Tribhuvan University

Faculty of Humanities and Social Science

ServiceSync:
Garage Management System

A PROJECT REPORT

Submitted to

Department of Computer Application

Kathmandu BernHardt College

In partial fulfillment of the requirements for the Bachelors in Computer Application

Submitted by

Mibek Shrestha(46102044)

October 2023

Under the Supervision of

Anand Kumar Sah

SUPERVISOR'S RECOMMENDATION

I hereby recommend that this project prepared under my supervision by **Mibek Shrestha** entitled "**ServiceSync: Garage Management System**" in partial fulfilment of the requirements for the degree of Bachelor of Computer Application is recommended for the final evaluation.

…….

Aanand Kumar Sah

SUPERVISOR

BCA Department

Kathmandu BernHardt College

Bafal, Kathmandu

Date:- ……..

LETTER OF APPROVAL

This is to certify that this project prepared by **Mibek Shrestha** entitled "**ServiceSync: Garage Management System**" in partial fulfilment of the requirements for the degree of Bachelor in Computer Application has been evaluated. In our opinion it is satisfactory in the scope and quality as a project for the required degree.

……
Aanand Kumar Sah
SUPERVISOR
BCA Department
Kathmandu BernHardt College
Bafal, Kathmandu

……
Ram Babu Mahato
Co-ordinator
BCA Department
Kathmandu BernHardt College
Bafal, Kathmandu

…….
Internal Examiner
Abhimanu Yadav

……..
External Examiner
Bhoj Raj Joshi

Date:-…….

ABSTRACT

ServiceSync is a web-based garage management system that allows for better management and proper handling of the different repair jobs at hand. This system enables administrators to input customer and vehicle details, create repair tasks with estimated time and cost, assign jobs to mechanics, and track real-time progress. This system provide a user-friendly platform to enter new repair jobs and these job is based on priority and priority is based according to estimated time to finish the job ,on the base of cost and depend upon the numbers of repair . Workers can update task statuses and mark them as completed. The administrator's screen displays the real-time progress, allowing them to monitor ongoing repairs. After completion, the system generates bills based on the actual time taken and tasks performed, considering labor costs and materials used. This streamlines the repair process, improves communication, and enhances operational efficiency in the mechanic shop.

KEYWORDS: ***Garage, Vehicles, Mechanics, Priority Queue, Completion Email, Invoice Generation.***

ACKNOWLEDGEMENT

I would like to extend my sincere appreciation and gratitude to our supervisor **Mr. Anand Kumar Sah** for his guidance and support throughout the duration of this project. His expertise and valuable input have played a crucial role in the development of this project and has ensured smooth progress and development. I would also like to thank the rest of the faculty member for giving us this opportunity to develop this project.

Project Member

Mibek Shrestha

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

CSS	Cascading Style Sheet
DFD	Data Flow Diagram
ERD	Entity Relationship Diagram
HTML	Hyper Text Markup Diagram
JS	JavaScript
JSP	Java Server Page
MYSQL	Microsoft Server Structured Query Language
UI	User Interface

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

1.1 Introduction

Managing a garage can be a difficult task with so many vehicles coming in and out every day with different tasks that need to be performed on each on them. The process of keeping track of what repairs need to be performed and who performed the repairs can quickly become overwhelming with the increase in the number of vehicles. Our system aims to simplify and optimize this task with a garage management system which keeps track of various aspect of an auto repair shop and make the whole job easier for everyone involved in it. The system will be user-friendly and accessible to authorized personnel within the same network [1].

The system's primary function is to enter new customers, their vehicles, and the required repairs, along with the estimated cost and time. This data will be entered by the admin in the admin panel after questioning the customer at the time of check-in. When a job is given to a mechanic, the system updates the admin's screen with real-time information about the job's status and the time it takes to perform each task. The system will automatically create an invoice based on the time spent and the services provided after the project is finished. The garage management system will streamline the entire repair process, allowing employees to focus on their primary duties. The scheduling module of the system will allow the administrator to assign jobs to mechanics depending on their availability, ensuring that the repair work is performed within the time limit specified. The administrator will be able to track the progress of each job and shift duties to other mechanics as needed.

Furthermore, the system will allow the administrator to save important information such as customer and vehicle details, employee information, and maintenance history. This information can be used by the administrator to generate reports on revenue generated, pending payments, and repair work accomplished. The reporting section of the system will provide significant insights into the company's performance, enabling management to make data-driven decisions.

In conclusion, the proposed garage management system will optimize and simplify the way auto repair companies operate. By automating various processes, the system will increase efficiency, reduce manual errors, and improve customer satisfaction.

1.2 Problem Statement

Managing a garage business can be difficult due to the large number of vehicles that come in and out every day, each with its own set of maintenance requirements. Mechanics and service staff must keep track of various kinds of jobs and tasks, which can be difficult to do without the proper tools and methods. Also, manually managing and tracking customer and vehicle information can be time-consuming and error-prone. Many vehicle repair businesses existing method involves manual data entry and tracking, which can be inefficient and error-prone. This frequently results in confusion and delays, resulting in dissatisfied consumers and lost income. additionally, the lack of real-time progress tracking can make it difficult to manage the process and distribute staff effectively.

To address these issues, a comprehensive and user-friendly garage management system is needed to automate and streamline various parts of the auto repair process. This system would allow customers to enter their vehicle and maintenance requests online, making the process simpler and more efficient for both customers and staff. The main challenge in building such a system is to create a user-friendly interface that allows members to enter and track information easily and efficiently. The system must also be flexible enough to support various types of repairs and give real-time notifications to keep clients updated on the status of their jobs. Furthermore, in order to satisfy the needs of a growing organization, the system must be dependable, secure, and expandable.

1.3 Objectives

The main objective of this project is to provide a comprehensive and user-friendly garage management system that automates and keeps track of various aspects. The elaboration of the objective is as follows:

- To provide a user-friendly platform to enter new repair jobs.
- To track time spent on each vehicle and to facilitate bill generation.
- Priority based job completion.

1.4 Scope and Limitation

Scope

- Register and manage customer information, including vehicle details.
- Create and track repair orders with details of repairs, estimated time, and cost.
- Assign repair jobs to mechanics for efficient distribution of workload.
- Real-time progress monitoring allows to monitor the progress of repair jobs in real-time.
- Billing and invoicing: Generate invoices based on completed repair jobs, including time taken and tasks performed.
- Administrator dashboard provides a comprehensive view of operations, pending orders, completed jobs, and financial summaries.

Limitations

- The system may not have all the advanced features and capabilities of more comprehensive garage management systems.
- Need for a proper network infrastructure and device may prevent users from adopting this system.
- The system may have basic security measures but may lack more advanced security features for data protection.

1.5 Development Methodology

Waterfall methodology is used in this project. In waterfall model, each phase must be completed before next phase can begin and there is no overlapping in the phases. This means output of previous phase works as input to another phase.

The major reason behind choosing this methodology is because of its simplicity. It is easy to implement as well. It works well for smaller projects. It is easy to arrange tasks as well. It would be suitable when there is no ambiguous requirement in project.

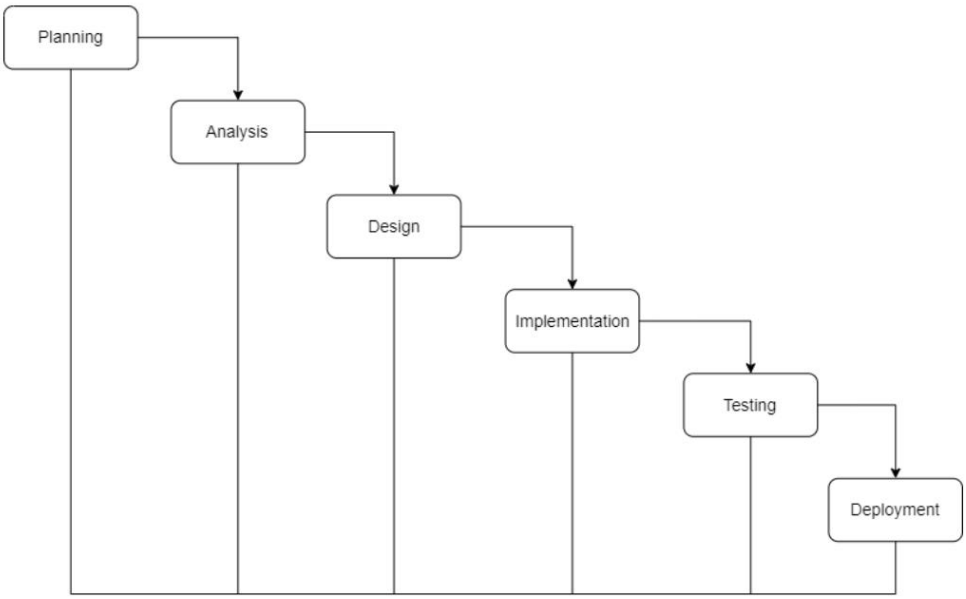


Figure 1.5 : Waterfall Methodology for ServiceSync

1.6 Report Organization

The report on “Service Sync” is organized in the following way:

Chapter 1: Introduction

This chapter provides an introduction to the "Service Sync" garage management system. It outlines the challenges commonly encountered in garage business and how our system addresses these issues. The objectives, scopes, and limitations of the project are also defined in this section.

