Data Visualization for Text-based Documents

By

Yii Kuo Chong

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR INFORMATION SYSTEMS (HONOURS)
INFORMATION SYSTEMS ENGINEERING
Faculty of Information and Communication Technology
(Kampar Campus)

May 2023

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: Data Visualization for Text-based Documents

Academic Session: 202305

I YII KUO CHONG

(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in

Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,

[illegible]	[illegible]
(Author's signature)	(Supervisor's signature)
Address:	
1244, Jalan Seksyen 1/3,	
31900, Kampar,	Ooi Boon Yaik
Perak	Supervisor's name
Date: 14 Sep 2023	**Date**: 15/9/2023

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

UNIVERSITI TUNKU ABDUL RAHMAN

Date: 14th September 2023

SUBMISSION OF FINAL YEAR PROJECT

It is hereby certified that **Yii Kuo Chong** (ID No: **21ACB02670**) has completed this final year project/ dissertation/ thesis* entitled "Data Visualization for Text-based Documents" under the supervision of **Ooi Boon Yaik** (Supervisor) from the Department of Computer Science, Faculty of Information Systems and Communication Technology, and **Mohammad Dalvi Esfahani** (Moderator) from the Department of **Information Systems**, Faculty of **Information and Communication Technology**.

I understand that the University will upload a softcopy of my final year project in PDF format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,

Chong

(Yii Kuo Chong)

DECLARATION OF ORIGINALITY

I declare that this report entitled “**Data Visualization for Text-based Documents**” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature	:	[illegible]
Name	:	Yii Kuo Chong
Date	:	14th September 2023

ACKNOWLEDGEMENTS

I would like to thank the following people, whose guidance and help provided me with invaluable insights and inspirations while developing this project.

- Dr. Ooi Boon Yaik <ooiby@utar.edu.my>, my final year project supervisor, for the original project title/idea, and guiding me throughout my final year project. His valuable, constructive suggestions and critiques have been extremely valuable to ensuring the project is successful.
- Aaron Imming <aaim@protonmail.com>, for his help answering my NodeJS/React related question when I was developing the project. His advice, and help has allowed me to complete my work faster and produce better code.

I would also like to thank my university, University Tunku Abdul Rahman (UTAR), without which I would not have the opportunity to participate in academic research and development work.

ABSTRACT

This paper examines the availability, and needs of simple, easy to use, and easy to install data visualization and debugging software. Recently, the prominent rise of the Internet of Things, and Artificial Intelligence has caused an increase of interest and research projects in big data. Along with this, there is an increased demand for simple and easy to use visualization tools to assist in research. Among these sources of demand is Dr. Ooi Boon Yaik, a researcher and lecturer at University Tunku Abdul Rahman who is running quite a few research projects that need these simple to use tools. Incorporating various web technologies, and software tools, this project aims to produce a tool that fills a niche that has not yet been done but greatly needed by researchers. Which is, a data visualisation and debugging tool that is lightweight, easy to use, simple and customizable.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

JS	Javascript
NPM	Node Package Manager

CHAPTER 1: Introduction

1.0 Introduction

In the past semester, I've been tasked by Dr. Ooi Boon Yaik to build a system that processes and visualizes data extracted from data files. The system also doubles as a simple debugging tool to help with over the air debugging.

1.1 Problem Statement and Motivation

There are a few problems with the current setup.

- The lack of easy to install, easy to use, and simple tools means that Dr. Ooi Boon Yaik must spend a lot of precious time setting up the systems for other people.
- There are significant costs incurred since the current setup relies on the internet, and cloud servers.
- The reliance on the internet does not allow for fully local deployments.

The new system should reduce cost (money, time), and improve the ease of use for end users.

1.2 Project Scope

The project will deliver a simple, easy to use, and easy to deploy, data visualization and debugging tool, that can be accessed over the network (local and internet).

1.3 Project Objectives

The project's goal is to create a system that fulfills the following objectives.

- Easy to install.
- Minimal and clear user interface.
- Visualize data from file-based documents.
- Able to visualize data in real time.
- Be able to load the app on web browsers over the network (local and internet).

1.4 Contributions

By providing this visualization tool, we will be able to tremendously speed up research work by saving time and costs through improvements of various aspects of the research process.

1.5 Report Organization

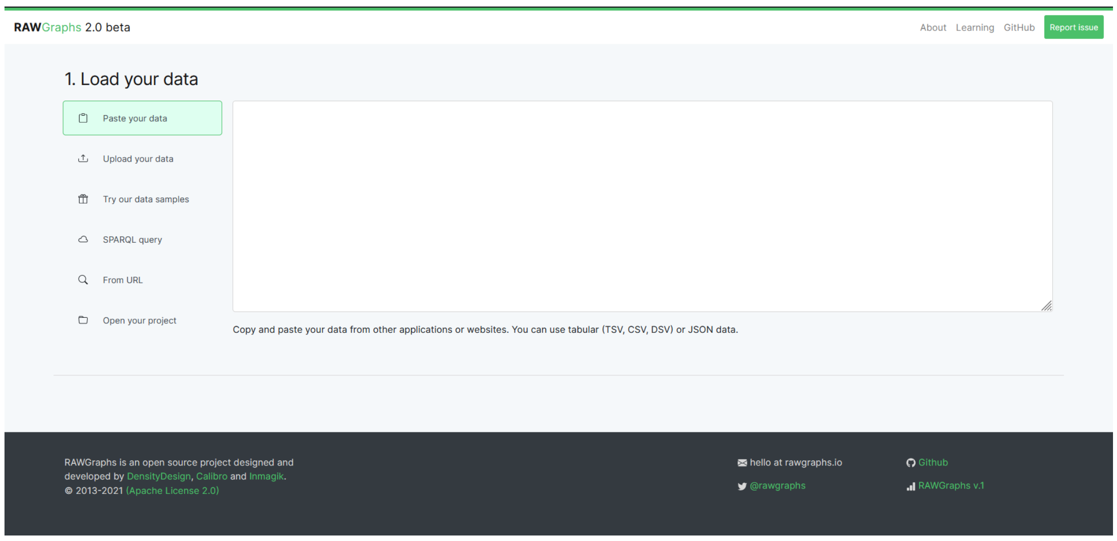
The following chapters make up the project's structure. In Chapter 2, research on previous solutions of similar systems and their limitations are done. Then, a proposed solution and methodology to develop this system is presented in Chapter 3. Next, Chapter 4 describes the system design. Chapter 5 describes the system implementation. Finally, Chapter 6 is simply the conclusion containing a summary of everything in the document.

CHAPTER 2: Literature Reviews

2.1 Review of the Existing Systems/Applications

2.1.1 RAWGraphs

RAWGraphs is an open source data visualization software that makes visualization of complex data easy for everyone.



Copy and paste your data from other applications or websites. You can use tabular (TSV, CSV, DSV) or JSON data.

RAWGraphs is an open source project designed and developed by DensityDesign, Calibro and Inmagik.
© 2013-2021 (Apache License 2.0)

Figure 2.1: Screenshot of RAWGraphs

Installation Methods

Easiest way to access this application is through the web interface. There's no need to install anything.

There is also an option for a local instance, by copying the project files, installing the dependencies, building the project, then running it. With this option, Git, Node.js, and Yarn need to be installed first.

Notable Features

Data Loading

RAWGraphs offers a few methods to load the data, namely

- Pasting the data into the application.
- Uploading the data files, which supports JSON, and CSV but not excel.
- SPARQL Support.
- Using URL pointing to public endpoints (Dropbox file, API endpoints)

It also allows users to load RAWGraphs projects from previous use.

Chart Selection

The application allows users to select from many different chart types with a filter to filter different types of chart types.

The supported charts are as follows.

Alluvial Diagram	Circle Packing	Parallel Coordinates
Arc Diagram	Circular Dendrogram	Pie Chart
Bar Chart	Contour Plot	Radar Chart
Multi-set Bar Chart	Convex Hull	Sankey Diagram
Stacked Bar Chart	Linear Dendrogram	Slope Chart
Beeswarm Plot	Gantt Chart	Streamgraph (area chart)
Box Plot	Hexagonal Binning	Sunburst Diagram
Bubble Chart	Horizon Graph	Treemap
BumpChart	Line Chart	Violin Plot
Calendar Heatmap	Matrix Plot	Voronoi Diagram
Treemap (Voronoi)		

Table 2.1: Supported Chart Types of RAWGraphs

Customizable Data to Chart Mapping

Selecting the charts will show different settings for different mapping menus to assign different variables to different dimensions of the chart. These settings are quite detailed, and allow users to set almost anything.

Customizable Chart UI

Users can also customize the UI in detail, such as setting the width/height, background colour, margin, text colours, and labels.

Export to Image

After customizing the chart, users can just simply export the chart into different formats, notably .svg, .png, .jpg, and their custom .rawgraphs files.

<u>Comparison to our objectives</u>

RAWGraphs actually fulfills a lot of our objectives. It is relatively easy to install, requiring only a few commands, and it can be accessed even without installing it through their web UI. It has a minimal, and clear user interface, and data can be visualized by uploading files to it. Finally, it is a react app so users can run it over the network.

Main technical issue is that it does not actually support real time data, and it generally supports only one chart at a time. We want to support a dashboard style, real time data visualization. To select the files, we also need to use the file picker instead of being able to pick inside the app itself which makes it a bit troublesome.

2.1.2 Datawrapper

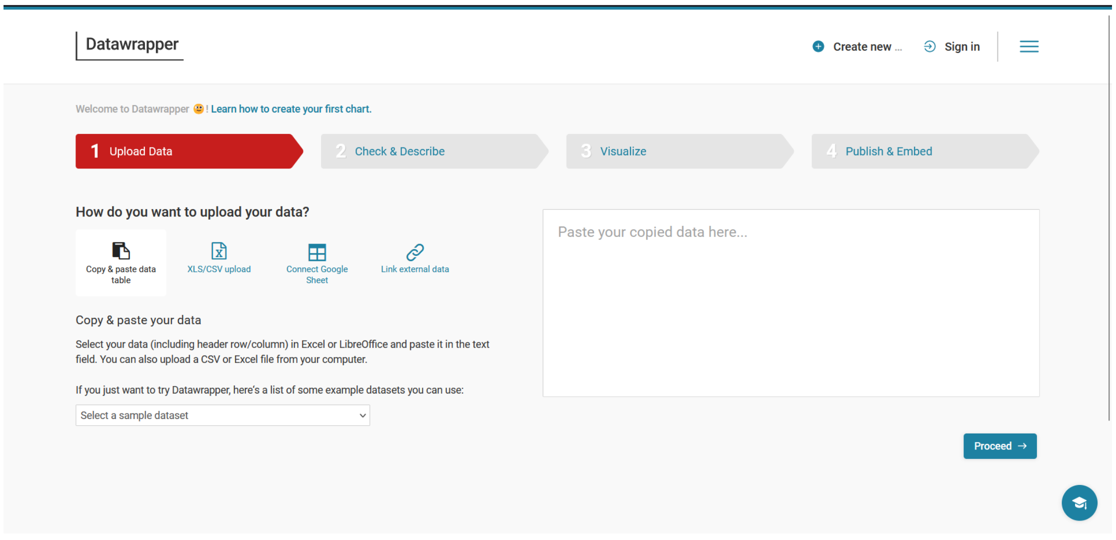
Datawrapper is similar to the RAWGraphs application in the sense that it provides an easy to use web interface for data visualization and sharing. It has a core application that is open source, but most of its functionality is through closed source plugins.

It is not a standalone installable application but rather a web application that is a software as a service, thus we cannot actually install it.

<u>Features</u>

Upload Data

It supports copy/paste data, XLS/CLS file upload, connecting to Google Sheet, and linking to the data directly.



Welcome to Datawrapper 🙂! Learn how to create your first chart.

How do you want to upload your data?

Copy & paste your data

Select your data (including header row/column) in Excel or LibreOffice and paste it in the text field. You can also upload a CSV or Excel file from your computer.

If you just want to try Datawrapper, here's a list of some example datasets you can use:

Figure 2.2: Screenshot of Upload Data portion of Datawrapper

Visualize Data

Datawrapper allows comprehensive chart customisation. Chart types, column, axis, sorting and appearance, and annotations all can be set.

