



INDIVIDUAL ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

CT027-3-3-EPDA

ENTERPRISE PROGRAMMING FOR DISTRIBUTED APPLICATION

NAME: WONG HORNG WOEI

TP NUMBER: TP055241

INTAKE: APD3F2211CS(IS)

HAND OUT DATE: 12/12/2022

HAND IN DATE: 10/03/2023

LECTURER: MR. KAU GUAN KIAT

Table of Contents

1.0 Part One	6
1.1 Brief history on distributed computing and architectural evolution of distributed computing	6
1.1.1 Discussion on architectural evolution of distributed computing	9
1.2 Overview of types of enterprise application and architecture	16
1.2.1 Customer Relationship Management (CRM) Software.....	16
1.2.2 Enterprise Resource Planning (ERP) Systems.....	17
1.2.3 Supply Chain Management (SCM) Systems	18
1.2.4 Client-side Architecture	19
1.2.5 Microservices Architecture.....	22
1.2.6 Service-oriented Architecture	24
1.3 Differentiate Customer Relationship Management Software, Enterprise Resource Planning Systems, Supply Chain Management Systems.	25
1.4 Explore and identify APU Systems to implement suitable architecture.....	32
1.5 Improve another system similar with chosen system	35
2.0 Part Two.....	42
2.1 Design of Web Components	42
2.1.1 JSP.....	42
2.1.2 Servlet	46
2.2 User Manual.....	49
2.2.1 Managing Staff.....	53
2.2.2 Salesman	72
2.2.3 Customer	83
2.2.4 General Navigation Chart (Sitemap)	90
3.0 Part 3	91
ASIA PACIFIC UNIVERSITY	2

3.1 Overview of application and brief description of the system architecture	91
3.2 Design of Business Tier	94
3.2.1 Session bean	94
3.2.2 Message-driven bean	95
3.2.3 Entity bean	96
3.3 UML Diagrams (Use Case & Class).....	97
3.3.1 Use Case Diagram.....	97
3.3.2 Class Diagram.....	98
4.0 Part 4	99
4.1 Database Design.....	99
4.1.1 Entity Relationship Diagram.....	105
4.1.2 Design of Database access APIs	106
5.0 Description and evidence of additional features which have been incorporated in the solution	117
5.1 Bootstrap Framework.....	117
5.2 Cascading Style Sheets (CSS)	119
5.3 JavaScript (JS)	121
5.4 Charts	123
5.5 Transaction Report for Managing Staff	124
5.6 Set Current Date.....	125
5.7 Lombok library	126
6.0 References.....	127

Assumption

The web-based car sales system consists of 3 user types: Managing Staff, Customer, Salesman. The system has implemented with full validations to ensure that user to input the fields whenever is required. All 3 user types are required to login to use the system. The following will show the functionalities that has been implemented for managing staff dashboard, salesman dashboard, customer dashboard:

Managing Staff:

- Report and analysis that consists of 5 charts (Payment Analysis Report, Feedback Analysis Report, Gender Analysis Report, Customer and Salesman Analysis Report, Inventory Analysis Report).
- A table to view Managing Staff records, the table also has the function to add, search, edit, and delete the staff.
- A table to view Salesman records, the table also has the function to approve, search, edit, and delete the salesman.
- A table to view Customer records, the table also has the function to approve, search, edit, and delete the customer.
- A table to view Car records, the table also has the function to add, search, edit, and delete the car.
- A transaction sales report that has been added as a modal to show all transaction records throughout the car sales system.
- A logout feature for managing staff to logout.

Salesman:

- Able to register through the homepage, then will have to wait for Managing Staff's approval.
- Able to edit individual profile such as name, age, email, contact information, gender.
- A table view available car, salesman can proceed to book the car for customer. A button is implemented solely for "Cancelled" status car for salesman to make them available once again.
- Booking modal asked for customer name salesman can input comment for the sales.
- A table to view sales record, salesman can proceed to do payment for the customer. As well as cancel the booking if needed.
- A sales report table that has been added as a modal to show individual sales records.
- A logout feature for staff to logout.

Customer:

- Able to register through the homepage, then will have to wait for Managing Staff's approval.
- Able to edit individual profile such as name, age, email, contact information, gender.
- A table to view pending transaction about the car that customer has booked.
- A table to view purchase history about the car that customer has purchased.
- A table to view feedback and give feedback and rating once on the car purchased.
- A logout feature for customer to logout.

1.0 Part One

1.1 Brief history on distributed computing and architectural evolution of distributed computing

Distributed computing is a field of computer science that involves the distribution of computing tasks across a network of computers (Amazon Web Services, Inc., 2023b). It allows for the processing of large amounts of data and the execution of complex computations by dividing the work among multiple computers (IBM Corporation, 2021).

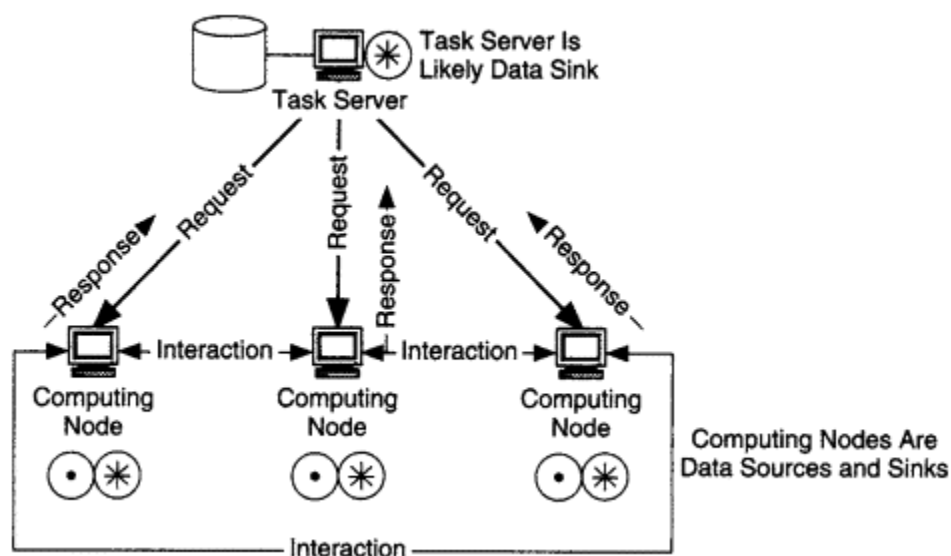


Figure 1: Distributed Computing Flow Model (McCabe, 2007).

This allows for the efficient use of resources and the ability to handle larger workloads than would be possible with a single computer (Amazon Web Services, Inc., 2023b). According to Amazon Web Services (2023b), distributed computing "refers to the practice of dividing a large problem into smaller problems, solving them concurrently, and aggregating the results." IBM defines distributed computing as "the process of dividing an application into separate components that can be executed concurrently across a distributed system, such as a network of computers or a cloud computing environment." (IBM Corporation, 2021). Both definitions emphasize the use of multiple computers or devices to perform tasks concurrently, with the goal of improving efficiency and scalability. Distributed computing systems are used in a wide range of applications, including

scientific research, data analysis, and the operation of large-scale online services (IBM Corporation, 2021).

According to aryasinghbjc (2022), the history of distributed computing can be traced back to the 1960s with the development of time-sharing systems, which allowed multiple users to access and execute programs on a single computer simultaneously. In the 1970s, the development of local area networks (LANs) and wide area networks (WANs) from ARPANET that facilitated the sharing of resources and information among computers, leading to the creation of distributed computing systems. These systems allowed multiple computers to work together to perform tasks. In the 1980s, the concept of client-server computing emerged, in which a central server provided resources and services to client computers (aryasinghbjc, 2022). The rise of the internet in the 1990s further expanded the capabilities of distributed computing systems, as it allowed for the creation of distributed applications that could be accessed from anywhere with an internet connection (aryasinghbjc, 2022). Today, distributed computing is used in a wide range of fields, including scientific research, financial analysis, and data processing (aryasinghbjc, 2022). It has also played a significant role in the development of cloud computing, which allows users to access and use shared computing resources over the internet (aryasinghbjc, 2022).

```
#include <stdio.h>
#include <mpi.h>

main(int argc, char **argv)
{
    int ierr;

    ierr = MPI_Init(&argc, &argv);
    printf("Hello world\n");

    ierr = MPI_Finalize();
}
```

If you compile hello.c with a command like

```
mpicc hello.c -o hello
```

you will create an executable file called hello, which you can execute by using the mpirun command as in the following session segment:

```
$ mpirun -np 4 hello
Hello world
Hello world
Hello world
Hello world
$
```

Figure 2: Sample code of Message Passing Interface program in C language (Thomasset & Grobe, 2023).

The code is an example of a basic MPI (Message Passing Interface) program in C. It includes the necessary header files and calls the `MPI_Init` function to initialize MPI (Thomasset & Grobe, 2023). The program then prints the message "Hello world" to the console (Thomasset & Grobe, 2023). The `MPI_Finalize` function is called to terminate MPI and free any resources that were allocated (Thomasset & Grobe, 2023).

To run the program, the code needs to be compiled using the `mpicc` command with the `-o` option to specify the output file name (Thomasset & Grobe, 2023). The resulting executable file is called "hello" (Thomasset & Grobe, 2023). The program is then executed using the `mpirun` command with the `-np` option to specify the number of processes to run (Thomasset & Grobe, 2023). In this case, the program is run with four processes, which results in the "Hello world" message being printed four times (Thomasset & Grobe, 2023).

1.1.1 Discussion on architectural evolution of distributed computing

The architectural evolution of distributed computing can be divided into several stages:

1. Client-server architecture: In this model, a central server provides services to clients over a network (aryasinghbjc, 2022). The clients send requests to the server, which processes the requests and sends back a response (aryasinghbjc, 2022). This model was developed in the 1970s and 1980s and is still widely used today (aryasinghbjc, 2022).

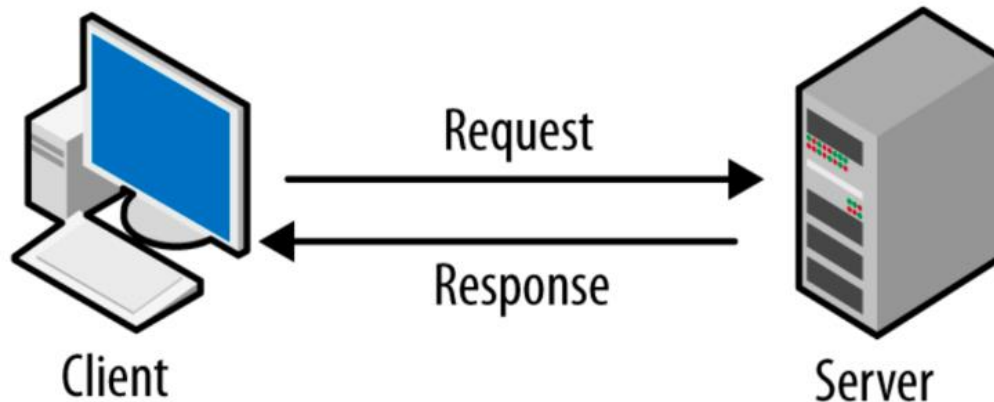


Figure 3: Client-server architecture (Madooei, 2023).

```
# first of all import the socket library
import socket

# next create a socket object
s = socket.socket()
print ("Socket successfully created")

# reserve a port on your computer in our
# case it is 12345 but it can be anything
port = 12345

# Next bind to the port
# we have not typed any ip in the ip field
# instead we have inputted an empty string
# this makes the server listen to requests
# coming from other computers on the network
s.bind(('', port))
print ("socket binded to %s" %(port))

# put the socket into listening mode
s.listen(5)
print ("socket is listening")

# a forever loop until we interrupt it or
# an error occurs
while True:

    # Establish connection with client.
    c, addr = s.accept()
    print ('Got connection from', addr )

    # send a thank you message to the client. encoding to send byte type.
    c.send('Thank you for connecting'.encode())

    # Close the connection with the client
    c.close()

    # Breaking once connection closed
    break
```

Figure 4: Sample code of Server side for Client-server architecture (GeeksforGeeks, 2023).

```
# Import socket module
import socket

# Create a socket object
s = socket.socket()

# Define the port on which you want to connect
port = 12345

# connect to the server on local computer
s.connect(('127.0.0.1', port))

# receive data from the server and decoding to get the string.
print (s.recv(1024).decode())
# close the connection
s.close()
```

Figure 5: Sample code of Client side for Client-server architecture (GeeksforGeeks, 2023).

The server in a client-server architecture has a method called `bind()`, which enables it to connect to a specific IP address and port number to receive incoming requests (GeeksforGeeks, 2023). Additionally, the `listen()` method allows the server to be in a state where it can listen for incoming connections (GeeksforGeeks, 2023). Lastly, the server has an `accept()` method that establishes a connection with a client and a `close()` method that terminates the connection with the client (GeeksforGeeks, 2023).

2. Peer-to-peer (P2P) architecture: In a P2P network, all nodes are equal and can act as both clients and servers (aryasinghbjc, 2022). This model is more decentralized and allows for more efficient resource sharing (aryasinghbjc, 2022). P2P networks became popular in the late 1990s with the emergence of file-sharing networks like Napster (aryasinghbjc, 2022).

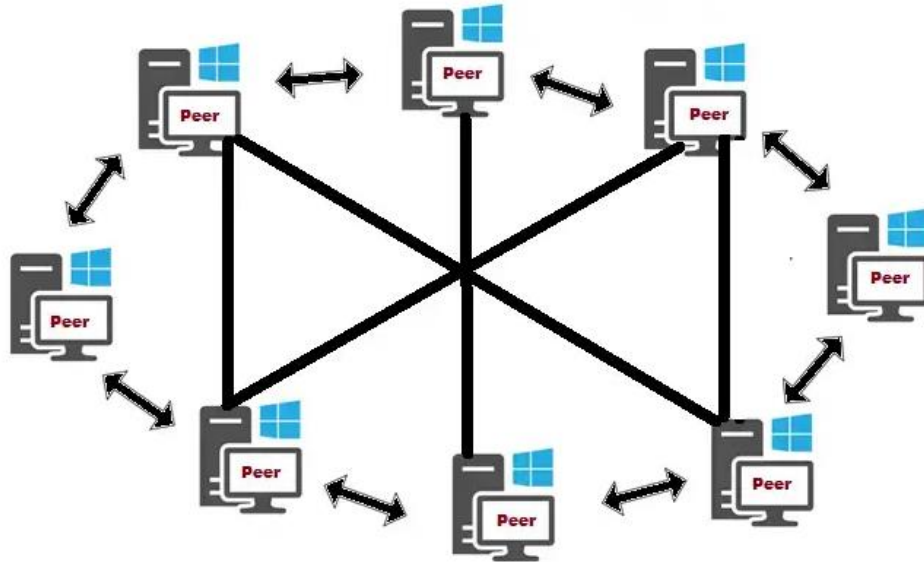


Figure 6: Peer-to-peer network architecture (Banger, 2023).

3. Grid computing: A large number of computers are connected together to form a "grid" that can be used to solve a single problem (aryasinghbjc, 2022). Grid computing is often used for scientific and technical computing applications (aryasinghbjc, 2022). It was developed in the 1990s and early 2000s as a way to harness the power of distributed computing for large-scale scientific projects (aryasinghbjc, 2022).



HOW GRID COMPUTING WORKS

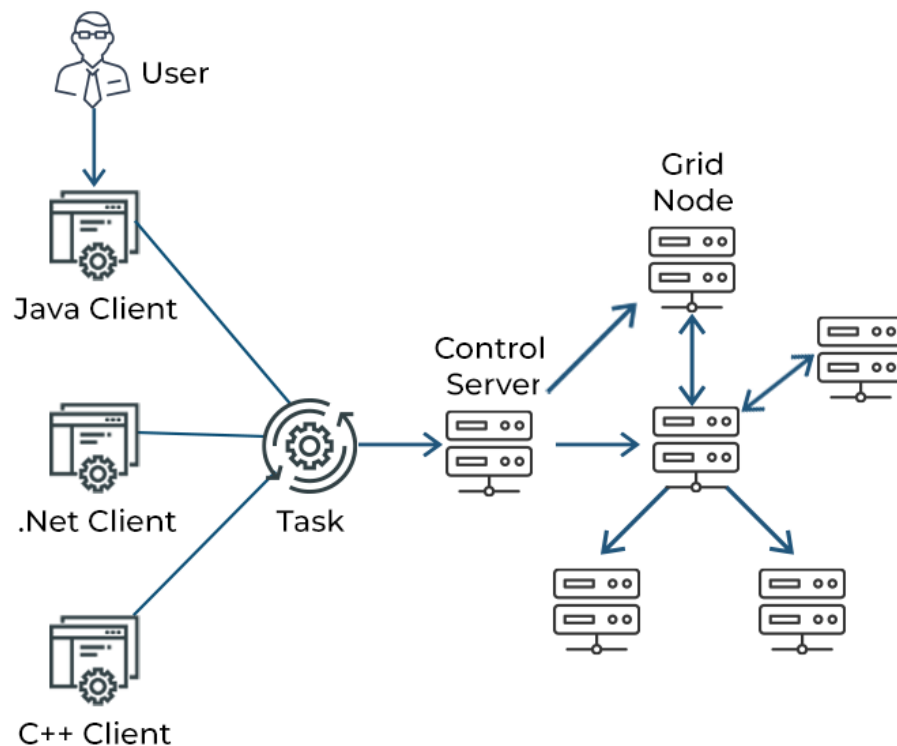


Figure 7: Grid Computing (Kanade, 2022).

4. Cloud computing: In this model, computing resources are provided as a service over the Internet (aryasinghbjc, 2022). Users can access these resources on demand and only pay for what they use (aryasinghbjc, 2022). Cloud computing emerged in the 2000s as a way to make it easier for businesses to access and use computing resources (aryasinghbjc, 2022).

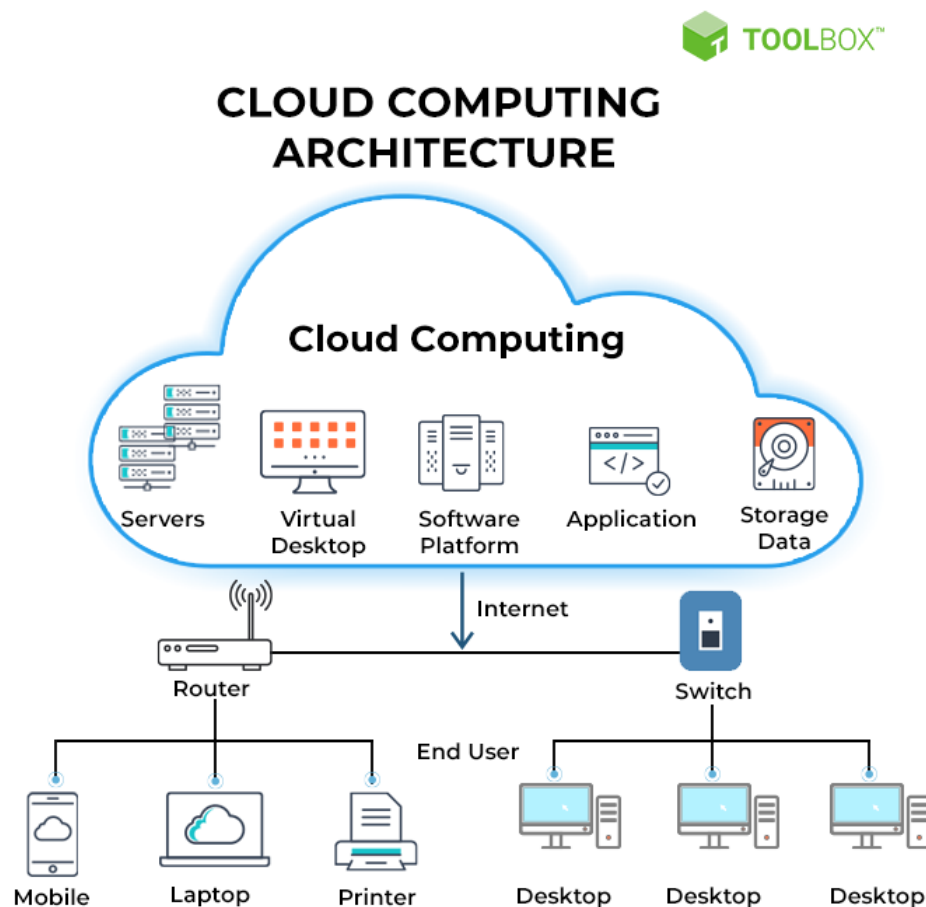


Figure 8: Cloud Computing Architecture (Patil, 2022).

5. Edge computing: Computing resources are placed closer to the users, at the "edge" of the network (aryasinghbjc, 2022). This allows for faster processing of data and reduces the need to transmit large amounts of data over the network (aryasinghbjc, 2022). Edge computing has become increasingly popular in recent years with the growth of the Internet of Things (IoT) and the need to process large amounts of data in real-time (aryasinghbjc, 2022).

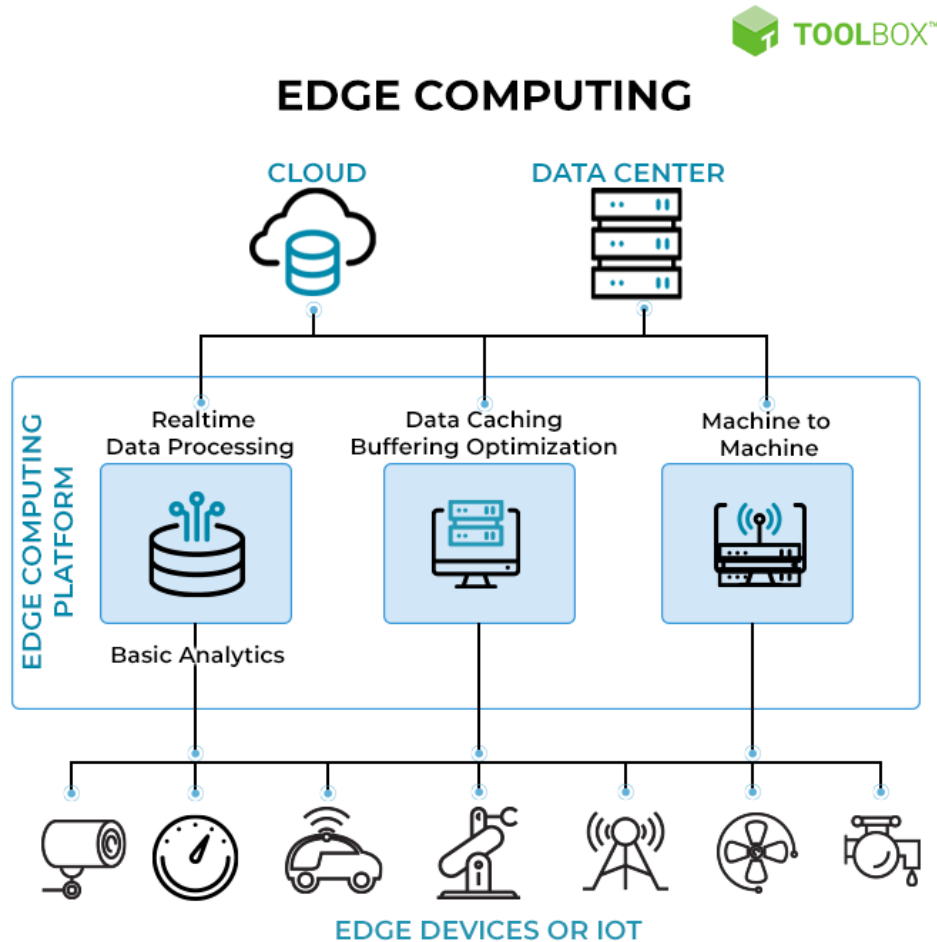


Figure 9: Edge Computing (BasuMallick, 2022).

1.2 Overview of types of enterprise application and architecture

Enterprise applications are large-scale software systems that are designed to support the complex business processes of an organization. They are critical to the success of any modern business and are used to manage various aspects of an enterprise, such as customer relationships, finances, and supply chain operations. This report will provide an overview of three major types of enterprise applications: Customer Relationship Management Software (CRM), Enterprise Resource Planning Systems (ERP), and Supply Chain Management Systems (SCM).

1.2.1 Customer Relationship Management (CRM) Software



Figure 10: Customer Relationship Management (Carter, 2022).

CRM software is designed to help businesses manage their interactions with customers and prospects (Carter, 2022). This includes managing customer information, sales leads, and marketing campaigns. The main functionality of CRM software includes lead tracking, sales forecasting, pipeline management, customer segmentation, and customer analytics (Trovato & Bottorff, 2022). With CRM software, businesses can improve their customer engagement, build stronger relationships with customers, and drive more revenue.

1.2.2 Enterprise Resource Planning (ERP) Systems

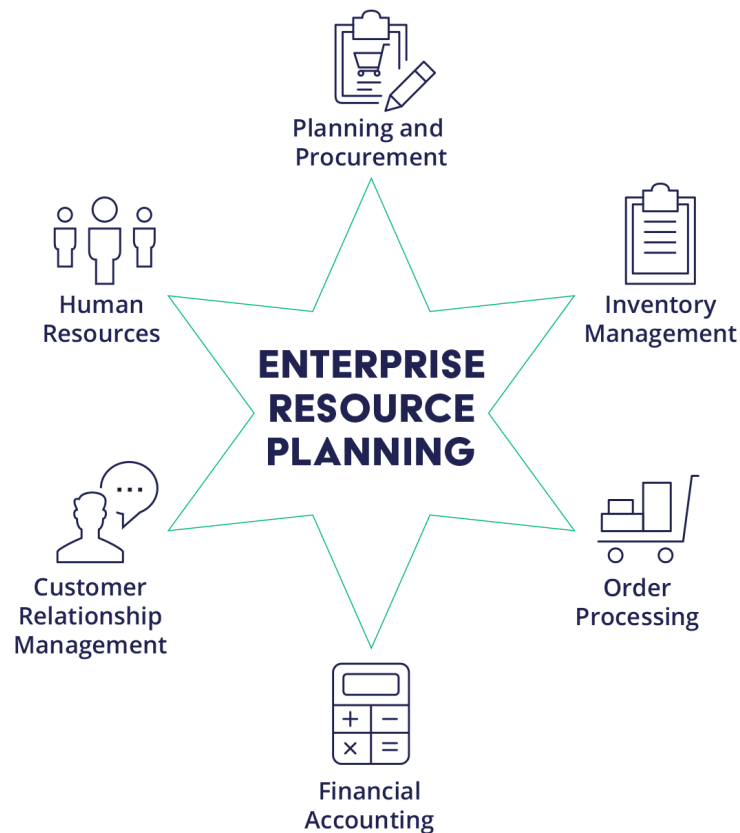


Figure 11: Enterprise Resource Planning (PTC Inc., 2023).

ERP systems are designed to manage the core business processes of an organization, including finance, accounting, human resources, inventory management, and supply chain management (PTC Inc., 2023). ERP systems typically integrate these functions into a single system, providing real-time information and visibility into business operations (PTC Inc., 2023). The main functionality of ERP systems includes financial management, supply chain management, production planning, human resources management, and customer relationship management. With ERP systems, businesses can improve efficiency, reduce costs, and make better-informed decisions.

1.2.3 Supply Chain Management (SCM) Systems



Figure 12: Supply Chain Management (IRC Group Global LTD, 2020).

SCM systems are designed to manage the flow of goods and services from suppliers to customers (IRC Group Global LTD, 2020). This includes managing inventory levels, production schedules, and delivery schedules (IRC Group Global LTD, 2020). The main functionality of SCM systems includes inventory management, demand planning, supply planning, transportation management, and supplier management (IRC Group Global LTD, 2020). With SCM systems, businesses can optimize their supply chain processes, reduce costs, and improve customer satisfaction (IRC Group Global LTD, 2020).

In conclusion, these three enterprise applications are crucial for businesses looking to improve their operations, streamline processes, and enhance customer experiences. With CRM software, businesses can improve customer engagement and drive revenue. With ERP systems, businesses can improve efficiency, reduce costs, and make better-informed decisions. And with SCM systems, businesses can optimize their supply chain processes and improve customer satisfaction.

1.2.4 Client-side Architecture

Client-server architecture is a type of computing architecture in which client computers request and receive data and services from server computers (Terra, 2022). In this model, the client is responsible for the user interface and the server is responsible for the back-end processing and storage (Intellipaat Software Solutions Pvt. Ltd., 2022). There are several variations of client-server architecture, ranging from 1-tier to 3-tier.

1-tier client-server architecture, also known as single-tier, is a system where the client and server are combined into a single system. An example of a 1-tier client-server architecture is a simple database application where the client is a desktop application that communicates with the server (a database) to retrieve and update data (Intellipaat Software Solutions Pvt. Ltd., 2022).

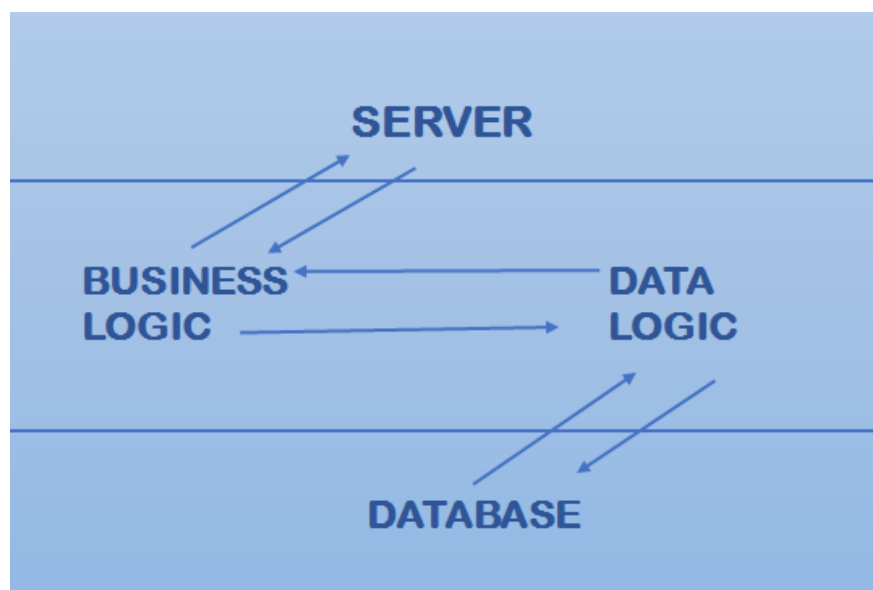


Figure 13: 1-tier architecture.

