

**PROFMATE: AI-POWERED LECTURER ASSISTANT**

BY

LOH ZI HIN

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

**BACHELOR OF COMPUTER SCIENCE (HONOURS)**

Faculty of Information and Communication Technology

(Kampar Campus)

JUNE 2025

## **COPYRIGHT STATEMENT**

© 2025 Loh Zi Hin. All rights reserved.

This Final Year Project report is submitted in partial fulfillment of the requirements for the degree of Bachelor of Computer Science (Honours) at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project report represents the work of the author, except where due acknowledgment 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 express thanks and appreciation to my supervisor, Dr. Tan Joi San and my moderator, Ms. Tseu Kwan Lee who have given me a golden opportunity to explore the application of artificial intelligence in academic administration. Besides, they have given me a lot of guidance in order to complete this project. When I was facing problems in this project, the advice from them always assists me in overcoming the problems. Again, a million thanks to my supervisor and moderator.

To a very special person in my life, Yeoh Yun Yuan, for her patience, unconditional support, and love, and for standing by my side during hard times. Finally, I must say thanks to my parents and my family for their love, support, and continuous encouragement throughout the course.

# ABSTRACT

ProfMate is a cutting-edge AI-powered lecturer assistant aimed at revolutionizing the way academic tasks are managed and executed. The project aims to address the inefficient and time-consuming aspects of manual grading and administrative duties through the integration of machine learning and automation technologies. ProfMate comprises several core modules: (1) an AI-driven image recognition system designed to assess and validate the accuracy of manually marked papers, identifying missed questions and verifying correct score computation; (2) an automated grade entry tool that streamlines the process of recording and updating student grades; (3) a scheduling interface that facilitates seamless appointment booking between students and lecturers via a shared calendar; (4) a comprehensive login system that distinguishes between student and lecturer access, enabling efficient management of grades, meetings, and review tools; and (5) a results portal that aggregates and presents outcomes from paper reviews and assessments. By leveraging these technologies, ProfMate aims to significantly enhance the efficiency, accuracy, and overall effectiveness of academic administration, thus allowing educators to devote more time to teaching and student engagement. The proposed solution promises to transform educational management, offering a more organised and responsive approach to handling academic tasks.

Area of Study: Computer Vision, Artificial Intelligence (AI)

Keywords: Tick Mark Detection, CNN, Faster R-CNN, YOLOv5, Computer Vision in Education, Automated Grading Tools, Custom Object Detection Model

# TABLE OF CONTENTS

<b>TITLE PAGE</b>	<b>i</b>
<b>COPYRIGHT STATEMENT</b>	<b>ii</b>
<b>ACKNOWLEDGEMENTS</b>	<b>iii</b>
<b>ABSTRACT</b>	<b>iv</b>
<b>TABLE OF CONTENTS</b>	<b>v</b>
<b>LIST OF FIGURES</b>	<b>viii</b>
<b>LIST OF TABLES</b>	<b>x</b>
<b>LIST OF ABBREVIATIONS</b>	<b>xi</b>
<b>CHAPTER 1 INTRODUCTION</b>	<b>1</b>
1.1 Problem Statement and Motivation	2
1.2 Objectives	3
1.3 Project Scope and Direction	4
1.4 Contributions	5
1.5 Report Organization	6
<b>CHAPTER 2 LITERATURE REVIEW</b>	<b>7</b>
2.1 Previous Studies	7
2.1.1 Object Detection in image recognition	7
2.1.2 Faster R-CNN	8
2.1.3 You Only Look Once (YOLO)	9
2.2 Review on Existing Systems	10
2.3 Strengths and Weaknesses of Existing Systems	14
2.4 Comparison between Existing Systems	16
<b>CHAPTER 3 SYSTEM METHODOLOGY/APPROACH</b>	<b>18</b>
3.1 Methodology	18
3.2 System Requirement	19
3.2.1 Hardware	19
3.2.2 Software	19
3.3 System Design Diagram	22

3.3.1 Use Case Diagram	22
3.3.2 Use Case Description	22
<b>CHAPTER 4 SYSTEM IMPLEMENTATION</b>	<b>29</b>
4.1 Program Development	29
4.1.1 Registration / Authentication Module	29
4.1.2 Home Page	30
4.1.3 Tick Scanner	31
4.1.4 Results Portal	31
4.1.5 Meeting Scheduler	32
4.1.6 Profile Page	33
4.1.7 Back-end Implementation	33
4.1.8 Real-Time Tick Detection API	34
4.2 Implementation Issues and Challenges	36
4.3 Gantt Chart	38
4.4 Summary	39
<b>CHAPTER 5 SYSTEM EVALUATION AND DISCUSSION</b>	<b>40</b>
5.1 Tick Scanner	40
5.2 Results Portal	42
5.3 Meeting Scheduler	43
5.4 Login / Register	44
5.5 Profile Page	45
<b>CHAPTER 6 COMPARISON OF DEEP LEARNING MODELS FOR TICK DETECTION</b>	<b>47</b>
6.1 Model Architecture & Training	47
6.1.1 Faster R-CNN	47
6.1.2 YOLOv5n	49
6.2 Dataset Preparation	52
6.3 Model Evaluation	54
6.3.1 Faster R-CNN	54
6.3.2 YOLOv5n	56

6.4	Comparison of Metrics and Visualisation	60
6.5	Summary	62
<b>CHAPTER 7 CONCLUSION AND RECOMMENDATION</b>		<b>63</b>
<b>REFERENCES</b>		<b>64</b>
<b>APPENDIX</b>		<b>66</b>
<b>POSTER</b>		<b>66</b>

# LIST OF FIGURES

<b>Figure Number</b>	<b>Title</b>	<b>Page</b>
Figure 2.1.1	General flow of object detection	7
Figure 2.2.1.1	Lesson Plan topic selection	10
Figure 2.2.1.2	Saving generated lesson plan	11
Figure 2.2.1.3	Text Leveler	11
Figure 2.2.2.1	Slideshow Generator	12
Figure 2.2.2.2	Assessment Task Ideas Generator	13
Figure 2.2.3.1	Form to create an assessment	14
Figure 2.2.3.2	Review page on marked assessment	14
Figure 3.1.1	Flow of RAD	18
Figure 3.3.1.1	Use case diagram for ProfMate	22
Figure 4.1.1.1	auth.js	30
Figure 4.1.2.1	Home.jsx	30
Figure 4.1.3.1	TickScanner.jsx	31
Figure 4.1.4.1	StudentResults.jsx	32
Figure 4.1.5.1	MeetingScheduler.jsx	32
Figure 4.1.5.1	Profile.jsx	33
Figure 4.1.7.1	server.js	34
Figure 4.1.8.1	yolo_api.py	35
Figure 4.3.1	Gantt Chart of Project Timeline	38
Figure 5.1.1	Tick Detection Scanner Page (before scanning starts)	40
Figure 5.1.2	Tick Detection Scanner (Manual Mode)	41
Figure 5.1.3	Tick Detection Scanner (Automatic Mode)	41
Figure 5.2.1	Results Portal (Lecturer View)	42
Figure 5.2.1	Results Portal (Student View)	43
Figure 5.3.1	Meeting Scheduler (Lecturer View)	43
Figure 5.3.1	Meeting Scheduler (Student View)	44
Figure 5.4.1	Register Page	45
Figure 5.4.2	Login Page	45



Figure 5.5.1	Profile Page	46
Figure 6.1.1.1	Faster R-CNN model with ResNet-50 and FPN [17]	48
Figure 6.1.1.2	Model Architecture Definition	48
Figure 6.1.1.3	Optimizer and Learning Rate Scheduler Setup	49
Figure 6.1.2.1	YOLOv5 Network Structure Diagram	50
Figure 6.1.2.2	YOLOv5n Training Pipeline Setup in Google Colab	50
Figure 6.2.1	Annotating images to create respective Pascal VOC XML files	52
Figure 6.2.2	Dataset Splitter	53
Figure 6.3.1.1	Loss Curve for 30 Epoch Run	54
Figure 6.3.1.2	Loss Curve for Free-Run (19 epochs)	55
Figure 6.3.1.3	Confusion Matrix of Faster R-CNN	55
Figure 6.3.1.4	Precision-Recall Curve (Faster R-CNN)	56
Figure 6.3.2.1	Loss Graphs for 30 Epoch Run	57
Figure 6.3.2.2	Loss Graphs for Free Run (76 Epochs)	58
Figure 6.3.2.3	Confusion Matrix of YOLOv5n	59
Figure 6.3.2.4	Precision-Recall Curve (YOLOv5n)	59
Figure 6.4.1	Visualised Detections of Faster R-CNN (left) and YOLOv5n (right)	
Figure 6.4.2	Visualised Detections of Faster R-CNN (left) and YOLOv5n (right)	
Figure 6.4.3	Visualised Detections of Faster R-CNN (left) and YOLOv5n (right)	

## LIST OF TABLES

<b>Table Number</b>	<b>Title</b>	<b>Page</b>
Table 2.5.1	Comparison between Existing Systems	16
Table 3.2.1.1	Specifications of desktop PC	19
Table 3.2.2.1	Software components and specifications	20
Table 3.3.2.1	Login Use Case Description	22
Table 3.3.2.2	Register Use Case Description	23
Table 3.3.2.3	View Results Use Case Diagram	24
Table 3.3.2.4	Upload Excel File Use Case Diagram	24
Table 3.3.2.5	Automatically Verify Marks (Tick Detection) Use Case Diagram	25
Table 3.3.2.6	Automatically Enter Grades into System Use Case Diagram	26
Table 3.3.2.7	Manage Meeting Slots Use Case Diagram	26
Table 3.3.2.8	View Scheduled Meetings Use Case Diagram	27
Table 3.3.2.9	View Grades Use Case Diagram	27
Table 3.3.2.10	Book Meetings Use Case Diagram	28
Table 6.1.3.1	Comparison of Architecture and Training between Custom CNN, Faster R-CNN and YOLOv5n	51
Table 6.4.1	Comparison of Performance Metrics for Faster R-CNN and YOLOv5n	60

## **LIST OF ABBREVIATIONS**

AI	Artificial Intelligence
IDE	Integrated Development Environment
SVM	Support Vector Machine
K-NN	K-Nearest Neighbours
VOC	Visual Object Classes
COCO	Common Objects in Context
R-CNN	Region-based Convolutional Neural Network
YOLO	You Only Look Once
RPN	Region Proposal Network
ResNet	Residual Network
UI	User Interface

# Chapter 1

## Introduction

In the field of education, educators including teachers, lecturers, and professors face growing challenges in managing their time and responsibilities. With the heavy workload filled with tasks like lesson planning, grading and scheduling meetings with students which takes up a lot of time, educators are left with less time for other, more crucial tasks, and it may disrupt their focus on engaging with students and improving the learning experience. Balancing between the core responsibility of teaching and these side tasks has become a common struggle for educators, as the substantial amount of workload causes fatigue and a lack of time for meaningful interactions with students.

Traditionally, teaching assistants have played a vital role in helping educators, especially in tertiary education like lecturers and professors manage their workload, taking on side responsibilities such as grading papers, data entry of grades for students, scheduling student meetings and more. However, with recent advancements of AI technology, virtual assistants and educational apps are slowly beginning to replace the tasks originally handled by teaching assistants. Furthermore, these systems are often more reliable than traditional teaching assistants, as human teaching assistants have limitations that can be solved by technology. Unlike human assistants, these AI-powered systems are available 24/7, can provide immediate feedback, and most importantly, able to perform tasks with a high level of consistency. Unaffected by factors such as fatigue, availability and performance, the shift towards the usage of AI to replace teaching assistants shows a growing trust in technology and an increasing demand of AI-powered teaching assistants.

To address these growing demands, ProfMate is proposed as an AI-powered solution, specifically designed to assist lecturers and professors with tasks outside their core responsibility of teaching. Instead of focusing on activities like lesson planning or assessment creation, ProfMate will be handling time-consuming side tasks that often overwhelm lecturers. The primary modules of this system will include an AI-powered image recognition and object detection tool to scan marked papers and check if the papers have been marked correctly. This tool will essentially scan for markings like red ticks to ensure no questions are missed. Additionally, ProfMate will automate the entry of student grades, in which the marks collected from the checked papers will be automatically fill in the grade for the respective students. Finally, ProfMate will provide a platform where students can easily schedule meetings with

lecturers and professors by accessing a shared calendar, replacing the tedious process of messaging back-and-forth and reducing wait times. From handling these potentially redundant tasks, ProfMate will allow lecturers and professors to focus on the core responsibility of educators, which is to teach and effectively deliver knowledge to students.

### **1.1 Problem Statement and Motivation**

In a tertiary education environment, lecturers and professors are faced with high pressure and a huge amount of workload as a wide array of tasks outside of their core responsibility of teaching are expected to be handled. As a result, there are some challenges and problems faced by lecturers and professors. These problems include time-consuming manual paper checking, inefficient grade entry process, and complicated and delayed meeting scheduling.

#### **i. Time-Consuming Manual Checking of Marked Papers**

After student papers have been marked, lecturers and professors are often required to manually review the papers to ensure that all the questions have been marked and graded, and the total marks are correctly calculated. Although this process is crucial in terms of accuracy, it is extremely time-consuming, especially for courses with a large number of students. This tedious process involves carefully checking on each page of the paper to ensure that no questions were overlooked, and no errors were made in the previous marking process. This can lead to long delays in returning papers to students and takes away the time that can be better utilised on performing core tasks of an educator such as lesson planning or mentoring.

#### **ii. Inefficient Grade Entry Process**

Once the student papers are marked, lecturers and professors must manually key in grades into the university database, which is a long and repetitive task. Not only that, this process is also highly prone to mistakes, especially in large classes, where human error can result in recording incorrect grades. These errors cause confusion and frustration for students, and in turn produce even more work for lecturers when corrections are required. The large amount of time spent on manual student grades entry distracts lecturers and professors from putting their focus on the main task of teaching, which may even slow down the progress of the whole class.

### **iii. Complicated and Delayed Meeting Scheduling**

When students and lecturers schedule meetings for consultation or other agendas, the process is often troublesome and inefficient. It typically starts with the student trying to reach out by sending an e-mail or message to the lecturer or professor to ask about their availability, which can lead to delays due to the late response from the lecturers and professors due to their busy schedule. The repetitive nature of this process can be extremely time-consuming for both sides, and often takes longer than necessary. Moreover, lecturers and professors may unintentionally overlook meeting requests due to the sheer volume of messages received, causing further delays. The lack of consistency and efficiency of this process can hinder the learning process of students, which may negatively affect their academic performance.

## **1.2 Objectives**

The main objective of this project is to develop a web application to reduce the workload of lecturers and professors by automating and simplifying side tasks that are out of their main responsibilities of teaching, which will free up more time for lecturers and professors to focus on teaching tasks, thus enhancing the teaching experience.

### **i. To automate the process of reviewing marked assessment papers**

ProfMate will automate the process of reviewing marked assessment papers by implementing a marked paper review tool that uses object detection in image recognition and machine learning to scan marked papers and verify that all the questions on the paper are marked correctly and the total marks tally up accurately. The automation of this tedious task aims to reduce the workload of lecturers and professors, allowing them to allocate more time on teaching tasks instead of side tasks which are potentially redundant in nature. This accuracy and consistency of this process will also be enhanced as a result of the reliability and efficiency of AI technology.

### **ii. To automate the grade management of students**

Another objective of ProfMate is to automate the grade management of students by introducing an automated feature that enters the grades of students after papers have been reviewed. This feature is aimed to be integrated seamlessly with the marked paper review tool, in which this feature will capture student information such as name and student ID, and with this information the grade will be automatically recorded according to the score on the marked paper. By

automating this process, ProfMate will simplify the long process and reduce inconsistencies previously caused by human error, and this benefits both lecturers and students.

### **iii. To simplify scheduling meetings and communication**

ProfMate aims to provide a scheduling platform for student meetings which will serve as the main portal for students to book appointments with lecturers. Students will be able to check the availability of lecturers without having to send messages or e-mails, which often leads to delays due to the sheer number of messages received by lecturers. With this, ProfMate will help to reduce the number of messages lecturers have to reply by having a platform to handle the bookings, in which it also prevents delays and ensure a smoother academic experience for students.

## **1.3 Project Scope and Direction**

The scope of this project is to decrease the workload of lecturers and professors by providing an AI-powered lecturer assistant web application, ProfMate, to automate side tasks which includes reviewing marked papers and data entry of student grades. ProfMate will also include a platform where students and lecturers can schedule meetings for consultation or other purposes without the need for back-and-forth messaging. Features of ProfMate will include marked paper review tool, automated grade entry, meeting scheduler, login, and results portal.

### **i. Marked paper review tool**

This tool will allow lecturers and professors to scan or upload assessment papers to automatically check for red ticks and other markings. The system will verify that all questions are marked and the total marks tally up accordingly, which streamlines the originally manual process. This feature ensures accurately marked papers and students receive correct grades by automating the review process before releasing the results.

### **ii. Automated grade entry**

Connected to the marked paper review tool, this feature will scan papers and collect student details like name and student ID. After successfully reviewing papers, this feature will automatically enter grades into the database, replacing the manual data entry task. This tool introduces efficiency and decreases the risk of human error in recording grades for students.

### **iii. Meeting Scheduler**

The meeting scheduler will be provided to simplify the process of scheduling meetings between lecturers and students. Using this platform, students can check for available time slots of a particular lecturer through the posted schedules, which students can then directly book meetings on the platform. This module essentially reduces the delays caused by the traditional way of messaging lecturers to book appointments manually.

### **iv. Login Module**

Access to ProfMate is restricted to students and lecturers. Upon login, students can check grades and schedule meetings, while lecturers will be granted access to additional tools, including the marked paper review tool, automated grade entry, and the results portal. Different user groups will have access to different functionalities to ensure secure and suitable access.

### **v. Results Portal**

The results portal in ProfMate will allow students and lecturers to view the results of reviewed papers of various assessments, such as midterms and quizzes. This provides a convenient and centralised platform for students and lecturers to track academic progress, without having the delay that may be caused by the manual process of posting results on a separate platform.

## **1.4 Contributions**

In the present educational environment, lecturers and professors are increasingly burdened by additional tasks that distracts them from performing their core responsibilities of teaching and delivering knowledge to students. These tasks are repetitive in nature and prone to mistakes due to human error, which in the long term will affect the quality of education and increasing stress levels of educators.

To tackle these challenges, ProfMate will be developed to address the specific issues face by educators. This system targets to automate time-consuming tasks such as reviewing marked papers and data entry for student grades, which will increase efficiency and accuracy. Most importantly, ProfMate aims to reduce the burden on educators, allowing them to put more focus on meaningful student interactions and improve the teaching process. ProfMate will also provide a platform to simplify the scheduling of student meetings, making it easier for students



## CHAPTER 1

to receive academic support on time, without having to go through the long and tedious traditional method.

As for the contributions of ProfMate, it will extend beyond solving the aforementioned problems. ProfMate introduces an innovative use of AI in education, and this displays how AI technology can improve the efficiency and accuracy of the side tasks. This will set a new standard for educational tools and the use of AI in other areas of education, as ProfMate will provide a view for how technology can decrease the burdens of educators. The success could inspire more innovations and improvements on educational tools, which will benefit a large number of institutions and educators.

### 1.5 Report Organisation

In this report, the development of ProfMate is divided into seven chapters. Chapter 1 introduces the project by outlining the problem statement, objectives, scope, and overall direction. Chapter 2 presents a literature review of related systems and studies that serve as references and comparisons for this work. Chapter 3 explains the methodology of the system, including its design architecture, use case, and activity diagrams. Chapter 4 describes the detailed system design, covering block diagrams, component specifications, and how the different modules interact. Chapter 5 focuses on the system implementation, including the hardware and software setup, configuration, and the overall operation of ProfMate, along with challenges faced during development. Chapter 6 discusses the evaluation of the system, presenting the testing process, performance metrics, and results in relation to the stated objectives. Finally, Chapter 7 concludes the report by summarizing the outcomes of the project and providing recommendations for future enhancements.

## Chapter 2

### Literature Review

#### 2.1 Previous Studies

##### 2.1.1 Object Detection in Image Recognition

Object detection is a technique in image recognition that involves identifying and locating objects within an image. According to the author of [1], object detection is defined as detection of instances of objects from one or multiple classes in an image. Object detection aims to detect and identify all instances of objects of different classes like faces, fruits or plants that are present in an image. Object detection systems work by building a model for a class of objects from training examples provided. Generally, objects that are fixed and rigid may only need one example, but it is always necessary to have a larger set of training examples to capture the minor characteristics of the class of object that might change. The general flow of object detection is illustrated by the author of [2] as such:

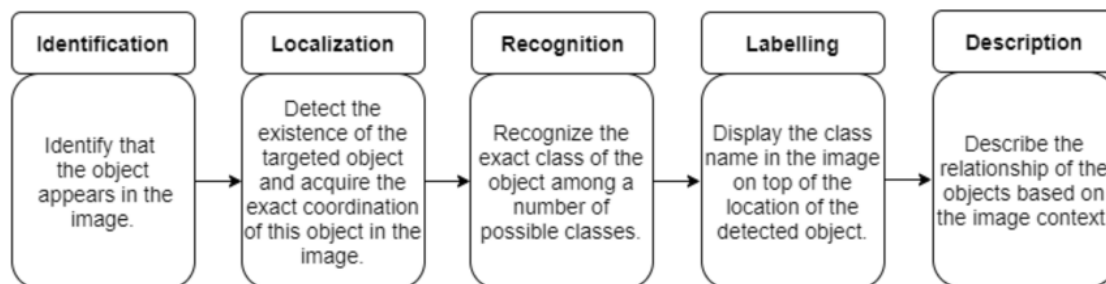


Figure 2.1.1: General flow of object detection [2]

The author of [1] also stated that the images of an object class are highly variable. Factors such as brightness, angle of capture, and noise can result in different variants of the same object. As for things like handwriting or symbols, people have different ways of writing and drawing, which is also a contributing factor to the variety. Authors of [3] found that most traditional object detection techniques consist of three parts, which begins with the inspection of an image at a variety of positions and scales to essentially create boxes and margins using different methods. The methods include region proposal, max-margin object detection and more. The study also pointed out that the process of image scanning may be extended due to the complex and computationally expensive calculation methods in the first phase. The following step is to run feature extraction of the scanned images by performing analysis on the

regions generated from the previous step to extract features or patterns within an image. This part of the process has utmost importance as it is vital for the performance of the algorithm. The last step is to leverage classification algorithms like Support Vector Machine (SVM), K-Nearest Neighbours (K-NN) to perform classification on the images to determine whether the object appears in the image. The author states that in traditional pipelines, performance depended on feature set, classifier, learning method, and training set. In modern deep learning approaches, these are often learned jointly within the network. This study also compared and evaluated the performance of object detection models using benchmark datasets such as Pascal VOC and Microsoft COCO. It was concluded that single-stage detectors have seen improvements in recent years and matches the accuracy of two-stage detectors.

