly, although the system is capable of generating standard e-invoices by forwarding the required documents to the LHDN platform via APIs, it currently does not support generating credit notes or debit notes. These features are needed in situations when transactions are likely to need adjustment on the basis of returns, concessions or even corrections. The future development should therefore, enlarge the current implementation with credit and debit note processes to improve the flexibility of the system.

Secondly, the current system is lacking in terms of having advanced reporting or data analysis features. The current implementation primarily involves generating e-invoices, storing the details in the database and the data into a ledger. The ledger is then used as a reporting mechanism for viewing all

the e-invoices issued within the system. However, since the ledger operates on a traditional database, it still has the risks of having its records to be altered or tampered with, if the system happens to be compromised. Hence, potential future enhancements could include integrating blockchain-based approaches to provide greater transparency on records, thereby enhancing auditability and compliance of e-invoicing records.

Lastly, while the project has established the groundwork for ERP integration in the form of developing modular architecture and a structured database schema, there is no actual direct connection to enterprise resource planning (ERP) applications. Therefore, future suggestion could include having a logical extension to integrate the e-Invoicing system with existing ERP software. This would allow invoices and accounting entries to be reconciled between systems easily, thereby reducing redundancy and making invoice process much more uniform with other business processes.

REFERENCES

Autocount (2024) *LHDN e-Invoice - Auto Count Sdn Bhd*. Available at: https://www.autocountsoft.com/autocount-einvoice-solution-malaysia.html?utm_source=ggl&utm_medium=pmax_general_einvoice&utm_campaign=06_2024&gad_source=1&gbraid=0AAAADP7Zj-LAfRRu7RffgpZzOZLMGFvk&gclid=Cj0KCQjwt8zABhDKARIsAHXuD7bzT85Hm90wMI2J9bB52tBZJE6lM55UYG_KH3MwhwaXtfn0Foq7mmkaArTqEALw_wcB (Accessed: 20 March 2025).

Agile Dynamics Solutions (2024) *The History of Electronic Invoice*. Available at: https://adynamics.com.my/blog/the-history-of-electronic-invoice/ (Accessed: 24 March 2025).

Bukku (2024) *Malaysia E-Invoice Software Compliant with LHDN MyInvois & Peppol Network*. Available at: https://bukku.my/malaysia-e-invoice-compliance-lhdn-myinvois-peppol-network (Accessed: 20 March 2025).

Chan, D. and Alias, A. (2023) 'IRB's 'e-invoicing' initiative will save time, reduce cost, say experts', *New Straits Times*, 28 March. Available at: https://www.nst.com.my/news/nation/2023/03/893797/irbs-e-invoicing-initiative-will-save-time-reduce-cost-say-experts (Accessed: 10 March 2025).

C Fagarasan *et al.* (2021) *'Agile, waterfall and iterative approach in information technology projects',* IOP Conference Series Materials Science and Engineering, 1169(1), pp. 012025–012025. Available at: https://doi.org/10.1088/1757-899x/1169/1/012025. (Accessed 24 March 2025).

Cleartax (2024) *Role of CTC and Hybrid CTC in Malaysia e-Invoicing*. Available at: https://www.cleartax.com/my/en/centralized-continuous-transactions-hybrid-centralized-continuous-transactions-malaysia-e-invoicing (Accessed: 24 March 2025).

Cleartax (2024) *E-Invoice Malaysia Penalties: Consequences of Not Generating E-invoice.* Available at: https://www.cleartax.com/my/en/einvoice-malaysia-penalties (Accessed: 15 April 2025).

cPanel (2023) *Hosting Platform of Choice*. Available at: https://cpanel.net/#:~:text=cPanel%20provides%20the%20most%20reliable,focus%20on%20growing%20their%20businesses. (Accessed: 10 March 2025).

Darash, A. (2022) *What is an Electronic Invoice?*. Available at: https://www.regpacks.com/blog/what-is-e-invoicing/ (Accessed: 24 March 2025).

Edicom (2024) *Electronic Invoicing Models: CTC, Clearance, Real-Time, Centralized, Interoperable, and More*. Available at: https://edicomgroup.com/blog/electronic-invoice-models-ctc-clearance-interoperability (Accessed: 24 March 2025).