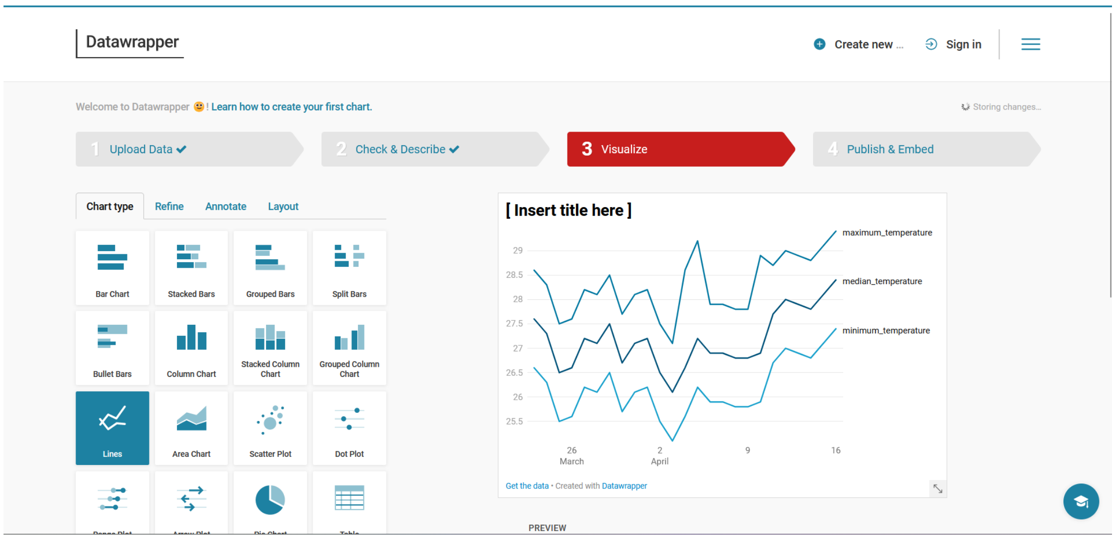


Figure 2.3: Screenshot of Visualize portion of Datawrapper

Supported Chart Type

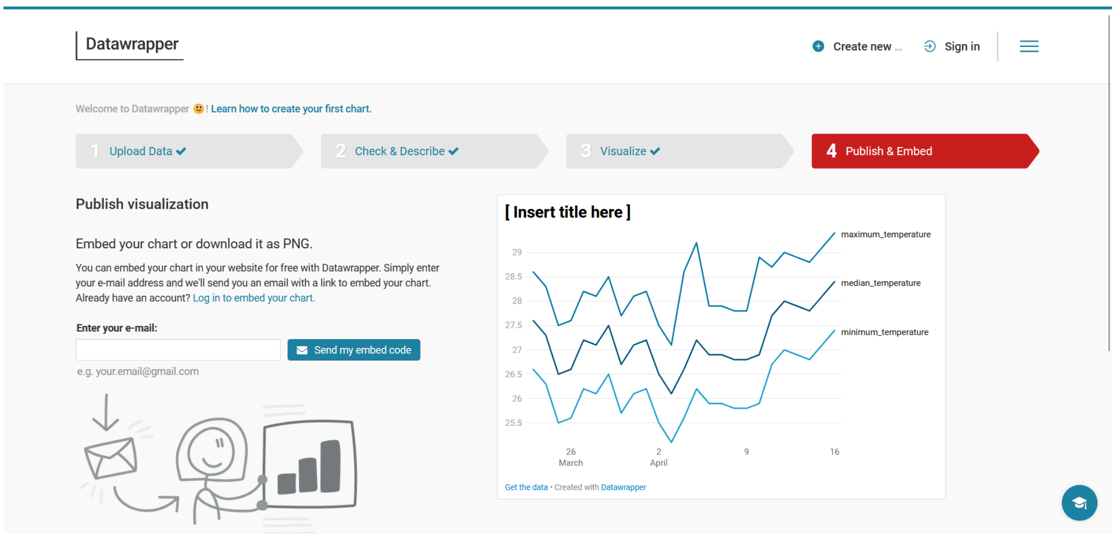
Bar Chart	Stacked Bars	Grouped Bars	Split Bars
Bullet Bars	Column Chart	Stacked Column Chart	Grouped Column Chart
Lines	Area Chart	Scatter Plot	Dot Plot

Range Plot	Arrow Plot	Pie Chart	Table
Donut Chart	Multiple Pies	Multiple Donuts	Element Donut

Table 2.2: Supported Chart Types of Datawrapper

Publishing/Embedding Chart

After uploading the data, and customizing the chart, it is possible to publish the chart to be shared with others, or embedding it somewhere else. The service will send an email with the embed link.



Publish visualization

Embed your chart or download it as PNG.

You can embed your chart in your website for free with Datawrapper. Simply enter your e-mail address and we'll send you an email with a link to embed your chart. Already have an account? Log in to embed your chart.

Enter your e-mail:

e.g. your.email@gmail.com

Figure 2.4: Screenshot of Publish & Embed portion of Datawrapper

<u>Comparison to our objectives</u>

Datawrapper only fulfills some of our requirements, namely minimal and clear user interface, and visualizing data from file-based documents.

On the other hand, users cannot install the application as it is a SaaS application, and it does not support real time data. It also does not support loading the application over the network locally.

Overall, it’s a nice charting application but does not fulfill our requirements.

2.1.3 Redash

Redash is an open source graphing application similar to Grafana that is intentionally built to make it easy to use for people without technical skills.

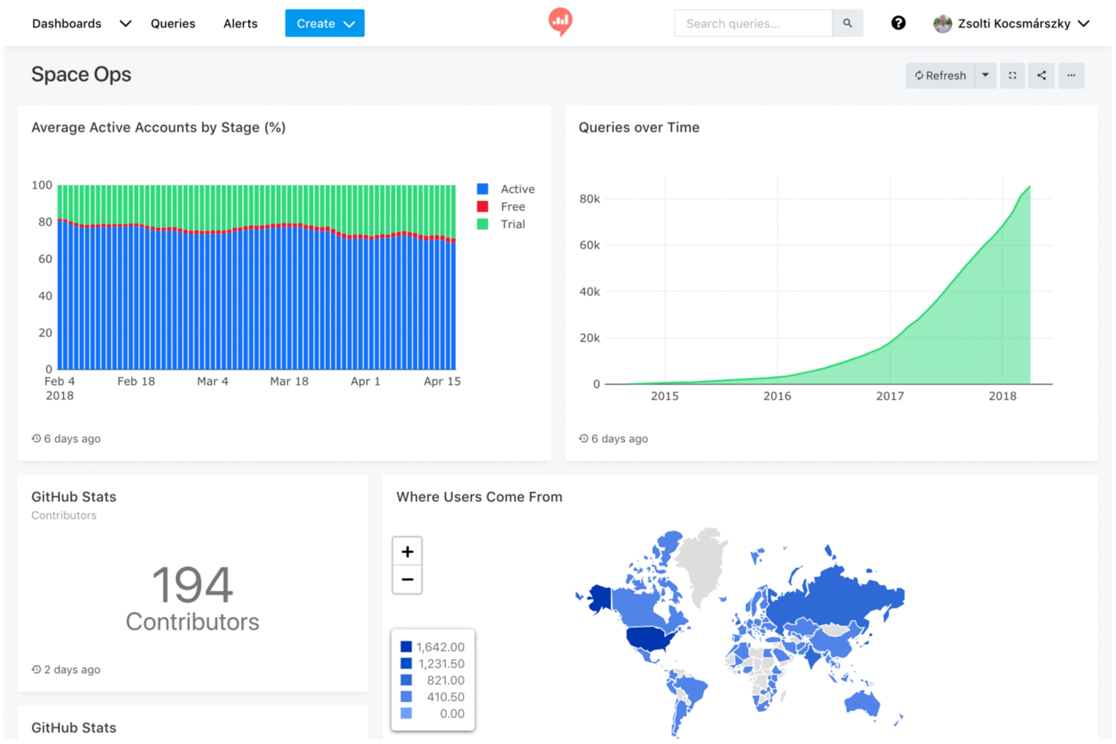


Figure 2.5: Screenshot of the Redash Dashboard

<u>Installation</u>

The installation process of Redash is quite complicated. The first method is using the selected cloud provider's Redash image to automatically spin up a server instance with Redash already installed. The second method is to use the setup script provided which would help users automatically install docker, and docker compose then download the Redash docker image then start it. The third way is to manually pull the Redash docker image and install it that way.

After this, the application would need to be configured through configuration files such as Google OAuth, Email Credentials, and HTTPS. There's also a need to setup the administrator account on first start.

<u>Features</u>

Data Querying

The data querying is usually done through the use of the querying language native to the data source. It features a syntax autocomplete, keyboard shortcuts, and schema browser. It is also possible to publish a chart to be shared, archiving a query, and forking a query.

Supported Data Source

Amazon Athena	Google Spreadsheets	Oracle
Amazon CloudWatch	Graphite	PostgreSQL
Amazon CloudWatch Logs Insights	Greenplum	Presto
Amazon DynamoDB	Hive	Prometheus
Amazon Redshift	Impala	Python
Axibase Time Series Database	InfluxDB	Qubole
Cassandra	JIRA	Rockset
ClickHouse	JSON	Salesforce
CockroachDB	Apache Kylin	ScyllaDB
CSV	OmniSciDB (Formerly MapD)	Shell Scripts
Databricks	MemSQL	Snowflake
DB2 by IBM	Microsoft Azure Data Warehouse / Synapse	SQLite
Druid	Microsoft Azure SQL Database	TreasureData
Elasticsearch	Microsoft SQL Server	Vertica
Google Analytics	MongoDB	Yandex AppMetrica
Google BigQuery	MySQL	Yandex Metrica

Table 2.3: Supported Data Sources of Redash

Supported Charts Types

Line	Bar	Area	Pie
Scatter	Bubble	Heatmap	Box

Table 2.4: Supported Chart Types of Redash

Dashboard

Redash has a feature that allows users to create multiple charts in one page, and arrange them as they see fit forming a dashboard that users can use to manage related charts and see all their charts at once.

Integrations

Redash has support for integrating with Zapier, Slack bot and through API. This allows for migration of data both ways.

<u>Comparison to our objectives</u>

This application fulfills some of our objectives, namely minimal/clean user interface, able to visualize data from file-based documents, and be able to load the app on web browsers over the network (local and internet).

However, it is not easy to install, and does not provide the ability to visualize data in real time.

2.1.4 Summary of the Existing Systems

The systems we reviewed in this section are all very feature rich, and much more advanced from what we've built so far. They have integrations with many other services, they support many different charts, and they are able to accept many different kinds of data.

Unfortunately, all of them did not have the live data requirement, and none of them fulfills all our objectives at once.