Looking further into classification-based object detection frameworks, the study [4] states that in earlier approaches like R-CNN, these stages were trained separately. However, Faster R-CNN introduced end-to-end training that unifies these steps. Sharing the same finding as the author of [3], the author of [4] also concluded that one-step classification-based frameworks can reduce time expense, as it maps straightly from image pixels to bounding boxes. In the study [5], the authors proposed an approach of deep neural network training and stated the importance of deep representations in improving performance for object detection tasks. The winning solution for the 2015 COCO object detection challenge leveraged Faster R-CNN with deep residual networks (ResNet), highlighting the power of deep representations. The increased depth of the network effectively captured the details at various levels of an image, which is important in object detection where accurate localisation and identification of objects are required.

### 2.1.2 Faster R-CNN

Faster R-CNN (Faster Region-based Convolutional Neural Network) is a deep learning model used for object detection, which identifies and localizes objects within an image by drawing bounding boxes and assigning labels. It is a two-stage object detection framework which combines a Region Proposal Network (RPN) with a Fast R-CNN detector, which produces an efficient and accurate object detection model. From the authors of [6], a study of object detection based on Faster R-CNN was done by displaying the experimental results of the different Faster R-CNN models that are built by training with deep learning framework of Caffe. In the study, the results show that the effectiveness of Faster-RCNN models is contributed by the convolutional layers and the RPN module. Previous studies have also

utilised Faster R-CNN with ResNet-50 backbones for highly accurate object detection tasks. For instance, the authors of [7] used ResNet-50 depths on a Pascal VOC dataset to evaluate the model performance on human detection. The model demonstrated strong performance even under complex environments, which can be useful in detecting small visual cues like tick marks. This supports the effectiveness of using a ResNet-50 based Faster R-CNN model for object detection on small tick marks. In another study, [8] applied Faster R-CNN with an FPN backbone to detect and classify fruits under different lighting conditions and noise. Two datasets were built, one comparing to apples with oranges and another comparing a group of apples. The model performed well on both datasets, showing that this model can handle visually similar objects.

### 2.1.3 You Only Look Once (YOLO)

YOLO (You Only Look Once) is a deep learning model used for object detection, which identifies and localises objects in an image by dividing the image into grids and predicting bounding boxes and class probabilities in a single pass. YOLO is a one-stage object detection framework that processes images in real time with high speed and accuracy by combining region proposal and classification into one unified network. From the authors of [9], a study on object detection using YOLO was conducted by evaluating the performance of different YOLO versions trained on the Darknet framework. The results demonstrate that the speed and efficiency of YOLO models are largely attributed to their grid-based prediction approach and fully convolutional architecture. Previous studies have also employed YOLOv5 with lightweight backbones for detecting small objects in real-time scenarios. For instance, the authors of [10] applied YOLOv5 on the COCO dataset to assess its performance in multi-object detection under complex backgrounds. The model achieved strong accuracy while maintaining low latency, which is important for detecting small-scale features such as tick marks. This supports the suitability of YOLO-based models in cases where both speed and precision are required. In another study, [11] utilised YOLOv4 with a CSPDarknet53 backbone to detect defects in industrial components under varying noise and lighting conditions. The model performed robustly across different environments, indicating that YOLO can effectively generalise to visually challenging tasks.

## 2.2 Review on Existing Systems

### 2.2.1 Khanmigo by Khan Academy

Khanmigo, released by nonprofit educational organisation Khan Academy, is a personal tutor and teaching assistant that utilises AI technology. Khanmigo offers 3 different experiences for different groups of users, which are educators, learners and parents. Although learners and parents are required to sign up with a paid subscription, Khanmigo is exclusively free for educators [12].

Focusing on the Khanmigo for educators, there are numerous features that are designed to perform tasks for educators, to reduce the workload and in turn allows educators to focus more on students. One of the main features of Khanmigo is the lesson planner. The lesson planner aims to design customised plans that adequately align the curriculum and the students' needs. A user can choose with topic to create a lesson plan about, and from Khan Academy's large course offerings, Khanmigo will try to find the topic if available. Users can also provide content to the lesson planner if the topic is not found in the course offerings.

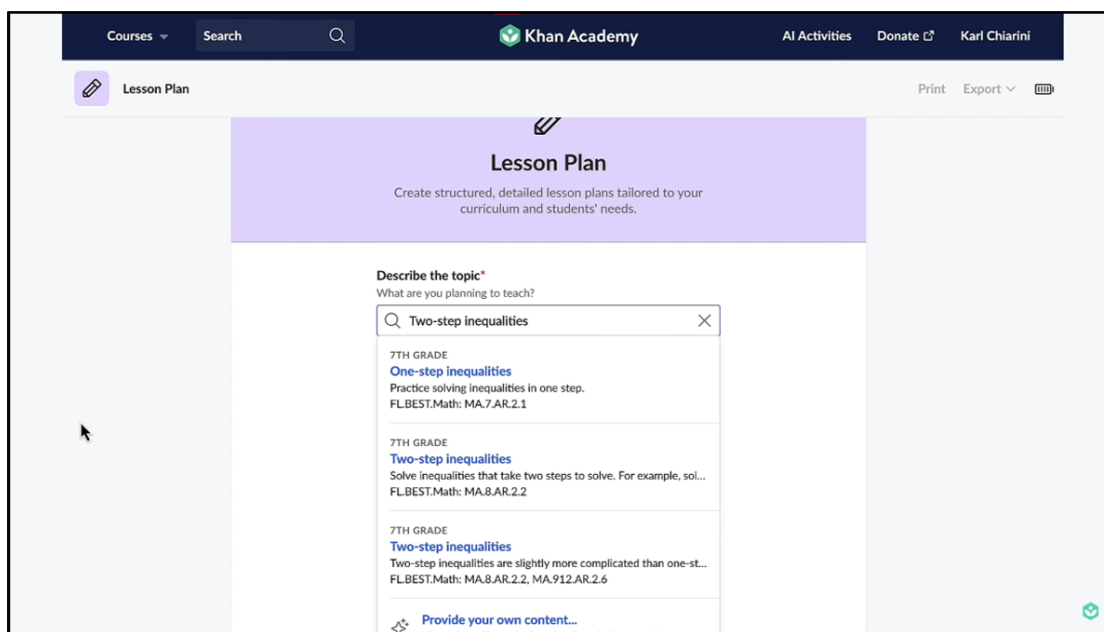


Figure 2.2.1.1: Lesson Plan topic selection [13]

Khanmigo will then generate a structured and detailed lesson plan that includes learning objectives, warm-up activities, practices and quiz questions. The generated lesson plan can be further customised and manually changed if it does not suit the user's needs. The lesson plan can then be printed out, exported in PDF document (.pdf) or Word file (.docx), or saved to the user's Google Drive.

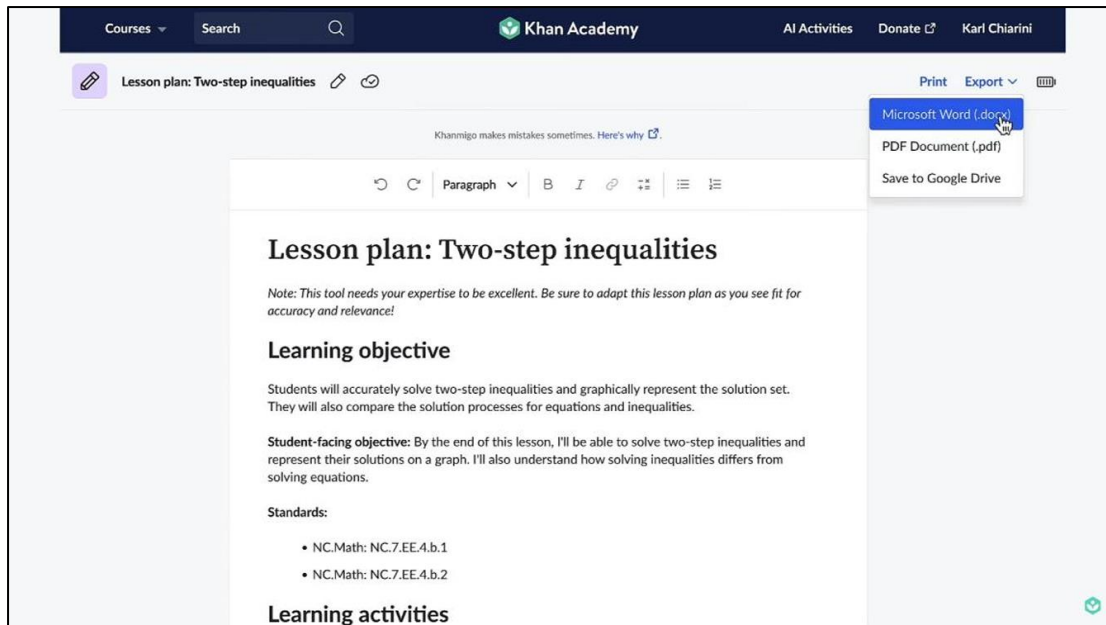


Figure 2.2.1.2: Saving generated lesson plan [13]

Another interesting feature that Khanmigo offer is the text leveler. The text leveler takes a body of text limited at the length of 4500 words and the desired level that the user would like to regenerate the text into. For instance, a user can enter a long paragraph into this feature, select the reading level, and the text will be rewritten in the chosen level of complexity. The goal of this feature is to adjust the level of complexity of a text to an extent that students can fully understand, as it is important for students to be able to read the text while understanding its meaning without having difficulties with the complex words used in the original piece of text [10].

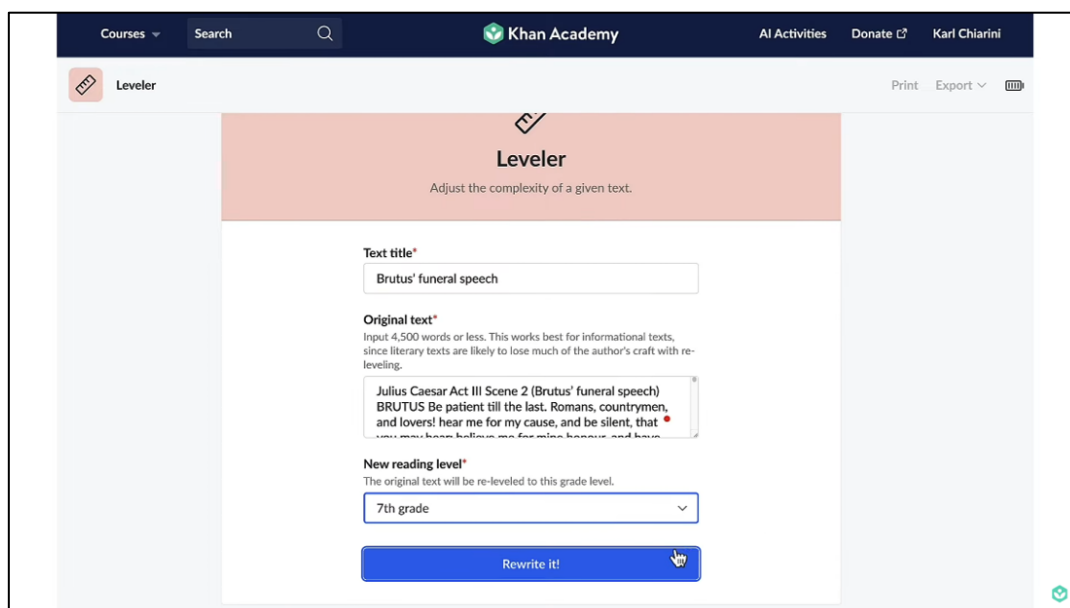


Figure 2.2.1.3: Text Leveler [13]

### 2.2.2 TeachMateAI

TeachMateAI is an education AI assistant, with over 200,000 users worldwide. Designed by educators and tech experts, it offers more than 116 AI-powered tools that are capable of handling tasks such as planning, creating resources, generating assessments, reports and more. Aimed to improve teaching and learning experiences and reducing educators' workload, a free trial version is available for users to try out the different AI tools available in TeachMateAI. A paid subscription is also offered if users wish to gain access to all the AI tools. TeachMateAI supports curricula from England, Scotland, Wales, Northern Ireland and core curricula from United States, Canada, Australia, New Zealand and Ireland. Users can select the preferred curriculum in the user settings to fine-tune the AI model to match the curriculum that the user will teach.

One of the top tools listed in TeachMateAI is the slideshow generator. This tool generates and downloads a PowerPoint presentation (.pptx) for any learning objective depending on user input. In this tool, a form is provided for users to enter options such as the preferred curriculum, year group, subject, learning objectives and the number of slides. After that, the tool utilises AI to draft the slides according to the input, including simple assessment questions and answers. The slideshow generator can help educators to prevent spending too much time on designing slides for every lesson, which takes up a lot of time for educators to come up with different designs every slideshow. Instead, educators will be able to put more focus on the content and syllabus, and more importantly the way that the knowledge will be presented to the students.

**Slideshow Generator**

This tool can be used to create teaching slide presentations. Simply enter the learning objective, year group of pupils, subject and number of slides required. The AI will then draft the slides including assessment questions and answers.

0 Tool Uses    0 hours Time Saved    Add Favourite    View Instructions

**Inputs**

**Curriculum**  
You are viewing curriculums from your country setting in "my account" (other).  
2014 National Curriculum for England

**Year group**  
The inputs and output from this tool are customised based on your year group. We change the background AI significantly based on your school setting.  
Year Group  
☐ Mixed Year Groups

**Subject**  
e.g. Science, English, etc...

**Learning Objective**  
E.g. I know the different parts of the digestive system

**Number of Slides (max 10 - recommended 5)**  
5

Figure 2.2.2.1: Slideshow Generator [14]

TeachMateAI also offers an assessment task ideas generator. This tool generates a wide range of ideas on how assessments can be carried out on students by providing ideas for assessment tasks, depending on user input. Similar to the slideshow generator, users are required to provide information such as curriculum, year group, subject and learning objectives. By default, this tool generates a list of 10 assessments tasks, paired with their respective descriptions [14].

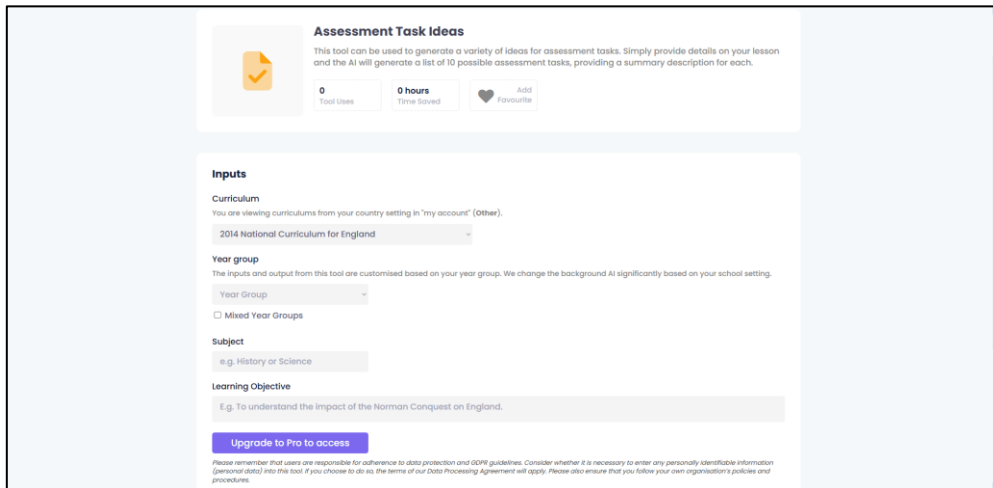
The screenshot shows the 'Assessment Task Ideas' web interface. At the top, there's a header with a yellow checkmark icon and the title 'Assessment Task Ideas'. Below the title, a brief description states: 'This tool can be used to generate a variety of ideas for assessment tasks. Simply provide details on your lesson and the AI will generate a list of 10 possible assessment tasks, providing a summary description for each.' To the right of this text are three buttons: 'Tool Uses', '0 hours Time Saved', and 'Add Favourite'. The main section is titled 'Inputs' and contains four input fields: 'Curriculum' (with a dropdown menu showing '2014 National Curriculum for England'), 'Year group' (with a dropdown menu and a checkbox for 'Mixed Year Groups'), 'Subject' (with a text input field showing 'e.g. History or Science'), and 'Learning Objective' (with a text input field showing 'E.g. To understand the impact of the Norman Conquest on England.'). At the bottom of the input section is a purple button labeled 'Upgrade to Pro to access'. A small disclaimer at the very bottom reads: 'Please remember that users are responsible for adherence to data protection and GDPR guidelines. Consider whether it is necessary to enter any personally identifiable information (personal data) into this tool if you choose to do so the terms of our Data Processing Agreement will apply. Please also ensure that you follow your own organisation's policies and procedures.'

Figure 2.2.2.2: Assessment Task Ideas Generator [14]

### 2.2.3 Marking.ai

Marking.ai is an AI-powered assessment tool which offers an array of tools that supports not only marking assessments, but also creating assessments and using the platform as a submission portal. Having more than 1000 users worldwide, this tool has been constructed using data centric insights provided by educators globally. Marking.ai aims to save time for educators by automating the process of marking assessments and collecting answer sheets from students. The platform features assessment setup tools, multiple questions and high accuracy marking, submission system and a student feedback system. With these tasks automated, educators will be handling a lower amount of workload compared to manually performing these tasks that takes up a big portion of their working hours.

The main feature of this platform starts by taking in user input to create an assessment. The input form takes in basic information like assessment name, subject, grade of year, total marks and due date. Educators are then required to upload two separate files, which are assessment questions and the marking rubric, followed by a text box that takes description of marking instructions from the educator so that the AI model has a better understanding of the marking task. After taking in all the necessary input, the tool will provide a URL of the assessment,



## CHAPTER 2

which will be given to the student to download the assessment. Students will then upload the answer sheets to the same link as submission of their work, and the marking process will be automatically executed by Marking.ai. For each automatically marked assessment, it will be put in the ‘to be reviewed’ state and educators need to review and check if all the assessments have been correctly marked, following the marking rubric provided earlier. Educators are able to make amendments to any mistakes Marking.ai has made and complete the assessment process. Marks are also calculated by the tool, and it will be released under the educator’s instructions [15].

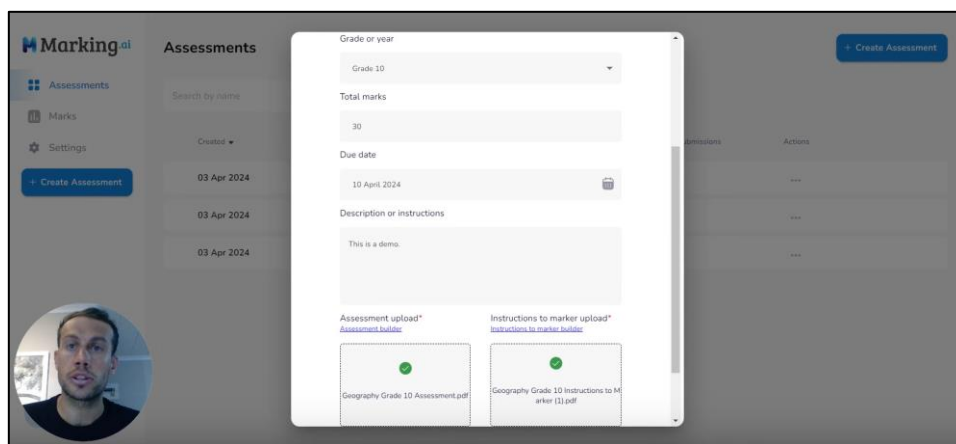
The screenshot shows the 'Create Assessment' form in the Marking.ai interface. On the left is a sidebar with navigation links: 'Assessments', 'Marks', 'Settings', and a 'Create Assessment' button. The main area is titled 'Assessments' and contains a search bar and a list of existing assessments, all dated '03 Apr 2024'. A modal form is open for creating a new assessment. It includes fields for 'Grade or year' (set to 'Grade 10'), 'Total marks' (set to '30'), and 'Due date' (set to '10 April 2024'). There is a 'Description or instructions' text area containing the text 'This is a demo.'. Below this are two sections: 'Assessment upload\*' with a file input showing 'Geography Grade 10 Assessment.pdf', and 'Instructions to marker upload\*' with a file input showing 'Geography Grade 10 Instructions to Marker (1).pdf'. A 'Create Assessment' button is in the top right corner.