2-tier client-server architecture is a system where the client communicates directly with the server, which is responsible for both processing and storing data. An example of a 2-tier client-server architecture is a file server where the client (a desktop computer) can access and download files from the server (Intellipaat Software Solutions Pvt. Ltd., 2022).

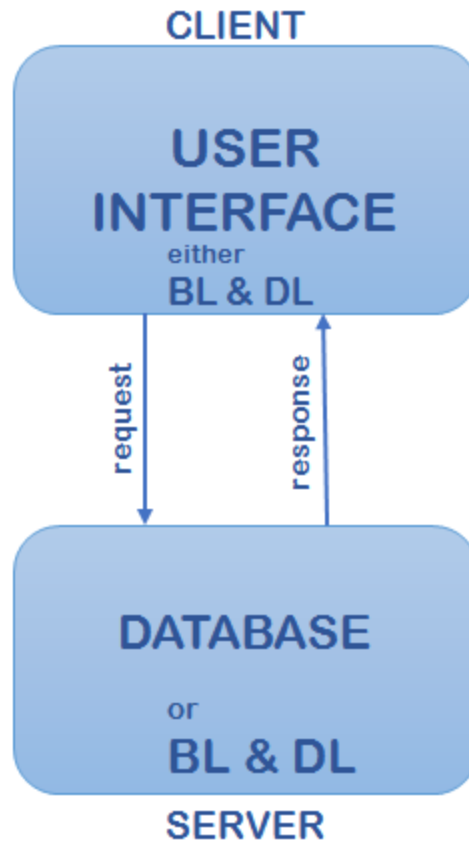


Figure 14: 2-tier architecture.

3-tier client-server architecture is a system where there is an additional layer between the client and server, known as the middleware, which is responsible for translating requests and responses between the client and server, and can also perform tasks such as load balancing and security (Terra, 2022). An example of a 3-tier client-server architecture is a web-based application, where the client (a web browser) communicates with the server (a web application) through the middleware (a web server) (Intellipaat Software Solutions Pvt. Ltd., 2022).



Figure 15: 3-tier architecture.

Client-server architectures are widely used in enterprise computing, as they allow for centralization of data and resources and can be more scalable and reliable than other architectures (Terra, 2022). However, they also have some limitations, such as the need for a stable network connection and the potential for a single point of failure (Intellipaat Software Solutions Pvt. Ltd., 2022).

1.2.5 Microservices Architecture

Microservices architecture is a way of designing software systems as a collection of independent, self-contained services (Google Inc., 2023). These services are designed to be small, modular, and focused on a specific business goal (Amazon Web Services, Inc., 2023a). Each service is responsible for a specific business capability and communicates with other services through well-defined interfaces, usually using APIs (SmartBear Software, 2023).

For example, consider an e-commerce website. The website could be designed using microservices architecture, with separate services for the following functionality:

- Product catalog management
- Shopping cart management
- Order processing
- User authentication and authorization
- Payment processing
- Recommendation engine

Each of these services would be independently developed and deployed, and would communicate with each other through APIs to provide a seamless experience for the end user.

One of the main benefits of microservices architecture is the ability to scale and deploy individual services independently (Google Inc., 2023). This allows for faster development and deployment cycles, as well as the ability to update and maintain parts of the system without affecting the entire system (Amazon Web Services, Inc., 2023a). For example, if the recommendation engine service needs to be updated with new algorithms, it can be deployed independently without affecting the other services or the overall functioning of the e-commerce website.

Another advantage of microservices architecture is the ability to use different technologies and programming languages for different services (SmartBear Software, 2023). This allows developers to choose the best tools and technologies for each specific service, rather than being limited to a single technology stack for the entire system. For example, the recommendation engine service could be built using machine learning algorithms and implemented in Python, while the

product catalog management service could be built using a NoSQL database and implemented in Java.

However, microservices architecture also introduces additional complexity, as it requires the development and maintenance of multiple independent services and their interfaces (Google Inc., 2023). It also requires a strong focus on communication and coordination between services, as well as a robust infrastructure to support the deployment and management of these services (Amazon Web Services, Inc., 2023a).

Overall, microservices architecture is a powerful approach to designing software systems that allows for greater flexibility, scalability, and maintainability. It is particularly well-suited for large, complex systems that require the ability to evolve and adapt over time (SmartBear Software, 2023).

```
1 import org.springframework.boot.*;
2 import org.springframework.boot.autoconfigure.*;
3 import org.springframework.stereotype.*;
4 import org.springframework.web.bind.annotation.*;
5
6 @RestController
7 @EnableAutoConfiguration
8 public class Example {
9     @RequestMapping("/")
10    String home() {
11        return "Hello World!";
12    }
13
14    public static void main(String[] args) throws Exception {
15        SpringApplication.run(Example.class, args);
16    }
17 }
```

Figure 16: Sample code of Microservices with Spring Boot (Stringfellow, 2022).

1.2.6 Service-oriented Architecture

According to Amazon Web Services (2023c), service-oriented architecture (SOA) is a software design approach that promotes the use of independent, self-contained units of functionality called services. These services are designed to be modular, reusable, and loosely coupled, which means that they can be easily combined and reused to build new applications and systems (Amazon Web Services, Inc., 2023c). SOA helps organizations to build flexible, scalable systems that can be easily modified and extended over time, as services can be added, removed, or replaced without affecting the overall functionality of the system (Amazon Web Services, Inc., 2023c). Additionally, SOA promotes the use of standard protocols and interfaces, which makes it easier for different services to communicate and interoperate with each other, reducing the complexity and cost of integrating different systems and applications (Amazon Web Services, Inc., 2023c).

According to IBM (2023), SOA is often used in large, complex systems that involve the integration of multiple services and applications. It can help to improve the agility and flexibility of these systems by enabling them to be built and modified in a modular way (IBM Corporation, 2023).

According to SarthakGarg (2022), examples of systems that use SOA include e-commerce platforms, healthcare systems, and enterprise resource planning systems. Another example is the games that operate using built-in features that requires GPS, in which case it makes use of the device's built-in GPS capabilities (SarthakGarg, 2022). Thus, this is SOA in mobile solution. By using SOA, these systems can be built in a flexible and scalable way, which enables them to evolve and adapt over time (SarthakGarg, 2022).

1.3 Differentiate Customer Relationship Management Software, Enterprise Resource Planning Systems, Supply Chain Management Systems.

	Customer Relationship Management (CRM)	Enterprise Resource Planning (ERP)	Supply Chain Management (SCM)
Definition	A software system for managing customer interactions and data throughout the customer lifecycle (Viñarás, 2022).	An integrated software system for managing and coordinating all the resources, information, and processes of an organization (Oracle, 2023).	A software system for managing and coordinating the movement and storage of goods, services, and related information from the point of origin to the point of consumption (Hayes, 2022).
Key features	Contact management, sales management, marketing automation, customer service and support (Anand, 2023).	Financial management, human resources management, inventory management, risk management, project management (Meloeny, 2022).	Demand planning, inventory management, transportation management, warehousing, and supplier collaboration (Roy, 2023).
Primary goal	Improve customer relationships and loyalty (Viñarás, 2022).	Streamline and optimize business processes (Oracle, 2023).	Increase efficiency and lower the costs in the supply chain (Hayes, 2022).
Examples of industries/companies	Retail, banking, hotel, financial services, insurance, consulting, agriculture (QuickDesk Pte Ltd., 2023).	Manufacturing, healthcare, distribution (Meloeny, 2022).	Manufacturing, retail, transportation (Hayes, 2022).
Data focus	Customer data, interactions, and behavior (Viñarás, 2022).	Organizational data, such as financial transactions and employee information (Oracle, 2023).	Supply chain data, such as demand and inventory levels.
Integration with other systems	Often integrated with marketing and sales systems (Viñarás, 2022).	Integrates with all major business functions within an organization (Oracle, 2023).	Often integrated with ERP and CRM systems (Edwards, 2022).



Figure 17: Customer Relationship Management Flow (Lund, 2023).

Customer Relationship Management (CRM) is a process that companies use to manage their interactions with customers and potential customers. CRM is all about building strong relationships with customers and ensuring that they are satisfied with the company's products and services. The process of CRM can be broken down into several steps:

1. **Marketing:** The first step in the CRM process flow is marketing, which involves identifying potential customers and creating targeted campaigns to reach them (Lund, 2023). In this step, the company collects information about the customer, such as their contact information, demographics, and preferences, which is then stored in the CRM system (Lund, 2023).
2. **Sales:** Once potential customers have been identified, the next step is to convert them into actual customers through the sales process (Lund, 2023). This involves using the information collected in the marketing stage to understand the customer's needs and preferences, and tailoring the sales pitch to meet those needs (Lund, 2023). The sales

team uses the CRM system to track customer interactions, store important documents, and manage the sales process, all while updating the customer profile in the CRM system as necessary (Lund, 2023).

3. Feedback: After the sale has been made, the company needs to ensure that the customer remains satisfied with their purchase. This is where the feedback stage comes in, which involves collecting feedback from customers about their experience with the company and its products or services. The company can use the CRM system to track customer feedback, respond to customer inquiries or complaints, and identify areas for improvement (Lund, 2023).
4. Support: Finally, the support stage involves providing ongoing support to customers to ensure that they continue to have a positive experience with the company (Lund, 2023). This can involve providing technical support, troubleshooting, or answering general questions about the company's products or services (Lund, 2023). The support team can use the CRM system to track customer interactions, store important documents, and manage the support process, all while updating the customer profile in the CRM system as necessary (Lund, 2023).

Overall, the CRM process flow involves using a centralized database of customer information to manage and track interactions with customers throughout the entire customer lifecycle. This can help companies to build strong relationships with their customers, improve customer satisfaction, and ultimately drive business growth.

ERP Implementation Stages

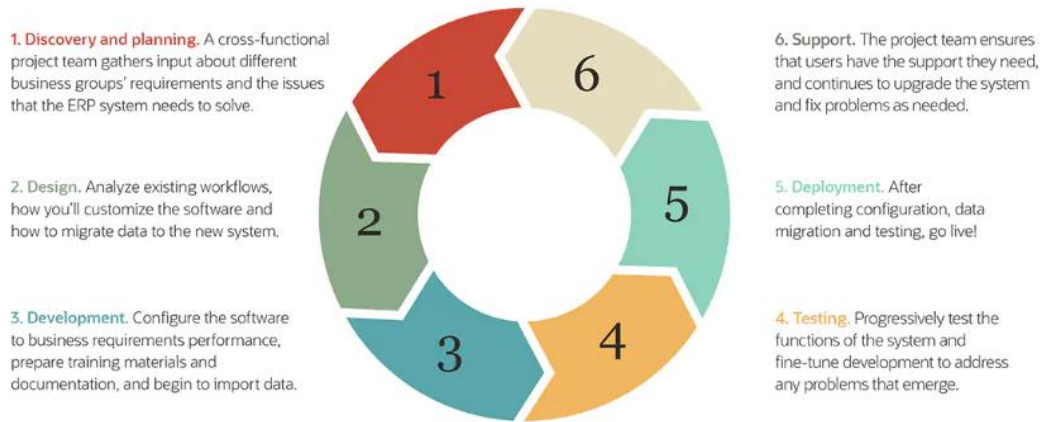


Figure 18: ERP implementation flow (Schwarz, 2022).

Then, Enterprise Resource Planning (ERP) is a software-based approach to managing a company's business processes. ERP systems are designed to integrate all aspects of a company's operations, including finance, accounting, manufacturing, distribution, and human resources. The process of ERP can be broken down into several steps:

1. **Discovery and Planning:** It involves understanding the business requirements and identifying the key stakeholders involved in the project (Schwarz, 2022). This stage includes gathering data about existing systems, processes, and data sources, as well as identifying any potential gaps or areas for improvement (Schwarz, 2022). The ERP team then develops a detailed project plan that outlines the scope, budget, and timeline of the project (Schwarz, 2022).
2. **Design:** Once the project plan has been developed, the ERP team moves on to the design stage, which involves creating a detailed system design that meets the business requirements identified in the previous stage (Schwarz, 2022). This can involve creating data models, designing user interfaces, and configuring workflows and processes (Schwarz, 2022). The design stage typically involves collaboration between the ERP team and the key stakeholders to ensure that the system meets their needs and expectations (Schwarz, 2022).
3. **Development:** With the system design in place, the ERP team moves on to the development stage, which involves building and configuring the system according to

the design specifications (Schwarz, 2022). This can involve developing custom software, integrating third-party applications, and configuring the system to meet the specific needs of the organization (Schwarz, 2022).

4. Testing: Once the system has been developed, it undergoes extensive testing to ensure that it meets the business requirements and functions as intended (Schwarz, 2022). The ERP team uses a variety of testing tools and methodologies to identify and resolve any defects or issues that arise during testing (Schwarz, 2022).
5. Deployment: Once the system has been thoroughly tested and any issues have been resolved, it is ready for deployment. This involves installing the system in the production environment, migrating data from existing systems, and training users on the new system (Schwarz, 2022). The ERP team works closely with the organization to ensure that the deployment goes smoothly and that any issues are addressed promptly.
6. Ongoing Support: Finally, the ERP team provides ongoing support to ensure that the system continues to meet the needs of the organization over time (Schwarz, 2022). This can involve providing technical support, troubleshooting, and making updates or modifications to the system as needed (Schwarz, 2022).

Next, Supply Chain Management (SCM) is the process of managing the flow of goods and services from the point of origin to the point of consumption (Fernando, 2022). SCM involves coordinating the activities of suppliers, manufacturers, distributors, and retailers to ensure that products are delivered to customers in a timely and cost-effective manner (Fernando, 2022). The process of SCM can be broken down into several steps:

1. Planning: The process commences with planning phase to align the supply of goods with the demands of customers and production requirements (Fernando, 2022). To achieve this, companies are required to make forecasts about their future requirements and take appropriate action (Fernando, 2022). These estimates encompass the raw materials required at every stage of the production process, constraints on equipment capacity, and staffing needs throughout the supply chain management process (Fernando, 2022).

2. Sourcing: After that, it is time to source the materials and products that will be needed to produce (Fernando, 2022). This can involve negotiating contracts with suppliers, managing inventory levels, and tracking shipments (Fernando, 2022).

3. Manufacturing: The next step is to manufacture the products that will be delivered to customers. This can involve managing production schedules, ensuring that products meet quality standards, and managing the flow of materials through the production process (Fernando, 2022).

4.. Delivering: After the products have been manufactured, they need to be delivered to customers. This can involve managing logistics, transportation, and distribution networks to ensure that products are delivered on time and in good condition (Fernando, 2022).

5.. Managing returns: In some cases, customers may need to return products for various reasons such as defects, damages, or dissatisfaction (Fernando, 2022). Therefore, SCM also involves managing returns and exchanges processes efficiently to maintain customer satisfaction (Fernando, 2022).

By studying the architectures in depth, when monolithic architecture applied in CRM systems, it involves building a single, large application that handles all functionality within the system (Harris, 2023). This can impact the performance, scalability, reliability, and latency of CRM systems in different ways. For example, if the system becomes too large and complex, it may become difficult to scale or update the system without affecting the entire system (Harris, 2023). On the other hand, a monolithic system may be more reliable, as it is less likely to fail due to the failure of a single service.

Furthermore, microservices architecture are commonly used in ERP systems. This approach involves breaking down a large application into smaller, independent services that communicate with each other through APIs (SmartBear Software, 2023). This can improve the performance, scalability, reliability, and latency of ERP systems by allowing different parts of the system to scale independently, reducing downtime and improving overall system availability. In figure 16, the code snippet sets up a Spring Boot application for a product service in an ERP system, registers it with Eureka for service discovery, defines a REST endpoint for retrieving a product by ID, defines a JPA repository for accessing product data, and defines a product entity class.

```
@SpringBootApplication
@EnableEurekaClient
public class ProductServiceApplication {

    public static void main(String[] args) {
        SpringApplication.run(ProductServiceApplication.class, args);
    }

    @RestController
    class ProductController {

        @Autowired
        private ProductRepository productRepository;

        @GetMapping("/products/{id}")
        public Product getProduct(@PathVariable Long id) {
            return productRepository.findById(id);
        }

    }

    @Repository
    public interface ProductRepository extends JpaRepository<Product, Long> {
        // CRUD methods for accessing Product data
    }

    public class Product {
        // Product entity class with properties and methods
    }

}
```

Figure 19: Sample code of Microservices architecture applied in ERP (Netflix, 2023).

In addition, service-oriented architecture is also widely known to be applied in SCM systems. This approach involves breaking down a large application into smaller, independent services that communicate with each other through APIs (SarthakGarg, 2022). This can improve the performance, scalability, reliability, and latency of SCM systems by allowing different parts of the system to scale independently, reducing downtime and improving overall system availability (SarthakGarg, 2022).

Overall, the architecture approach that is most suitable for CRM, ERP, and SCM systems will depend on the specific needs and requirements of the system. Monolithic approaches may be more suitable for systems that do not require as much flexibility, while microservices and SOA approaches may be more suitable for systems that require high levels of scalability and reliability.

1.4 Explore and identify APU Systems to implement suitable architecture

By exploring the APU systems, there are a few systems that are worth to bring into discussion. The current existing APU systems are the APCard, APU Classroom Finder, APU Attendance System, and many more. APCard acts as a e-wallet that only available in APU, it allows student to purchase food and beverages from restaurants and convenient stores inside APU campus, it also allows students to pay for their parking. APCard shows a purchase history on the APSPACE application for students to keep their finance on track. APU Classroom Finder is a system for APU students to locate available rooms and book them in advance based on students' date and time input. Finally, the APU Attendance System is an attendance tracking system where students are able to get their attendance checked on each class they attended. It is a very crucial system as it affects the student availability to sit for examinations, as well as for lecturers to track students attendances and performances.

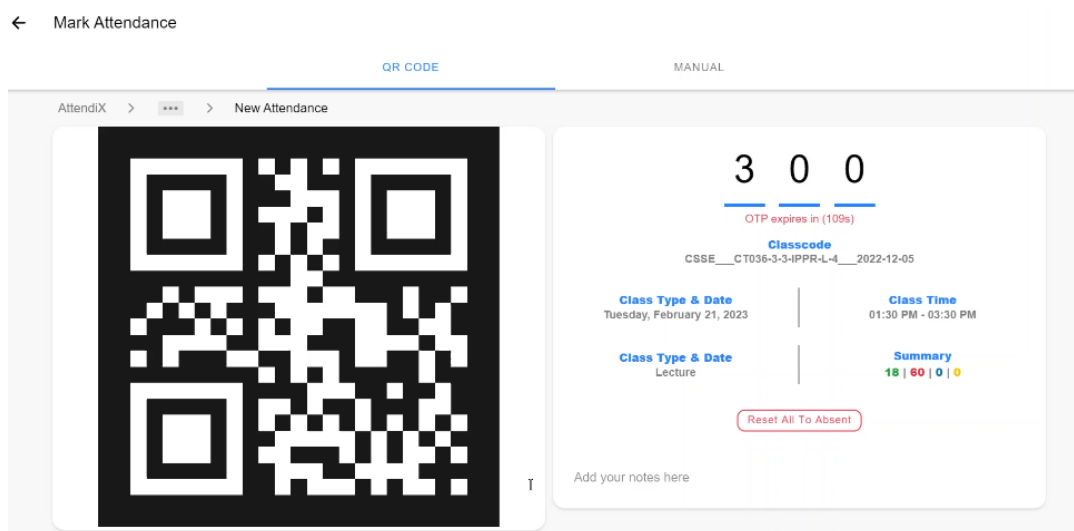


Figure 20: Example of APU Attendance System.

The Service-Oriented Architecture is extremely suitable to be implemented into the APU attendance system. The service-oriented architecture is a collection of services that can be utilized to perform specific functions. Services communicate with each other using specific protocols instead of incorporating calls to other services (w3computing.com, 2023). The system diagram below demonstrates how services are accessed within the system. Services can be general-purpose and can be obtained from external sources, including the internet (w3computing.com, 2023). However, other services are tailored to the specific needs of a particular business, providing

business logic and setting them apart from others (w3computing.com, 2023). Services can be accessed as needed and can be reused in various application modules (w3computing.com, 2023).

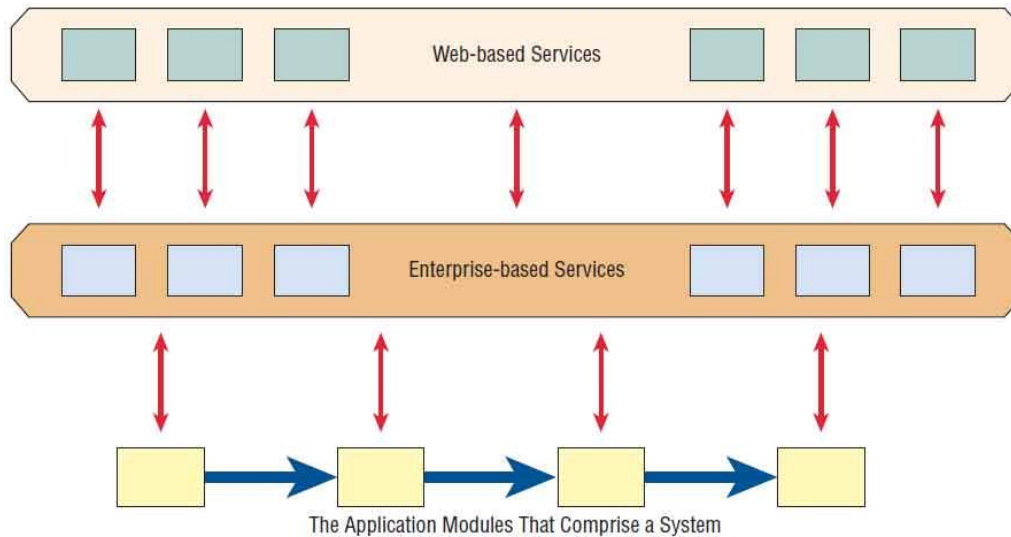


Figure 21: Service-Oriented Architectures (w3computing.com, 2023).

An attendance system can be suitable for a service-oriented architecture (SOA) for several reasons. One of the key benefits of SOA is modularity, which allows the system to be designed as a set of independent services that can be easily deployed, maintained, and scaled separately (IBM, 2022). This means that each service can be developed and updated independently, without affecting other services. For example, if the attendance tracking service needs to be updated with new features, it can be done without affecting other services like the student information system or exam registration system. This enhances the flexibility and agility of the system, making it easier to modify or extend as needed.

SOA also enables reusability of services, which is another key benefit of using it for the APU Attendance System. The services in the attendance system can be designed to be reusable by other systems or applications, such as payroll, or time tracking system (Talend, 2023). By making the attendance tracking service reusable, the system can become more cost-effective and efficient (Sinha, 2022). Moreover, reusability reduces development time and enhances system integration and data exchange, thus reducing the risk of errors and improving overall system performance (Sinha, 2022).

SOA also supports interoperability of services. Services in an attendance system can be designed to communicate with each other and with other systems using standard protocols like HTTP and REST (Stevens, 2005). This promotes seamless communication between the attendance system and other systems, enhancing data exchange and system integration (Sinha, 2022). For example, the attendance tracking service can communicate with the student information system to obtain student details, or with the exam registration system to verify eligibility for exams.

Finally, SOA supports loose coupling of services. Services in an attendance system can operate independently of each other and can be modified or replaced without affecting other services (Rehman, 2023). This enhances the system's resilience, making it easier to maintain and reduce the risk of system failure. For example, if a service like the exam registration system needs to be replaced or updated, it can be done without affecting other services like the attendance tracking system.

Overall, the benefits of SOA for the APU Attendance System include increased flexibility, modularity, reusability, interoperability, and loose coupling. By adopting SOA, the attendance system can be more efficient, cost-effective, and reliable.

1.5 Improve another system similar with chosen system

Service-oriented architecture (SOA) involves breaking down a large application into smaller, independent services that communicate with each other through APIs. This approach can be used to improve the performance, scalability, reliability, and latency of attendance systems in several ways:

Scalability: By breaking down the attendance system into smaller, independent services, it becomes easier to scale different parts of the system independently as needed (Nehra, 2021). This means that if certain parts of the system are experiencing higher usage or demand, they can be scaled up without affecting the rest of the system (Nehra, 2021). This can improve overall system performance and responsiveness.

Reliability: SOA can also improve the reliability of the attendance system by reducing the risk of downtime due to the failure of a single service (Nehra, 2021). If a service within the system fails, it can be isolated and fixed without affecting the rest of the system, improving overall system availability (Nehra, 2021).

Performance: SOA can also improve the performance of the attendance system by allowing different services to be optimized for specific tasks (Nehra, 2021). This can result in faster and more efficient processing of data within the system.

In this section, an author works has been taken as a reference for the similar attendance system: <https://github.com/ArunGopinathan/AttendanceManagementSystemUsingQRCode>. The attendance system from ArunGopinathan (2015) is using QR code to mark attendance for students.

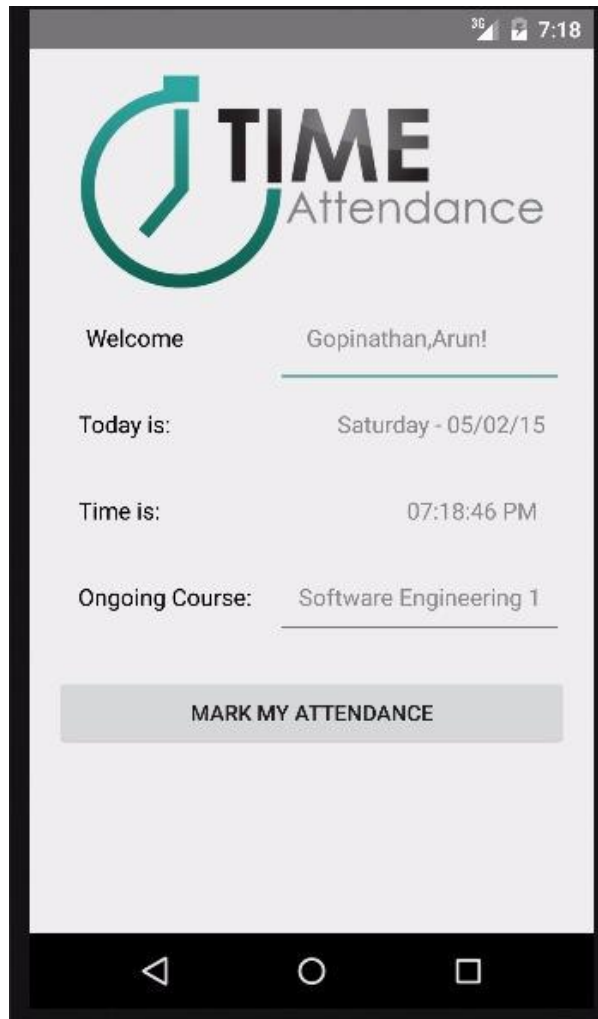


Figure 22: Attendance System from existing resources (ArunGopinathan, 2015).

The system is developed for android application to ease the attendance process for both professors and the students (ArunGopinathan, 2015). Next, a sample code from an existing resource that have not integrated SOA into the attendance system is displayed below.

```

private Bitmap generateQRCode(String content)
{
    QRCodeWriter writer = new QRCodeWriter();
    try {
        //Find screen size

        EnumMap<EncodeHintType, Object> hint = new EnumMap<EncodeHintType, Object>(EncodeHintType.class);
        hint.put(EncodeHintType.CHARACTER_SET, "UTF-8");
        BitMatrix bitMatrix = writer.encode(content, BarcodeFormat.QR_CODE, 800, 800, hint);
        int width = bitMatrix.getWidth();
        int height = bitMatrix.getHeight();
        int[] pixels = new int[width * height];
        for (int y = 0; y < height; y++) {
            int offset = y * width;
            for (int x = 0; x < width; x++) {
                // pixels[offset + x] = bitMatrix.get(x, y) ? 0xFF000000
                // : 0xFFFFFFFF;
                pixels[offset + x] = bitMatrix.get(x, y) ? Color.BLACK : Color.WHITE;
            }
        }

        Bitmap bitmap = Bitmap.createBitmap(width, height, Bitmap.Config.ARGB_8888);
        bitmap.setPixels(pixels, 0, width, 0, 0, width, height);
        return bitmap;
    }
    catch (Exception e)
    {
        Log.w(LOGTAG, "QR Code Generator Error:" + e.toString());
    }
    return null;
}

private class AsyncCallWS extends AsyncTask<String, Void, Void> {
    @Override
    protected Void doInBackground(String... params) {
        // Log.i(LOGTAG, "doInBackground");
        String address = "0";
        try {
            WifiManager manager = (WifiManager) getSystemService(Context.WIFI_SERVICE);
            WifiInfo info = manager.getConnectionInfo();
            address = info.getMacAddress();
        }
        catch (Exception ex)
        {
            ex.printStackTrace();
        }
        // String address = "00:0a:95:9d:68:16";

        // String content = "1001160219;CSE5324;" + address;
        String content = user.getUserId() + ";" + courseId + ";" + address + ";" + timeStamp;
        Log.w("AMS", content);
        generatedImage = generateQRCode(content);
        return null;
    }

    @Override
    protected void onPostExecute(Void result) {
        mProgressBar.setVisibility(View.GONE);
        if (generatedImage != null) {
            mQRCodeImageView.setImageBitmap(generatedImage);
        }
        else
            Log.w(LOGTAG, "QR Code was null");
    }
}

```

Figure 23: Attendance System Sample Code (ArunGopinathan, 2015).

This is a sample code that generates a QR code image for attendance purposes. It includes two methods - "generateQRCode" and "AsyncCallWS."

The "generateQRCode" method takes a string input "content" and generates a QR code image using the QRCodeWriter class. It sets the character set to UTF-8 and uses an EnumMap to store the encoding hint type and associated value. It then creates a BitMatrix object using the "encode" method of the QRCodeWriter class with the input content, barcode format (QR code), and specified height and width values. The BitMatrix object is then used to create an array of pixel values (integers) for each pixel of the QR code. The color of each pixel is determined based on the value of the corresponding BitMatrix element. Finally, the pixel array is used to create a Bitmap object that represents the QR code image, and this object is returned.

