TRANSPORT MANAGEMENT SYSTEM

The Project submitted to

Sant Gadgebaba Amravati University, Amravati

Towards partial fulfilment of the Degree of

Bachelor of Engineering

In

Information Technology

Guided by

Dr. A. S. Manekar

Submitted by

Mr. Nayan Sonkalyari
Mr. Advait Patil
Mr. Tiwar Zade
Mr. Kaushal Sharma

DEPARTMENT OF INFORMATION TECHNOLOGY
SHRI SANT GAJANAN MAHARAJ COLLEGE OF ENGINEERING, SHEGAON (M.S.)
2022- 2023

01

SHRI SANT GAJANAN MAHARAJ COLLEGE OF ENGINEERING, SHEGAON

EST.1983

2022-2023

CERTIFICATE

This is to certify that **Mr Nayan Sonkalyari, Mr. Advait Patil, Mr. Tiwar Zade, Mr. Kaushal Sharma,** students of final year B.E. (Information Technology) in the year 2022-2023 of the Information Technology Department of this institute have completed the project work entitled "**Transport Management System**" based on syllabus and has submitted a satisfactory account of his/her work in this report which is recommended for the partial fulfilment of the degree of Bachelor of Engineering in Information Technology.

[illegible]

Name of Guide
Dr. A. S. Manekar
Head of the Department

[illegible]

Dr. A S Manekar
Head of the Department
SSGMCE, Shegaon

[illegible]

Dr. S. B. Somani
Principal
SSGMCE, Shegaon

SHRI SANT GAJANAN MAHARAJ COLLEGE OF ENGINEERING, SHEGAON

2022-2023

<u>CERTIFICATE</u>

This is to certify that the project work entitled "**Transport Management System**" submitted by **Mr Nayan Sonkalyari, Mr. Advait Patil, Mr. Tiwar Zade, Mr. Kaushal Sharma**, students of final year B.E. (Information Technology) in the year 2022-2023 of the Information Technology Department of this institute, is a satisfactory account of his work based on the syllabus which is approved for the award of the degree of Bachelor of Engineering in Information Technology.

Internal Examiner	External Examiner
Date:	Date:

ACKNOWLEDGEMENT

The real spirit of achieving goals through the way of excellence and lustrous discipline. We would have never succeeded in completing our task without the cooperation, encouragement and help provided to us by various personalities.

We would like to take this opportunity to express our heartfelt thanks to our guide ***Dr. A. S. Manekar,*** *for his esteemed guidance and encouragement, especially through difficult times. His suggestions broaden our vision and guide us to succeed in this work. We are also very grateful for his guidance and comments while studying part of our project and learning many things under his leadership.*

We would also like to extend our sincere thanks to ***Prof. F. I. Khandwani****, Project-In-Charge for his valuable support and feedback during the entire course of the project.*

We also extend our thanks to ***Dr. A. S. Manekar****, Head of Information Technology Department, Shri Sant Gajanan Maharaj College of Engineering, Shegaon for providing us with a variety of opportunities and inspirations to gather professional knowledge and material that made us consistent performers.*

We also extend our thanks to ***Dr. S. B. Somani****, Principal, Shri Sant Gajanan Maharaj College of Engineering, Shegaon for providing us the infrastructure and facilities without which it was impossible to complete this work.*

Also, we would like to thank all teaching and non-teaching staff of the department for their encouragement, cooperation and help. Our greatest thanks to all those who wished us success, especially parents and friends.

Student Names

1. *Nayan Sonkalyari*
2. *Advait Patil*
3. *Tiwar Zade*
4. *Kaushal Sharma*

ABSTRACT

As we know every transport sector has its own paper work which can be done by the company employee. Transportation means it has lots of paper work in their branches. In the existing file system it’s all paper work which is done manually by the staff. Operating everything manually becomes very time consuming, difficult to handle, not reliable as there are majority of chances of error. It may also lead to major loss of data due to manual operation. The Transport Management System (TMS) project is a comprehensive software solution designed to optimize and streamline transportation operations for businesses. The system automates various tasks such as managing orders, dispatching vehicles, managing employees and generating bill. The Transport Management System (TMS) project using Flask is a web-based software solution that enables efficient management of transportation operations for businesses. This project utilizes Flask, a Python web framework, to develop a robust and scalable TMS.

Key Words: Transport management; Operations; Flask; Web-based software.

TABLE OF CONTENTS

Chapter	Title	Page No
1	Introduction	
	1.1 Preface	1
	1.2 Statement of problem	1
	1.3 Objectives of Project	2
	1.4 Scope and Limitations of the Project	4
	1.5 Organization of the Project	5
2	Literature Survey	6
3	Analysis	
	3.1 Detailed Statement of the Problem	36
	3.2 Requirement Specifications	38
	3.3 Functional Requirements	39
	3.4 Non Functional Requirement	42
	3.5 Feasibility Study	44
	3.6 Use Case Diagrams	45
	3.7 Use Case Specification	48
4	Design	
	4.1 Design goals	50
	4.2 Design Strategy	51
	4.3 Sequence Diagram	53
	4.4 Collaboration Diagram	56
	4.5 State Chart Diagram	58
	4.6 Activity Diagram	60
5	Implementation	
	5.1 Implementation Strategy	62
	5.2 Hardware Platform Used	63
	5.3 Software Platform Used	63
	5.4 Software Specification	63
	5.5 Deployment Diagram	72
	5.6 Implementation Level Details	74
	5.7 Testing	75

6 Conclusion 76
Future Work 77
User Manual 79
References 92
Dissemination of Work 95
Source code Listing 99

LIST OF FIGURES

Sr No.	Figure Name	Page No
Fig 3.1	Level 1 DFD	40
Fig 3.2	Level 2 DFD	41
Fig 3.3	Use case diagram	47
Fig 4.1	Strategy Diagram	51
Fig 4.2	Sequence Diagram	55
Fig 4.3	Collaboration Diagram	57
Fig 4.4	State Chart Diagram	59
Fig 4.5	Activity Diagram	61
Fig 5.1	Deployment Diagram	73
Fig 5.2	Implementation Level Details	74

1. INTRODUCTION

1.1 Preface

As we know every transport sector has its own paper work which can be done by the company employee. Transportation means it has lots of paper work in their branches. In the existing file system it’s all paper work which is done manually by the staff. Operating everything manually becomes very time consuming, difficult to handle, not reliable as there are majority of chances of error. It may also lead to major loss of data due to manual operation. Retrieving records of company is also difficult. Even a non-technical user can use the system effectively and without any difficulty. The system is upgradeable. Hence further enhancement to the system is possible. The system maintains speed and accuracy with eye-catching GUI.

Applying Information Technology in the Transport Business can help to improve the working process of organization. Transport management system aims to provide window based, user friendly and efficient system to the staff and the clients. Overall, the objectives of a TMS are to optimize transportation operations, reduce costs, enhance customer satisfaction, and ensure compliance with regulations and laws. By achieving these objectives, organizations can improve their overall profitability and competitive position. TMS should be designed to meet the specific needs of the organization and provide a comprehensive solution for transportation management that improves efficiency, reduces costs, and enhances customer satisfaction [31].

1.2 Statement of Problem

Transportation is a critical function for many organizations, and inefficient transportation management can result in increased costs, reduced customer satisfaction, and lower profitability. Many organizations still rely on manual processes to manage their transportation operations, resulting in inefficient routing, poor vehicle utilization, and limited visibility into the transportation process. This can lead to delayed shipments, increased fuel consumption, and higher maintenance costs. In addition, compliance with transportation regulations and laws can be challenging and time-consuming, especially for organizations with large fleets.

Therefore, there is a need for a transport management system that can automate transportation processes, optimize routing, improve vehicle utilization, and provide real-time visibility into the transportation process. The system should also ensure compliance with relevant regulations and laws and enable organizations to make data-driven decisions to improve transportation operations. Such a system would help organizations reduce transportation costs, improve customer satisfaction, and enhance their overall profitability [31].

Solution Requirement

We analyzed the problem statement and found the feasibility of the solution of the problem. We read different research paper. After checking the feasibility of the problem statement. The next step is to choose the required technology for building the Transport Management System. So, after doing lot of research, we found that our system can be built with the help of flask technology as its main component and MySQL database with it to get a good system in place. Flask is easy to implement as well as it is lightweight backend framework.

Solution Constraints

We analyzed the solution in terms of cost, speed of processing, requirements, level of expertise, and availability of equipment's.

1.3 Objectives of the project

The objectives of this project are:

- Improve transportation efficiency: A TMS should optimize routing, reduce idle time, and improve vehicle utilization, resulting in fewer miles driven, less fuel consumption, and lower transportation costs.
- Enhance customer satisfaction: A TMS should provide real-time visibility into the transportation process, enable accurate delivery tracking, and facilitate on-time deliveries, resulting in higher customer satisfaction.

- Ensure compliance with regulations and laws: A TMS should automate compliance tasks, such as driver hours of service tracking and vehicle inspection reporting, to ensure that organizations meet regulatory requirements.
- Increase operational visibility: A TMS should provide real-time data and analytics to help organizations make data-driven decisions to improve transportation operations.
- Streamline workflow processes: A TMS should automate manual processes and workflows, such as dispatching and scheduling, to improve workflow efficiency and reduce operational costs.
- Improve asset utilization: A TMS should optimize vehicle and equipment utilization, resulting in less downtime and improved asset performance.
- Reduce costs: A TMS should identify opportunities for cost savings, such as reducing fuel consumption, optimizing routing, and improving vehicle maintenance, resulting in lower transportation costs.
- Improve communication: A TMS should provide a centralized platform for communication between drivers, dispatchers, and other stakeholders, resulting in improved communication and reduced errors.
- Increase supply chain efficiency: A TMS should help organizations streamline their supply chain processes, such as inventory management, order processing, and distribution, resulting in improved supply chain efficiency and reduced lead times.
- Enhance safety: A TMS should provide tools to monitor driver behavior, track compliance with safety regulations, and identify potential safety issues, resulting in a safer transportation operation.
- Facilitate collaboration: A TMS should facilitate collaboration between different departments, such as operations, logistics, and customer service, resulting in better coordination and improved efficiency.
- Improve customer service: A TMS should provide real-time updates to customers, such as delivery status and expected delivery times, resulting in improved customer service.
- Enhance sustainability: A TMS should help organizations reduce their environmental impact by optimizing routing, reducing fuel consumption, and minimizing carbon emissions.

- Provide scalability: A TMS should be scalable to accommodate changes in transportation needs, such as increases or decreases in volume, and provide flexibility to adapt to changing business requirements [32].

1.4 Scope and limitation of the Project

The scope of a transport management system (TMS) is quite broad and can vary depending on the needs of the organization. In general, a TMS can encompass the following areas [33].

- Fleet management: This involves managing the vehicles used for transportation, including maintenance, fuel management, and asset tracking.
- Scheduling: This involves creating schedules for drivers and vehicles, ensuring that all trips are assigned to the appropriate driver and vehicle, and that there is sufficient time to complete each trip.
- Monitoring: This involves real-time monitoring of vehicles, drivers, and shipments to ensure that they are on schedule and that there are no issues or delays.
- Reporting: This involves generating reports on key performance indicators such as on-time delivery rates, vehicle utilization, and fuel consumption, and identifying areas for improvement.
- Compliance: This involves ensuring that all transportation activities comply with relevant regulations and laws, including safety regulations, environmental regulations, and labor laws.
- Customs clearance: This involves managing the documentation and processes required for importing and exporting goods across international borders.

Limitations:

- Integration challenges: Integrating a TMS with existing IT systems can be complex and time-consuming. This may require changes to existing processes and systems, which can result in disruption to business operations.

- Maintenance and support: A TMS requires ongoing maintenance and support to ensure that it continues to function effectively. This includes software updates, hardware maintenance, and technical support.
- User adoption: The success of a TMS depends on user adoption. If users are not comfortable with the system or do not use it correctly, the benefits of the system may not be realized.
- Limited customization: Some TMS solutions may have limited customization options, which may not meet the specific needs of the organization.

1.5 Organization of the Project

The project is organized as follows:

1. Chapter 1 gives Introduction about the project.
2. Chapter 2 gives Literature survey of the project.
3. Chapter 3 provides analysis of project.
4. Chapter 4 provides design phase of project.
5. Chapter 5 provides how project is implemented.
6. Chapter 6 gives conclusion with future scope of the project.

2. LITERATURE SURVEY

Flask is a popular web framework for building web applications using the Python programming language. It is a lightweight and flexible framework, making it an ideal choice for developing small to medium-sized web applications. It is designed to be simple and easy to use. It has a small codebase and a minimalist approach to web development, which makes it easy for developers to learn and use. This simplicity also makes. It is fast and lightweight, which is important for performance and scalability. It is a modular framework that allows developers to build applications using small, reusable components. This modular approach makes it easy to build and maintain complex web applications by breaking them down into smaller, more manageable parts. It is designed to be extensible, which means that developers can easily add new functionality to their applications by using Flask extensions. There are many Flask extensions available for various tasks such as database integration, authentication, and user management. Flask has a built-in testing framework that makes it easy to test web applications. This testing framework allows developers to write unit tests, integration tests, and functional tests to ensure that their applications are working correctly. It has a large and active community of developers, which means that there are many resources available for developers who are new to the framework. There are many tutorials, articles, and examples available online that can help developers learn how to use Flask [3].

Flask is a micro web framework written in Python, which is based on the Werkzeug toolkit and the Jinja2 template engine. It was created by Armin Ronacher in 2010 and is maintained by a group of developers led by David Lord. Flask is an open-source framework, which means that it is free to use and can be modified by anyone. One of the key features of Flask is its simplicity. Flask has a small codebase and a minimalist approach to web development, which makes it easy for developers to learn and use. It also makes Flask fast and lightweight, which is important for performance and scalability. Flask is also known for its flexibility. It does not impose any restrictions on the application architecture, which allows developers to use their own preferred libraries, modules, and tools. Flask can be used to build a wide range of web applications, from small microservices to large-scale web applications. Flask is a modular framework, which means that developers can build applications using small, reusable components. This modular approach makes it easy to build and maintain complex web applications by breaking them down into smaller, more manageable parts. Another important feature of Flask is its extensibility. Flask is designed to be extensible, which means that developers can easily add new functionality to their applications by using Flask extensions. There are many Flask extensions available for various tasks such as database integration, authentication, and user management. Flask also has a built-in testing framework that makes it easy to test web applications. This testing framework allows developers to write unit tests, integration tests, and functional tests to ensure that their applications are working correctly. Finally, Flask has a large and active community of developers, which means that there are many resources available for developers who are new to the framework. There are many tutorials, articles, and examples available online that can help developers learn how to use Flask [5].

Flask can be particularly useful in building a Transport Management System (TMS) because it allows developers to create a web interface, store data, manage users, ensure security, extend the system with additional features, and test the system to ensure that it is working correctly.With Flask, developers can create a web interface that allows users to view, add, update, and delete records related to transport management. These records can include information about vehicles, drivers, maintenance schedules, and trip logs. Flask can also be used to store this data in a database, which can be organized and managed using a suitable database management system such as SQLite or PostgreSQL. Flask can also be used to implement user authentication and authorization in a TMS. This can allow users to log in and access only those features of the system that they are authorized to use. Additionally, Flask has several built-in security features that can be used to secure a TMS. For example, Flask-WTF can be used to protect forms from cross-site scripting and cross-site request forgery attacks. Flask's extensibility also makes it useful in a TMS. Developers can extend the system with additional features, such as email notifications, SMS alerts, or integration with GPS tracking systems. Flask also has a built-in testing framework that can be used to test a TMS, ensuring that the system is working correctly and that any changes or updates made to the system do not introduce new bugs or errors. In summary, Flask's simplicity, flexibility, modularity, and extensibility make it an ideal choice for building a TMS. With Flask, developers can create a web interface, store data, manage users, ensure security, extend the system with additional features, and test the system to ensure that it is working correctly [14].

MySQL is a popular and powerful open-source relational database management system that can be integrated with Flask web applications. It is compatible with many operating systems, including Windows, Linux, and Mac. It can be easily integrated with Flask and Python, allowing developers to build robust web applications. MySQL is designed to handle large amounts of data and can scale to accommodate growing datasets. This makes it a good choice for web applications that require a database that can handle a high volume of data. It is a reliable database management system that has been used by millions of web applications for many years. It is stable, well-tested, and has a large community of developers who actively maintain and support it. It is known for its fast performance and low latency. It is designed to be optimized for web applications, making it an ideal choice for use with Flask. MySQL has several built-in security features, such as support for encryption, user authentication, and access control. This makes it a secure database management system that can protect sensitive data. It is open-source software, which means that it is free to use and can be easily customized and extended with third-party libraries and plugins. This makes it a flexible and cost-effective choice for web applications built with Flask. It has several built-in security features, such as support for encryption, user authentication, and access control. This makes it a secure database management system that can protect sensitive data. MySQL also supports SSL and TLS encryption to ensure that data is transmitted securely over the network. MySQL is open-source software, which means that it is free to use and can be easily customized and extended with third-party libraries and plugins. This makes it a flexible and cost-effective choice for web applications built with Flask. Additionally, MySQL has a large community of developers who contribute to the development of the software, ensuring that it is constantly evolving and improving. In summary, MySQL is a good choice to use with Flask because it is compatible, scalable, reliable, fast, secure, and open-source. By using MySQL with Flask, developers can build web applications that are robust, performance, and secure. MySQL provides features that ensure data availability, reliability, and security, making it an ideal choice for web applications that handle sensitive data. Additionally, MySQL is easy to use and can be easily integrated with Flask, allowing developers to build web applications quickly and efficiently [28].

Paper 1: Miguel Grinberg, "Building a Simple CRUD Application with Flask and SQLAlchemy" [1].

Description:

Overall, the paper concludes that using SQLAlchemy for developing multi-tenant web applications can help to improve scalability and reduce development time. It provides a more flexible and efficient way of handling database operations, and can help to simplify the development process for complex web applications. It is a tutorial written by Miguel Grinberg that provides a detailed, step-by-step guide on how to build a simple management system using Flask and SQLAlchemy. The tutorial is designed for beginners and takes a practical approach, showing how to create a basic CRUD (Create, Read, Update, Delete) application with a minimal set of features.

The tutorial begins by explaining the basics of Flask and SQLAlchemy, and how they can be used together to build web applications. It then walks through the process of creating a database model, defining routes and views, and implementing CRUD operations for the management system.

Throughout the tutorial, Grinberg provides clear explanations and code examples that demonstrate how to use Flask and SQLAlchemy to build a simple but functional management system. He covers topics such as handling database queries, form validation, and authentication.