Chapter 2: Background Study and Literature Review

In this chapter, we delve into a comprehensive background study and literature review related to garage management systems. We explore similar projects and concepts that inform the current project's development.

Chapter 3: System Analysis and Design

This chapter delves into the workings of the "Service Sync" system. It includes feasibility analysis, functional and non-functional analysis, as well as the schema and architectural

design. Data and process modeling diagrams are also presented here. We have also discussed about the algorithms used in our system.

Chapter 4: Implementation and Testing

Chapter 4 covers the implementation and testing phase. It outlines the creation of test cases to verify that the system and its components operate as intended during development. Additionally, we provide insights into the tools employed to complete the "Service Sync" project.

Chapter 5: Conclusion and Future Recommendations

The concluding chapter summarizes the entire "Service Sync" project. It discusses the outcomes and insights gained from its development. Furthermore, future recommendations for enhancing the system are presented at the end of Chapter 5.

CHAPTER 2: BACKGROUND STUDY AND LITERATURE REVIEW

2.1Background Study

The car repair industry is a very dynamic and fast-paced industry in which many vehicles are serviced and repaired on a daily basis. However, running an garage shop may be complicated and time-consuming. Delays, miscommunication, and customer discontent are frequently caused by manual record-keeping, poor job allocation, and a lack of real-time progress tracking.

To optimize and automate different areas of the repair shop operations, a comprehensive garage management system is required. The management and mechanics can gain from increased productivity, improved communication, and better customer service by deploying a digital solution, like a web-based application. A garage management system's main goals are to simplify the customer enrollment process, accurately record vehicle information, produce a thorough list of repairs that need to be done, estimate repair time and cost, assign jobs to mechanics, monitor progress in real-time, and generate invoices for jobs that are finished.

The system's goal is to give the administrator a centralized location where they can effectively manage client data, handle repair requests, check on mechanic performance, and guarantee task completion in a timely manner. Effective resource management and oversight are made possible by real-time progress updates on the administrator's screen.

Auto repair companies may increase operational effectiveness, boost client satisfaction, and optimize resource allocation by putting in place a garage management system. The solution decreases errors, does away with the need for human documentation, and allows for better real-time data-based decision-making.

2.2 Literature Review

While conducting a literature review on garage management systems, we found several projects that share similar goals and objectives to our project. These projects aimed to automate and streamline the management of auto repair shops to improve efficiency and customer service.

One notable project focused on developing a web-based garage management system. The system allowed customers to enter their vehicle information, repair requests, and estimated time and cost. Mechanics could then view the job details, update the progress in real-time, and generate invoices upon completion. The system aimed to simplify the workflow, improve communication between customers and mechanics, and enhance the overall operational efficiency of the repair shop [2].

Another project we came across focused on creating a comprehensive garage management system with additional features such as employee scheduling, inventory management, and customer relationship management. The system aimed to centralize all aspects of the repair shop's operations, from customer intake to job assignment and invoicing, to ensure smooth and efficient management of tasks [3].

Additionally, some projects incorporated advanced features like data analytics and reporting to provide insights into key performance indicators, such as job completion time, customer satisfaction, and revenue generation. These features aimed to assist repair shop owners in making data-driven decisions and optimizing their operations.

Chapter 3: System Analysis and Design

3.1 System Analysis

System analysis is done to investigate a system or its components in order to pinpoint its goals. It is a method of resolving issues that enhances the system and guarantees that every part functions effectively to provide its intended function.

3.1.1 Requirement Analysis

i. <u>Functional requirements</u>

- The system should allow the administrator to add new jobs and customers with car information.
- The system should allow the mechanic to accept a job and update the work's real-time status.
- The system should create bills for completed jobs depending on the amount of time spent and the activities done.
- The system should allow the administrator to examine the progress and status of each job in real time.
- The system should have an intuitive user interface that is simple to navigate and utilize.

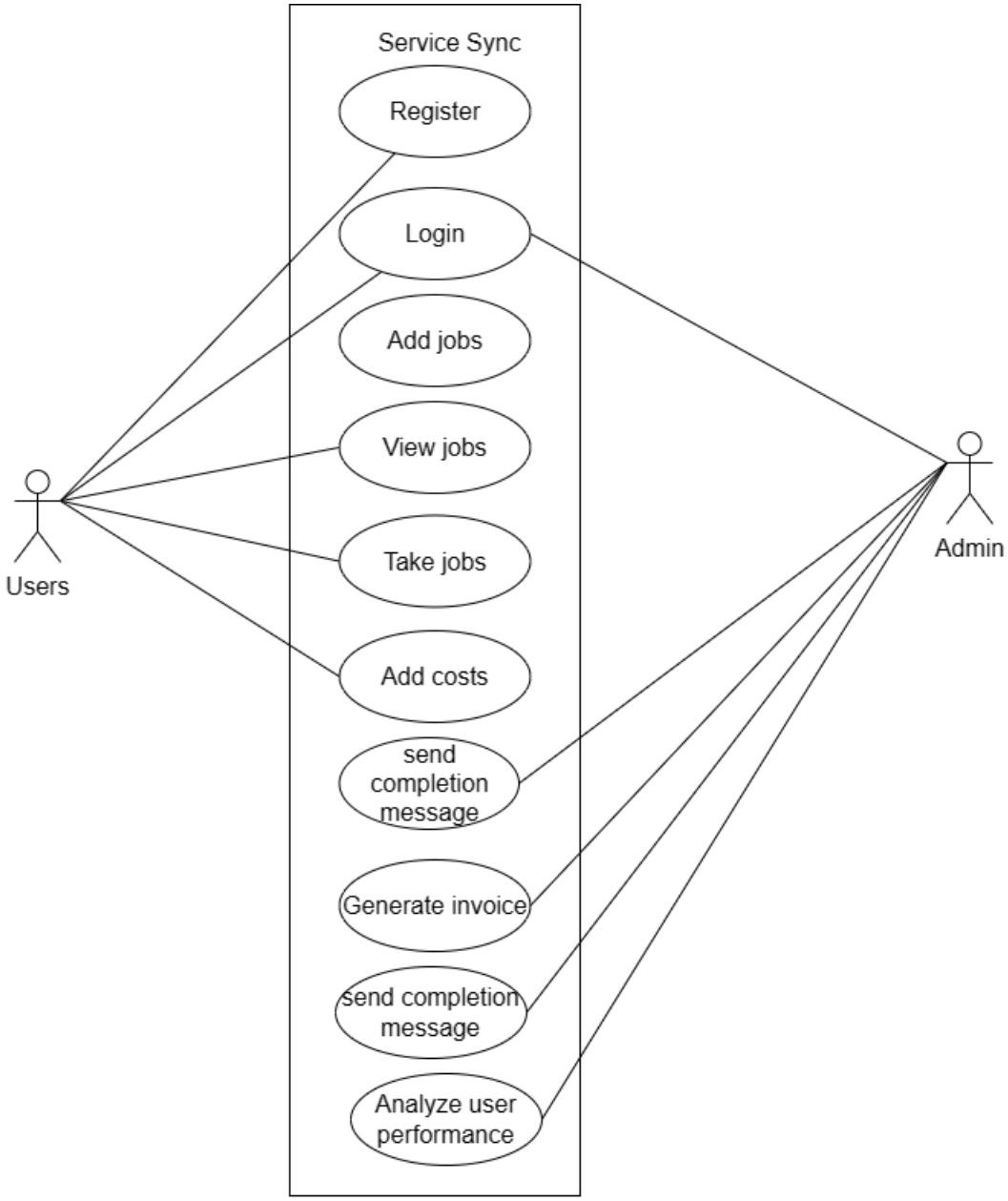


Figure 3.1.1 : Use Case Diagram of Service Sync

The given use-case diagram shows the role of the admin and the user/mechanics in the system. They both will have their own tasks like in the case of admin, they will be responsible for adding new jobs, monitoring them and after the completion of the jobs send completion message to the clients. They will also generate invoice with the cost associated with the repairs and be able to analyze the user's performance. In the case of the user, they will be able to view the available works, take on jobs and add the costs generated during the repairs.

ii. Non-functional requirements

- The system should be fast and responsive, with minimal downtime and interruptions.
- The system should be compatible with different devices and operating systems, allowing users to access it from a range of devices.
- The system should be scalable, able to handle a growing number of jobs and customers without affecting performance.