Financio (2025) *LHDN E-Invoice Software - Financio Malaysia*. Available at: https://financio.co/my/lhdn-e-invoice (Accessed: 12 March 2025).

GeeksforGeeks (2024) *Iterative Incremental Model in Designing System*. Available at: https://www.geeksforgeeks.org/iterative-incremental-model-in-designing-system/ (Accessed: 15 April 2025).

Golden Owl (2025) *Discover the Difference Between HTML, CSS, and Javascript.* Available at: https://goldenowl.asia/blog/difference-between-html-css-and-javascript (Accessed: 20 April 2025).

IBSS Ltd. (2025) *What Is The History Of E-Invoicing?*. Available at: https://www.ibsoftwaresolutions.com/blog/what-is-the-history-of-e-invoicing (Accessed: 24 March 2025).

InvoiceMate (2021) *10 Challenges of Manual Invoice Processing –*

InvoiceMate. Available at: https://www.invoicemate.net/10-challanges-of-manual-invoice-processing/ (Accessed: 19 March 2025).

JomeInvoice (2025) *Malaysia Top e-invoicing Software Provider for All System*. Available at: https://jomeinvoice.my/ (Accessed: 20 March 2025).

Lembaga Hasil Dalam Negeri (LHDN) (2025) *Integration Practices*, *Hasil.gov.my*. Available at: https://sdk.myinvois.hasil.gov.my/integration-practices/ (Accessed: 10 March 2025).

MyGOV - The Government of Malaysia's Official Portal (2025) *Twelfth Malaysia Plan (12MP).* Available at: https://www.malaysia.gov.my/portal/content/31186 (Accessed: 16 March 2025).

Nonyelum, Ogwueleka Francisca (2020) *Iterative and Incremental Development Analysis Study of Vocational Career Information Systems*, Ssrn. Available at: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3715318 (Accessed: 27 April 2025).

Outsized (2024) *Financio.* Available at: https://outsized.com/perks/financio/#:~:text=Financio.%20Designed%20for%20small%20businesses%20and%20non%2Daccountants%2C,driven%20document%20scanning%2C%20making%20accounting%20seamless%20and (Accessed: 20 April 2025).

Oyebola Okunogbe and Pouliquen, V. (2022) *'Technology, Taxation, and Corruption: Evidence from the Introduction of Electronic Tax Filing'*, American Economic Journal Economic Policy, 14(1), pp. 341–372. Available at: https://doi.org/10.1257/pol.20200123. (Accessed: 27 April 2025).

Pagero (2025) *The rise of e-invoicing CTC models across the world.* Available

at: https://www.pagero.com/blog/e-invoicing-ctc-models#what-is-ctc (Accessed: 24 March 2025).

RTC Suite (2024) *Real-Time Invoice Reporting Transforming Businesses.* Available at: https://rtcsuite.com/invoice-reporting/#:~:text=Real%2DTime%20Invoice%20Reporting%20(RTIR,of%20transaction%20or%20shortly%20thereafter (Accessed 24 March 2025).

Siamand Hesami, Jenkins, H. and Jenkins, G.P. (2024) '*Digital Transformation of Tax Administration and Compliance: A Systematic Literature Review on E-Invoicing and Prefilled Returns',* Digital Government Research and Practice, 5(3), pp. 1–20. Available at: https://doi.org/10.1145/3643687 (Accessed: 27 March 2025).

Surana, E. (2024) *E-Invoice Malaysia Penalties: Consequences of Not Generating E-invoice*. Available at: https://www.cleartax.com/my/en/einvoice-malaysia-penalties (Accessed: 28 April 2025).

SME Corp (2024) *Development of SMEs always a high priority: SME Corp chairman.* Available at: https://smecorp.gov.my/index.php/en/resources/2015-12-21-10-55-22/news/4962-development-of-smes-always-a-high-priority-sme-corp-chairman (Accessed: 21 March 2025).

SQL Account HQ (2024). *How to Register for E-Invoice on the LHDN Mytax MyInvois Portal*. YouTube. Available at: https://www.youtube.com/watch?v=JtZqHKzVj24 (Accessed 15 September 2025).