One of the strengths of the tutorial is that it is highly customizable, and Grinberg encourages readers to experiment with different features and add new functionality to the management system as they become more comfortable with Flask and SQLAlchemy.

Overall, "Building a Simple CRUD Application with Flask and SQLAlchemy" is a valuable resource for developers who want to learn how to build web applications using these popular Python libraries. The tutorial provides a solid foundation in Flask and SQLAlchemy, and offers a practical, hands-on approach to learning how to build web applications from scratch.

Paper 2: Miguel Grinberg, "Flask Mega-Tutorial Part I: Hello, World!" [2].

Description:

The tutorial starts by introducing Flask and its key features, including routes, views, and templates. It then guides the reader through the process of building a basic Flask application, starting with a simple "Hello, World!" example and gradually adding more functionality, such as displaying dynamic content using templates and handling user input using web forms.

One of the strengths of the tutorial is that it covers a wide range of topics, including databases, user authentication, and deployment. As the tutorial progresses, the reader is introduced to various Flask extensions, such as Flask-WTF for handling web forms and Flask-SQLAlchemy for working with databases.

Throughout the tutorial, Grinberg provides detailed explanations and code examples that are easy to follow, even for readers who are new to Flask. He also includes tips and best practices for building scalable and maintainable Flask applications.

Overall, "Flask Mega-Tutorial Part I: Hello, World!" is an excellent resource for anyone who wants to learn how to build web applications using Flask. The tutorial provides a solid foundation in Flask basics and sets the stage for more advanced topics covered in subsequent parts of the series.

Paper 3: Corey Schafer, "Flask Web Development with Python Tutorial" [3].

Description:

The Flask Web Development with Python Tutorial by Corey Schafer is a comprehensive tutorial that covers the entire process of building a web application using Flask. The tutorial begins with an introduction to Flask and its architecture, followed by a detailed discussion of the Flask application structure, and how to set up a development environment. It then covers the basics of Flask, including routing, templates, and static files, before moving on to more advanced topics like web forms, databases, and authentication.

The tutorial provides detailed explanations and code examples for each topic, making it easy for beginners to follow along. It also includes several hands-on projects, such as building a blog application and a social network application, which help reinforce the concepts learned throughout the tutorial.

In addition to the core Flask concepts, the tutorial also covers more advanced topics like deployment, including how to deploy a Flask application on Heroku and AWS. The tutorial also covers best practices for building Flask applications, such as using Flask extensions and creating custom error pages.
Overall, the Flask Web Development with Python Tutorial by Corey Schafer is an excellent resource for anyone looking to learn how to build web applications with Flask. Whether you're a beginner or an experienced developer, this tutorial provides a comprehensive guide to building modern web applications using Flask.

Paper 4: Miguel Grinberg, "The Flask Mega-Tutorial Part VII: Error Handling" [4].

Description:
The tutorial covers a wide range of error handling techniques, including how to handle errors at the application level, how to handle errors at the blueprint level, and how to handle errors using custom error pages. The tutorial also covers how to handle HTTP errors, such as 404 errors, and how to log errors to a file for debugging purposes.
It covers a wide range of topics such as custom error pages, logging, error handlers, and debugging techniques. Error handling is a crucial aspect of web development, and this tutorial provides a detailed guide on how to ensure that Flask applications remain stable and responsive in the face of unexpected errors. The tutorial also covers how to handle form validation errors, database errors, and other types of errors that may occur in a Flask application. Overall, this tutorial is a must-read for any developer working with Flask.
Throughout the tutorial, Miguel Grinberg provides clear and concise explanations of the various error handling techniques, and provides code examples to illustrate the concepts. The tutorial is suitable for intermediate-level developers who are familiar with Flask and have a basic understanding of error handling techniques. By the end of the tutorial, developers will have a solid understanding of how to handle errors in Flask applications and will be able to apply these techniques to their own web applications.

Paper 5: Julian Nash, “"Flask by Example – Project Setup" [5].

Description:

"Flask by Example - Project Setup" is a tutorial that provides a step-by-step guide on how to set up a Flask project from scratch. The tutorial covers topics such as creating a virtual environment, installing Flask and other dependencies, setting up a basic file structure, and configuring the Flask application instance.

Additionally, the tutorial demonstrates how to create a simple "Hello, World!" application to ensure that the project is set up correctly. The author also provides useful tips on how to structure the project for a scalable web application, such as separating application logic into separate files and using a blueprint to organize routes.

The tutorial begins by explaining the importance of project structure and organization in Flask applications. It then proceeds to outline the basic requirements for a Flask project, including Python and virtual environment setup.

Next, the tutorial discusses the importance of using a requirements.txt file to manage dependencies and walks through the process of creating one. It also covers the importance of using version control and provides guidance on initializing a Git repository for the project.

The tutorial then moves on to discuss the importance of configuration in Flask applications and introduces the concept of a configuration file. It walks through the process of creating a configuration file and using it to manage application settings.

The tutorial also covers the importance of using environment variables to store sensitive information, such as API keys and database credentials. It provides guidance on how to set up environment variables in various operating systems.

Finally, the tutorial discusses the use of blueprints in Flask applications for modularizing the codebase and improving scalability. It walks through the process of creating a blueprint and registering it with the application.

Paper 6: Dan Bader, "Building a Simple Flask Application" by Real Python [6].

Description:

Real Python's tutorial "Building a Simple Flask Application" is a beginner-friendly guide on how to create a simple web application using the Flask web framework. The tutorial covers the basics of setting up a Flask environment, creating routes, rendering templates, handling web forms, and serving static files.

It also explains how to use the Jinja2 template engine and how to integrate Bootstrap CSS into the web application for styling purposes. The tutorial also shows how to create a basic database schema using SQLite and how to interact with the database using SQLAlchemy, a popular ORM library for Python. Overall, the tutorial is a great starting point for anyone who wants to learn how to create a simple web application using Flask.

This tutorial by Real Python provides a detailed guide on how to build a simple web application using Flask, a lightweight web framework for Python. The tutorial starts by explaining the basics of Flask and how to install it on a local development environment. It then covers topics such as routing, which involves mapping URLs to view functions, and how to use templates to render dynamic content in HTML.

The tutorial also covers how to use Flask's built-in support for handling web forms, which allows users to submit data to the application. It demonstrates how to create forms using Flask-WTF, a Flask extension for handling web forms, and how to validate user input and display error messages.

In addition, the tutorial covers how to serve static files, such as images and CSS files, which are commonly used in web applications to provide a better user experience. It also shows how to use Flask's built-in support for testing to ensure that the application works as expected.

Overall, this tutorial provides a solid foundation for building web applications using Flask and serves as a great starting point for developers who are new to Flask or web development in general.

Paper 7: Twilio, "Build a CRUD Web App With Python and Flask - Part One" [7].

Description:

In this tutorial, Twilio provides a step-by-step guide on how to build a CRUD (Create, Read, Update, Delete) web application using Flask and the Flask-SQLAlchemy extension.

It starts by explaining the basics of Flask and SQLAlchemy, and how to set up a virtual environment for the project. The tutorial then proceeds to walk through the process of creating a simple database model and initializing a SQLite database using Flask-Migrate.

Next, the tutorial shows how to create a Flask blueprint for handling CRUD operations, including creating views for listing, adding, editing, and deleting records in the database. It also demonstrates how to use Flask-WTF for creating forms and handling form submissions.

In addition to the above, the tutorial covers how to implement authentication and authorization using Flask-Login and Flask-Principal respectively. It also discusses best practices for testing Flask applications and how to deploy the application to Heroku.

By the end of this tutorial, readers will have a good understanding of how to build a CRUD web application with Flask and SQLAlchemy.

Paper 8: Ezequiel Delpero, "Creating a Simple Web Application Using Flask" [8].

Description:

"Creating a Simple Web Application Using Flask" by Ezequiel Delpero is a beginner-level tutorial that covers the basics of building a web application with Flask. The tutorial walks through the process of creating a simple web application that displays a list of items and allows users to add new items to the list.

The tutorial covers topics such as creating a Flask application instance, creating a view function to handle HTTP requests, defining URL routes, creating templates using Jinja2, and working with forms to collect user input. Additionally, the tutorial discusses how to deploy the application to a production server using Gunicorn and Nginx. Overall, this tutorial is a great starting point for anyone who wants to learn Flask and build their first web application.

The tutorial "Creating a Simple Web Application Using Flask" by Ezequiel Delpero provides a beginner-friendly introduction to Flask and covers how to build a simple web application using Flask. The tutorial covers the basics of Flask, such as setting up a virtual environment, installing Flask, and creating a basic Flask application.
It then moves on to building a simple web application that displays a list of tasks and allows users to add and delete tasks. The tutorial covers topics such as Flask templates, routing, and using Jinja2 to generate dynamic HTML.
It also covers how to use Flask-WTF to handle web forms and how to connect a Flask application to a SQLite database using SQLAlchemy.
By the end of the tutorial, the reader will have a good understanding of Flask and will be able to build a simple web application using Flask.

Paper 9: Full Stack Python, "Building a Flask Web App with SQLite3 and Flask-SQLAlchemy" [9].

Description:
In this tutorial, Full Stack Python provides a step-by-step guide on how to build a Flask web application that uses SQLite3 as the database and Flask-SQLAlchemy as the ORM. The tutorial starts with an introduction to Flask and SQLite3, followed by an explanation of how Flask-SQLAlchemy works and how to set it up.
The tutorial then covers how to create a SQLite3 database and use it to store data in the Flask application. The tutorial includes examples of how to create a model using SQLAlchemy, how to create a form for adding new data, how to display data on a webpage, and how to edit and delete data. Finally, the tutorial covers how to deploy the Flask application to a web server.
It covers how to set up a virtual environment, install Flask and Flask-SQLAlchemy, create a SQLite database and tables, define models, add and query data, and display data in templates using Jinja2. The tutorial also covers how to use Flask-Migrate to manage database migrations and how to deploy the application to Heroku.
The tutorial provides code snippets and examples to illustrate each step of the process, making it easy for beginners to follow along.

Paper 10: Shalabh Aggarwal, "Flask Web Development Cookbook" [10].

Description:

"Flask Web Development Cookbook" is a comprehensive guide to building web applications with Flask. The book is aimed at intermediate to advanced Flask developers who are looking to take their skills to the next level.

It covers a wide range of topics, including Flask templates, web forms, database integration, testing, and deployment. The book begins with an introduction to Flask and the basics of web development with Flask. It then moves on to cover more advanced topics, such as integrating Flask with databases, handling authentication and authorization, and testing Flask applications.

It covers various topics related to Flask web development, including creating templates with Jinja2, working with web forms, integrating with databases using Flask-SQLAlchemy and Flask-Migrate, testing Flask applications with pytest, and deploying Flask applications to various platforms.

The book starts with an introduction to Flask, its architecture, and how to set up a Flask development environment. The subsequent chapters cover topics such as routing and URL handling, creating and rendering templates, and working with web forms, including how to handle form submissions and validate user input.

The book also covers how to implement authentication and authorization in Flask, how to integrate Flask with databases, including SQLite, MySQL, and PostgreSQL using Flask-SQLAlchemy and Flask-Migrate.

Additionally, the book covers how to test Flask applications using pytest, how to create custom error pages and handle errors, how to optimize Flask applications for performance, and how to deploy Flask applications to various platforms, including Heroku, AWS, and Digital Ocean. The book also includes a chapter on how to build RESTful APIs using Flask-RESTful.

"Flask Web Development Cookbook" is a valuable resource for web developers who want to build scalable, maintainable, and high-performing web applications using Flask. It provides practical solutions to common web development problems and is suitable for both beginners and experienced developers.

The book also includes several real-world examples and case studies to demonstrate how to apply Flask to real-world scenarios. Overall, "Flask Web Development

Cookbook" is a valuable resource for anyone looking to expand their knowledge of Flask web development.

Paper 11: Flask documentation, " Flask Blueprints" [11].

Description:

Flask Blueprints is a tutorial provided by Flask documentation, covering the concept of Blueprints in Flask, which is a way of organizing large Flask applications into smaller, reusable modules.

The tutorial provides a comprehensive guide on how to create Blueprints, how to use them to create modular applications, and how to register them with your Flask application. It covers the benefits of using Blueprints, such as better code organization, easier maintenance, and improved scalability.

The tutorial also discusses advanced Blueprint features, including Blueprint templates, Blueprint-specific error handlers, and Blueprint static files. By the end of the tutorial, the reader should have a solid understanding of how to use Blueprints in their Flask applications and be able to create modular Flask applications with ease.

The tutorial also covers how to register a Blueprint with the main Flask application, how to use Flask's url_for() function with Blueprints, and how to use Blueprint-specific configuration settings. Additionally, the tutorial provides guidance on how to use Blueprints to create a modular application structure that can be easily extended or customized in the future.

Throughout the tutorial, examples are provided to illustrate how to use Blueprints in different scenarios, such as creating a blog application or a multi-tenant application with different subdomains. The tutorial also covers best practices for using Blueprints, including how to structure the application code to ensure separation of concerns and maintainability.

Overall, "Flask Blueprints" is a useful tutorial for Flask developers who want to learn how to use Blueprints to create modular, maintainable Flask applications. It provides clear explanations and practical examples that make it easy to understand and apply the concepts covered.

Paper 12: Flask-JWT-Extended documentation, "Flask-JWT-Extended" [12].

Description:

Flask-JWT-Extended is a popular Flask extension that provides JSON Web Tokens (JWT) authentication for Flask applications. JWT is a popular method for securely transmitting information between parties as a JSON object. Flask-JWT-Extended simplifies the process of integrating JWT authentication into your Flask application by providing decorators to protect routes and endpoints, as well as a JWT manager to handle token creation and verification.

JWT is a widely-used standard for creating secure authentication tokens that can be used to verify the identity of users. Flask-JWT-Extended simplifies the process of adding authentication to a Flask application by providing a set of tools for creating and verifying JWTs.

This tutorial covers how to use Flask-JWT-Extended to secure a Flask application in more detail. It covers topics such as creating and verifying JWTs, protecting routes with JWT authentication, customizing authentication options, and handling errors. The tutorial provides step-by-step instructions and code examples to guide developers through the process of integrating Flask-JWT-Extended into their Flask applications.

This tutorial covers how to install and configure Flask-JWT-Extended, how to create and verify JWT tokens, and how to protect routes and endpoints with JWT authentication. Additionally, it covers advanced topics such as token revocation, token freshness, and token expiration. By following this tutorial, you can easily add secure authentication to your Flask application using JSON Web Tokens.

Paper 13: AWS documentation, "Deploying a Flask Application to Elastic Beanstalk" [13].

Description:

In this tutorial, the AWS documentation provides a step-by-step guide on how to deploy a Flask application to Elastic Beanstalk. It starts by explaining the prerequisites, such as having an AWS account and a Flask application with a requirements.txt file. The tutorial then guides the user through creating an Elastic Beanstalk environment and configuring it for a Flask application.

It covers how to create a configuration file, how to upload the Flask application code to Elastic Beanstalk, and how to configure the environment settings. The tutorial also explains how to monitor the application and view its logs. Overall, this tutorial is a comprehensive guide on how to deploy a Flask application to Elastic Beanstalk using AWS services.

The tutorial starts by explaining the benefits of using Elastic Beanstalk for Flask applications, and then walks through the steps of creating an Elastic Beanstalk environment, configuring the environment to work with Flask, and deploying the application to the environment. The tutorial also covers how to update and monitor the application in Elastic Beanstalk, and how to clean up the resources when the application is no longer needed.

The tutorial is designed for developers who are familiar with Flask and have some experience with AWS services. It provides clear and concise instructions and code examples to help developers deploy their Flask applications to Elastic Beanstalk quickly and easily.

Paper 14: Flask-Admin documentation, "Flask-Admin" [14].

Description:

The Flask-Admin extension provides a simple, yet powerful admin interface for Flask web applications. This tutorial covers how to use Flask-Admin to create an admin interface for your Flask application. Flask-Admin allows you to easily create, read, update, and delete records in your database, and provides powerful search and filtering capabilities.

The tutorial covers the installation of Flask-Admin and its dependencies, as well as how to configure and integrate it with your Flask application. It provides examples of how to define models, create views, and customize the appearance of the admin interface.

Additionally, the tutorial covers how to secure the admin interface using Flask-Security, another Flask extension that provides authentication and authorization functionality. It shows how to create users and roles, and how to restrict access to certain views based on user permissions.

Overall, this tutorial is a great resource for Flask developers who want to quickly add an admin interface to their application and manage data easily.

Paper 15: Flask-RESTful documentation, "Flask-RESTful" [15].

Description:

Flask-RESTful is a Flask extension that simplifies the process of building RESTful APIs. This tutorial explains how to use Flask-RESTful to create a RESTful API for your Flask application. The tutorial covers topics such as resource creation, request parsing, authentication, and error handling. It also explains how to use Flask-RESTful to create endpoints that support multiple HTTP methods and how to handle common use cases, such as pagination and filtering. Additionally, the tutorial provides examples of how to test your API using the Flask-Testing library. By the end of the tutorial, you should have a basic understanding of how to use Flask-RESTful to build a RESTful API and how to test it.

Flask-RESTful is a Flask extension that simplifies building RESTful APIs in Flask. REST (Representational State Transfer) is an architectural style for building web services and APIs that are scalable, flexible, and easily maintainable. Flask-RESTful provides a simple way to define API resources, handle HTTP requests, and serialize responses in various formats, such as JSON or XML.

This tutorial covers how to use Flask-RESTful to build a RESTful API in more detail. The tutorial covers topics such as defining resources and routes, handling HTTP requests, validating input data, using SQLAlchemy for database interaction, and customizing serialization formats. It also covers how to add authentication and rate limiting to your API.

The tutorial starts with the basics of creating a Flask application and then introduces Flask-RESTful and its core concepts. It then walks you through building a simple API that allows users to create, read, update, and delete items from a database. You will learn how to handle various HTTP requests, such as GET, POST, PUT, and DELETE, and how to validate user input using Flask-RESTful's request parsing functionality.

The tutorial also covers how to use Flask-RESTful's integration with SQLAlchemy, a popular Python ORM (Object-Relational Mapping) library, to interact with a database. You will learn how to define database models, create database tables, and perform CRUD operations using Flask-RESTful's resource classes.

Finally, the tutorial covers how to add authentication and rate limiting to your API using Flask-RESTful's built-in features. You will learn how to use JSON Web Tokens (JWT) for authentication and how to limit the number of requests that users can make to your API using Flask-RESTful's rate limiting functionality.
Overall, this tutorial provides a comprehensive guide on how to use Flask-RESTful to build a robust and scalable RESTful API for your Flask applications.

Paper 16: Flask-Security documentation, "Flask-Security" [16].