CHAPTER 3: System Methodology/Approach

3.1 System Design Diagram / Equation

3.1.1 System Architecture Diagram

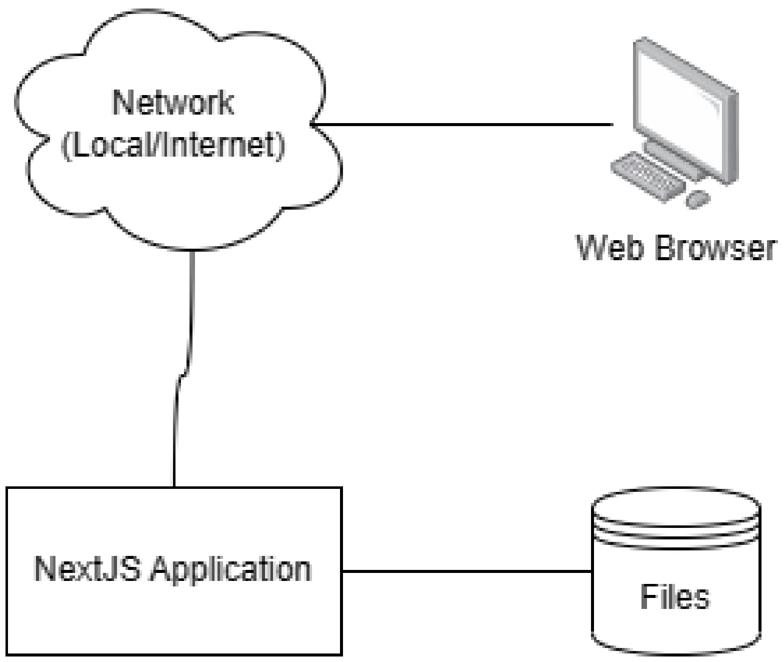


Figure 3.1: System Architecture Diagram

3.1.2 Use Case Diagram and Description

3.1.2.1 Use Case Diagram

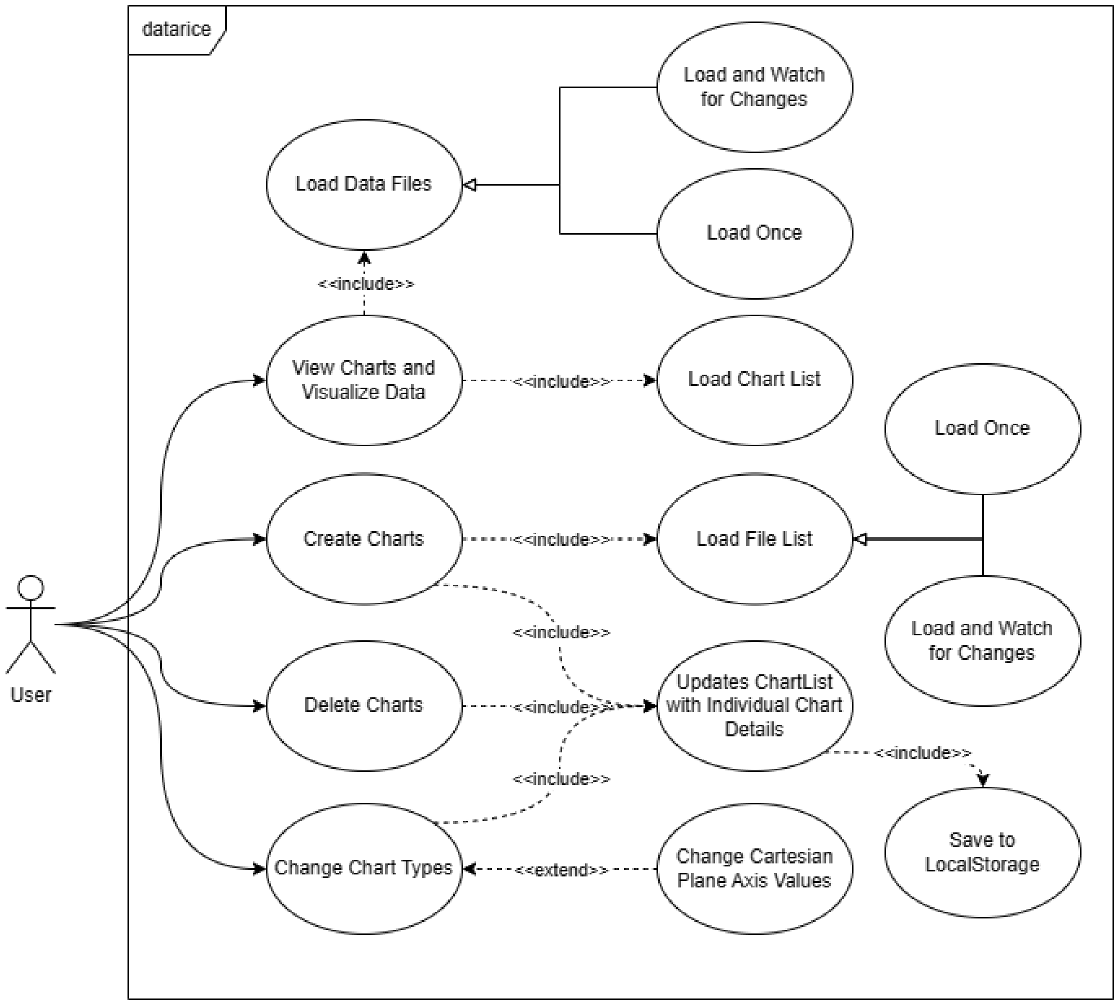


Figure 3.2: Use Case Diagram

3.1.2.2 Use Case Descriptions

Use Case Name: View Charts and Visualize Data	**ID:**	**Important Level:** Very Important
Primary Actor: User		**Use Case Type:**
Stakeholder and Interest: ● User - User will be interested in viewing charts and seeing visualized data. ● System -		
Brief Description: Users will want to see their data visualized.		
Trigger: User loads the dashboard page of the application. **Type:**		
Relationships: ● **Association:** User ● **Include:** Load Data Files, Load Chart List ● **Extend: -** ● **Generalization: -**		
Normal Flow of Event: 1. Users load the dashboard page that lists existing charts. 2. Application loads chart list from the web browser's local storage. 3. If chart data exists in the chart list, load the corresponding data file through WebSocket. 4. Application shows the charts on the page for the user. 5. If there are any changes in the file, the chart is updated through WebSocket events.		
SubFlows: -		
Alternate/Exceptional Flows: -		

Table 3.1: Use Case Description 1

Use Case Name: Create Charts	**ID:**	**Important Level:** Very Important
Primary Actor: User		**Use Case Type:**
Stakeholder and Interest: ● **User** - Users would be interested in creating new charts, and visualizing the data they want.		
Brief Description: This use case describes how users will create new charts.		
Trigger: User clicks the floating button on the bottom right with the "+" button. **Type:**		

Relationships: ● **Association:** User ● **Include:** Load File List, Update ChartList with chart details ● **Extend: -** ● **Generalization: -**
Normal Flow of Event: 1. User clicks the floating button on the bottom right with the "+" label. 2. Chart UI is created, and the Chart UI's settings model is shown. 3. User selects what file they want to load and visualizes. 4. By default, the "line" chart type is selected by default. 5. The application loads the file and updates the UI. 6. The chart is now visualized. 7. Chart details are saved into the web browser's localstorage to be loaded when the user refreshes the page.
SubFlows: -
Alternate/Exceptional Flows: -

Table 3.2: Use Case Description 2

Use Case Name: Delete Charts	**ID:**	**Important Level:** Very Important
Primary Actor: User		**Use Case Type:**
Stakeholder and Interest: ● **User** - User will be interested in deleting some charts they've created. Maybe they want to reduce clutter, or it could be as simple as not having a use for the chart anymore.		
Brief Description: Deletes the chart from the UI.		
Trigger: Clicking the "delete" button inside the ChartSettings model. **Type:**		
Relationships: ● **Association:** User ● **Include:** Update ChartList with chart details ● **Extend: -** ● **Generalization: -**		
Normal Flow of Event: 1. User opens the settings model of the chart they want to delete. 2. User clicks the delete button. 3. The application will delete the chart, and remove it from the UI. 4. The application updates the chart details in local storage.		
SubFlows: -		

Alternate/Exceptional Flows: -

Table 3.3: Use Case Description 3

Use Case Name: Change Chart Types	**ID:**	**Important Level:** Very Important
Primary Actor: User		**Use Case Type:**
Stakeholder and Interest: ● **User** - User may want to change the chart types of a chart. Some data are better visualized using different chart types such as pie charts, or bar charts.		
Brief Description: Change the Chart Types on charts to visualize data differently.		
Trigger: User selecting different chart types in a dropdown menu. **Type:**		
Relationships: ● **Association:** User ● **Include:** Update ChartList with chart details ● **Extend:** Change Cartesian Plane Axis Values ● **Generalization: -**		
Normal Flow of Event: 1. User opens the settings modal of the chart they want to change. 2. User selects the chart types they want to change to. 3. User then click submit. 4. New settings will be applied. 5. The application will update localstorage.		
SubFlows: 1. Line Chart Sub-settings a. When the cartesian based charts (e.g line, bar, scatter) is selected, a new setting will be shown specifically for the line chart. b. User will select the data to be used for the X-Axis. c. Users click submit. d. New settings are applied.		
Alternate/Exceptional Flows: -		

Table 3.4: Use Case Description 4

3.1.3 Activity Diagram

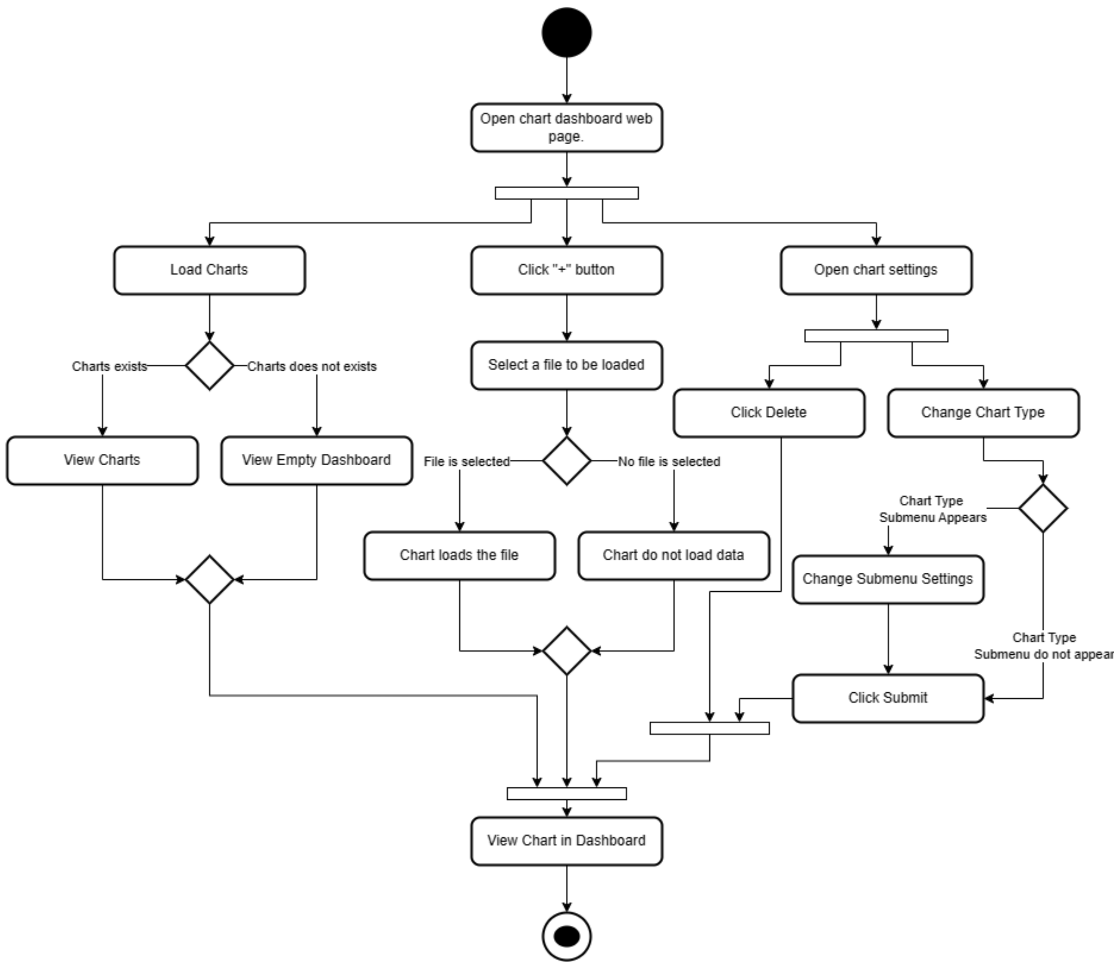


Figure 3.3: Activity Diagram

CHAPTER 4: System Design

4.1 Brief description on the technologies, and how it's built

The application is a server rendered web application that makes use of the latest (as of 2023) frontend technologies.

The core technologies are built upon NodeJS, and NPM, both of which should be installed manually.

Must Install

- **Node.js** - Javascript backend runtime environment.
- **NPM** - Node Package Manager

Libraries, and frameworks

- **Next.js** - React server side rendering framework.
- **React** - Client-side library to build user interface in a modular and declarative way.
- **Zustand** - Small and fast state management solution that improves upon React's native state management system.
- **Socket.io** - A client and server WebSocket library.
- **XLSX -** Library to read sheet files like CSV, or Excel.

The server side serves a WebSocket API that provides the ability to read and return data from a specific file, as well as watch individual files and directories for changes, and automatically send these changes to be reflected on the client side.

While on the client, websocket is used to retrieve raw data from the server, then transformed, and fed to a charting library to visualize a chart. These are all contained within a React application.

There are no databases used in this application, as it relies fully on reading files to get its data.

Essentially, it's a React application that visualizes data, and a small backend that retrieves the data from files on the local machine.

Deployment and installation is done by copying the project files, building the application, and running it.

Note: The repository containing the source code of this application can be found here.

4.2 Block Diagram and System Component Explanation

Here is a system block diagram illustrating how the application works, and how the various components interact with each other.

