S4585727

INFS3202 Design Document

MechanicAssist

Daniel Ciccotosto-Camp
Due 8-4-2024

Contents

Overview

Invoicing can become a burdensome, time-consuming and costly task for small businesses. This can be attributed to the diverse and often complex nature of servicing and repairs provided, which require detailed and customised invoicing. Mechanics may find it time-consuming to create invoices manually, especially for services that vary in complexity or require detailed itemisation. Additionally, the need to maintain accurate records for warranty claims, customer queries, and financial reporting add to the complexity of invoicing in the automotive industry. MechanicAssist aims to address these challenges by offering a platform where customers can choose between existing invoicing templates or utilise Large Language Models (LLM) to generate customised invoice items tailored to the specific requirements of their repairs or services.

The target demographic for MechanicAssist includes mechanics and automotive businesses that require invoicing services for vehicle repairs and maintenance. These users can benefit from MechanicAssist's customisation options, which can help streamline their invoicing processes and improve efficiency. As MechanicAssist evolves, there is potential to generalise its application for use in other industries beyond automotive. By expanding its features, such as integrating inventory management and scheduling tools, MechanicAssist could become a comprehensive business operations tool for various service-based industries.

MechanicAssist will be developed using Angular 17 for the frontend and a Python backend with Flask RESTful controllers. Angular was chosen for its ability to create dynamic and responsive user interfaces, which are essential for a modern invoicing application. Python and Flask were chosen for the backend due to their flexibility and ease of use, which will allow for integration with the frontend (outlined further in *Technology Research)*. Additionally, a MySQL server will be used for database management, ensuring security and integrity of invoicing data.

The target demographic for MechanicAssist includes mechanics and automotive businesses that require invoicing services for vehicle repairs and maintenance. These users can benefit from MechanicAssist's user-friendly interface and customization options, which can help streamline their invoicing processes and improve efficiency.

Key Features

MechanicAssist includes several key features designed to streamline and enhance the invoicing process for mechanics:

- **Invoicing with Generative AI**
  MechanicAssist offers a versatile invoicing system where users can choose to import templates, create custom invoice items, or utilise Generative AI for automatic generation. Users can create reusable "standard templates" for common services, outlining service descriptions, charges, and labour costs through via a csv import.
  The Generative AI option provides a prompt-based approach; for example, typing 'timing belt replacement,' will generate a custom invoice item outlining the task and materials needed in the service. Due to the changing demand and supplies of material and labour, it will not predict and associated service charge due to the volatility of parts and labour. After the invoicing item has been generated, the free-text description field remains editable for transparency and adaptability.

- **Templating Creation:**
  Users have the ability to import standard templates onto MechanicAssist via an imported csv file. During the creation of a new invoice, users have the option to add an invoice item from their previously imported templates. Once imported, all templates can be viewed on the templating screen with the option to delete.

- **Invoicing Search and View**

  After the invoice has been created, administrators of the MechanicAssist application will have the ability to search previously created invoices in their system. Once they have found the desired card, the user will be able to be directed to a separate page to view in full-screen mode.

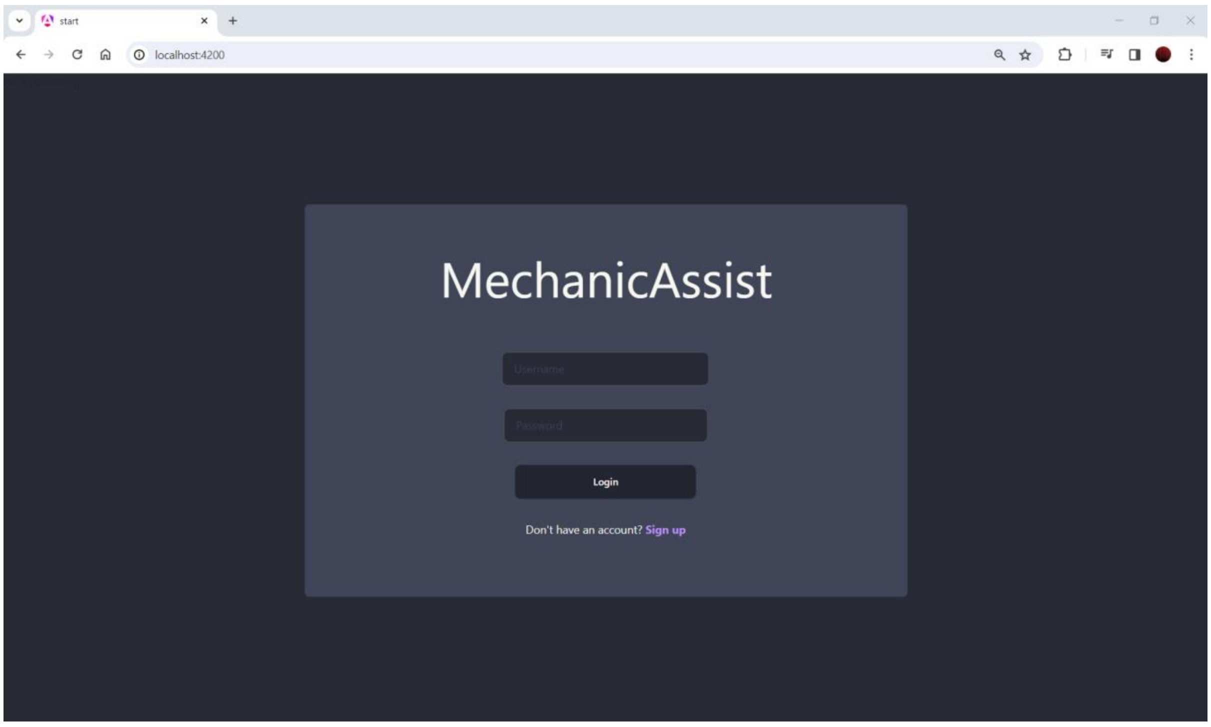
- **Servicing Dashboard**
  The application includes a comprehensive servicing dashboard that provides users with valuable insights into their servicing operations. Users can view charts outlining total servicing and labour costs, as well as track the frequency of use for standard templates. The dashboard also features a bar chart displaying weekly turnover and the proportion of paid versus unpaid invoices, helping users track their financial performance.