Tantawy, M. (2024) *What Is A General Ledger: Importance And How It Works*. Available at: https://www.enerpize.com/hub/what-is-general-ledger (Accessed: 26 April 2025).

Thomas, N., (2020). How To Use The System Usability Scale (SUS) To Evaluate The Usability Of Your Website. Available at:

https://usabilitygeek.com/how-to-use-the-system-usability-scale-sus-to-evaluate-the-usability-of-your-website/ (Accessed: 15 September 2025).

Turkington, L. (2024) *Supplier E-Invoicing Vs Traditional Invoicing: A Comparison*. Available at: https://www.b2be.com/blog/supplier-e-invoicing-vs-traditional-invoicing/ (Accessed: 19 March 2025).

Uyar, A. *et al.* (2021) *'Can e-government initiatives alleviate tax evasion? The moderation effect of ICT'*, Technological Forecasting and Social Change, 166, pp. 120597–120597. Available at: https://doi.org/10.1016/j.techfore.2021.120597 (Accessed: 28 March 2025).

Versapay (2023) *10 Reasons Why Paper-based Invoicing is Bad for Business*. Available at: https://www.versapay.com/resources/paper-based-invoicing-bad-for-business (Accessed: 18 March 2025).

Wejke, C. (2023) *Invoice fraud and how to avoid it*. Available at: https://www.pagero.com/blog/avoid-invoice-fraud (Accessed: 19 March 2025)

APPENDICES

Appendix A: System Usability Scale (SUS) Questionnaire Result

Participant number: 1

System Usability Scale (SUS)

This is a standard questionnaire that measures the overall usability of a system. Please select the answer that best expresses how you feel about each statement after using the e-invoicing system.

		Strongly Disagree	Somewhat Disagree	Neutral	Somewhat Agree	Strongly Agree
1.	I think I would use this system frequently to submit invoices for e-invoice generation.	☐	☐	☐	☑	☐
2.	I found the process of submitting invoices to generate e-invoices unnecessarily complex.	☑	☐	☐	☐	☐
3.	I thought the system was easy to use.	☐	☐	☐	☐	☑
4.	I think I would need the support of a technical person to onboard with the e-invoicing system effectively.	☐	☐	☑	☐	☐
5.	I found the various features were well integrated into the system.	☐	☐	☐	☐	☑
6.	I thought there was too much inconsistency in how the system handled invoice tasks.	☐	☑	☐	☐	☐
7.	I would imagine that most people would learn to use this e-invoicing system very quickly.	☐	☐	☐	☐	☑
8.	I found the system to be cumbersome when performing invoice-related tasks.	☐	☑	☐	☐	☐
9.	I felt very confident using the system to submit invoices and generate e-invoices.	☐	☐	☐	☐	☑
10.	I needed to equip myself with sufficient knowledge to use the e-invoice system effectively.	☑	☐	☐	☐	☐

Figure 8.1 System Usability Scale for User 1

Participant number: 2

System Usability Scale (SUS)

This is a standard questionnaire that measures the overall usability of a system. Please select the answer that best expresses how you feel about each statement after using the e-invoicing system.

		Strongly Disagree	Somewhat Disagree	Neutral	Somewhat Agree	Strongly Agree
1.	I think I would use this system frequently to submit invoices for e-invoice generation.	☐	☐	☐	☐	☑
2.	I found the process of submitting invoices to generate e-invoices unnecessarily complex.	☑	☐	☐	☐	☐
3.	I thought the system was easy to use.	☐	☐	☐	☐	☑
4.	I think I would need the support of a technical person to onboard with the e-invoicing system effectively.	☐	☑	☐	☐	☐
5.	I found the various features were well integrated into the system.	☐	☐	☐	☐	☑
6.	I thought there was too much inconsistency in how the system handled invoice tasks.	☐	☑	☐	☐	☐
7.	I would imagine that most people would learn to use this e-invoicing system very quickly.	☐	☐	☐	☐	☑
8.	I found the system to be cumbersome when performing invoice-related tasks.	☑	☐	☐	☐	☐
9.	I felt very confident using the system to submit invoices and generate e-invoices.	☐	☐	☐	☑	☐
10.	I needed to equip myself with sufficient knowledge to use the e-invoice system effectively.	☐	☐	☑	☐	☐

Figure 8.2 System Usability Scale for User 2

Participant number: 3

System Usability Scale (SUS)

This is a standard questionnaire that measures the overall usability of a system. Please select the answer that best expresses how you feel about each statement after using the e-invoicing system.