Figure 2.2.3.1: Form to create an assessment [15]

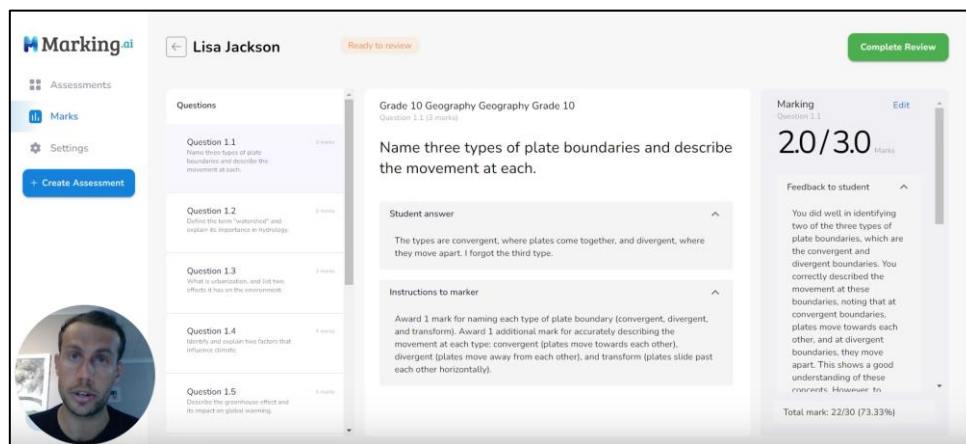
The screenshot shows the 'Review' page for a marked assessment in the Marking.ai interface. The user 'Lisa Jackson' is logged in, with a 'Ready to review' status and a 'Complete Review' button. The page is divided into three main sections. On the left is a 'Questions' list with five items, each with a '1 minute' timer. The middle section displays 'Grade 10 Geography' and 'Question 1.1 (2 marks)'. It shows the question text: 'Name three types of plate boundaries and describe the movement at each.' Below this is the 'Student answer' section with the text: 'The types are convergent, where plates come together, and divergent, where they move apart. I forgot the third type.' and the 'Instructions to marker' section with the text: 'Award 1 mark for naming each type of plate boundary (convergent, divergent, and transform). Award 1 additional mark for accurately describing the movement at each type: convergent (plates move towards each other), divergent (plates move away from each other), and transform (plates slide past each other horizontally)'. On the right is a 'Marking' section showing a score of '2.0/3.0' and a 'Feedback to student' section with a detailed comment: 'You did well in identifying two of the three types of plate boundaries, which are the convergent and divergent boundaries. You correctly described the movement at these boundaries, noting that at convergent boundaries, plates move towards each other, and at divergent boundaries, they move apart. This shows a good understanding of these movements. However, th...'. At the bottom right, it shows 'Total mark: 22/30 (73.33%)'.

Figure 2.2.3.2: Review page on marked assessment [15]

### 2.3 Strengths and Weaknesses of Existing Systems

From the three reviewed existing systems, each of them offer valuable features that support educators in different ways. Khanmigo stands out with its ability to adapt content based on student comprehension levels, using natural language processing to create personalised learning experiences. It also integrates a structured dashboard that makes its tools easy to

access, allowing educators to plan and manage their lessons efficiently. TeachMateAI is especially strong in content generation, offering tools like a slideshow generator and lesson planner that help educators save time during lesson preparation. It also includes an assessment idea generator, which can serve as a useful starting point for designing classroom evaluations. The clean, user-friendly interface contributes to the general experience of the system, making it accessible even to educators who are not deeply familiar with AI tools. Marking.ai introduced an innovative way to automate the tedious assessment workflow, where AI was used to mark student responses based on predefined rubrics and then generate structured feedback. This approach has the potential to save educators significant time, particularly when dealing with large volumes of submissions. In short, these systems display the growing role of AI in the field of education, aiming to reduce repetitive manual work by offering targeted, time-saving features that ease the burden on educators while enhancing efficiency in daily tasks.

There are quite a number of limitations on the previously reviewed systems. These limitations may be what functionalities the system is lacking, which functionalities can be improved, and which functionalities are not favourable. In Khanmigo, most of the features offered on the platform seem to assist educators by automating tasks that are related to the main objective of their job – teaching. While functionalities like the slideshow generator on TeachMateAI helps on generating content for teaching, the core responsibilities of an educator should never be replaced by anything as it is important for educators to familiarize themselves with what they are teaching. Hence, the lesson planner in Khanmigo should not be overused as educators may overly rely on this functionality and lose track on the knowledge that are being presented to students. The assessment idea generator from TeachMateAI may be useful, but it can be further improved by implementing a grading function for the assessments, which is a task that takes up a large portion of an educator's time. Most of the tools offered on this platform are also focused on suggesting teaching content for educators based on their input, which is not automating redundant tasks like marking assessments, data entry for grades and scheduling student meetings. Last but not least, Marking.ai has a concept of automated marking and human checking for assessment papers. While this may seem as a great idea, the marking should not be automated, as the AI model only follows the marking rubric in the marking process, which does not work great for open-ended questions. Assessments with open-ended questions or questions that require critical thinking skills should always be marked by a human being, not an AI model.

## 2.4 Comparison between Existing Systems

To provide a clearer understanding of the scope and design of existing systems, a comparison was made between Khanmigo, TeachMateAI, and Marking.ai. The comparison focuses on four main aspects, which were core functionalities, AI capabilities, cost and licensing, and user interface. These aspects were chosen to highlight the different approaches each system takes in supporting educators, and to establish a baseline for identifying how ProfMate differs in focus and implementation. The table below outlines these key elements for each system.

Table 2.5.1: Comparison between Existing Systems

	<b>Khanmigo</b>	<b>TeachMateAI</b>	<b>Marking.ai</b>
<b>Core Functionalities</b>	Lesson planner, text leveler	Writer, lesson planner, slideshow generator, assessment idea generator and more.	Assessment release and submission platform, automatic assessment marker with student feedback form.
<b>AI Capabilities</b>	Uses Natural Language Processing (NLP) to interact with students and create learning experiences based on students' level of understanding.	Generative AI for creating educational content such as slideshows, assessments and lesson plans based on user input.	AI-based assessment marking, on open-ended or essay questions. It analyses answers and evaluate based on given rubrics.
<b>Cost/Licensing</b>	Free for educators, USD\$ 4 per month for other user groups.	Free, limited version available. Upgradable to pro for GBP£ 6.99 per month.	Limited version available for free, with a quota of 2 submissions per month. Upgradable for USD\$ 35 per month.

<b>User Interface</b>	A comprehensive dashboard which includes categorised functionalities which is easily accessible. Mobile responsive.	A page that serves a library of tools classified by categories and collections. Mobile responsive and good colour combinations.	Simple and tidy design, mobile responsive, and easy navigation.
-----------------------	---	---	---

## Chapter 3

### System Methodology/Approach

This project aims to propose an application to better assist educators on side tasks that are not mainly related to the teaching process, tasks that can be automated such as a tool to check marked assessments papers, grade-keeping system, and student meeting scheduling portal. Instead of developing tools that automates tasks that are closely related to the teaching process, the proposed application will focus on automating side tasks, allowing educators to put additional focus on their main responsibilities on effectively and efficiently transferring knowledge to students.

#### 3.1 Methodology

The proposed methodology to be used in project is Rapid Application Development (RAD). This software development methodology emphasises quick development and is driven by user interface needs. The flow of RAD consists of requirements planning phase, user design phase, construction phase and finally the cutover phase.

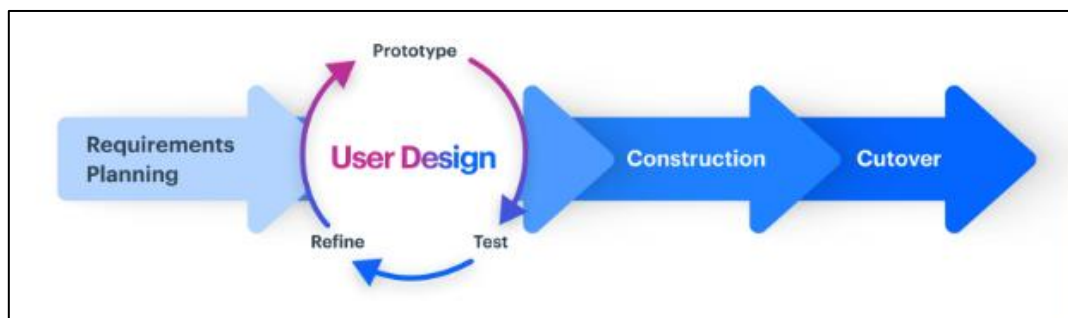


Figure 3.1.1: Flow of RAD [16]

#### Requirements Planning Phase

In requirements planning, it starts with the initial idea of proposing a solution to reduce the workload and burden of educators developing an online teaching assistant. The problem statements are listed out in detail to highlight the problems to be solved. Next, the objectives of this project are stated to firmly establish the goals and direction of this project. Previous studies on image recognition and object detection concepts were reviewed and existing systems were analysed to determine the strengths and weaknesses that were used to construct a set of guidelines to build the proposed system. The scope of the project, which includes the description of the modules to be built for the system were clearly stated to meet the requirements to reach the objectives.

### User Design Phase

For the phase of user design, it includes three different stages, which are prototyping, testing and refining. In each cycle, one module or functionality of the web application is handled. The modules that will be designed here are marked paper review tool, automated grade entry, meeting scheduler, login module and results portal. The machine learning model and object detection in image recognition that powers the marked paper review tool will be written in Python using Google Colab. The user interface of ProfMate and remaining modules will be built using the ASP .NET Core MVC framework written in C#.

### Construction Phase

The construction phase is where the ultimate designs of the modules are brought to life, by coding and integrating all modules together and form a preliminary version of the web application. The modules and overall design of the user interface will be fine-tuned and further customised to improve the appearance and performance of ProfMate.

### Cutover Phase

Lastly, the cutover phase will include final testing on ProfMate, to ensure that web application is functioning correctly and meet the requirements determined in the first phase. This phase is utmost important as it will be a performance evaluation on the web application before deployment. Errors found in this phase will be corrected until the testing finds no other errors, and the project is declared complete and ready for release.

## 3.2 System Requirement

### 3.2.1 Hardware

The hardware involved in this project is a desktop computer. The computer was used to develop and train the object detection model used in ProfMate. The user interface of ProfMate was also developed and tested using this computer, along with the deployment of an API for the model.

Table 3.2.1.1 Specifications of desktop PC

Description	Specifications
Processor	AMD Ryzen 5 3600 6-Core Processor
Operating System	Windows 10 Pro 22H2
Graphic	NVIDIA GeForce GTX 1080
Memory	16GB DDR4 RAM

Storage	500GB NVME SSD + 480GB SATA SSD
---------	---------------------------------

### 3.2.2 Software

The software components involved in this project are separated into two sections, which are tools and programming languages, frameworks and libraries.

Table 3.2.2.1 Software components and specifications

Component	Specifications
Tools	<b><u>Visual Studio Code</u></b> Visual Studio Code (VS code) is an open-source code editor developed by Microsoft. Supports multiple programming languages like C++, HTML and JavaScript. It offers features such as syntax highlighting, intelligent code completion, and extensions for additional functionalities.
	<b><u>Google Colaboratory (Colab)</u></b> Google Colaboratory (Colab) is a free cloud-based development environment provided by Google. It allows users to write and execute Python code in Jupyter notebooks with no local setup required. Colab supports GPU and TPU acceleration.
	<b><u>Labellmg</u></b> Labellmg is an open-source graphical image annotation tool used for labeling images with bounding boxes. It allows users to annotate objects in images and export annotations in standard formats like Pascal VOC.
	<b><u>Iruin Webcam</u></b> Iruin Webcam is a free application that allows users to use their smartphones as a high-quality webcam for PC or Mac. This allowed wireless or USB connection for smartphones to be used as a webcam.
Languages, Frameworks, Libraries	<b><u>Python</u></b> Python is a high-level, interpreted programming language known for its simplicity and readability. It is widely used in data science, web development, and automation.

	<p><b><u>PyTorch</u></b></p> <p>PyTorch is an open-source machine learning library developed by Facebook. It provides a flexible framework for building and training deep learning models, particularly useful for computer vision and natural language processing tasks.</p>
	<p><b><u>Faster R-CNN</u></b></p> <p>Faster R-CNN (Region-based Convolutional Neural Network) is a state-of-the-art deep learning model used for object detection tasks. It combines a region proposal network (RPN) with a CNN to efficiently detect objects in images.</p>
	<p><b><u>React</u></b></p> <p>React is a JavaScript Library developed by Facebook for building dynamic, interactive and reusable user interfaces. It allows the creation of component-based architectures that can be reused across the application.</p>
	<p><b><u>SQLite</u></b></p> <p>SQLite is a lightweight, self-contained relational database management system that stores data in a single file. It supports standard SQL queries, ACID transactions, and requires no separate servers, making it highly portable and easy to integrate into applications.</p>
	<p><b><u>HTML, CSS, JavaScript</u></b></p> <p>HTML (HyperText Markup Language), CSS (Cascading Style Sheets), and JavaScript are fundamental web technologies used to build the structure, style, and interactivity of web pages. HTML provides the skeleton of the webpage, CSS is used for layout and design, and JavaScript adds interactivity, such as handling user inputs and dynamically updating content.</p>



### 3.3 System Design Diagram

#### 3.3.1 Use Case Diagram

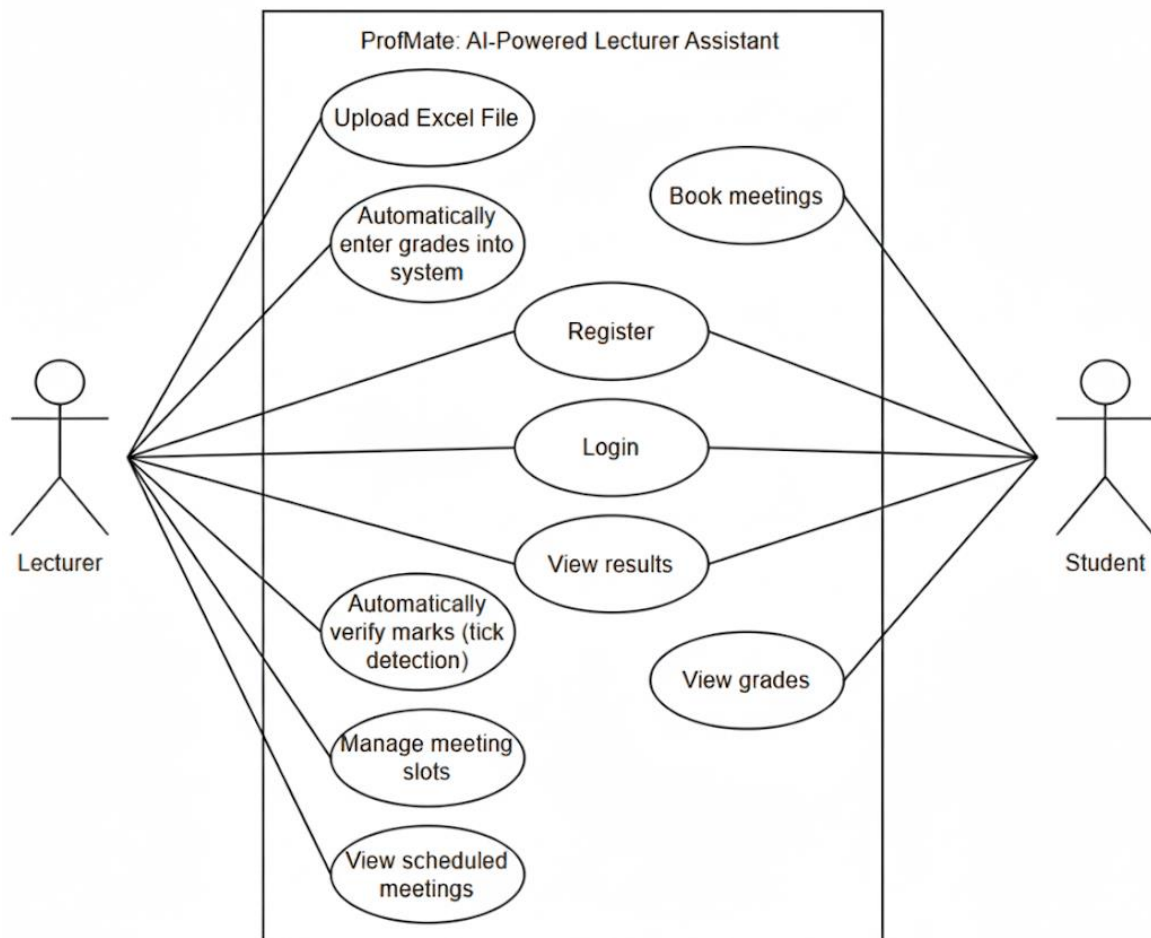


Figure 3.3.1.1: Use case diagram for ProfMate

#### 3.3.2 Use Case Description

Table 3.3.2.1 Login Use Case Description

Use Case ID	001
Use Case Name	Login
Brief Description	Lecturers and students are allowed to access the ProfMate platform by logging in with their credentials.
Actor	Lecturer, Student
Trigger	User navigates to the ProfMate login page.
Precondition	User account must already exist in the system and must not be currently logged in.
Normal flow of events	1. Lecturer or student fills in email and password.

	<ol style="list-style-type: none"> <li>2. The system validates whether the account exists, and the email and password are correct.</li> <li>3. The system creates a session for the user.</li> <li>4. The system redirects the user to the respective dashboard based on their role.</li> </ol>
Sub flows	<ol style="list-style-type: none"> <li>3a. If the session is still active, the user does not need to log in again within the session duration.</li> <li>3b. If the session expires, the user is required to re-login to access the platform.</li> </ol>
Alternate flows	<ol style="list-style-type: none"> <li>2a. If the account does not exist, or the email and password combination is invalid, the system displays an error message.</li> <li>2b. User is prompted to reinsert correct email and password credentials.</li> </ol>

Table 3.3.2.2 Register Use Case Description

Use Case ID	002
Use Case Name	Register
Brief Description	Students and lecturers can create a new account on the ProfMate platform to access system features.
Actor	Lecturer, Student
Trigger	User navigates to the Registration page.
Precondition	User must not already have an existing account registered with the same username.
Normal flow of events	<ol style="list-style-type: none"> <li>1. User fills in required registration details (username, password, role: student or lecturer).</li> <li>2. The system validates the inputs (e.g., proper username format, password strength).</li> <li>3. The system checks if the username is unique (not already registered).</li> <li>4. The system creates a new account and stores the user information securely.</li> </ol>

	5. The system redirects the user to the login page with a success message.
Sub flows	2a. The system suggests improvements if the password entered is too weak.
Alternate flows	3a. If the username already exists in the system, an error message is displayed, and the user is asked to login instead. 2b. If input validation fails (e.g., missing required fields), the user is prompted to correct them.

Table 3.3.2.3 View Results Use Case Diagram

Use Case ID	003
Use Case Name	View Results
Brief Description	Students and lecturers can view results of reviewed assessments through a centralized portal.
Actor	Lecturer, Student
Trigger	User navigates to the "View Results" section.
Precondition	User must be logged in.
Normal flow of events	1. Student or lecturer accesses the results portal. 2. The system retrieves the relevant assessment results. 3. Results are displayed, categorized by assessment type (e.g., quiz, midterm).
Sub flows	2a. Lecturer may choose to filter results by student or subject.
Alternate flows	2b. If no results are available, the system displays a "No results available" message.

Table 3.3.2.4 Upload Excel File Use Case Diagram

Use Case ID	004
Use Case Name	Upload Excel File
Brief Description	Lecturers can upload an Excel file containing student grades, which will be processed and stored in the system database.
Actor	Lecturer
Trigger	Lecturer selects and uploads an Excel file.

Precondition	Lecturer must be logged into the platform.
Normal flow of events	<ol style="list-style-type: none"> <li>1. Lecturer selects an Excel file from local storage.</li> <li>2. The system validates the file format and processes the uploaded data.</li> <li>3. Student grades are extracted and stored in the database.</li> <li>4. A confirmation message is displayed to the lecturer.</li> </ol>
Sub flows	2a. The system checks that the file type is supported (.xls or .xlsx) before processing.
Alternate flows	2b. If the file is invalid or corrupted, the system displays an error message and prompts the lecturer to re-upload.

Table 3.3.2.5 Automatically Verify Marks (Tick Detection) Use Case Diagram

Use Case ID	005
Use Case Name	Automatically Verify Marks (Tick Detection)
Brief Description	System automatically verifies if all questions are marked and if the total marks tally up correctly using tick detection.
Actor	Lecturer
Trigger	Triggered as part of the Review Marked Papers workflow.
Precondition	Lecturer must upload assessment papers.
Normal flow of events	<ol style="list-style-type: none"> <li>1. System detects tick marks on the scanned papers.</li> <li>2. System calculates if all questions have been attempted and marked.</li> <li>3. System checks if the total marks match the expected value.</li> </ol>
Sub flows	2a. Lecturer can manually verify markings if automated detection identifies uncertainty.
Alternate flows	3a. If the paper contains unrecognized markings, the system flags it for manual checking.

Table 3.3.2.6 Automatically Enter Grades into System Use Case Diagram

Use Case ID	006
Use Case Name	Automatically Enter Grades into System
Brief Description	After verifying papers, system extracts student grades and details to automatically record into the database.
Actor	Lecturer
Trigger	Completion of paper review and verification.
Precondition	Paper review must be completed successfully.
Normal flow of events	<ol style="list-style-type: none"> <li>1. System extracts student name, ID, and total marks from scanned papers.</li> <li>2. System cross-references extracted data with existing student records.</li> <li>3. Grades are automatically inserted into the grade database.</li> </ol>
Sub flows	2a. If minor mismatches are detected, lecturer can manually correct before submission.
Alternate flows	3a. If student record cannot be matched, system prompts lecturer to manually input missing information

Table 3.3.2.7 Manage Meeting Slots Use Case Diagram

Use Case ID	007
Use Case Name	Manage Meeting Slots
Brief Description	Lecturers can manage their available time slots for student consultations.
Actor	Lecturer
Trigger	Lecturer navigates to meeting slot management section.
Precondition	Lecturer must be logged into the platform.
Normal flow of events	<ol style="list-style-type: none"> <li>1. Lecturer views current available slots.</li> <li>2. Lecturer adds, edits, or deletes time slots.</li> <li>3. The system updates the available slots database.</li> </ol>
Sub flows	2a. Lecturer can set specific rules like meeting duration or maximum bookings per day.

Alternate flows	2b. If an existing slot already has a booking, the system prevents deletion and prompts rescheduling instead.
-----------------	---

Table 3.3.2.8 View Scheduled Meetings Use Case Diagram

Use Case ID	008
Use Case Name	View Scheduled Meetings
Brief Description	Lecturers can view all meetings booked by students.
Actor	Lecturer
Trigger	Lecturer selects "View Scheduled Meetings."
Precondition	Lecturer must be logged in.
Normal flow of events	<ol style="list-style-type: none"> <li>1. Lecturer accesses scheduled meetings page.</li> <li>2. System retrieves upcoming and past meetings data.</li> <li>3. Meetings are displayed in an organized view (by date, time).</li> </ol>
Sub flows	2a. Lecturer can filter by date or student.
Alternate flows	3a. If no meetings are scheduled, the system displays a "No upcoming meetings" message.