The "AsyncCallWS" class extends the AsyncTask class and is used to generate the QR code image asynchronously. It overrides the "doInBackground" and "onPostExecute" methods. In the "doInBackground" method, it retrieves the MAC address of the device using the WifiManager class and generates the input content string for the QR code using user ID, course ID, MAC address, and timestamp. Then, it calls the "generateQRCode" method to generate the QR code image and assigns it to the "generatedImage" variable. In the "onPostExecute" method, the progress view is hidden, and if the generatedImage is not null, it sets the generated QR code image to the mQRCodeImageView.

Overall, this code can be modified and integrated with SOA architecture to enhance the attendance system. For instance, the attendance data can be stored in a database and accessed through web services using RESTful APIs. Moreover, the generation of QR codes can be performed by a separate service, which can be invoked by the main application using web services. By decoupling the components and services, SOA architecture can improve scalability, flexibility, and interoperability of the attendance system.

To apply Service Oriented Architecture (SOA) to this attendance system, the system can be break into separate services, each responsible for a specific function. Here are some modifications that can be made:

1. Create a separate service for generating QR codes:

```
public class QRCodeService {
    public Bitmap generateQRCode(String content) {
        QRCodeWriter writer = new QRCodeWriter();
        try {
            EnumMap<EncodeHintType, Object> hint = new EnumMap<EncodeHintType, Object>(EncodeHintType.class);
            hint.put(EncodeHintType.CHARACTER_SET, "UTF-8");
            BitMatrix bitMatrix = writer.encode(content, BarcodeFormat.QR_CODE, 800, 800, hint);
            int width = bitMatrix.getWidth();
            int height = bitMatrix.getHeight();
            int[] pixels = new int[width * height];
            for (int y = 0; y < height; y++) {
                int offset = y * width;
                for (int x = 0; x < width; x++) {
                    pixels[offset + x] = bitMatrix.get(x, y) ? Color.BLACK : Color.WHITE;
                }
            }
            Bitmap bitmap = Bitmap.createBitmap(width, height, Bitmap.Config.ARGB_8888);
            bitmap.setPixels(pixels, 0, width, 0, 0, width, height);
            return bitmap;
        } catch (Exception e) {
            Log.w(LOGTAG, "QR Code Generator Error:" + e.toString());
        }
        return null;
    }
}
```

Figure 24: Attendance System enhanced with SOA.

Instead of generating QR codes within the existing code, a separate service can be created that generates QR codes. This service can be accessed by the attendance system to generate QR codes as needed.

2. Create a separate service for getting the device's MAC address:

```
public class DeviceService {
    public String getMacAddress() {
        try {
            WifiManager manager = (WifiManager) getSystemService(Context.WIFI_SERVICE);
            WifiInfo info = manager.getConnectionInfo();
            return info.getMacAddress();
        } catch (Exception ex) {
            ex.printStackTrace();
        }
        return "0";
    }
}
```

Figure 25: Attendance System improvement codes with SOA application.

Instead of getting the MAC address within the existing code, a separate service can be created to retrieve the MAC address of the device. This service can be accessed by the attendance system to get the device's MAC address as needed.

3. Use RESTful APIs to communicate between services:

```

public class AttendanceService {
    private static final String QR_CODE_GENERATION_SERVICE_URL = "http://localhost:8080/qr-code";
    private static final String DEVICE_INFO_SERVICE_URL = "http://localhost:8080/device-info";
    private static final String AUTHORIZATION_TOKEN = "YOUR_AUTHORIZATION_TOKEN";

    public void takeAttendance(User user, String courseId, String timeStamp) {
        String macAddress = getMacAddressFromDeviceService();
        String content = user.getUserId() + ";" + courseId + ";" + macAddress + ";" + timeStamp;
        Bitmap generatedImage = generateQRCodeFromQRCodeService(content);
        // do something with the generated QR code
    }

    private Bitmap generateQRCodeFromQRCodeService(String content) {
        try {
            URL url = new URL(QR_CODE_GENERATION_SERVICE_URL);
            HttpURLConnection connection = (HttpURLConnection) url.openConnection();
            connection.setRequestMethod("POST");
            connection.setRequestProperty("Content-Type", "application/json");
            connection.setRequestProperty("Authorization", "Bearer " + AUTHORIZATION_TOKEN);
            connection.setDoOutput(true);
            OutputStream outputStream = connection.getOutputStream();
            outputStream.write(content.getBytes());
            outputStream.flush();
            outputStream.close();
            int responseCode = connection.getResponseCode();
            if (responseCode == HttpURLConnection.HTTP_OK) {
                InputStream inputStream = connection.getInputStream();
                Bitmap bitmap = BitmapFactory.decodeStream(inputStream);
                inputStream.close();
                return bitmap;
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        return null;
    }

    private String getMacAddressFromDeviceService() {
        try {
            URL url = new URL(DEVICE_INFO_SERVICE_URL);
            HttpURLConnection connection = (HttpURLConnection) url.openConnection();
            connection.setRequestMethod("GET");
            connection.setRequestProperty("Authorization", "Bearer " + AUTHORIZATION_TOKEN);
            int responseCode = connection.getResponseCode();
            if (responseCode == HttpURLConnection.HTTP_OK) {
                InputStream inputStream = connection.getInputStream();
                BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(inputStream));
                String response = bufferedReader.readLine();
                bufferedReader.close();
                inputStream.close();
                return response;
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        return null;
    }
}

```

Figure 26: Attendance System with SOA application.

To communicate between services, RESTful APIs is recommended to use. Each service can expose its functionality through RESTful APIs, which can be accessed by other services or the attendance system. These methods use HTTP requests to communicate with the QR Code Generation Service and Device Info Service, respectively. The “generateQRCodeFromQRCodeService” method sends a POST request to the QR Code Generation Service endpoint with the content as the request body, and the “getMacAddressFromDeviceService” method sends a GET request to the Device Info Service endpoint.

2.0 Part Two

2.1 Design of Web Components

Web components are the fundamental building blocks of a web application, comprising everything from basic HTML elements to advanced user interface elements. JavaServer Pages (JSPs) and Servlets are widely used web component technologies in Java-based web development.

2.1.1 JSP

Java Server Pages (JSP) is a technology that allows developers to create dynamic web pages using Java code. It is essentially a server-side scripting language that enables the creation of web pages that can interact with the server, database, and other web services (W3schools of Technology, 2023).

The importance of JSP lies in its ability to create dynamic web pages that can display different content based on user input or other factors. This means that a web page can be customized for each user, making the browsing experience more interactive and engaging. Furthermore, JSP pages are compiled into Java servlets, which provide better performance and scalability than traditional CGI scripts.

One of the main benefits of using JSP is that it allows for the separation of business logic and presentation, making it easier to maintain and modify code (W3schools of Technology, 2023). JSP also supports a wide range of web technologies, including HTML, CSS, and JavaScript, which means that developers can create complex web applications that are both functional and visually appealing.

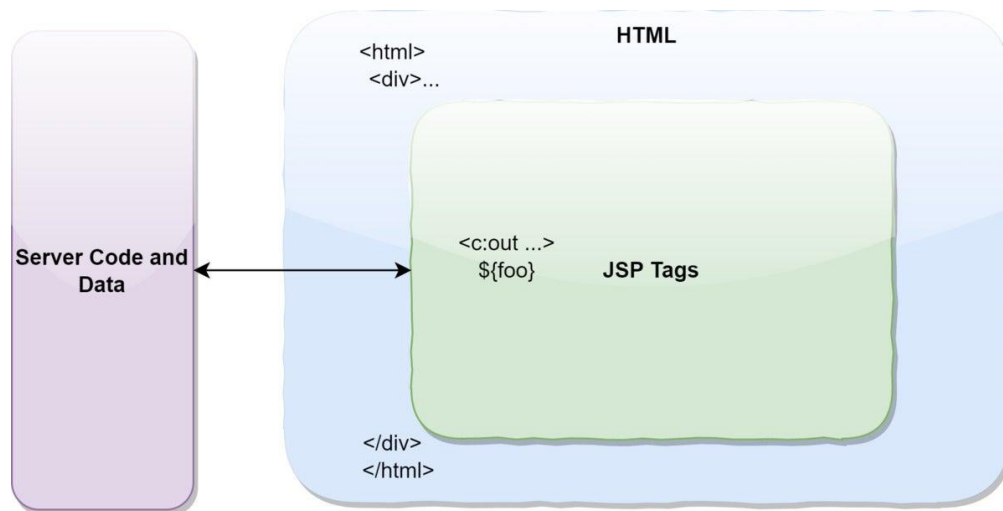


Figure 27: JSP tags (Tyson, 2022).

JSP tags are essentially special keywords that are used to embed Java code within an HTML page (GeeksforGeeks, 2022). These tags allow developers to access variables, iterate through collections, and perform other operations within the JSP page. For example, the `jsp:include` tag can be used to include the contents of another JSP page within the current page.

In terms of how JSP works, when a user requests a JSP page from a server, the server retrieves the JSP file and passes it to a JSP compiler. The compiler then converts the JSP code into a Java servlet, which is executed by the server to generate the HTML output that is sent back to the user's browser (O'Reilly & Associates, 2023). This process happens dynamically, which means that the JSP page can be customized based on user input or other factors.

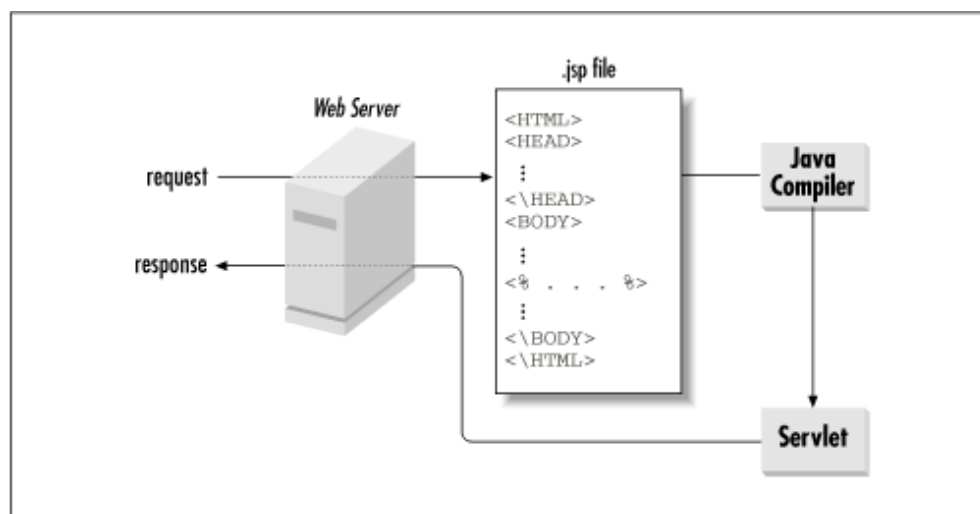


Figure 28: How JSP Works(O'Reilly & Associates, 2023).

The following figure shows the JSP's implementation in the project, homepage.jsp is the JSP file that is being used to discuss in the section.

```
<!-- Login Modal -->
<div class="modal fade" id="loginModal" tabindex="-1" role="dialog" aria-labelledby="loginModalLabel" aria-hidden="true">
  <div class="modal-dialog modal-dialog-centered" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="loginModalLabel">Login</h5>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="modal-body">
        <form action="Login" method="POST">
          <div class="form-group">
            <label for="text">Username</label>
            <input type="text" class="form-control" id="username" name="x" placeholder="Enter Username" required>
          </div>
          <div class="form-group">
            <label for="password">Password</label>
            <div class="input-group">
              <input type="password" class="form-control" id="password" name="y" placeholder="Enter Password" required>
              <span class="input-group-append">
                <span class="input-group-text">
                  <input type="checkbox" onclick="togglePassword()"> Show
                </span>
              </span>
            </div>
          </div>
          <p><input type="submit" class="btn btn-primary" value="Login"></p>
          <br>
          <p>Don't have an account? <a href="#" data-dismiss="modal" data-toggle="modal" data-target="#registrationModal">Register</a></p>
        </form>
      </div>
    </div>
  </div>
</div>
```

Figure 29: Code snippet for login modal.

The code snippet of this JSP page shows that it is a modal and it mainly receive the user inputs for their username and password and submit the JSP page to the server with “POST” method. The form action attribute tells the server what to do when the form is submitted. In this case, the action is to pass the name of the JSP page along with two parameters, “x” and “y” as they are being recognized as the input. Then, the login controller (Login.java) recognizes the .jsp extension and performs the appropriate action. The server then sends the information to the relevant JSP page to take further action based on the parameters it receives. The homepage with login modal is shown at the image below for reference.

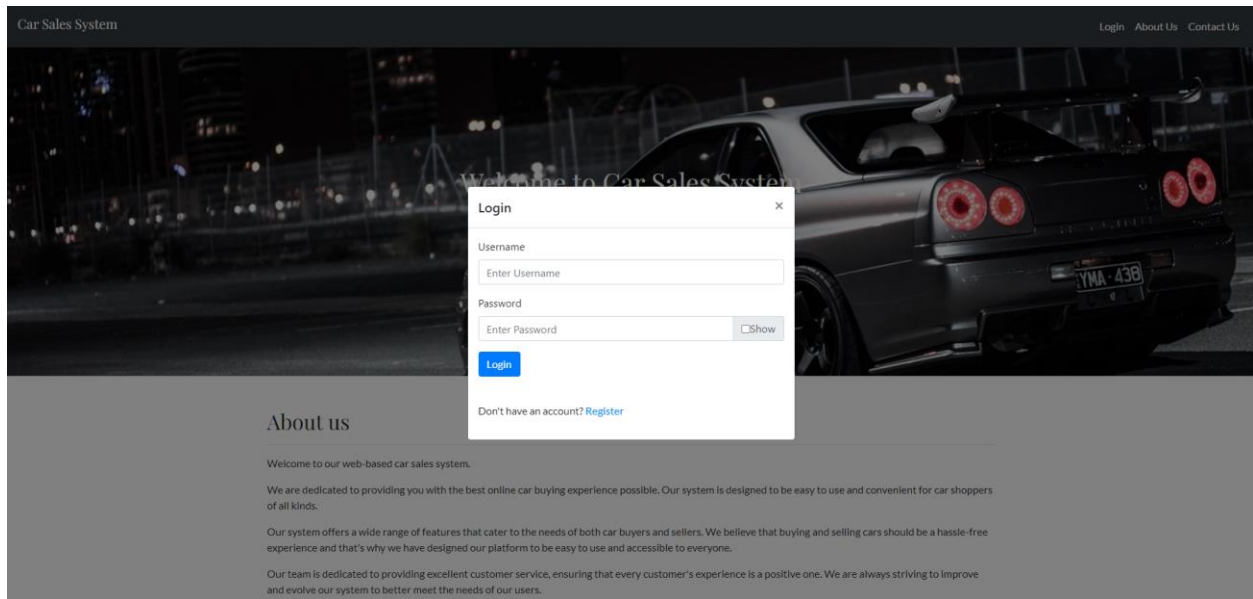


Figure 30: Screenshot for login modal.

2.1.2 Servlet

Java Servlets are small, server-side programs written in the Java programming language that run on a web server and handle incoming requests from client applications (Ravikiran, 2023). They are essentially the backbone of dynamic web applications, allowing developers to create dynamic, interactive web pages that can respond to user input in real-time.

The importance of Java Servlets lies in their ability to handle a wide range of tasks related to web application development, including database access, and session management (Ravikiran, 2023). They are also highly scalable and can handle large amounts of traffic without slowing down or crashing.

Some of the benefits of using Java Servlets include:

- **Portable:** The portability of Java Servlets means that a Servlet program developed on one operating system platform can run seamlessly on another platform (Ravikiran, 2023).
- **Efficient:** Java Servlets provide an immediate response to client requests and can perform well in any environmental condition, regardless of the operating system platform (Ravikiran, 2023).
- **Scalable:** The scalability of Java Servlets is achieved through the use of lightweight threads to handle multiple client requests simultaneously (Ravikiran, 2023).
- **Robust:** Java Servlets are robust due to their implementation of advanced Java features such as the Java Security Manager, Java Garbage Collector, and Exception Handling. These features make Java Servlets resistant to security threats and eliminate memory management issues in real-time (Ravikiran, 2023).

Controller logics are housed within Java Servlets. Application-specific logic is handled by code at the controller layer. This comprises requests like receiving data from web sites, delivering data to service layer classes, and redirecting the user to the next servlet or JSP. Since the controller does not access the database directly, it is done through the facades.

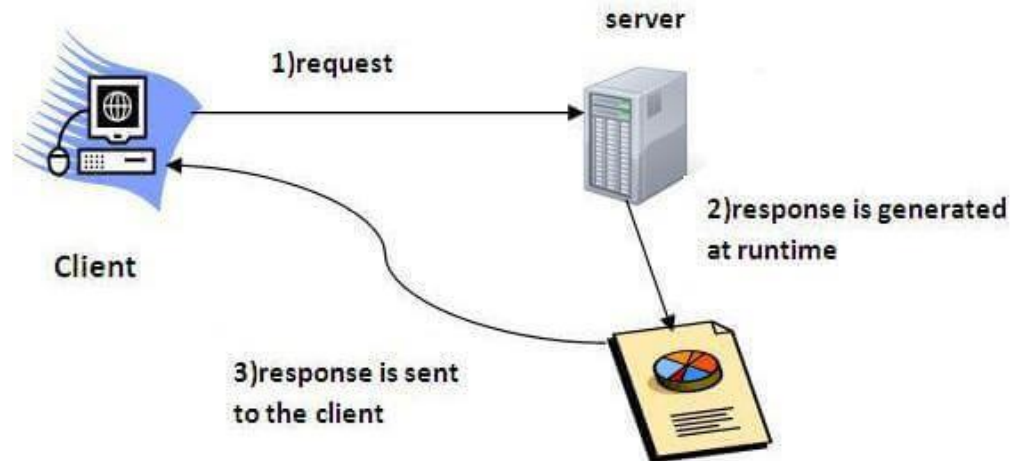


Figure 31: How servlet work (JavaTpoint, 2023a).

In terms of how Java Servlets work, they are typically deployed as part of a web application in a Java servlet container, such as Glassfish. When a user makes a request to the server, the servlet container intercepts the request and passes it on to the appropriate servlet. The servlet then processes the request and generates a response, which is sent back to the client. This process repeats for each incoming request, allowing the servlet to handle multiple users simultaneously. The image below shows part of the code snippet of `register.java` servlet that is implemented in the system.

```

@WebServlet(name = "Register", urlPatterns = {"/Register"})
public class Register extends HttpServlet {

    @EJB
    private SalesmanFacade salesmanFacade;

    @EJB
    private CustomerFacade customerFacade;

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {

            // Get the values submitted in the form
            String username = request.getParameter("x1");
            String password = request.getParameter("x2");
            String email = request.getParameter("x3");
            String name = request.getParameter("x4");
            String age = request.getParameter("x5");
            String hpnumber = request.getParameter("x6");
            String gender = request.getParameter("x7");
            String usertype = request.getParameter("x8");

            try {
                switch (usertype) {
                    case "salesman":
                        // Check if the username already exists
                        Salesman found = salesmanFacade.findBySalesmanUsername(username);
                        if (found != null) {
                            throw new IllegalArgumentException("Username already exists");
                        }
                        // Create a new Salesman instance with the submitted values
                        Salesman salesman = new Salesman();
                        salesman.setUsername(username);
                        salesman.setPassword(password);
                        salesman.setEmail(email);
                        salesman.setName(name);
                        salesman.setAge(age);
                        salesman.setHpnumber(hpnumber);
                        salesman.setGender(gender);
                        salesman.setStatus("Pending");
                        salesman.setUsertype("salesman");
                        salesmanFacade.create(salesman);

```

Figure 32: Code snippet of register modal.

2.2 User Manual

This section provides guidance on how to use the car sales system for managing staff, customer, and salesman. The first page when all 3 users get to see at first glance is the homepage, the homepage briefly talks about the car sales system and it includes 3 items in the navigation bar which is login button, about us section, and contact us section. Login button will prompt the login modal, about us section will just lead user to the section below, same goes for contact us section.

Homepage

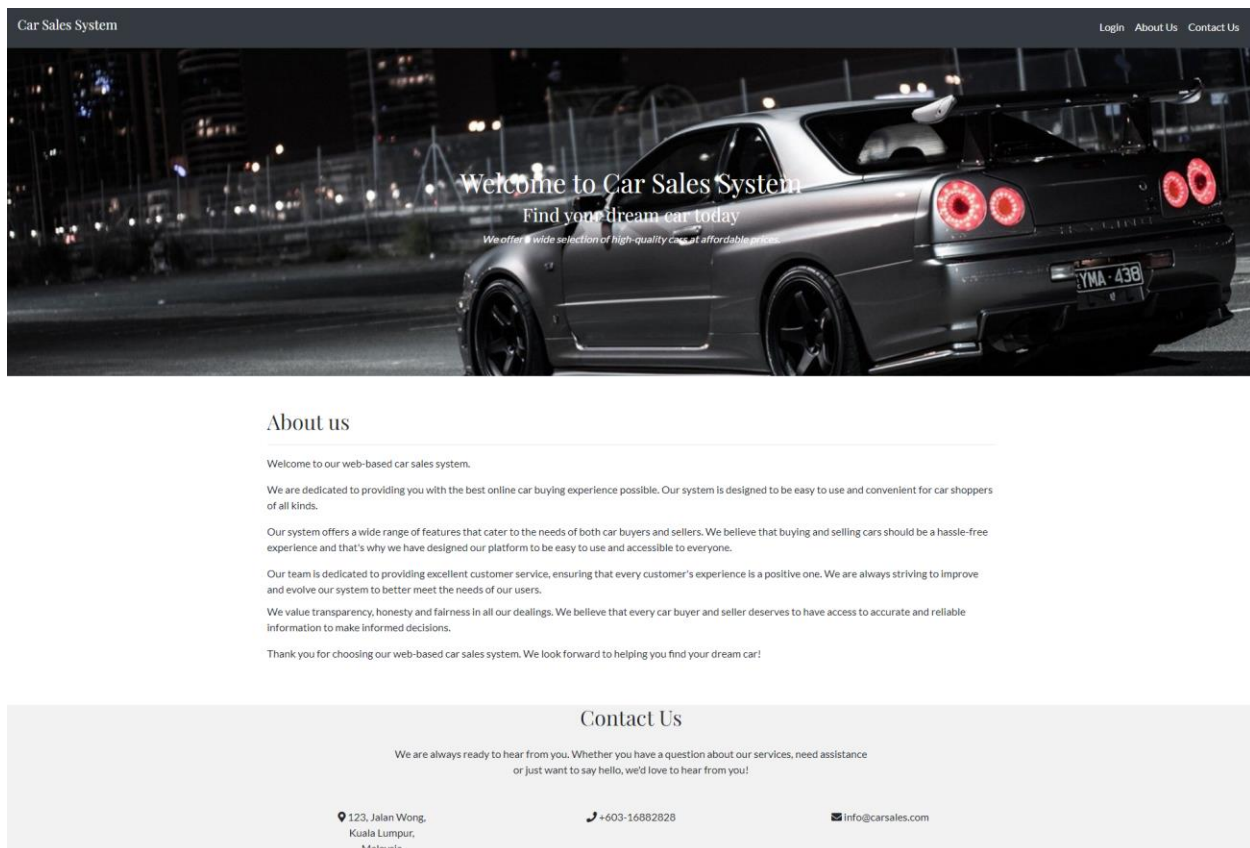


Figure 33: Screenshot of homepage.

Login

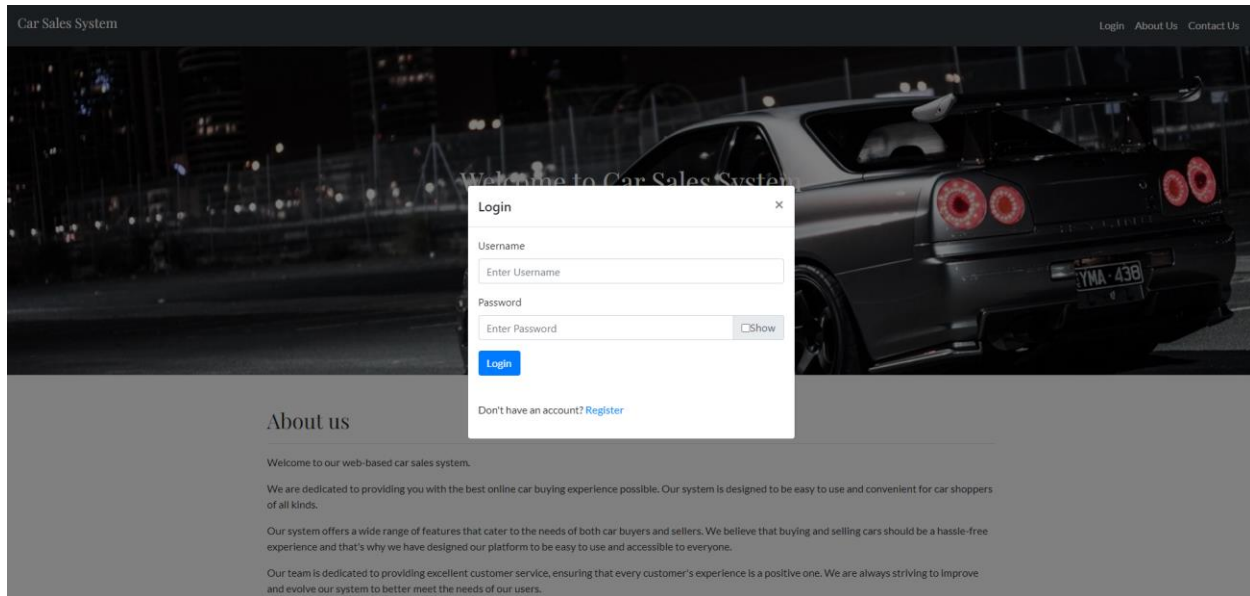


Figure 34: Screenshot of login.

The login modal is a feature in the web-based car sales system that allows all 3 types of users to enter their login credentials in order to access their account. The modal window appears on the screen when the user clicks on the "Login" button. The user enters their username and password, and then clicks the "Login" button to submit the form. If the login is successful, the user is redirected to their dashboard or main page, figure xxx shows the prompt to user once login is successful. If the login is unsuccessful, the user is presented with an error message and given the opportunity to try again, figure xxx displays the prompt to user if they have entered incorrect password. The login modal also includes a link to the registration modal for new users who need to create an account.

Register

Car Sales System

Login About Us Contact Us

Create an Account

Username
Enter Username

Password
Enter Password
Password must be at least 8 characters long, contain at least 1 uppercase letter, 1 lowercase letter, 1 numeric number, and 1 special character.

Email address
Enter email
We'll never share your email with anyone else.

Full Name
Enter Full Name

Age
Enter Age

Phone Number
Enter phone number

Gender: Select gender

User Type: Select user type

Register

About us

Welcome to our web-based car sales system

We are dedicated to providing you with the best experience of all kinds.

Our system offers a wide range of features to enhance your buying and selling experience and that's why we have designed it to be easy to use and convenient for car shoppers.

Buying and selling cars should be a hassle-free experience. We are always striving to improve our system to better meet the needs of our users.

Figure 35: Screenshot of register.

The registration modal is a form that allows users to create an account on the website. It requires the user to fill in several fields, such as their desired username, password, email address, full name, age, phone number, gender, and user type. The form also includes validation for the password field, ensuring that it is at least 8 characters long and contains at least one uppercase letter, one lowercase letter, one numeric character, and one special character. Including validation for the password field is important for security purposes. A strong password is essential for protecting user accounts and sensitive information from unauthorized access. The criteria used for password validation in this form are commonly recommended by security experts, and help ensure that passwords are not easily guessable or susceptible to brute force attacks. By requiring users to create a strong password, the registration form can help prevent security breaches and protect both the user and the website from potential harm. Once the user fills in all the required fields, they can submit the form by clicking the "Register" button.

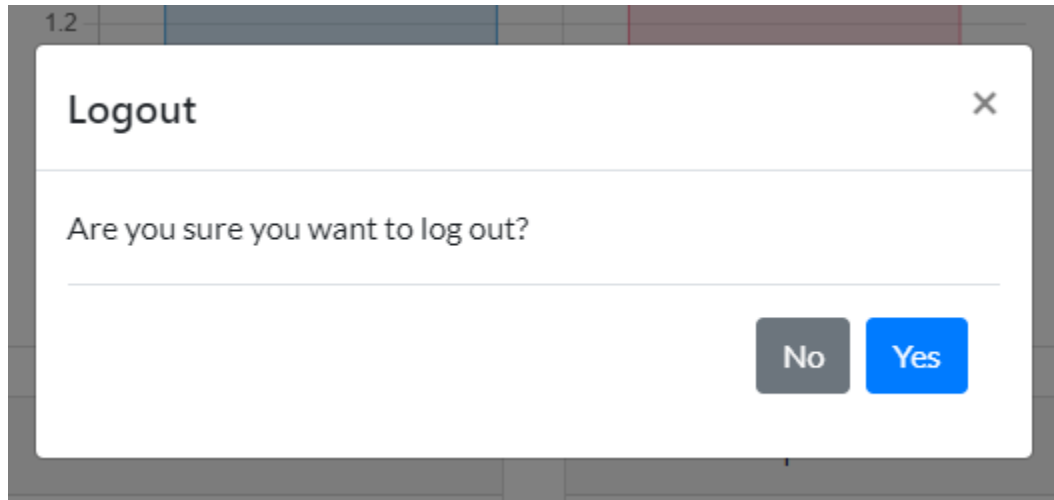
Logout

Figure 36: Screenshot of logout.

The logout modal is a pop-up window available for all 3 usertypes that appears when users clicks on the "Logout" button in the application. The purpose of this modal is to confirm whether the user actually wants to log out of the application. The modal contains a message asking the user if they are sure they want to log out, and two buttons labeled "Yes" and "No". If the user clicks on the "Yes" button, the form is submitted to the server which logs the user out and redirects them to the login page. If the user clicks on the "No" button, the modal is closed and the user stays logged in. The modal provides a simple and clear way for the user to log out of the application while preventing accidental logouts.