3.1.2 Feasibility Analysis

A feasibility of a garage management system needs to perform certain studies that focus on its technical, economic and operational feasibility [4].

i. Technical Feasibility

Based on the requirements and objectives of the garage management system, it appears to be technically feasible. The available hardware is enough to operate the current scope of the system efficiently with the help of other open-source software.

ii. Economic Feasibility

The system's development and maintenance costs must be evaluated also considering the cost associated with hardware and software requirements. Since this is not a commercial product, we are not concerned with any revenue stream but we do need to consider the cost generated during the system's development lifecycle and not to go beyond pre-decided budget.

iii. Operational Feasibility

The system needs to be designed to meet the user's requirements and ease of use. The system needs to easy to navigate and fulfills the promises it has made to the user i.e., the mechanics and the administrator. Both administrators and mechanics should find the system user-friendly, simple to understand, and simple to operate. The system should also be able to handle a high number of job orders and consumers without compromising service speed or quality.

3.1.3 Data Modeling :ER Diagram

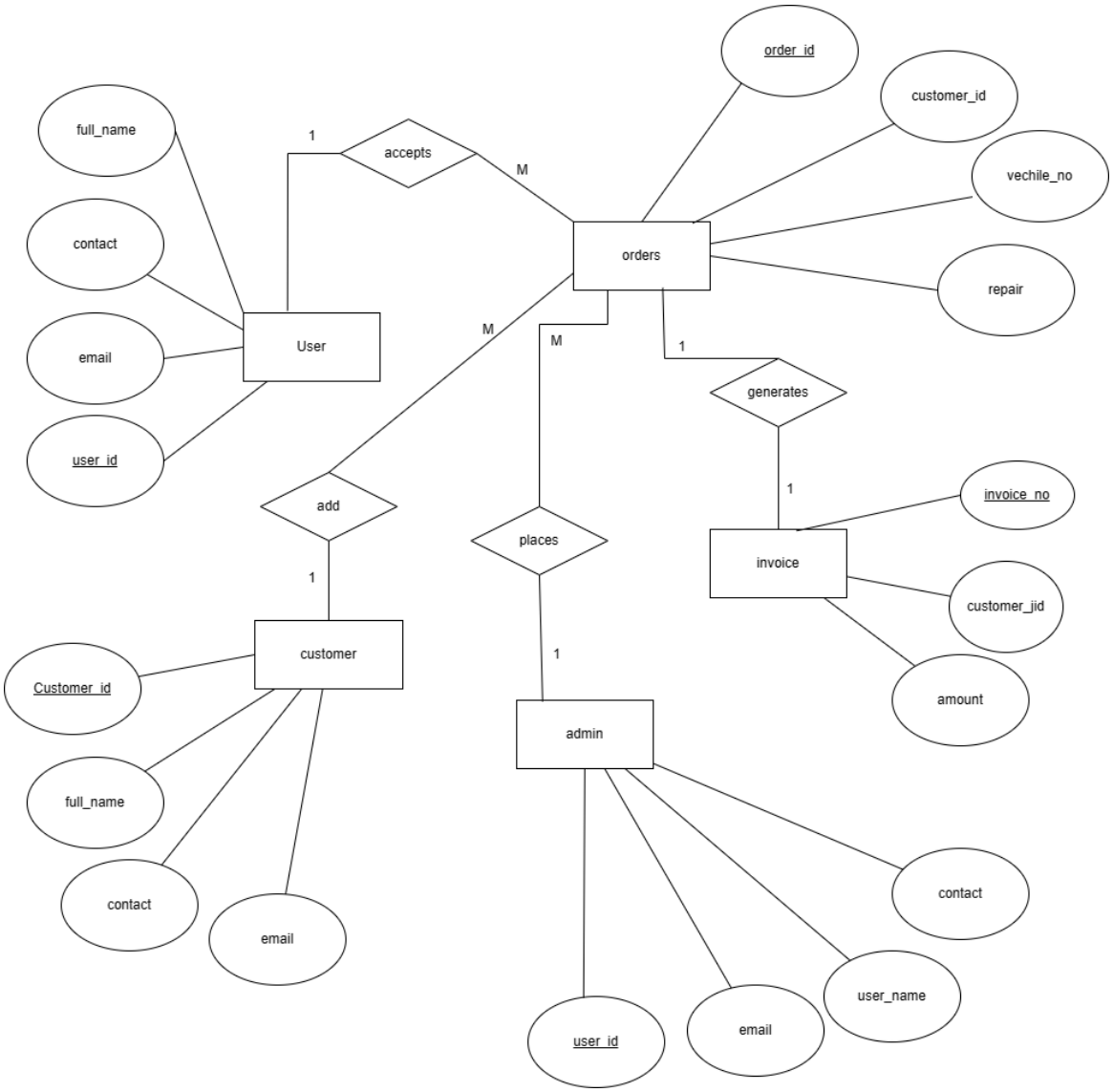


Figure 3.1.3 : ER Diagram

The above entity-relationship diagram shows the different entities involved in the system and the relationship between them.

3.1.4 Process Modeling DFD

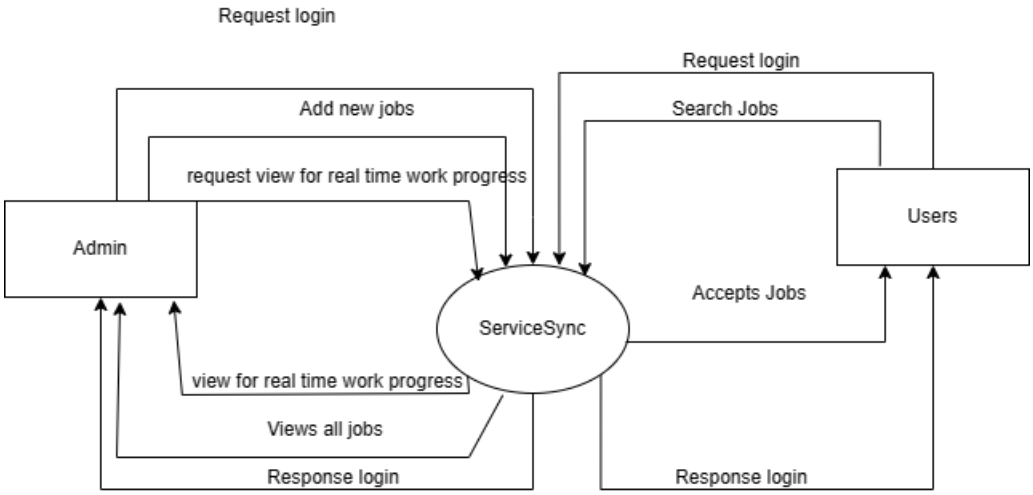


Figure 3.1.4. 1: Level 0 DFD Model

Here in the figure above, we can see the process modeling of our system where the admin is able to request login. After their login is approved, they can add new jobs and view real time work progress of the jobs at hand. Similarly, users are able to log in to the system and take on jobs that are available.