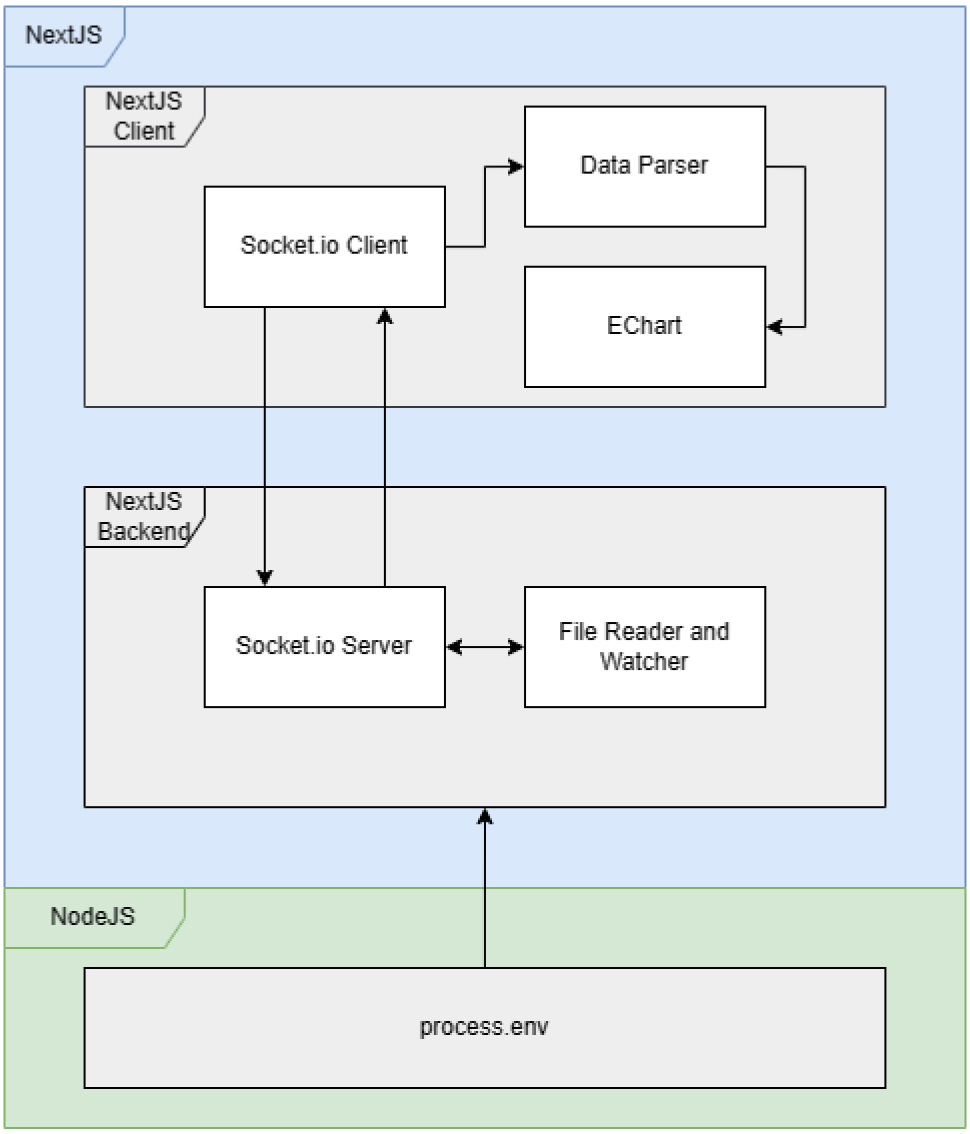


Figure 4.1: System Block Diagram

Configuration

When starting, the application first reads the .env.local file that is responsible for configuring the application.

There are 2 environment variables used to configure the application, and they are:

- NODE_ENV - Sets the build mode of the project. Available values are development and production. This affects how the project is built.
- DATA_STORE_DIRECTORY - Sets the directory on which the application is looking at. Usually point this to the folder where the user has data files that they want to visualise.

If .env.local is not set, it will by default read either the .env.production, or the .env.development file.

WebSocket Server

After reading the configuration file, the NextJS application will start a WebSockets server that will work with 2 different WebSocket events.

- **list-files** - handle file listing
- **load-data** - handle data loading

WebSocket events are 2 ways, so they function differently depending on who is sending them.

When the server sends `**list-files**`, it is usually responding to the client's request, and sending the list of files to the client, and when it sends `**load-data**` it is sending the data from the file it read to the requesting client.

When the client sends `**list-files**` it means that the client requests a list of files to be sent to the client, and when it sends `**load-data**` it means that the client is requesting the data of a specific file to be sent.

File Reader and Watcher

This module handles reading files, and watching files and directories. The module provides a simple read function that reads the file and returns the data once. The unique part is that there is also a file/directory watcher module that will continuously read and watch file/directories for changes then call a websocket callback that will send the changes to the client automatically. This results in the chart being continuously updated as it is written to the file.

WebSocket Client and Data Parser

The client will initialize a connection to retrieve the file list, and data as needed. The data received from the 2 events are passed to the Data Parser, and transformed into echart’s dataset format.

Dataset Format Documentation:
https://apache.github.io/echarts-handbook/en/concepts/dataset/

The Data Parser is responsible for handling the differences between various file formats, and transforming them to a common data format that the chart can use.

4.3 System Compilation and Deployment

The system compilation, and deployment is actually very simple. The commands for compilation, and deployment can be summarized as follows. Inside the root directory of the project, we run:

- npm run ci
- npm run build
- npm run start

Respectively, these commands will install the dependencies, build the production version of the application, and then run it on the local machine.

CHAPTER 5: System Implementation

5.1 Hardware Setup

The software does not need any special hardware, and is designed to run on any hardware capable of supporting the more recent versions of Windows, Linux, and MacOS.

In general, a computer with a 2 GHz Intel Dual Core Processor, and more than 4GB RAM should be considered minimum.

5.1 Software Setup

The base project is built using NodeJS v18, and managed by git. Thus, the basic softwares that is needed to be installed are NodeJS v18, together with NPM. Git is optional, but recommended as it simplifies the process of updating the project in the future.

To install the application we need to do the following,

- Make a copy of the project files on the local machine.
- Install the project dependencies.
- Build the project.
- Run the application.

And there! The software is now installed, and it is now accessible on the localhost URL provided.

5.3 Setting and Configuration

The application is configurable through environment variables through a few .env.* files.

The .env, .env.development, and .env.production files are all configuration files setting some default environment variables. For more information please have a look at this documentation.

The environment variables for this project are

- NODE_ENV - Sets the build mode of the project. Available values are development and production. This affects how the project is built.
- DATA_STORE_DIRECTORY - Sets the directory on which the application is looking at. Usually point this to the folder where the user have data files that they want to visualise.

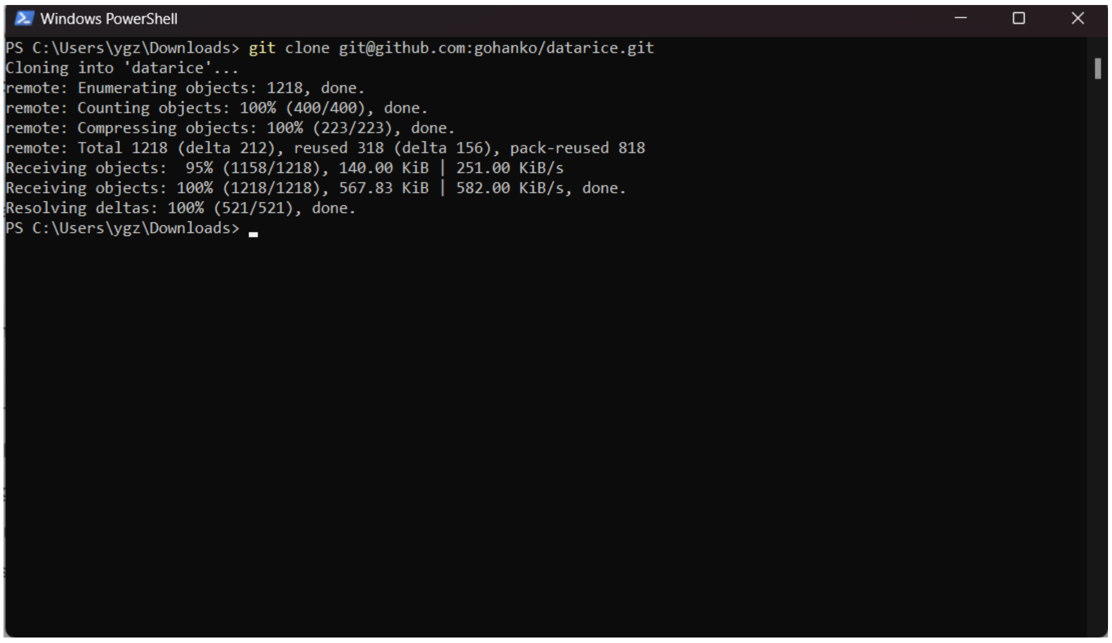
To override the defaults, one should add a .env.local file in the root of this project, and set the variables there.

5.4 System Operation (with screenshot)

The system operation for this application is actually quite simple. There are 2 parts, the setup, and the usage part.

<u>Setup</u>

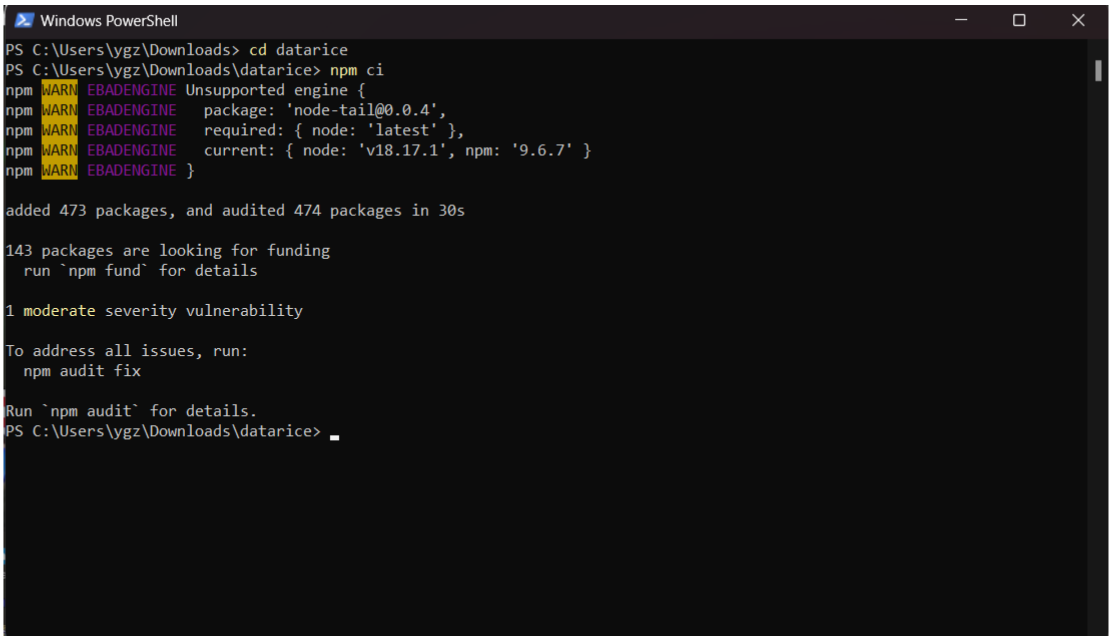
Cloning the project files



```
PS C:\Users\ygz\Downloads> git clone git@github.com:gohanko/datarice.git
Cloning into 'datarice'...
remote: Enumerating objects: 1218, done.
remote: Counting objects: 100% (400/400), done.
remote: Compressing objects: 100% (223/223), done.
remote: Total 1218 (delta 212), reused 318 (delta 156), pack-reused 818
Receiving objects:  95% (1158/1218), 140.00 KiB | 251.00 KiB/s
Receiving objects: 100% (1218/1218), 567.83 KiB | 582.00 KiB/s, done.
Resolving deltas: 100% (521/521), done.
PS C:\Users\ygz\Downloads>
```