2.2.1 Managing Staff

Navigation Bar

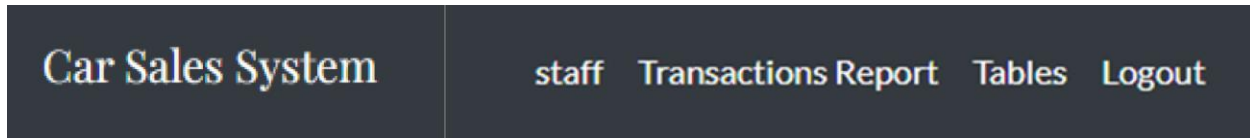


Figure 37: Screenshot of navigation bar.

The Managing Staff Navigation bar is a part of the Car Sales System that allows managing staff to navigate through the different features of the system. The navigation bar includes a logo, a collapsible menu, and various links to different pages such as transaction reports, tables, and a logout button. In addition, the “staff” wording is actually the username, it displays based on username. The navigation bar is designed to be user-friendly, making it easy for staff members to access the features they need quickly and efficiently. The navigation bar is also designed to be visually appealing, with a dark background and white text, which helps it stand out on the screen.

Report & Analysis Section

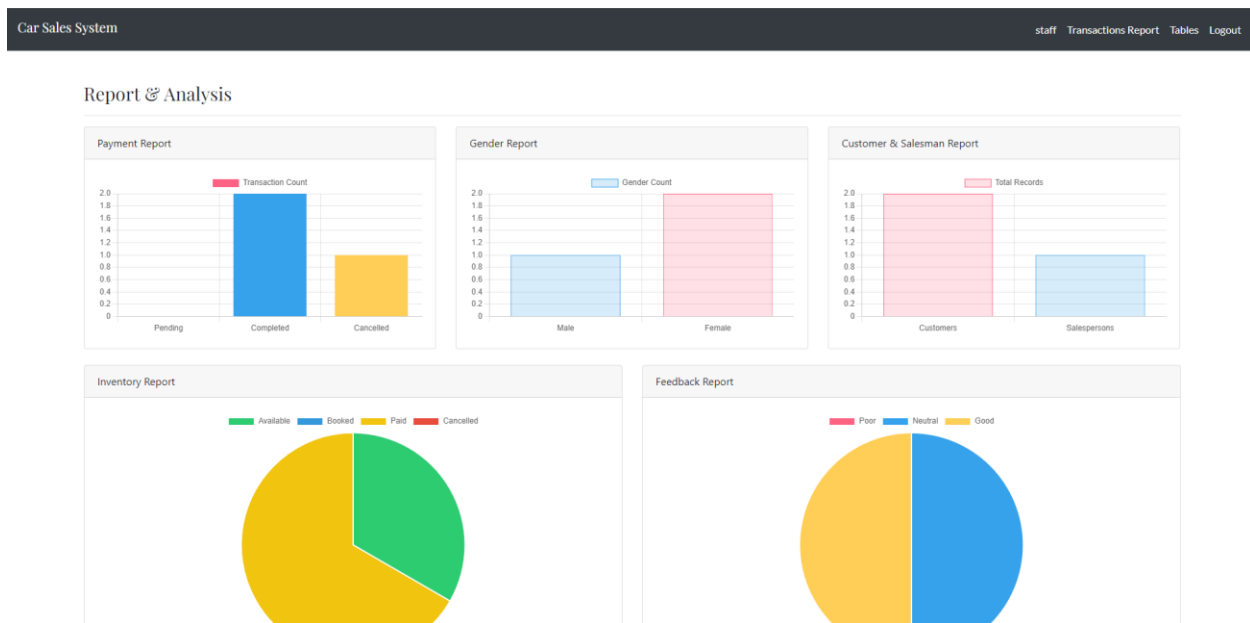


Figure 38: Screenshot of report and analysis section.

The report and analysis section is designed to provide insights into various aspects of the car sales system. It includes five different charts, each with a specific focus. The Payment Report shows a visual representation of payment transactions made by salesman with customers. The

Gender Report highlights the distribution of male and female for salesman and customers. The Customer & Salesman Report shows the number of customers and salesmen in the system. The Inventory Report displays the current inventory car status of the system. For instance, the status has “Available”, “Booked”, “Paid”, “Cancelled”. Finally, the Feedback Report shows a summary of the feedback provided by customers, it categories as “Good”, “Neutral”, “Poor”. Overall, this section is useful for managers to analyze the performance of the car sales system and make informed decisions based on the insights provided by the charts.

Transaction Report

Sales Record ×

Transaction ID	Transaction Date	Car Information	Car Chassis Number	Price	Customer Feedback & Rating	Customer Name	Comment	Transaction Status
55	09/03/2023	Subaru Forester 2020 Pearl White	FB205259241	250000	Transaction Cancelled Transaction Cancelled	Tester	Booking for full loan payment	Cancelled
57	09/03/2023	Toyota 86 2021 Pearl White	FA20QWERTY86	180000	Good Service 5	Tester	Booking for full loan payment	Completed
59	09/03/2023	Subaru Forester 2020 Pearl White	FB205259241	250000	Good Car 3	Tester	Booking for full loan payment	Completed

Figure 39: Screenshot of transaction report.

The Managing Staff transaction report is a modal that displays the sales. It contains a table with information such as transaction ID, transaction date, car information (make, model, year, color), car chassis number, price, customer feedback and rating, customer name, comment, and transaction status. The table is populated with data from the database.

This report is useful for Managing Staff to keep track of sales and transactions made by the salesman. It can also be used to identify trends and patterns in customer behavior, such as which cars are popular, what feedback customers are giving, and what factors influence sales.

View Managing Staff Table

Managing Staff	Salesman	Customer	Car
----------------	----------	----------	-----

Managing Staff Table

Manage your staff here:

[Add Staff](#)

Search by Username, Name, Email, Phone

Staff ID	Staff Username	Staff Password	Name	Email	Phone	Action
7	staff	Staff1234.	Staff	staff@email.com	0122645874	Edit Delete

Figure 40: Screenshot of view managing staff table.

The Managing Staff table is a section of the system that allows users to manage their staff information. It contains a table that displays information such as Staff ID, Staff Username, Staff Password, Name, Email, Phone, and Action. Managing Staff can add new staff members, search for existing staff members, edit their information, and delete staff accounts. The table also includes a search bar for easy navigation and a delete button with confirmation to prevent accidental deletions.

Add Staff

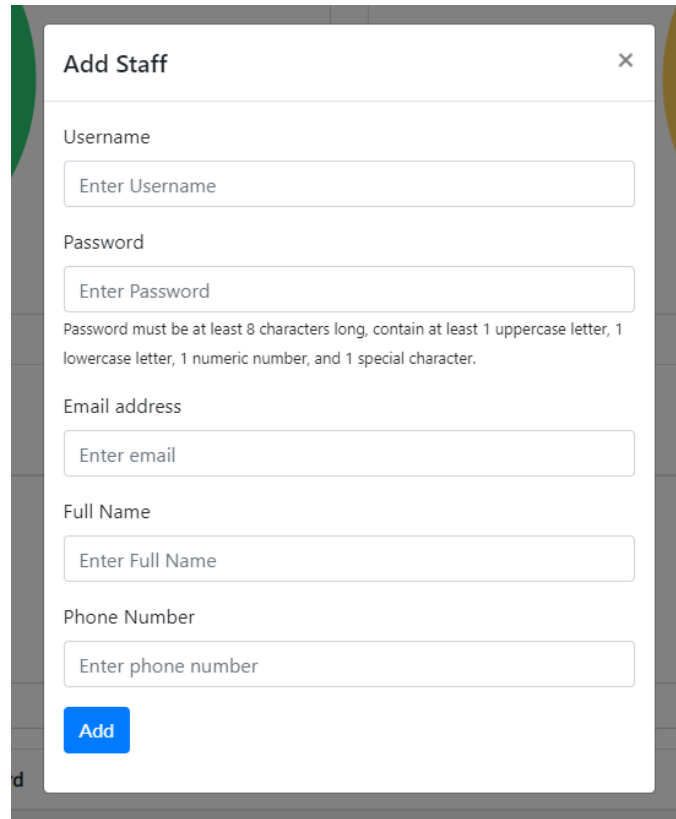
A screenshot of a web application modal titled "Add Staff" with a close button (X) in the top right corner. The modal contains several input fields: "Username" with a placeholder "Enter Username", "Password" with a placeholder "Enter Password" and a validation message below it stating "Password must be at least 8 characters long, contain at least 1 uppercase letter, 1 lowercase letter, 1 numeric number, and 1 special character.", "Email address" with a placeholder "Enter email", "Full Name" with a placeholder "Enter Full Name", and "Phone Number" with a placeholder "Enter phone number". At the bottom left of the modal is a blue "Add" button.

Figure 41: Screenshot of add staff modal.

The Managing Staff Add Staff modal is a user interface element that allows a user with appropriate permissions to add a new staff member to a system. The modal contains several input fields, including username, password, email address, full name, and phone number, that the user must fill out in order to add a new staff member. The modal also includes a password validation feature to ensure that the password meets certain complexity requirements.

Once the user has filled out all the necessary information, they can submit the form and the system will create a new staff member account with the information provided. This modal is a key feature of the Managing Staff functionality and allows administrators to easily add new staff members to the system as needed. It provides a simple and intuitive interface that minimizes the possibility of user error and streamlines the process of adding new staff members.

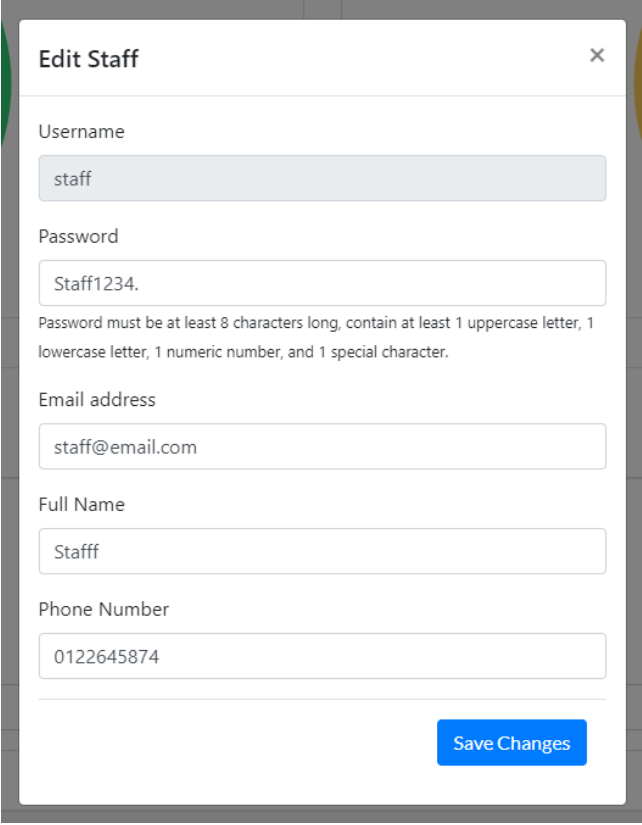
Edit StaffA screenshot of a web application modal titled "Edit Staff" with a close button (X) in the top right corner. The modal contains a form with the following fields: "Username" with the value "staff"; "Password" with the value "Staff1234." and a validation message below it stating "Password must be at least 8 characters long, contain at least 1 uppercase letter, 1 lowercase letter, 1 numeric number, and 1 special character."; "Email address" with the value "staff@email.com"; "Full Name" with the value "Staff"; and "Phone Number" with the value "0122645874". At the bottom right of the form is a blue button labeled "Save Changes".

Figure 42: Screenshot of edit staff modal.

The Managing Staff edit Staff modal is a pop-up window that allows an authorized user to edit the details of a staff member. The modal displays a form with pre-filled fields for the current details of the staff member. The user can then modify any of these fields and click on the "Save Changes" button to update the details. The form includes fields for the staff member's password, email address, full name, and phone number. Validations have been implemented to ensure all the fields has to be filled in order to proceed. The password field has a pattern requirement, which ensures that the password is at least 8 characters long, contains at least one uppercase letter, one lowercase letter, one numeric number, and one special character. The modal helps authorized users to manage the details of staff members in an efficient and secure manner.

Delete Staff

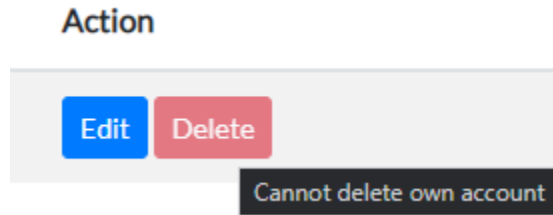


Figure 43: Screenshot of delete staff button

Generally, staff members can be deleted by clicking on the "Delete" button next to the respective staff member. However, staff members cannot delete their own account as this would cause issues with the system. A confirmation prompt will appear to confirm whether the user wants to proceed with the deletion. If confirmed, the staff member's account will be permanently removed from the system.

View Salesman Table

Managing Staff Salesman Customer Car

Salesman Table
Manage salesman here:

Search by ID, Name, Email, Gender, Hpnumber, or Status Search

ID	Username	Password	Name	Age	Email	Phone	Gender	Status	Action
2	salesman	Abcd1234.	Salesman	22	salesman@email.com	0198765431	Female	Active	Edit Delete
251	salesman1	Abcd1234.1	Salesman1	21	salesman1@email.com	0198765432	Male	Pending	Approve Edit Delete

Figure 44: Screenshot of view salesman table.

The Salesman table is a section in the managing staff dashboard that allows the managing staff to manage the information of salesman working for the company. The table displays information such as the salesman's ID, username, password, name, age, email, phone number, gender, and status.

The managing staff can search for specific salespeople by entering keywords in the search bar provided. The table also has a set of actions that can be performed on each salesman's information, such as approving their account (if their status is pending), editing their details, and deleting their account. The Salesman table provides an easy way to manage the information of the salespeople working for the company.

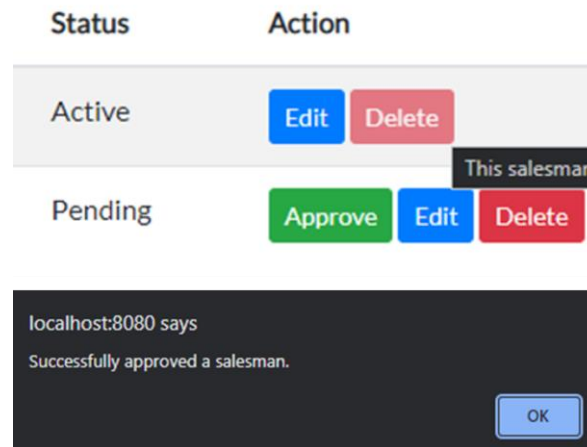
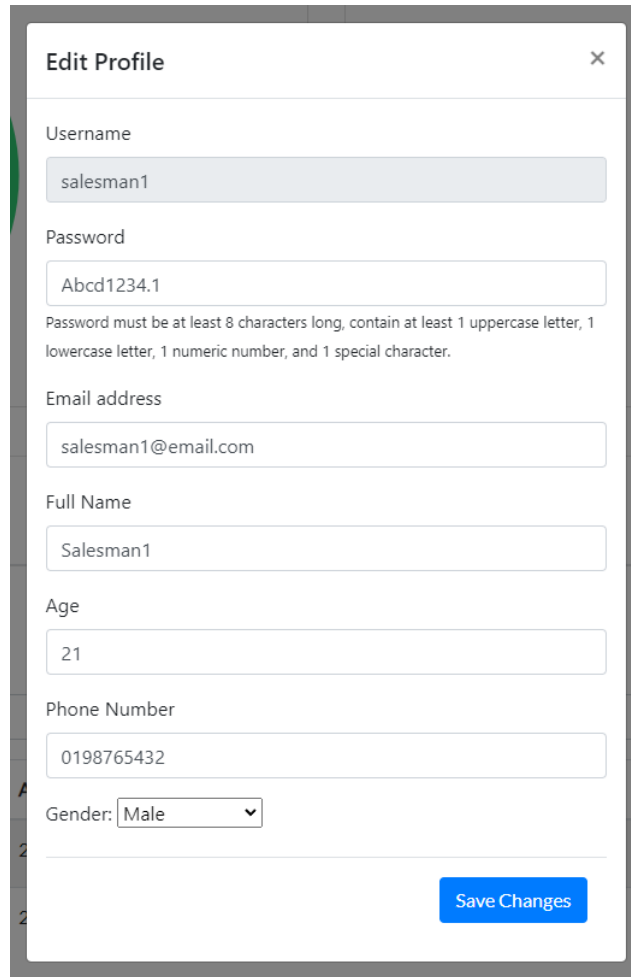
Approve Salesman

Figure 45: Screenshot of approve salesman function.

This logic checks if the status of the salesman is 'Pending' and displays an 'Approve' button next to their information in the Salesman table. Clicking on this button will trigger an action that will update the salesman's status to 'Active', indicating that they have been approved.

Edit Salesman

Edit Profile ×

Username
salesman1

Password
Abcd1234.1
Password must be at least 8 characters long, contain at least 1 uppercase letter, 1 lowercase letter, 1 numeric number, and 1 special character.

Email address
salesman1@email.com

Full Name
Salesman1

Age
21

Phone Number
0198765432

Gender: Male ▼

Save Changes

Figure 46: Screenshot of edit salesman modal.

The edit salesman modal is a pop-up window that allows a managing staff member to edit the details of a salesman's profile. It contains a form where the managing staff member can update the salesman's password, email, name, age, phone number, and gender. The salesman's username is displayed but cannot be edited. The form also includes validation for the password, email, and gender fields to ensure that they meet certain requirements. Once the managing staff member has made the necessary changes, they can click the "Save Changes" button to update the salesman's profile.

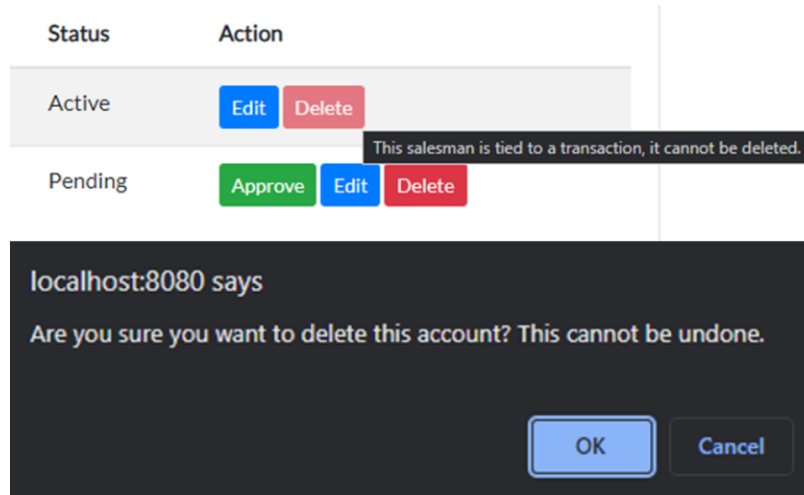
Delete Salesman

Figure 47: Screenshot of delete salesman function.

The delete logic checks whether a salesman is tied to a transaction or not. If the salesman is tied to a transaction, then the delete button is disabled and the user cannot delete the salesman account. If the salesman is not tied to any transaction, then the delete button is enabled and the user can delete the salesman account by clicking on the button. A confirmation message is displayed before deleting the account to ensure that the user is sure about the deletion as it cannot be undone.

View Customer Table

ID	Username	Password	Name	Age	Email	Phone	Gender	Status	Action
1	test	Test1234.	Tester	21	test@email.com	0123456789	Male	Active	Edit Delete
51	test1	Test1234.1	Tester1	21	test1@email.com	0123344556	Female	Pending	Approve Edit Delete

Figure 48: Screenshot of view customer table.

The customer table is a section of the managing staff dashboard that allows users to view and manage customer information. The table displays customer details such as ID, username, name, age, email, phone number, gender, and actions that can be taken for each customer. Users can search for a specific customer by ID, name, email, gender, or phone number using the search bar. The actions that can be taken for each customer include editing their information, deleting their account.

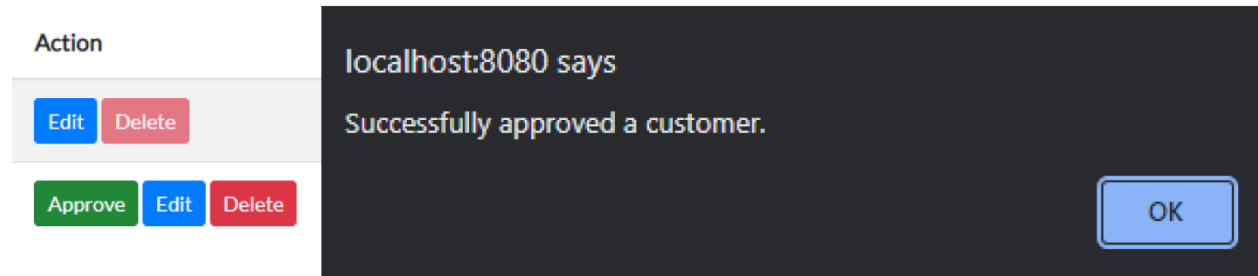
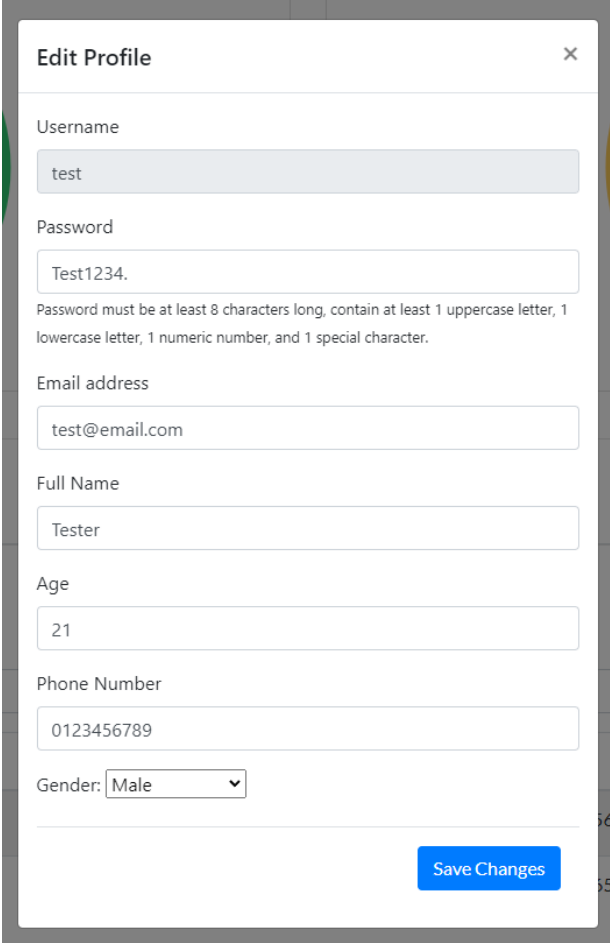
Approve Customer

Figure 49: Screenshot of approve customer.

This logic checks if the status of the customer is 'Pending' and displays an 'Approve' button next to their information in the Customer table. Clicking on this button will trigger an action that will update the salesman's status to 'Active', indicating that they have been approved.

Edit Customer



The screenshot shows a modal window titled "Edit Profile" with a close button (X) in the top right corner. The form contains the following fields and controls:

- Username:** A text input field containing the value "test".
- Password:** A text input field containing the value "Test1234.". Below the field is a password requirement note: "Password must be at least 8 characters long, contain at least 1 uppercase letter, 1 lowercase letter, 1 numeric number, and 1 special character."
- Email address:** A text input field containing the value "test@email.com".
- Full Name:** A text input field containing the value "Tester".
- Age:** A text input field containing the value "21".
- Phone Number:** A text input field containing the value "0123456789".
- Gender:** A dropdown menu with "Male" selected.
- Save Changes:** A blue button located at the bottom right of the form.

Figure 50: Screenshot of edit customer.

The edit customer modal is a user interface component that allows a managing staff member to modify the personal information of a customer. The modal is triggered when the managing staff member clicks on the "Edit Profile" button associated with a particular customer in the managing staff dashboard. The modal displays a form with fields for the managing staff member to enter or modify the customer's password, email, name, age, phone number, and gender. The form is pre-populated with the customer's current information for easy reference, and the username is marked as read-only to prevent accidental changes.

Once the managing staff member has made the desired changes to the customer's information, they can click the "Save Changes" button to submit the form and update the customer's record in the system. This modal is an important feature for managing staff members who need to keep customer records up to date and accurate.

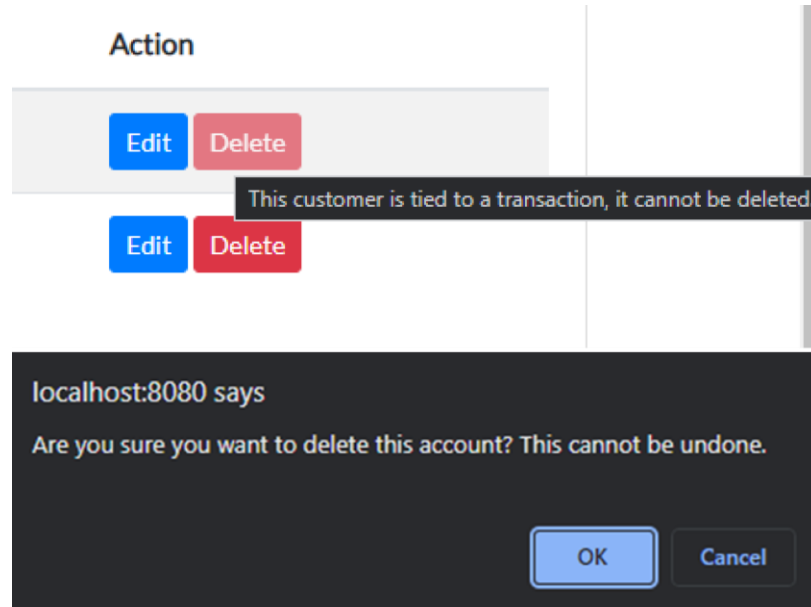
Delete Customer

Figure 51: Screenshot of delete customer.

This delete logic for customer involves disabling the delete button if the customer is tied to a transaction. This is to prevent accidental deletion of important information that could cause errors in the system. If the customer is not tied to any transaction, the delete button is enabled and the user can delete the customer account by clicking the button and confirming the action.

View Car Table

Make	Model	Chassis	Year	Color	Price	Description	Status	Set Car Status	Actions
Subaru	WRX STI	EJ20XVAB65286	2018	World Rally Blue	250000	Blue Subaru WRX STI with modified exhaust	Available	Available	Edit Delete
Subaru	Forester	FB205259241	2020	Pearl White	250000	Family Car	Paid	Available	Edit Delete
Toyota	86	FA20QWERTY86	2021	Pearl White	180000	White Toyota 86 with modified exhaust	Paid	Available	Edit Delete

Figure 52: Screenshot of view car table.

The car table displays car information such as make, model, chassis, year, color, price, description, and status. The table provides various functionalities, including adding new cars, searching for cars by specific criteria, editing and deleting car details. The status column indicates whether the car is available, canceled or not available for editing and deletion. The set car status button enables changing the car's status from canceled to available. The edit button allows modifying car details, while the delete button enables deleting car details, but only when the car status is available.

Add Car

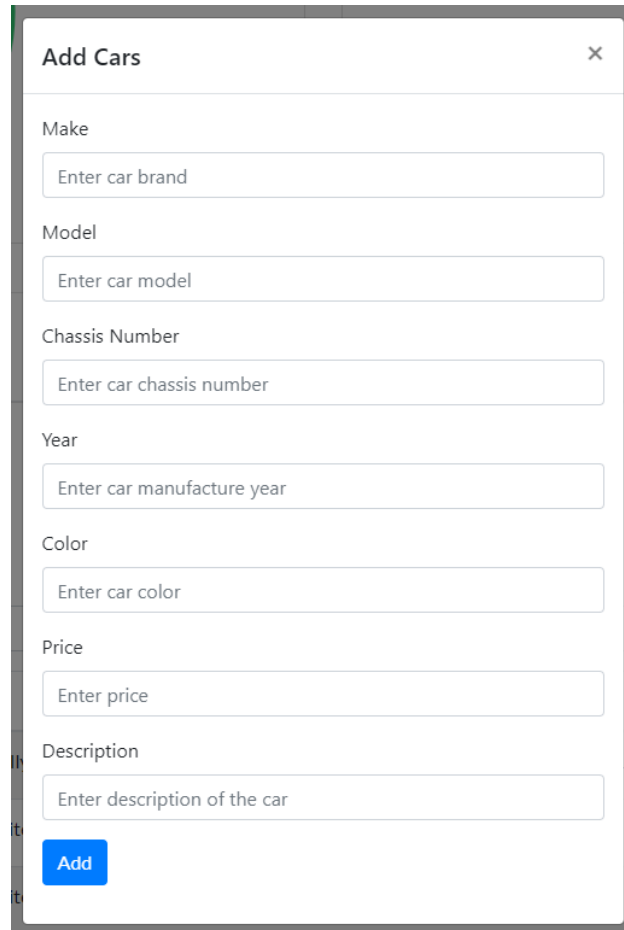
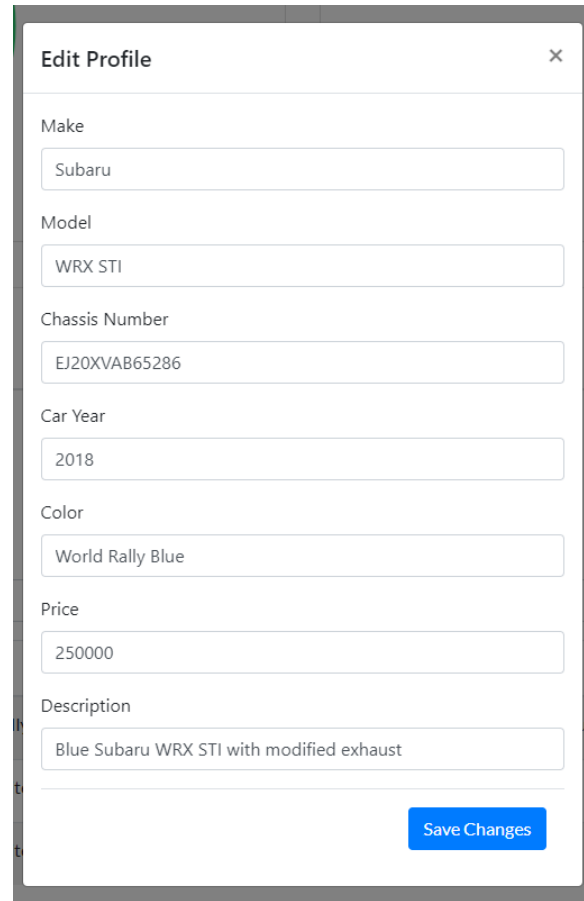
A screenshot of a web application modal titled "Add Cars" with a close button (X) in the top right corner. The modal contains a vertical stack of form fields, each with a label and a text input box with a placeholder. The fields are: "Make" (placeholder: "Enter car brand"), "Model" (placeholder: "Enter car model"), "Chassis Number" (placeholder: "Enter car chassis number"), "Year" (placeholder: "Enter car manufacture year"), "Color" (placeholder: "Enter car color"), "Price" (placeholder: "Enter price"), and "Description" (placeholder: "Enter description of the car"). At the bottom of the modal is a blue button labeled "Add".