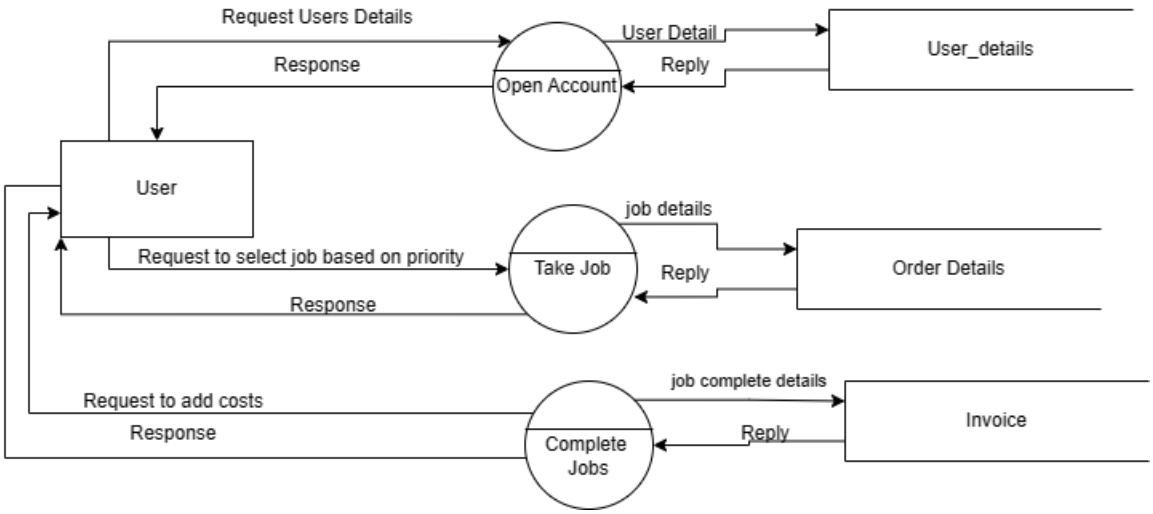


Figure 3.1.4. 2 Level-1 User DFD Model

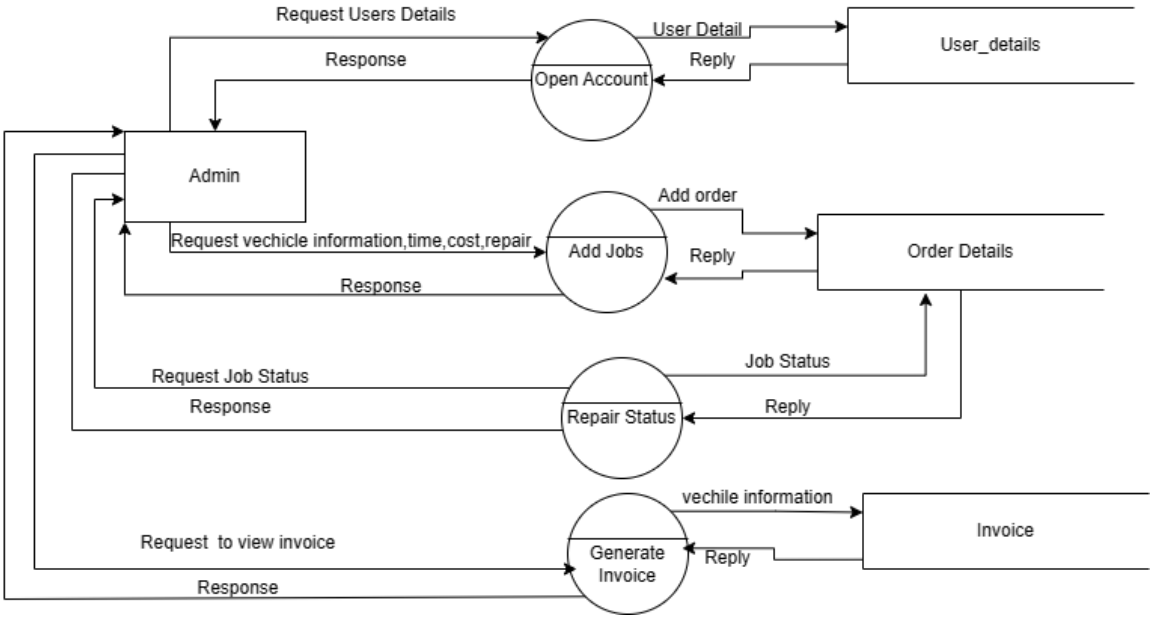


Figure 3.1.4. 3 Level-1Admin DFD Model

3.2 System Design

3.2.1 Architectural Design

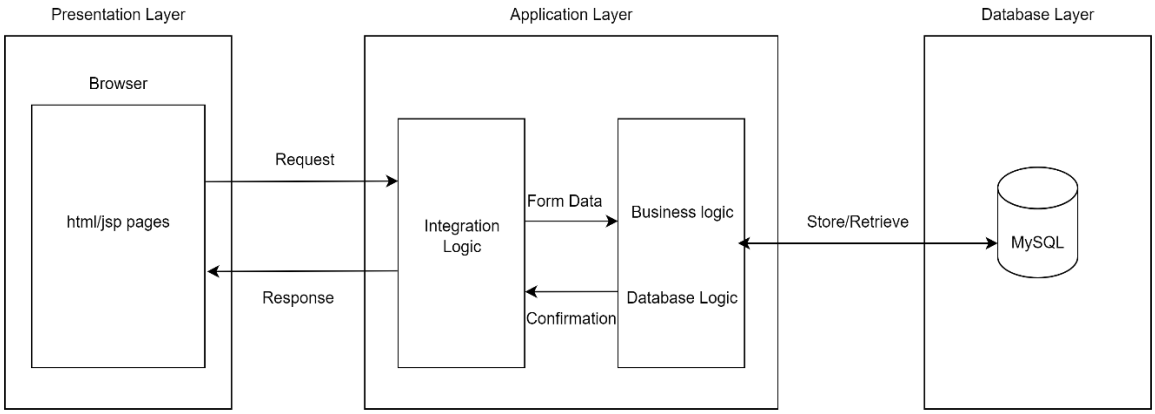


Figure 3.2.1 : Architectural Design

Here, the client-side, server-side, and database layers make up ServiceSync architecture design. The user side, which is part of the client side, is where the JSP/HTML page asks the server side to execute the http methods. The server side is further divided into two sublayers, the servlet component and the business tier component. The server component is responsible for integration logics, while the business tier component handles business layer and database logics. The database layer comprises of a database used to store data, and a business tier component is used to get the desired data.