Table 3.3.2.9 View Grades Use Case Diagram

Use Case ID	009
Use Case Name	View Grades
Brief Description	Students can view their individual grades through their personal portal.
Actor	Student
Trigger	Student navigates to "View Grades" section after login.
Precondition	Student must have grades available in the system.
Normal flow of events	<ol style="list-style-type: none"> <li>1. Student accesses "View Grades" section.</li> <li>2. The system retrieves and displays grades by subject and assessment.</li> <li>3. Grades are shown with details such as assessment name, date, and total marks.</li> </ol>
Sub flows	2a. Student can select a specific subject to view detailed results.

Alternate flows	3a. If no grades are available yet, the system informs the student appropriately.
-----------------	---

Table 3.3.2.10 Book Meetings Use Case Diagram

Use Case ID	010
Use Case Name	Book Meetings
Brief Description	Students are allowed to book available meeting slots with lecturers through the platform.
Actor	Student
Trigger	Student selects a lecturer and initiates a meeting booking.
Precondition	Student must be logged into the platform.
Normal flow of events	<ol style="list-style-type: none"> <li>1. Student views available meeting slots of the selected lecturer.</li> <li>2. Student selects a preferred date and time slot.</li> <li>3. The system confirms the slot is available.</li> <li>4. The booking is recorded and displayed in both the student and lecturer's schedules.</li> </ol>
Sub flows	2a. The student can view multiple available slots before making a selection.
Alternate flows	3a. If the selected slot is no longer available, the system displays an error and prompts the student to choose another available time.

## CHAPTER 4

# System Implementation

### 4.1 Program Development

The ProfMate web application was developed using a full-stack JavaScript and Python-based approach, integrating both web development and AI-powered functionalities. The frontend was built with React.js, a component-based architecture and React Hooks for dynamic UI updates. The backend was implemented with Node.js and Express.js, providing RESTful APIs for the tick scanner, authentication, student bookings, and lecturer meeting slot management. SQLite was used as the relational database which manages tables for users, students, lecturers, meeting slots, bookings, and student scores.

#### 4.1.1 Registration / Authentication Module

The registration and authentication module in ProfMate provides secure access control for both students and lecturers. During registration, users provide essential information such as name, email, password, and role (student or lecturer). Passwords are securely hashed using bcrypt before being stored in the database, ensuring sensitive credentials are never saved in plaintext. For student users, the system optionally links their account to a student record in the students table, while lecturer accounts can be associated with a lecturers table entry containing faculty information.

Authentication is handled through session-based login using express-session, which maintains user sessions across requests. On login, the system verifies credentials by comparing the hashed password stored in the database with the input password. Upon successful login, relevant user details, including user\_id, role, and optionally student\_id or lecturer\_id. These are stored in the session, allowing the backend to enforce role-based access control. This ensures that students can only book meetings and view their bookings, whereas lecturers can manage slots and approve or reject bookings. All the endpoints for the aforementioned functionalities are defined as such in Figure 4.1.1.1.



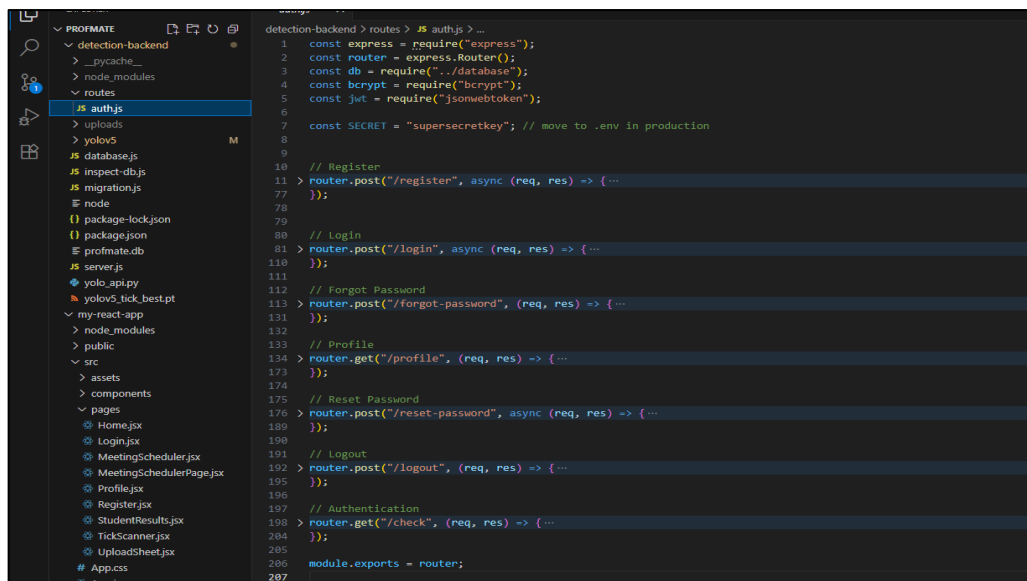


Figure 4.1.1.1: auth.js

## 4.1.2 Home Page

The home page serves as the central dashboard for all users. The page dynamically its content based on the logged-in user's role. Upon login, the system checks the user session and renders role-specific interfaces. For students, the user is presented with quick access to their booked meetings and check results page, while lecturers see an overview of their managed slots, pending booking requests, and administrative actions such as approving or cancelling meetings. The home page also provides a navigation bar to other modules, such as the meeting scheduler, results portal, and grade management, like in Figure 4.1.2.1.

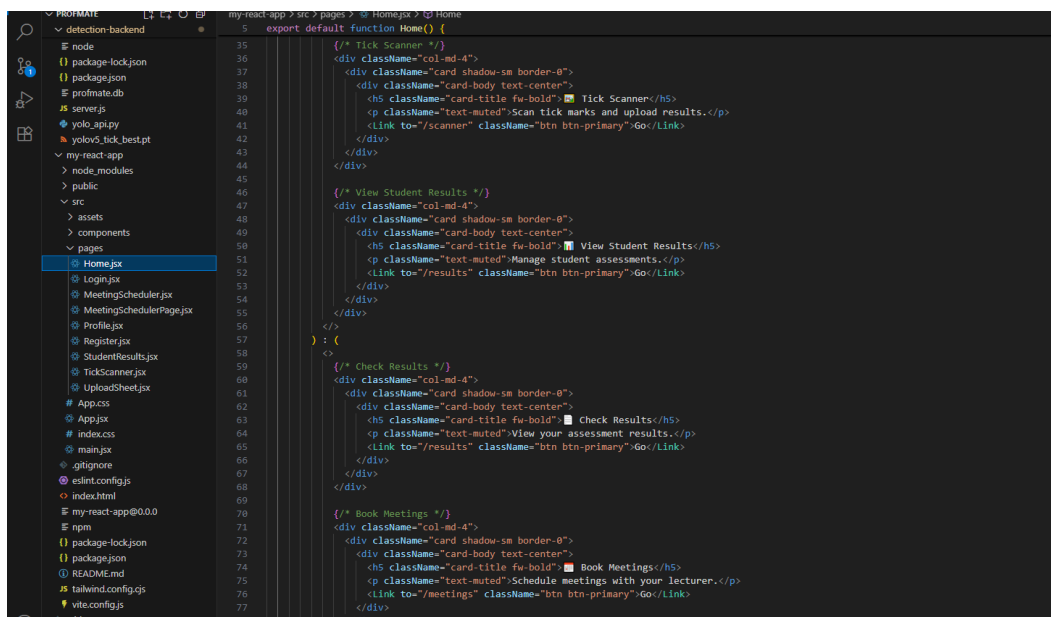


Figure 4.1.2.1: Home.jsx

### 4.1.3 Tick Scanner

The Tick Scanner page, with code shown in Figure 4.1.3.1, allows lecturers to feed live captures of answer sheets to the tick-detection model. For each sheet the model detects individual tick marks, with bounding boxes and confidence scores. Those detections are then added into a running total for that student and immediately compared against the stored value in the database. If the running total matches the stored value the result can be flagged as verified, otherwise the lecturer can inspect detections, accept or correct the count, and then save a verified score. The page also records the final total in the `student_scores` table.

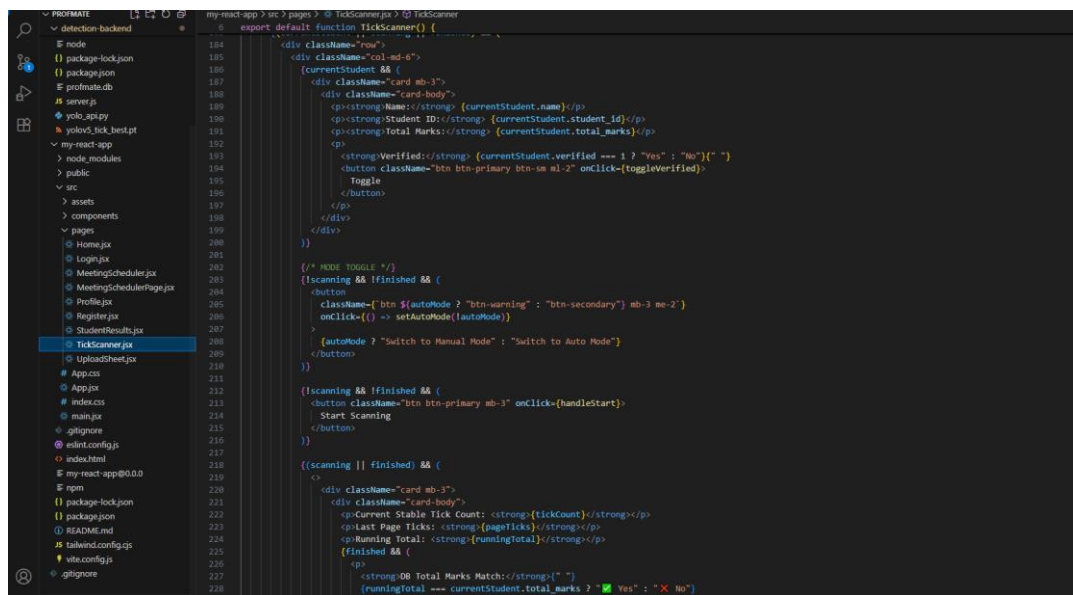


Figure 4.1.3.1: TickScanner.jsx

### 4.1.4 Results Portal

The results portal with the code in Figure 4.1.4.1 serves different purposes to the two different roles, which are lecturers and students. For lecturers, the results portal is used to upload excel sheets to store the total marks of each student. Rows of data in the excel sheet uploaded are stored into the database, which can be used in the tick scanner to compare with the number of ticks detected so review the marked paper scanned. After the scores are verified, the results will be released to the students and each student can view their own results only. For the lecturers, the results portal also allows editing of student scores, verified status and student details. The page is dynamically rendered based on the role of the user.

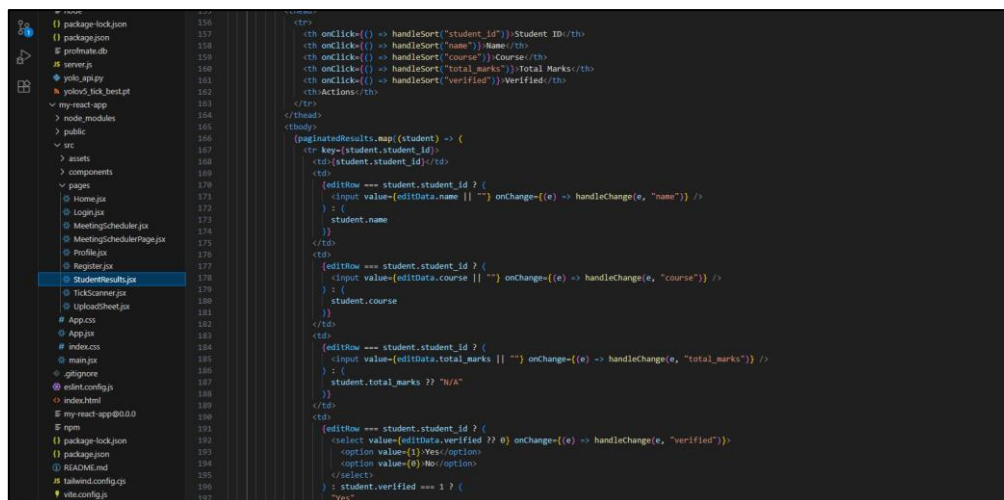


Figure 4.1.4.1: StudentResults.jsx

### 4.1.5 Meeting Scheduler

For the meeting scheduler, it is also implemented following a role-based approach, where the view and available actions of the page depends on the role of the logged in user. For students, the page renders a dropdown to select lecturers and view the available meeting slots posted by that lecturer. Students can book the meetings and the list of bookings obtained from the endpoint shown in Figure 4.1.5.1 are displayed with the status, which can be “Confirmed”, “Pending” or “Cancelled”. For cancelled bookings, lecturers can add an optional remark to inform the student about the reason for cancellation or rescheduling. For the lecturers, the meeting scheduler allows meeting slot creation, deleting slots, confirming or cancelling bookings, and adding remarks to cancelled bookings. Meeting slots and bookings are stored as two separate tables in the database.

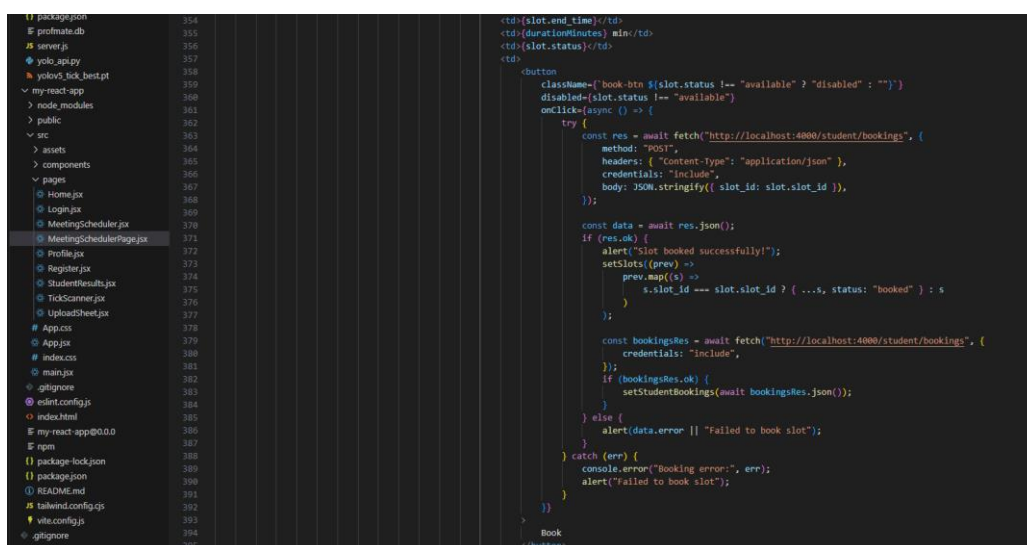


Figure 4.1.5.1: MeetingScheduler.jsx

### 4.1.6 Profile Page

A simple profile page was created with the codes in Figure 4.1.6.1, which displays stored user information, with slight differences between the two user roles. For students, student ID number and course is displayed, while for lecturers, faculty is displayed. The information displayed is obtained from endpoints defined in server.js.

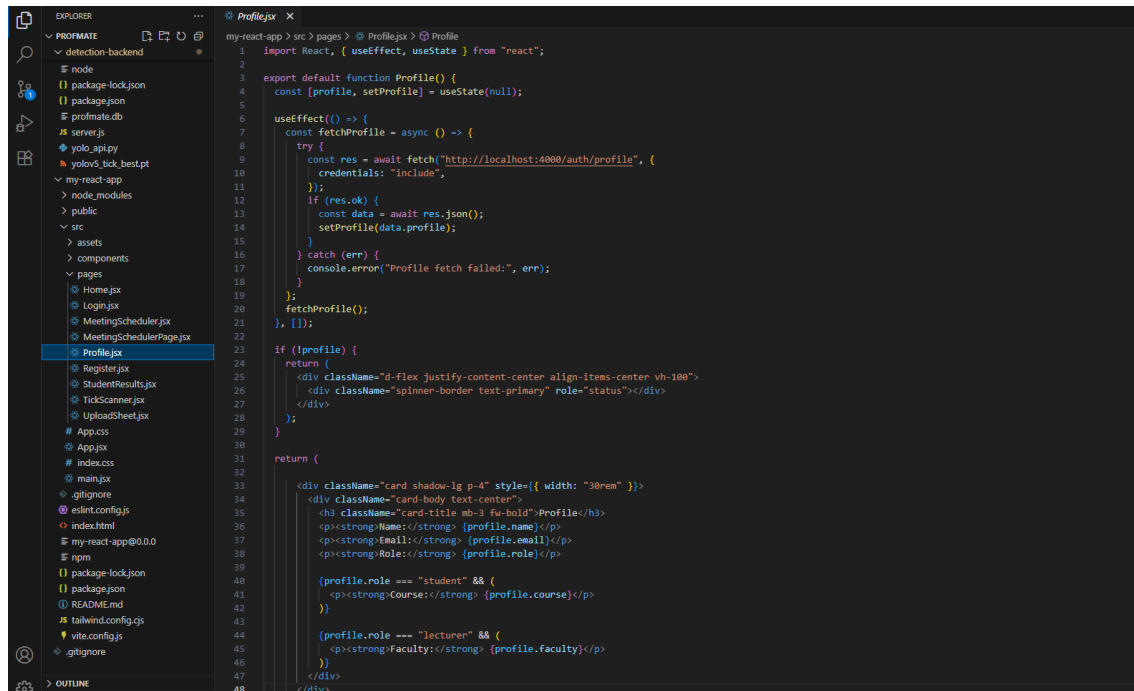


Figure 4.1.6.1: Profile.jsx

### 4.1.7 Back-end Implementation

The back-end of ProfMate was developed using Node.js with Express.js and serves as the foundation for API endpoints, authentication, and database management. The server.js file, shown in Figure 4.1.7.1, is the main entry point where core middleware is configured. This includes express.json() for handling JSON requests, express-session for maintaining user sessions, bcrypt for password hashing, and CORS policies to ensure secure cross-origin communication with the front-end.

Authentication is enforced through middleware functions that verify whether a user is logged in before granting access to protected routes. Once a login request is successful, a session object is created to store user details such as the unique user identifier, role, and other attributes. This enables role-based access control so that lecturers and students are presented with different parts of the system while sharing the same authentication flow.

The backend exposes several RESTful endpoints that are grouped by functionality. The authentication routes include register, login, logout, and session validation, which handle user creation, authentication, session management, and validation of current sessions. Meeting scheduler routes allow lecturers to create, update, and confirm meeting slots, while students can fetch available slots and review their own bookings. The results portal routes provide access to assessment outcomes stored in the database. Additional endpoints also support the tick scanner, where detected tick counts are compared with values stored in the database in order to validate the grading process automatically.

All persistent data is managed through a SQLite database, which is integrated directly with the Express server using SQL queries with `db.prepare()`. The schema enforces relational integrity with the users table serving as the base entity and linking to role-specific tables such as lecturers. Foreign key constraints maintain data consistency, while insert, update, and retrieval operations are coupled with backend endpoints to ensure secure and reliable transactions.

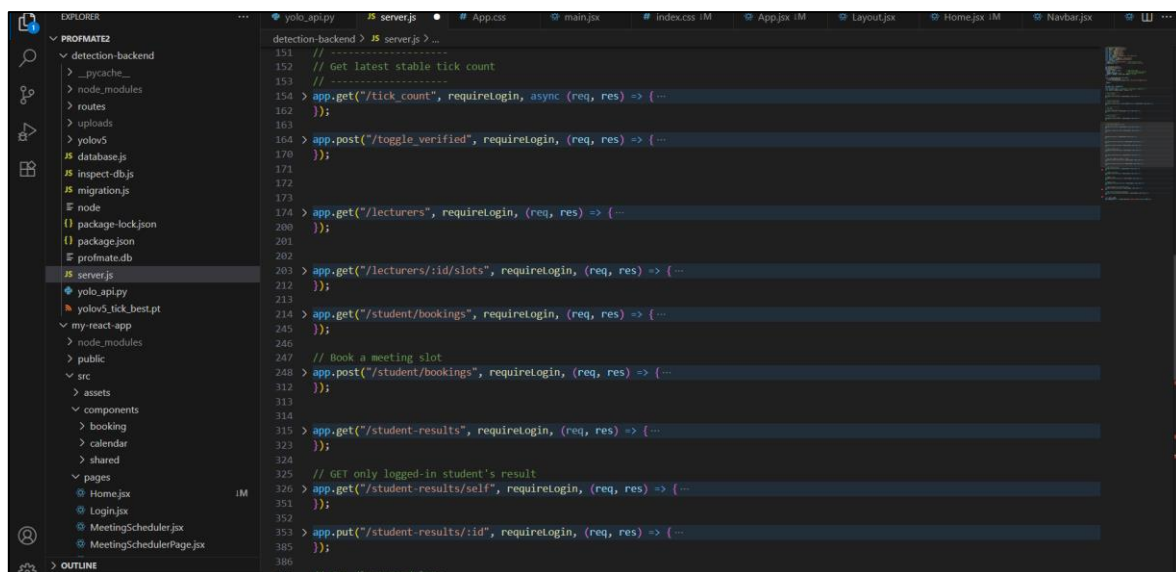


Figure 4.1.7.1: server.js

#### 4.1.8 Real-Time Tick Detection API

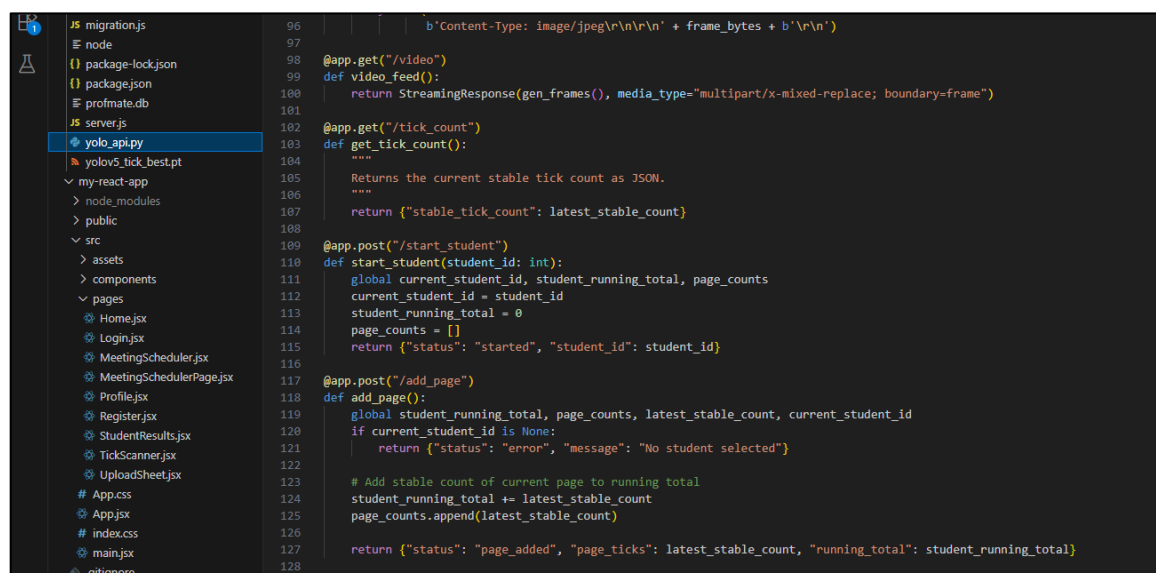
The tick detection module was implemented as in Figure 4.1.8.1, as a separate microservice using FastAPI and integrates directly with a YOLOv5 object detection model trained on tick annotations. This service is responsible for real-time detection of tick marks from a live camera feed and provides endpoints for both video streaming and count retrieval.