- **Payment Tracking**
  MechanicAssist includes a payment tracking feature that allows users to easily mark invoices as paid. Users can simply click a checkbox in the invoice overview to indicate that an invoice has been paid, updating the financials on the main dashboard of the application. This feature helps users keep track of outstanding payments and manage their finances more effectively. In this same section, users can click on a previously created invoice card to view the full invoice in a view mode only.

Mockups

Please note: in all of the below mockups, the purple button ‘Open Drawer’ will be removed for the final design. It has been placed as a temporary measure to navigate between pages only.

Landing Page:


start

localhost:4200

MechanicAssist

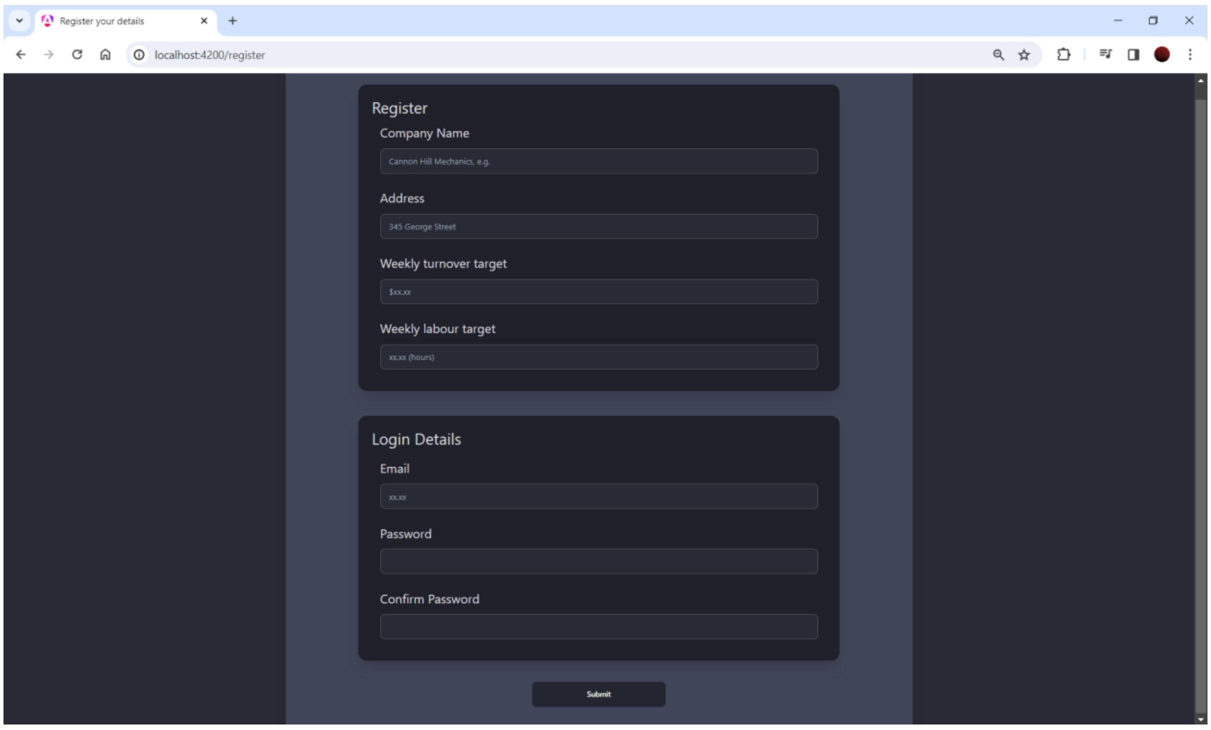
Username

Password

Login

Don't have an account? Sign up

Register details (zoom 67%):


Register your details

localhost:4200/register

Register

Company Name

Cannon Hill Mechanics, e.g.

Address

345 George Street

Weekly turnover target

$xx.xx

Weekly labour target

xx.xx (hours)