3.2.2 Database Schema Design

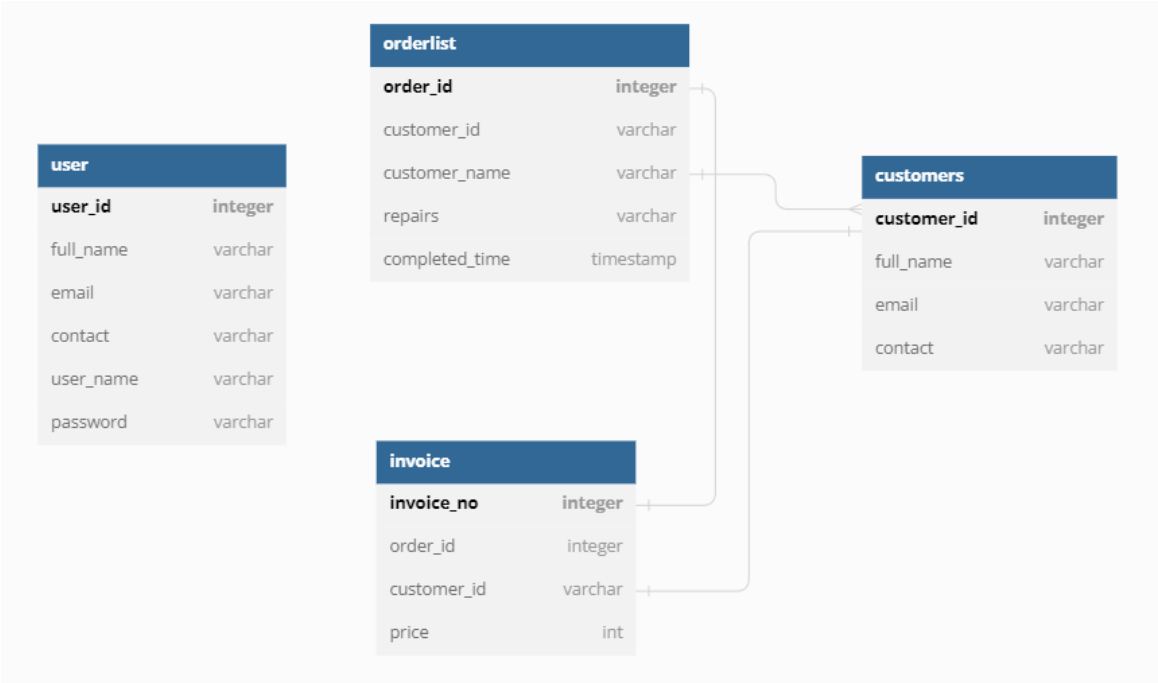


Figure 3.2.2 : Database Schema Design

3.2.3 Interface Design (UI/UX)

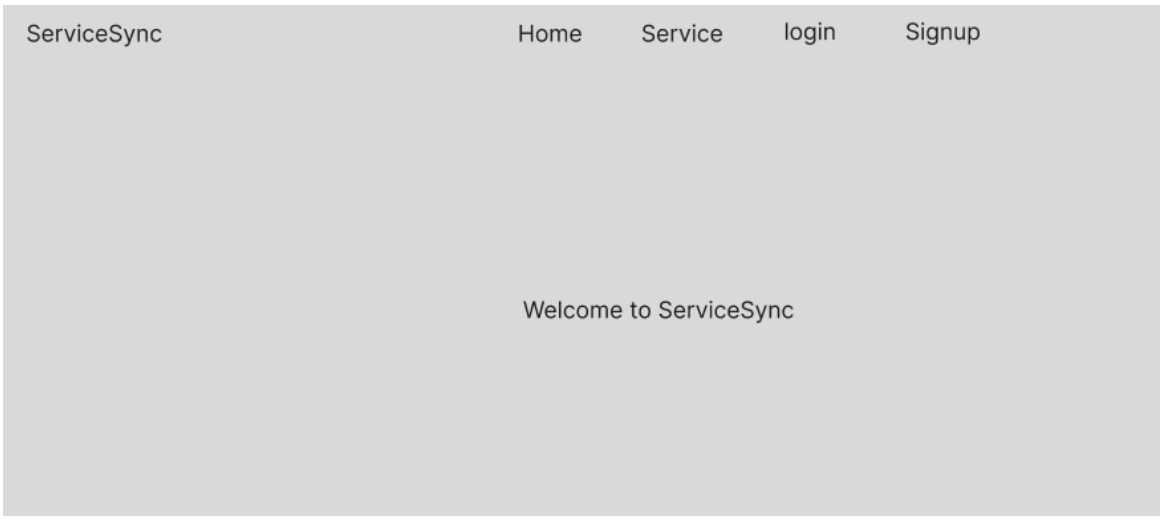


Figure 3.2.3.1: Home page

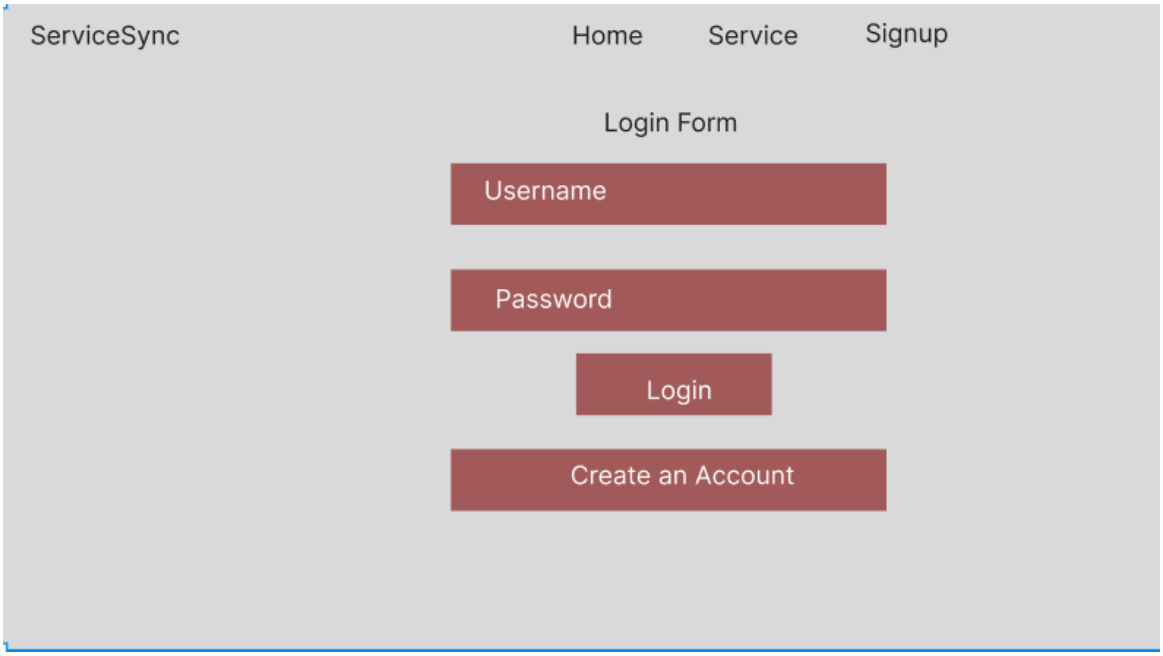


Figure 3.2.3.2: Login Page

3.3 Algorithm

Priority-Based Sorting Algorithm

The garage management system employs a priority-based sorting algorithm to effectively manage the sequence of jobs and tasks. This algorithm plays a crucial role in orchestrating the order in which jobs are addressed, ensuring that the most critical and time-sensitive tasks are given precedence. In our garage management setup, we use a smart system to decide which job gets done when. It's like having a special way of organizing tasks to make everything run smoothly.

Priority-Based Sorting Algorithm**

Objective:

The Priority-Based Sorting Algorithm aims to efficiently manage tasks by assigning priorities based on several factors. This ensures that the most critical tasks are addressed first, optimizing workflow.

Inputs:

- Task start time (startTime)
- Estimated completion time (estimatedCompleted)
- Estimated cost (estimatedCost)

- Concatenated job descriptions (concatenatedJobs)

Outputs:

- Priority score (priority)

Steps:

1. Initialization:Upon receiving task information, the algorithm initializes variables for timeEstimated, amount, and numberOfJobs.

2. Calculating timeEstimated: The algorithm calculates the estimated time for task completion in minutes using the formula:

timeEstimated = (estimatedCompletedMillis - currentTimeMillis) / (60 * 1000)`

Where:

- estimatedCompletedMillis is the time in milliseconds when the task is estimated to be completed.
- currentTimeMillis is the current time in milliseconds when the task is received.

3. Calculating amount:The algorithm extracts the estimated cost (estimatedCost) of the task, which represents the financial value associated with the task.

4. Calculating numberOfJobs: The algorithm determines the number of concatenated job descriptions (concatenatedJobs) by splitting them based on a delimiter (". "). If no jobs are provided, numberOfJobs is set to 0; otherwise, it counts the number of jobs.

5. Assigning Priority Based on timeEstimated: The algorithm assesses timeEstimated to assign a priority score (pr). It considers the following formula to determine priority:

pr = 0 (Initial priority)

If timeEstimated is less than or equal to 1 hour:

pr += 1

If timeEstimated is less than or equal to 30 minutes:

pr += 1

If timeEstimated is less than 24 hours (1 day):

pr += 3 + (urgencyFactor * 5)

Where:

- urgencyFactor = 1.0 - (timeEstimated / (24 * 60 * 60))(The closer the task is to the deadline, the higher the factor).

6. Assigning Priority Based on amount: The algorithm assigns priority based on the estimated cost (amount). It uses the following formula:

If amount is greater than or equal to 500:

pr += 1

If amount is greater than or equal to 1000:

pr += 1

If amount is greater than or equal to 5000:

pr += 1

If amount is greater than or equal to 10000:

pr += 1

7. Assigning Priority Based on numberOfJobs: The algorithm considers the number of concatenated jobs. It uses the following formula:

If numberOfJobs is greater than 0 and less than or equal to 5:

`pr += numberOfJobs` (More jobs increase priority, up to a maximum of 5 additional points).

8. Priority Adjustment Based on Estimated Duration: The algorithm calculates the estimated job duration in days and adjusts the priority accordingly. It uses the following formula:

If estimatedDurationDays is less than or equal to 1:

pr += 7 (Highest priority for tasks estimated to complete within 1 day)

If estimatedDurationDays is less than or equal to 2:

pr += 6

If estimatedDurationDays is less than or equal to 3:

pr += 5

If estimatedDurationDays is less than or equal to 4:

pr += 4

If estimatedDurationDays is less than or equal to 5:

pr += 3

If estimatedDurationDays is less than or equal to 6:

pr += 2

If estimatedDurationDays is less than or equal to 7:
pr += 1 (Priority for tasks estimated to complete within 7 days)

9. Final Priority Score: The algorithm calculates the final priority score (priority) by summing up the scores from the previous steps.

10. Output:The calculated priority score is returned as the output, providing a measure of the task's importance and urgency.

The Priority-Based Sorting Algorithm is a crucial component of our garage management system, ensuring that tasks are sequenced optimally for timely and efficient completion.

CHAPTER 4: IMPLEMENTATION AND TESTING

4.1 Implementation

4.1.1 Tools used (CASE tools, programming languages, database platforms)

For the designing and implementation of the system code we used the following CASE tools given below:

1.CASE Tools

Eclipse IDE

Eclipse is a popular integrated development environment (IDE) for Java programming. We used Eclipse to write and manage our Java code, providing a robust environment with features like code editing, debugging, and project management.

Xampp

XAMPP has the ability to serve web pages on the World Wide Web. We hosted our system in our own devices using Xampp and we used it to access and manipulate our databases in MYSQL.