Figure 53: Screenshot of add car.

The add car modal includes various form fields for the user to input information such as the make, model, year, color, price, chassis number, and a description of the car. The modal also includes a submit button to add the entered data to the system. Validations have been implemented for all input fields to ensure the no empty input when submit the modal. The modal has clear labels and placeholders for each form field.

Edit Car

The screenshot shows a modal dialog titled "Edit Profile" with a close button (X) in the top right corner. The form contains several input fields, each with a label above it: "Make" (Subaru), "Model" (WRX STI), "Chassis Number" (EJ20XVAB65286), "Car Year" (2018), "Color" (World Rally Blue), "Price" (250000), and "Description" (Blue Subaru WRX STI with modified exhaust). A blue "Save Changes" button is located at the bottom right of the form.

Figure 54: Screenshot of edit car.

The edit car modal is a form that allows the user to edit a car's information. It is used in a Managing Staff Dashboard Function where the staff can manage the cars' information such as make, model, chassis number, car year, color, price, and description. The form includes input fields for each piece of information and a submit button to save changes made. Validations have been implemented for all input fields to ensure that there is no empty input when the modal is submitted. The modal is triggered by a button click and displays a modal dialog in the center of the screen.

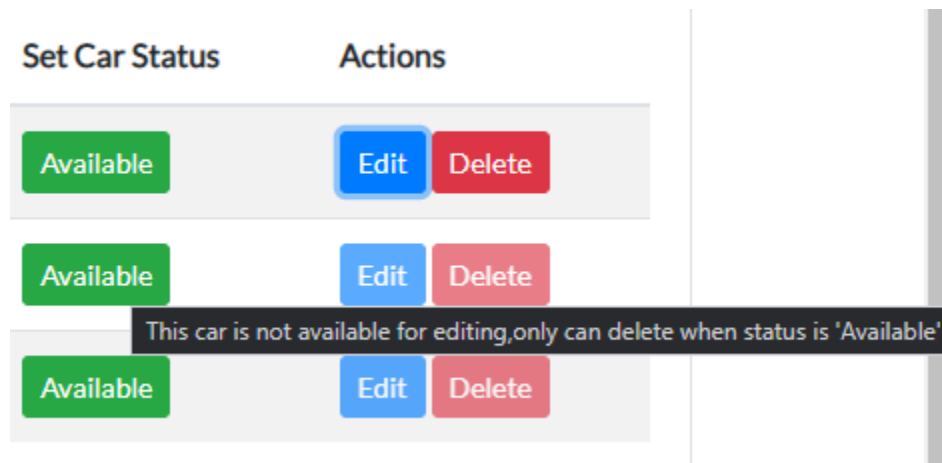


Figure 55: Screenshot of shaded edit button.

The edit button in actions has a logic for determining whether a car can be edited or not. If the car's status is "Available," then the "Edit" button will be clickable, and it will open the "Edit Car" modal. On the other hand, if the car's status is not "Available," then the "Edit" button will be disabled and will display a tooltip indicating that the car cannot be edited because it is not available. This logic helps ensure that users can only edit cars that are available and prevent any unintended changes to non-available cars.

Delete Car

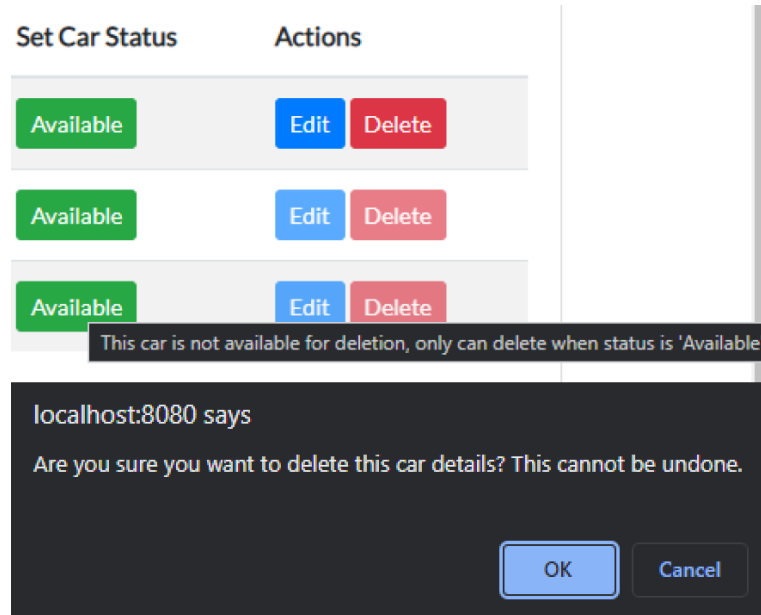


Figure 56: Screenshot of delete car.

The delete logic allows a managing staff user to delete a car entry from the system, but only if the car's status is set to "Available". If the car is not available, such as it is booked or otherwise unavailable, the delete button will be disabled and the user will not be able to delete the car. When the delete button is clicked, a confirmation prompt will appear asking the user to confirm whether they want to proceed with the deletion. Once the user confirms, the car entry will be permanently removed from the system and cannot be undone.

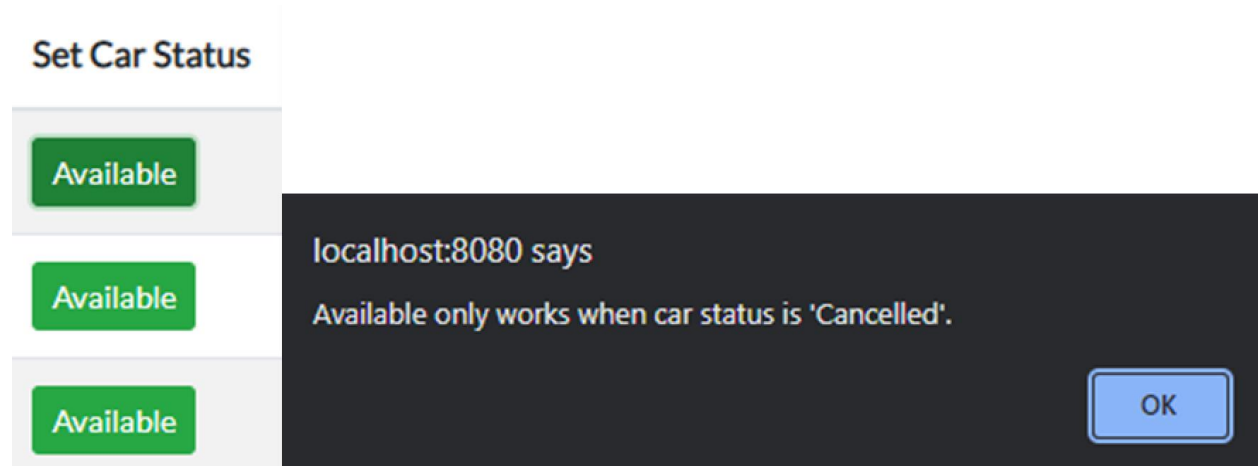
Set Car Status

Figure 57: Screenshot of set car status.

This function is used to set the status of a car to "Available". If the status of the car is already "Cancelled", then an "Available" button will be enabled which, when clicked, it will redirect the user to a function that sets the status of the car to "Available". If the status is anything other than "Cancelled", then a disabled "Available" button will be displayed, and when clicked, it will show an alert to the user indicating that the car cannot be set to "Available" at this time. This logic ensures that the user cannot set the status of a car to "Available" if it is already available or if it has not been cancelled first.

2.2.2 Salesman

When user first register as salesman, salesman do not have access to the system yet, salesman will have to wait for approval from the Managing Staff in order to use the car sales system. Once approved, the system will prompt successful message for salesman.

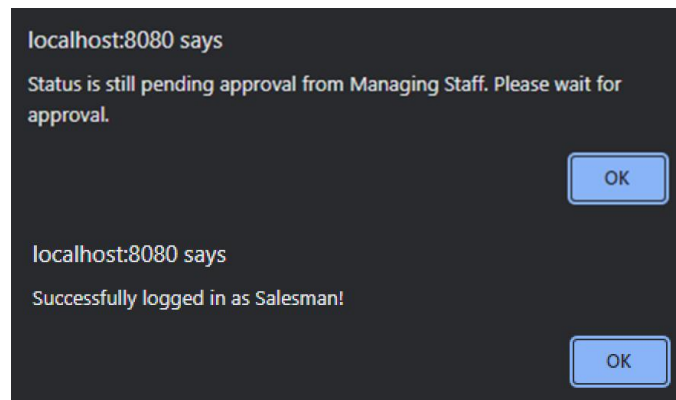


Figure 58: Screenshot of salesman login message handling.

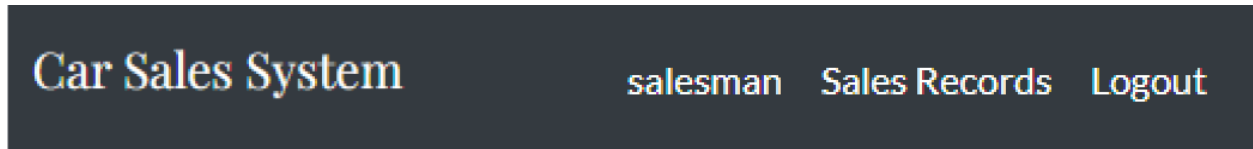
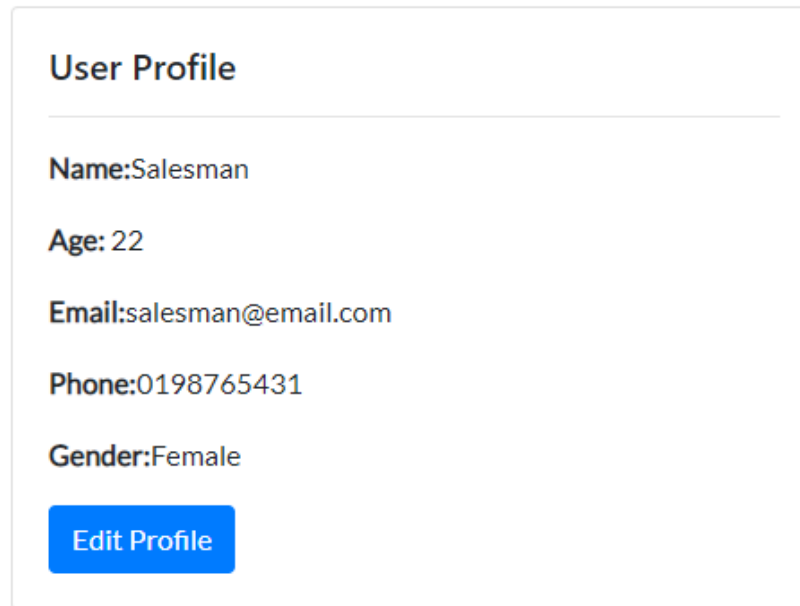
Navigation Bar

Figure 59: Screenshot of salesman navigation bar.

The navigation bar for salesman user type is a responsive navigation bar that contains the logo of the Car Sales System and links to the user's sales records and the logout modal. The sales record link will open a modal that displays the sales records of the salesman. The user's username is also displayed on the navigation bar. When the user clicks the logout link, the logout modal will appear asking the user to confirm if they want to log out.

View Salesman Profile

User Profile

Name:Salesman

Age: 22

Email:salesman@email.com

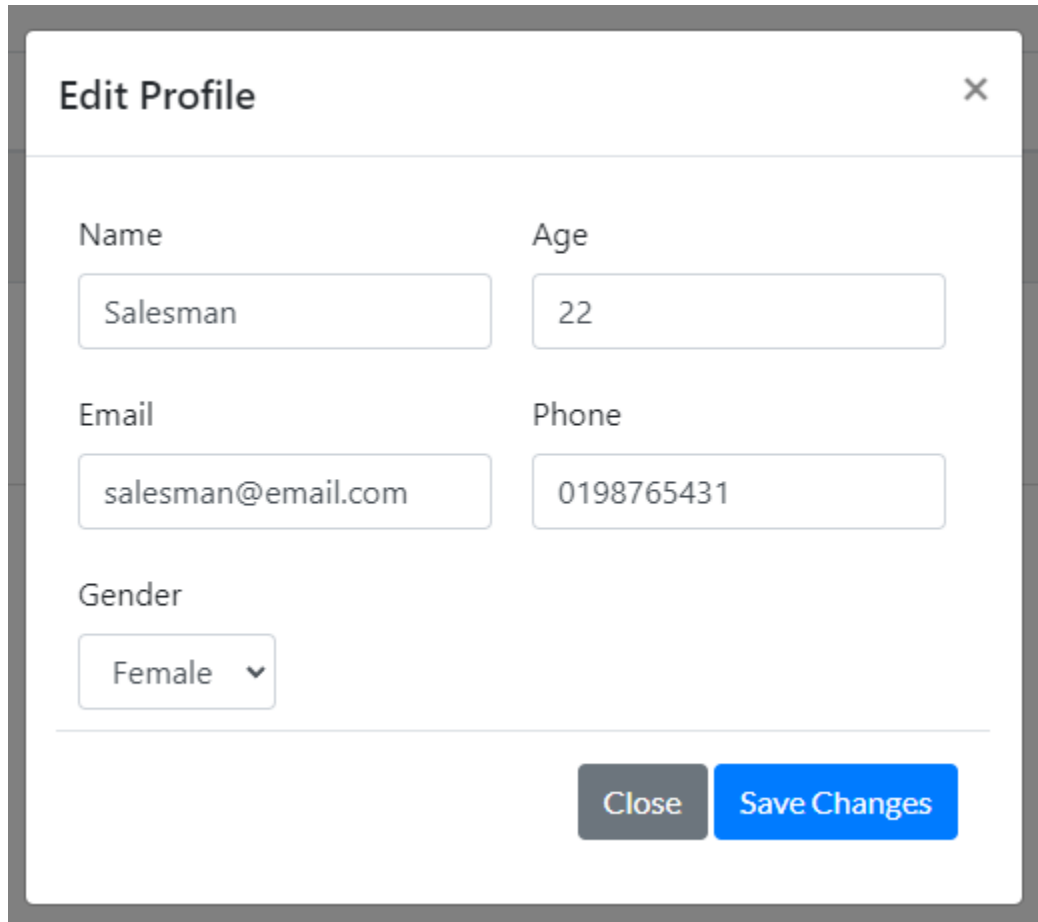
Phone:0198765431

Gender:Female

[Edit Profile](#)

Figure 60: Screenshot of view salesman profile.

The user profile section displays the personal information of the salesman user, such as their name, age, email, phone number, and gender. The information is retrieved from the database and displayed in a card layout. The salesman user can also edit their profile by clicking on the "Edit Profile" button, which will open a modal where they can update their personal information.

Edit Profile

The screenshot shows a modal window titled "Edit Profile" with a close button (X) in the top right corner. The form contains five input fields arranged in three rows. The first row has "Name" (text input with "Salesman") and "Age" (text input with "22"). The second row has "Email" (text input with "salesman@email.com") and "Phone" (text input with "0198765431"). The third row has "Gender" (dropdown menu with "Female" selected). At the bottom right, there are two buttons: "Close" (grey) and "Save Changes" (blue).

Figure 61: Screenshot of edit salesman profile.

The edit modal allows the salesman to edit their personal information. It has a form that displays the salesman's current information for their name, age, email, phone number, and gender, and allows them to modify any of these fields. Validations have been implemented for all input fields to ensure the no empty input when submit the modal. The form includes a "Save Changes" button to submit the modified information and a "Close" button to cancel the operation.

View Available Car Table

Available Cars

Sales

Available Cars

Book the car for your customer here:

Set Car Available

Make	Model	Chassis	Year	Color	Price	Description	Actions
Subaru	WRX STI	EJ20XVAB65286	2018	World Rally Blue	250000	Blue Subaru WRX STI with modified exhaust	Book

Figure 62: Screenshot of view available car table.

The available car table section displays a list of cars that are available for sale. It includes columns for make, model, chassis, year, color, price, description, and actions. The actions column contains a "Book" button that allows the salesman to book the car for a customer by opening a sales modal. If there are no available cars, the table will display a message indicating that no data is available. There is also a "Set Car Available" button to let salesman set "Cancelled" status car to available once again.

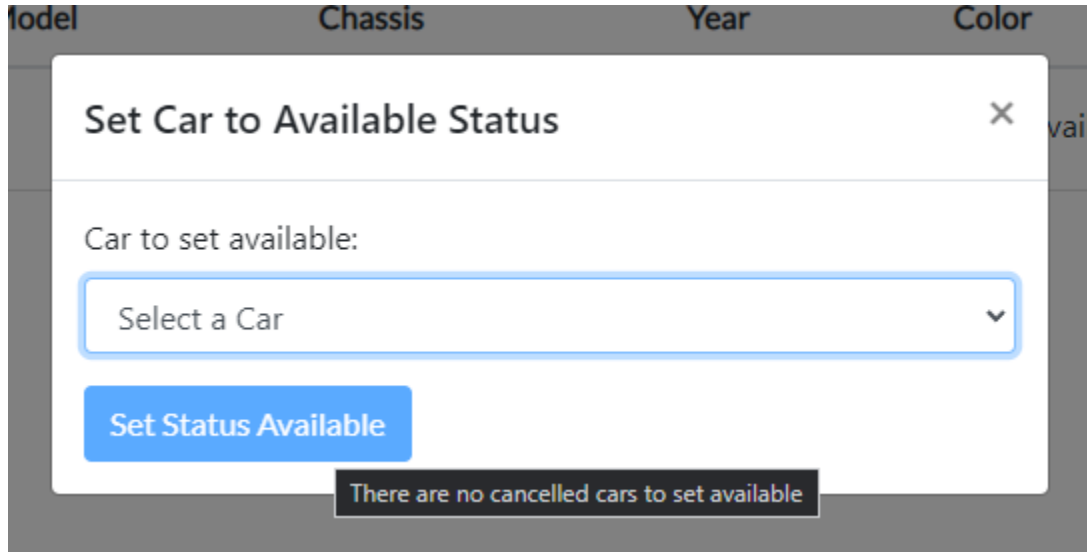
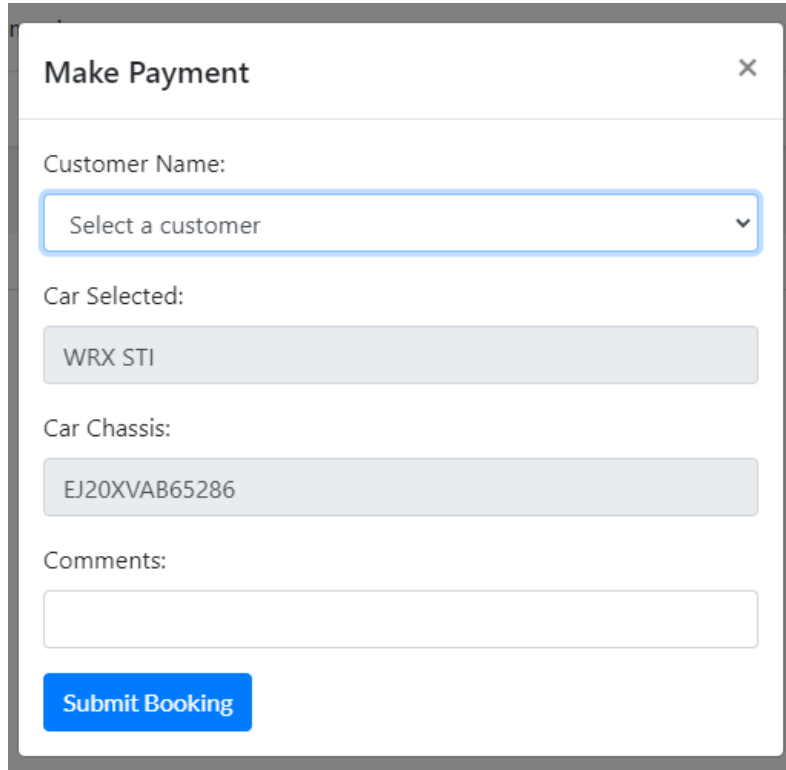
Set Car Status Available

Figure 63: Set car status available.

This modal works when the car status is “Cancelled”, salesman can select the car that is “Cancelled” and set the car status back to “Available”. Otherwise, if there are no “Cancelled” status car, a tooltip will present and show user that there are no cancelled cars to set available at the moment and disabled the button.

Set Car Status Booked

The screenshot shows a modal window titled "Make Payment" with a close button (X) in the top right corner. The modal contains the following fields:

- Customer Name:** A dropdown menu with the placeholder text "Select a customer" and a downward arrow.
- Car Selected:** A text field displaying "WRX STI".
- Car Chassis:** A text field displaying "EJ20XVAB65286".
- Comments:** An empty text input field.
- Submit Booking:** A blue button with white text.

Figure 64: Screenshot of set car status booked modal.

The sales booking modal is a pop-up window that appears when the user clicks on the "Book" button in the available car table section. It allows the user to select a customer from a drop-down list, input any comments they may have, and submit the booking. To ensure that there is no empty input when the modal is submitted, validations have been implemented for all input fields. The modal also displays information about the selected car, such as the make, model, chassis, and price. Once the user submits the booking, it will be processed and recorded in the system, and the car will be marked as "Booked" in the available car table section.

View Car Sales Table

[Available Cars](#)
[Sales](#)

Car Sales

View your Sales here:

Make	Model	Chassis	Year	Color	Price	Description	Status	Actions(Set Status)
Toyota	86	FA20QWERTY86	2020	Pearl White	180000	White Toyota 86 with modified exhaust	Paid	No action needed
Subaru	WRX STI	EJ20XVAB65286	2018	World Rally Blue	250000	Blue Subaru WRX STI with modified exhaust	Booked	Pay Cancel
Subaru	WRX STI	EJ20XVAB65286	2018	World Rally Blue	250000	Blue Subaru WRX STI with modified exhaust	Booked	Pay Cancel

Figure 65: Screenshot of view car sales table.

The car sales table displays a list of cars that have been paid or booked or cancelled, along with their details such as make, model, year, color, price, and description. Additionally, the table provides an option to take action on the car status, which is paying for a booked car. The table is dynamically populated based on the transactions in the system, and if there are no transactions, a message is displayed indicating that there is no data available. The cancel button work when customer does not want to purchase the car anymore and salesman has to cancel the booking.

Set Car Status Paid

Subaru	Forester	FB205259241	2020	Pearl White	250000	Family Car	Paid	No action needed
--------	----------	-------------	------	-------------	--------	------------	------	------------------

Figure 66: Screenshot of set car status paid.

The logic to set car status to Paid is executed when the user clicks on the "Pay" button in the "Actions" column of the "Car Sales" table. This is triggered only when the car status is "Booked". When the user clicks on "Pay", a request is sent to the server with the transaction ID and car ID as parameters. The server then updates the car status to "Paid" and the transaction status to "Completed".

Set Car Status Cancel

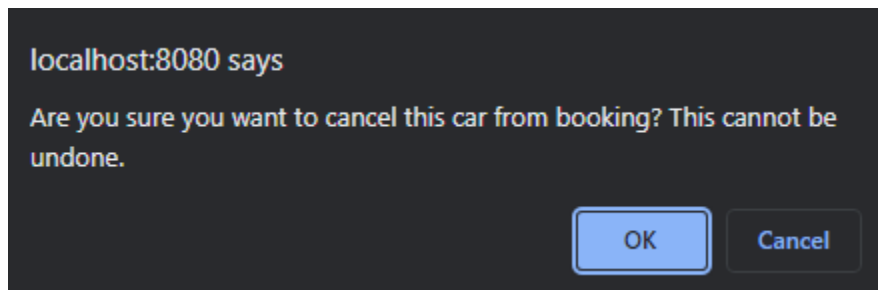


Figure 67: Screenshot of set car status cancel.

This function is responsible for allowing the user to cancel a booking and set the car status to "Cancelled". When a car is booked, the user can click on the "Cancel" button which will prompt them with a confirmation message. If the user confirms the cancellation, the car status will be updated to "Cancelled" and the car will become available for other users to book. The cancelled status can be seen at the figure below.

Set Car Status Available

Available Cars

Sales

Car Sales

View your Sales here:

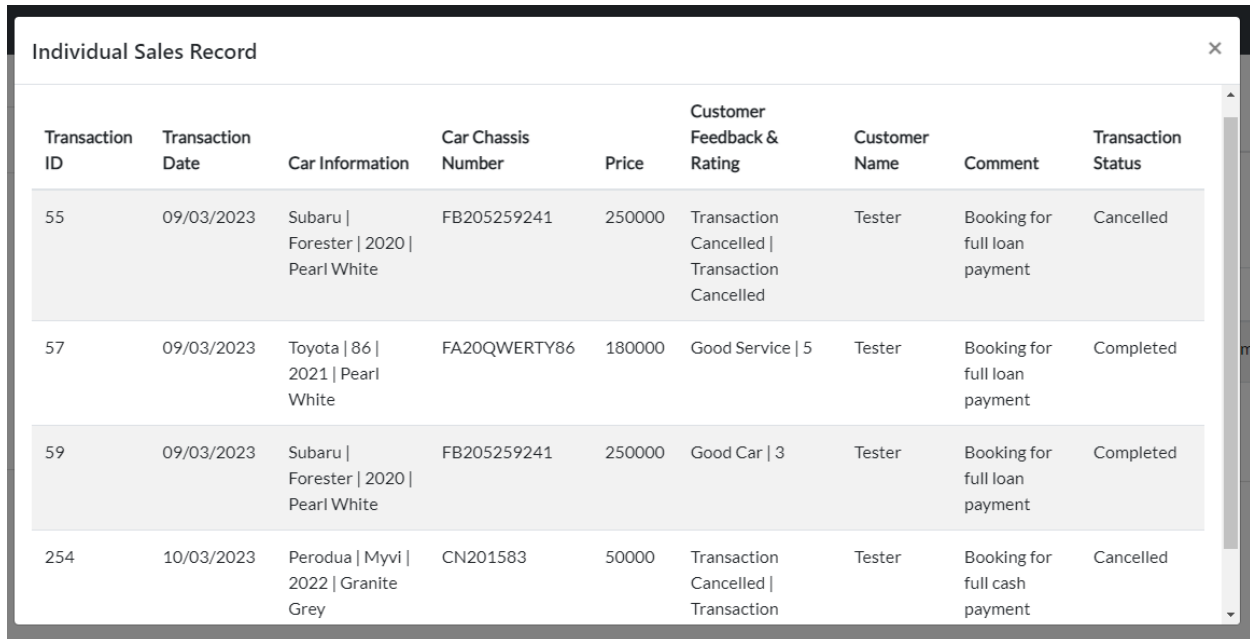
Make	Model	Chassis	Year	Color	Price	Description	Status	Actions(Set Status)
Toyota	86	FA20QWERTY86	2021	Pearl White	180000	White Toyota 86 with modified exhaust	Paid	No action needed
Subaru	Forester	FB205259241	2020	Pearl White	250000	Family Car	Paid	No action needed
Perodua	Myvi	CN201583	2022	Granite Grey	50000	Myvi King	Cancelled	Available

Figure 68: Screenshot of set car status available.

In this function, the logic behind is checking the current status of a car sales system, and if the status is "Cancelled," it provides an option for a user to change the status to "Available."

The logic behind setting a car's status to "Available" is straightforward: it simply means that the car is no longer booked or reserved and is now available for other customers to purchase. Once the status is changed to "Available," the car should show up as an option for other customers to purchase.

View Sales Records



The screenshot shows a modal window titled "Individual Sales Record" with a close button (X) in the top right corner. Inside the modal is a table with the following data:

Transaction ID	Transaction Date	Car Information	Car Chassis Number	Price	Customer Feedback & Rating	Customer Name	Comment	Transaction Status
55	09/03/2023	Subaru Forester 2020 Pearl White	FB205259241	250000	Transaction Cancelled Transaction Cancelled	Tester	Booking for full loan payment	Cancelled
57	09/03/2023	Toyota 86 2021 Pearl White	FA20QWERTY86	180000	Good Service 5	Tester	Booking for full loan payment	Completed
59	09/03/2023	Subaru Forester 2020 Pearl White	FB205259241	250000	Good Car 3	Tester	Booking for full loan payment	Completed
254	10/03/2023	Perodua Myvi 2022 Granite Grey	CN201583	50000	Transaction Cancelled Transaction	Tester	Booking for full cash payment	Cancelled

Figure 69: Screenshot of view sales records.

The "View Sales Records" feature in the salesman section allows the salesman to view individual sales records. When the salesman clicks on the "View Sales Record" on the navigation bar, a modal window pops up showing the sales record that is done by the salesman. The modal displays a table that includes transaction details such as the transaction ID, transaction date, car information, car chassis number, price, customer feedback and rating, customer name, comment, and transaction status. The salesman can use this information to get an overview of the sales history for the selected car and make informed decisions about pricing, marketing, and inventory management.

2.2.3 Customer

When user first register as customer, customer do not have access to the system yet, customer will have to wait for approval from the Managing Staff in order to use the car sales system. Once approved, the system will prompt successful message for customer.