Figure 5.1: Screenshot of Cloning the Project using Git

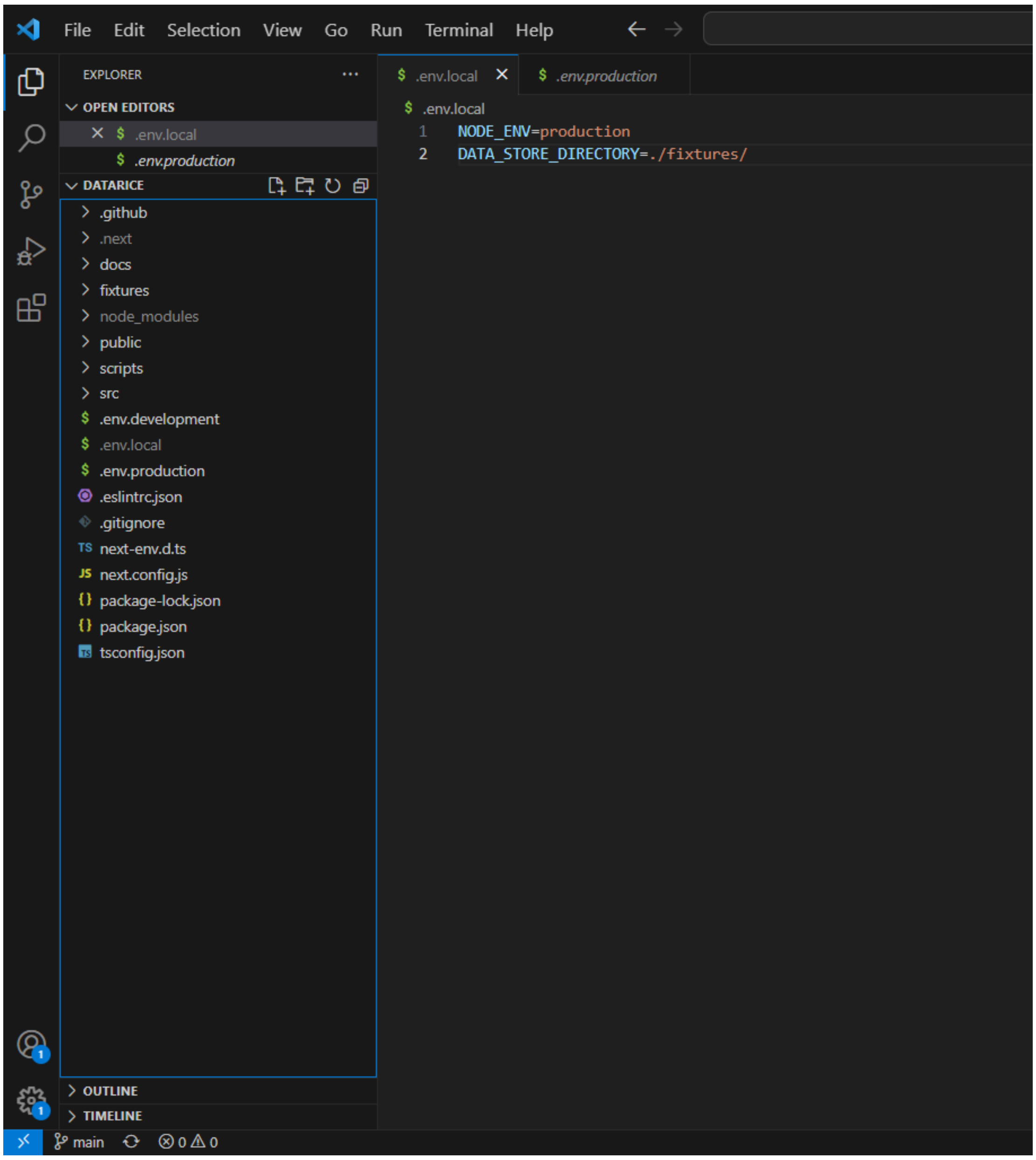
Installing the dependencies



```
PS C:\Users\ygz\Downloads> cd datarice
PS C:\Users\ygz\Downloads\datarice> npm ci
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: 'node-tail@0.0.4',
npm WARN EBADENGINE   required: { node: 'latest' },
npm WARN EBADENGINE   current: { node: 'v18.17.1', npm: '9.6.7' }
npm WARN EBADENGINE }

added 473 packages, and audited 474 packages in 30s

143 packages are looking for funding
  run `npm fund` for details

1 moderate severity vulnerability

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
PS C:\Users\ygz\Downloads\datarice>
```

Figure 5.2: *Installing Dependency using NPM*

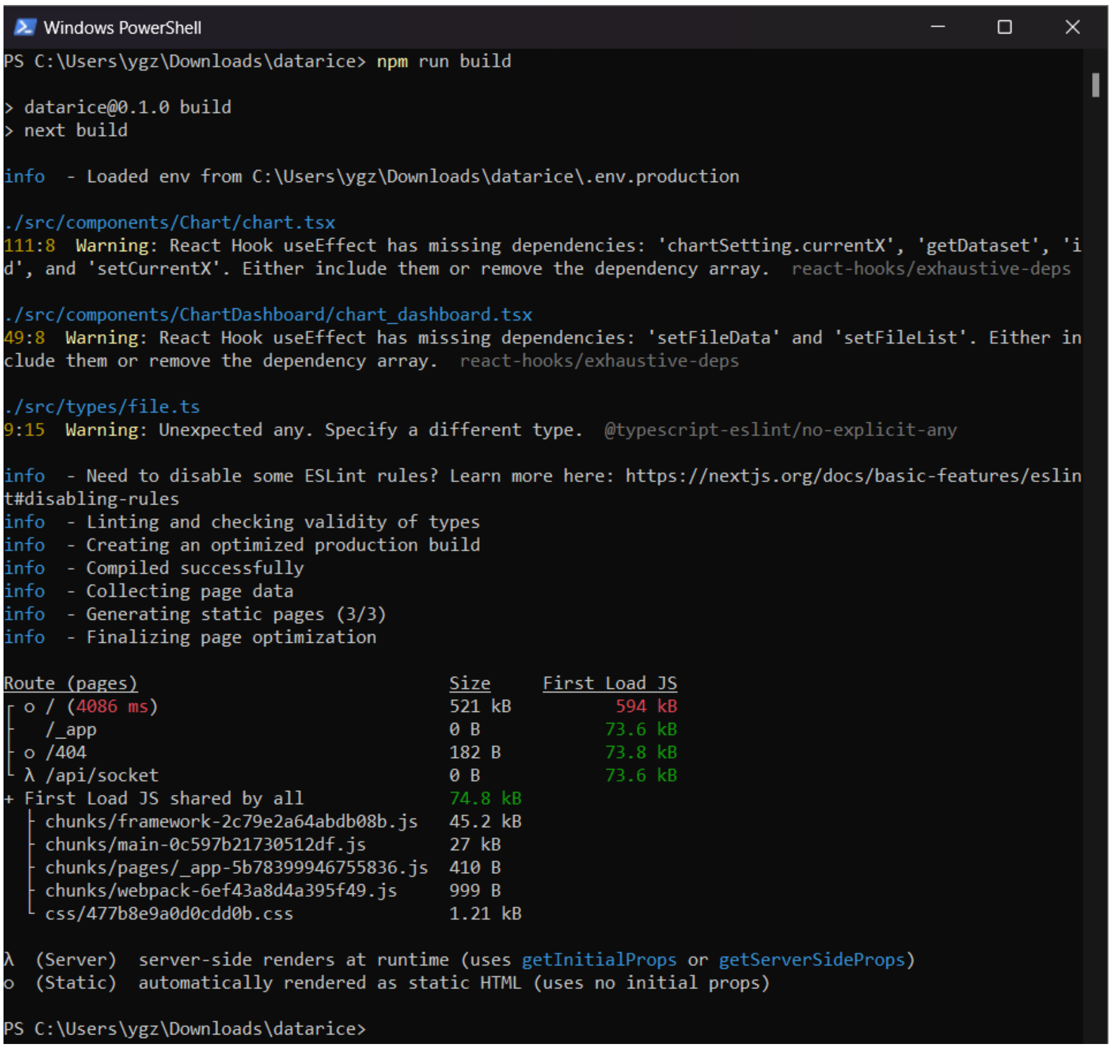
Configuring the project



```
NODE_ENV=production
DATA_STORE_DIRECTORY=./fixtures/
```

Figure 5.3*: Configuring the app using VSCode*

Building the application

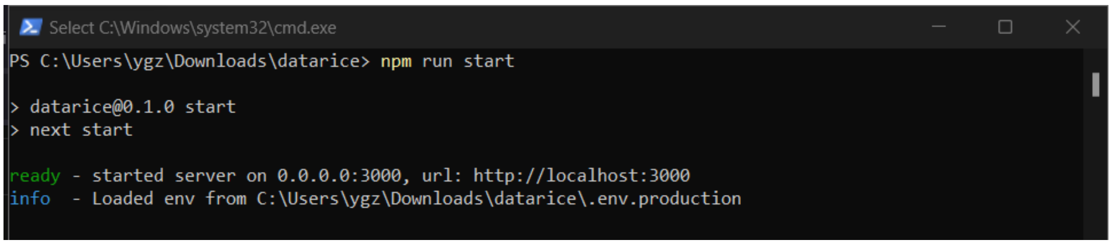


Windows PowerShell

```
PS C:\Users\ygz\Downloads\datarice> npm run build

> datarice@0.1.0 build
> next build

info  - Loaded env from C:\Users\ygz\Downloads\datarice\.env.production

./src/components/Chart/chart.tsx
111:8  Warning: React Hook useEffect has missing dependencies: 'chartSetting.currentX', 'getDataset', 'i
d', and 'setCurrentX'. Either include them or remove the dependency array.  react-hooks/exhaustive-deps

./src/components/ChartDashboard/chart_dashboard.tsx
49:8  Warning: React Hook useEffect has missing dependencies: 'setFileData' and 'setFileList'. Either in
clude them or remove the dependency array.  react-hooks/exhaustive-deps

./src/types/file.ts
9:15  Warning: Unexpected any. Specify a different type.  @typescript-eslint/no-explicit-any

info  - Need to disable some ESLint rules? Learn more here: https://nextjs.org/docs/basic-features/eslin
t#disabling-rules
info  - Linting and checking validity of types
info  - Creating an optimized production build
info  - Compiled successfully
info  - Collecting page data
info  - Generating static pages (3/3)
info  - Finalizing page optimization

Route (pages)                              Size     First Load JS
┌ ○ / (4086 ms)                            521 kB         594 kB
├   /_app                                  0 B           73.6 kB
├ ○ /404                                   182 B         73.8 kB
└ λ /api/socket                            0 B           73.6 kB
+ First Load JS shared by all              74.8 kB
  ├ chunks/framework-2c79e2a64abdb08b.js   45.2 kB
  ├ chunks/main-0c597b21730512df.js        27 kB
  ├ chunks/pages/_app-5b78399946755836.js  410 B
  ├ chunks/webpack-6ef43a8d4a395f49.js     999 B
  └ css/477b8e9a0d0cdd0b.css               1.21 kB

λ  (Server)  server-side renders at runtime (uses getInitialProps or getServerSideProps)
○  (Static)  automatically rendered as static HTML (uses no initial props)

PS C:\Users\ygz\Downloads\datarice>
```

Figure 5.4: Building the application using NPM

Running the application



Select C:\Windows\system32\cmd.exe

```
PS C:\Users\ygz\Downloads\datarice> npm run start

> datarice@0.1.0 start
> next start

ready - started server on 0.0.0.0:3000, url: http://localhost:3000
info  - Loaded env from C:\Users\ygz\Downloads\datarice\.env.production
```

Figure 5.5: Running the application through NPM

Usage

Adding a new chart

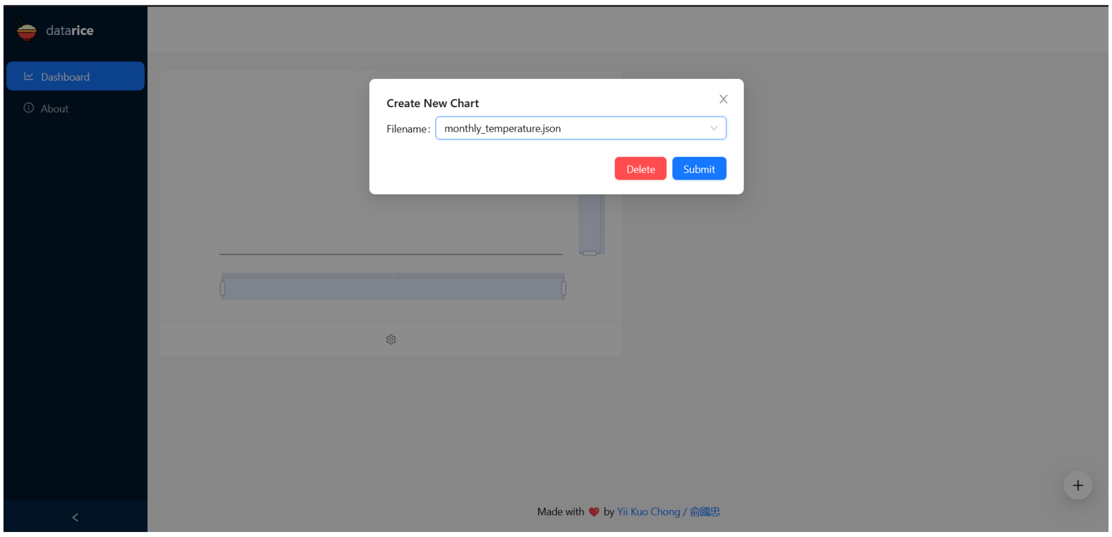

datarice
Dashboard
About
Create New Chart
Filename: monthly_temperature.json
Delete
Submit
Made with ❤ by Yii Kuo Chong / 俞國忠