Tomcat Server

Apache Tomcat is an open-source web server and servlet container that is used to deploy and run Java web applications. We used Tomcat as our server to host and run our Java-based web application, allowing us to serve web pages, handle requests, and manage the application's lifecycle. Tomcat provides a reliable and efficient environment for deploying and testing web applications.

draw.io

draw.io is a free and open-source cross-platform graph drawing software. We used it to create diagrams such as flowcharts, UML diagrams, and ER Diagram.

2. Programming languages

For coding purpose, we used the following programming languages for our project

HTML

The Hyper Text Markup Language or HTML is the standard markup language for documents designed to be displayed in a web browser. All the frontend development was done using HTML and to enhance its style we used CSS.

Java

Java is a widely-used programming language that is often used for building web applications. In our project, we utilized Java for the backend development. We implemented the business logic, database connections, and processing of data using Java

JSP

JSP is a technology that allows the embedding of Java code into HTML pages. It provides a way to dynamically generate HTML content on the server-side. In our project, we used JSP for the presentation layer, where we combined Java code with HTML markup to generate dynamic web pages

MySQL

MySQL is a relational database management system based on SQL – Structured Query Language. Database used to store all information in this system was created and maintained using MySQL.

JavaScript

JavaScript is used to create dynamic and interactive web content. We used JavaScript for various client-side validation. We also used JavaScript to make our system more user-friendly and interactive.

4.1.2 Implementation details of modules

Login Validation

This JSP module facilitates user authentication for "System Sync." It verifies user credentials against a MySQL database, sets up sessions for valid users, and redirects them to admin or user dashboards accordingly. Error handling is included for invalid logins.

Registration Validation

This JSP code provides user registration functionality for a web application. Users can input their full name, email, contact number, username, and password. The code includes client-side form validation using JavaScript. It also handles server-side registration by inserting user data into a database.

The registration form checks for various validations, such as full name, email, contact, username, and password. If the form is filled out correctly, the data is inserted into the database. If there are any errors, they are displayed to the user

Job entering module

This module serves as a user interface for creating repair orders in the "ServiceSync" system. Users can input customer details, vehicle information, and a list of repair jobs. The system automatically timestamps the order's start time. Additionally, it enables users to dynamically add multiple repair jobs. Upon submission, the order data is stored in the database for future reference**.**

Priority calculation module

The priority module is a crucial component in job prioritization and scheduling systems. It takes essential inputs like start time, estimated completion time, estimated cost, and a concatenated job description. With this information, it calculates the estimated time needed for the job in minutes, the job's cost, and the number of individual jobs within the description.

The core function, 'calculate_priority()', assigns priority values to jobs. It considers the urgency of completion, where imminent deadlines result in higher priority. The estimated cost of the job also affects priority, with higher costs leading to higher prioritization. Additionally, the number of individual jobs plays a role, as more jobs can increase priority up to a certain limit. Lastly, the estimated duration of the job is factored in.

This module streamlines job management, ensuring efficient scheduling and prompt handling of high-priority tasks. It's particularly valuable in scenarios demanding intricate considerations of time, cost, and workload, enhancing overall operational efficiency.

Repair Module

This module is responsible for displaying detailed information about a repair order in the "ServiceSync" system. It allows users to start a stopwatch to track the time taken for repairs. Users can also mark individual repair tasks as completed using checkboxes. Once all repairs are completed, the "Complete" button becomes enabled, allowing the user to mark the entire order as complete.

Admin dashboard

The given module is for an admin dashboard in a garage management system. It connects to a database and displays job details, including customer names, vehicle numbers, repair counts, priorities, and statuses, in a tabular format. It also calculates and shows the count of repairs for each job. This dashboard is designed to assist administrators in overseeing and prioritizing jobs based on their status and priority levels, contributing to efficient garage operations.

Invoice generation module

This JSP code creates an invoice list webpage for a garage management system. It connects to a database, retrieves invoice data, and displays it in a tabular format. The page includes a scrollbar for a user-friendly display of a potentially long list of invoices. Each invoice entry shows details such as invoice ID, customer name, mechanic, invoice date, total amount, paid amount, payment status, and a "Print" button to generate a printable invoice.

User report module

This module enables users to assess mechanics or users' performance within a specified date range by summarizing the number of jobs they've undertaken and the total time spent on those tasks. Users input a date range, and the module retrieves relevant invoice records from a database. It then calculates the total number of jobs and cumulative work time, presenting this data in an HTML table. A 'formatTime' function converts the time into a more readable format. This functionality provides valuable insights into work efficiency and productivity, aiding in performance analysis and decision-making.

4.2 Testing

4.2.1 Test Cases for Unit Testing

Login:

Table 4.2.1.1 : Test case for Login

Test Case	Test Data	Expected Outcome	Test-Result
Login	Username: null Password : null	Login not successful	Pass
	Username: bikash Password: hello123	Redirect to admin page	Pass
	Username: anikashstha Password : hello123	Redirect to customer page	Pass

Registration:

Table 4.2.1.1: Test case for Registration

Test Case	Test Data	Expected Outcome	Test Result
Registration	Any input field: null Full name: 1134 email: abcd contact: 1234 username: abcd password:1234	Registration not successful	Pass
	Full name: Mibek Shrestha email: mibekstha @gmail.com contact:98180601178 username: mibekstha password:hello123	Registration successful	Pass

Priority Calculation:

Table 4.2.1.3: Test case for Priority Calculation

Test Case	Test Case Description	Test Data	Expected Result	Test Result
Priority	Test the priority calculation when timeEstimated is low.	timeEstimated = 300, amount = 100, numberOfJobs = 3	Expected priority = 4	Pass
	Test the priority calculation when timeEstimated is moderate.	timeEstimated = 7200, amount = 2500, numberOfJobs = 2	Expected priority = 6	Pass
	Test the priority calculation when timeEstimated is high.	timeEstimated = 95000, amount = 15000, numberOfJobs = 7	Expected priority = 12	Pass
	Test the priority calculation with minimum amount.	timeEstimated = 18000, amount = 50, numberOfJobs = 1	Expected priority = 2	Pass
	Test the priority calculation with maximum amount.	timeEstimated = 3600, amount = 20000, numberOfJobs = 5	Expected priority = 7	Pass
	Test the priority calculation with no concatenated jobs.	timeEstimated = 14400, amount = 750, numberOfJobs = 0	Expected priority = 6	Pass
	Test the priority calculation with the maximum number of jobs.	timeEstimated = 7200, amount = 500, numberOfJobs = 6	Expected priority = 11	Pass
	Test the priority calculation for a short estimated duration.	timeEstimated = 7200, amount = 800, numberOfJobs = 3	Expected priority = 8	Pass

	Test the priority calculation for a long estimated duration.	timeEstimated = 259200, amount = 2000, numberOfJobs = 1	Expected priority = 13	Pass
	Test the priority calculation for a medium estimated duration.	timeEstimated = 14400, amount = 1200, numberOfJobs = 4	Expected priority = 9	Pass
	Test the priority calculation for invalid inputs.	timeEstimated = -100, amount = -50, numberOfJobs = -1	Expected priority = 0	Pass

Priority based Scheduling:

Table 4.2.1.4: Test case for Priority based Scheduling

Test Case	**Test Case Description**	**Test Data**	**Expected Result**	**Test Result**
Priority based scheduling	Test that a higher priority job is scheduled first before a lower priority order.	Order ID: 601 (Priority: 3), Start Time: 2023-07-15 10:00, Estimated Completed: 2023-07-15 12:00, Estimated Cost: 150.00, Jobs: Brake Repair, Oil Change	Schedules job for Order ID 601 before Order ID 602 (Priority: 2).	Pass
	Test that a lower priority order is not scheduled before a higher priority job.	Order ID: 602 (Priority: 2), Start Time: 2023-07-15 11:00, Estimated Completed: 2023-07-15 14:00, Estimated Cost: 80.00, Jobs: Tire Rotation	Mechanic schedules job for Order ID 602 after Order ID 601 (Priority: 3).	Pass

	Test that a mechanic schedules jobs based on priority correctly.	Order ID: 603 (Priority: 4), Start Time: 2023-07-15 09:00, Estimated Completed: 2023-07-15 10:30, Estimated Cost: 60.00, Jobs: Oil Change	Mechanic schedules job for Order ID 603 before Order ID 601 (Priority: 3).	Pass

4.2.2 Test Cases for System Testing

Scenario 1: Employee login and job selection

- Employee visits the login page.
- It the employee enters the correct login details; they are redirected to home page.
- They can either enter new jobs or taken on existing jobs.
- They choose from the existing available jobs that are arranged according to their priority.
- They perform mechanical repairs and confirm job completion.
- Details about the repair are saved for invoice generation.
- They send job completion email to the clients.