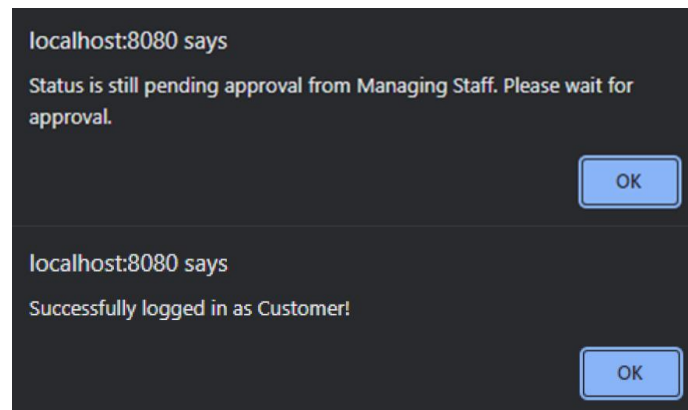


Figure 70: Screenshot of customer login message handling.

Navigation Bar

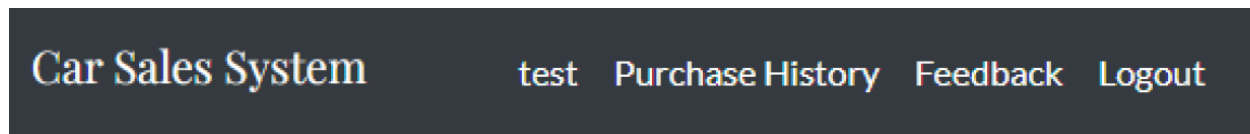
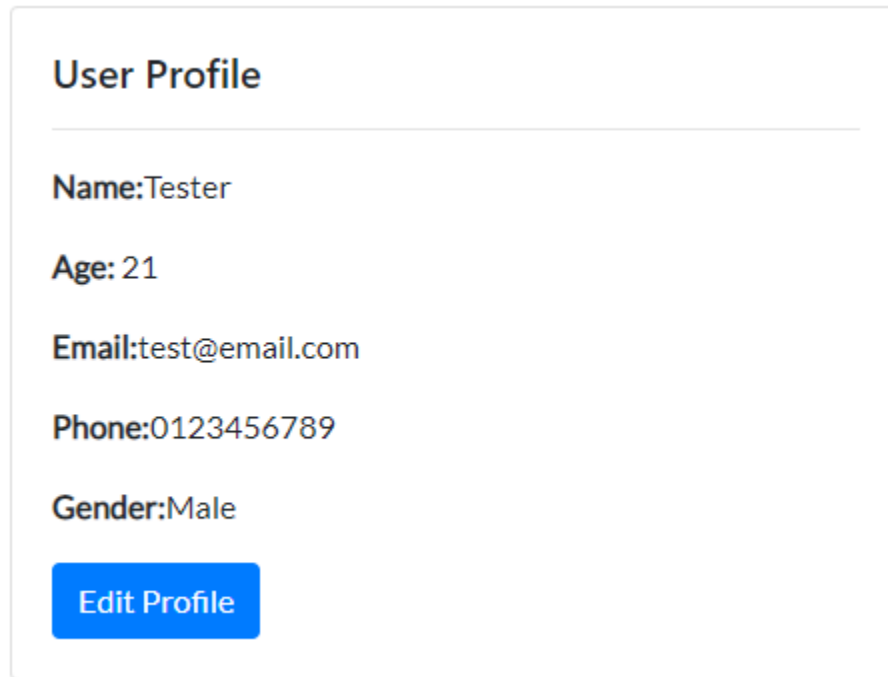


Figure 71: Screenshot of customer navigation bar.

The navigation bar consists of a brand name and a list of menu items. The menu items include "Purchase History", "Feedback", and "Logout". The "Purchase History" menu item will link customer to the purchase history table section, while the "Feedback" menu item will link customer to the feedback section. The "Logout" menu item allows the customer to safely logout from the system.

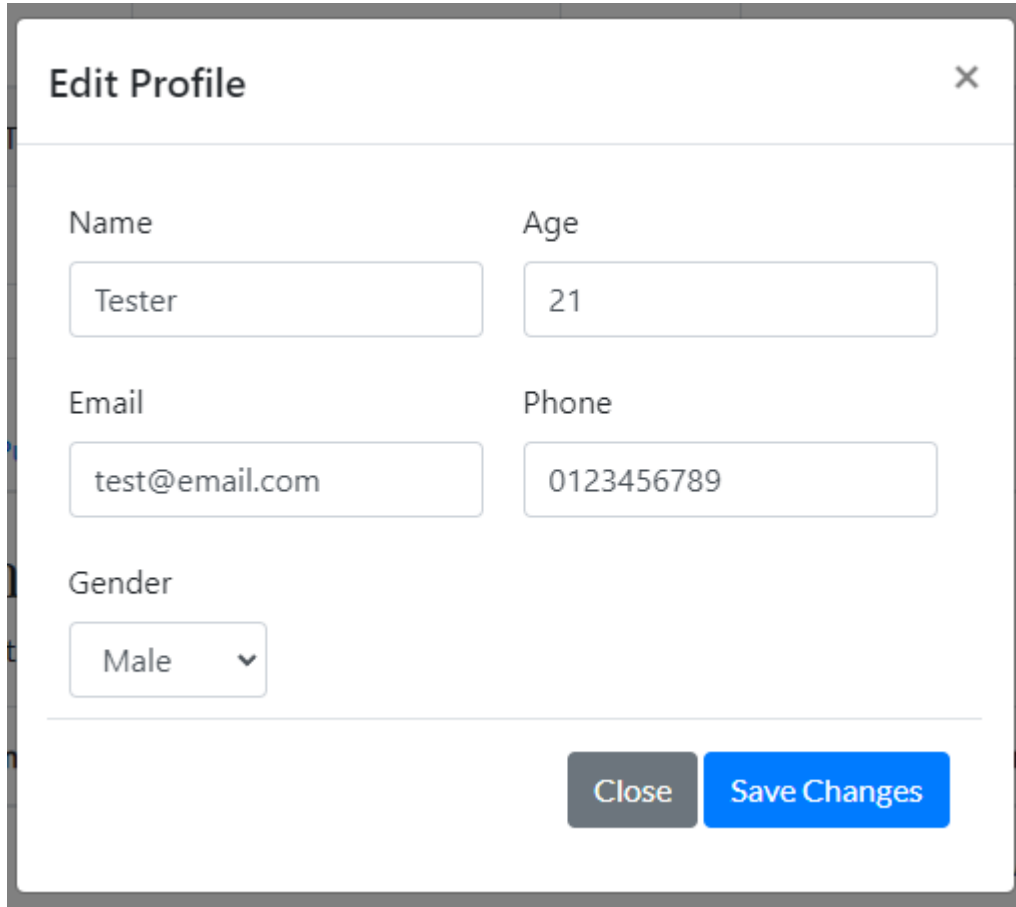
In addition, the username of the customer is displayed on the navigation bar, which allows the customer to easily identify their account. Overall, the navigation bar is a simple and effective user interface element that helps customers to easily navigate the system.

View Customer Profile

The screenshot shows a 'User Profile' section within a light gray bordered box. At the top, the title 'User Profile' is displayed in a bold, dark blue font, followed by a horizontal separator line. Below this, the profile information is listed in a clean, sans-serif font: 'Name: Tester', 'Age: 21', 'Email: test@email.com', 'Phone: 0123456789', and 'Gender: Male'. At the bottom of the profile section, there is a prominent blue rectangular button with the text 'Edit Profile' in white.

Figure 72: Screenshot of view customer profile.

The user profile section is where the customer can view their personal information such as their name, age, email, phone number, and gender. The information displayed can be edited by clicking on the "Edit Profile" button which will open a modal where the customer can update their information.

Edit ProfileThe image shows a modal window titled "Edit Profile" with a close button (X) in the top right corner. The form contains five input fields: "Name" with the value "Tester", "Age" with the value "21", "Email" with the value "test@email.com", "Phone" with the value "0123456789", and "Gender" with a dropdown menu showing "Male". At the bottom right of the modal are two buttons: "Close" (grey) and "Save Changes" (blue).

Name	Age
<input type="text" value="Tester"/>	<input type="text" value="21"/>
Email	Phone
<input type="text" value="test@email.com"/>	<input type="text" value="0123456789"/>
Gender	
<input type="text" value="Male"/>	

Figure 73: Screenshot of edit customer profile.

The edit modal contains a form that allows users to edit their name, age, email, phone number, and gender. The form fields are pre-populated with the customer's current information. All input fields have validations in place to make sure there are no empty fields when the modal is submitted. Upon submission of the form, the user's information will be updated in the database.

View Available Cars

Available Cars						
Here are the available cars for sale.						
Make	Model	Chassis	Year	Color	Price	Description
Subaru	WRX STI	EJ20XVAB65286	2018	World Rally Blue	250000	Blue Subaru WRX STI with modified exhaust
Perodua	Myvi	CN201583	2022	Granite Grey	50000	Myvi King

Figure 74: Screenshot of customer view available cars.

The "Available Cars" section is a table that displays a list of cars that are available for sale. The table contains columns for the make, model, chassis, year, color, price, and description of each car. If there are no cars available, a message stating "No data available" is displayed. The table is designed to provide an easy-to-read and organized display of available cars to potential buyers.

View Pending Transactions

Pending Transactions

Purchase History

Pending Transactions

View your pending transactions here:

Date	Car Information	Car Chassis	Price	Salesman Name	Salesman Phone Number	Status
10/03/2023	Perodua Myvi 2022 Granite Grey	CN201583	50000	Salesman	0198765431	Booked

Figure 75: Screenshot of view pending transactions.

The tab interface allows the user to switch between two different views: "Pending Transactions" and "Purchase History". The "Pending Transactions" view displays a table of the user's pending transactions, including the date, car information, chassis number, price, salesman name, salesman phone number, and transaction status. The table also includes a message indicating that no data is available if there are no pending transactions. This feature provides a convenient way for the user to keep track of their transactions and see which ones are still pending.

View Purchase History

Pending Transactions

Purchase History

Purchase History

View your completed transactions here:

Date	Car Information	Car Chassis	Price	Salesman Name	Salesman Phone Number	Status
09/03/2023	Toyota 86 2021 Pearl White	FA20QWERTY86	180000	Salesman	0198765431	Paid
09/03/2023	Subaru Forester 2020 Pearl White	FB205259241	250000	Salesman	0198765431	Paid
10/03/2023	Perodua Myvi 2022 Granite Grey	CN201583	50000	Salesman	0198765431	Paid

Figure 76: Screenshot of view purchase history.

The "Purchase History" tab shows the user completed transactions, while the "Pending Transactions" tab shows transactions that are still pending. Each table displays the transaction date, car information (make, model, year, and color), car chassis, price, salesman name, salesman phone number, and status. If there is no data available to display in either table, a message saying "No data available" is shown.

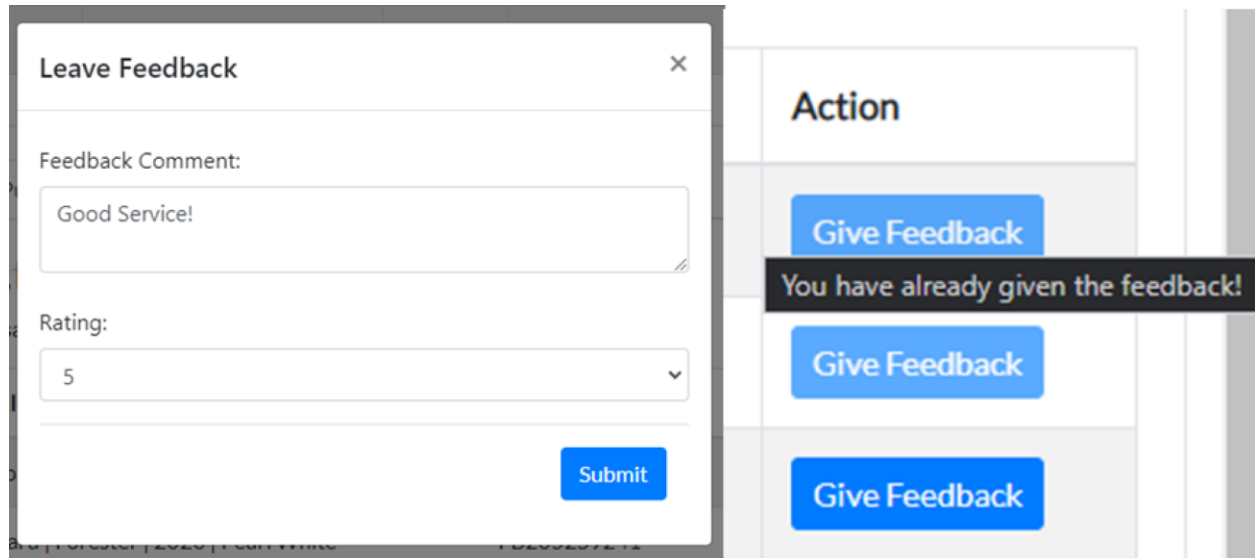
View Feedback

Feedback						
Share your feedback about our cars and services.						
Car Information	Chassis Number	Price	Salesman	Payment Status	Feedback & Rating	Action
Toyota 86 2021 Pearl White	FA20QWERTY86	180000	Salesman	Completed	Good Service 5	Give Feedback
Subaru Forester 2020 Pearl White	FB205259241	250000	Salesman	Completed	Good Car 3	Give Feedback
Perodua Myvi 2022 Granite Grey	CN201583	50000	Salesman	Completed	No Feedback Yet No Rating Yet	Give Feedback

Figure 77: Screenshot of view feedback.

The view feedback table displays a list of completed transactions and allows the user to give feedback and rating on the cars and services received. The table shows information about the car, such as make, model, year, and color, as well as the chassis number, price, salesman, and payment status. It also displays any existing feedback and rating given by the user and provides a button to give feedback if none has been given or edit existing feedback if it has already been given.

Give Feedback and Rating



The image shows a 'Leave Feedback' modal window on the left and a table on the right. The modal has a title bar with a close button (X). It contains two input fields: 'Feedback Comment:' with the text 'Good Service!' and 'Rating:' with a dropdown menu showing '5'. A blue 'Submit' button is at the bottom right. The table on the right has a header row with the text 'Action'. Below the header, there are three rows, each containing a blue 'Give Feedback' button. A black notification banner with the text 'You have already given the feedback!' is overlaid on the middle row of the table.

Figure 78: Screenshot of give feedback and rating.

The feedback modal is a popup window that allows users to leave feedback and rating for a car they have purchased. It contains two input fields: "Feedback Comment" and "Rating". The "Feedback Comment" field is a textarea where the user can leave a comment about the car or service they have received. The "Rating" field is a dropdown menu where the user can rate the car or service from 1 to 5 stars.

Once the user has filled out the feedback and rating fields, they can submit the feedback by clicking on the "Submit" button in the modal footer. The feedback will then be sent to the server to be processed and stored in the database. The user will then be noticed that their feedback has been received and will be able to view it in the feedback table on the dashboard.

2.2.4 General Navigation Chart (Sitemap)

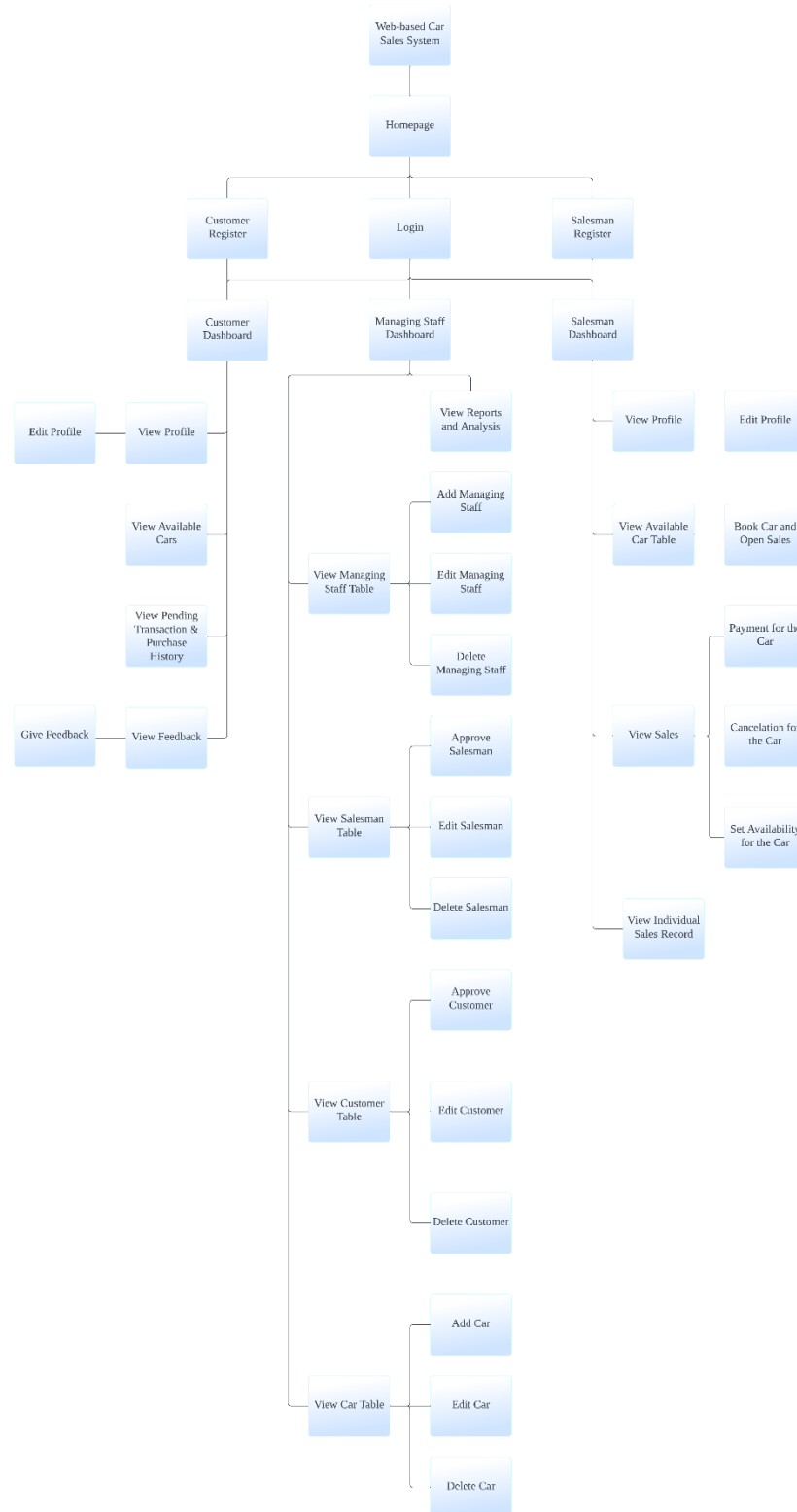


Figure 79: Screenshot of general navigation chart.

3.0 Part 3

3.1 Overview of application and brief description of the system architecture

The Web-based Car sales system is an online platform that facilitates the buying and selling of cars. The application is designed to allow customer users to view a catalog of available cars, do booking with salesman, and give feedbacks. It allows salesman users to handle car status, help customers to do booking and payment, and view individual sales records. The system also allows managing staff user to manage their inventory, manage salesman and customer information, and track customer and sales information.

The web application's architecture that the system is using is the model-view-controller (MVC). The model-view-controller (MVC) architecture is a widely used software design pattern for web applications. It separates an application into three interconnected components: the Model, View, and Controller. The Model manages the data logic, the View presents the data to the user, and the Controller acts as the mediator between the two (Svirca, 2021).

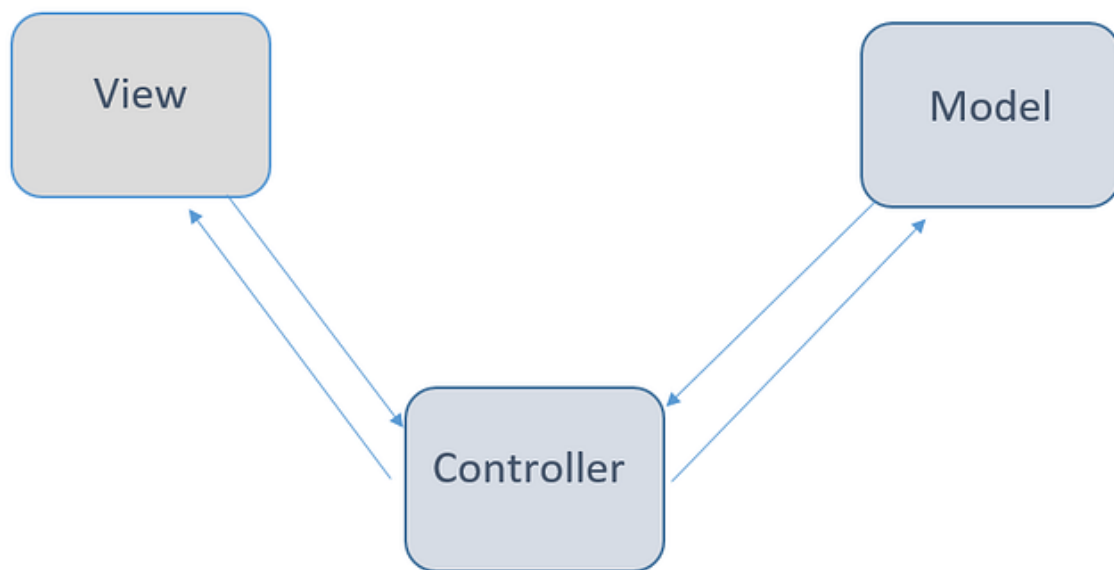


Figure 80: MVC architecture (Svirca, 2021).

One of the key benefits of the MVC architecture is that it allows for easier maintenance and modification of code (Svirca, 2021). Because each component is independent, it is easier to make changes to one component without affecting the others (Svirca, 2021). Additionally, the MVC pattern improves the scalability of the application by enabling the use of multiple Views for a single Model. This allows for the development of different versions of the application for different platforms, such as desktop and mobile devices. In addition, the Model-View-Controller (MVC) architecture can be applied to describe the interconnection among the tiers.

The presentation tier (View) is responsible for displaying the data to the user through the user interface (IBM, 2023). This layer includes the user interface components that the user interacts with, such as forms, buttons, and menus. The View receives user input and sends it to the Controller for processing. It is designed to be visually appealing and easy to use, and it should provide a seamless user experience. The View is responsible for presenting the data to the user in an organized and understandable format.

The application tier (Controller) is responsible for handling user actions and routing the application flow. It receives input from the View, updates the Model accordingly, and sends the Model data back to the View for display. The Controller is the mediator between the View and the Model (IBM, 2023). It handles application-level events and communicates with other Controllers as necessary. The Controller is responsible for validating user input and performing other operations, such as authentication and authorization.

The data tier (Model) is responsible for managing the data and interacting with the database. It contains the application logic layer, which validates the user input and performs other operations, such as retrieving or updating data (IBM, 2023). The Model receives requests from the Controller to retrieve or update data, and it returns the data to the Controller for processing. The Model includes the data access layer, which interacts with the database to retrieve or update data. It is responsible for processing user input and generating the output for the user.

Overall, the MVC architecture allows for a clear separation of concerns and interconnection among the different layers of the application. This helps to make the system easier to maintain, modify, and scale in the future. The image below shows the MVC architecture of the car sales system.

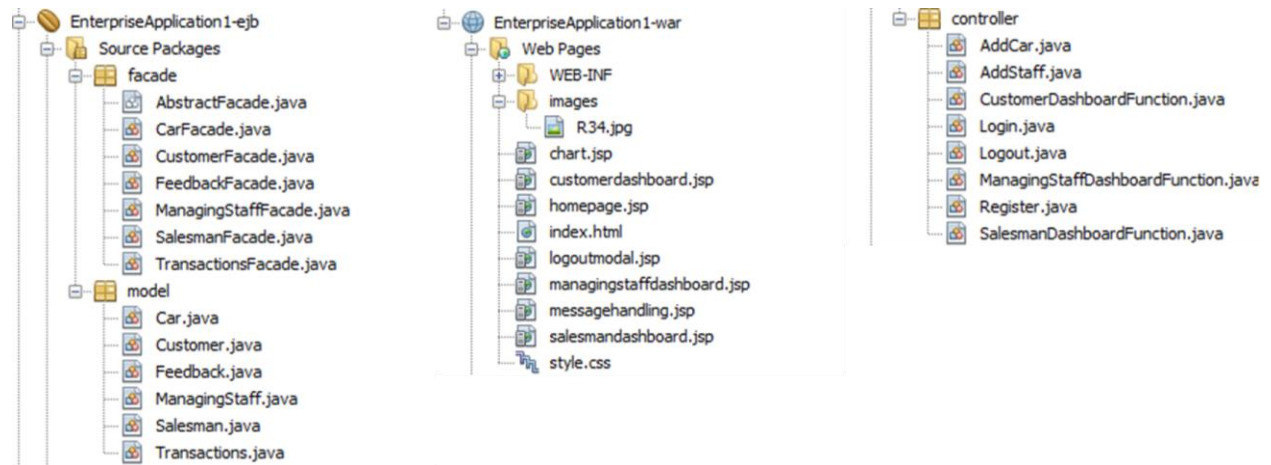


Figure 81: MVC in car sales system.

3.2 Design of Business Tier

The design of the business tier is a critical aspect of developing enterprise-level applications. The business tier is responsible for implementing the business logic of an application and typically consists of Enterprise JavaBeans (EJBs).

Enterprise JavaBeans (EJB) is a technology used to develop scalable, distributed applications in Java. It provides a component-based architecture for building server-side business logic that can be deployed on a Java EE application server. EJBs are managed by the application server, which provides services such as transaction management, security, and resource pooling.

The EJB component model has three object types, and each object type serves a specific purpose in the business tier:

1. Session bean
2. Message-driven bean,
3. Entity bean.

3.2.1 Session bean

Session bean is a component that contains business logic that can be accessed programmatically by a client through local, remote, or web service client views (Oracle, 2013). Clients access the methods of the session bean to perform business tasks within the server, and the session bean executes these tasks while shielding the client from complexity (Oracle, 2013). Session beans are not persistent, meaning that their data is not saved to a database.

There are three types of session beans: stateful, stateless, and singleton.

Stateful session beans store a unique client/bean session state in its instance variables, which is often referred to as the conversational state (Oracle, 2013). The state is retained for the duration of the client/bean session, and it disappears when the session ends. Stateless session beans do not maintain a conversational state with the client (Oracle, 2013). Instead, the bean's instance variables may contain a state specific to that client only for the duration of the invocation, and client-specific state should not be retained when the method is finished (Oracle, 2013). All instances of a stateless bean are equivalent, which allows the EJB container to assign an instance to any client (Oracle, 2013). Singleton session beans are instantiated once per application and exist for the lifecycle of the application (Oracle, 2013). They are designed for situations in which a single enterprise bean

instance is shared across and concurrently accessed by clients. Singleton session beans offer similar functionality to stateless session beans but differ in that there is only one singleton session bean per application (Oracle, 2013). Singleton session beans maintain their state between client invocations and can be used to perform initialization and cleanup tasks for the application (Oracle, 2013).

3.2.2 Message-driven bean

Message-driven bean is an essential component of a lightweight enterprise application that is designed to process messages asynchronously, meaning that the user does not always receive immediate results (Techopedia Inc., 2011a). Unlike session and entity beans, message-driven beans cannot be accessed through interfaces and only have a bean class (Techopedia Inc., 2011a). They can be used by any component to send messages, regardless of whether they use J2EE technology (Techopedia Inc., 2011a).

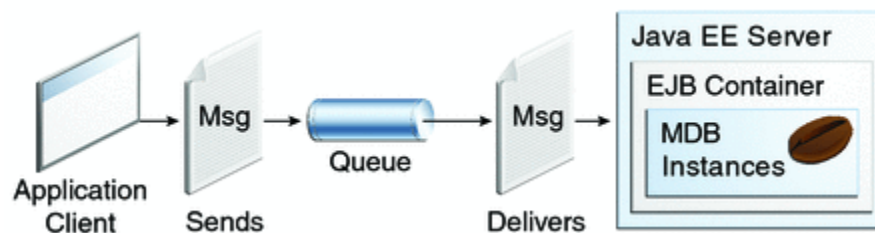


Figure 82: Message-driven bean.

Message-driven beans are known for their unique features:

- They do not retain data or conversational state for a specific client.
- All instances of a message-driven bean are equivalent, allowing the EJB container to assign a message to any message-driven bean instance. The container can pool these instances to allow streams of messages to be processed concurrently.
- A single message-driven bean can process messages from multiple clients.

These beans are commonly used in asynchronous communication between enterprise application components. For instance, when a new message arrives, the `onMessage` method of the message-driven bean is called by the enterprise JavaBeans container to process the message (Techopedia Inc., 2011a). The message is then cast as a JMS message and handled based on the application's business logic (Techopedia Inc., 2011a). If the `onMessage` method only wants to

process the message, it invokes a session bean. However, if the method wants to store the message in a database, it invokes an entity bean (Techopedia Inc., 2011a). The message is then delivered to a message-driven bean so that the above operations become part of a single and complete transaction (Techopedia Inc., 2011a). In some cases, a message may be redelivered if there is a rollback in message processing (Techopedia Inc., 2011a).

3.2.3 Entity bean

In the Java Platform 2, Enterprise Edition (J2EE) context, an entity bean refers to a business object that is stored in a persistent storage mechanism even after the end of a session (Techopedia Inc., 2011b). This can include information such as customer name, account number, and balance. A relational database is commonly used as the persistent storage mechanism, with a table being created for each entity bean and each bean instance corresponding to a specific row in the table (Techopedia Inc., 2011b).

There are several distinguishing features that set entity beans apart from session beans. Entity beans remain after the end of a session, can be accessed by multiple clients, and have a primary key or unique identifier (Techopedia Inc., 2011b).

There are two different methods of entity bean persistence: bean-managed and container-managed (Techopedia Inc., 2011b). Since entity beans are stored in a relational database, data can persist even after a session ends (Techopedia Inc., 2011b). Since multiple clients may need to access and change the same data simultaneously, it is important to manage entity transaction carefully (Techopedia Inc., 2011b). The EJB container provides transaction management for each bean to ensure data integrity (Techopedia Inc., 2011b). Each entity bean is assigned a unique identifier, which allows clients to locate a specific bean. Entity beans are typically used when a bean is a business object and not just a method, or when a bean's state needs to remain persistent (Techopedia Inc., 2011b).