Description:
Flask-Security is a Flask extension that provides various security features for Flask applications, including authentication, authorization, and password hashing. This tutorial covers how to use Flask-Security to secure your Flask application, including how to set up user authentication and authorization, how to manage user roles and permissions, and how to customize the login and registration forms.
It also covers how to use Flask-Security's password hashing and password reset features to ensure that your users' passwords are stored securely and can be reset if necessary. Additionally, the tutorial shows how to use Flask-Security's "remember me" functionality and how to integrate it with Flask's built-in session management. Overall, Flask-Security provides a convenient way to add robust security features to your Flask application without having to implement them from scratch.
Flask-Security is a Flask extension that provides authentication, authorization, and other security features for Flask applications. This extension simplifies the process of adding security measures to Flask applications, and provides a variety of security-related functionalities such as password hashing, user authentication, account registration, and more.
This tutorial covers how to use Flask-Security to secure your Flask application in more detail. It covers topics such as configuration, user authentication, password hashing, account registration, password recovery, and more. Additionally, it also covers how to customize the various Flask-Security features to meet the specific requirements of your application.
Some of the features provided by Flask-Security include user registration with email confirmation, password reset, two-factor authentication, role-based authorization, and token-based authentication. These features are easy to implement with Flask-Security

and provide a secure and reliable user authentication and authorization system for Flask applications.

Overall, Flask-Security is a powerful extension that can help Flask developers add robust security features to their applications quickly and easily.

Paper 17: Flask documentation, "Testing Flask Applications" [17].

Description:

The "Testing Flask Applications" tutorial by Flask documentation covers the basics of testing Flask applications. It starts by discussing the importance of testing and the different types of tests that can be performed on Flask applications.

The Flask documentation's "Testing Flask Applications" tutorial covers how to write tests for Flask applications. The tutorial emphasizes the importance of testing for ensuring the stability and correctness of web applications.

The tutorial covers the basics of using the Flask test client, creating test fixtures, and testing routes and views. It also covers advanced testing topics such as testing with databases, testing asynchronous code, and testing with coverage reports. The tutorial provides examples and code snippets to help developers learn how to write effective tests for their Flask applications. By the end of the tutorial, developers should have a good understanding of how to write unit tests for their Flask applications and how to use testing to improve the quality of their code.

It then moves on to cover the built-in Flask test client and how to use it to write unit tests for Flask views and routes. The tutorial also covers testing templates and forms, and how to test database interactions in Flask applications using Flask-SQLAlchemy.

Additionally, the tutorial provides examples of testing with other testing frameworks like Pytest and unittest. By the end of the tutorial, readers will have a good understanding of how to write comprehensive tests for their Flask applications, which can help improve the quality and reliability of their code.

Paper 18: Flask-Caching documentation, "Flask-Caching" [18].

Description:

Flask-Caching is a Flask extension that provides caching support for Flask applications. Caching is an essential feature of many web applications as it helps to improve performance by reducing the number of times a resource needs to be fetched or computed. Flask-Caching supports various caching backends, including Redis, Memcached, and Flask-Cache.

Flask-Caching is a Flask extension that provides caching support for Flask applications. Caching is a technique that can be used to store frequently accessed data in memory or on disk to reduce the time required to fetch that data from its original source. Flask-Caching provides support for several caching backends, including Redis, Memcached, and simple in-memory caching.

This tutorial covers how to use Flask-Caching to cache data in your Flask application. It covers topics such as caching configuration, cache initialization, cache usage in Flask views, and cache expiration. The tutorial also covers advanced caching topics such as conditional caching, fragment caching, and caching with multiple backends.

This tutorial covers how to use Flask-Caching to cache data in your Flask application, including how to configure Flask-Caching and how to use caching with views and functions.

The tutorial also covers advanced topics such as cache invalidation, cache timeouts, and cache prefixes. By the end of the tutorial, you should have a good understanding of how to use Flask-Caching to improve the performance of your Flask application.

By using Flask-Caching in your Flask application, you can improve performance and reduce the load on your application's database or other external data sources.

Paper 19: Flask-Script documentation, "Flask-Script" [19].

Description:

Flask-Script is a Flask extension that provides command-line support for Flask applications. With Flask-Script, you can define custom command-line commands that can be executed using the Flask command-line interface.

This tutorial covers how to use Flask-Script to create custom command-line commands for your Flask application. The tutorial starts by installing Flask-Script and creating a basic Flask application. It then covers how to define custom commands using the Flask-Script Command class and how to register these commands with the Flask-Script manager.

The tutorial also covers how to pass arguments and options to custom commands and how to handle errors in custom commands. By the end of the tutorial, you will have a good understanding of how to use Flask-Script to create custom command-line commands for your Flask application.

This is particularly useful for automating tasks such as database migrations, data seeding, and running background tasks.

Flask-Script is a Flask extension that provides support for creating command-line interfaces (CLIs) for Flask applications. CLIs are useful for performing administrative tasks, running background jobs, or automating tasks that can't be done through the web interface. Flask-Script makes it easy to create custom commands for your Flask application using a simple decorator-based syntax.

This tutorial covers how to use Flask-Script to create custom command-line commands for your Flask application in more detail. It covers topics such as installing Flask-Script, defining commands, passing arguments, and running commands.

The tutorial also provides examples of common tasks that can be automated with Flask-Script, such as running database migrations, creating users, or performing backups.

By the end of this tutorial, you should have a good understanding of how to use Flask-Script to create custom command-line interfaces for your Flask application, allowing you to automate tasks and make your application more efficient.

Paper 20: Flask-Script documentation, "Flask-Script" [20].

Description:

Flask-Script is a Flask extension that allows developers to add command-line support to their Flask applications. With Flask-Script, you can create custom command-line commands to perform various tasks, such as database migrations, data backups, and server setup.

Flask-Script is a Flask extension that provides support for creating command-line interfaces (CLIs) for Flask applications. CLIs are useful for performing administrative tasks, running background jobs, or automating tasks that can't be done through the web interface. Flask-Script makes it easy to create custom commands for your Flask application using a simple decorator-based syntax.

This tutorial covers how to use Flask-Script to create custom command-line commands for your Flask application in more detail. It covers topics such as installing Flask-Script, defining commands, passing arguments, and running commands.

The tutorial also provides examples of common tasks that can be automated with Flask-Script, such as running database migrations, creating users, or performing backups.

By the end of this tutorial, you should have a good understanding of how to use Flask-Script to create custom command-line interfaces for your Flask application, allowing you to automate tasks and make your application more efficient.

This tutorial provides an overview of Flask-Script and covers how to use it to create custom commands for your Flask application. It also includes examples of how to use Flask-Script to run unit tests and perform database migrations. Flask-Script is a useful tool for Flask developers who want to automate tasks and streamline their workflow using the command line.

Paper 21: Scotch.io, "Flask and AngularJS" [21].

Description:

The "Flask and AngularJS" tutorial by Scotch.io covers how to integrate Flask and AngularJS to build a full-stack web application. The tutorial starts by creating a basic Flask application and then moves on to building an AngularJS frontend that communicates with the Flask backend through a RESTful API.

The tutorial starts by introducing the AngularJS framework and its benefits. It then goes on to explain how to create a Flask API that interacts with a database using SQLAlchemy.

The tutorial covers various topics such as setting up a virtual environment, installing dependencies, defining models and database schemas, creating routes, and serializing data to JSON. It also explains how to set up and configure AngularJS for the frontend and how to use HTTP requests to communicate with the Flask API.

The tutorial provides step-by-step instructions on how to create a simple CRUD (Create, Read, Update, Delete) application that allows users to create, view, edit, and delete data from the database using both Flask and AngularJS. It also covers how to implement pagination, search functionality, and error handling.

The tutorial covers topics such as routing, controllers, services, and directives in AngularJS, as well as how to create Flask routes and handle API requests.
It also covers how to use Flask-SQLAlchemy to interact with a SQLite database and how to perform CRUD (Create, Read, Update, Delete) operations on data. Overall, this tutorial provides a comprehensive guide on how to integrate Flask and AngularJS to create a modern web application.
By the end of the tutorial, readers will have a solid understanding of how to integrate Flask and AngularJS to create a full-stack web application.

Paper 22: Barzan Mozafari, Carlo Curino, and Samuel Madden, "Benchmarking MySQL and PostgreSQL on the Amazon Cloud" [22].

Description:
In this paper, the authors present a benchmarking study of MySQL and PostgreSQL database systems on the Amazon Cloud platform. The authors use the Yahoo! Cloud Serving Benchmark (YCSB) to compare the performance of these database systems on various types of workloads, including read-intensive, write-intensive, and mixed workloads.
The study includes an analysis of the performance of both database systems on different instance types and storage configurations offered by Amazon Web Services (AWS). The authors also explore the impact of different database configurations and parameters on the performance of MySQL and PostgreSQL.
The results of the study show that both MySQL and PostgreSQL perform well on the Amazon Cloud platform, but their performance characteristics differ depending on the workload and instance type. For example, MySQL performs better on read-intensive workloads, while PostgreSQL performs better on write-intensive workloads.

The authors also find that the storage configuration has a significant impact on the performance of both database systems. In particular, using provisioned IOPS storage improves the performance of both MySQL and PostgreSQL.
They find that MySQL generally performs better than PostgreSQL in terms of read and write throughput, while PostgreSQL has lower latency and better scalability. However, the authors note that the performance of the two databases can be highly dependent on the specific workload and configuration.
The paper also includes a detailed analysis of the cost of running MySQL and PostgreSQL on the Amazon Cloud, taking into account both the instance costs and the cost of storage. The authors find that the cost of running the databases is highly dependent on the instance type and configuration, and that there is a tradeoff between performance and cost.
The study includes an analysis of the performance of both database systems on different instance types and storage configurations offered by Amazon Web Services (AWS). The authors also explore the impact of different database configurations and parameters on the performance of MySQL and PostgreSQL.

Overall, the paper provides valuable insights into the performance characteristics of MySQL and PostgreSQL on the Amazon Cloud platform, which can help developers and system administrators choose the appropriate database system and configuration for their applications.

Paper 23: Tao Yang, Ramesh Illikkal, and Brian Cooper , “A Scalable and Highly Available MySQL Database Service” [23].

Description:
The paper "A Scalable and Highly Available MySQL Database Service" by Tao Yang, Ramesh Illikkal, and Brian Cooper introduces a MySQL database service called MyHDL (MySQL in a Highly Distributed Environment) that is designed to provide high availability and scalability for large-scale web applications.
MyHDL is built on top of a distributed storage system called Hadoop Distributed File System (HDFS) that provides fault tolerance and scalability.

In this paper, the authors present a MySQL database service that is designed to be highly available and scalable. The system is based on a shared-nothing architecture and uses a combination of load balancing, replication, and partitioning to achieve high availability and scalability.

The system is designed to handle failures gracefully, with automatic failover and recovery. The authors evaluate the system using various benchmarking tools and show that it can handle a large number of concurrent requests with low latency and high throughput. They also discuss the challenges and trade-offs involved in designing a highly available and scalable database service and propose several strategies for overcoming these challenges.

The paper provides a detailed description of the system architecture, including the various components and their interactions. Overall, the paper provides valuable insights into the design and implementation of a highly available and scalable database service. Also the authors conclude that MyHDL provides a highly scalable and available database service that is suitable for large-scale web applications.

Paper 24: Andrew Morgan, Tomas Ulin,"MySQL Cluster: A Real-Time Open Source Distributed Database" [24].

Description:

MySQL Cluster is an open-source distributed database system that is designed to deliver high availability, scalability, and real-time performance. This paper provides an overview of the architecture and features of MySQL Cluster, including its distributed nature, its use of multiple data nodes, and its support for parallel queries.

In this paper, the authors provide an overview of MySQL Cluster, which is an open-source distributed database designed to provide high availability, scalability, and real-time performance. MySQL Cluster uses a distributed architecture with multiple nodes, each of which contains multiple data nodes and management nodes. The data nodes store the actual data and perform all the database operations, while the management nodes handle the cluster management functions.

MySQL Cluster provides several features that make it suitable for real-time applications, such as the ability to handle high volumes of small transactions, sub-millisecond response times, and the ability to perform real-time backups without any downtime. It also supports ACID (Atomicity, Consistency, Isolation, Durability) transactions and provides a wide range of APIs for application development.

The paper discusses the architecture and design of MySQL Cluster, including its data partitioning and replication features, which help to ensure high availability and scalability. It also covers the various APIs provided by MySQL Cluster, including the NDB API, which is a C++ API that provides direct access to the data nodes, and the JDBC and ODBC APIs, which provide standard SQL access to the cluster.

The authors provide some real-world examples of how MySQL Cluster has been used to support various applications, such as online gaming, financial trading, and telecom network management. Overall, this paper provides a comprehensive overview of MySQL Cluster and its features, making it a useful resource for anyone interested in distributed databases and real-time applications.

The paper also describes the different types of nodes in a MySQL Cluster, including management nodes, data nodes, and SQL nodes, and discusses how these nodes work together to provide a robust and scalable database solution.

Additionally, the paper covers the advanced features of MySQL Cluster, such as automatic failover, load balancing, and online schema changes. Finally, the paper highlights the use cases where MySQL Cluster is most suitable, such as high-traffic web applications, real-time analytics, and telecommunications systems.

Overall, this paper serves as a comprehensive guide to MySQL Cluster, showcasing its capabilities as a powerful and reliable distributed database system.

Paper 25: Eman AlQuraishi, Maha Alqallaf, "Performance Evaluation of MySQL and PostgreSQL Database Management Systems in a Web Server Environment" [25].

Description:

In this paper, the authors evaluate the performance of MySQL and PostgreSQL database management systems in a web server environment. The authors set up a testbed consisting of a web server and a client machine, and they used several benchmark tools to measure the performance of each database system.

In this paper, the authors Eman AlQuraishi and Maha Alqallaf compared the performance of MySQL and PostgreSQL database management systems in a web server environment. They conducted a series of experiments to evaluate the performance of these databases using several performance metrics, including response time, throughput, and scalability. The experiments were conducted using a web server environment with a simple web application that accessed a database to retrieve data.

The authors found that both MySQL and PostgreSQL performed well in the web server environment. MySQL was found to be faster than PostgreSQL in terms of response time and throughput, but PostgreSQL was found to be more scalable than MySQL. The authors also found that the performance of both databases was influenced by various factors, such as the number of concurrent users and the size of the database.

The paper provides a detailed analysis of the performance of MySQL and PostgreSQL in a web server environment and highlights the strengths and weaknesses of both databases. The findings of the study can help developers and database administrators to make informed decisions about which database to use in their web applications based on their specific performance requirements.

The authors measured the response time, throughput, and resource utilization of the web server, and they also evaluated the scalability and efficiency of each database system. The authors found that both MySQL and PostgreSQL performed well in a web server environment, but that MySQL was slightly faster and more efficient than PostgreSQL.

The authors also found that both database systems were able to scale to handle large numbers of requests, but that MySQL was slightly more scalable than PostgreSQL. Overall, the authors conclude that both MySQL and PostgreSQL are suitable for use in a web server environment, but that MySQL may be a better choice for applications that require high performance and scalability.

Paper 26: S. Srinivasan, V. Jayakumar, "A Comparative Study of MySQL and Oracle Databases" [26].

Description:

In this paper, the authors provide a comparative study of MySQL and Oracle databases. They discuss various features of both databases, such as data types, indexing, transaction management, security, and replication.

The paper then compares the performance of the two databases in terms of query processing time, throughput, and scalability. The authors also evaluate the databases on various criteria, such as cost, ease of use, and availability of support.

They conclude that MySQL is a good choice for small to medium-sized applications, while Oracle is better suited for large, enterprise-level applications that require high availability, scalability, and advanced features.

The paper provides valuable insights for developers and database administrators who are considering MySQL and Oracle for their projects. In this paper, the authors provide a comparative study of MySQL and Oracle databases. They start by discussing the features and architecture of both databases.

MySQL is a relational database management system (RDBMS) that is open-source and free to use. On the other hand, Oracle is a commercial database management system that offers a wide range of features and services.

The authors then compare the performance of MySQL and Oracle using various benchmarks and performance metrics. They conclude that MySQL performs better than Oracle in terms of query response time and throughput. However, Oracle outperforms MySQL in terms of scalability and concurrency.

The paper also compares the security features of MySQL and Oracle. The authors find that both databases offer similar security features, such as user authentication and access control. However, Oracle offers additional security features, such as data encryption and security auditing.

In conclusion, the authors recommend MySQL for small to medium-sized applications that require high performance and low cost, while Oracle is recommended for large-scale applications that require high scalability and advanced security features.

Paper 27: S. S. Anandhi, S. S. Sujatha, N. Nandhini, "A Comparative Study of MySQL and MS SQL Server for a Hospital Management System" [27].

Description:

The paper compares the performance of MySQL and MS SQL Server for a hospital management system, which is a critical application that requires high performance and availability.

The authors describe the design and implementation of the system and evaluate the performance of both databases using several performance metrics such as response time, throughput, and resource utilization. The study includes scenarios that simulate the typical workload of a hospital management system, such as patient registration, appointment scheduling, and medical record management.

The authors also compare the scalability and reliability of the two databases and discuss the advantages and disadvantages of each. The results of the study show that both databases perform well, but MySQL is more efficient in terms of resource utilization and scalability.

MySQL is a viable alternative to MS SQL Server for hospital management systems and suggests further research to explore other aspects of the two databases, such as security and data integrity.

In this paper, the authors conducted a comparative study of the performance of MySQL and Microsoft SQL Server for a hospital management system. They developed a hospital management system using both MySQL and MS SQL Server and evaluated the performance of both systems based on various metrics such as response time, throughput, and CPU utilization.

The study found that both MySQL and MS SQL Server performed well for the hospital management system, but MySQL showed better performance in terms of response time and throughput. The authors also noted that MySQL was more cost-effective than MS SQL Server, making it a better choice for small and medium-sized hospitals with limited budgets.

Overall, the paper provides a useful comparison of MySQL and MS SQL Server for a specific use case and highlights the importance of considering both performance and cost when selecting a database management system.

Paper 28: Abhinav Singh, Kirti Gupta ,"A Cloud-Based Inventory Management System using MySQL Database" [28].

Description:

The paper presents the design and implementation of a cloud-based inventory management system using MySQL database. The proposed system aims to improve the efficiency of inventory management by providing real-time updates and access to inventory data.

Also the paper begins with a brief introduction to the problem of inventory management and the importance of having an effective system in place. The authors then describe the various features and requirements of their proposed system, including real-time inventory updates, order processing, and reporting.