The system initializes the YOLOv5 model by loading custom-trained weights and configuring it on the appropriate device, whether CPU or GPU. A webcam feed is captured with OpenCV and preprocessed using letterboxing to match the model input dimensions. Each frame is then passed through the model, followed by non-maximum suppression to filter overlapping detections. Bounding boxes and confidence scores are drawn on the video frames to visualize detected ticks.

To ensure robustness in real-time counting, the API maintains a sliding window of the last thirty frame predictions and computes the most frequent tick count in that range. This produces a stable tick count that reduces flickering from frame-to-frame inconsistencies. The stable count is displayed on the live stream and also exposed through a dedicated endpoint.

The FastAPI service exposes several endpoints. The video endpoint streams the webcam feed as a sequence of JPEG-encoded frames with bounding boxes and a live stable tick count. The tick count endpoint returns the current stable tick count in JSON format. The start student endpoint initializes a session for a student, resetting the running total and preparing to track multiple pages. The add page endpoint adds the stable tick count for the current page to the student's running total and stores per-page tick counts.

Through these endpoints, the tick detection service functions as a real-time API that can be consumed by the ProfMate frontend. The detected tick counts are compared with stored values in the database, automating the process of result verification and reducing the workload of lecturers.



```

96 |         b'Content-Type: image/jpeg\r\n\r\n' + frame_bytes + b'\r\n')
97 |
98 | @app.get("/video")
99 | def video_feed():
100 |     return StreamingResponse(gen_frames(), media_type="multipart/x-mixed-replace; boundary=frame")
101 |
102 | @app.get("/tick_count")
103 | def get_tick_count():
104 |     """
105 |     Returns the current stable tick count as JSON.
106 |     """
107 |     return {"stable_tick_count": latest_stable_count}
108 |
109 | @app.post("/start_student")
110 | def start_student(student_id: int):
111 |     global current_student_id, student_running_total, page_counts
112 |     current_student_id = student_id
113 |     student_running_total = 0
114 |     page_counts = []
115 |     return {"status": "started", "student_id": student_id}
116 |
117 | @app.post("/add_page")
118 | def add_page():
119 |     global student_running_total, page_counts, latest_stable_count, current_student_id
120 |     if current_student_id is None:
121 |         return {"status": "error", "message": "No student selected"}
122 |
123 |     # Add stable count of current page to running total
124 |     student_running_total += latest_stable_count
125 |     page_counts.append(latest_stable_count)
126 |
127 |     return {"status": "page_added", "page_ticks": latest_stable_count, "running_total": student_running_total}
128 |

```

Figure 4.1.8.1: yolo\_api.py

### 4.2 Implementation Issues and Challenges

There were several issues and challenges faced during the model and user interface development of ProfMate in this project.

#### **Variable Mark Weight Per Tick**

One of the notable challenges encountered during the implementation of ProfMate was the variable mark weight per tick on certain exam papers. The tick detection system was designed under the assumption that each tick carried a single mark, allowing for a direct accumulation of scores as ticks were detected. However, some exam papers included questions where each tick represents more than one mark. This inconsistency created a situation where the running total calculated by the system did not always match the actual marks recorded on the exam paper. To resolve this issue, it required careful consideration on how to account for different mark weights while maintaining the efficiency of the tick counting. Ensuring that system could handle this challenge without introducing errors in the total score calculation was a key implementation challenge, which highlighted the complexities involved in adapting AI detection systems to real-world exam formats where marking schemes and questions are varied.

#### **Overlapping Tick Marks**

In some exam papers, multiple ticks can appear close to each other or even overlap, which poses a challenge for the detection model. The model may incorrectly merge two or more ticks into a single detection, split one long tick into two small ticks, or completely miss some ticks due to their proximity. This affects both the accuracy of the live tick count and the final running total of marks. To suppress this, buffer-based stability checks and careful calibration of detection thresholds were necessary. Handling overlapping ticks remains a key difficulty in achieving consistent and reliable tick detection in real-time scanning.

#### **Limited Dataset Size Impact on Custom CNN-Based Tick Detection Model**

During the development of the custom CNN-based tick detection model, one of the significant challenges was the limited size of the available dataset. Convolutional neural networks generally need a large amount of diverse data to learn robust and generalizable features, which a small dataset can severely restrict the model to perform well on unseen data. With only a few hundred annotated images, the custom CNN was prone to overfitting, where it learned to recognize patterns specific to the training set rather than general features of tick

marks. This limitation required careful consideration of data augmentation techniques, regularization methods, and careful monitoring of the training process to prevent the model from memorizing the training data. The restricted dataset size also posed difficulties when attempting to compare the custom CNN fairly against other models such as Faster R-CNN or YOLOv5n, as the performance could be constrained by the lack of sufficient training examples rather than the model architecture itself. Addressing this challenge highlighted the importance of dataset quality and quantity in developing reliable object detection systems.



## CHAPTER 4

### 4.3 Gantt Chart

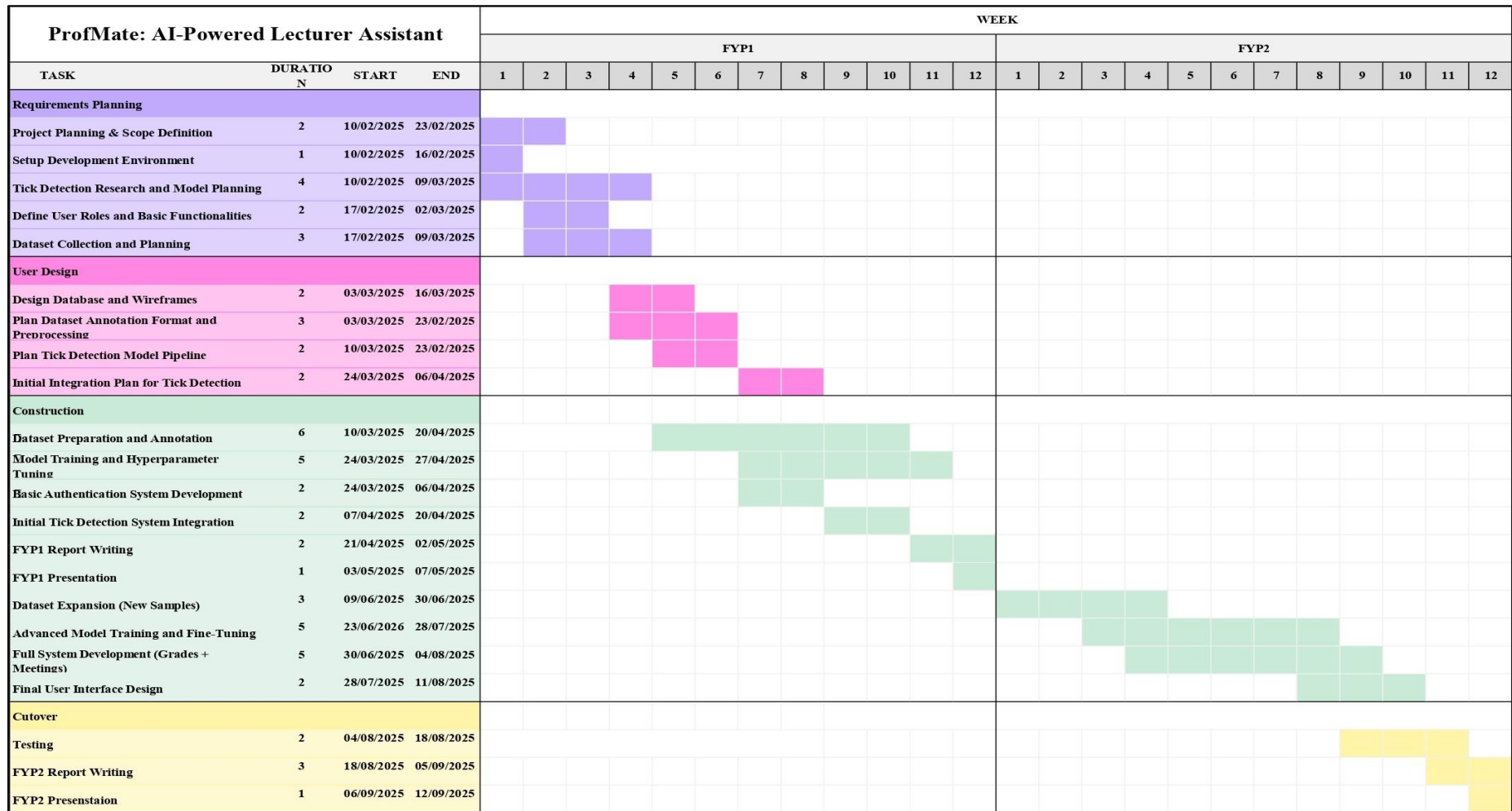


Figure 4.3.1: Gantt Chart of Project Timeline

### 4.4 Summary

In a nutshell, this chapter has described the full system implementation of ProfMate, covering both the frontend and backend development as well as the integration of the AI-powered tick detection service. It outlined the main modules such as authentication, the tick scanner, results portal, meeting scheduler, and profile page, while also addressing the technical challenges faced during development. Together, these implementations form the foundation for evaluating the system's performance in the next chapter.

## CHAPTER 5

### System Evaluation and Discussion

In this project, the ProfMate web application was developed, along with a real-time tick detection model deployed as an API. Modules that were developed for the ProfMate web application include a tick scanner using real-time tick detection model, an authentication module, a results portal, and a meeting scheduler and booking platform.

#### 5.1 Tick Scanner

In this project, a YOLOv5n object detection model was trained to detect tick marks on marked exam papers, and subsequently the model was deployed to an API service that provides live camera footage to the user interface of ProfMate, where detections are performed on every frame. The live camera footage visualises the live detection of tick marks by drawing bounding boxes around the ticks on the window. In the front-end, the ProfMate web application tick scanner page accesses this API service through an endpoint, and users can pick a student before starting to scan a paper. Coming into this page, users can pick students with scores stored in the database from an excel spreadsheet upload before coming to this page, and click start scanning, as shown in Figure 5.1.1.

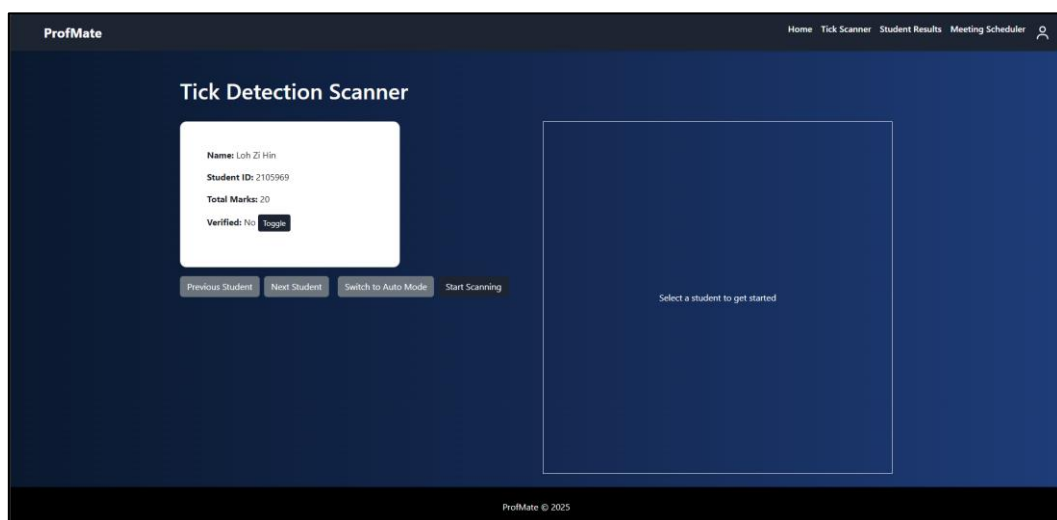


Figure 5.1.1: Tick Detection Scanner Page (before scanning starts)

Upon clicking on start scanning, live webcam footage will be shown on the right side of the screen, where tick detections are being done, frame by frame. Since bounding boxes flickers on low confidence detections, the stable count of ticks is continuously displayed so that users know the mode detection count, which can be used as the actual count.

The scanner supports two distinct modes of operation, manual and automatic. In manual mode, as shown in Figure 5.1.2, lecturers can review the stable tick count displayed on the interface and decide when to add the count for a page to the student's running total by clicking the add page button. This gives lecturers control to verify detections before updating the database. In automatic mode, as shown in Figure 5.1.3, the system continuously monitors the stable tick count across multiple frames and, once it reaches a consistent value, it automatically adds the detected ticks to the student's total without requiring lecturer input. After scanning all pages, lecturers can finish the process, which finalises the student's total and compares it with the marks stored in the database. This dual-mode approach balances automation with lecturer oversight, ensuring flexibility in different scanning situations while maintaining accuracy and efficiency in tick mark detection.

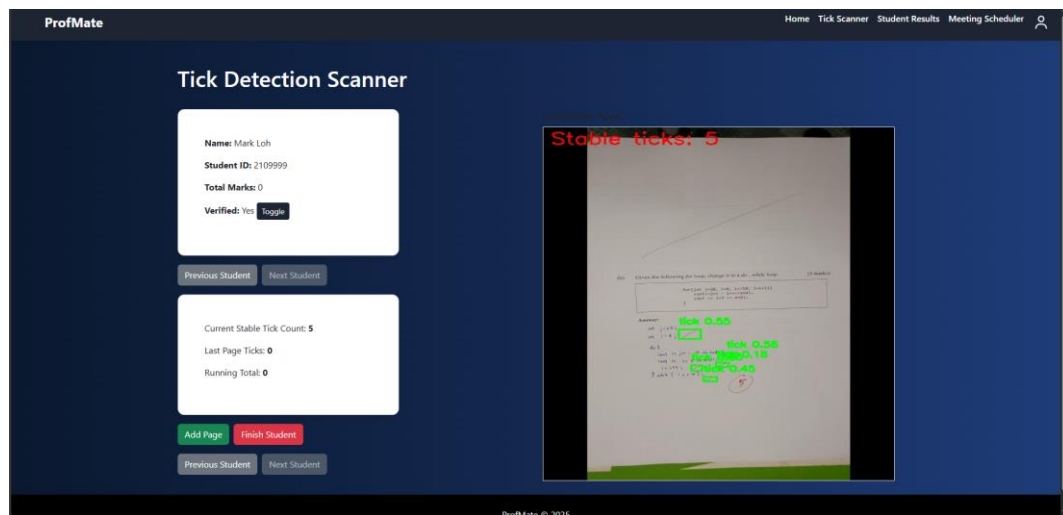


Figure 5.1.2: Tick Detection Scanner (Manual Mode)

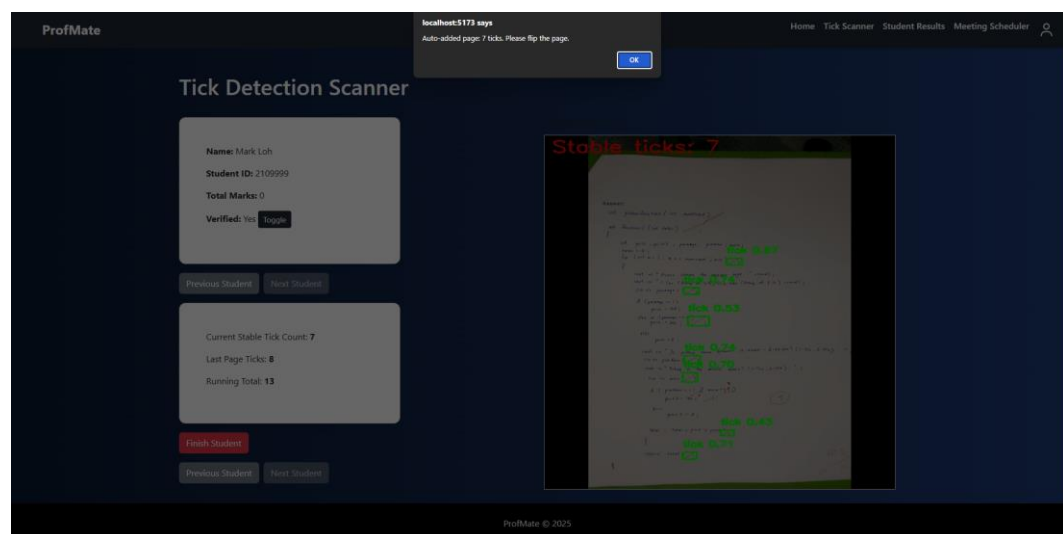


Figure 5.1.3: Tick Detection Scanner (Automatic Mode)

## 5.2 Results Portal

In ProfMate, a results portal was also developed to act as a centralised platform for lecturers to manage results in a more effective and organised way. This platform allows lecturers to upload and release results to students without having to manually create a spreadsheet and upload to a separate platform. Students can log in with their credentials and securely access and view examination results, where the marks are presented in a clear and structured format. This allows them to immediately see their performance in each assessment without confusion or delays.

For lecturers, the results portal provides a more extensive set of features. Since the system has multiple roles access, only lecturers can upload, edit and review marks that have been generated either manually or through automated tick detection, as shown in Figure 5.2.1. Lecturers also have the ability to ensure that marks align with examination schemes, make corrections if needed and then publish the results for students to see. Role-based access ensures that permissions are carefully controlled, so that only authorised users can update or validate marks while students remain limited to viewing results.

	Student ID	Name	Course	Total Marks	Verified	Actions
<input type="checkbox"/>	1702744	CHEW JOSHUA	IB	28	No	<a href="#">Edit</a>
<input type="checkbox"/>	1803271	ENG MUN SHUEN	IA	15	Yes	<a href="#">Edit</a>
<input type="checkbox"/>	1803578	LIM WEN TING	CN	22	No	<a href="#">Edit</a>
<input type="checkbox"/>	1901075	HENG WEN XIAN	CS	27	Yes	<a href="#">Edit</a>
<input type="checkbox"/>	1901566	KHOO JIA WEI	CS	15.5	No	<a href="#">Edit</a>

Figure 5.2.1: Results Portal (Lecturer View)

For the students on the other hand, the results portal was designed with simplicity and clarity in mind. Once logged in, students are able to view their results in a secure environment, with the interface displaying their marks in a straightforward layout, as shown in Figure 5.2.2. To maintain control over the release of results, lecturers can set the verified status of each assessment. If the verified status is marked as “Yes,” the result is shown to the student as part of their record. If it has not been verified, the portal will display the status as “Unreleased”, making it clear that the marks are not yet final.

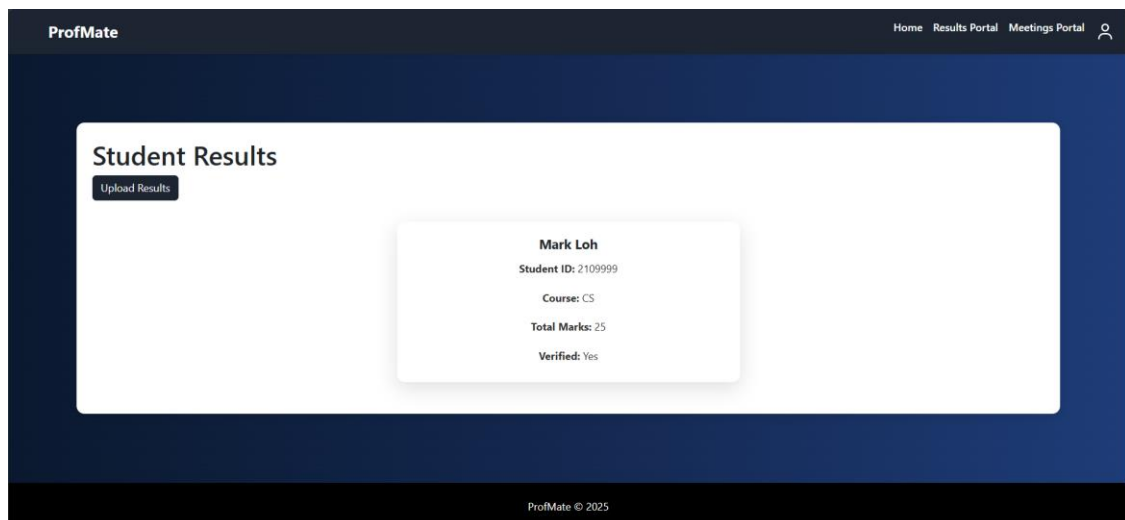


Figure 5.2.2: Results Portal (Student View)

### 5.3 Meeting Scheduler

A meeting scheduler has also been developed for ProfMate to provide an organised and more efficient way for students and lecturers to arrange meetings and academic sessions. This feature removes unnecessary back and forth communication between lecturers and students by acting as a platform for students to book appointments with lecturers. In the meeting scheduler as shown in Figure 5.3.1, lecturers can upload their availability by creating slots that define specific times they are open to meeting with students. These slots are then displayed in the system and made accessible to students, ensuring that the process of arranging meetings is clear and structured. Lecturers are able to edit or delete their slots when needed, and they also retain the ability to manage bookings by confirming or cancelling requests. When cancelling a booking, lecturers may leave remarks to provide students with additional context.