	Strongly Disagree	Somewhat Disagree	Neutral	Somewhat Agree	Strongly Agree
1. I think I would use this system frequently to submit invoices for e-invoice generation.	☐	☐	☐	☐	☑
2. I found the process of submitting invoices to generate e-invoices unnecessarily complex.	☑	☐	☐	☐	☐
3. I thought the system was easy to use.	☐	☐	☐	☐	☑
4. I think I would need the support of a technical person to onboard with the e-invoicing system effectively.	☐	☐	☐	☑	☐
5. I found the various features were well integrated into the system.	☐	☐	☐	☐	☑
6. I thought there was too much inconsistency in how the system handled invoice tasks.	☐	☑	☐	☐	☐
7. I would imagine that most people would learn to use this e-invoicing system very quickly.	☐	☐	☐	☐	☑
8. I found the system to be cumbersome when performing invoice-related tasks.	☑	☐	☐	☐	☐
9. I felt very confident using the system to submit invoices and generate e-invoices.	☐	☐	☐	☑	☐
10. I needed to equip myself with sufficient knowledge to use the e-invoice system effectively.	☐	☑	☐	☐	☐

Figure 8.3 System Usability Scale for User 3

Participant number: 4

System Usability Scale (SUS)

This is a standard questionnaire that measures the overall usability of a system. Please select the answer that best expresses how you feel about each statement after using the e-invoicing system.

	Strongly Disagree	Somewhat Disagree	Neutral	Somewhat Agree	Strongly Agree
1. I think I would use this system frequently to submit invoices for e-invoice generation.	☐	☐	☐	☑	☐
2. I found the process of submitting invoices to generate e-invoices unnecessarily complex.	☑	☐	☐	☐	☐
3. I thought the system was easy to use.	☐	☐	☐	☐	☑
4. I think I would need the support of a technical person to onboard with the e-invoicing system effectively.	☐	☐	☐	☑	☐
5. I found the various features were well integrated into the system.	☐	☐	☑	☐	☐
6. I thought there was too much inconsistency in how the system handled invoice tasks.	☐	☑	☐	☐	☐
7. I would imagine that most people would learn to use this e-invoicing system very quickly.	☐	☐	☐	☐	☑
8. I found the system to be cumbersome when performing invoice-related tasks.	☑	☐	☐	☐	☐
9. I felt very confident using the system to submit invoices and generate e-invoices.	☐	☐	☑	☐	☐
10. I needed to equip myself with sufficient knowledge to use the e-invoice system effectively.	☑	☐	☐	☐	☐

Figure 8.4 System Usability Scale for User 4

Participant number: 5

System Usability Scale (SUS)

This is a standard questionnaire that measures the overall usability of a system. Please select the answer that best expresses how you feel about each statement after using the e-invoicing system.

	Strongly Disagree	Somewhat Disagree	Neutral	Somewhat Agree	Strongly Agree
1. I think I would use this system frequently to submit invoices for e-invoice generation.	☐	☐	☐	☑	☐
2. I found the process of submitting invoices to generate e-invoices unnecessarily complex.	☑	☐	☐	☐	☐
3. I thought the system was easy to use.	☐	☐	☐	☐	☑
4. I think I would need the support of a technical person to onboard with the e-invoicing system effectively.	☐	☐	☑	☐	☐
5. I found the various features were well integrated into the system.	☐	☐	☐	☐	☑
6. I thought there was too much inconsistency in how the system handled invoice tasks.	☐	☑	☐	☐	☐
7. I would imagine that most people would learn to use this e-invoicing system very quickly.	☐	☐	☐	☐	☑
8. I found the system to be cumbersome when performing invoice-related tasks.	☐	☑	☐	☐	☐
9. I felt very confident using the system to submit invoices and generate e-invoices.	☐	☐	☐	☐	☑
10. I needed to equip myself with sufficient knowledge to use the e-invoice system effectively.	☑	☐	☐	☐	☐

Figure 8.5 System Usability Scale for User 5