Scenario 2: Admin login

- Admin goes to the login page.
- After entering the correct login details, the admin is redirected to the admin dashboard.
- From the dashboard the admin can all the jobs that are available, ongoing or completed.
- They can view all the invoice details
- Employee details can be viewed.

CHAPTER 5: CONCLUSION AND FUTURE RECOMMENDATIONS

5.1 Conclusion

In summary, our project introduces a user-friendly system that simplifies the process of handling new customers, their vehicles, repair tasks, estimated timeframes, and costs. The system ensures seamless coordination between customers, mechanics, and administrators through an easy-to-use interface. This is made possible by utilizing an Apache Server for hosting and MySQL for database management, both conveniently provided by Xampp.

Developing the garage management system provided essential insights for effective software development.

Throughout the development process, we learned valuable lessons about effective software design and development. We realized the importance of creating interfaces that are simple and accessible for everyone. Breaking down complex tasks into smaller parts showed us how this can make the development process easier. Using optimized algorithms taught us how technology can help speed up tasks.

The guidance from our supervisor was really helpful and taught us the value of feedback and learning from experienced individuals. Using tools like Git helped us manage our project better and work as a team. Creating clear and helpful documentation showed us how important it is for users to understand and use our system.

In conclusion, our garage management system offers a practical solution for everyday challenges and taught us important skills in designing user-friendly software. Just like a well-organized garage, successful software needs a blend of thoughtful design, technical know-how, and good communication.

5.2 Lessons learnt

Developing the garage management system provided a rich learning experience, highlighting key insights that contribute to effective software development. As the project evolved, a range of valuable lessons were learned, offering essential guidance for future endeavors and fostering overall growth in software design and implementation.

- **User-Centric Design**: Prioritize user experience and interface design for better engagement.
- **Modular Approach**: Break the system into manageable components for easier development and maintenance.
- **Algorithm Efficiency**: Optimize algorithms, improving performance and responsiveness.
- **Guidance Matters**: Value expert input and feedback, enhancing project quality.
- **Version Control**: Use tools like Git for organized project management and collaboration.
- **Thorough Documentation**: Clear documentation aids users and eases troubleshooting.

5.3 Future recommendation

User Feedback Loop: Establish regular user feedback channels to refine and adapt the system based on real-world usage.

Extended Reporting: Develop more detailed reporting features for better insights into operations and decision-making.

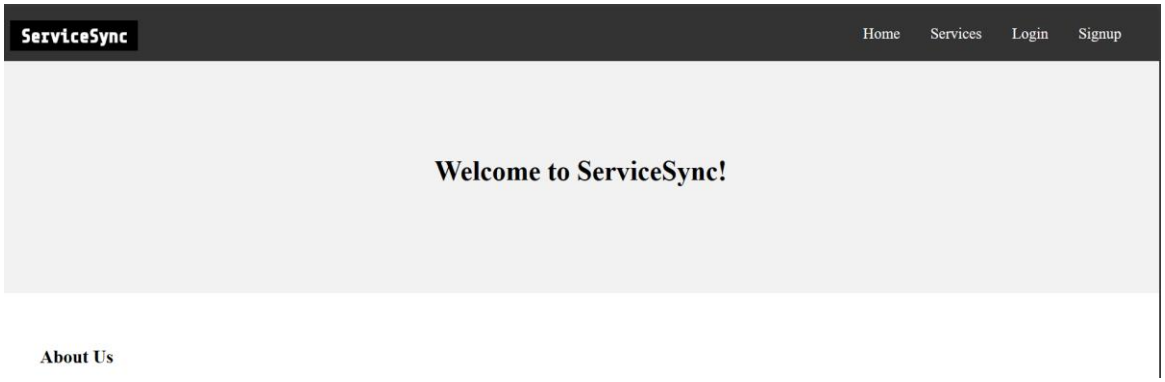
Mobile Access: Consider a mobile app or responsive interface for on-the-go access and convenience.

Diverse Payment Options: Integrate multiple payment gateways to offer varied payment choices to customers.

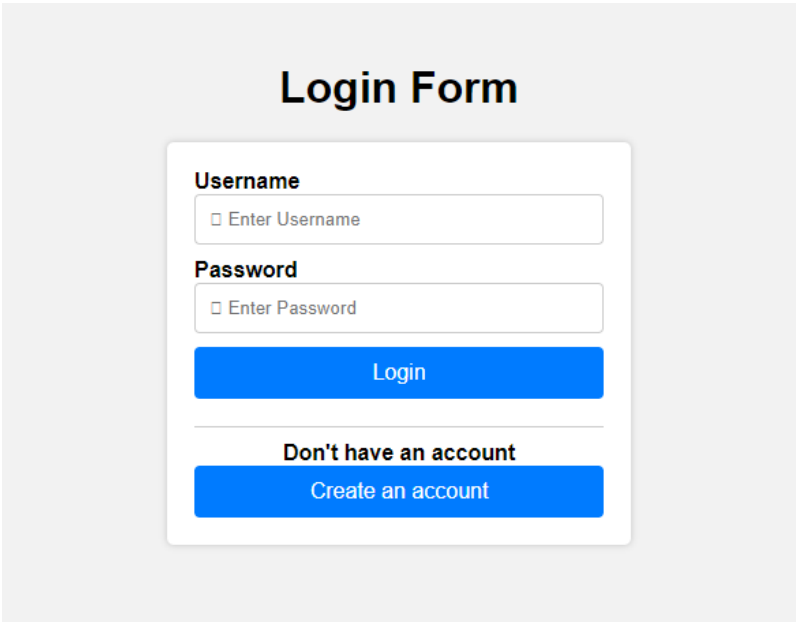
Automated Reminders: Implement automated reminders for customer appointments to reduce no-shows.

Predictive Maintenance: Explore predictive maintenance features to prevent breakdowns throughdataanalysi