Login Details

Email

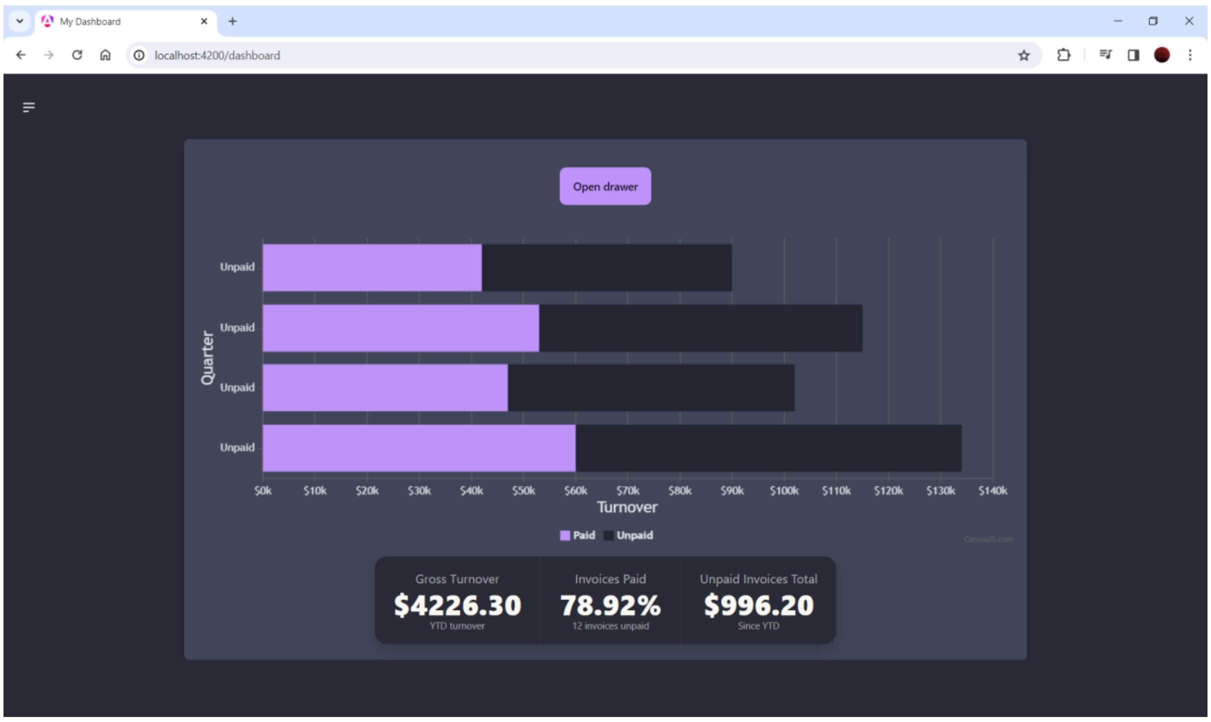
xx.xx

Password

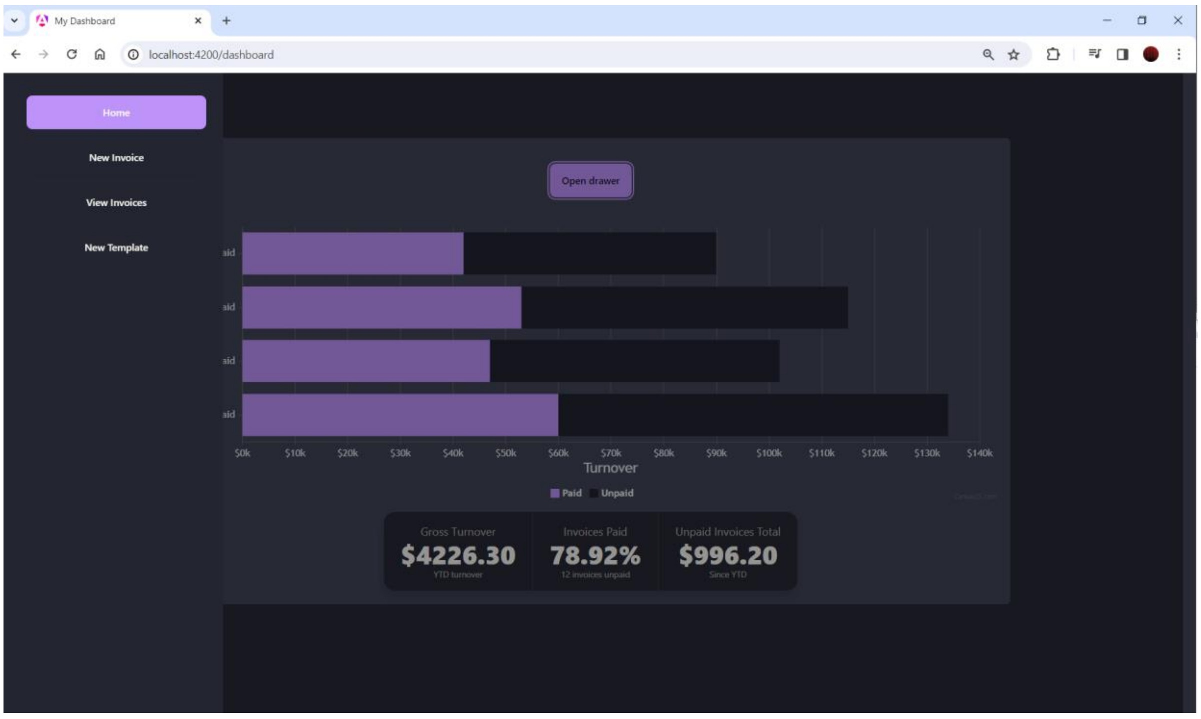
Confirm Password

Submit

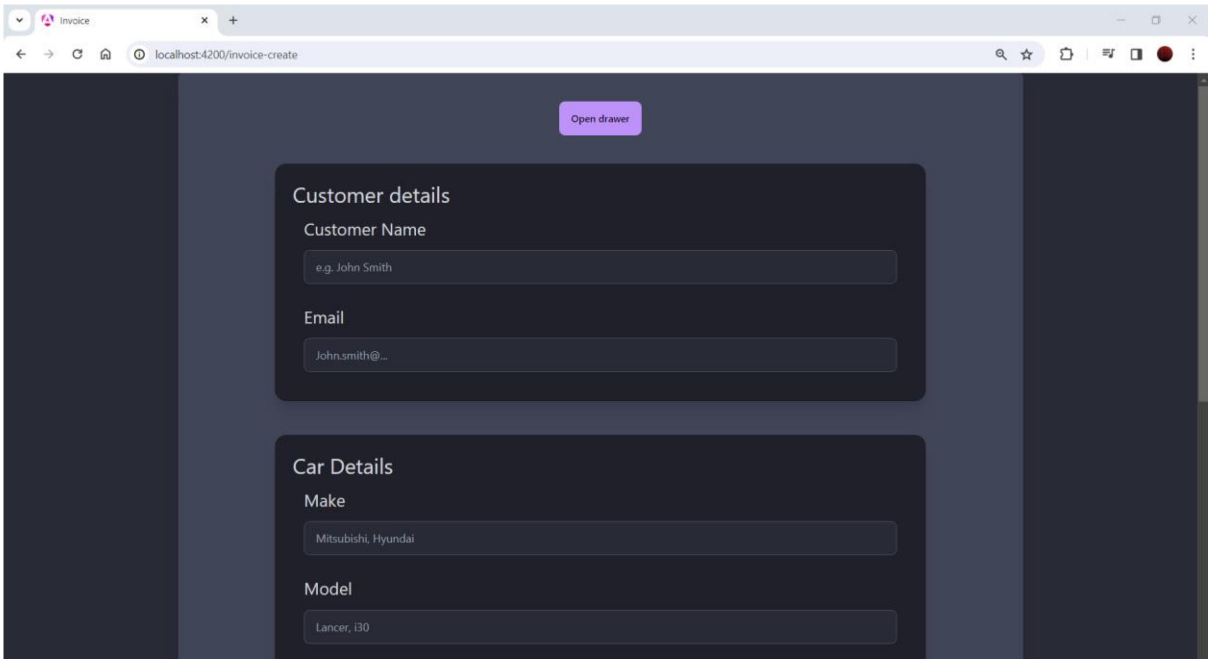
Dashboard:



Dashboard – Open drawer:



Invoicing (create, select line item to edit):



Customer details

Customer Name

e.g. John Smith

Email

John.smith@...