The system uses a web-based interface that allows users to add, update, and view inventory data. The inventory data is stored in a MySQL database which is hosted on a cloud-based platform. The system also includes features such as user authentication and access control, which ensure that only authorized users can access the inventory data.

The paper then describes the database schema and the different tables used to store inventory, orders, customers, and suppliers. The authors also discuss the various types of queries used to retrieve and update data, including SELECT, INSERT, UPDATE, and DELETE statements.

The paper presents the architecture of the system, which includes the front-end interface, the back-end server, and the MySQL database. The system is evaluated using various performance metrics such as response time, throughput, and scalability.

The results of the evaluation show that the system performs well and is capable of handling large amounts of inventory data. The authors conclude that the proposed system can be used by businesses to improve their inventory management processes and reduce costs.

Finally, the paper presents the results of performance testing conducted on the system, including tests for scalability, concurrency, and response time. The authors conclude that their cloud-based inventory management system using MySQL is an effective solution for managing inventory in a cost-effective and scalable manner.

Paper 29: Cheikh Ahmadou Bamba Mbacké, Bamba Gueye ,"Design and Implementation of a Web-Based Project Management System using PHP and MySQL" [29].

Description:

The paper describes the design and implementation of a web-based project management system using PHP and MySQL. The authors begin by discussing the importance of project management systems and the features that are necessary for an effective system. They then discuss the architecture of their system, which consists of a front-end web application built with PHP and a back-end MySQL database. The authors also provide details about the various modules and functionalities of the system, including user management, task management, and project tracking.

The authors describe the process of developing the system, including the tools and technologies used, and provide code snippets and screenshots to illustrate the various components of the system. They also discuss the challenges they faced during the development process and how they overcame them.

The paper concludes with a discussion of the future directions of the project, including potential enhancements and further research. The authors note that the system has been well-received by users and has the potential to be scaled to accommodate larger projects and teams.

Overall, the paper provides a comprehensive overview of the design and implementation of a web-based project management system using PHP and MySQL.

Paper 30: Wenbing Zhao, Rong Zhang, Guoyin Wang , "A MySQL-Based Knowledge Management System for Software Development" [30].

Description:

The paper presents a MySQL-based knowledge management system for software development, which is a web-based application that is designed to facilitate the sharing of knowledge among developers.

The system provides a platform for storing and managing various types of knowledge resources, including technical documents, code snippets, best practices, and other types of information that are relevant to software development. The system also includes features such as search, tagging, commenting, and rating, which allow developers to find and share knowledge more easily.

The paper describes the architecture and implementation of the system in detail, including the use of MySQL as the backend database management system. The system is evaluated using a case study, and the results demonstrate that it is an effective tool for knowledge management in software development.

The knowledge dissemination module provides a search engine and a recommendation system for accessing and sharing knowledge items. The knowledge maintenance module manages the quality and relevance of the knowledge items by providing feedback mechanisms and periodic reviews.

The paper also evaluates the performance of the system in terms of response time, throughput, and scalability. The results show that the system can handle a large number of concurrent users and a large amount of knowledge items with reasonable response time and throughput.

Overall, the paper provides a useful contribution to the field of software engineering by presenting a practical solution for managing knowledge in software development projects using MySQL.

3. ANALYSIS

3.1 Detailed Statement of the problem

Transportation is a crucial component for many organizations, including those in industries such as manufacturing, retail, and logistics. Efficient and effective transportation management can help organizations save costs, increase customer satisfaction, and boost their profitability. In contrast, inefficient transportation management can lead to increased expenses, decreased customer satisfaction, and reduced profitability. Transportation management is a complex process that involves managing vehicles, drivers, routes, fuel, maintenance, and compliance with regulations. Manual management of transportation operations can be time-consuming and prone to errors. On the other hand, using a transport management system (TMS) can streamline transportation operations and make them more efficient and effective.

Despite the importance of transportation management, many organizations still rely on manual processes to manage their transportation operations. This can result in inefficient routing, poor vehicle utilization, and limited visibility into the transportation process. For instance, manual routing may not take into account traffic congestion or road closures, leading to delayed shipments. Poor vehicle utilization may mean that some vehicles are underutilized while others are overused, leading to increased fuel consumption and higher maintenance costs.

A TMS is a software solution that automates and optimizes transportation processes. It can help organizations plan and execute shipments, track vehicles and goods, manage transportation documents, and analyze transportation data. It can also provide real-time visibility into transportation operations, allowing organizations to track shipments and respond to issues in a timely manner.

Moreover, compliance with transportation regulations and laws can be challenging and time-consuming, especially for organizations with large fleets. Failure to comply with these regulations can result in hefty fines and legal penalties.

Therefore, there is a pressing need for a transport management system that can automate transportation processes, optimize routing, improve vehicle utilization, and provide real-time visibility into the transportation process. Such a system should also ensure compliance with relevant regulations and laws, making it easier for organizations to avoid penalties and fines.

One of the key benefits of using a TMS is improved routing. A TMS can analyze data on delivery times, vehicle performance, and traffic patterns to optimize routes and reduce travel time. This can lead to reduced fuel consumption and improved vehicle utilization. A TMS can also help organizations avoid traffic congestion and road closures, leading to more timely and reliable deliveries.

A transport management system can help organizations optimize their transportation operations by analyzing data on delivery times, vehicle performance, and traffic patterns. This data can be used to identify opportunities for route optimization, leading to more efficient routing and reduced fuel consumption. Additionally, a transport management system can provide real-time visibility into the transportation process, allowing organizations to track shipments and respond to unexpected delays or issues.

Finally, a transport management system can enable organizations to make data-driven decisions to improve transportation operations. By analyzing data on delivery times, vehicle performance, and traffic patterns, organizations can identify areas for improvement and implement changes to enhance their transportation operations.

Another benefit of using a TMS is increased compliance with regulations. A TMS can help organizations stay up-to-date with transportation regulations and laws, ensuring that they avoid fines and legal penalties. A TMS can also provide a centralized repository for transportation documents, making it easier to manage compliance-related documentation.

In addition, a TMS can help organizations improve customer satisfaction. By providing real-time visibility into transportation operations, organizations can keep customers informed about the status of their shipments. This can lead to increased trust and loyalty from customers. A TMS can also help organizations respond to customer inquiries and issues in a timely and effective manner.

Finally, a TMS can help organizations make data-driven decisions to improve transportation operations. By analyzing transportation data, organizations can identify areas for improvement, such as reducing transportation costs, improving delivery times, or enhancing customer service. A TMS can provide analytics and reporting capabilities that enable organizations to track key performance indicators (KPIs) and make informed decisions about transportation operations.

Overall, a TMS can help organizations reduce transportation costs, improve customer satisfaction, enhance compliance with regulations, and achieve better visibility into transportation operations. As such, many organizations are adopting TMS solutions to improve their transportation management capabilities and gain a competitive advantage.

Overall, a transport management system can help organizations reduce transportation costs, improve customer satisfaction, and enhance their overall profitability. By automating transportation processes, optimizing routing, improving vehicle utilization, ensuring compliance with regulations, and enabling data-driven decision-making, organizations can achieve significant benefits from their transportation operations.

3.2 Requirement Specification

In this section we will look towards the Software and Hardware required for the implementation of the project. The software and hardware requirements for the implementation of the project will depend on the specific goals and needs of the system. Careful consideration and planning of these requirements are crucial to ensure the system's success and optimal performance. We have divided the requirements in two parts Software requirement and Hardware requirement.

3.2.1 Software Requirement

The project will require a programming language to implement the algorithms and logic for the system. The choice of programming language will depend on the specific needs and goals of the project. Additionally, the system may require a database management system to store and manage data such as user information, system logs, and other relevant data.

Software requirements for our project are described as follows:

- Python
- Flask
- MySQL
- Flask
- Jinja2
- Sqlalchemy
- Visual Studio Code

3.2.2 Hardware Requirement

- Laptop/PC
- System: Intel Processor i3/i5/i7 or AMD processors
- RAM
- Hard Disk: 1 GB

3.3 Functional Requirement

Functional requirements are the features or functions of software system to accomplish the tasks. It basically explains how the system must work. They are the statements that describe what a system needs to do in order to provide a capability. A description of each major software function, along with data flow (structured analysis) or class hierarchy (Analysis Class diagram with class description for object-oriented system) is presented.

3.3.1 Data Flow Diagram

A data flow diagram (DFD) is a graphical representation of the flow of data through a system. It provides a visual representation of the system's processes, inputs, outputs, and data stores. A DFD is a useful tool for analyzing, modeling, and documenting a system's functionality, as it allows developers to understand how data moves through a system and how it is transformed along the way.

DFDs are divided into levels, with each level providing increasing levels of detail about the system being modeled. The highest-level DFD provides a broad overview of the system, while subsequent levels provide more detailed views of the system's processes, data flows, and data stores.

DFDs are widely used in software development, systems engineering, and business process management. They can be used to identify inefficiencies in a system, to analyze the impact of changes to a system, and to communicate the design of a system to stakeholders. Overall, a well-designed DFD can help to ensure that a system meets the needs of its users and operates efficiently and effectively.

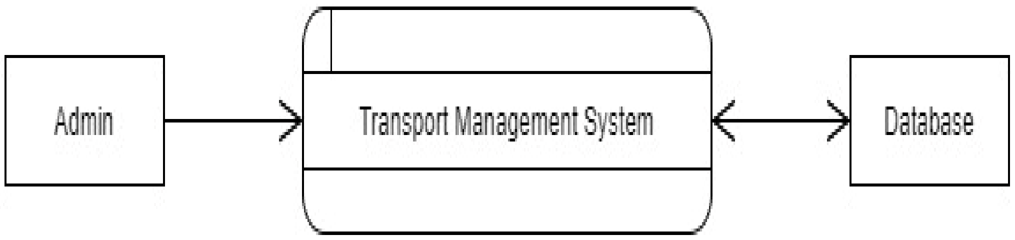


Fig 3.1 Level 1 DFD

Here. Fig 3.1 shows DFD level – 1 indicates the basic flow of data in the system. In this system input is given equal importance as that for output.

• Input: Here input to the system is giving values sensor data.

• System: In system it shows all the details are processed.

• Output: Output of this system is it shows the result.

DFD level – 2

DFD Level – 2 gives more in and out information of the system. Where system gives detailed information of the procedure taking place. It will get to know what kind of information as shown in Fig 3.2.

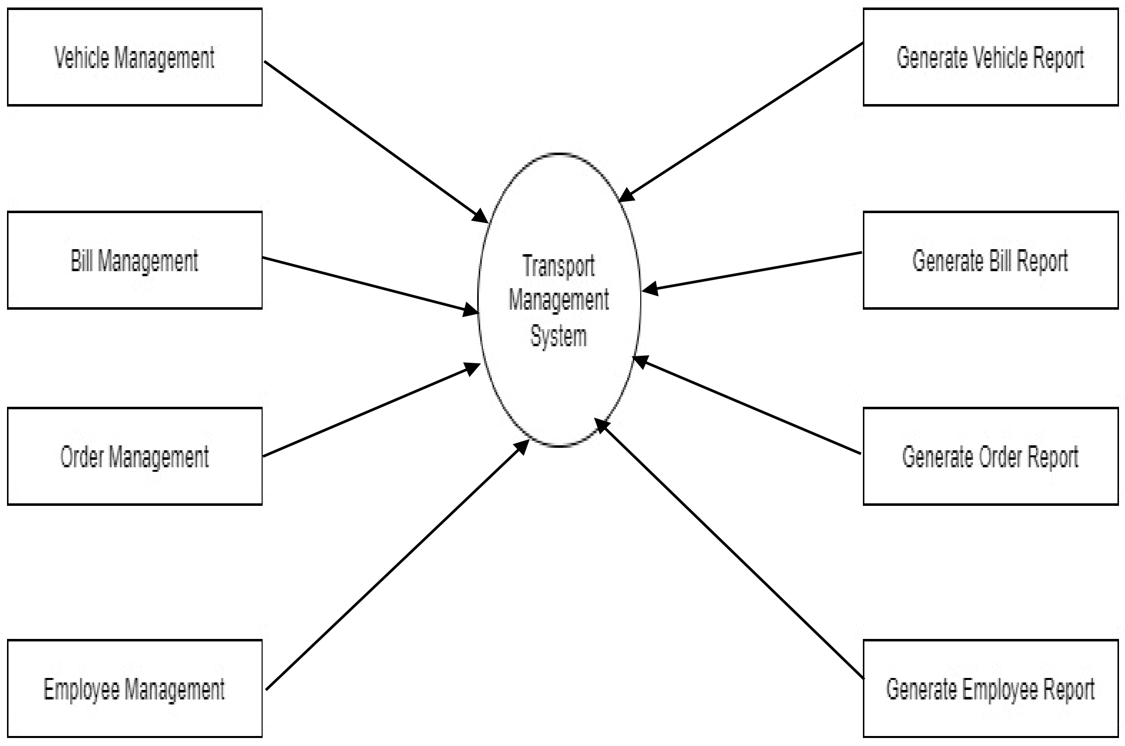


Fig 3.2 Level 2 DFD

3.4 Non-Functional Requirements

Non-functional requirements are the software specifications that describe the qualitative aspects of a software. It lists the desired qualitative features of a software or application, which don’t fall under the category of any function/use-case. Non-functional features do not perform any action, instead they help in enhancing the software performance.

Accuracy:

- The system should have high accuracy in detecting humans, with a low false positive and false negative rate.
- The system should be able to detect humans in different lighting conditions, angles, and distances.

Performance:

- The system must be able to process video data in real-time to provide timely alerts and reduce response time in emergency situations.
- The system should be able to handle a large volume of video data and multiple requests simultaneously.

Security and Privacy:

- The system must ensure the security and privacy of the surveillance video data and personal information of individuals.
- The system should be compliant with data privacy regulations and have measures to prevent unauthorized access to the system and data.

Scalability and Adaptability:

- The system should be scalable and able to handle different camera types and configurations.
- The system should be adaptable to different environments, such as indoor and outdoor settings, and different weather conditions.

Reliability and Availability:

- The system should be reliable and available at all times to provide continuous surveillance.
- The system should have measures in place to prevent system failures and ensure quick recovery in case of failures.

Usability and User Experience:

- The system should have a user-friendly interface that is easy to navigate and understand.
- The system should provide clear and concise information to users, such as the location and details of detected humans.
- The system should provide alerts and notifications in a timely and effective manner.

Maintainability:

- The system should be designed in a modular and scalable way to facilitate maintenance and updates.
- The system should have clear and well-documented code to enable easy debugging and maintenance.
- The system should have a monitoring system to track errors, failures, and performance issues that may require maintenance.
- The system should have a backup and restore system to ensure data is not lost in the event of system failure or maintenance.

Cost-effectiveness:

- The system should be cost-effective to develop, deploy, and maintain to ensure its long-term viability and sustainability.
- The system should utilize open-source technologies and tools to minimize software licensing costs.
- The system should be designed to run efficiently on low-cost hardware to minimize hardware costs.
- The system should be designed to be easily upgradable and scalable to reduce the need for costly replacements or upgrades in the future.
- The system should have a clear and transparent pricing model to help customers understand the costs associated with using the system.

3.5 Feasibility Study

The aim of the feasibility study activity is to determine whether it would be the financially and technically feasible to develop the system or not. A feasibility studies is carried out from following different aspects:

1) Operational Feasibility

This assessment involves undertaking a study to analyze and determine whether and how well the organization's needs can be met by completing the project. Operational feasibility studies also examine how a project plan satisfies the requirements identified in the requirements analysis phase of system development. The system has been developed for all the users who are interested in this product, irrespective of technical background. We have given a demo of our project to technical as well as non-technical users and all the users found the system user friendly.

2) Technical Feasibility

This assessment focuses on the technical resources available to the organization. It helps organizations determine whether the technical resources meet capacity and whether the technical team is capable of converting the ideas into working systems. Technical feasibility also involves the evaluation of the hardware, software, and other technical requirements of the proposed system.

3) Implementation Feasibility

This project can easily be made available online without much consideration of the hardware and software. The only required thing at the applicant's side is the Internet connection, which is a not difficult issue these days. After setting up the project, all the users can access and configure the system from any smartphone connected with the same network. Also, these particular modules can be controlled remotely through other devices.

4) Scheduling Feasibility

This assessment is the most important for project success; after all, a project will fail if not completed on time. In scheduling feasibility, an organization estimate how much time the project will take to complete. When these areas have all been examined, the feasibility analysis help identify any constraints the proposed project may face, including:

• Internal Project Constraints: Technical, Technology, Budget, Resource, etc.

• External Constraints: Logistics, Environment, Laws, and Regulations, etc.

3.6 Use Case Diagram

A use case diagram in UML is a type of behavioral diagram that shows the interactions between users (actors) and a system or a software application. It is a high-level view of the system's requirements from the perspective of its users and can help to ensure that all stakeholders have a clear understanding of the system's functionality. A use case diagram in UML is a useful tool for visualizing the functional requirements of a system from the perspective of its users. It can help to ensure that all stakeholders have a clear understanding of the system's requirements and can guide the design and development of the system. To create a use case diagram in UML, you should identify the actors and use cases, create the diagram, add relationships, and refine the diagram with additional details as needed.

To create a use case diagram in UML, you should follow the following steps:

Identify the actors: The first step is to identify the actors who will interact with the system. An actor is someone or something that interacts with the system and can be a user, another system, or an external entity. Actors are represented by stick figures on the left-hand side of the diagram. You should identify all the relevant actors for the system and add them to the diagram.

Identify the use cases: The next step is to identify the use cases that the system must perform to meet the needs of its users. A use case describes a specific goal or objective that a user wants to achieve when interacting with the system. Each use case should have a clear and specific goal that the user wants to achieve. Use cases are represented by ovals or ellipses on the right-hand side of the diagram.

Create the diagram: Once you have identified the actors and use cases, you can create the use case diagram in UML. The diagram should show the actors as stick figures on the left-hand side of the diagram and the use cases as ovals or ellipses on the right-hand side of the diagram. Arrows connect the actors to the use cases that they interact with.

The use case diagram should provide a clear and concise representation of the system's functionality and how it meets the needs of its users.

Add relationships: You can add relationships between the actors and use cases to indicate the types of interactions that occur. For example, an actor might initiate a use case or be an external system that communicates with the system. You can use various symbols, such as solid lines, dashed lines, and arrows, to represent the different types of relationships. You should ensure that all the actors and use cases are connected in a way that accurately reflects the system's functionality and requirements.

Refine the diagram: Finally, you can refine the diagram by adding additional details, such as preconditions, postconditions, and exceptions. Preconditions describe the conditions that must be met before a use case can be executed, while postconditions describe the conditions that exist after the use case has been executed. Exceptions describe the situations where the use case may fail or not produce the expected outcome. Adding these details can help to clarify the requirements of the system and ensure that all scenarios are covered. Use case diagram in shown in Fig 3.3.