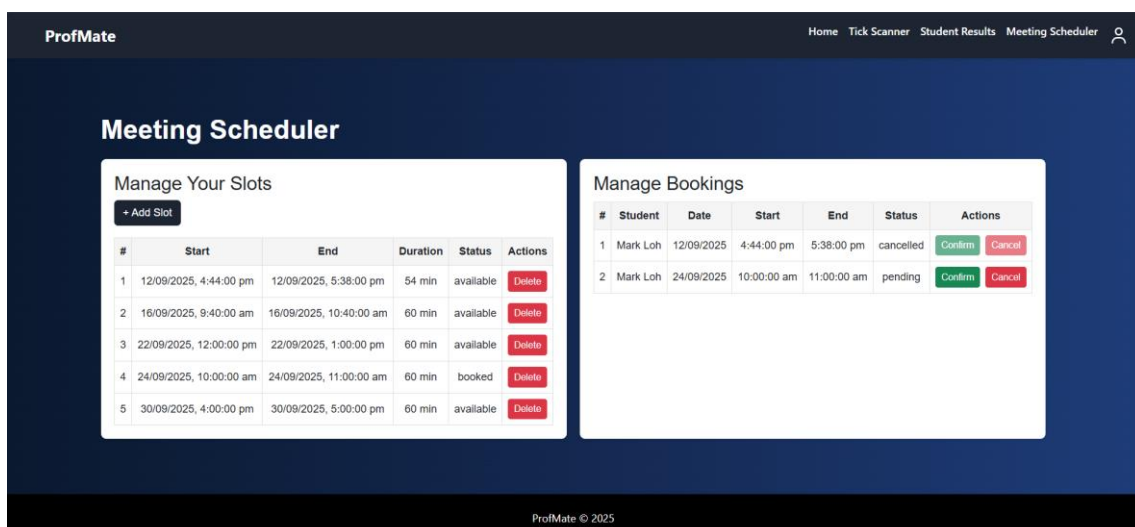


Figure 5.3.1: Meeting Scheduler (Lecturer View)

From the view of students as shown in Figure 5.3.2, the meeting scheduler provides a page where they can select a lecturer and view the time slots available. Slots are presented with details such as start and end time, duration, and status so that students can easily identify a suitable appointment. Once a slot is chosen, students can book it directly, and the status of the slot updates immediately to prevent bookings overlapping. Students are also able to view a list of their confirmed or pending appointments, with information displayed in a clear and organised table. If a booking is cancelled by the lecturer, the system indicates this and shows any remarks provided, giving students a transparent view of their scheduling status.

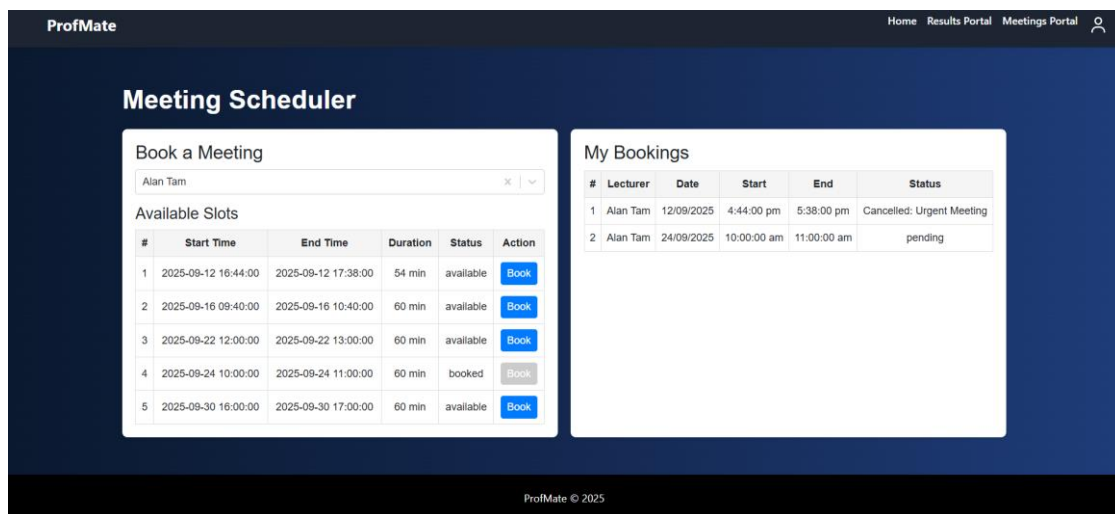


Figure 5.3.2: Meeting Scheduler (Student View)

## 5.4 Login / Register

A login and registration module has also been implemented in ProfMate to provide secure and personalised access for students and lecturers. The system ensures that each account is tied to a specific role, enabling lecturers to manage meetings, grade entries, and results, while students can access grades and book appointments. The registration page, as shown in Figure 5.4.1, captures key information including name, email, and password, and applies additional validation for role-specific details such as faculty for lecturers and student identification with course details for students. This ensures that each account is linked with the correct academic context. The login page, as shown in Figure 5.4.2, verifies credentials against stored records and restricts access to authorised users, while failed attempts are managed with appropriate error handling to maintain system integrity.

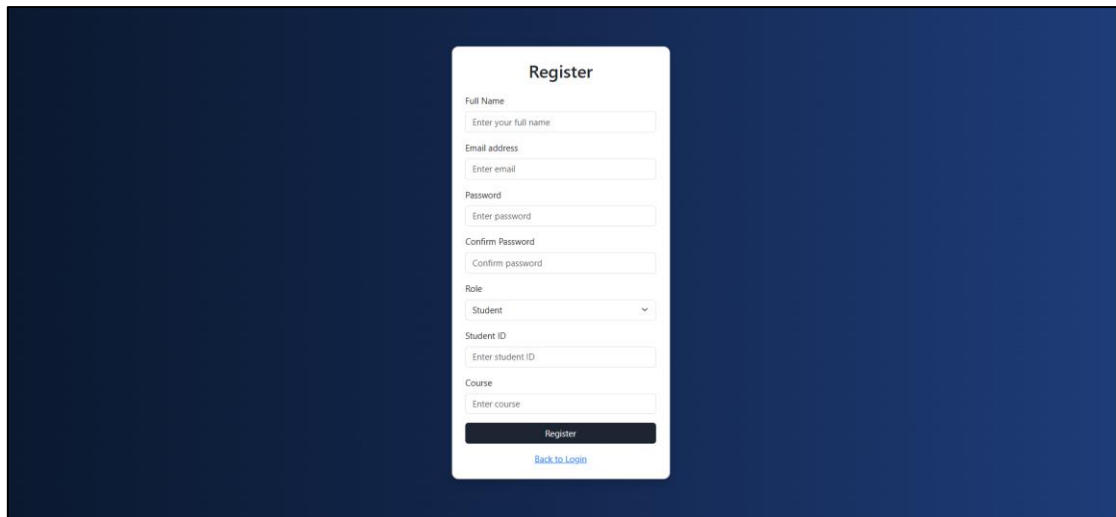
The image shows a 'Register' form on a dark blue background. The form is white and contains the following fields: 'Full Name' with a placeholder 'Enter your full name', 'Email address' with a placeholder 'Enter email', 'Password' with a placeholder 'Enter password', 'Confirm Password' with a placeholder 'Confirm password', 'Role' with a dropdown menu showing 'Student', 'Student ID' with a placeholder 'Enter student ID', and 'Course' with a placeholder 'Enter course'. At the bottom of the form are a black 'Register' button and a blue link 'Back to Login'.

Figure 5.4.1: Register Page

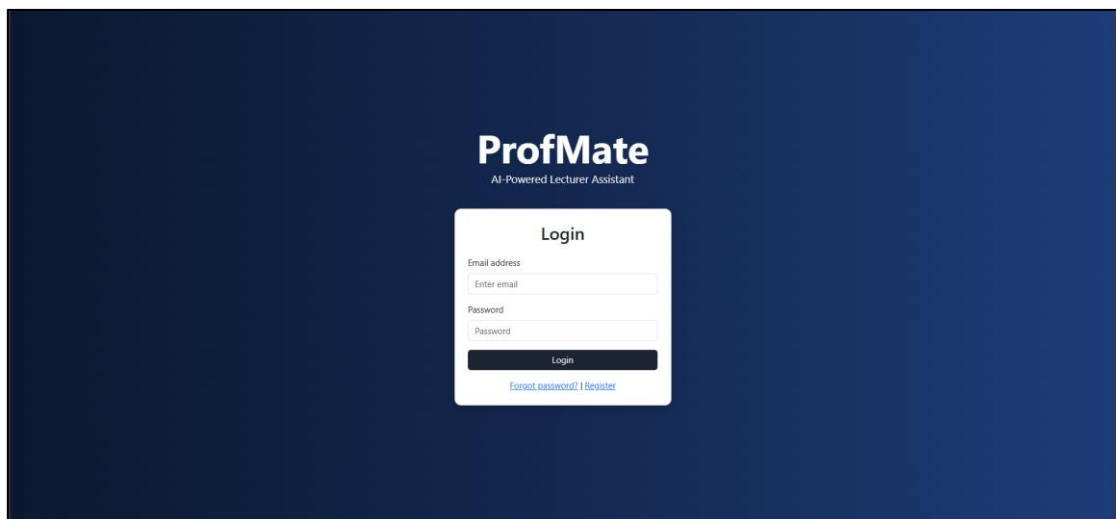
The image shows a 'Login' form on a dark blue background. At the top, the text 'ProfMate' is displayed in a large, bold, white font, with 'AI-Powered Lecturer Assistant' in a smaller font below it. The form is white and contains the following fields: 'Email address' with a placeholder 'Enter email' and 'Password' with a placeholder 'Password'. At the bottom of the form are a black 'Login' button and two blue links: 'Forgot password?' and 'Register'.

Figure 5.4.2: Login Page

## 5.5 Profile Page

A simple profile page was also developed in ProfMate to allow users to view and manage their personal information according to their assigned roles. This page shown in Figure 5.5.1 displays essential details such as name, email, and role, along with faculty information for lecturers and course details for students. The profile page provides a clear and organised interface where users can confirm the accuracy of their records, ensuring that stored information remains consistent and up to date within the system.



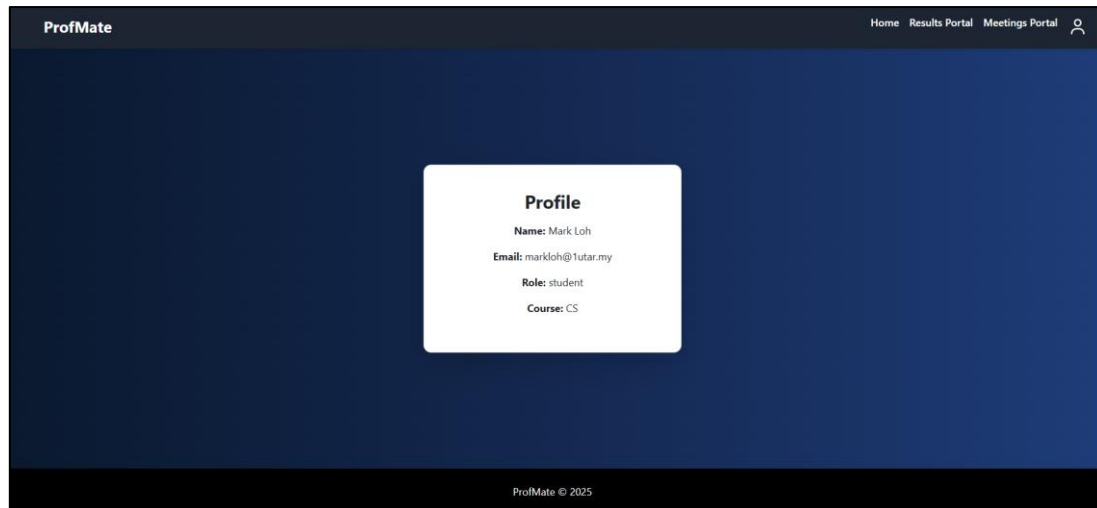


Figure 5.5.1: Profile Page

## CHAPTER 6

# Comparison of Deep Learning Models for Tick Detection

In this chapter, the performance of two object detection models was compared to determine the most effective approach for tick mark detection. The models under study are Faster R-CNN and YOLOv5n. Each model was trained on the same dataset and evaluated using identical metrics to ensure fairness. The chapter begins with an overview of the architecture and training process of the three models, followed by a description of the unified dataset preparation method. The results are then analyzed through both quantitative and qualitative comparisons, concluding with a summary of the overall findings.

### 6.1 Model Architecture and Training

#### 6.1.1 Faster R-CNN

One of the object detection models evaluated in this project is Faster R-CNN, sourced from the `torchvision.models.detection` module provided by PyTorch. This architecture is well-suited for object detection tasks, particularly when both classification and localisation are required. Faster R-CNN operates in two stages, in which the first stage uses a Region Proposal Network (RPN) to generate candidate object regions, while the second stage classifies these regions and refines their bounding box coordinates.

The backbone used for this model is a pretrained ResNet-50 integrated with a Feature Pyramid Network (FPN), which serves as the feature extractor. ResNet-50 is a deep convolutional neural network composed of 50 layers, enhanced by residual connections that allow for better gradient flow during training. The flow of this setup is shown as in Figure 6.1.1.1. The inclusion of FPN enables the model to process features at multiple scales, which is especially beneficial for detecting small objects such as tick marks. The pretrained ResNet-50 with FPN was fine-tuned for this specific task to improve accuracy and generalization.

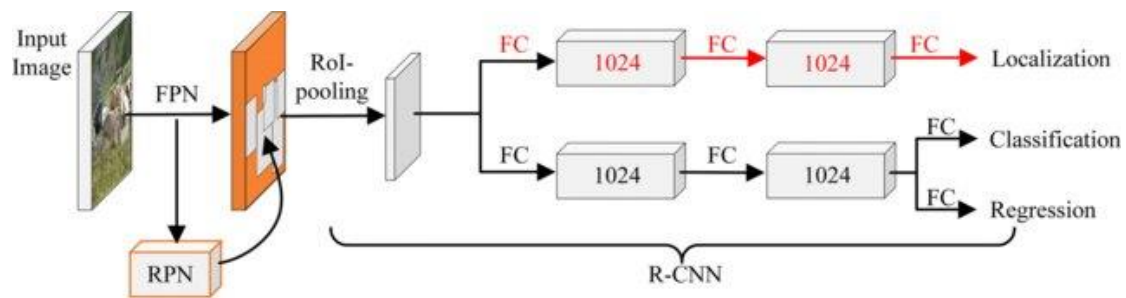


Figure 6.1.1.1: Faster R-CNN model with ResNet-50 and FPN [17]

To tailor the model for the tick detection task, the final classification head was modified to detect two classes: "tick" and background. This was done by replacing the original prediction head with a new FastRCNNPredictor that outputs predictions for two classes. As shown in Figure 6.1.1.2, the model was also configured to use a GPU if available, which accelerates the training and evaluation processes.

```
[ ] import torchvision
from torch.optim import SGD

# Load pre-trained Faster R-CNN model
model = torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=True)

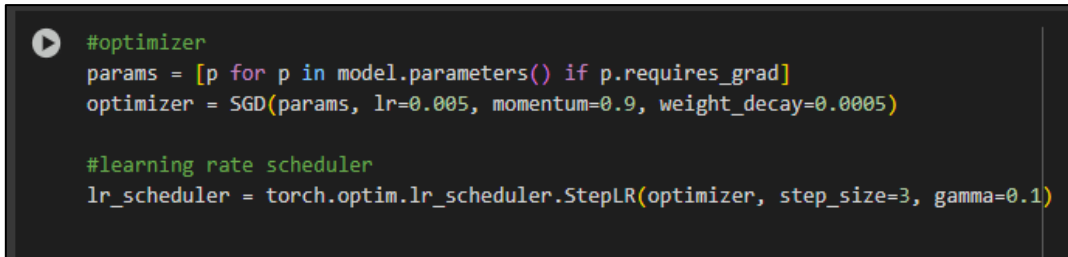
# Modify the classifier for 2 classes (background + tick)
num_classes = 2
in_features = model.roi_heads.box_predictor.cls_score.in_features
model.roi_heads.box_predictor = torchvision.models.detection.faster_rcnn.FastRCNNPredictor(in_features, num_classes)

# Move model to GPU if available
device = torch.device('cuda') if torch.cuda.is_available() else torch.device('cpu')
model.to(device)
```

Figure 6.1.1.2: Model Architecture Definition

This configuration enables the ProfMate system to effectively detect and localize tick marks on scanned pages with high accuracy, forming the basis of its automated assessment review functionality.

For optimisation, Stochastic Gradient Descent (SGD) was chosen for this model, with the settings of 0.005 learning rate, 0.9 momentum, and 0.0005 weight decay as shown in Figure 6.1.1.3. These hyperparameters were set based on standard practices for fine-tuning object detection models, as it provides a good balance between convergence speed and generalistaion. Finally, a learning scheduler was fitted to help reduce the learning rate by 0.1 every 3 epochs. This helped to fine-tune the model parameters to avoid overfitting.



```
#optimizer
params = [p for p in model.parameters() if p.requires_grad]
optimizer = SGD(params, lr=0.005, momentum=0.9, weight_decay=0.0005)

#learning rate scheduler
lr_scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=3, gamma=0.1)
```

Figure 6.1.1.3: Optimizer and Learning Rate Scheduler Setup

The two training strategies were ran using a preprocessed dataset in Google Colab. For the first strategy, the model trained for 30 epochs, with a batch size of 8, and the run completed in approximately 35 minutes. To explore the learning capacity of this model over a long training cycle, the second strategy was performed, which is a free-run mode, where training stopped at 19 epochs due to early stopping patience of 15, with a runtime of approximately 22 minutes.

### 6.1.2 YOLOv5n

The third object detection model evaluated in this project is YOLOv5n, sourced from the open-source Ultralytics YOLOv5 repository. YOLO (You Only Look Once) is a one-stage object detection framework that performs both classification and localization in a single forward pass, making it faster and lighter than two-stage models such as Faster R-CNN. The YOLOv5n variant (“nano”) is the smallest in the YOLOv5 family, specifically designed for real-time inference on resource-constrained environments.

Unlike Faster R-CNN, YOLOv5n does not rely on a Region Proposal Network. Instead, it divides the input image into a grid and predicts bounding boxes and class probabilities directly from the feature maps. The backbone network is a CSPDarknet-based architecture, which balances efficiency and accuracy by using cross-stage partial connections to reduce computation while preserving gradient flow. The model is pretrained on the COCO dataset and was fine-tuned for tick mark detection by configuring the output head to predict a single class, which is “tick”. The flow of the architecture can be illustrated as in Figure 6.1.2.1, which shows the mechanism of YOLOv5.

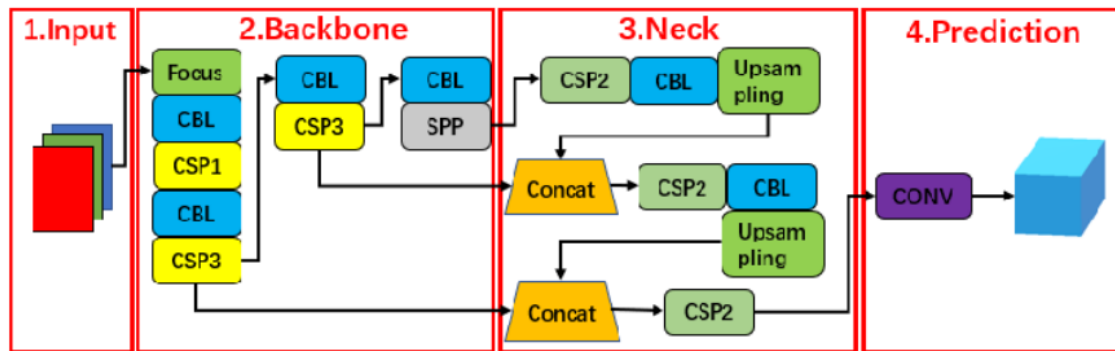


Figure 6.1.2.1: YOLOv5 Network Structure Diagram [18]

To tailor the model for this project, the training dataset was defined in a custom YAML configuration file specifying one object class (“tick”) and paths to the training, validation, and testing images. The model was trained using input images resized to  $384 \times 384$ , a batch size of 8, and GPU acceleration on Google Colab. Two training strategies were applied: one with a fixed 30-epoch schedule, and another with a free-run schedule of up to 300 epochs using early stopping (patience set to 15), allowing the training process to terminate automatically once validation performance plateaued. The training pipeline setup is shown in Figure 6.1.2.2.

```
!python train.py \
  --img 384 \
  --batch 8 \
  --epochs 30 \
  --data tick.yaml \
  --weights yolov5n.pt \
  --name tick_yolov5n_30 \
  --cache

!python train.py \
  --img 384 \
  --batch 8 \
  --epochs 300 \
  --data tick.yaml \
  --weights yolov5n.pt \
  --name tick_yolov5n_freerun \
  --cache \
  --patience 15
```

Figure 6.1.2.2: YOLOv5n Training Pipeline Setup in Google Colab

The optimization process used the default YOLO training setup, which applies stochastic gradient descent with adaptive learning rate scheduling, confidence thresholding, and non-maximum suppression to refine predictions. Data caching was enabled during training to speed up the data loading process. The YOLOv5 framework also incorporates a built-in loss function combining objectness, classification, and bounding box regression losses. Training logs and validation metrics were recorded automatically by the Ultralytics pipeline, and model checkpoints were saved under the experiment names *tick\_yolov5n\_30* and *tick\_yolov5n\_freerun*.

The two training strategies were ran using a preprocessed dataset in Google Colab. For the first strategy, the model trained for 30 epochs, with a batch size of 8, and the run completed in approximately 24 minutes. To explore the learning capacity of this model over a long training cycle, the second strategy was performed, which is a free-run mode, with an early stopping patience of 15 epochs. It reached 76 epochs and it ran for about an hour. In both training sessions, the YOLOv5n framework handled optimization automatically, employing its default configuration of SGD-based optimization, adaptive learning rate scheduling, and built-in loss functions that combine objectness, classification, and bounding box regression losses.