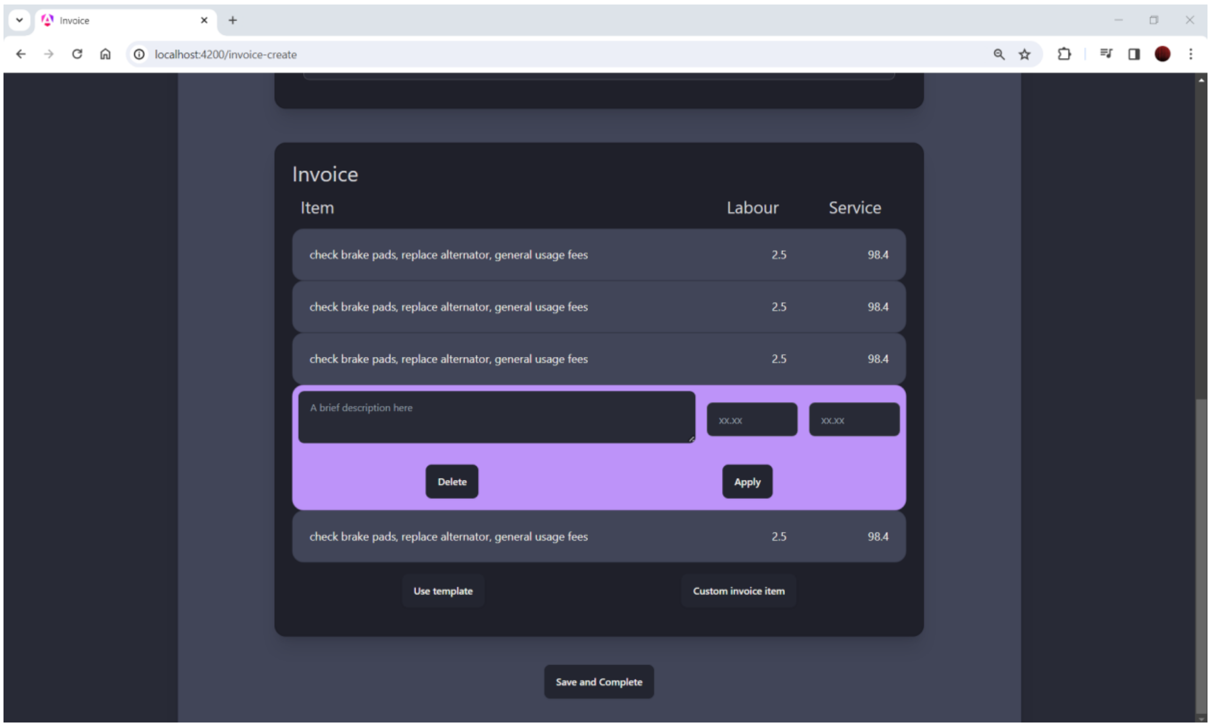
Car Details

Make

Mitsubishi, Hyundai

Model

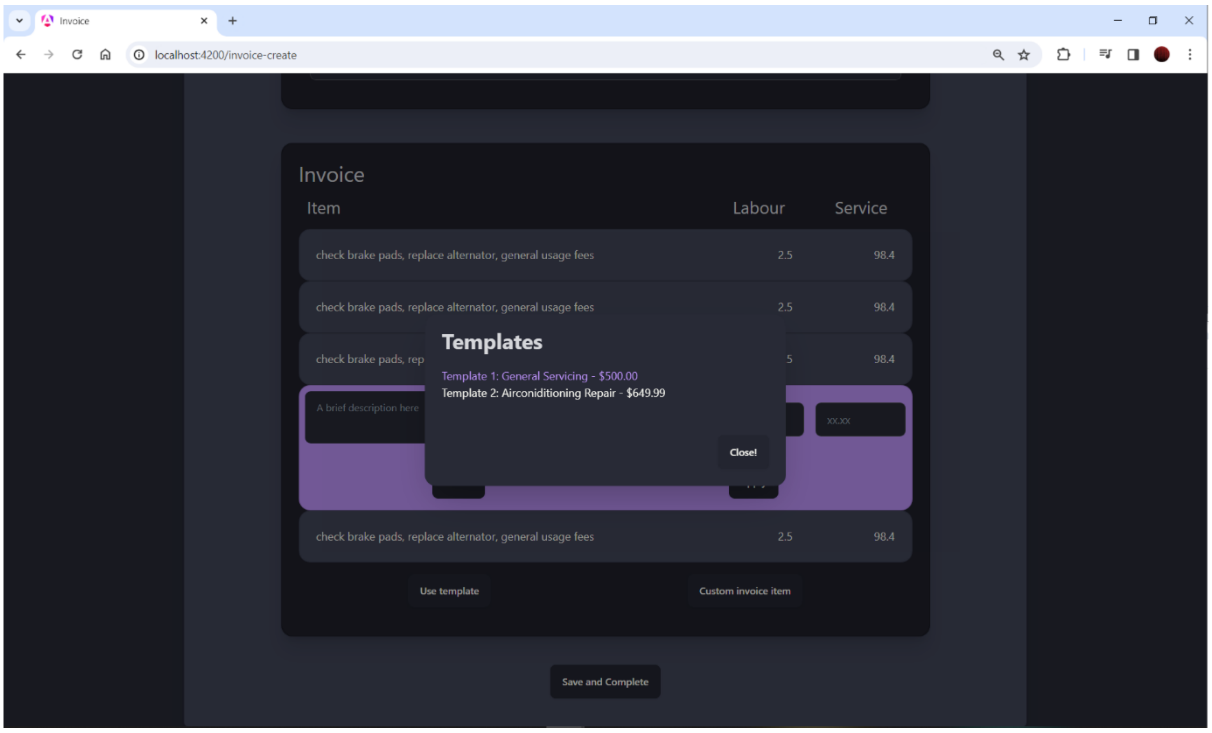
Lancer, i30



Invoice

Item	Labour	Service
check brake pads, replace alternator, general usage fees	2.5	98.4
check brake pads, replace alternator, general usage fees	2.5	98.4
check brake pads, replace alternator, general usage fees	2.5	98.4
A brief description here	xx.xx	xx.xx
check brake pads, replace alternator, general usage fees	2.5	98.4