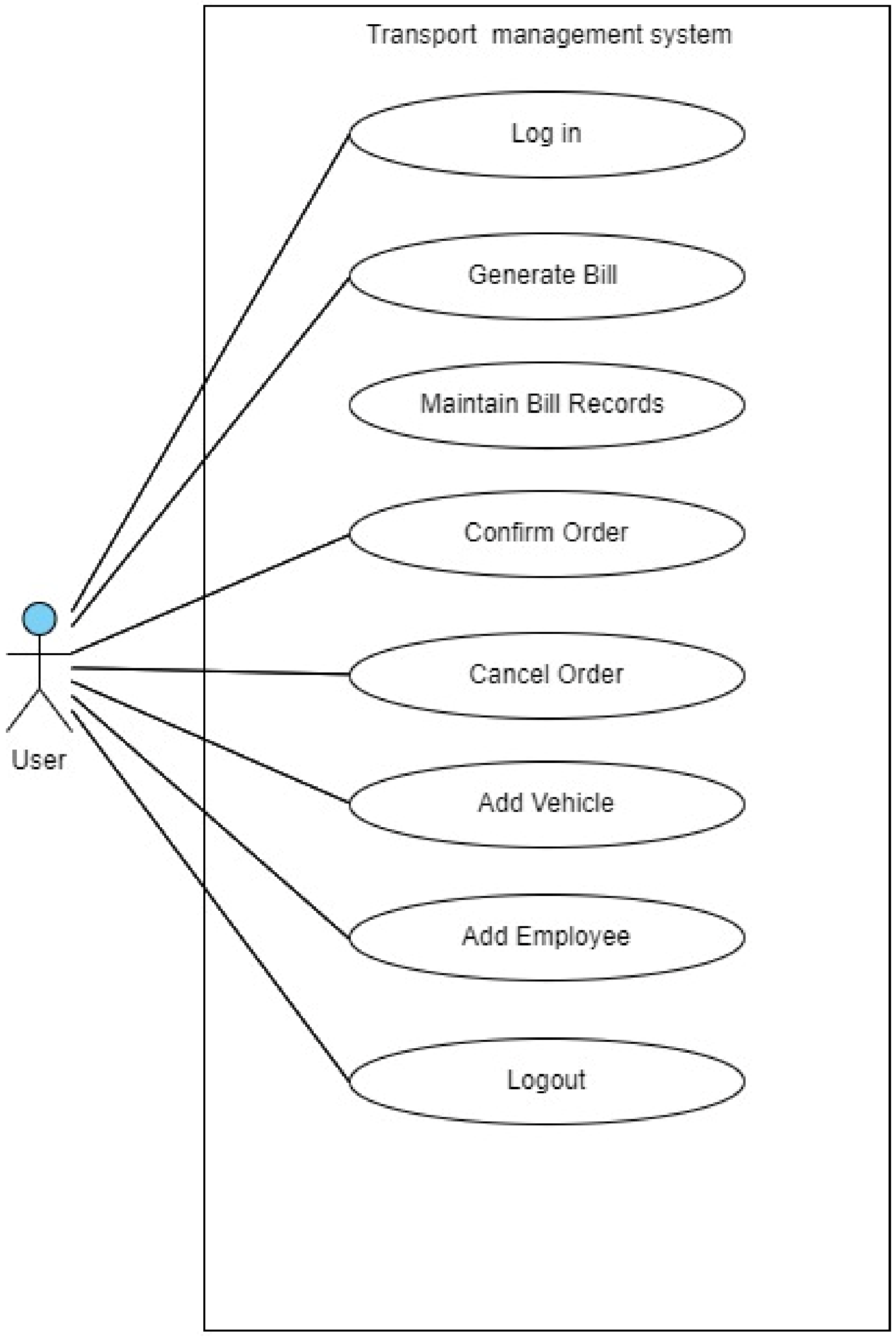


Fig 3.3: Use case diagram

3.7 Use Case Specification

Use Case Name: Manage Transportation Requests
Primary Actor: User
Goal in Context: The Operations Manager creates and manages transportation requests for shipments.

Preconditions:
The Operations Manager has a valid user account and is logged in to the transport management system.
The Operations Manager has appropriate permissions to create and manage transportation requests.

Success End Condition:
The transportation request is successfully created and added to the system.

Main Success Scenario:

1) The Operations Manager selects the "Create New Transportation Request" option from the main menu.
2) The system displays a form for the Operations Manager to enter the details of the transportation request, including shipment details, pickup and delivery locations, preferred carrier, and any special requirements or instructions.
3) The Operations Manager fills out the form and submits it to the system.
4) The system validates the form data and adds the transportation request to the system.
5) The Operations Manager is notified that the transportation request has been successfully added to the system.
6) Extensions:
7) If the form data is invalid or incomplete, the system displays an error message and prompts the Operations Manager to correct the errors.
8) If the preferred carrier is not available or cannot fulfill the request, the system prompts the Operations Manager to select an alternative carrier.

9) If there are any special requirements or instructions that cannot be accommodated by the selected carrier, the system prompts the Operations Manager to modify the request or select a different carrier.

Alternative Scenarios:

- Cancel Transportation Request: The Operations Manager can cancel a transportation request that has not yet been fulfilled by selecting the "Cancel Request" option from the request details page.
- Edit Transportation Request: The Operations Manager can edit a transportation request that has not yet been fulfilled by selecting the "Edit Request" option from the request details page.
- View Transportation Request Details: The Operations Manager can view the details of a transportation request by selecting the request from the list of active requests.
- Approve Transportation Request: If the Operations Manager is not authorized to approve transportation requests, the system prompts the Operations Manager to forward the request to the appropriate approver.

4. DESIGN

4.1 Design Goal

A transport management system (TMS) is a software application that helps businesses manage their transportation operations, including planning, execution, and optimization of freight movements. Here are some design goals for a TMS:

- Efficiency: A TMS should be designed to help businesses manage their transportation operations more efficiently. This includes automating processes, reducing manual labor, and streamlining workflows.
- Flexibility: A TMS should be flexible enough to accommodate different types of transportation modes, including air, land, and sea transportation. It should also be able to handle different types of freight, such as bulk, containerized, and hazardous materials.
- Scalability: A TMS should be designed to scale as the business grows. It should be able to handle an increasing volume of transactions and users without compromising performance.
- Integration: A TMS should be designed to integrate with other systems such as ERP (enterprise resource planning) and WMS (warehouse management system). This allows businesses to have a seamless flow of information across different systems.
- Visibility: A TMS should provide real-time visibility into transportation operations. This includes tracking shipments, providing alerts for exceptions, and generating reports to help businesses make informed decisions.
- Cost-Effectiveness: A TMS should be designed to help businesses reduce transportation costs. This includes optimizing routes, consolidating shipments, and negotiating better rates with carriers.
- User-friendly: A TMS should be user-friendly and intuitive to use. This ensures that users can easily navigate the system and perform tasks without extensive training.
- Security: A TMS should be designed with security in mind. It should have adequate measures to protect sensitive data, prevent unauthorized access, and ensure data privacy.

4.2 Design Strategy

As we have researched a lot about how and what to do in our project. As we go deeper and deeper into research, we come to know there are various aspects to do. That's why we figured out how we can go further and plan our task so that the requirements of our project get fulfilled. So here we go following the approach shown in Fig 4.1. We have divided into a certain tasks. They are as follows:

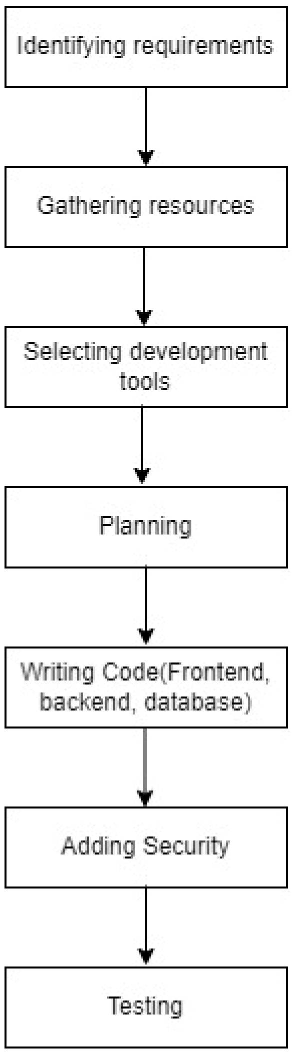


Fig 4.1: Strategy Diagram

Task 1: Identifying requirements

Firstly, we identified the requirements for this project. This task involves identifying the functionality required in the system. We analysed the company needs of what exactly the system they need to get designed.

Task 2: Gathering resources

We searched for best available resources for our project.

Task 3: Selecting Development Tools

In this step we selected the development tools that we found useful to build the interactive system. This involves choosing the programming language, backend framework, selecting database, etc.

Task 4: Planning

This step involves distributing task among group members.

Task 5: Writing Code

This step involves writing code for frontend, backend and database.

Task 6: Adding security

Adding security by building a secured login system for user authentication.

Task 7: Testing

This step involves checking errors in the code, identifying the bugs, finding out whether the system provides the required functionalities, etc.

4.3 Sequence Diagram

Sequence diagrams are a popular dynamic modeling solution in UML because the specifically focus on lifelines, or the processes and objects that live simultaneously, and the messages exchanged between them to perform a function before the lifeline ends. They are the most commonly used Interaction diagrams. The sequence diagram represents the flow of messages in the system and is also termed as an event diagram. It helps in envisioning several dynamic scenarios. It portrays the communication between any two lifelines as a time-ordered sequence of events, such that these lifelines took part in the run time. In UML, the lifeline is represented by a vertical bar, whereas the message flow is represented by a vertical dotted line that extends across the bottom of the page. It incorporates iterations as well as branches.

Purpose of Sequence Diagrams

- To model high-level interaction among active objects within a system.
- To model interaction among objects inside a collaboration realizing a use case.
- It either models generic interactions or some certain instances of interaction

A sequence diagram is a type of interaction diagram in UML (Unified Modeling Language) that shows the interactions between objects in a system over time. It depicts the messages exchanged between objects and the order in which they occur. Here are the main components of a sequence diagram in UML:

- ➢Objects: Objects represent instances of classes or components that participate in the interaction. They are typically named with a noun or noun phrase that describes their role in the system. For example, in a sequence diagram for an online shopping system, objects might include "Customer," "ShoppingCart," "Inventory," and "PaymentProcessor."
- ➢Lifelines: Lifelines are vertical lines that show the existence of an object over time. They are typically labeled with the name of the object and a horizontal dashed line to indicate the object's destruction. Lifelines can also be nested within other lifelines to represent objects that are created or destroyed by other objects.

- ➢Messages: Messages show the communication between objects in the system. They are represented by arrows that point from the sender to the receiver, and are labeled with the name of the message and any parameters or data that are transmitted. Messages can be synchronous, asynchronous, or self-referential:
- ➢Synchronous messages are represented by solid arrows and indicate that the sender object waits for a response from the receiver object before continuing.
- ➢Asynchronous messages are represented by open arrows and indicate that the sender object does not wait for a response from the receiver object before continuing.
- ➢Self-referential messages are represented by a loop arrow and indicate that an object sends a message to itself.
- ➢Activation Bar: The activation bar represents the time period during which an object is executing a message. It is a horizontal bar that extends from the lifeline of the object and is labeled with the name of the message. The length of the activation bar indicates the duration of the message processing time.
- ➢Return Message: A return message is a message that shows the response from the receiving object to the sending object. It is represented by a dashed arrow that points back from the receiver to the sender. Return messages are typically labeled with the name of the message and any return values or data that are transmitted.
- ➢Combined Fragment: A combined fragment is a UML construct that shows a complex interaction between objects. It is represented by a rectangle with a specific keyword that indicates the type of interaction. Some common keywords include:
    - alt (alternatives): Shows alternative paths of execution based on conditions or decisions.
    - opt (optional): Shows optional paths of execution that may or may not be executed.
    - loop (iteration): Shows a repeating sequence of messages or interactions.
    - par (parallel): Shows concurrent or parallel execution of multiple interactions.

Sequence Diagram in shown in Fig. 4.2.

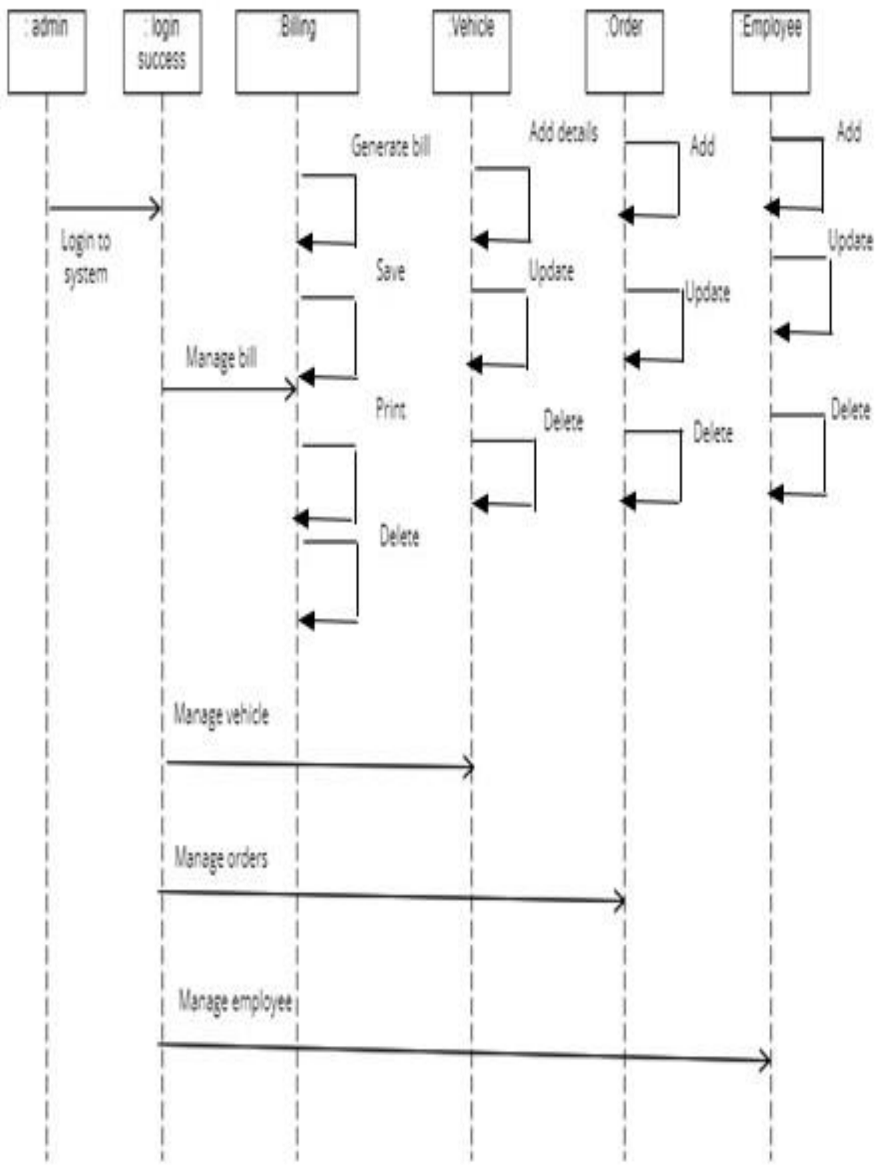


Fig 4.2: Sequence Diagram

4.4 Collaboration Diagram

The collaboration diagram is used to show the relationship between the objects in a system. Both the sequence and the collaboration diagrams represent the same information but differently. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming. An object consists of several features. Multiple objects present in the system are connected to each other. The collaboration diagram, which is also known as a communication diagram, is used to portray the object's architecture in the system. The collaborations are used when it is essential to depict the relationship between the object. Both the sequence and collaboration diagrams represent the same information, but the way of portraying it is quite different. The collaboration diagrams are best suited for analyzing use cases.

Purpose of Collaboration Diagrams

- The collaboration diagram is also known as Communication Diagram.
- It mainly puts emphasis on the structural aspect of an interaction diagram, i.e., how lifelines are connected.
- The syntax of a collaboration diagram is similar to the sequence diagram; just the difference is that the lifeline does not consist of tails.
- The messages transmitted over sequencing is represented by numbering each individual message.
- The collaboration diagram is semantically weak in comparison to the sequence diagram.
- The special case of a collaboration diagram is the object diagram.
- It focuses on the elements and not the message flow, like sequence diagrams.

Fig 4.3 shows a Collaboration diagram.

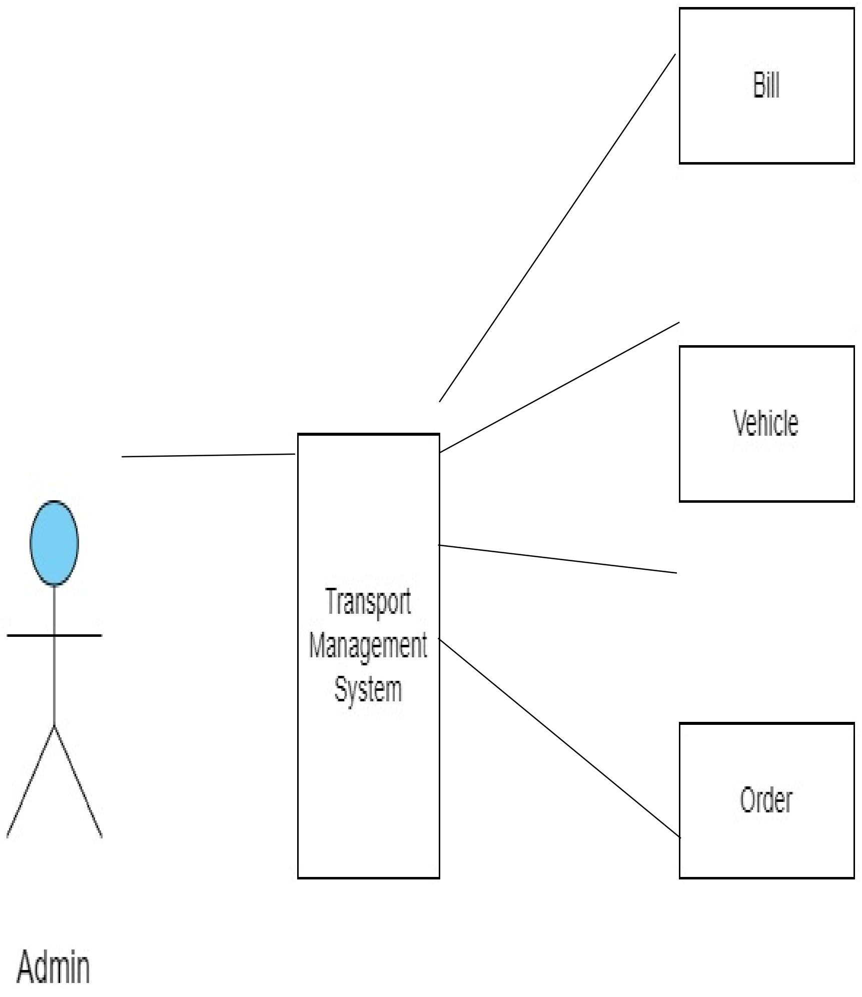


Fig 4.3: Collaboration Diagram

4.5 State Chart Diagram

State chart diagrams provide us an efficient way to model the interactions or communication that occurs within the external entities and a system. These diagrams are used to model the event-based system. A state of an object is controlled with the help of an event. State chart diagrams are used to describe various states of an entity within the application system State chart diagrams provide a visual representation of the behavior of a system. They can help developers and stakeholders understand the interactions between different components of the system and identify areas where improvements can be made. By modeling the behavior of a system in this way, it becomes easier to test and validate the system's functionality. State chart diagrams are an essential tool for software developers and engineers working on complex systems.