In summary, the EJB component model provides a powerful architecture for building enterprise-level applications in Java. By using EJBs, developers can focus on implementing the business logic of their application while relying on the application server to handle services such as transaction management and security. The three types of EJBs (session beans, message-driven beans, and entity bean) provide different functionalities and are used in different scenarios, allowing developers to build robust and scalable applications.

3.3 UML Diagrams (Use Case & Class)

3.3.1 Use Case Diagram

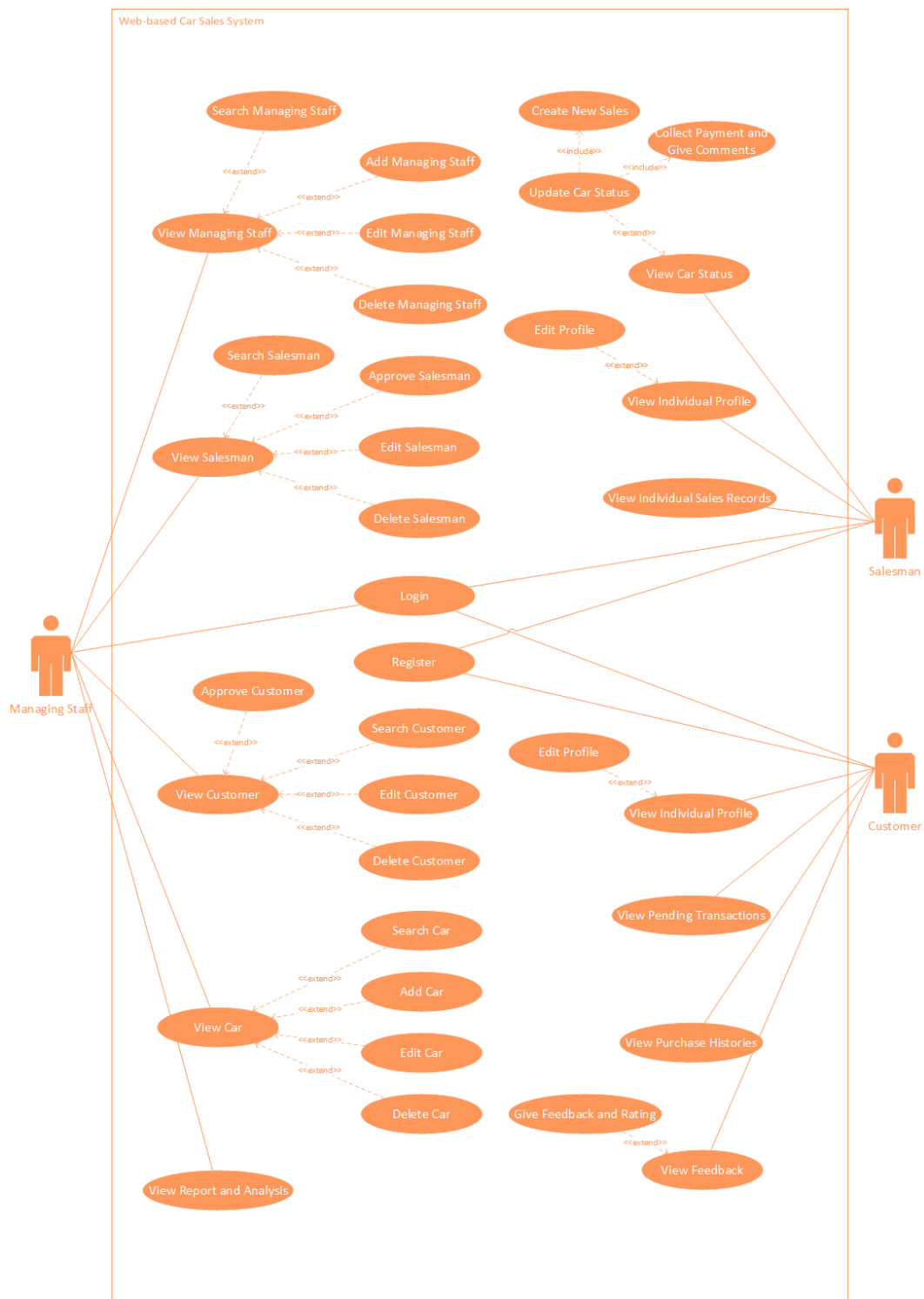


Figure 83: Use Case Diagram for Car Sales System.

3.3.2 Class Diagram

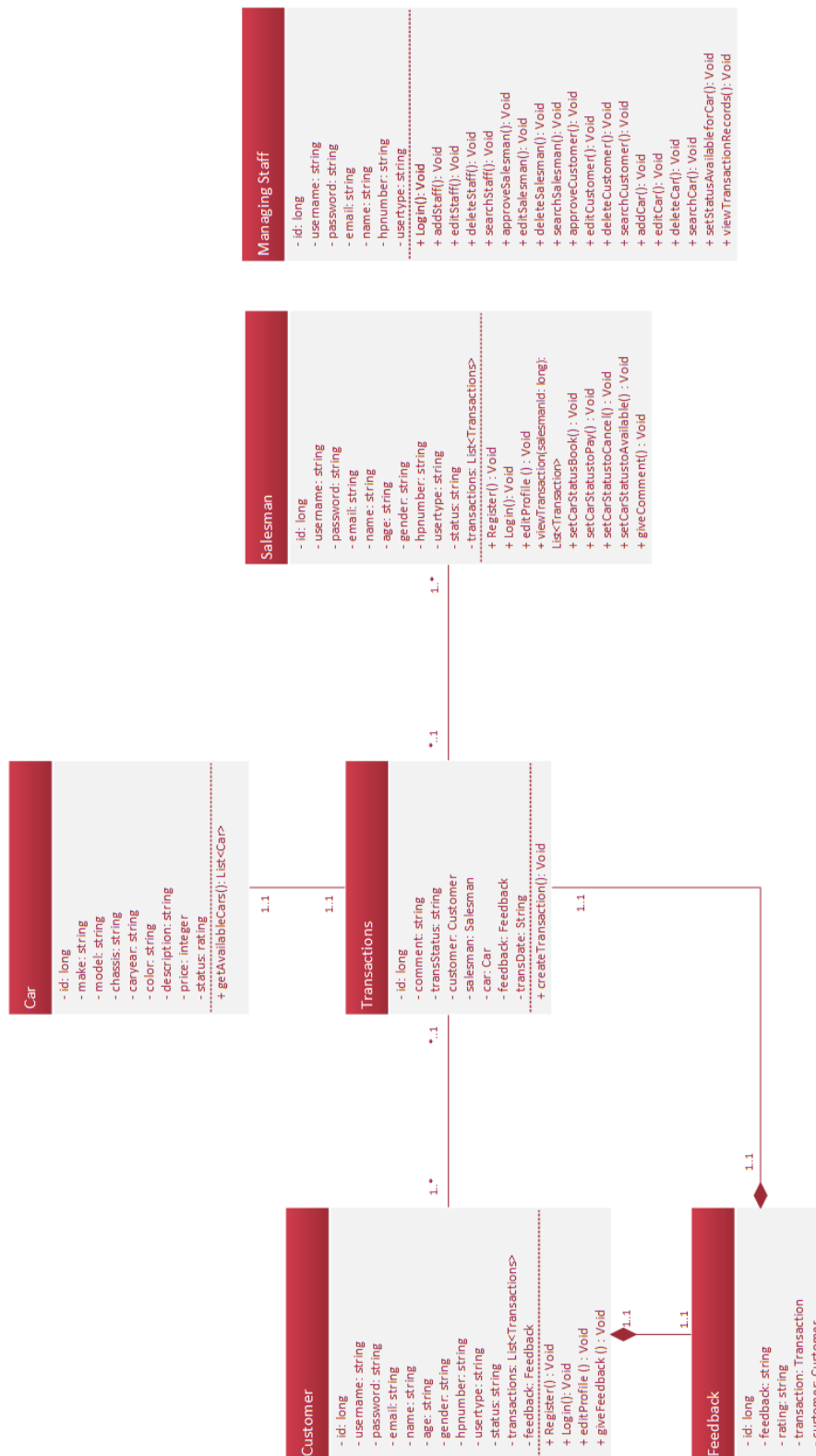


Figure 84: Class diagram of car sales system.

4.0 Part 4

4.1 Database Design

The database consisted of 6 tables:

- 1) Customer
- 2) Salesman
- 3) ManagingStaff
- 4) Car
- 5) Feedback
- 6) Transactions

1. Customer

Table Description: Stores customer information such as the id, username, password, email, name, age, gender, hpnumber, usertype, status. The id is the primary key of this table and is has Long as its data type, the rest of the column name is String data type.

Column Name	Data Type
id (Primary Key)	Long
username	String
password	String
email	String
name	String
age	String
gender	String
hpnumber	String
usertype	String
status	String

2. Salesman

Table Description: Stores customer information such as the id, username, password, email, name, age, gender, hpnumber, usertype, status. The id is the primary key of this table and is has Long as its data type, the rest of the column name is String data type.

Column Name	Data Type
id (Primary Key)	Long
username	String
password	String
email	String
name	String
age	String
gender	String
hpnumber	String
usertype	String
status	String

3. Managing Staff

Table Description: Stores managing staff information such as the id, username, password, name, email, hpnumber, usertype.

Column Name	Data Type
id (Primary Key)	Long
username	String
password	String
name	String
email	String
hpnumber	String
usertype	String

4. Car

Table Description: Stores car information such as the id, make, model, chassis, caryear, color, description, price, status. The columns include the primary key "id", the make and model of the car, the chassis number, the year of the car, the color, the description, the price and the status. The data types include Long, String and Integer. The primary key "id" uniquely identifies each car in the table, while the status column indicates whether the car is available or not.

Column Name	Data Type
id (Primary Key)	Long
make	String
model	String
chassis	String
caryear	String
color	String
description	String
price	Integer
status	String

5. Feedback

Table Description: Stores customer's feedback such as the id, feedback, rating, customer_id. "id" is the primary key and has a data type of "Long". "feedback" and "rating" columns have data types of "String" and are used to store feedback and rating information. The "customer_id" column is a foreign key that references the "id" column in the "Customer" table. It is used to establish a relationship between the feedback and the customer who provided it.

Column Name	Data Type
id (Primary Key)	Long
feedback	String
rating	String
customer_id (Foreign Key)	Customer

6. Transaction

Table Description: This table has several columns with different data types. The first column is the primary key column, named "id", which has a data type of Long. The second column is "comment", which is a string type column for any comments related to the transaction. The third column is "transStatus", also a string type column, which represents the status of the transaction. The fourth column is "transDate", which is a string type column for the date of the transaction. The last four columns are foreign key columns that reference other tables. "customer" is a foreign key referencing the "Customer" table, "salesman" is a foreign key referencing the "Salesman" table, "car" is a foreign key referencing the "Car" table, and "feedback" is a foreign key referencing the "Feedback" table.

Column Name	Data Type
id (Primary Key)	Long
comment	String
transStatus	String
transDate	String
customer (Foreign Key)	Customer
salesman (Foreign Key)	Salesman
car (Foreign Key)	Car
feedback (Foreign Key)	Feedback

4.1.1 Entity Relationship Diagram

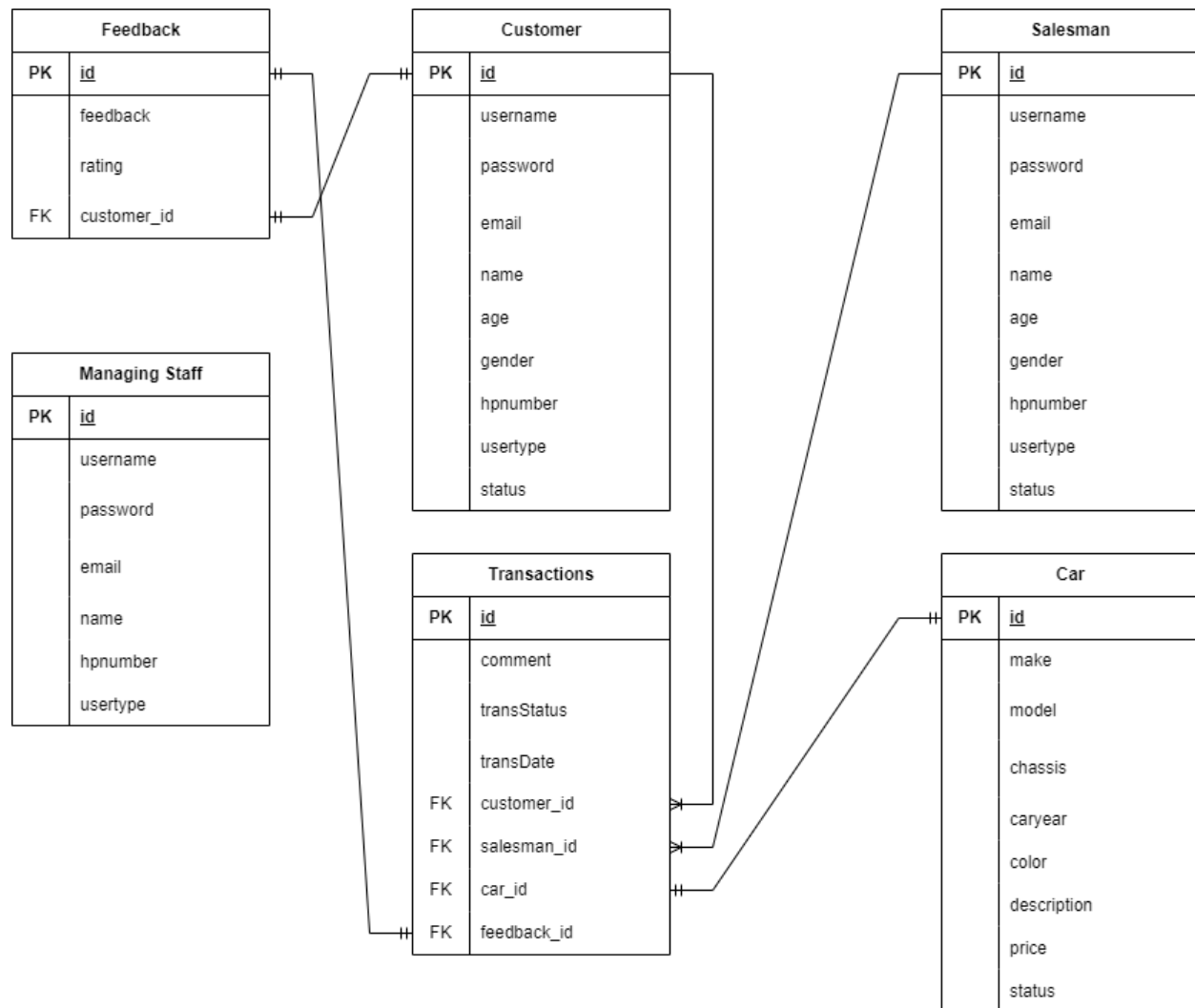


Figure 85: Entity Relationship Diagram for Car Sales System.

4.1.2 Design of Database access APIs

This section discusses about the database access APIs in the car sales system. These APIs provide a standardized set of methods and interfaces for performing common database operations, such as connecting to the database, executing queries, and manipulating data. Examples of database access APIs in Java include Java Database Connectivity (JDBC), Hibernate, and Spring Data. A facade pattern can be used to provide a simplified interface to a complex set of APIs or systems, but it is not a database access API in itself.

JDBC is mostly used for this system, JDBC is a Java API that provides a standard interface for connecting to and interacting with a relational database. JDBC allows Java applications to send SQL statements to a database, retrieve and process the results, and perform database updates and modifications. JDBC provides a set of classes and interfaces that represent the different components of a database system, such as connections, statements, result sets, and metadata. JDBC drivers are used to connect Java applications to specific database systems.

Java Persistence API (JPA) is a high-level API for interacting with databases that is built on top of JDBC. JPA provides an object-relational mapping (ORM) framework that allows developers to map Java classes to database tables and vice versa. JPA also provides a set of APIs for querying and manipulating data, as well as managing transactions and persistence contexts. JPA abstracts away much of the low-level JDBC code and provides a simpler and more object-oriented approach to working with databases. JPA is implemented by various persistence providers, such as Hibernate and EclipseLink, that translate JPA code into JDBC calls to interact with the database. The following shows the code snippet and explanation about the façade that is designed for the database.

CustomerFacade.java

```

public Customer findByCustomerUsername(String username) {
    try {
        return em.createQuery("SELECT c FROM Customer c WHERE c.username = :username", Customer.class)
            .setParameter("username", username)
            .getSingleResult();
    } catch (NoResultException e) {
        return null;
    }
}

public Customer loginAsCustomer(String username, String password) {
    Customer customer = findByCustomerUsername(username);
    if (customer != null && customer.getPassword().equals(password)) {
        return customer;
    }
    return null;
}

public List<Customer> searchCustomer(String keyword) {
    return em.createQuery("SELECT c FROM Customer c WHERE LOWER(c.username) LIKE :keyword "
        + "OR LOWER(c.name) LIKE :keyword "
        + "OR LOWER(c.email) LIKE :keyword "
        + "OR LOWER(c.gender) LIKE :keyword "
        + "OR LOWER(c.honenumber) LIKE :keyword ", Customer.class)
        .setParameter("keyword", "%" + keyword.toLowerCase() + "%")
        .getResultList();
}

public List<Customer> findAllCustomer() {
    return em.createQuery("SELECT c FROM Customer c WHERE c.usertype = 'customer'", Customer.class)
        .getResultList();
}

public Long findCustomerGenderCount(String gender) {
    return em.createQuery("SELECT COUNT(c) FROM Customer c WHERE c.gender = :gender", Long.class)
        .setParameter("gender", gender)
        .getSingleResult();
}

```

Figure 86: Code snippet of CustomerFacade.java.

These are several methods in a Java class that are using JPA to access a database and retrieve data related to customers. Here is a brief explanation of each method:

- `findByCustomerUsername(String username)` - This method searches the database for a customer with a specific username and returns a `Customer` object. It uses JPA's `createQuery()` method to create a query that searches the `Customer` table for a row where the username column matches the given username parameter. If a matching customer is found, the method returns a `Customer` object. If no customer is found, it returns `null`.

- `loginAsCustomer(String username, String password)` - This method is used to verify a customer's login credentials. It calls the `findByCustomerUsername()` method to retrieve a `Customer` object with the given username. If a matching customer is found and the password matches the one stored in the database, the method returns the `Customer` object. Otherwise, it returns null.
- `searchCustomer(String keyword)` - This method searches the database for customers whose usernames, names, emails, genders, or phone numbers match a given keyword. It uses JPA's `createQuery()` method to create a query that searches the `Customer` table for rows where any of these columns match the given keyword. The method returns a list of `Customer` objects that match the search criteria.
- `findAllCustomer()` - This method retrieves all customers from the database whose `usertype` column is set to "customer". It uses JPA's `createQuery()` method to create a query that searches the `Customer` table for rows where the `usertype` column is "customer". The method returns a list of `Customer` objects that match the search criteria.
- `findCustomerGenderCount(String gender)` - This method returns the number of customers in the database whose `gender` column matches a given value. It uses JPA's `createQuery()` method to create a query that counts the number of rows in the `Customer` table where the `gender` column matches the given `gender` parameter.

The queries used in these methods are written in JPQL (Java Persistence Query Language), which is similar to SQL (Structured Query Language) but uses entity class names and field names instead of table names and column names. The queries are executed using JPA's `createQuery()` method, which takes a JPQL query string and returns a `Query` object that can be used to set query parameters and execute the query. The `getResultList()` and `getSingleResult()` methods are used to retrieve the results of the query as a list or a single object, respectively.

SalesmanFacade.java

```

public Salesman findBySalesmanUsername(String username) {
    try {
        return em.createQuery("SELECT s FROM Salesman s WHERE s.username = :username", Salesman.class)
            .setParameter("username", username)
            .getSingleResult();
    } catch (NoResultException e) {
        return null;
    }
}

public Salesman loginAsSalesman(String id, String password) {
    Salesman salesman = findBySalesmanUsername(id);
    if (salesman != null && salesman.getPassword().equals(password)) {
        return salesman;
    }
    return null;
}

public List<Salesman> searchSalesman(String keyword) {
    return em.createQuery("SELECT s FROM Salesman s WHERE s.usertype = 'salesman' AND (LOWER(s.username) LIKE :keyword "
        + "OR LOWER(s.name) LIKE :keyword "
        + "OR LOWER(s.email) LIKE :keyword "
        + "OR LOWER(s.gender) LIKE :keyword "
        + "OR LOWER(s.hpnnumber) LIKE :keyword "
        + "OR LOWER(s.status) LIKE :keyword)", Salesman.class)
        .setParameter("keyword", "%" + keyword.toLowerCase() + "%")
        .getResultList();
}

public List<Salesman> findAllSalesman() {
    return em.createQuery("SELECT s FROM Salesman s WHERE s.usertype = 'salesman'", Salesman.class)
        .getResultList();
}

public Long findSalesmanGenderCount(String gender) {
    return em.createQuery("SELECT COUNT(s) FROM Salesman s WHERE s.gender = :gender", Long.class)
        .setParameter("gender", gender)
        .getSingleResult();
}

```

Figure 87: Code snippet of SalesmanFacade.java.

This is a set of methods that operate on the Salesman entity in the database using JPA.

- The `findBySalesmanUsername` method takes a username as an argument, and returns the Salesman object with that username. It uses the JPA `createQuery` method to create a query that searches for the Salesman object by username, and sets the username parameter using the `setParameter` method. If the query returns no results, the method returns null.
- The `loginAsSalesman` method takes a salesman ID and password as arguments, and returns the Salesman object with the corresponding ID and password. It calls the `findBySalesmanUsername` method to retrieve the Salesman object with the given ID, and then checks if the password matches the Salesman object's password. If the password matches, the method returns the Salesman object, otherwise it returns null.

- The searchSalesman method takes a keyword as an argument, and returns a list of Salesman objects that match the keyword. It uses the JPA createQuery method to create a query that searches for Salesman objects by various fields, including username, name, email, gender, hpnumber, and status, and sets the keyword parameter using the setParameter method. The query includes an additional condition that restricts the search to Salesman objects with the usertype 'salesman'. The method returns a list of Salesman objects that match the query.
- The findAllSalesman method returns a list of all Salesman objects in the database with the usertype 'salesman'. It uses the JPA createQuery method to create a query that selects all Salesman objects with the usertype 'salesman', and returns the list of Salesman objects that match the query.
- The findSalesmanGenderCount method takes a gender as an argument, and returns the count of Salesman objects with the given gender. It uses the JPA createQuery method to create a query that selects all Salesman objects with the given gender, and returns the count of Salesman objects that match the query. The gender parameter is set using the setParameter method.

ManagingStaffFacade.java

```

public List<ManagingStaff> findAllManagingStaff() {
    return em.createQuery("SELECT m FROM ManagingStaff m WHERE m.usertype = 'managingStaff'", ManagingStaff.class)
        .getResultList();
}

public List<ManagingStaff> searchManagingStaff(String keyword) {
    return em.createQuery("SELECT m FROM ManagingStaff m WHERE LOWER(m.username) LIKE :keyword "
        + "OR LOWER(m.name) LIKE :keyword "
        + "OR LOWER(m.email) LIKE :keyword "
        + "OR LOWER(m.hpnumber) LIKE :keyword ", ManagingStaff.class)
        .setParameter("keyword", "%" + keyword.toLowerCase() + "%")
        .getResultList();
}

public ManagingStaff findByMgmtStaffUsername(String username) {
    try {
        return em.createQuery("SELECT m FROM ManagingStaff m WHERE m.username = :username", ManagingStaff.class)
            .setParameter("username", username)
            .getSingleResult();
    } catch (NoResultException e) {
        return null;
    }
}

public ManagingStaff loginAsManagingStaff(String username, String password) {
    ManagingStaff manager = findByMgmtStaffUsername(username);
    if (manager != null && manager.getPassword().equals(password)) {
        return manager;
    }
    return null;
}

```

Figure 88: Code snippet of ManagingStaffFacade.java.

This is a set of Java methods that interact with the database to retrieve, update, and search for ManagingStaff objects.

- The `findAllManagingStaff()` method returns a list of all ManagingStaff objects stored in the database. It uses a JPQL query to retrieve all ManagingStaff objects where the usertype is 'managingStaff'.
- The `searchManagingStaff(String keyword)` method searches the database for ManagingStaff objects that match a given keyword. It uses a JPQL query to search for ManagingStaff objects where the username, name, email, or hpnumber (phone number) match the keyword provided in a case-insensitive manner.
- The `findByMgmtStaffUsername(String username)` method retrieves a single ManagingStaff object from the database based on the username provided. It uses a JPQL query to select a ManagingStaff object where the username matches the one provided.

- The `loginAsManagingStaff(String username, String password)` method checks if a `ManagingStaff` object exists with the given username and password. It uses the `findByMgmtStaffUsername()` method to retrieve the `ManagingStaff` object, and then checks if the password matches the one provided.
- All of these methods use the `em` `EntityManager` object to interact with the database. The JPQL queries used in these methods are executed on the database through the `EntityManager`'s persistence context, which maps the objects retrieved from the database to their corresponding Java objects.

CarFacade.java

```

public List<Car> searchCar(String keyword) {
    return em.createQuery("SELECT c FROM Car c WHERE LOWER(c.make) LIKE :keyword "
        + "OR LOWER(c.model) LIKE :keyword "
        + "OR LOWER(c.chassis) LIKE :keyword "
        + "OR LOWER(c.caryear) LIKE :keyword "
        + "OR LOWER(c.color) LIKE :keyword "
        + "OR LOWER(c.description) LIKE :keyword "
        + "OR LOWER(c.status) LIKE :keyword ", Car.class)
        .setParameter("keyword", "%" + keyword.toLowerCase() + "%")
        .getResultList();
}

public List<Car> findAllCars() {
    return em.createQuery("SELECT c FROM Car c", Car.class).getResultList();
}

public List<Car> findAllAvailableCars() {
    return em.createQuery("SELECT c FROM Car c WHERE c.status = 'Available'", Car.class).getResultList();
}

public Car findByCarChassis(String make) {
    try {
        return em.createQuery("SELECT c FROM Car c WHERE c.make = :make", Car.class)
            .setParameter("make", make)
            .getSingleResult();
    } catch (NoResultException e) {
        return null;
    }
}

public Long getAvailableCarCount() {
    return em.createQuery("SELECT COUNT(c) FROM Car c WHERE c.status = 'Available'", Long.class)
        .getSingleResult();
}

public Long getBookedCarCount() {
    return em.createQuery("SELECT COUNT(c) FROM Car c WHERE c.status = 'Booked'", Long.class)
        .getSingleResult();
}

public Long getPaidCarCount() {
    return em.createQuery("SELECT COUNT(c) FROM Car c WHERE c.status = 'Paid'", Long.class)
        .getSingleResult();
}

public Long getCancelledCarCount() {
    return em.createQuery("SELECT COUNT(c) FROM Car c WHERE c.status = 'Cancelled'", Long.class)
        .getSingleResult();
}

```

Figure 89: Code snippet of CarFacade.java.

The code snippet contains several methods that allow searching and retrieving car data from the database.

- The searchCar method searches for cars that match a certain keyword, which is provided as a parameter. The search is done on several car attributes such as make, model, chassis, and color.

- The findAllCars method retrieves all cars from the database.
- The findAllAvailableCars method retrieves all cars that have the status "Available".
- The findByCarChassis method retrieves a single car based on its make.
- The getAvailableCarCount, getBookedCarCount, getPaidCarCount, and getCancelledCarCount methods return the count of cars with specific statuses, such as "Available" or "Cancelled".

All of these methods use JPA's Criteria API to build queries based on the provided parameters. The queries are then executed using the getResultList or getSingleResult methods. The results are returned as lists or single objects, depending on the method used.

TransactionsFacade.java

```

public List<Transactions> findAllTransactionsBySalesman(Salesman salesman) {
    return em.createQuery("SELECT t FROM Transactions t WHERE t.salesman = :salesman", Transactions.class)
        .setParameter("salesman", salesman)
        .getResultList();
}

public Transactions findTransactionByCarId(Long id) {
    return em.createQuery("SELECT t FROM Transactions t WHERE t.car.id = :id", Transactions.class)
        .setParameter("id", id)
        .getSingleResult();
}

public List<Transactions> findAllTransactionsByCustomer(Customer customer) {
    return em.createQuery("SELECT t FROM Transactions t WHERE t.customer = :customer", Transactions.class)
        .setParameter("customer", customer)
        .getResultList();
}

public Long findTransactionsStatusCount(String transStatus) {
    return em.createQuery("SELECT COUNT(t) FROM Transactions t WHERE t.transStatus = :transStatus", Long.class)
        .setParameter("transStatus", transStatus)
        .getSingleResult();
}

public List<Transactions> findAllTransactions() {
    return em.createQuery("SELECT t FROM Transactions t", Transactions.class)
        .getResultList();
}

```

Figure 90: Code snippet of TransactionsFacade.java.

This code snippet includes several methods for querying transactions from the database.

- The first method `findAllTransactionsBySalesman` takes a `Salesman` object as a parameter and returns a list of `Transactions` that belong to that salesman. The query used selects `Transactions` from the `Transactions` table where the `salesman` attribute matches the provided salesman object.
- The second method `findTransactionByCarId` takes a car id as a parameter and returns the corresponding transaction object. The query used selects a single `Transaction` object from the `Transactions` table where the `car id` attribute matches the provided car id.
- The third method `findAllTransactionsByCustomer` takes a `Customer` object as a parameter and returns a list of `Transactions` that belong to that customer. The query used selects `Transactions` from the `Transactions` table where the `customer` attribute matches the provided customer object.
- The fourth method `findTransactionsStatusCount` takes a transaction status string as a parameter and returns the count of `Transactions` that have that status. The query used selects

the count of Transactions from the Transactions table where the transaction status attribute matches the provided transaction status string.

- The last method `findAllTransactions` simply returns a list of all Transactions in the Transactions table. The query used selects all Transactions from the Transactions table.

FeedbackFacade.java

```
public int countFeedbackByRating(String rating) {  
    return findAll().stream()  
        .filter(f -> f.getRating().equalsIgnoreCase(rating))  
        .collect(Collectors.toList())  
        .size();  
}
```

Figure 91: Code snippet of FeedbackFacade.java.