Invoice: choose template



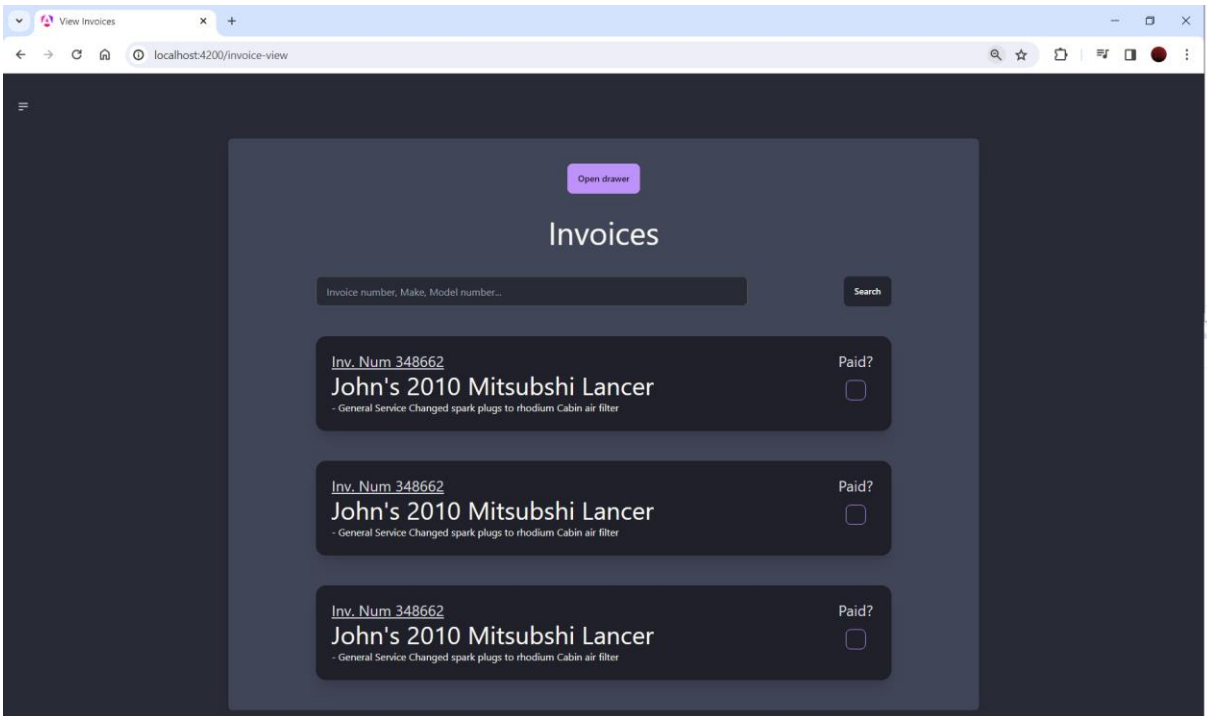
Invoice

Item	Labour	Service
check brake pads, replace alternator, general usage fees	2.5	98.4
check brake pads, replace alternator, general usage fees	2.5	98.4
check brake pads, rep	5	98.4
check brake pads, replace alternator, general usage fees	2.5	98.4

Templates

Template 1: General Servicing - $500.00
Template 2: Airconiditioning Repair - $649.99

Invoice (view)



Invoices

Invoice number, Make, Model number...

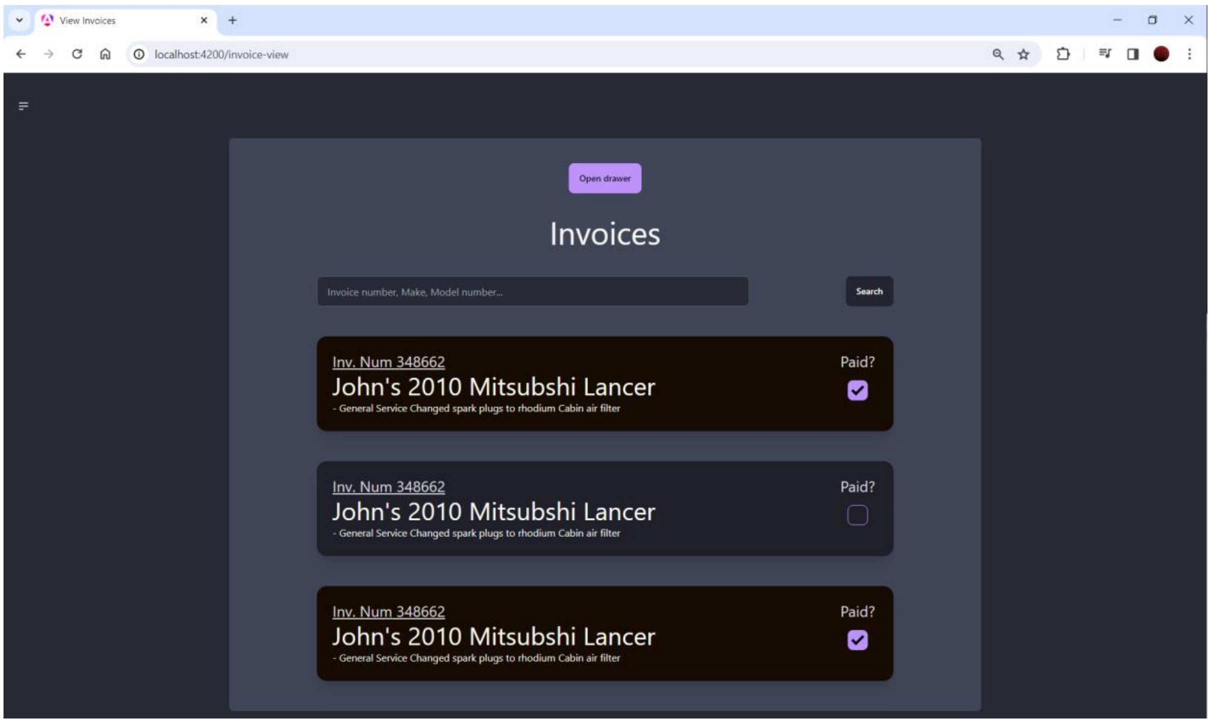
Search

Inv. Num 348662
John's 2010 Mitsubshi Lancer
- General Service Changed spark plugs to rhodium Cabin air filter
Paid? [ ]

Inv. Num 348662
John's 2010 Mitsubshi Lancer
- General Service Changed spark plugs to rhodium Cabin air filter
Paid? [ ]

Inv. Num 348662
John's 2010 Mitsubshi Lancer
- General Service Changed spark plugs to rhodium Cabin air filter
Paid? [ ]

Invoice (view, select as paid)



Invoices

Invoice number, Make, Model number...