Purpose of State Chart Diagrams

State chart diagram is one of the five UML diagrams used to model the dynamic nature of a system. They define different states of an object during its lifetime and these states are changed by events. State chart diagrams are useful to model the reactive systems. Reactive systems can be defined as a system that responds to external or internal events. State chart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of State chart diagram is to model lifetime of an object from creation to termination. State chart diagrams are also used for forward and reverse engineering of a system. However, the main purpose is to model the reactive system. Following are the main purposes of using State chart diagrams –

- To model the dynamic aspect of a system.
- To model the life time of a reactive system.
- To describe different states of an object during its life time.
- Define a state machine to model the states of an object

Fig 4.4 shows State Chart Diagram.

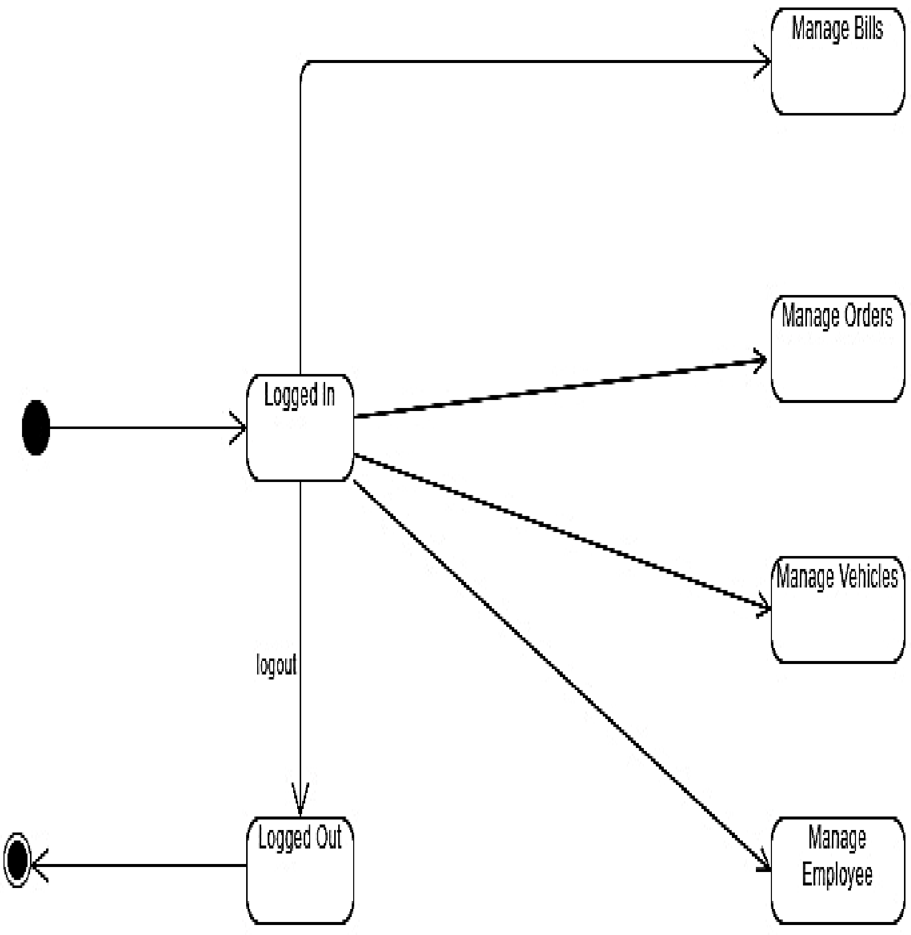


Fig 4.4 : State Chart Diagram

4.6 Activity Diagram

The general work-flow of the planner can be graphically represented in an activity diagram. Figure 4. shows how user will use the system and the step-by-step process they will go through as they progress through the site. The diagram shows the workflow for all average user. The user is then able to interact with selected modules, or open new modules. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another

Purpose of Activity Diagrams

The basic purpose of activity diagrams is similar to other UML diagrams. It captures the dynamic behavior of the system. Other UML diagrams are used to show the message flow from one object to another but the activity diagram is used to show message flow from one activity to another.
Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part.
It does not show any message flow from one activity to another. Activity diagram is sometimes considered as the flowchart. Although the diagrams look like a flowchart, they are not. It shows different flows such as parallel, branched, concurrent, and single
The purpose of an activity diagram can be described as:

- Draw the activity flow of a system.
- Describe the sequence from one activity to another.
- Describe the parallel, branched and concurrent flow of the system

Fig 4.5 shows activity diagram.

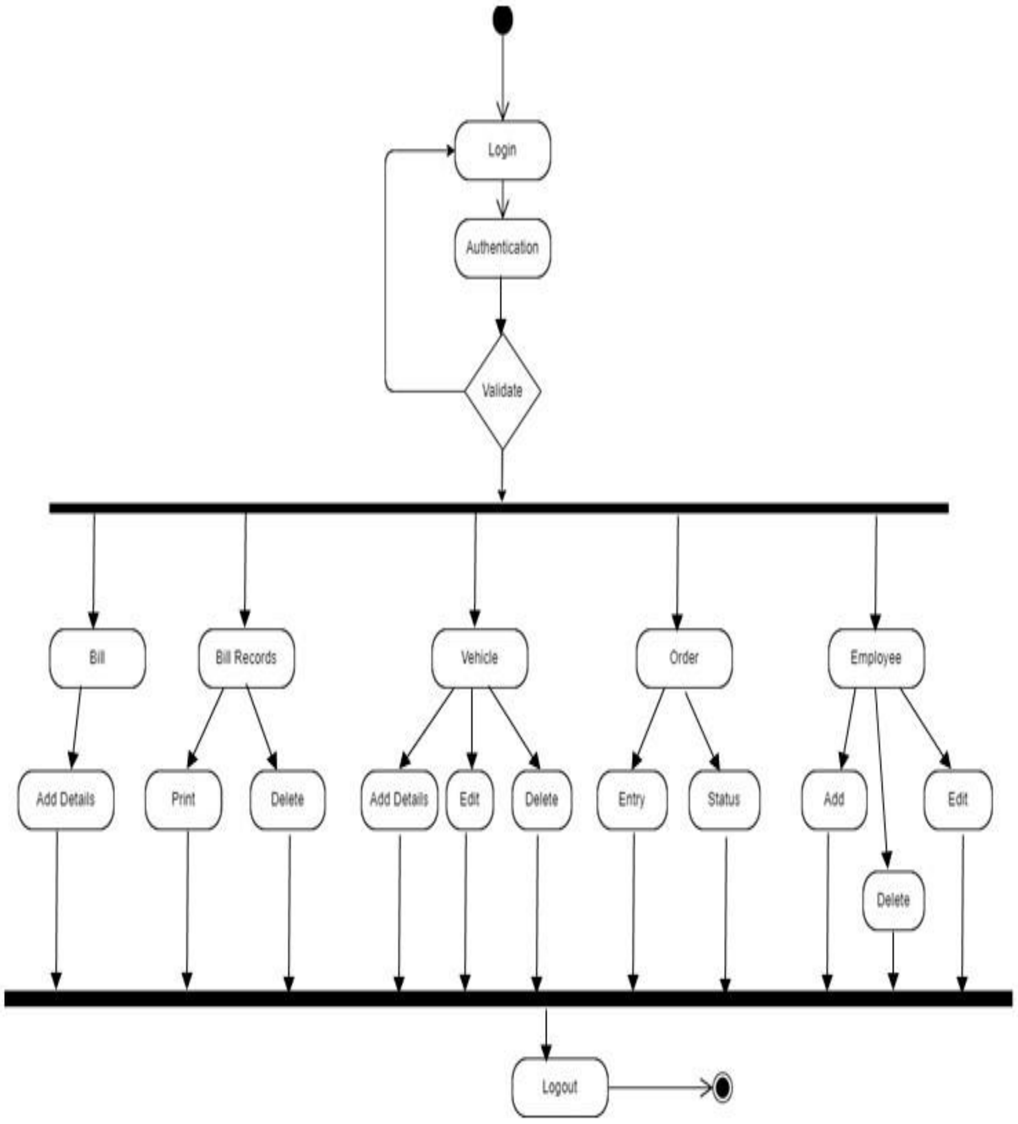


Fig 4.5 : Activity Diagram

5. IMPLEMENTATION

5.1 Implementation Strategy

To implement a transport management system using Flask and MySQL, you would need to set up a MySQL database with tables for storing data related to vehicles, drivers, customers, and bookings. Next, you would need to install the Flask framework along with necessary extensions such as Flask-MySQL and Flask-WTF. You would then create a Flask application and define the routes and views for the transport management system, such as a login page, dashboard, and pages to create, update, and delete vehicles, drivers, customers, and bookings.

To represent the data for the transport management system, you would define necessary models and classes, such as a Vehicle class with attributes like ID, make, model, year, and capacity, along with methods to interact with the database. You would then use Flask-MySQL to connect to the MySQL database and handle database interactions, such as querying for data, inserting new records, and updating or deleting existing records.
Using Flask-WTF, you would create forms for user input, such as a form to create a new vehicle or booking, and validate the form data before submitting it to the database. Flask's built-in templating engine can then be used to render HTML templates for the web pages, while CSS and JavaScript can be used to style and add interactivity to the pages.

Testing the transport management system would be necessary to ensure that it works as expected, which can include testing the database interactions, form validation, and user interface. Finally, you would deploy the application to a web server such as Apache or Nginx to make it accessible to users over the internet. Overall, using Flask and MySQL together provides a powerful and flexible platform for building web applications, and can be used to create a wide variety of software systems, including transport management systems.

5.2 Hardware Platform Used

- Laptop/PC
- System: Intel Processor i3/i5/i7 or AMD processors
- RAM
- Hard Disk: 1GB

5.3 Software Platform Used

- Python
- Flask
- MySQL
- Jinja2
- Sqlalchemy
- Visual Studio Code
- XAMPP

5.4 Software Specification

In this project we used various python libraries to create web application, we also used scripting languages like HTML, CSS for web application development and web framework which is flask uses python.

1) Python

Python is a popular programming language used in web development for building web applications, web APIs, and dynamic websites. Here are some key areas where Python is used in web development:

- ➢ Web frameworks: Python has several web frameworks that simplify the process of building web applications. Django is a popular web framework that follows the Model-View-Controller (MVC) architecture and provides a lot of out-of-the-box features such as authentication, admin interface, and ORM. Flask is a lightweight framework that follows the Model-View-Template (MVT) architecture and allows developers to have more control over the application's structure. Pyramid is a flexible framework that supports various development styles and is suitable for building large-scale applications.

- Web scraping: Python is commonly used for web scraping, which involves extracting data from websites. There are several Python libraries like Beautiful Soup, Scrapy, and Requests that make web scraping easy and efficient.
- Web APIs: Python is often used to create web APIs (Application Programming Interfaces) that allow different applications to communicate with each other. Flask and Django Rest Framework are popular libraries for building web APIs.
- Server-side scripting: Python is used as a server-side scripting language, which means it is executed on the server-side to generate dynamic content that is sent to the client-side. Python can be used in conjunction with web servers like Apache and Nginx to create dynamic web pages.
- Machine learning: Python has a strong ecosystem for machine learning and data science, which can be used to build web applications that use artificial intelligence and predictive models. Flask and Django are often used in conjunction with machine learning libraries like TensorFlow and Keras.

Python is important in web development for several reasons:

- Versatility: Python is a versatile language that can be used for various web development tasks, including web scraping, web development, web APIs, server-side scripting, and machine learning.
- Large ecosystem: Python has a large ecosystem of libraries and frameworks that make it easy to develop web applications. There are several web frameworks like Django, Flask, Pyramid, and Bottle, as well as libraries like Beautiful Soup, Scrapy, Requests, and others.
- Easy to learn: Python is an easy-to-learn language that has a simple and readable syntax. This makes it easy for developers to understand and maintain the codebase, and to onboard new developers to the project.

- Rapid prototyping: Python's simplicity and ease-of-use make it ideal for rapid prototyping. Developers can quickly build and test web applications, and iterate on them based on feedback.
- Community support: Python has a large and active community of developers who contribute to the language and its ecosystem. This community provides support, documentation, and resources to help developers build better web applications.

2) Flask

Flask is a Python web framework that is used to build web applications. It is a lightweight framework that provides a lot of flexibility to developers. Flask's small codebase and minimal dependencies make it easy to install and use. The framework is known for its simplicity and easy-to-learn syntax, which makes it an attractive choice for developers who are new to web development or who want to quickly build and test web applications. Flask is a modular web framework, meaning that developers can use only the features they need, keeping the codebase small and efficient. Additionally, Flask is an extensible web framework that allows developers to easily add new functionality through a large ecosystem of plugins and extensions [8].

There are several reasons why developers choose to use Flask for web development:

- Lightweight: Flask is a lightweight web framework that has a small codebase and minimal dependencies. This means that it is easy to install and use, and the resulting web applications tend to be faster and more efficient.
- Flexibility: Flask is a flexible web framework that allows developers to build web applications the way they want. It does not impose any particular structure or requirements, so developers can customize their applications based on their needs.
- Easy to learn: Flask has a simple and easy-to-learn syntax. This makes it a popular choice for developers who are new to web development or who want to quickly build and test web applications.

- Modularity: Flask is a modular web framework that allows developers to use only the features they need. This means that developers can keep the codebase small and efficient, and reduce the chances of bugs or security vulnerabilities.
- Extensibility: Flask is an extensible web framework that allows developers to easily add new functionality through a large ecosystem of plugins and extensions. This makes it easy to incorporate features like authentication, database integration, and more into web applications.

In addition to the features mentioned above, Flask also provides a number of other benefits for web developers:

- Low barrier to entry: Flask is an excellent choice for developers who are new to web development, as it is easy to learn and understand. The framework has a small codebase and is well-documented, making it easy to get started with.
- Rapid prototyping: Flask is a great choice for rapid prototyping, as it allows developers to quickly build and test web applications. This can be particularly useful for startups and other organizations that need to quickly test new ideas and iterate on them.
- Large and active community: Flask has a large and active community of developers who contribute to the framework and its ecosystem. This community provides support, documentation, and resources to help developers build better web applications.
- Good for microservices: Flask is a popular choice for building microservices, which are small, independent services that work together to form a larger application. Flask's lightweight and modular architecture makes it well-suited for this type of development.
- Good for small projects: Flask is an excellent choice for small web development projects, as it allows developers to quickly build and deploy applications without a lot of overhead. This can be particularly useful for individuals or small teams who need to build web applications on a tight budget or timeline.

3) MySQL

MySQL is an open-source relational database management system (RDBMS) that is widely used in web development. It is a popular choice for web developers because it is fast, reliable, and easy to use. Here are some key features of MySQL [27].

- Relational database: MySQL is a relational database, which means that it stores data in tables that are related to each other. This allows developers to easily organize and manage data in a structured way.
- ACID-compliant: MySQL is ACID-compliant, which means that it ensures that transactions are processed reliably and consistently. This ensures data integrity and prevents data corruption or loss.
- Fast and scalable: MySQL is designed to be fast and scalable, making it an excellent choice for web applications that require high performance and can handle a large amount of data.
- Widely used: MySQL is one of the most widely used database management systems in the world. This means that there is a large community of developers who are familiar with the system and can provide support and resources.
- Open-source: MySQL is an open-source software, which means that it is freely available and can be modified and distributed by anyone. This makes it a popular choice for developers who want to use a reliable and flexible database system without the cost of licensing fees.
- Cross-platform compatibility: MySQL is compatible with a wide range of operating systems, including Windows, Linux, and macOS. This makes it a versatile option for web developers who need to work with different systems.
- Easy to use: MySQL is easy to learn and use, even for developers who are new to database management systems. It has a simple and intuitive user interface, and its command line interface is also straightforward.
- Flexible data storage: MySQL supports different data storage engines, including InnoDB and MyISAM. This allows developers to choose the best storage engine for their specific needs, whether they need to optimize for speed, space, or other factors.
- High availability: MySQL supports high availability through features such as replication and clustering. This allows developers to ensure that their web applications remain available and responsive even in the event of hardware or software failures.
- Security: MySQL has a range of security features, including user authentication, encryption, and auditing. This helps developers to protect sensitive data and prevent unauthorized access.

- Integrations: MySQL integrates with a wide range of web development tools and frameworks, including PHP, Python, and Ruby on Rails. This allows developers to build web applications in their preferred programming language and framework, while still leveraging the power and flexibility of MySQL.
- Large-scale deployments: MySQL is used by some of the world's largest web applications, including Facebook, Google, and Twitter. This demonstrates its ability to handle large-scale deployments and high volumes of data.

4) Jinja2

Jinja2 is a popular templating engine for Python that is widely used in web development. It allows developers to generate dynamic HTML, XML, and other markup languages by combining templates with data from a web application. Here are some key features of Jinja2 [10].

- Templating language: Jinja2 is a templating language that allows developers to separate presentation logic from business logic in web applications. This helps to make code more modular, maintainable, and reusable.
- Easy to learn: Jinja2 is easy to learn and use, even for developers who are new to web development. Its syntax is simple and intuitive, and it integrates well with other Python frameworks and libraries.
- Expressive syntax: Jinja2 has an expressive syntax that makes it easy to write complex templates. It supports a wide range of control structures, including loops, conditionals, and macros, and it allows developers to define their own custom filters and functions.
- Inheritance: Jinja2 supports template inheritance, which allows developers to define a base template and then override specific blocks of content in child templates. This helps to reduce duplication and makes it easier to manage and maintain templates.
- Extensibility: Jinja2 is highly extensible, allowing developers to create their own custom tags, filters, and extensions. This makes it possible to customize Jinja2 to meet the specific needs of a web application.
- Fast: Jinja2 is designed to be fast and efficient, which makes it a good choice for web applications that require high performance. It uses a caching system to

optimize the rendering of templates and minimize the overhead of parsing and compiling.