Figure 5.6: Creating new charts

Configuring a chart

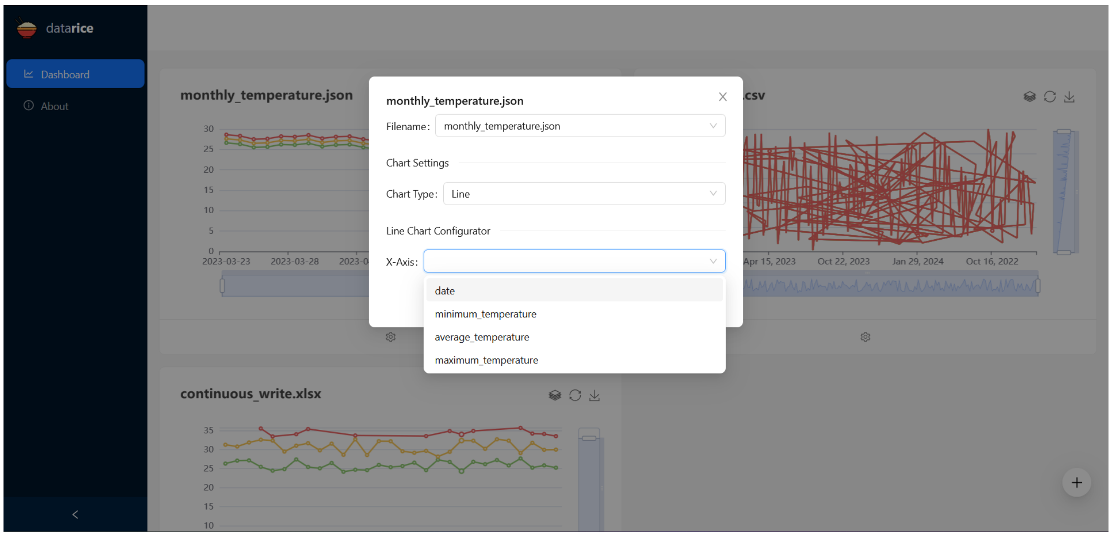
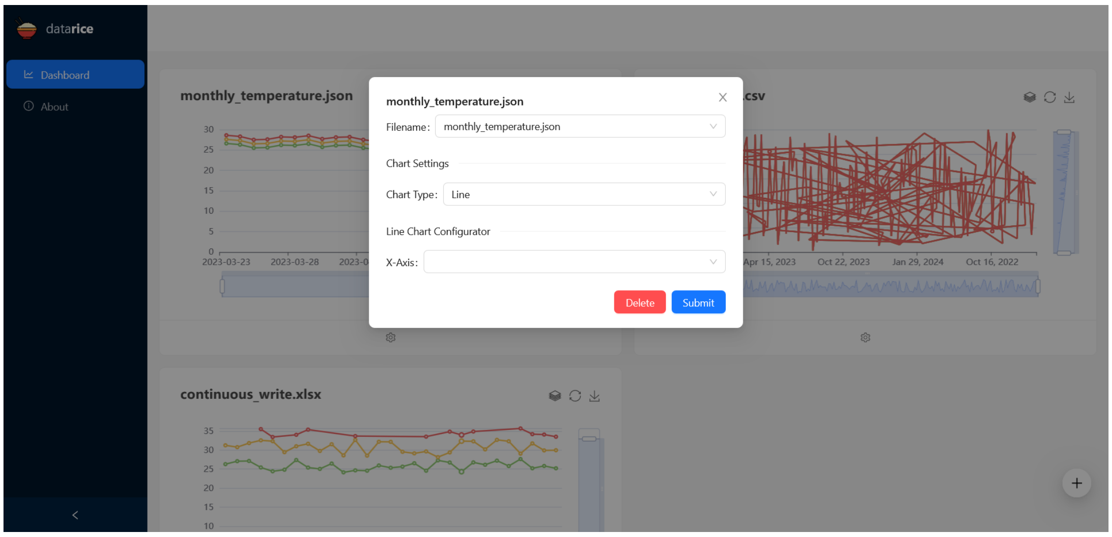


Figure 5.7: Configuring existing charts

Deleting a chart by pressing the delete button



monthly_temperature.json
Filename: monthly_temperature.json
Chart Settings
Chart Type: Line
Line Chart Configurator
X-Axis:

Figure 5.8: Deleting a chart

5.5 Implementation Issues and Challenges

There were quite a few issues during the implementation of this project. They were all resolved.

Dependency Integration Issues

The issue mostly comes from integrating 3rd party charting libraries into the React application.

At first, I chose a charting library called Ant Design Charts, because it has the same UI design of the UI library (Ant Design v5) I used so that it doesn't look out of place. However, when using Ant Design Charts, I noticed that there was a lot of issue integrating it into the application, and I found out that it’s because Ant Design Charts v4 doesn’t support the latest version of Ant Design.

Then, I chose a charting library called Recharts, but it had a lot of issues with viewing live data. The application would re-render the chart every time the data is updated causing an issue where the chart flickers every time new data is added. It also chose to animate it as if the chart is initialized for the first time, instead of smoothly adding new data points onto the chart. The second problem for this chart is related to ES6 and CommonJS, because D3 which ReCharts relies on CommonJS, does not play nice with our ES6 based codebase.

Finally, I chose a charting library that does not rely on D3, has a lot of pre-built features that I need, and is fully ES6 compatible called ECharts. There's also a problem with this since it is not a React application, and React usually does not play nice with non-React code, but the problem was minimal.

Websockets Integration

The application has both the server implementation, and the client implementation of WebSockets, which helps get the data to serve, and to receive the data to display on screen. However, NextJS (the React framework we're using) does not support server side use of WebSockets which severely limits what the Websockets can do. So far, the current implementation works well to serve our purpose so there's not much that needs to be done.

Data Streaming Challenge

Since the application's purpose is to visualize data, we will have to load and retrieve the data from somewhere. The problem now is that we are using React as the UI framework, and React is a declarative, state based framework. That would mean that when storing data, we should store it in a state which would then trigger an update of the user interface, and render the new chart. This would work well when data is loaded and displayed at once, but it starts becoming a problem when data is streamed to the client since every data update will trigger a component re-render causing some performance issues, and side effects like the chart being reinitialized over and over again. This was fixed by reducing the amount of updates triggered, and configured a setting that does not reinitialize the chart every update. This creates a seemingly continuously updating chart without graphical glitches.

CHAPTER 6: Conclusion and Recommendation

6.1 Conclusion

With the application's development complete, we provided an easy to use, easy to install, real time data visualization web application for students, and researchers to use. This will provide a platform for students, and researchers to do their work which hopefully will reduce time and effort which can be directed elsewhere in their work.

That said, the system is currently very barebones, and we should take the effort to add more features, and improvements to the application to make it a full fledged and more useful application.

6.2 Recommendation

The table below contains recommendations for future developments.

No.	Recommendation	Explanation	Benefit
01.	Split the application into server, and client parts. Then package it as if it's one application.	NextJS is ill-equipped to handle WebSockets, so it is more ideal to split the websocket server into its own thing. Then the packaging should be done in a way that makes it seamlessly look like one application.	We will be able to properly split server, and client code thus reducing bugs.
02.	Refactor the code into more modular pieces with proper decoupling of the user interface, and business logic.	The business logic, and the user interface is too intertwined which makes it hard to write tests for it.	We will be able to write tests for it.
03.	Write more tests for the entire codebase.	Currently, we have not written any tests for the components of the application. Manually testing the application is a time consuming and error prone	Less bugs, more free time.

		process. Once recommendation No. 02 is done, we should write tests that are automatically run by the CI/CD application.	
04.	Add support for more data sources.	Currently, we have implemented a data source for local files. However, there are many different types of data sources that can be implemented such as Rest APIs, GraphQLs, and custom Websocket endpoints to name a few.	Improve ease of use, and improve features.
05.	Improve UI design.	The current UI is decent but can be better. A better UI design should be done to make sure the application is good to use.	Improve ease to use, and strain on the eye.
06.	Add an extension system.	There are many functionalities that can be added, however not all need to be added in the core application. Adding to that, some users may want to maintain their own sets of extensions rather than contributing to the main application.	Being able to add more functionalities in a decentralized manner, and reduce code bloat.

Table 6.1: Recommendations for Future work

REFERENCES

Rawgraphs 2.0. (2023). *Rawgraphs 2.0*. RAWGraphs 2.0. https://app.rawgraphs.io/

RAWGraphs. (2023). *Rawgraphs*. https://www.rawgraphs.io/

Datawrapper. (2023). *Datawrapper*. https://www.datawrapper.de/

Datawrapper. (2023). *Datawrapper*. https://app.datawrapper.de/chart/Dq8b0/upload

Redash. (2023). *Redash*. https://redash.io/help/

APPENDIX

FINAL YEAR PROJECT WEEKLY REPORT 1

(Project I)

Trimester, Year: T3Y2	**Study week no.:** 14 & 15
Student Name & ID: Yii Kuo Chong (2102670)	
Supervisor: Ooi Boon Yaik	
Project Title: Data Visualisation for Text-based Document	

1. WORK DONE

After submitting the Final Year Project 1 Report, I continued working on the project to make sure I have enough time for FYP 2 next semester.

The following weekly reports are from the Git commits of the project as I worked during the semester break as well.

Apr 24, 2023

- Added capability to reload data when data file changes using websocket

Apr 25, 2023

- Save work
- Fixed bug where only one chart shows

During the final examination week, I kept on working on the project. Renaming the project a couple of times before settling on DataRice.

Apr 30, 2023

- Added mock data, and make the app add/remove charts as files are added and removed
- Clean up some code

May 01, 2023

- Rename project to hogmaster
- Changed name to DataRice
- Update UI elements to make it more presentable

2. WORK TO BE DONE

- Convert the project into TypeScript.
- Find and test a solution to compile Node application into binaries for packaging.
- Add systemd support.
- Use .env file for configuration.
- Add ESLint rule to cleanup code.
- Refactor the code into more modular parts.

● Standardize dataset.
3. PROBLEMS ENCOUNTERED No comment
4. SELF EVALUATION OF THE PROGRESS This week there is minimal progress. Mostly maintenance, and bug fixing rather than adding new features. Adding the code to reload data when files change is a good step forward towards showing live data as it is added or removed.

[illegible]

Supervisor's signature

[illegible]

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT 2

(Project I)

Trimester, Year: T3Y2	**Study week no.:** Not Applicable
Student Name & ID: Yii Kuo Chong (2102670)	
Supervisor: Ooi Boon Yaik	
Project Title: Data Visualisation for Text-based Document	

1. WORK DONE

May 20, 2023

- nginx: attempted to serve file using nginx
- Converted to TypeScript, and replaced Recharts with Echarts

May 21, 2023

- pkg: compileable version of the project. it throws an MODULE_NOT_FOUND error for socket-io for some reason
- Restructure folder structure, and remove pkg
- Update README
- Added a systemd service file
- Removed unused root index.js and server.js
- Added .env files to easily configure the application
- Fixed and issue where new data overwrites data in all charts
- Set Chart's yAxis minimum value to the minimum value of dataset minus 1
- Use Ant.Design Grid for charts
- Update folder structure, and update ESLint rules to reduce unused variables

May 22, 2023

- Added header Icon and Title. Adding new menu options and add new icons
- Moved folder/file watching logic into utility.ts
- Made the menu selectable
- Make the respond to the sidebar collapse event
- Standardize data set
- Chart now decides which data to use on the xAxis, and which to use on the yAxis. Anything beyond the first data will be plotted while the first one will be used on the xAxis
- Cleanup mock_data

2. WORK TO BE DONE

- Add a menu for configuring each chart.
- Add settings to set size, and chart types.

3. PROBLEMS ENCOUNTERED

While attempting to compile the NodeJS application into a single executable binary using pkg, I ran into a problem where it simply won't properly compile. Promptly, I abandoned the idea in favor of adding more features to the project first. The idea can be revisited in the future.

There's also the issue where when one chart updates, all the other chart updates. I found out that this is because everytime a new handler is bound to the WebSocket callback, it overwrites the previous one, so all charts suddenly share the same callback. That causes all the charts to update when one updates.

4. SELF EVALUATION OF THE PROGRESS

This week, it's mostly experimenting with packaging, refactoring code, and fixing issues. There are minor changes such as adding new icons, and standardizing the dataset.

Overall, the progress this week is kind of okay.