### 6.1.3 Comparison of Architecture and Training Among the Models

Table 6.1.3.1: Comparison of Architecture and Training between Custom CNN, Faster R-CNN and YOLOv5n

Model	Faster R-CNN	YOLOv5n
Type	Two-stage object detector	Single-stage object detector
Annotation Format	Pascal VOC XML	YOLO TXT
Loss Function	Multi-task loss (classification + bounding box regression)	Combination of objectness loss, classification loss and bounding box regression
Optimizer & LR Scheduling	SGD (lr=0.005, momentum=0.9, weight decay=0.0005) + StepLR ( $\times 0.1$ every 3 epochs)	SGD (lr=0.01, grouped weight decay, auto LR scheduling, early stopping=15)
Runtime Duration	30 epochs: 35m, free run (19 epochs): 22m	30 epochs: 24m, free run (76 epochs): 1h 1m

## 6.2 Dataset Preparation

The dataset used in the training of the tick detection models in this project underwent a preparation process that transformed the raw, unprocessed marked exam papers to fully annotated images ready for deep learning purposes. A total of 264 images were obtained by scanning 70 sets of marked exam papers on a printer. Each of the images were then cropped and resized to eliminate unnecessary margins and noise from the images.

Next, an open-source graphical image annotation tool, LabelImg, was used to annotate all 264 scanned images by drawing bounding boxes on every tick mark on every page, as shown in Figure 6.2.1. 264 annotation files in the format of Pascal VOC XML format were produced, containing coordinates of the position of every tick mark on its respective image. For compatibility with the different model architectures, the Pascal VOC XML annotations were further converted into other formats. For YOLOv5n, the annotations were converted into YOLO text format, where each line corresponds to a tick mark with normalized coordinates (class, x\_center, y\_center, width, height). For the custom CNN, the annotations were converted into grid-based label tensors of size (S, S, 5), containing objectness, bounding box offsets, and dimensions. These conversions ensured that the dataset could be used seamlessly across all three detection models while maintaining consistency in tick mark labels.

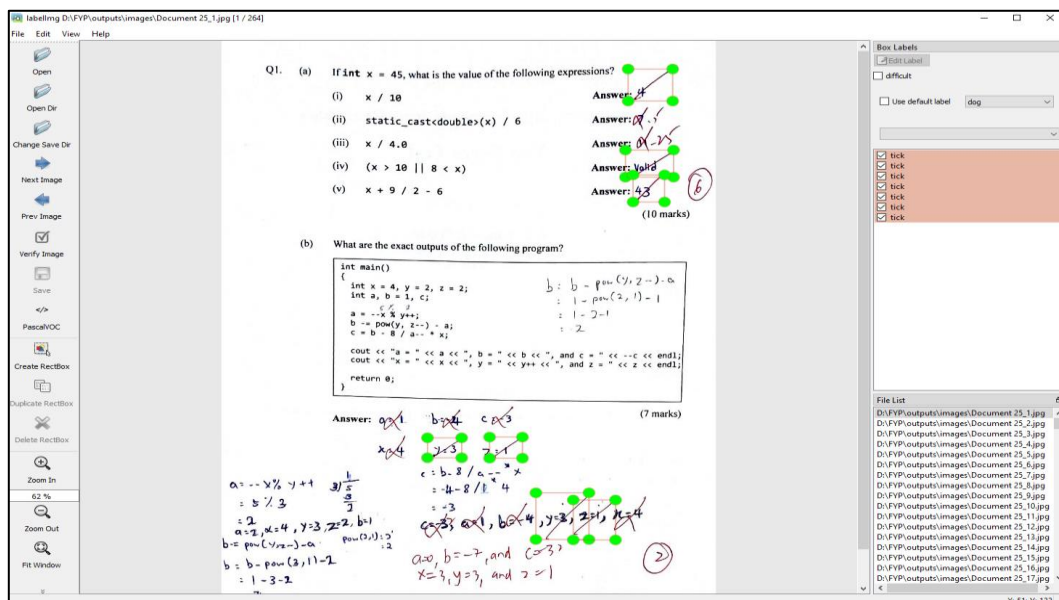


Figure 6.2.1: Annotating images to create respective Pascal VOC XML files

The images, along with the respective annotations, were uploaded to Google Drive to be accessed from Google Colab to perform further preprocessing. As shown in Figure 6.2.2, the

dataset was split into training, validation and testing folders, with a split of 80/10/10 on training, validation and testing sets respectively.

```

import shutil
from sklearn.model_selection import train_test_split

# Set base dataset directory
base_dir = "drive/MyDrive/fyp/new_frcnn/dataset"
images_dir = os.path.join(base_dir, "images")
annotations_dir = os.path.join(base_dir, "annotations")

# List all image files (assumes .jpg or .png)
image_files = [f for f in os.listdir(images_dir) if f.endswith(('.jpg', '.png'))]

# Split into train/val/test
train_files, testval_files = train_test_split(image_files, test_size=0.2, random_state=42)
val_files, test_files = train_test_split(testval_files, test_size=0.5, random_state=42)

# Target split folders
splits = {
    "train": train_files,
    "val": val_files,
    "test": test_files
}

# Create output directories and move files
for split, files in splits.items():
    split_img_dir = os.path.join(base_dir, split, "images")
    split_ann_dir = os.path.join(base_dir, split, "annotations")
    os.makedirs(split_img_dir, exist_ok=True)
    os.makedirs(split_ann_dir, exist_ok=True)

    for file in files:
        shutil.copy(os.path.join(images_dir, file), os.path.join(split_img_dir, file))
        xml_name = file.replace('.jpg', '.xml').replace('.png', '.xml')
        xml_src = os.path.join(annotations_dir, xml_name)
        if os.path.exists(xml_src):
            shutil.copy(xml_src, os.path.join(split_ann_dir, xml_name))
        else:
            print(f"Warning: Annotation not found for {file}")

```

Figure 6.2.2: Dataset Splitter

A custom dataset class “TickDataset” loads images and matching Pascal VOC XML annotations by parsing XML files to extract bounding box coordinates for objects labeled “tick”, and each image with the corresponding target dictionary. The dataset class is also resize-aware, meaning it resizes each image to a fixed size (384x384), while scaling bounding box coordinates accordingly to match the resized image to ensure the correctness in model training. The purpose of resizing the images is to ensure a consistent input for the model and reduce computational overhead, which is important for a shared resource environment like Google Colab. Furthermore, data augmentation was also performed to improve generalization for the model. The techniques applied were random horizontal flipping, and color jittering, to simulate real-world variability. Finally, each image was converted from a Python Imaging Library (PIL) format to a PyTorch tensor using “ToTensor()” to normalise pixel values to a [0,1] range, which keeps input values in a predictable range for the model.



## 6.3 Model Evaluation

### 6.3.1 Faster R-CNN

The loss graphs of the 30 epoch run and the free run, as shown in Figure 6.3.1.1 and Figure 6.3.1.2 respectively, showed a drastic decline in the early epochs. For the 30 epoch training, loss started high at around 0.62, with a steep decrease during the initial epochs. The rapid descent indicated quick learning and made significant improvements in predictions. After the drop, the loss curve flattened out, becoming much more stable from Epoch #6 onwards. The loss value hovered around 0.27 and 0.28, with only minor fluctuations.

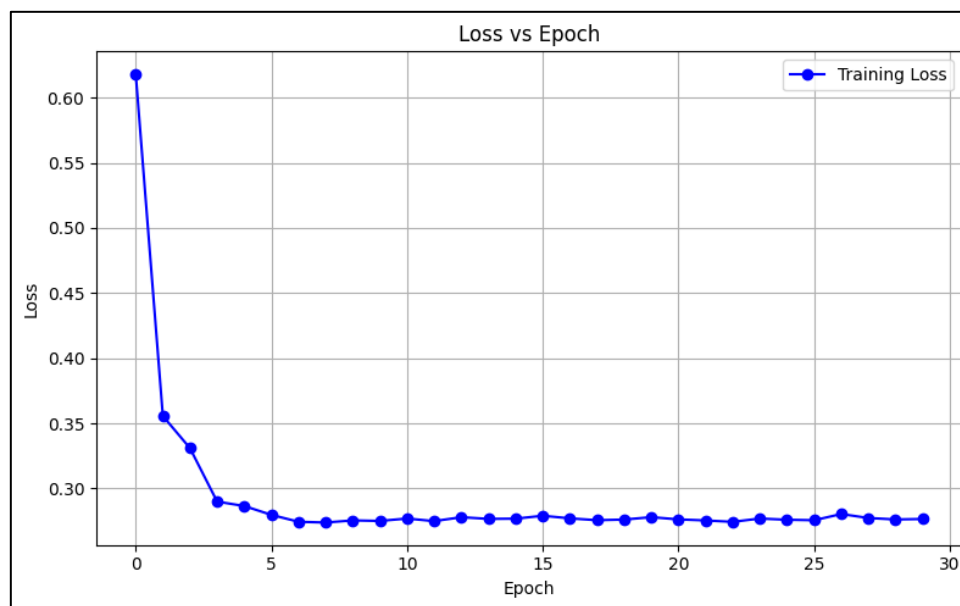


Figure 6.3.1.1: Loss Curve for 30 Epoch Run

In Figure 6.3.1.2, which depicts the loss curve for the free run that reached 19 epochs, a similar initial behaviour was observed. The loss began at a relatively high value and rapidly decreased during the first few epochs, indicating that the model was able to learn key patterns quickly as well. Unlike the 30 epoch run, the loss had a slightly lower level of stability after 10 epochs, with a more noticeable fluctuation in loss values. It seemed that the shorter training period allowed less time for the model to fully converge and stabilise at a lower loss value.

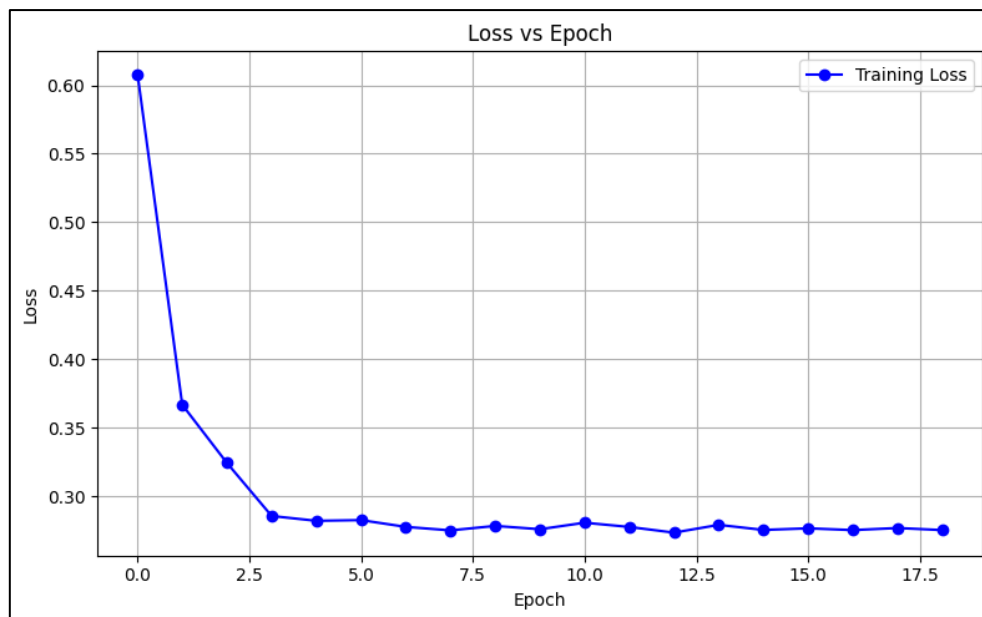


Figure 6.3.1.2: Loss Curve for Free-Run (19 epochs)

For the evaluation of the final model, the model was evaluated using the test set and the metrics were accuracy, precision, recall and F1-score. For the trained Faster R-CNN model, it achieved an accuracy of 77.9%, a precision of 79.8%, a recall of 97.1% and the F1-score of 87.6%. These figures indicated that the model not only correctly identified the majority of tick marks in the test set but also maintained a good level of correctness among the predictions made. This performance is further illustrated by the confusion matrix shown in Figure 6.3.1.3, where the distribution of true positives, false positives and false negatives were presented.

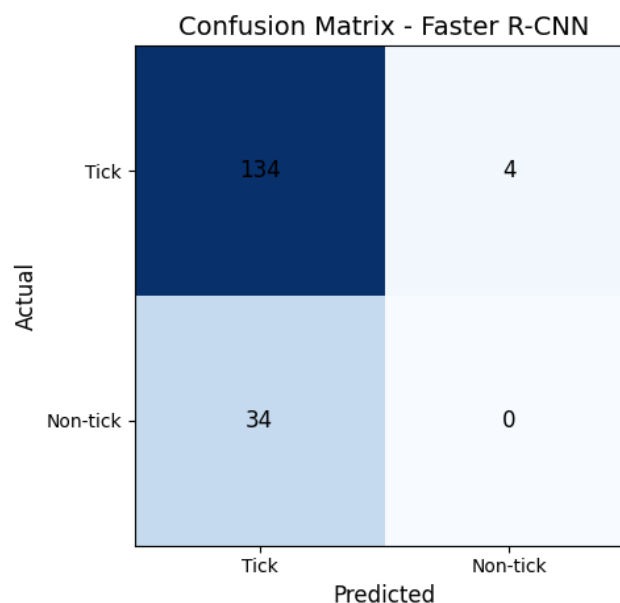


Figure 6.3.1.3: Confusion Matrix of Faster R-CNN

A precision-recall curve was also plotted, as shown in Figure 6.3.1.4. The model maintained a very high precision across a broad range of recall values. This indicates that the model is highly accurate when making positive predictions, but the precision starts to decline when recall is closing in towards 100%, which means that the model started to create false positives when trying to capture all the possible positive instances.

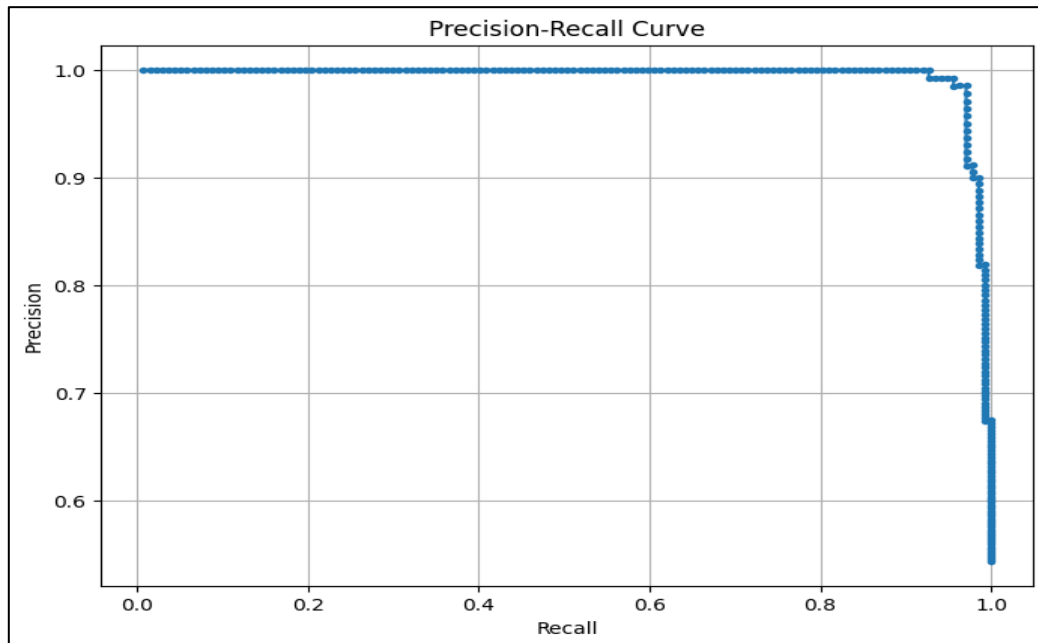


Figure 6.3.1.4: Precision-Recall Curve (Faster R-CNN)

### 6.3.2 YOLOv5n

For the loss graphs of YOLOv5n, which includes a 30-epoch run and a free run that reached 76 epochs, are as shown in Figure 6.3.2.1 and Figure 6.3.2.2. For the 30-epoch run, the graph of box loss showed a steep and consistent downward trend in both training and validation, indicating that the model is continuously and smoothly improving its ability to perform that task without overfitting. For the objectness loss however, the graph displayed a much more volatile trend for both training and validation losses. While there is an overall downward slope, the curves are characterized by significant, rapid fluctuations. This suggests instability in the learning process, with the model's performance on this specific task jumping up and down rather than smoothly converging.

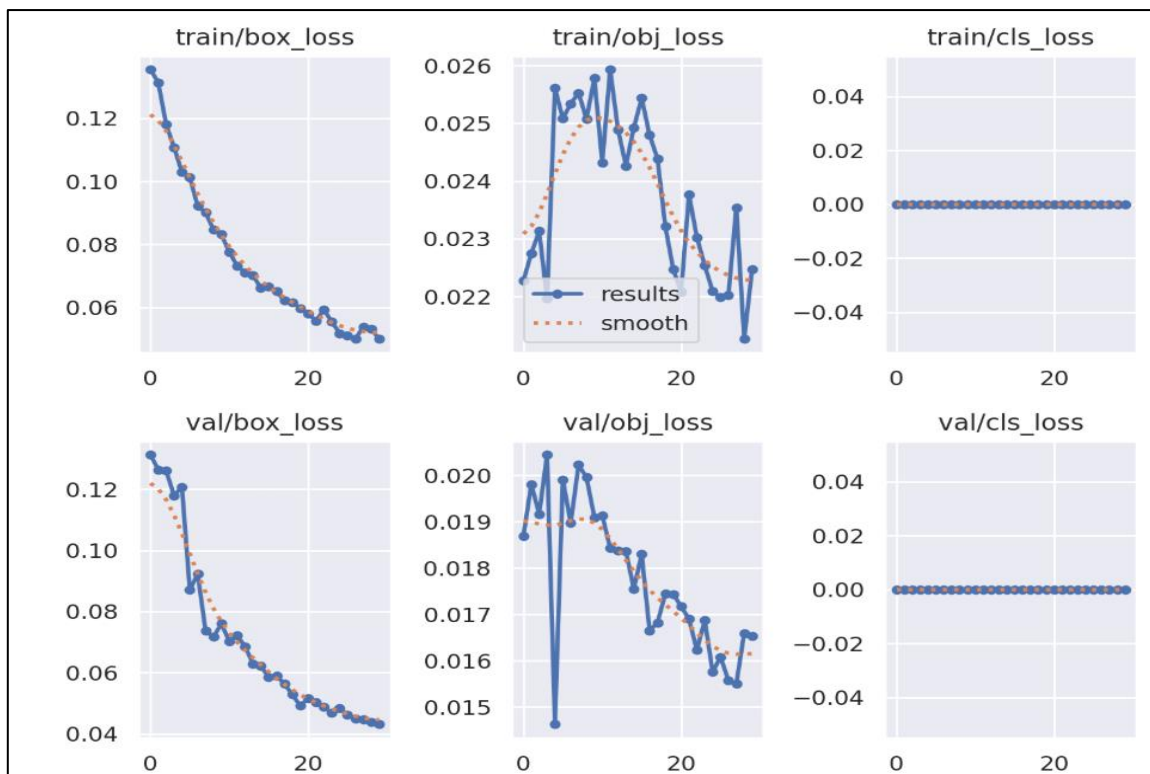


Figure 6.3.2.1: Loss Graphs for 30 Epoch Run

For the free-run that reached 76 epochs, as shown in Figure 6.3.2.2, the training and validation box loss curves show an excellent trend. It started at a high value and underwent a steep, smooth decline for the first 20-30 epochs. After this, the values continued to decrease at a slower rate, indicating a steady convergence. The validation loss also closely tracked the training loss, indicating the model did not overfit and generalized well to unseen data. However, like in the 30-epoch run, the objectness loss curves for both training and validation were more complex and showed significant fluctuations. The training curve showed an initial steep drop, followed by a period of high volatility before stabilizing and trending downwards more consistently after around 20 epochs.

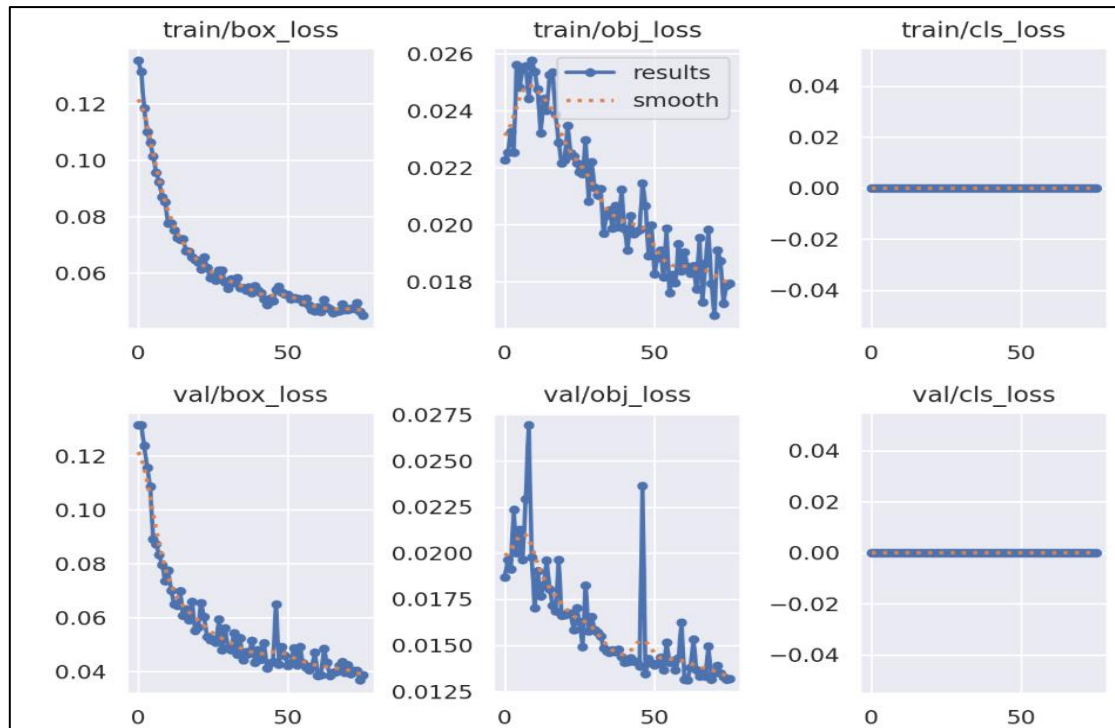


Figure 6.3.2.2: Loss Graphs for Free Run (76 Epochs)

For the evaluation of the final YOLOv5n model, the model was also assessed using the same test set with the metrics of accuracy, precision, recall, and F1-score. The trained YOLOv5n model achieved an accuracy of 84.5%, a precision of 85.5%, a recall of 98.6%, and an F1-score of 91.6%. These results show that the YOLOv5n model not only detected nearly all tick marks present in the test set but also produced a higher proportion of correct predictions among all detections. Compared to the Faster R-CNN model, YOLOv5n performed slightly better across all four metrics, demonstrating its stronger overall detection capability. This performance is further illustrated by the confusion matrix shown in Figure 6.3.2.3, which presents the distribution of true positives, false positives, and false negatives.