5) SQLAlchemy

SQLAlchemy is a popular ORM library for Python that provides an easy-to-use interface for working with relational databases. It allows developers to map Python objects to database tables, and to manipulate data using Python code, without having to write SQL directly. SQLAlchemy supports multiple database backends, including MySQL, PostgreSQL, SQLite, and Oracle, making it possible to write database-independent code that can be easily ported to different database systems. SQLAlchemy also provides a powerful querying interface that allows developers to build complex queries using Python code. It supports a wide range of filtering, sorting, grouping, and aggregation operations, as well as subqueries and joins. Additionally, SQLAlchemy helps to ensure data consistency by providing transactional support and enforcing database constraints, preventing data corruption and ensuring that database operations are atomic and consistent. Finally, SQLAlchemy is designed to be fast and efficient, with a lightweight object-relational mapping layer that minimizes overhead. It also supports advanced caching and connection pooling options to optimize performance [1].

SQLAlchemy is a very powerful ORM library for Python, and its features extend far beyond basic data mapping. Some additional features include:

- Relationships: SQLAlchemy allows developers to define relationships between objects and their corresponding tables. This can simplify complex data models and allow developers to easily access related data without writing additional queries.
- Database migration: SQLAlchemy includes a tool called Alembic for managing database migrations. This tool allows developers to create and apply database schema changes in a version-controlled and incremental way.
- Connection pooling: SQLAlchemy includes support for connection pooling, which can improve performance by reusing database connections instead of creating new ones for each request. Connection pooling can be configured to balance between the number of active connections and the number of idle connections, providing optimal performance and resource usage.

- Built-in SQL expression language: SQLAlchemy provides a built-in SQL expression language that can be used to build complex queries and statements. This allows developers to write SQL code in a Pythonic syntax, which is more readable and maintainable than traditional SQL code.
- Batch operations: SQLAlchemy supports batch operations, which can improve performance when inserting, updating, or deleting large numbers of rows. This allows developers to reduce the number of round-trips to the database, and to optimize the performance of their database operations.
- Integration with web frameworks: SQLAlchemy integrates with many popular Python web frameworks, such as Flask, Django, and Pyramid. This allows developers to easily use SQLAlchemy in their web applications, and to take advantage of its powerful features and capabilities.

5) Visual Studio Code

Visual Studio Code is a free and open-source code editor developed by Microsoft for Windows, macOS, and Linux operating systems. It provides a rich and customizable user interface, along with a wide range of features designed to enhance productivity and make coding easier and more efficient.

One of the key features of Visual Studio Code is its support for multiple programming languages and frameworks. It includes built-in support for popular languages such as Python, JavaScript, TypeScript, and C++, and provides extensions for many more. These extensions can be easily installed from the Visual Studio Code Marketplace, allowing developers to customize the editor to their specific needs.

Visual Studio Code also includes a powerful debugging and testing system, which can be used to debug code directly from the editor. It includes support for breakpoints, variable inspection, and step-by-step debugging, as well as integration with popular testing frameworks such as Jest and Mocha.

Another key feature of Visual Studio Code is its integrated source control system. It includes support for Git out-of-the-box, and provides a range of features for managing repositories, committing changes, and resolving conflicts. It also integrates with popular source control platforms such as GitHub, Bitbucket, and Azure DevOps.

In addition to its rich feature set, Visual Studio Code is also highly customizable. It provides a range of themes and icons, and allows users to configure keyboard shortcuts and other preferences to suit their workflow. It also includes a powerful extension API, which allows developers to create their own custom extensions to enhance the editor's functionality.

Visual Studio Code provides IntelliSense, a code completion feature that offers suggestions and auto-completion for code as you type. This feature is context-aware and language-specific, which means that it can provide accurate suggestions based on the current programming language and the context in which the code is being written.

Visual Studio Code includes a built-in terminal that allows developers to run commands directly from the editor. This can be very useful for tasks such as running scripts, starting servers, or installing dependencies. The terminal can also be customized to use different shells or command-line interfaces [36].

6) XAMPP

XAMPP is a popular, open-source software package that provides a complete web development environment. It is designed to work on Windows, Linux, and macOS operating systems, and includes all the components required to run a web server, including Apache HTTP Server, MySQL, and PHP.

One of the key benefits of XAMPP is that it allows developers to quickly and easily set up a local web server for testing and development purposes. This can be particularly useful for developers who need to test their web applications in a local environment before deploying them to a live server. By using XAMPP, developers can simulate a live server environment on their own computer, allowing them to test and debug their applications more easily.

XAMPP includes a number of components that are essential for web development. Apache HTTP Server is the web server component, which is responsible for serving web pages to clients. MySQL is the database component, which provides a powerful and flexible database management system. PHP is the scripting language component, which is used to create dynamic and interactive web pages.

In addition to these core components, XAMPP also includes several other useful tools and utilities. For example, it includes phpMyAdmin, a popular tool for managing MySQL databases through a web interface. It also includes FileZilla FTP Server, which can be used to transfer files to and from the server.

Another benefit of XAMPP is its ease of use. The software can be installed with just a few clicks, and once installed, it provides a simple and intuitive control panel for managing the web server and other components. The control panel allows developers to start and stop the server, configure settings, and access logs and other useful information.

5.5 Deployment Diagram

Deployment diagrams are used to visualize the topology of the physical components of a system, where the software components are deployed as shown in below figure 5.1. Deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.

A deployment diagram in UML is a type of diagram that shows the physical deployment of software artifacts on hardware nodes. It provides a high-level view of the system architecture and shows how software components are deployed and distributed across different hardware nodes. Here are some key elements of a deployment diagram [37]:

- Nodes: Nodes are the physical hardware devices on which the software artifacts are deployed. They can be represented as boxes or circles with the name of the device inside.
- Artifacts: Artifacts are the software components that are deployed on the nodes. They can be represented as rectangles with the name of the artifact inside.
- Deployment relationships: Deployment relationships show how artifacts are deployed on nodes. There are two types of deployment relationships:
- Dependency: A dependency relationship shows that one artifact depends on another artifact. For example, a web application may depend on a database management system to store data.
- Association: An association relationship shows that an artifact is deployed on a node. For example, a web application may be deployed on a web server.
- Communication pathways: Communication pathways show how nodes communicate with each other. They can be represented as lines connecting nodes, with arrows indicating the direction of communication.
- Environment: The environment is the context in which the system operates, including the physical hardware, software, and network infrastructure. It can be represented as a boundary around the nodes and artifacts.

Fig 5.1 shows a Deployment Diagram.

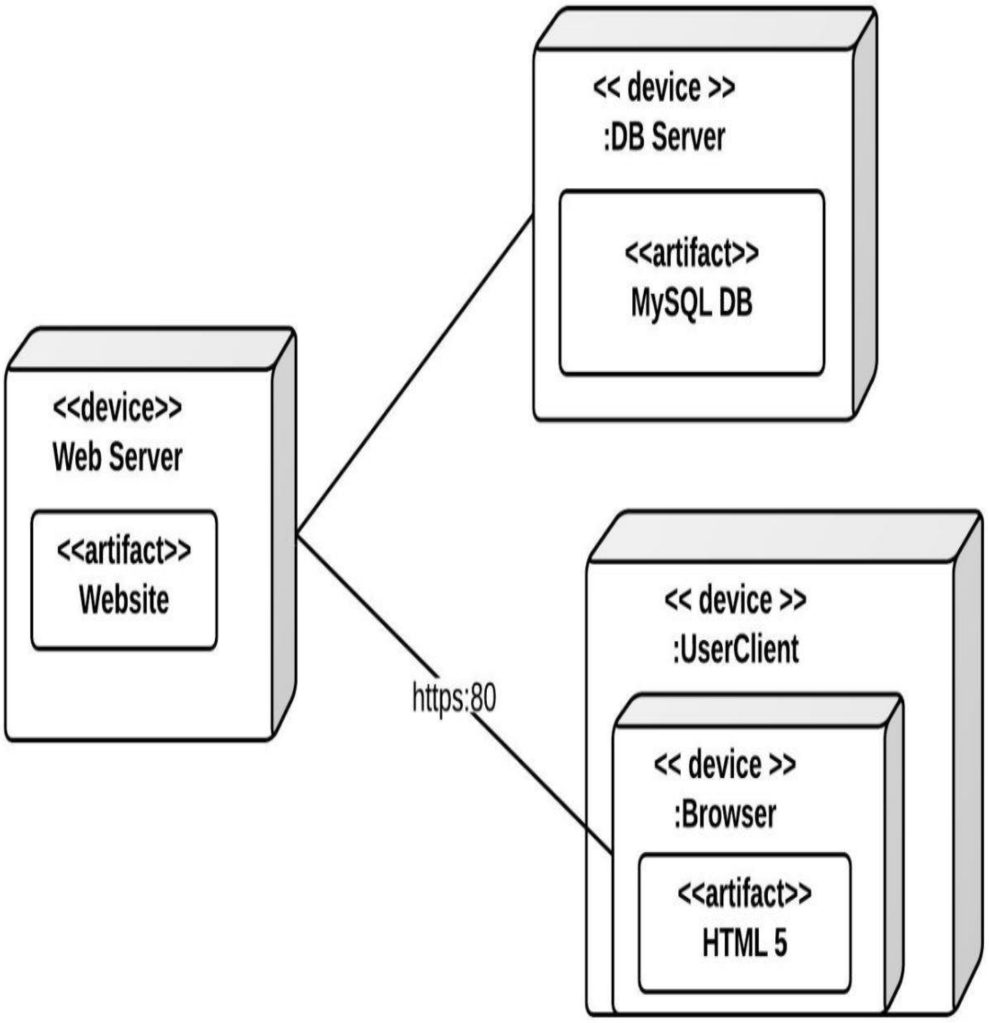


Fig 5.1 Deployment Diagram

5.6 Implementation Level Details

In this chapter we will see how the project works. As soon as the user enters the system, He has to enter user-id and password. If both the credentials match with the database then user will see the dashboard. User can see chat symbol in the homepage. In the navigation bar user will see many sections such as Bill, Records, Employee, Vehicle, Order. User can select the options according to their task. Fig 5.2 shows the flowchart of the system.

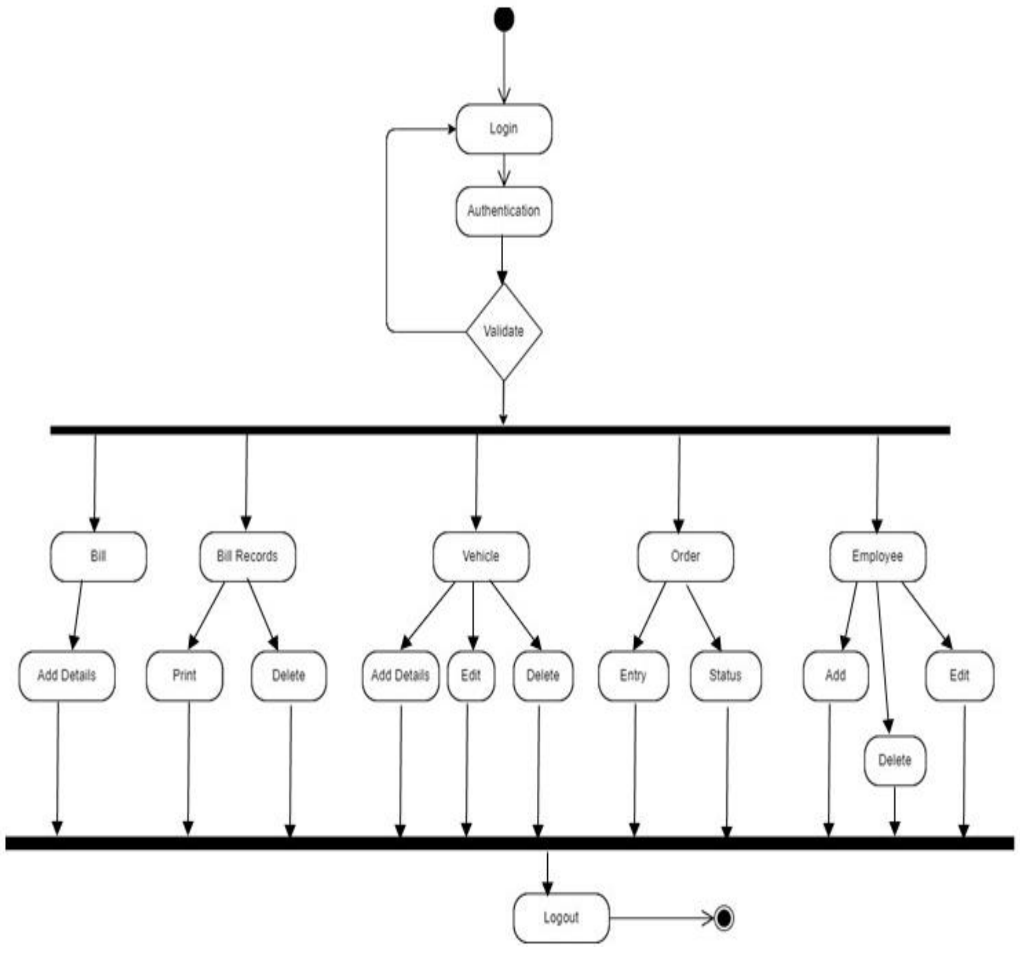


Fig 5.2 Implementation Level Details

5.7 Testing

GUI testing:

Graphical User Interface (GUI) testing is the one of the mechanism in which user interface developed System Under some graphical rules. GUI testing includes checking various controls- menus, buttons, icons, dialog boxes and windows etc.

Unit Testing:

It is the testing of individual software units of the application. It is done after the complexion of an individual unit before integration. Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. This is a structural testing that relies on knowledge of its construction and is invasive. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration Testing: Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

System testing: System testing tests a completely integrated system to verify that the system meets its requirements. For example, a system test might involve testing a logon interface, then creating and editing an entry, plus sending or printing results, followed by summary processing or deletion (or archiving) of entries, then logout [35].

6. CONCLUSION

Transport Management System project is a comprehensive software system designed to help transport companies streamline their operations and improve their overall efficiency. The system includes several key features such as vehicle management, order management, driver management, and customer invoicing. The system is designed to be user-friendly and intuitive, with a modern and visually appealing interface that allows users to quickly access the information they need. It is also highly scalable, allowing transport companies to easily add or remove vehicles and drivers as their business needs change. In addition to these core features, the TMS also includes a range of other features that are designed to improve the efficiency and effectiveness of the transport management process. This includes features such as customer invoicing, which allows companies to generate invoices for their customers automatically, and reporting and analytics, which provides companies with valuable insights into their operations. Overall, the Transport Management System project represents a significant improvement over traditional paper-based systems, which can be slow, error-prone, and difficult to manage. By implementing this system, transport companies can automate many of their daily tasks, reducing the risk of errors and improving the efficiency of their operations. This can help them to save time and money, while also improving the quality of their service and enhancing their reputation among customers.

FUTURE WORK

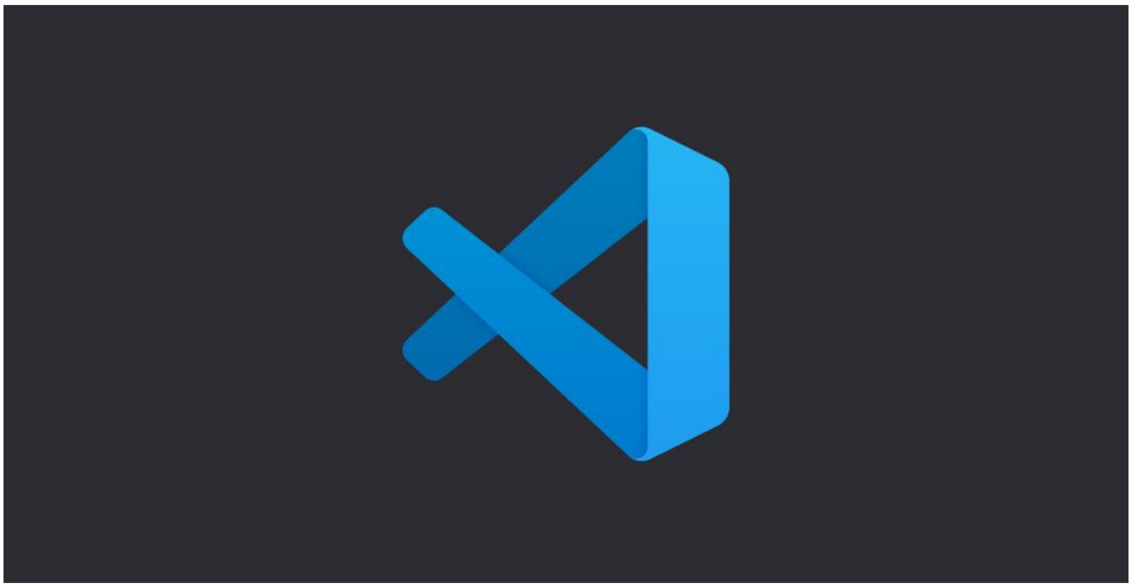
There are several potential future directions for the development of the Transport Management System (TMS). Here are some possible areas of focus for future work:

1. Integration with IoT: The TMS could be integrated with Internet of Things (IoT) devices, such as GPS sensors, to provide real-time information about vehicle location, fuel consumption, and driver behavior. This could help transport companies to optimize their operations further and reduce costs.
2. Machine learning: The TMS could be enhanced with machine learning algorithms that could analyze data from vehicle tracking, trip scheduling, and driver management to identify patterns and make predictions with the help of PSO Algorithm. This could help transport companies to make more informed decisions about their operations and improve their overall efficiency.
3. Mobile application: A mobile application for the TMS could be developed to allow drivers and customers to access information about trips, schedules, and invoices on-the-go. This could improve communication and collaboration between drivers and transport companies, as well as improve customer satisfaction.
4. Blockchain technology: The TMS could be integrated with blockchain technology to provide secure and transparent tracking of shipments and invoices. This could help transport companies to reduce fraud and increase transparency.
5. Integration with other systems: The TMS could be integrated with other systems, such as warehouse management systems or customer relationship management systems, to provide a more comprehensive view of the transport management process. This could help transport companies to optimize their entire supply chain and improve their overall efficiency.
6. Predictive maintenance: The TMS could be enhanced with predictive maintenance features that could analyze data from sensors on vehicles to identify potential maintenance issues before they occur. This could help transport companies to reduce downtime and maintenance costs.