[illegible] [illegible]

Supervisor's signature Student's signature

FINAL YEAR PROJECT WEEKLY REPORT 3

(Project I)

Trimester, Year: T3Y2	**Study week no.:** Not Applicable
Student Name & ID: Yii Kuo Chong (2102670)	
Supervisor: Ooi Boon Yaik	
Project Title: Data Visualisation for Text-based Document	

1. WORK DONE

June 03, 2023

- fix: fixed websocket not updating client file_list and chart still visualizing removed data

June 04, 2023

- Added modal for ChartSettings
- Separate Chart and ChartManager
- Make rendering charts more configurable. We can now set size and chart type.
- Refactor how chart_options is created
- Bump socket.io-parser from 4.2.2 to 4.2.4
- Merge pull request #3 from gohanko/dependabot/npm_and_yarn/socket.io-parser-4.2.4

June 05, 2023

- move convert_data_to_2D_table to UI Component folder
- move functions related to setting chart_options into its own module

2. WORK TO BE DONE

- Add ability to add charts.
- Add ability to remove charts.
- Improve chart settings.
- Refactor the code so that UI and logic is decoupled.

3. PROBLEMS ENCOUNTERED

For some reason, the websockets isn't updating the file list needed by the file selector in the chart.

4. SELF EVALUATION OF THE PROGRESS

Like previously, there's minimal feature addition, more refactoring, and maintenance to make the code easier to work with. We did however, add a new modal for configuring the charts.

[illegible]

Supervisor's signature

[illegible]

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT 4

(Project I)

Trimester, Year: T3Y2	**Study week no.:** Not Applicable
Student Name & ID: Yii Kuo Chong (2102670)	
Supervisor: Ooi Boon Yaik	
Project Title: Data Visualisation for Text-based Document	

1. WORK DONE

July 08, 2023
- Update Formatter
- Added ability to add chart using FloatButton

July 09, 2023
- Updated antd package

July 10, 2023
- Work in Progress: Adding ability to add charts while using chart's settings component
- Use Forms in ChartSettings Modal
- Add ability to remove chart
- Added custom submit and delete button to the modal

July 11, 2023
- fix: Fixed an issue where websocket doesn't load the data properly
- refactor: Renamed ChartManager to ChartDashboard
- refactor: Separate business logic from UI in Charts, ChartSettings, and ChartDashboard
- Limit eCharts magicType to just line, bar, and stack
- Added a list of supoorted chart types

2. WORK TO BE DONE
- Add settings for specific chart types.
- Add parsing support for XLSX files.
- Use Zustand for managing states to avoid prop drilling, and persistent charts through sessions.

3. PROBLEMS ENCOUNTERED

Fixed websocket issue where it doesn't load data properly.

4. SELF EVALUATION OF THE PROGRESS

Mostly tweaking, refactoring, and maintenance. Added ability to remove charts, and separated business logic from UI. Okay progress.

[illegible]

Supervisor's signature

[illegible]

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT 5
(Project I)

Trimester, Year: T1Y3	**Study week no.:** 5 & 6
Student Name & ID: Yii Kuo Chong (2102670)	
Supervisor: Ooi Boon Yaik	
Project Title: Data Visualisation for Text-based Document	

1. WORK DONE

July 22, 2023
- Added Chart Settings menu to chart form

July 24, 2023
- Refactored variable name from index -> chart_id. If no selected_filename is set, and setting closes, the chart is deleted
- refactor: isChartSettingsOpen, setIsChartSettingsOpen to isSettingsOpen, and setIsSettingsOpen respectively
- refactor: isChartSettingsOpen, setIsChartSettingsOpen to isSettingsOpen, and setIsSettingsOpen respectively
- Added list of chart_type to be selected (currently not functional)
- Added extra methods to set echart options title, chart_type, and data
- Added an option to select chart type
- Moved ChartOptionManager to it's own module
- Tweaked ESLint, and fixed Typescript Lint error
- Updated README with screenshot
- Removed variable from react-hook dependency array which caused an infinite loop bug
- Removed original README.md
- Fixed windows style slash in path to unix style

July 25, 2023
- Added Zustand state management framework for persistent cross session states
- Fixed Zustand hydration issue with a component that waits for the NextJS app to finish rehydrating
- Added a store for FileList, removing the need to pass the list down as props
- Added support allowing Charts to persists through sessions
- Fixed setDataURL parameter type
- Renamed mock_data -> fixtures. Moved up deployment files into their own folder

July 27, 2023
- Use index.ts in HydrationZustand to shorten import path
- chart_type is now persistent across sessions
- Added a new excel fixture. Added more metadata about file from the server. Refactored some file and variable names

July 28, 2023
- Added parsing support for XLSX
- Updated ESLint config and enforced the rules
- Reduce folder depth for a couple of helper modules
- Use ChartType for Chart component's prop types

- Fixed issue with settings not closing
- Remove usage of EChartOptionManager, and directly manage EChart options in the Chart component

July 30, 2023

- Remove unused imports

2. WORK TO BE DONE

- Use Immer to make management of immutable variables easier.
- Add a united storage using Zustand so that the app does not reload the same file over and over again.
- Lay the groundwork for building a debian package.
- Add CSV support.
- Fix CI/CD failure issue with empty test command.
- Add DataZoom feature.
- Add sliders to zoom for charts.
- Prepare README and project itself for external users. It should be stable, and have clear documentation.

3. PROBLEMS ENCOUNTERED

Encountered quite a few bugs, namely React infinite re-render caused by variables in hook dependency array, and windows style slash not working properly with one of the libraries.
Added Zustand, but Zustand doesn't work properly with NextJS because it cannot hydrate from the server side, so we have to force it to render server side before hydrating.
Note: Hydration is a process of loading the basic UI, and then filling it with data. The data we have are stored client-side so we cannot do it on the server.

4. SELF EVALUATION OF THE PROGRESS

A lot of refactoring, but also added a lot of new features. Addition of Zustand, means it's easier to manage states, and it is now persistent across sessions. Good progress.

Alex

Supervisor's signature

[illegible]

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT 6
(Project I)

Trimester, Year: T1Y3	**Study week no.:** 7 & 8
Student Name & ID: Yii Kuo Chong (2102670)	
Supervisor: Ooi Boon Yaik	
Project Title: Data Visualisation for Text-based Document	

1. WORK DONE

July 31, 2023
- Added immer and make use of selectors for zustand related items.
- Adapt our Zustand stores to use immer properly (manipulating immutable states are now so much cleaner)
- Added chart title in echart options
- Added foundation work for a line chart configurator to add ability to set x or y axis
- Added the option to set X axis on a cartesian grid chart
- Ensure labels are shown
- Check if values in createItemAndLabel has values
- Installed sharp packaage for optimized image in production
- Added 'standalone' to next.config.js to build standalone builds
- Added scripts to build a debian package

Aug 01, 2023
- Create a united store for all loaded data so charts can share the same data if they are loading the same file
- Refactor and cleanup some parts of ChartDashboard
- Fixed data loading issue

Aug 02, 2023
- Added script to demonstrate continuous data updates

Aug 03, 2023
- Added CSV dataset and support CSV
- Renamed LineChartConfigurator -> CartesianPlaneConfigurator, and made it support other CartesianPlane based charts

Aug 04, 2023
- Update README.md

Aug 06, 2023
- Add Acknowledgement section, and fix gramatical weirdness
- Fix typos and made changesin README.md
- Added a clause to acknowledge UTAR
- Added an empty test command to prepare for CI/CD
- Create run_test.yml
- Updated README with Aaron's real email

Aug 08, 2023
- Add DataZoom on chart
- Remove unused css and refactor chart props
- Rename variables to follow camelCase
- Added dayjs in the install list

Aug 09, 2023

• Fixed variable naming issue • Update chart.id generation and reduce state updates by updating echarts through ref • Created a deepCopy utility function and use it • Revert "Created a deepCopy utility function and use it" • Remove extra routes, and fixed issues with sliders showing up for pie chart • Fix infinite setState issue
2. WORK TO BE DONE • Prepare the README, and project for external users.
3. PROBLEMS ENCOUNTERED Not many issues in this part.
4. SELF EVALUATION OF THE PROGRESS As per previous reports, this week there's quite a lot of refactoring, and maintenance, to make it stable to be used by external users. At this point the development is pretty much done for FYP2, but there's interest in developing it further.

Alex

Supervisor's signature

[illegible]

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT 7

(Project I)

Trimester, Year: T1Y3	**Study week no.:** 9
Student Name & ID: Yii Kuo Chong (2102670)	
Supervisor: Ooi Boon Yaik	
Project Title: Data Visualisation for Text-based Document	

1. WORK DONE

Aug 14, 2023
- Properly read dates from xlsx
- Fixes some issues
- Update README.md
- Update README.md

Aug 16, 2023
- Added currentX to store

Aug 17, 2023
- Fix linting issues, and update installation instruction
- Refactor Charts to be more simple, and refactor ws implementations
- Refactor WebSockets implementation. Fix issue with data not sent to client in production. Renamed fixture files to be more clear.

Aug 18, 2023
- Update run_test.yml

Aug 20, 2023
- Remove yAxis min setting
- Update README
- Fixed typo in README
- Update README

2. WORK TO BE DONE
- Writing the report
  - Chapter 1
  - Chapter 2
  - Chapter 3
  - Chapter 4
  - Chapter 5
  - Chapter 6
  - Acknowledgement
  - Abstract
  - List of Figures
  - List of Tables
  - List of Abbreviations
  - Declaration of Originality
- Poster
- Presentation Slides

3. PROBLEMS ENCOUNTERED

No Comment

4. SELF EVALUATION OF THE PROGRESS
Preparing the project for external users. Decent progress.

Alex	[illegible]
Supervisor's signature	Student's signature

FINAL YEAR PROJECT WEEKLY REPORT 8

(Project I)

Trimester, Year: T1Y3	**Study week no.:** 11 & 10
Student Name & ID: Yii Kuo Chong (2102670)	
Supervisor: Ooi Boon Yaik	
Project Title: Data Visualisation for Text-based Document	

1. WORK DONE

Aug 24, 2023

- Finished the Declaration of Originality
- Finished the Acknowledgement
- Finished the Abstract
- Finished the Chapter 1

Aug 26, 2023

- Finished Chapter 3

Aug 28, 2023

- Starting to write Chapter 4

2. WORK TO BE DONE

- Writing the report
    - Chapter 2
    - Chapter 6
    - List of Figures
    - List of Tables
    - List of Abbreviations
- Poster
- Presentation Slides

3. PROBLEMS ENCOUNTERED

No Comment

4. SELF EVALUATION OF THE PROGRESS

A bit tight in terms of time. Decent progress.

[illegible]

Supervisor's signature

[illegible]

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT 9

(Project I)

Trimester, Year: T1Y3	**Study week no.:** 12
Student Name & ID: Yii Kuo Chong (2102670)	
Supervisor: Ooi Boon Yaik	
Project Title: Data Visualisation for Text-based Document	

1. WORK DONE

Sep 08, 2023
- Completed Chapter 5.1
- Completed Chapter 5.2
- Completed Chapter 5.3
- Completed Chapter 5.5

Sep 09, 2023
- Completed Chapter 5.4

Sep 10, 2023
- Continuing to write Chapter 4

Sep 12, 2023
- Completing Chapter 4
- Preparing weekly work report
- Completed Poster

Sep 13, 2023
- Continuing working on Chapter 2

Sep 14, 2023
- Finished List of Figures
- Finished List of Tables
- Finished List of Abbreviations
- Finished Chapter 2
- Finished Chapter 6
- Added references
- Completed final checks before sending in for supervisor review