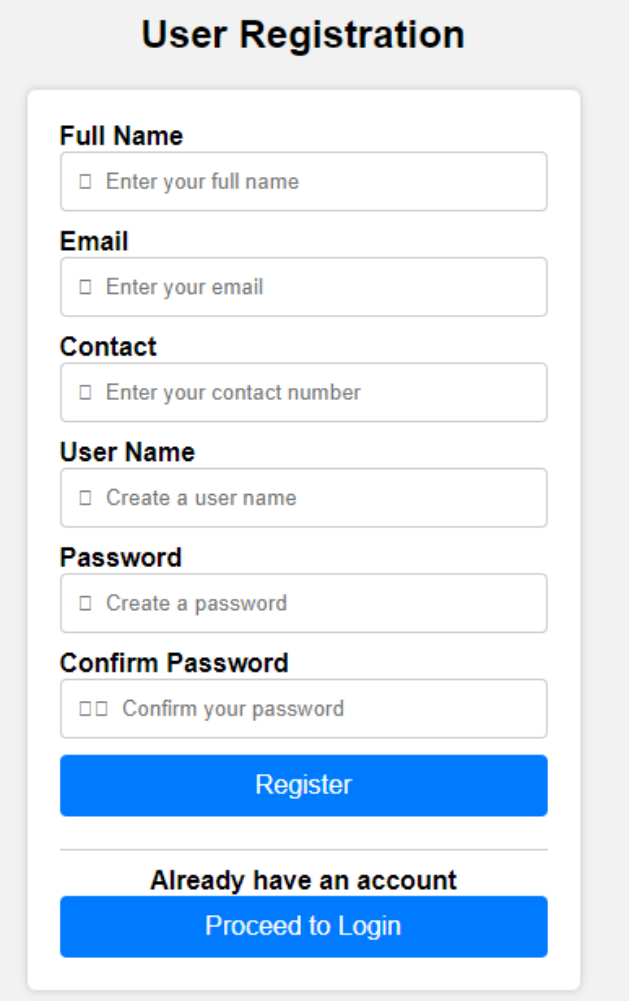
APPENDICES



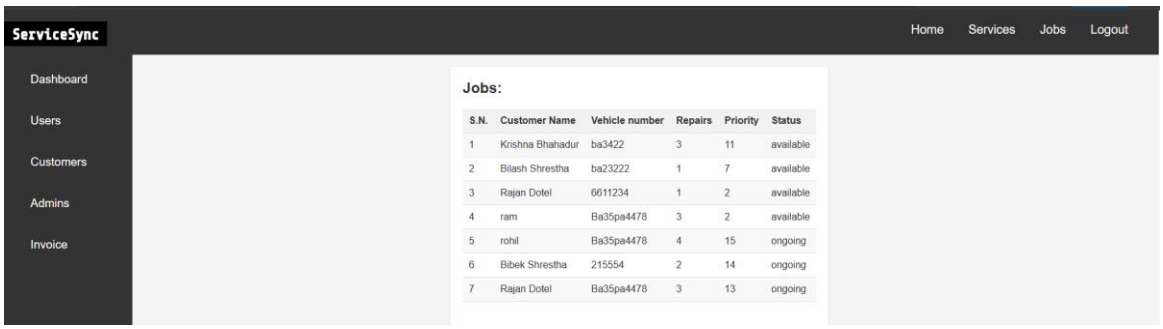
Home Page



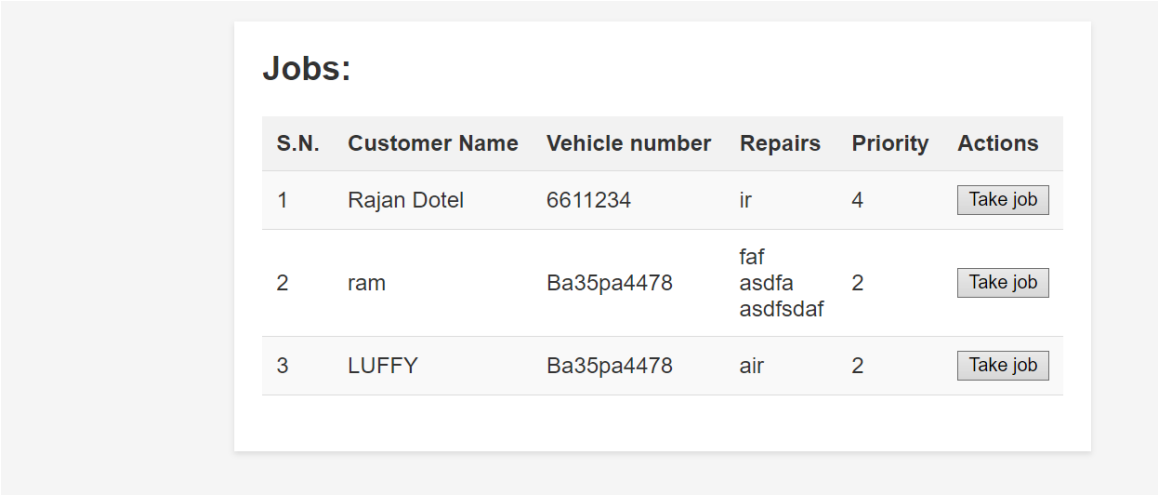
Login Form



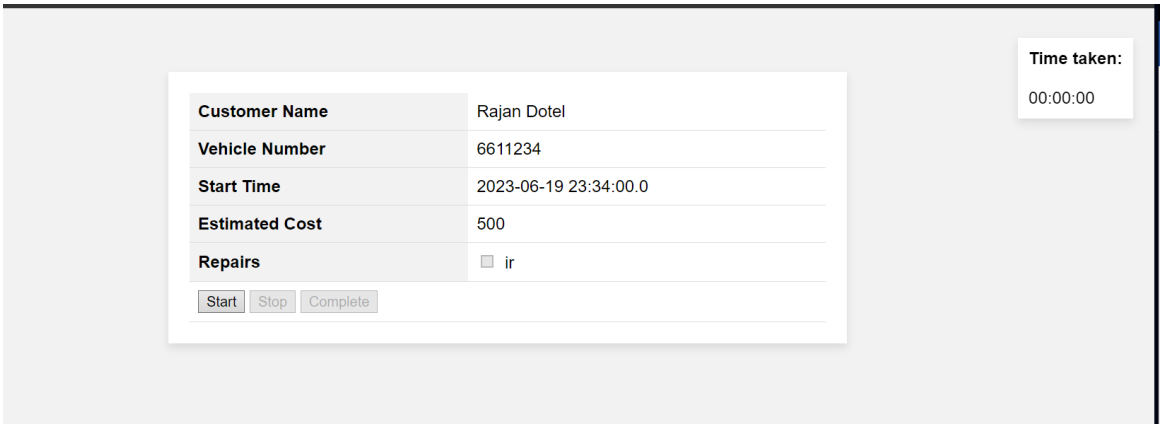
Registration Page



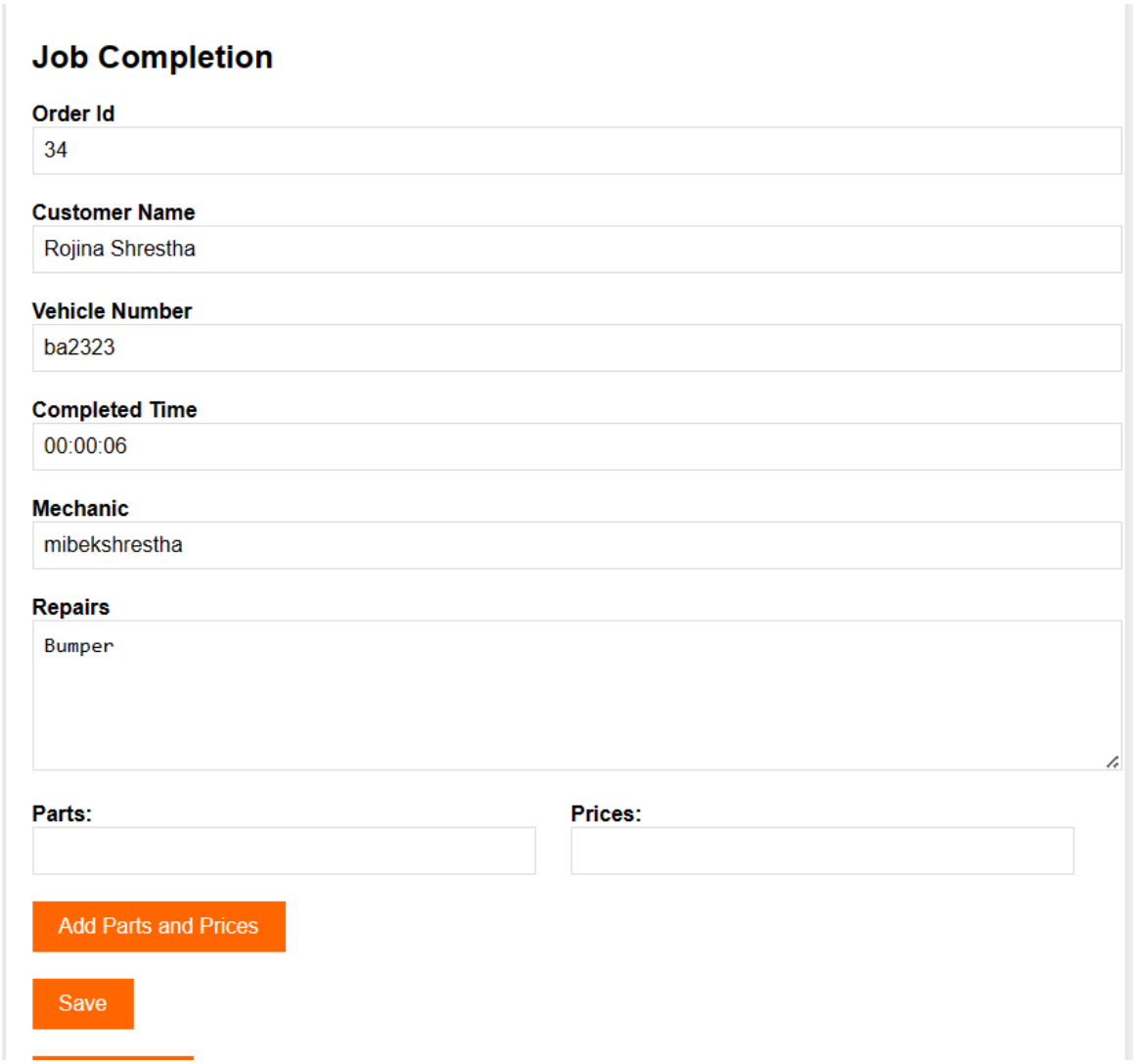
Admin page



Jobs page



Repair Page



Job Completion page

REFERENCES

[1] S. Sae-Khu, "Scandinavian Auto Shop Online," Bangkok, 2004.

[2] A. Patidar, "researchgate.net," October 2021. [Online]. Available: https://www.researchgate.net/publication/356063729_.

[3] mayuri_k, "sourcecodester.com," july 2022. [Online]. Available: https://www.sourcecodester.com/php/15485/garage-management-system-using-phpmysql-source-code.html.

[4] Simplilearn, "feasibility study article," 20 March 2023. [Online]. Available: https://www.simplilearn.com/feasibility-study-article.