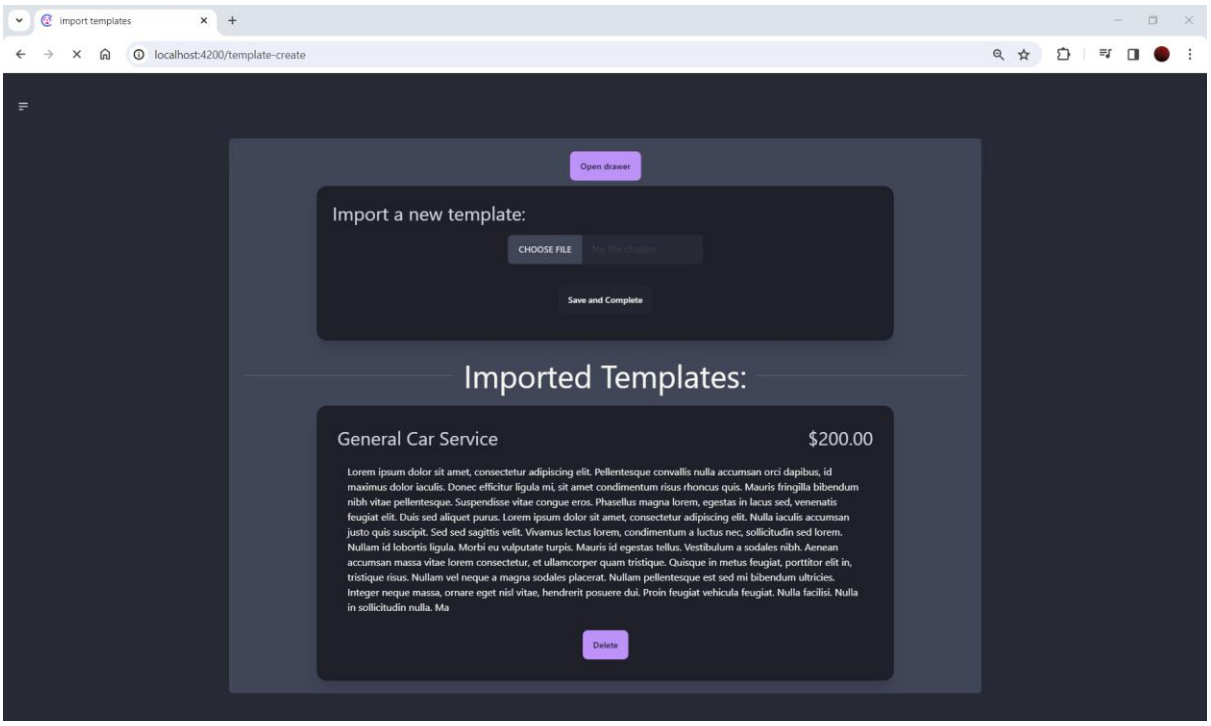
Search

Inv. Num 348662
John's 2010 Mitsubshi Lancer
- General Service Changed spark plugs to rhodium Cabin air filter
Paid? [✓]

Inv. Num 348662
John's 2010 Mitsubshi Lancer
- General Service Changed spark plugs to rhodium Cabin air filter
Paid? [ ]

Inv. Num 348662
John's 2010 Mitsubshi Lancer
- General Service Changed spark plugs to rhodium Cabin air filter
Paid? [✓]

Templates (import):



Import a new template:

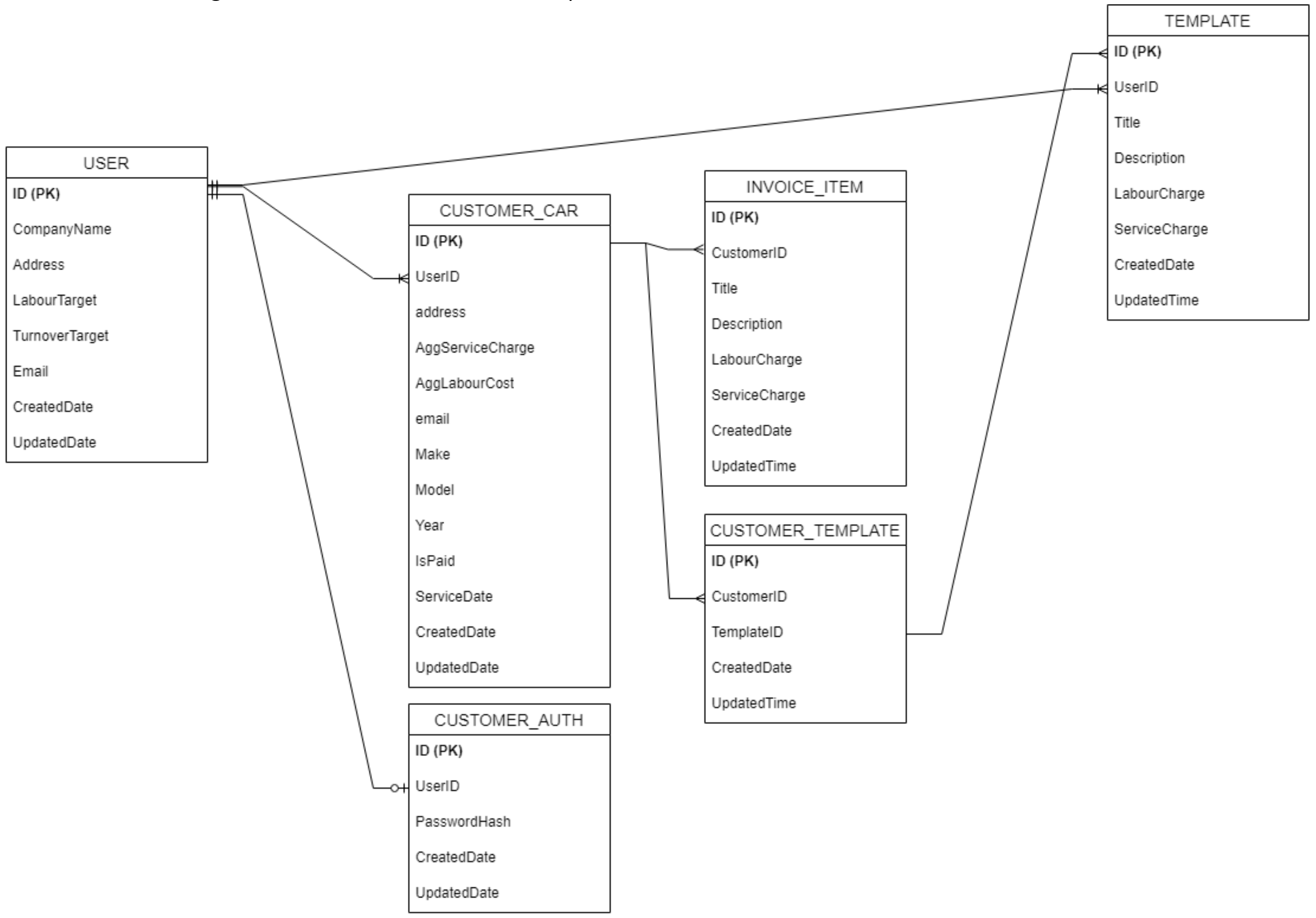
Imported Templates:

General Car Service $200.00

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque convallis nulla accumsan orci dapibus, id maximus dolor iaculis. Donec efficitur ligula mi, sit amet condimentum risus rhoncus quis. Mauris fringilla bibendum nibh vitae pellentesque. Suspendisse vitae congue eros. Phasellus magna lorem, egestas in lacus sed, venenatis feugiat elit. Duis sed aliquet purus. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla iaculis accumsan justo quis suscipit. Sed sed sagittis velit. Vivamus lectus lorem, condimentum a luctus nec, sollicitudin sed lorem. Nullam id lobortis ligula. Morbi eu vulputate turpis. Mauris id egestas tellus. Vestibulum a sodales nibh. Aenean accumsan massa vitae lorem consectetur, et ullamcorper quam tristique. Quisque in metus feugiat, porttitor elit in, tristique risus. Nullam vel neque a magna sodales placerat. Nullam pellentesque est sed mi bibendum ultricies. Integer neque massa, ornare eget nisl vitae, hendrerit posuere dui. Proin feugiat vehicula feugiat. Nulla facilisi. Nulla in sollicitudin nulla. Ma

Database Design

The database design schema has been centred on the user (administrator) of the account. The user creates invoices of their customer car details, which consist of invoice line items custom generated or extracted from a template



Technology Research

Large Language Model Choice:

Much research was taken in a separate course (DECO3801) to determine the best choice of Large Language Models, and the improvements in respective technologies have not changed significantly since research was originally undertaken. To put succinctly, the ChatGPT API was chosen over Google's BERT or Azure's AI Playground due to the API's versatility and ease of integration into small applications. Additionally, ChatGPT's continuous improvement to their trained models make it a reliable and future-proof choice for further implementation or evolution of this project.

Choice of Backend Architecture

As previously outlined, the project decision favoured Angular 17 for its modern architectures and its ability to create dynamic and responsive user interfaces. Utilising Angular 17 with Server Side Rendering (SSR) necessitates a secondary 'Backend-For-Frontend' (BFF) interface to manage Model CRUD operations and business logic. This requires setting up respective APIs in both the frontend (Angular) and BFF systems through the use of RESTful controllers.

Three primary contenders were considered for implementing the BFF for the Angular 17 application were Python, TypeScript Node.js, and Java Spring Boot applications. During the research process, various architectural components were evaluated, including the ability to create RESTful controllers, Object Relational Mapping (ORM) for CRUD operations, and integration with external libraries for complex implementations such as LLMs. The following paragraphs outline the advantages of each BFF application type, as well as the potential concerns and challenges of implementing each.

Python:

Python has well-supported Object-Relational Mapping (ORM) and API interfaces such as SQLAlchemy for ORM and Flask for REST API development. However, adopting a disciplined approach to Model-View-Controller (MVC) architecture can be challenging due to Python's flexibility in class and object creation. On the deployment front, Python is the easiest to configure for UQCloud. It also offers the easiest architecture setup for integrating the OpenAI API.

Java/Spring Boot Application:

Java provides strong support for ORM, with tools like Hibernate and native API configurations which simplify the development of CRUD operations (Hibernate, 2024). Spring Boot, designed to enforce the Model-View-Controller (MVC) framework, can help in organizing operations efficiently (Baeldung, 2024) (spring, 2024).

Java, particularly with Spring Boot, presents concerns regarding the amount of configuration required, especially in areas like ORM configuration and managing the Spring Boot starter (Baeldung, 2024). There are also concerns about the difficulty of deploying Java applications onto the UQCloud virtual environment.

The biggest concern with Java/Spring Boot is the support for the OpenAI platform. While the Azure OpenAI client library for Java is the best-supported Java client for dealing with OpenAI API requests, its implementation is complex. Other frameworks are simpler to use, but the OpenAI API changes frequently, making it challenging for third-party libraries to stay updated.

Node.js:

For Node.js, there is a steep learning curve, especially since I have never programmed in JavaScript before. However, it would be the most sensible option considering the heavy TypeScript dependence of the Angular frontend, and using a Node.js BFF this would ensure that both the backend and frontend systems have similar characteristics. The main challenge with Node.js is my unfamiliarity with supported libraries for developing API controllers and ORM tools to simplify CRUD operations (Prisma, 2024).

Conclusion:

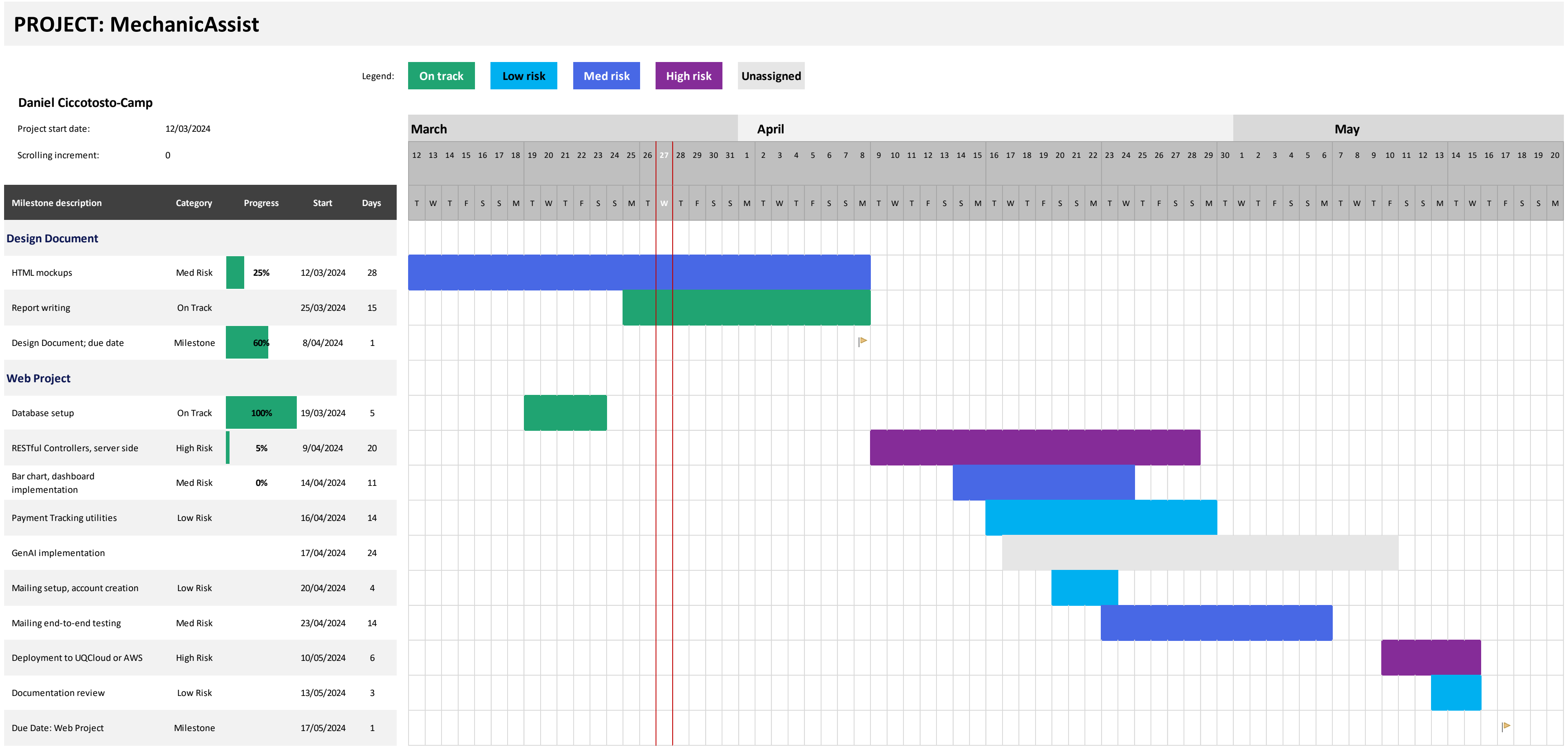
After consideration of the strengths and challenges of each option, Python was decided to the best choice of Backend-For-Frontend (BFF) system for the Angular 17 application. Its strong support for ORM and API interfaces, along with its ease of deployment on UQCloud, made it the preferred choice for MechanicAssist.

Timeline

The Gantt chart outlined in figure 1 below outlines the project development timeline of the MechanicAssist application. For development purposes, it was decided to breakdown the application into manageable ‘milestones’, given but separate subheading in their respective assessment item subheadings.

The Gantt chart outlines some features, namely:

- Category, comprised of ‘On track’, ‘Low Risk’ ‘Medium Risk’ and ‘High risk’, based on the difficulty of the implementation and the risk it poses to dependent implementation
- Start and date, outlining the start date and intended completion date of the feature.
- Progress, indicative of the approximate progress of the application completed to date.



PROJECT: MechanicAssist

Legend: On track | Low risk | Med risk | High risk | Unassigned

Daniel Ciccotosto-Camp

Project start date: 12/03/2024

Scrolling increment: 0

Milestone description	Category	Progress	Start	Days
Design Document				
HTML mockups	Med Risk	25%	12/03/2024	28
Report writing	On Track		25/03/2024	15
Design Document; due date	Milestone	60%	8/04/2024	1
Web Project				
Database setup	On Track	100%	19/03/2024	5
RESTful Controllers, server side	High Risk	5%	9/04/2024	20
Bar chart, dashboard implementation	Med Risk	0%	14/04/2024	11
Payment Tracking utilities	Low Risk		16/04/2024	14
GenAI implementation			17/04/2024	24
Mailing setup, account creation	Low Risk		20/04/2024	4
Mailing end-to-end testing	Med Risk		23/04/2024	14
Deployment to UQCloud or AWS	High Risk		10/05/2024	6
Documentation review	Low Risk		13/05/2024	3
Due Date: Web Project	Milestone		17/05/2024	1

References and uses of GenAI

ChatGPT:

1. I am creating a web application called MechanicAssist that aims to automate and simplify invoicing workflows for invoicing for mechanics in repairs/maintenance. With more implementation, what markets could I potentially expand and market this application to?
    - *Overview,* second paragraph
2. Can you reword this paragraph to remove personal pronouns
    - Technology research, subsection *Python*
3. Could you say that python has 'looser coupling' with classes and functions?
    - Technology research, subsection *Python*
4. Would it be appropriate to say that a system serving API's for an angular app would be called a backend for frontend?
    - *Technology research*

References

Baeldung. (2024, 04 07). *Intro to Spring Boot Starters*. Retrieved from Baeldung: https://www.baeldung.com/spring-boot-starters

Hibernate. (2024, 04 07). *Hibernate ORM*. Retrieved from Hibernate: https://hibernate.org/orm/what-is-an-orm/

Microsoft. (2024, 04 03). *Azure OpenAI client library for Java - version 1.0.0-beta.7*. Retrieved from Microsoft Learn: https://learn.microsoft.com/en-us/java/api/overview/azure/ai-openai-readme?view=azure-java-preview

Open AI Playground. (2024, 04 07). *ChatGPT 3.5*. Retrieved from chat.openai.com: https://chat.openai.com/

Prisma. (2024, 04 07). *Quickstart*. Retrieved from Prisma: https://www.prisma.io/docs/getting-started/quickstart#1-create-typescript-project-and-set-up-prisma-orm

spring. (2024, 04 07). *Serving Web Content with Spring MVC*. Retrieved from Spring by VMware Tnazu: https://spring.io/guides/gs/serving-web-content