2. WORK TO BE DONE
No Comment

3. PROBLEMS ENCOUNTERED
No Comment

4. SELF EVALUATION OF THE PROGRESS
A bit tight in terms of time. Decent progress.

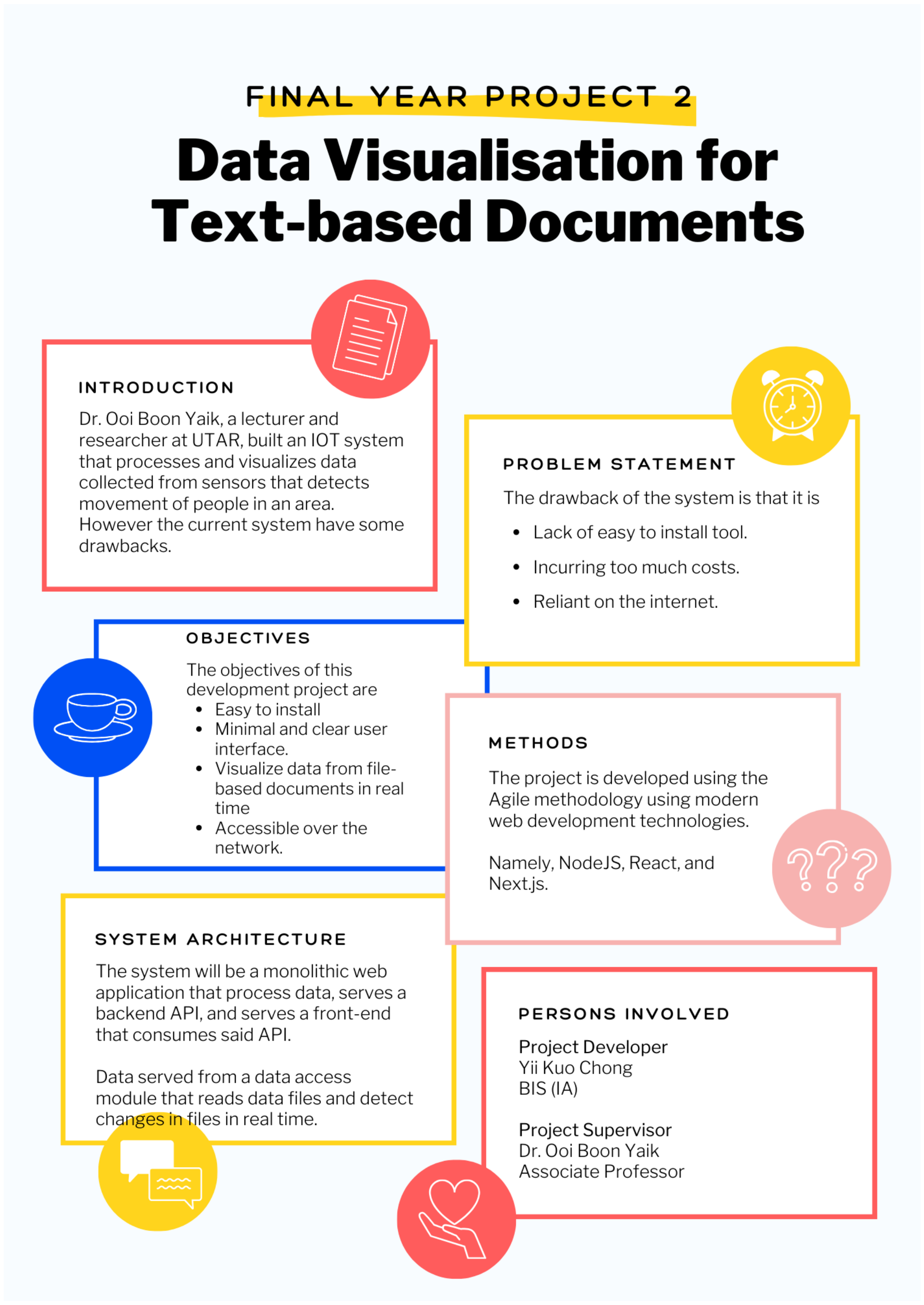
Alex

Supervisor's signature

[illegible]

Student's signature

POSTER



FINAL YEAR PROJECT 2

Data Visualisation for Text-based Documents

INTRODUCTION

Dr. Ooi Boon Yaik, a lecturer and researcher at UTAR, built an IOT system that processes and visualizes data collected from sensors that detects movement of people in an area. However the current system have some drawbacks.

PROBLEM STATEMENT

The drawback of the system is that it is

- Lack of easy to install tool.
- Incurring too much costs.
- Reliant on the internet.

OBJECTIVES

The objectives of this development project are

- Easy to install
- Minimal and clear user interface.
- Visualize data from file-based documents in real time
- Accessible over the network.

METHODS

The project is developed using the Agile methodology using modern web development technologies.

Namely, NodeJS, React, and Next.js.

SYSTEM ARCHITECTURE

The system will be a monolithic web application that process data, serves a backend API, and serves a front-end that consumes said API.

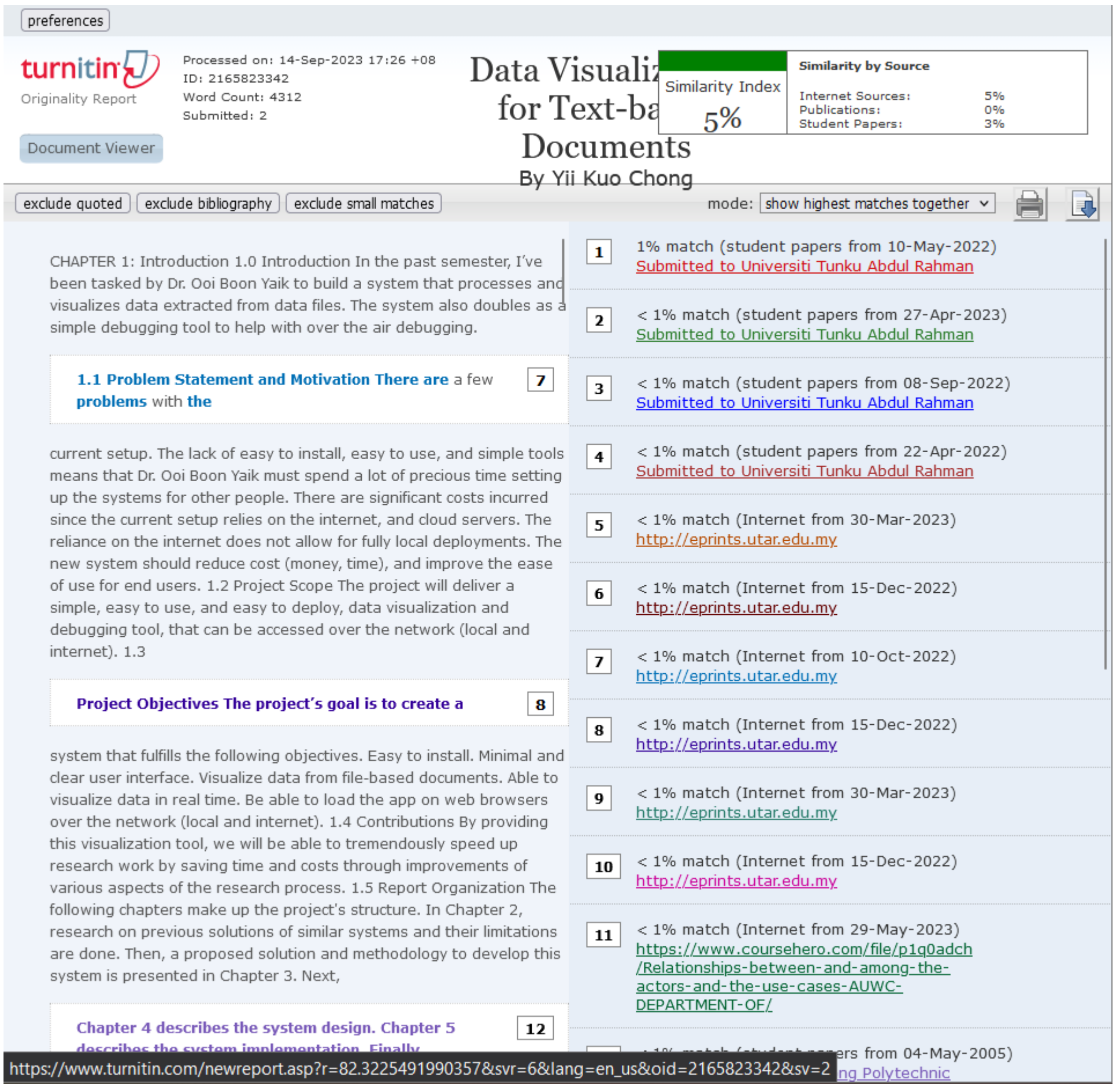
Data served from a data access module that reads data files and detect changes in files in real time.

PERSONS INVOLVED

Project Developer
Yii Kuo Chong
BIS (IA)

Project Supervisor
Dr. Ooi Boon Yaik
Associate Professor

PLAGIARISM CHECK RESULT


preferences

turnitin Originality Report

Processed on: 14-Sep-2023 17:26 +08
ID: 2165823342
Word Count: 4312
Submitted: 2

Data Visuali
for Text-ba
Documents
By Yii Kuo Chong

Similarity Index 5%

Similarity by Source
Internet Sources: 5%
Publications: 0%
Student Papers: 3%

Document Viewer

exclude quoted | exclude bibliography | exclude small matches | mode: show highest matches together

CHAPTER 1: Introduction 1.0 Introduction In the past semester, I've been tasked by Dr. Ooi Boon Yaik to build a system that processes and visualizes data extracted from data files. The system also doubles as a simple debugging tool to help with over the air debugging.

1.1 Problem Statement and Motivation There are a few **problems** with **the** 7

current setup. The lack of easy to install, easy to use, and simple tools means that Dr. Ooi Boon Yaik must spend a lot of precious time setting up the systems for other people. There are significant costs incurred since the current setup relies on the internet, and cloud servers. The reliance on the internet does not allow for fully local deployments. The new system should reduce cost (money, time), and improve the ease of use for end users. 1.2 Project Scope The project will deliver a simple, easy to use, and easy to deploy, data visualization and debugging tool, that can be accessed over the network (local and internet). 1.3

Project Objectives The project's goal is to create a 8

system that fulfills the following objectives. Easy to install. Minimal and clear user interface. Visualize data from file-based documents. Able to visualize data in real time. Be able to load the app on web browsers over the network (local and internet). 1.4 Contributions By providing this visualization tool, we will be able to tremendously speed up research work by saving time and costs through improvements of various aspects of the research process. 1.5 Report Organization The following chapters make up the project's structure. In Chapter 2, research on previous solutions of similar systems and their limitations are done. Then, a proposed solution and methodology to develop this system is presented in Chapter 3. Next,

Chapter 4 describes the system design. Chapter 5 12
describes the system implementation. Finally

1	1% match (student papers from 10-May-2022) Submitted to Universiti Tunku Abdul Rahman
2	< 1% match (student papers from 27-Apr-2023) Submitted to Universiti Tunku Abdul Rahman
3	< 1% match (student papers from 08-Sep-2022) Submitted to Universiti Tunku Abdul Rahman
4	< 1% match (student papers from 22-Apr-2022) Submitted to Universiti Tunku Abdul Rahman
5	< 1% match (Internet from 30-Mar-2023) http://eprints.utar.edu.my
6	< 1% match (Internet from 15-Dec-2022) http://eprints.utar.edu.my
7	< 1% match (Internet from 10-Oct-2022) http://eprints.utar.edu.my
8	< 1% match (Internet from 15-Dec-2022) http://eprints.utar.edu.my
9	< 1% match (Internet from 30-Mar-2023) http://eprints.utar.edu.my
10	< 1% match (Internet from 15-Dec-2022) http://eprints.utar.edu.my
11	< 1% match (Internet from 29-May-2023) https://www.coursehero.com/file/p1q0adch/Relationships-between-and-among-the-actors-and-the-use-cases-AUWC-DEPARTMENT-OF/
	< 1% match (student papers from 04-May-2005) ng Polytechnic

https://www.turnitin.com/newreport.asp?r=82.3225491990357&svr=6&lang=en_us&oid=2165823342&sv=2

Universiti Tunku Abdul Rahman			
Form Title: Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1of 1

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	Yii Kuo Chong
ID Number(s)	2102670
Programme / Course	IA
Title of Final Year Project	Data visualisation for text-based documents

Similarity	**Supervisor's Comments (Compulsory if parameters of originality exceed the limits approved by UTAR)**
Overall similarity index: 7 % Similarity by source Internet Sources: 5 % Publications: 3 % Student Papers: 4 %	
Number of individual sources listed of more than 3% similarity: 0	
Parameters of originality required, and limits approved by UTAR are as Follows: **(i) Overall similarity index is 20% and below, and** **(ii) Matching of individual sources listed must be less than 3% each, and** **(iii) Matching texts in continuous block must not exceed 8 words** ***Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.***	

<u>Note:</u> Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

Signature of Supervisor [illegible] Name: ______________		Signature of Co-Supervisor Name: ______________
Date: 15/9/2023		Date: ______________

FYP 2 CHECKLIST

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student ID	2102670
Student Name	Yii Kuo Chong
Supervisor Name	Ooi Boon Yaik

TICK (√)	**DOCUMENT ITEMS** Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
√	Title Page
√	Signed form of the Declaration of Originality
√	Acknowledgment
√	Abstract
√	Table of Contents
√	List of Figures (if applicable)
√	List of Tables (if applicable)
-	List of Symbols (if applicable)
√	List of Abbreviations (if applicable)
√	Chapters / Content
√	Bibliography (or References)
√	All references in bibliography are cited in the thesis, especially in the chapter of literature review
√	Appendices (if applicable)
√	Poster
√	Signed Turnitin Report (Plagiarism Check Result – Form Number: FM-IAD-005)
√	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report.

[illegible]

____

(Signature of Student)

Date: 14th September 2023