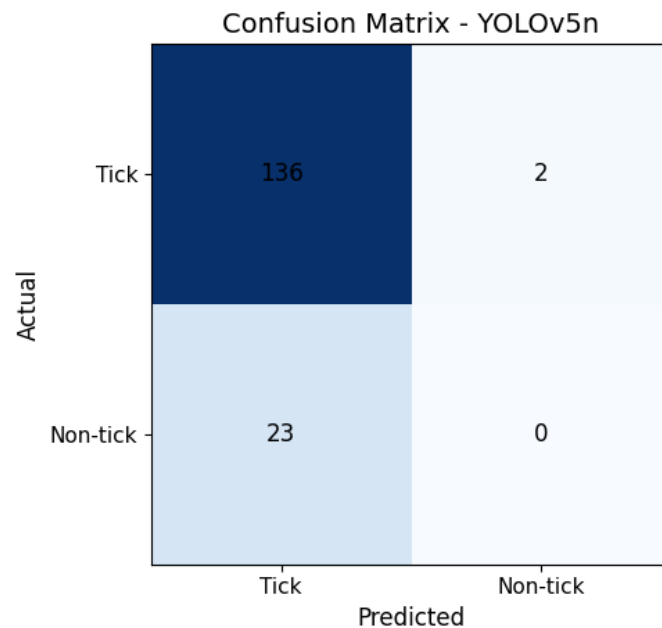


Figure 6.3.2.3: Confusion Matrix of YOLOv5n

A precision–recall curve was also plotted for the YOLOv5n model, as shown in Figure 6.3.1.4. Similar to the Faster R-CNN model, YOLOv5n maintained a very high precision across a broad range of recall values. However, compared to Faster R-CNN, its precision remained slightly higher even as recall approached 100%, indicating that YOLOv5n was able to capture almost all positive instances while generating fewer false positives. This demonstrates that YOLOv5n not only retains the strong predictive ability of Faster R-CNN but also achieves a marginal improvement in balancing precision and recall at the highest recall levels.

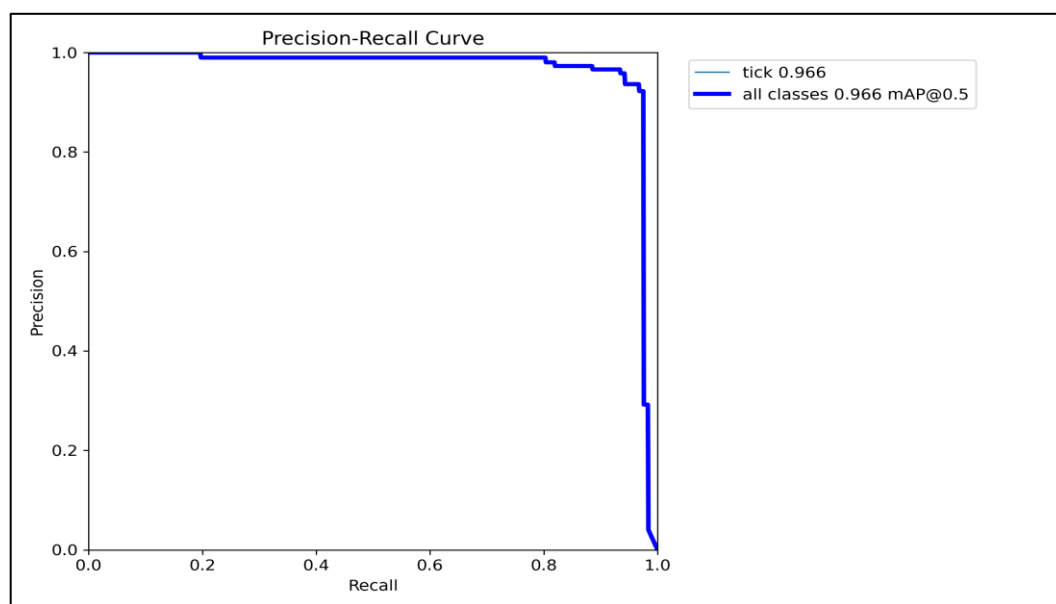


Figure 6.3.1.4: Precision-Recall Curve (YOLOv5n)

## 6.4 Comparison of Metrics and Visualisation

To provide a clear overview of how both models performed, the key evaluation metrics for Faster R-CNN and YOLOv5n were compared side by side. Table 6.4.1 summarizes the accuracy, precision, recall, and F1-score achieved by each model on the test set. This comparison highlights the relative strengths of each approach and shows how YOLOv5n slightly outperforms Faster R-CNN across most metrics.

Table 6.4.1: Comparison of Performance Metrics for Faster R-CNN and YOLOv5n

Model	Accuracy	Precision	Recall	F1-Score
<b>Faster R-CNN</b>	0.779	0.798	0.971	0.876
<b>YOLOv5n</b>	0.845	0.855	0.986	0.916

Alongside the quantitative evaluation, qualitative visualisations of the detection results were also produced. These images, which display bounding boxes over the detected tick marks, allow for a clearer understanding of how well each model identifies and localises the targets in real test images. By observing these outputs, the effectiveness of the models can be visually confirmed beyond numerical metrics alone.

In Figure 6.4.1, both models were applied to the same paper to detect and count tick marks. The image on the left shows the output from Faster R-CNN, while the image on the right shows the output from YOLOv5n. In this case, Faster R-CNN detected all tick marks but also produced one false positive, whereas YOLOv5n correctly identified all tick marks without any false positives. Figures 6.4.2 and 6.4.3 similarly present side-by-side comparisons for other papers, allowing a clear visual assessment of how YOLOv5n generally produces cleaner, more accurate detections compared with Faster R-CNN.

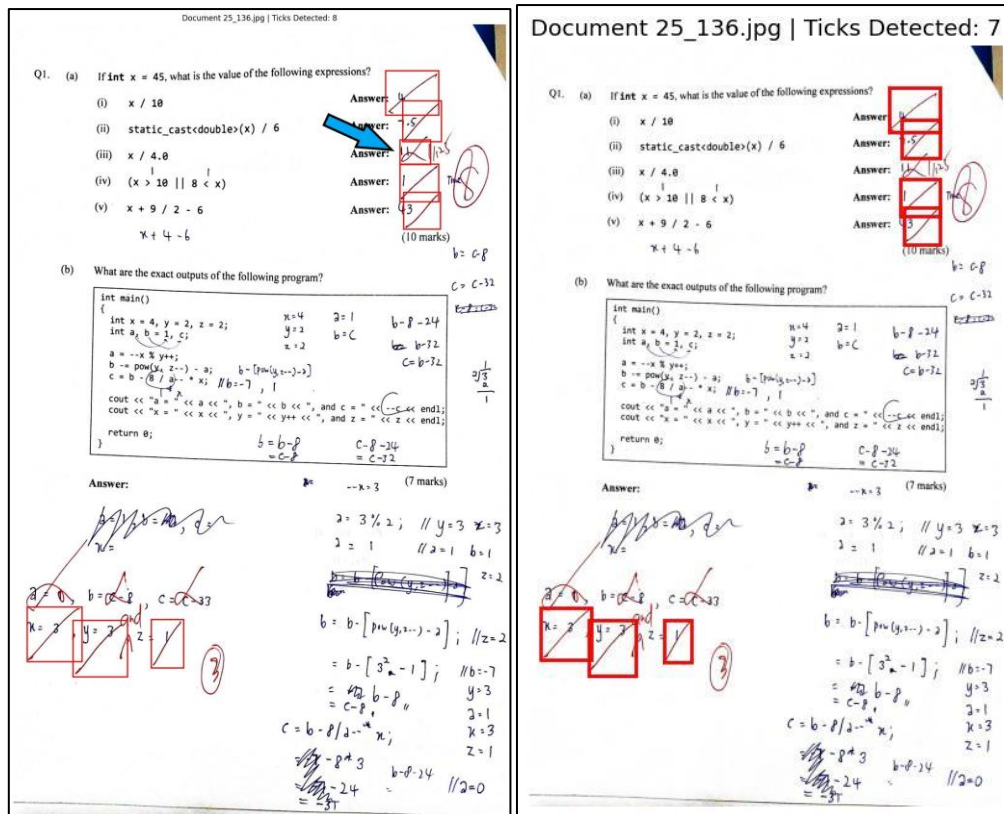


Figure 6.4.1: Visualised Detections of Faster R-CNN (left) and YOLOv5n (right)

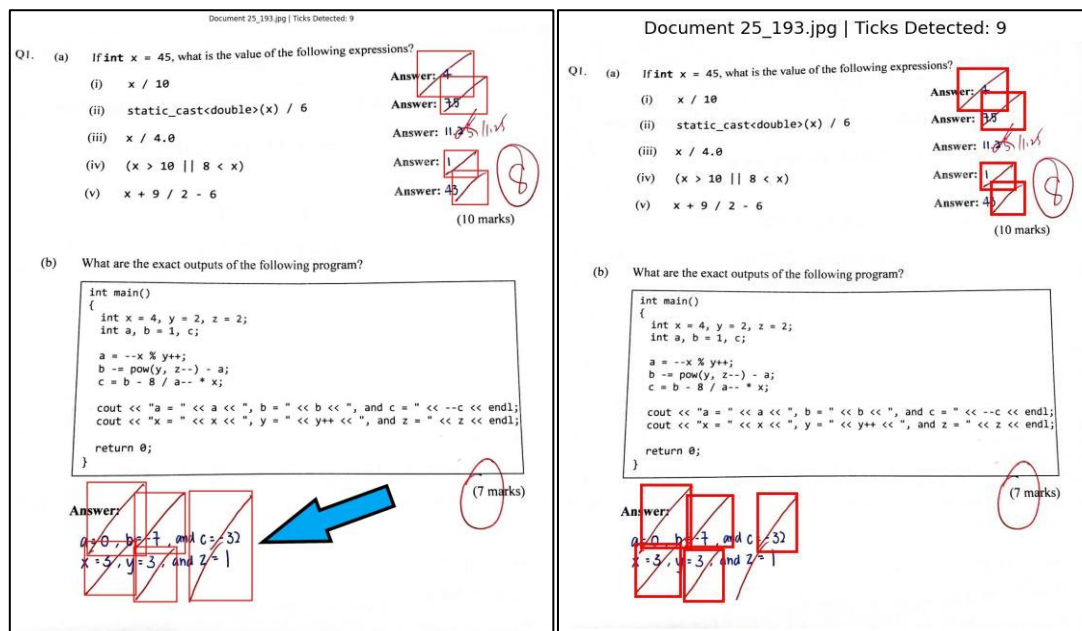


Figure 6.4.2: Visualised Detections of Faster R-CNN (left) and YOLOv5n (right)



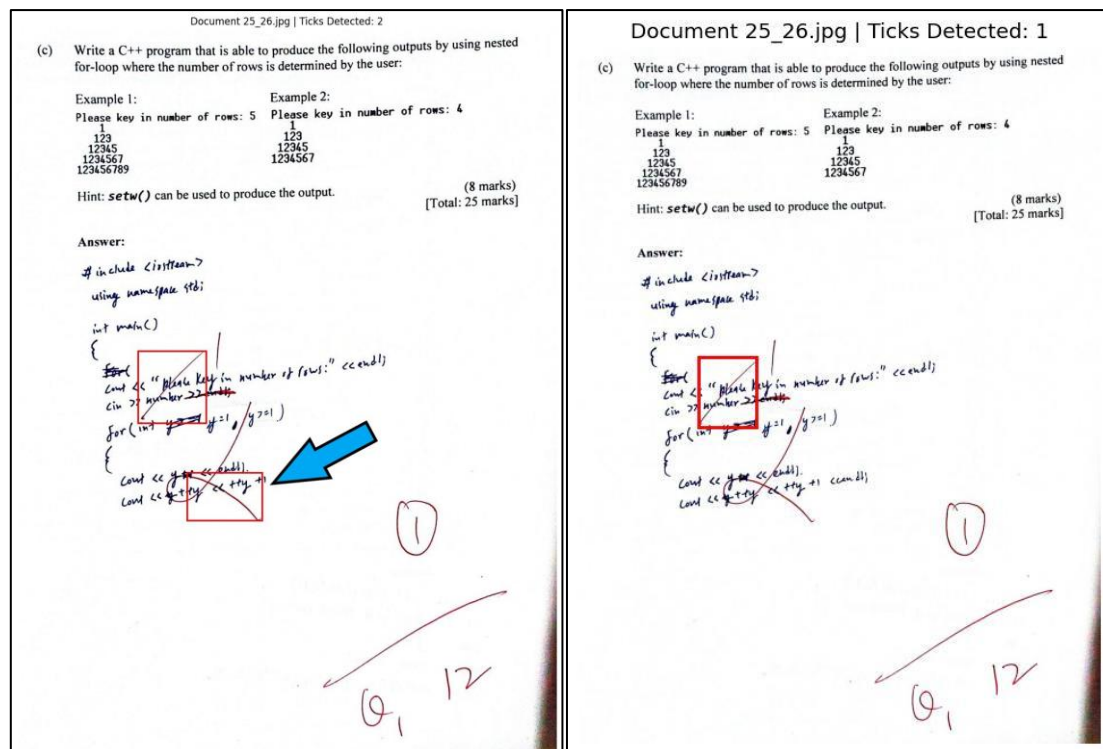


Figure 6.4.3: Visualised Detections of Faster R-CNN (left) and YOLOv5n (right)

## 6.5 Summary

The results from both the performance metrics and the visualisations demonstrate that Faster R-CNN and YOLOv5n achieved broadly comparable performance on the tick mark detection task. Both models consistently identified the majority of tick marks in the test set images, with high precision and recall scores. However, YOLOv5n showed a slight upper edge, producing fewer false positives and slightly higher overall precision, recall and F1-score compared with Faster R-CNN. In addition to this, the lighter architecture and single-stage detection pipeline of YOLOv5n makes it more suitable for real-time deployment. For this, YOLOv5n was ultimately chosen as the final model for the tick scanner module of ProfMate.

## CHAPTER 7

### Conclusion and Recommendation

Academic environments continue to challenge lecturers with administrative workloads that limit the time they can dedicate to teaching and student engagement. Building upon the groundwork established in Final Year Project 1, ProfMate has now evolved into a fully functioning platform that automates key administrative tasks such as grading, meeting scheduling, and result management. By integrating the AI-powered tick detection model with a responsive React front-end and a scalable Node.js back-end, the system delivers an efficient and user-friendly solution for lecturers and students alike.

In Final Year Project 2, the main objectives of model optimisation, system integration, and feature enhancement were successfully achieved. The tick detection model was further refined, resulting in improved accuracy, precision, recall, and F1-score. Not only that, but the model was also deployed as a real time tick detector which allowed live scanning for reviewing marked papers. These improvements translate into more reliable automated grading and verification. Additionally, the meeting scheduling platform and results portal have been expanded and integrated with secure authentication and role-based access controls, creating a seamless user experience.

The second phase of development also focused on enhancing system usability and performance. The user interface of ProfMate was refined to improve accessibility, while the underlying architecture was optimised for scalability and reliability. Together, these advancements have produced a robust platform that addresses the original project goals more comprehensively than in FYP1.

Looking ahead, future work on ProfMate can address the key challenges identified during development. One of improvements to be made is to handle variable mark weights per tick by uploading marking schemes or OCR-based analysis of the exam papers to ensure a more complete score calculation. Improvements are also needed in detecting overlapping ticks, where refined post-processing, instance segmentation, or temporal stability checks could reduce false positives. Finally, expanding and diversifying the dataset would strengthen generalisation and enable fairer comparisons with models such as Faster R-CNN and YOLOv5n. Through these future improvements, ProfMate will continue evolving and eventually become a more comprehensive and complete AI-powered lecturer assistant web application.

## REFERENCES

### REFERENCES

- [1] Y. Amit, P. Felzenszwalb, and R. Girshick, Object Detection, [https://www.researchgate.net/publication/300469375\\_Object\\_Detection](https://www.researchgate.net/publication/300469375_Object_Detection) (accessed Sep. 10, 2024).
- [2] C. H. Yan and K. T. Yee, "General flow of object detection," in The development of skin lesion detection application in smart handheld devices using deep neural networks, ResearchGate. Available: [https://www.researchgate.net/figure/General-flow-of-object-detection-The-process-of-object-detection-typically-begins-with\\_fig2\\_351532382](https://www.researchgate.net/figure/General-flow-of-object-detection-The-process-of-object-detection-typically-begins-with_fig2_351532382). (accessed Sept. 11, 2024)
- [3] A. B. Amjoud and M. Amrouch, "Object detection using deep learning, CNNs and Vision Transformers: A Review," *IEEE Access*, vol. 11, pp. 35479–35516, 2023. doi:10.1109/access.2023.3266093
- [4] Z.-Q. Zhao, "Object detection with deep learning: A Review," ar5iv, <https://ar5iv.labs.arxiv.org/html/1807.05511> (accessed Sep. 11, 2024).
- [5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," Microsoft Research. Available: <https://arxiv.org/pdf/1512.03385v1>. (accessed Sept.11, 2024)
- [6] B. Liu, W. Zhao, and Q. Sun, "Study of object detection based on faster R-CNN," 2017 Chinese Automation Congress (CAC), Oct. 2017. doi:10.1109/cac.2017.8243900
- [7] I. Oztel, "Human detection system using different depths of the resnet-50 in faster R-CNN," *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, pp. 1–5, Oct. 2020. doi:10.1109/ismsit50672.2020.9255109
- [8] M. I. Roslan *et al.*, "Fruit Detection and recognition using faster R-CNN with FPN30 pre-trained Network," *2023 IEEE 8th International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*, pp. 1–6, Dec. 2023. doi:10.1109/icraie59459.2023.10468169
- [9] S. Wang, "Research towards yolo-series algorithms: Comparison and analysis of object detection models for real-time UAV applications," *Journal of Physics: Conference Series*, vol. 1948, no. 1, Jun. 2021. doi:10.1088/1742-6596/1948/1/012021
- [10] J. Zhou, T. Su, K. Li, and J. Dai, "Small target-yolov5: Enhancing the algorithm for small object detection in drone aerial imagery based on Yolov5," *Sensors*, vol. 24, no. 1, Dec. 2023. doi:10.3390/s24010134
- [11] S. Ruan, C. Zhan, B. Liu, Q. Wan, and K. Song, "A high precision YOLO model for surface defect detection based on PyConv and CISBA," *Scientific Reports*, vol. 15, no. 1, May 2025. doi:10.1038/s41598-025-91930-z
- [12] "Free, ai-powered teacher assistant by Khan Academy", Khanmigo, <https://www.khanmigo.ai/teachers> (accessed Sep. 4, 2024).
- [13] Khan Academy. *Khanmigo for Teachers*. (May 22, 2024). Accessed: Sep. 4, 2024. [Online Video]. Available: [https://youtu.be/TuoWM2vc\\_a0](https://youtu.be/TuoWM2vc_a0)
- [14] "Teachmateai", TeachMateAI, <https://teachmateai.com/> (accessed Sep. 4, 2024)
- [15] "Marking.ai | AI for teachers", Marking.ai. <https://app.marking.ai/> (accessed Sep. 10, 2024)

## REFERENCES

- [16] T. Kissflow, “*Rapid application development (RAD): Definition, Steps & Full Guide*,” Kissflow, <https://kissflow.com/application-development/rad/rapid-application-development/> (accessed Sep. 11, 2024).
- [17] X. Wang, G. Wei, S. Chen, and J. Liu, “An efficient weakly semi-supervised method for Object Automated Annotation,” *Multimedia Tools and Applications*, vol. 83, no. 3, pp. 9417–9440, Jun. 2023. doi:10.1007/s11042-023-15305-0
- [18] D. Yang, C. Su, H. Wu, X. Xu, and X. Zhao, “Shelter identification for shelter-transporting AGV based on improved target detection model yolov5,” *IEEE Access*, vol. 10, 2022. doi:10.1109/access.2022.3220665

## APPENDIX

## POSTER

# PROFMATE

## AI-POWERED LECTURER ASSISTANT

### Introduction

Lecturers often face administrative inefficiencies when handling student grading, meeting scheduling, and review management. Manual processes are time-consuming, prone to errors, and detract from academic focus. There is a need for an integrated solution that automates repetitive tasks and improves the communication workflow between lecturers and students.

### Methodologies

- **Dataset Preparation:** Collected answer sheets and annotated tick marks using Pascal VOC format
- **Model Training:** Faster R-CNN with ResNet backbone
- **System Development:** ASP .NET Core MVC (C#)
- **Integration:** Integrated tick detection model using FastAPI

### Key Features

- **Automated Tick Detection:** Detects and counts tick marks on scanned answer sheets
- **Meeting Scheduler:** Book consultation sessions via shared calendar
- **Results Portal:** View automatically uploaded grades
- **User Authentication:** Different account roles for access

### Conclusion & Future Work

ProfMate shows how AI can ease educators' workload by automating tasks like grade entry and meeting scheduling. Future improvements may include automated grading for structured questions and performance analytics to further support teaching efficiency.

Student: Loh Zi Hin  
Supervisor: Dr Tan Joi San  
Faculty of information & Communication Technology