- The `countFeedbackByRating` method in the `FeedbackFacade` class counts the number of feedback entries with a particular rating. It does this by first calling the `findAll` method provided by the `AbstractFacade` superclass to retrieve all Feedback entries, then using the `stream` method to filter the list of feedback by the rating provided as a parameter. The resulting list is then collected to a list and its size is returned as the count of feedback entries with the specified rating. This method provides a simple and convenient way for clients to obtain feedback statistics based on the ratings provided by users.

All in all, the `CustomerFacade`, `SalesmanFacade`, `ManagingStaffFacade`, `CarFacade`, `TransactionsFacade`, `FeedbackFacade` are classes that extend an abstract facade class called `AbstractFacade`. This facade provides a simplified interface to all entity classes, allowing clients to perform CRUD (Create, Read, Update, and Delete) operations on the entity classes using a simplified API.

5.0 Description and evidence of additional features which have been incorporated in the solution

5.1 Bootstrap Framework

Bootstrap is a popular front-end web development framework. It was initially developed by Twitter and is now maintained by the Bootstrap Core Team, a group of developers and designers from around the world (JavaTpoint, 2023b). Bootstrap provides a collection of CSS and JavaScript components, as well as design templates, that can be used to build responsive, mobile-first websites and web applications (JavaTpoint, 2023b). To use Bootstrap, libraries must be imported.

```
<!-- Bootstrap CSS -->
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQV3Xipma34MD+dH/1fQ784/j6cY/IJTU/QhW7x9Jv0RkT2M2w1T" crossorigin="anonymous">
<!-- jQuery first, then Popper.js, then Bootstrap JS -->
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JSTQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.14.7/dist/umd/popper.min.js" integrity="sha384-Uo2eT0CpHqdSqQ6hJty5KVphPtPhzWj9W0lclHTMGa3JbUD2wvNq44F86dIHUDe0W1" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/js/bootstrap.min.js" integrity="sha384-JzSmWgyd0p3pXb2Rb27WUAYoIIy60rQ6VrjIEaFf/nJGcTxFd5f4x0x1H+807jRM" crossorigin="anonymous"></script>
```

Figure 92: Import Bootstrap Framework.

Then, example of Bootstrap is the "modal" is a component of the Bootstrap framework. It is used for creating dialog boxes or pop-up windows that appear on top of the main content. The modal component in Bootstrap provides a flexible, customizable, and easy-to-use solution for creating different types of modal windows with various options and styles. Another example is the "nav-bar" by Bootstrap, "nav-bar" is a component of Bootstrap framework that allows you to create responsive navigation menus for your website. The Bootstrap "nav-bar" component provides a wide range of options to create different styles of navigation bars, such as fixed or static, with dropdown menus or search bars, and can be customized to match the style of your website. The "nav-bar" component helps to create a consistent navigation menu across your website, making it easier for users to find and access the different pages of your website. The figure below shows the code of utilizing Bootstrap to create navigation bar.

```

<!-- Navigation bar -->
<nav class="navbar navbar-expand-lg navbar-light bg-dark">
  <h2 class="navbar-brand" style="color:white">Car Sales System</h2>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarNav">
    <ul class="navbar-nav ml-auto">
      <li class="nav-item">
        <a class="nav-link" href="#" data-toggle="modal" style="color:white" data-target="#loginModal">Login</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" style="color:white" href="#about-us">About Us</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" style="color:white" href="#contact-us">Contact Us</a>
      </li>
    </ul>
  </div>
</nav>

```

Figure 93: Code snippet of navigation bar.

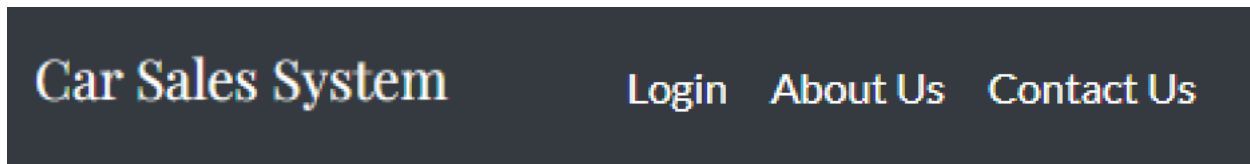


Figure 94: Screenshot of homepage navigation bar.

This is the results of using Bootstrap of navigation bar. Other than that, the image below will show the homepage design with the application of Bootstrap framework.

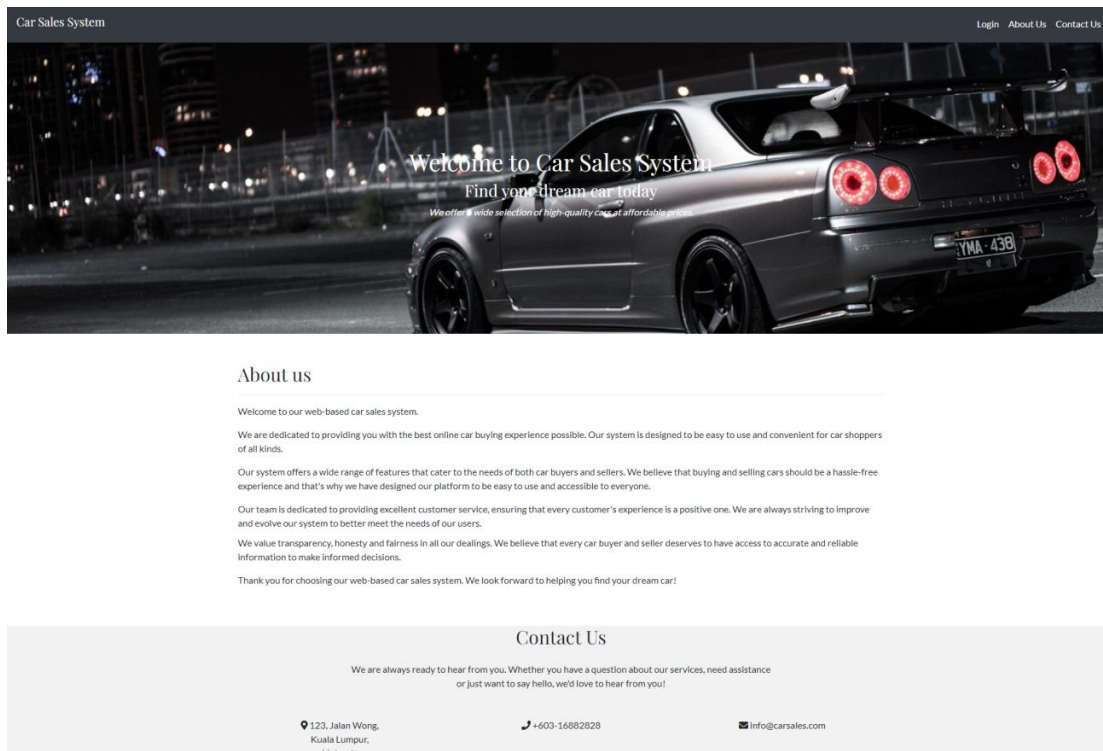


Figure 95: Screenshot of homepage design.

5.2 Cascading Style Sheets (CSS)

CSS is a style sheet language that is used to describe the look and formatting of a document written in a markup language such as HTML. CSS allows web developers to separate the presentation of a web page from its content, making it easier to maintain and update the look and feel of a website. CSS works by defining rules that specify how different elements of an HTML document should be displayed. These rules can define properties such as the color, size, font, and spacing of text, as well as the layout and positioning of page elements. CSS can be used to create responsive designs that adjust to different screen sizes and devices, making it a powerful tool for building modern websites and web applications. It is supported by all major web browsers and is constantly evolving with new features and capabilities.

The car sales system has used the CSS for its front-end designing. The code snippet is attached below to show the application of CSS.

```
/* Change font family of h1 element */
h1, h2, h3, h4 {
  font-family: 'Playfair Display', serif;
}

/* Change font family of p element */
p, a, td, tr, button, btn btn-primary {
  font-family: "Lato", sans-serif;
}

.navbar-nav .nav-link:hover {
  font-weight: bold;
}

.custom-class {
  max-width: 1700px; /* Change this value to your desired width */
  margin: 0 auto; /* Centers the div horizontally */
}

/* Add custom font family */
@import url('https://fonts.googleapis.com/css2?family=Open+Sans:wght@400;600&display=swap');

/* Style for nav items */
#myNavbar .navbar-nav li.nav-item a.nav-link {
  font-family: 'Open Sans', sans-serif;
  font-weight: 600;
  font-size: 18px;
  color: #555;
  text-transform: uppercase;
  padding: 15px;
  transition: all 0.3s ease;
}

.hero-image {
  background-image: url("images/R34.jpg");
  background-color: #cccccc;
  height: 500px;
  background-position: center;
  background-repeat: no-repeat;
  background-size: cover;
  position: relative;
}
```

Figure 96: Code snippet of Cascading Style Sheets.

This is a CSS code snippet that defines styles for different elements in a webpage. Here's a breakdown of each section:

- The first section changes the font family of all heading elements (h1-h4) to "Playfair Display", a serif font.
- The second section changes the font family of several other elements (p, a, td, tr, button, btn btn-primary) to "Lato", a sans-serif font.
- The third section defines a style for when a user hovers over a link in the navbar, making the font weight bold.
- The fourth section defines styles for the navbar items, setting the font family to "Open Sans", setting a font weight of 600, and styling the text to be uppercase, with a 15px padding and a transition effect.
- The final section defines a hero image, which is a large, full-width background image for the webpage. It sets the image URL, height, and background position, as well as setting the background size to cover the entire element. The "position: relative" property sets the position of the element relative to its normal position.

The result of using CSS is shown at image below.

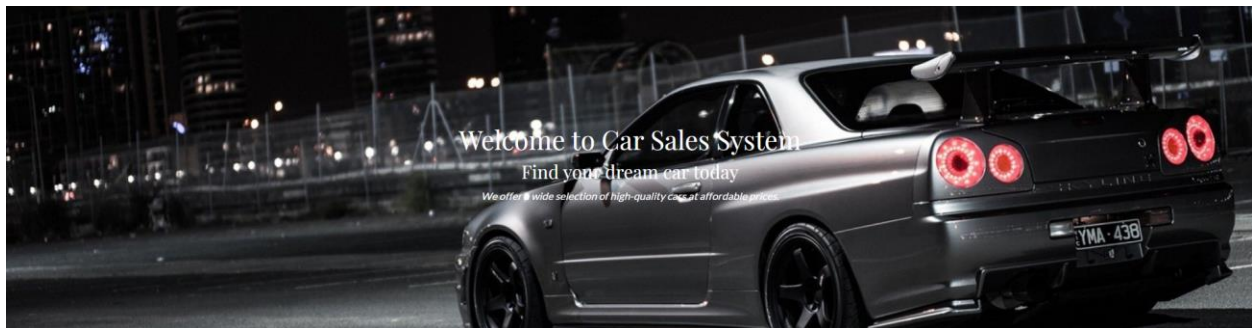


Figure 97: Screenshot of CSS design on homepage.

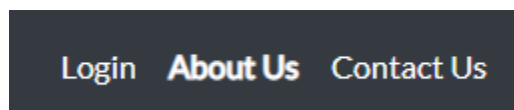


Figure 98: Screenshot of navigation bar with CSS implemented hover animation.

5.3 JavaScript (JS)

JavaScript is a scripting language used to create interactive web pages and applications. It can be used to dynamically manipulate HTML and CSS, respond to user events such as clicks and key presses, and communicate with servers to fetch or submit data without reloading the page. JavaScript code can be embedded directly in HTML files or included as separate files that are loaded by the browser.

```
<script>
function togglePassword() {
    var password = document.getElementById("password");
    if (password.type === "password") {
        password.type = "text";
    } else {
        password.type = "password";
    }
}
</script>
```

Figure 99: Code snippet for javascript togglePassword() function.

The code snippet above shows JavaScript function that toggles the visibility of a password field. When the user clicks the "Show" checkbox next to the password input field, the function is called, which gets the password field by its ID and changes its "type" attribute from "password" to "text" so that the password can be seen. If the user clicks the checkbox again, the function changes the "type" attribute back to "password" to hide the password. This is a useful function for allowing users to see what they are typing into a password field. The image below shows the result of using the togglePassword() function.

Password

☒ Show

Figure 100: Screenshot of togglePassword() function.

```
// Feedback Chart
var ctx = document.getElementById('feedbackChart').getContext('2d');
var chart = new Chart(ctx, {
  type: 'pie',
  data: {
    labels: ['Poor', 'Neutral', 'Good'],
    datasets: [{
      label: 'Feedback Count',
      data: [<%= request.getAttribute("poorCount") %>, <%= request.getAttribute("neutralCount") %>, <%= request.getAttribute("goodCount") %>],
      backgroundColor: ['#FF6384', '#36A2EB', '#FFCE56']
    }]
  },
  options: {
    responsive: true,
    maintainAspectRatio: false,
    aspectRatio: 0.8
  }
});
```

Figure 101: Code snippet of feedback chart with javascript.

This is a JavaScript code that creates a pie chart using the Chart.js library. The chart displays feedback data in three categories: Poor, Neutral, and Good, with the corresponding count of feedback for each category. The chart is created using the canvas element with the ID 'feedbackChart'. The options object contains the responsive, maintainAspectRatio, and aspectRatio properties, which determine the chart's responsiveness and size.

5.4 Charts

Additional features have been added in charts section, the additional charts for report and analysis will be “Gender Report”, “Inventory Report”, “Customer and Salesman Report”. These charts are generated using JavaScript. The “Gender Report” basically shows the total number of male and female from customer and salesman. The “Inventory Report” shows the count status of the car, how many cars are available / booked / paid / cancelled. The “Customer and Salesman Report” shows the total number of salesman and customer registered in the system.

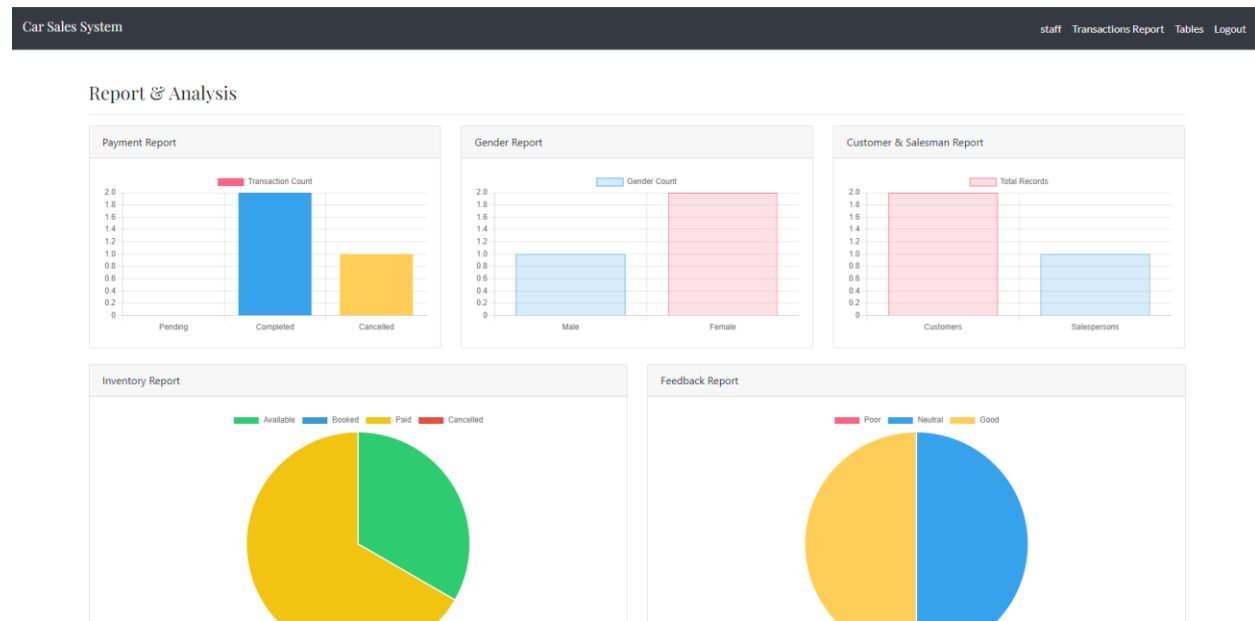


Figure 102: Screenshot of additional feature on report and analysis.

5.5 Transaction Report for Managing Staff

The transaction sales report generates for Managing Staff is different from the sales report for Salesman. Salesman only gets to see their individual sales record; Managing Staff has access to see all transaction sales record including all salesman.

Transaction ID	Transaction Date	Car Information	Car Chassis Number	Price	Customer Feedback & Rating	Customer Name	Salesman Name	Comment	Transaction Status
6	10/03/2023	Toyota 86 2020 Pearl White	FA20QWERTY86	180000	No Feedback Yet No Rating Yet	Tester	Salesman	Booking for full loan payment	Completed
10	10/03/2023	Subaru Forester 2020 Pearl White	FB205259241	250000	No Feedback Yet No Rating Yet	Tester	Salesman1	Full Cash Payment	Pending

Figure 103: Screenshot of transaction report for Managing Staff.

5.6 Set Current Date

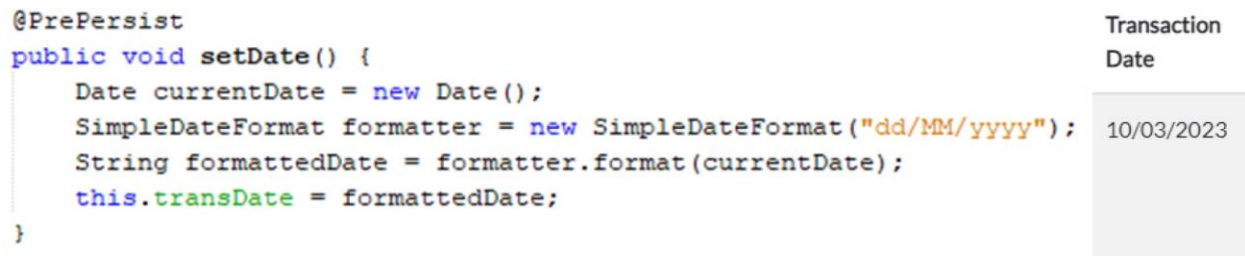


Figure 104: Code snippet and screenshot of set current date.

This is a method annotated with `@PrePersist`, which is a JPA annotation used to specify a callback method to be invoked before a new entity is persisted to the database. In this example, the `setDate()` method is called before a new entity is persisted.

The purpose of this method is to set the value of the `transDate` field to the current date in the format `"dd/MM/yyyy"`. It does this by creating a new `Date` object, formatting it using a `SimpleDateFormat` object, and then setting the value of `transDate` to the formatted date string.

By using the `@PrePersist` annotation, this method is automatically called by the JPA provider before a new entity is persisted to the database. This ensures that the `transDate` field is always set to the current date in the correct format.

5.7 Lombok library

Lombok is a Java library that provides a set of annotations to help reduce boilerplate code in Java classes (Baeldung, 2022). Lombok generates code at compile time, such as getters, setters, constructors, equals, hashCode and toString methods, by simply annotating the class or fields with Lombok annotations (Baeldung, 2022). This makes the code cleaner and more concise, and reduces the time needed to write and maintain code. Lombok also provides additional features such as logging, builders, and value objects that can further simplify Java development. Some popular Lombok annotations include @Getter, @Setter, @Data, @NoArgsConstructor, @AllArgsConstructor, @Builder, and @Slf4j (Baeldung, 2022).

```
package model;

import java.io.Serializable;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import lombok.*;

@Data
@NoArgsConstructor
@AllArgsConstructor
@Entity

public class ManagingStaff implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue
    private Long id;
    private String username;
    private String password;
    private String name;
    private String email;
    private String hpnumber;
    private String usertype;
}
```

Figure 105: Code snippet of Managing Staff entity class with lombok library.

The image above shows the example of importing the Lombok library, by adding @Data, @NoArgsConstructor, @AllArgsConstructor, the codes that generated by Java Entity Class is no longer needed, as Lombok generates them at compile time and it makes the code a lot more cleaner.

6.0 References

Amazon Web Services, Inc. (2023a, January 3). *What are Microservices?* Amazon Web Services. <https://aws.amazon.com/microservices/>

Amazon Web Services, Inc. (2023b, January 3). *What Is Distributed Computing?* Amazon Web Services. <https://aws.amazon.com/what-is/distributed-computing/>

Amazon Web Services, Inc. (2023c, January 3). *What Is Service-Oriented Architecture?* Amazon Web Services. <https://aws.amazon.com/what-is/service-oriented-architecture/>

Anand, S. (2023, January 3). *Top 36 CRM Features and Functionality List*.
<https://www.selecthub.com/customer-relationship-management/crm-features-functionality-list/>

ArunGopinathan. (2015, September 15). *AttendanceManagementSystemUsingQRCode: Attendance Management System using QR Code as a way to mark attendance for students*. GitHub.
<https://github.com/ArunGopinathan/AttendanceManagementSystemUsingQRCode>

aryasinghbjc. (2022, November 22). *Evolution of Distributed Computing Systems*.
GeeksforGeeks. <https://www.geeksforgeeks.org/evolution-of-distributed-computing-systems/>

Baeldung. (2022, July 4). *Introduction to Project Lombok*. Baeldung.
<https://www.baeldung.com/intro-to-project-lombok>

- Banger, E. (2023, January 21). *What is Peer to Peer Network? Architecture, Types, & Examples!!* DigitalThinkerHelp. <https://digitalthinkerhelp.com/what-is-peer-to-peer-p2p-network-with-architecture-types-examples/>
- BasuMallick, C. (2022, February 8). *Top 10 Edge Computing Platforms in 2022*. Spiceworks. <https://www.spiceworks.com/tech/edge-computing/articles/best-edge-computing-platforms/>
- Carter, R. (2022, February 1). *CRM 101: Customer Relationship Management*. CX Today. <https://www.cxtoday.com/crm/crm-101-customer-relationship-management/>
- Edwards, D. (2022, March 10). *What is the difference between ERP, Supply Chain Management and CRM*. SYSPRO Blog. <https://www.syspro.com/blog/supply-chain-management-and-erp/what-is-the-difference-between-erp-supply-chain-management-and-crm/>
- Fernando, J. (2022, July 7). *Supply Chain Management (SCM): How It Works and Why It Is Important*. Investopedia. <https://www.investopedia.com/terms/s/scm.asp>
- GeeksforGeeks. (2022, November 23). *Introduction to JSP*. <https://www.geeksforgeeks.org/introduction-to-jsp/>
- GeeksforGeeks. (2023, January 9). *Socket Programming in Python*. <https://www.geeksforgeeks.org/socket-programming-python/>
- Google Inc. (2023, January 3). *What Is Microservices Architecture?* Google Cloud. <https://cloud.google.com/learn/what-is-microservices-architecture>
- Harris, C. (2023, January 3). *Microservices vs. monolithic architecture*. Atlassian. <https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith>

Hayes, A. (2022, July 30). *The Supply Chain: From Raw Materials to Order Fulfillment*.

Investopedia. <https://www.investopedia.com/terms/s/supplychain.asp>

IBM. (2022, February 10). *Service-oriented architecture (SOA) overview*. IBM Corporation.

<https://www.ibm.com/docs/en/rbd/9.5.1?topic=overview-service-oriented-architecture-soa>

IBM. (2023). *What is Three-Tier Architecture*. <https://www.ibm.com/my-en/topics/three-tier-architecture>

IBM Corporation. (2021, April 19). *What is distributed computing*. IBM.

<https://www.ibm.com/docs/en/txseries/8.2?topic=overview-what-is-distributed-computing>

IBM Corporation. (2023, January 3). *What is SOA (Service-Oriented Architecture)?* IBM.

<https://www.ibm.com/topics/soa>

Intellipaat Software Solutions Pvt. Ltd. (2022, December 13). *What is Client Server*

Architecture? Intellipaat Blog. <https://intellipaat.com/blog/what-is-client-server-architecture/>

IRC Group Global LTD. (2020, January 14). *What is Supply Chain Management?* IRC Group.

<https://ircgroupglobal.com/what-is-supply-chain-management/>

JavaTpoint. (2023a). *Learn Servlet Tutorial - javatpoint*. www.javatpoint.com.

<https://www.javatpoint.com/servlet-tutorial>

JavaTpoint. (2023b). *What is Bootstrap*. www.javatpoint.com. <https://www.javatpoint.com/what-is-bootstrap>

- Kanade, V. (2022, January 19). *What Is Grid Computing? Key Components, Types, and Applications*. Spiceworks. <https://www.spiceworks.com/tech/cloud/articles/what-is-grid-computing/>
- Lund, J. (2023, February 9). *Why CRM Is Key To Unlocking Business Growth Potential*. SuperOffice. <https://www.superoffice.com/blog/what-is-crm/>
- Madooei, A. (2023). *Client-server Application - OOSE*. https://madooei.github.io/cs421_sp20_homepage/client-server-app/
- McCabe, J. D. (2007). Flow Analysis. *Network Analysis, Architecture, and Design*, 161–208. <https://doi.org/10.1016/b978-012370480-1/50005-0>
- Meloeny, S. (2022, January 31). *3 Examples of Different Industries Using ERP*. Calsoft Systems. <https://www.calsoft.com/examples-different-industries-using-erp/>
- Nehra, M. (2021, April 15). *Key Benefits of Service Oriented Architecture*. decipherzone.com. <https://www.decipherzone.com/blog-detail/service-oriented-architecture>
- Netflix. (2023). *Netflix/eureka: AWS Service registry for resilient mid-tier load balancing and failover*. GitHub. <https://github.com/Netflix/eureka>
- Oracle. (2013, January 1). *What Is a Session Bean? - The Java EE 6 Tutorial*. <https://docs.oracle.com/javaee/6/tutorial/doc/gipjg.html>
- Oracle. (2023, January 3). *What is ERP?* <https://www.oracle.com/my/erp/what-is-erp/>
- O'Reilly & Associates. (2023). *JavaServer Pages*. https://docstore.mik.ua/oreilly/java-ent/servlet/ch02_06.htm
- Patil, P. (2022, February 9). *What Is Cloud Computing? Definition, Benefits, Types, and Trends*. Spiceworks. <https://www.spiceworks.com/tech/cloud/articles/what-is-cloud-computing/>

PTC Inc. (2023). *Enterprise Resource Planning (ERP) Definition* / Arena. Arena.

<https://www.arenasolutions.com/resources/glossary/enterprise-resource-planning/>

QuickDesk Pte Ltd. (2023, January 3). *7 Industries That Use CRM the Most*. QuickDesk.

<https://www.quickdesk.io/sales-crm/7-industries-that-use-crm-the-most/>

Ravikiran, A. S. (2023, January 13). *Introduction To Java Servlets and Its Life-Cycle*.

Simplilearn.com. <https://www.simplilearn.com/tutorials/java-tutorial/java-servlets>

Rehman, J. (2023, January 3). *Advantages and disadvantages of service oriented architecture*

(SOA). IT Release. <https://www.itrelease.com/2018/10/advantages-and-disadvantages-of-service-oriented-architecture-soa/>

Roy, R. (2023, January 3). *Key Requirements and Features of Supply Chain Management*

(SCM). <https://www.selecthub.com/supply-chain-management/supply-chain-management-software-features-requirements/>

SarthakGarg. (2022, September 29). *Service-Oriented Architecture*. GeeksforGeeks.

<https://www.geeksforgeeks.org/service-oriented-architecture/>

Schwarz, L. (2022, August 10). *6 Key Phases of an ERP Implementation Plan*. Oracle NetSuite.

<https://www.netsuite.com/portal/resource/articles/erp/erp-implementation-phases.shtml>

Sinha, P. (2022, November 9). *Services Oriented Architecture (SOA): Principles, Benefits, and*

Implementation. Scout APM. <https://scoutapm.com/blog/services-oriented-architecture>

SmartBear Software. (2023, January 3). *What are Microservices?* smartbear.com.

<https://smartbear.com/solutions/microservices/>

Stevens, M. (2005, July 8). *Understanding Service-Oriented Architecture*. Developer.com.

<https://www.developer.com/design/understanding-service-oriented-architecture/>

Stringfellow, A. (2022, June 21). *Java Microservices: Code Examples, Tutorials, and More.*

dzone.com. <https://dzone.com/articles/java-microservices-code-examples-tutorials-and-more>

Svirca, Z. (2021, December 14). *Everything you need to know about MVC architecture -*

Towards Data Science. Medium. <https://towardsdatascience.com/everything-you-need-to-know-about-mvc-architecture-3c827930b4c1>

Talend. (2023, January 3). *What is Service-oriented Architecture (SOA)?*

<https://www.talend.com/resources/service-oriented-architecture/>

Techopedia Inc. (2011a, September 9). *Message-Driven Bean.* Techopedia.com.

<https://www.techopedia.com/definition/24343/message-driven-bean>

Techopedia Inc. (2011b, November 11). *Entity Bean.* Techopedia.com.

<https://www.techopedia.com/definition/24328/entity-bean>

Terra, J. (2022, October 17). *What is Client-Server Architecture? Everything You Should Know.*

Simplilearn.com. <https://www.simplilearn.com/what-is-client-server-architecture-article>

Thomasset, D., & Grobe, M. (2023). *Introduction to the Message Passing Interface (MPI) using*

C. <http://condor.cc.ku.edu/%7Egrobe/docs/intro-MPI-C.shtml>

Trovato, S., & Bottorff, C. (2022, August 11). *CRM Analytics Guide: 7 Key Metrics In 2023.*

Forbes Advisor. <https://www.forbes.com/advisor/business/software/crm-analytics/>

Tyson, M. (2022, September 9). *What is JSP? Introduction to Jakarta Server Pages.* InfoWorld.

<https://www.infoworld.com/article/3336161/what-is-jsp-introduction-to-javascript-server-pages.html>

Viñarás, E. (2022, April 28). *Key Elements of Customer Relationship Management*.

<https://www.cyberclick.net/numericalblogen/key-elements-of-customer-relationship-management>

w3computing.com. (2023). *Service-Oriented Architecture (SOA)*.

<https://www.w3computing.com/systemsanalysis/service-oriented-architecture-soa/>

W3schools of Technology. (2023). *What is JSP? Introduction to Java Server Pages*.

w3schools.in. <https://www.w3schools.in/jsp/intro>