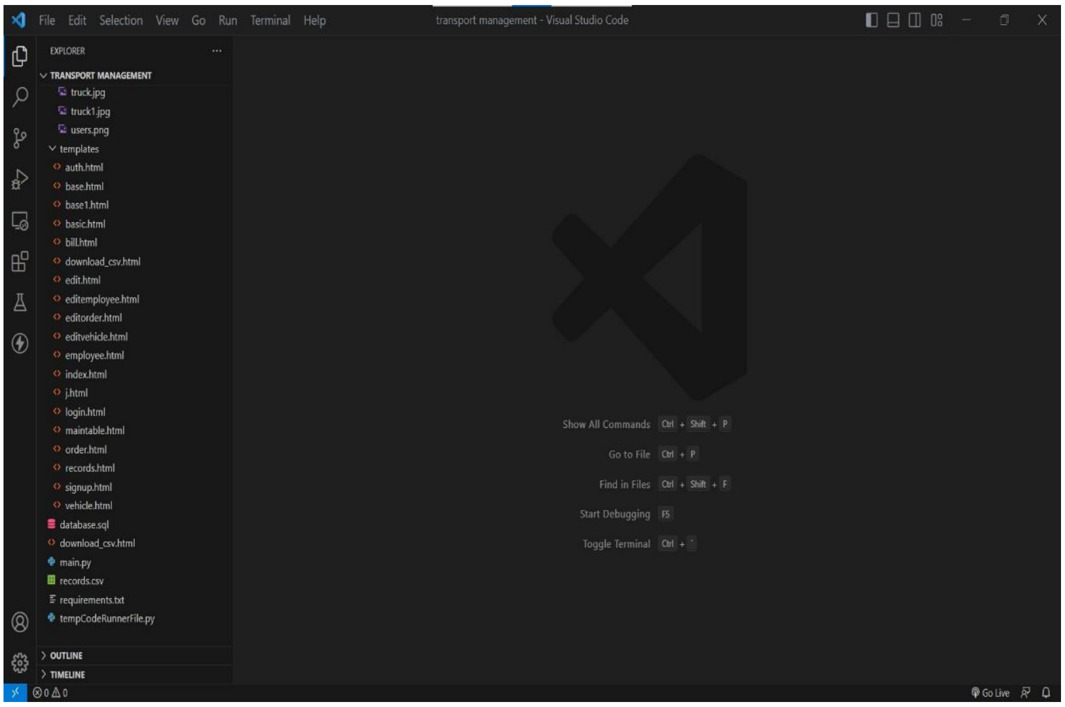
7. Autonomous vehicles: As autonomous vehicle technology advances, the TMS could be integrated with autonomous vehicles to provide real-time tracking and management of these vehicles. This could improve safety and reduce costs by eliminating the need for human drivers.
8. Advanced analytics: The TMS could be enhanced with advanced analytics tools that could provide deeper insights into transport operations. For example, the system could use data analytics to identify the most efficient routes, the most reliable vehicles, and the most productive drivers.
9. Customer portal: The TMS could be enhanced with a customer portal that would allow customers to view real-time information about their shipments, including the current location of their goods, expected delivery times, and any delays or issues that arise.
10. Integration with e-commerce platforms: The TMS could be integrated with e-commerce platforms to provide a seamless customer experience from the time an order is placed to the time it is delivered. This could improve customer satisfaction and increase repeat business.

User Manual

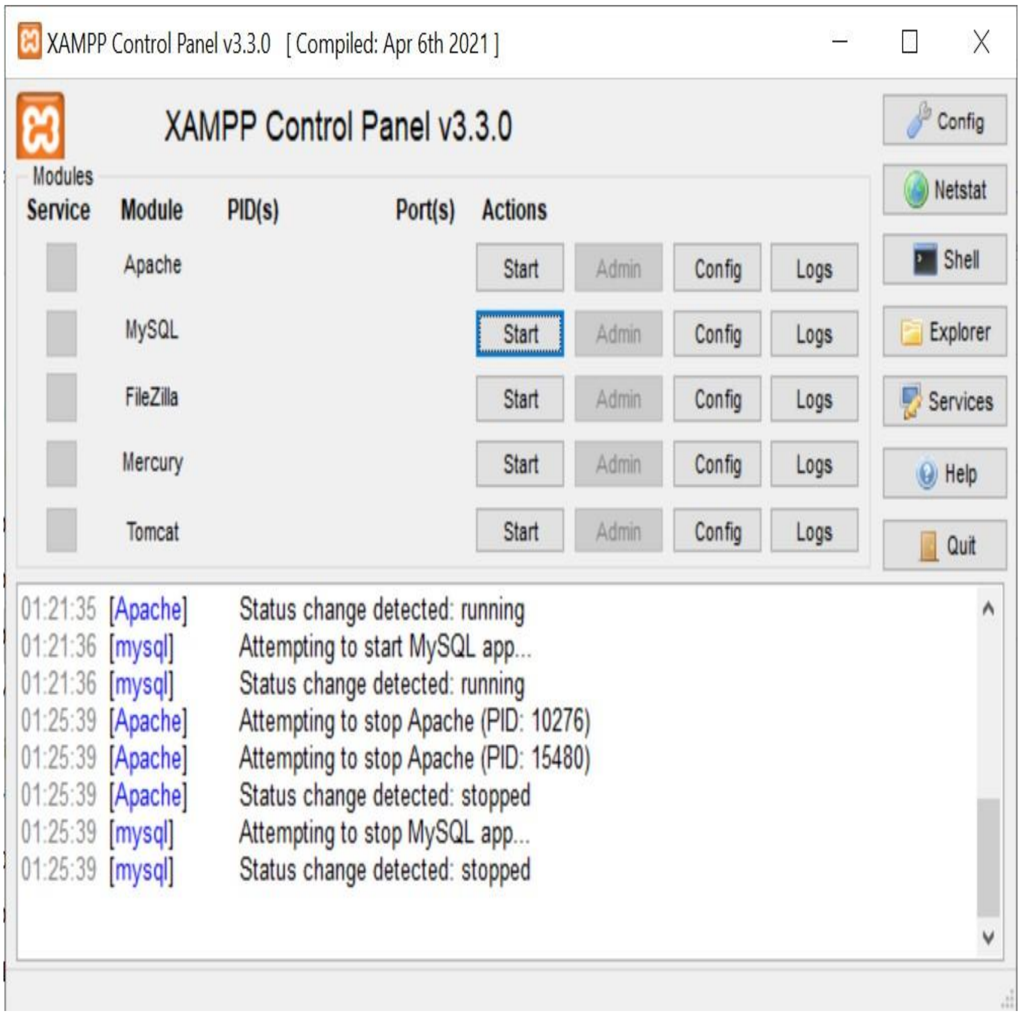
1. Open the Visual Studio Code for opening the project.



2. Open the folder in which web application files are stored.

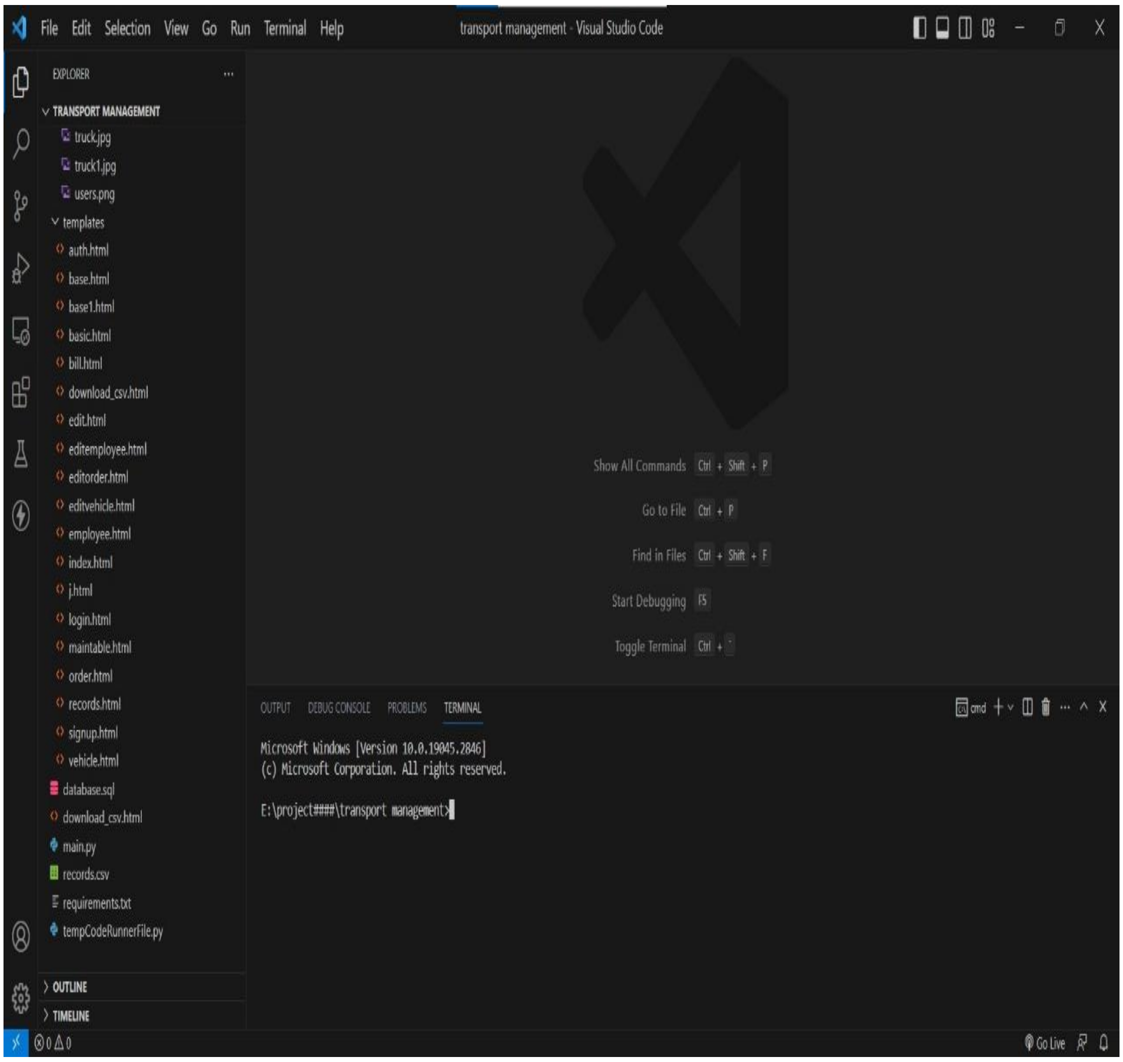


3. Open the XAMPP Control panel and start Apache and MySQL module.



```
01:21:35  [Apache]   Status change detected: running
01:21:36  [mysql]    Attempting to start MySQL app...
01:21:36  [mysql]    Status change detected: running
01:25:39  [Apache]   Attempting to stop Apache (PID: 10276)
01:25:39  [Apache]   Attempting to stop Apache (PID: 15480)
01:25:39  [Apache]   Status change detected: stopped
01:25:39  [mysql]    Attempting to stop MySQL app...
01:25:39  [mysql]    Status change detected: stopped
```

4. Click on the Terminal and select New Terminal



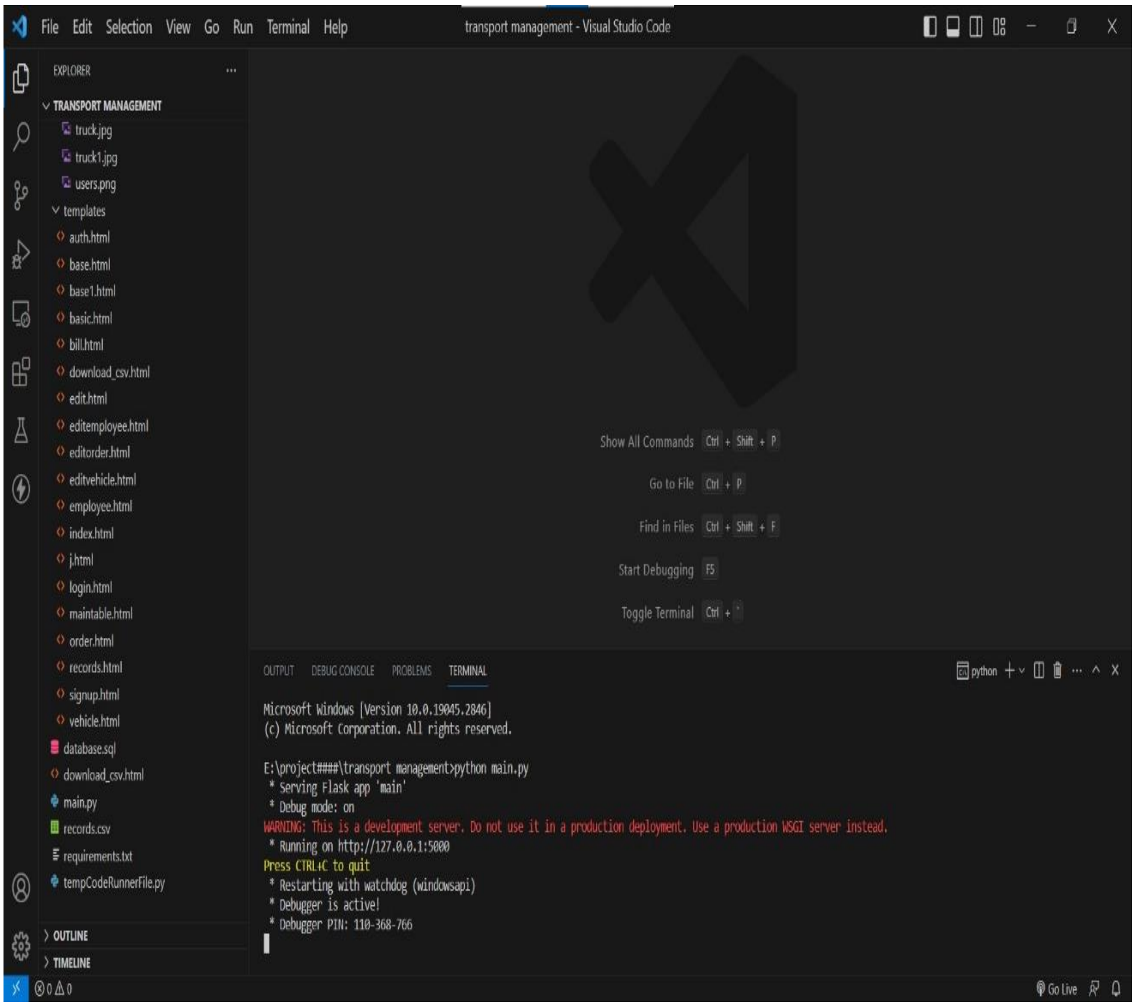
```
Microsoft Windows [Version 10.0.19045.2846]
(c) Microsoft Corporation. All rights reserved.

E:\project####\transport management>
```

5. On the terminal, type

python main.py

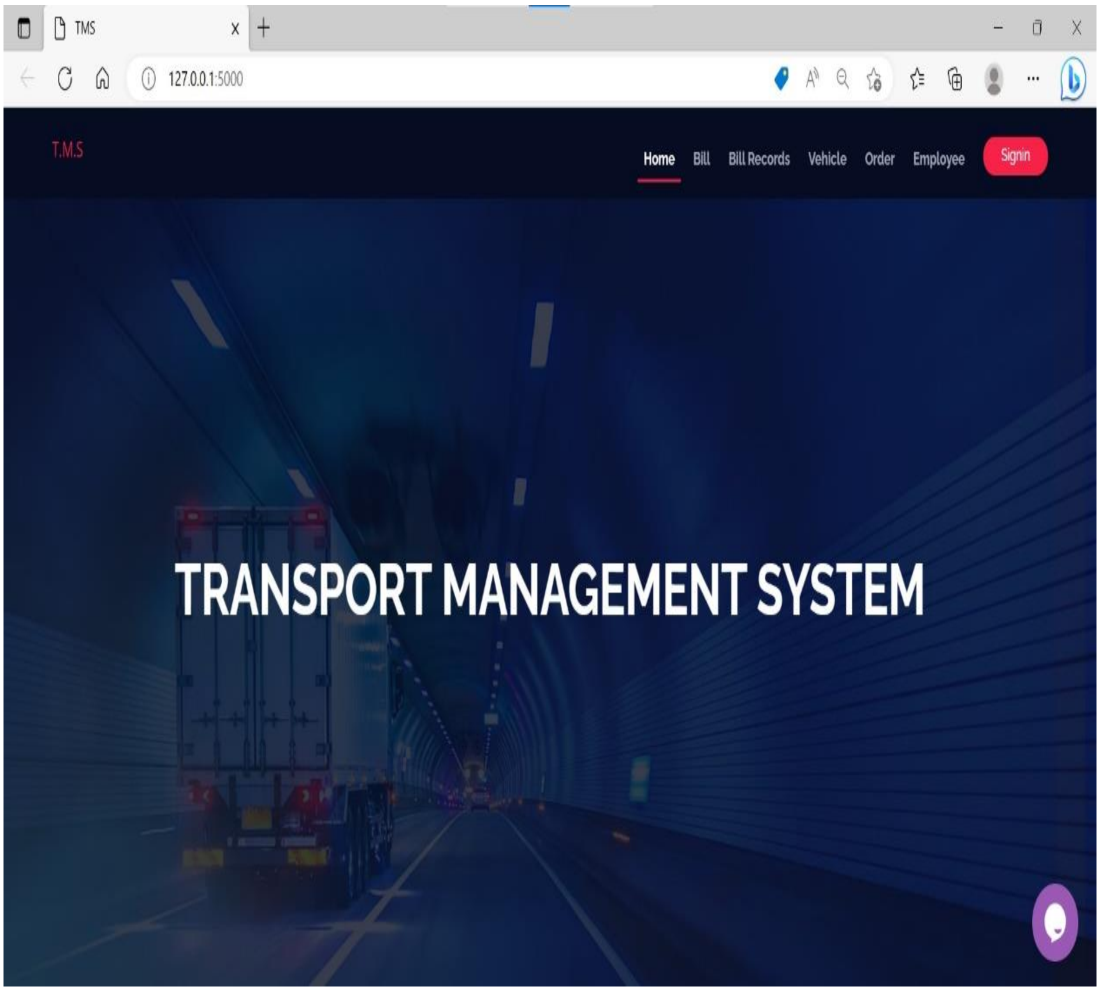
You will get following output with link to open web application on local host.



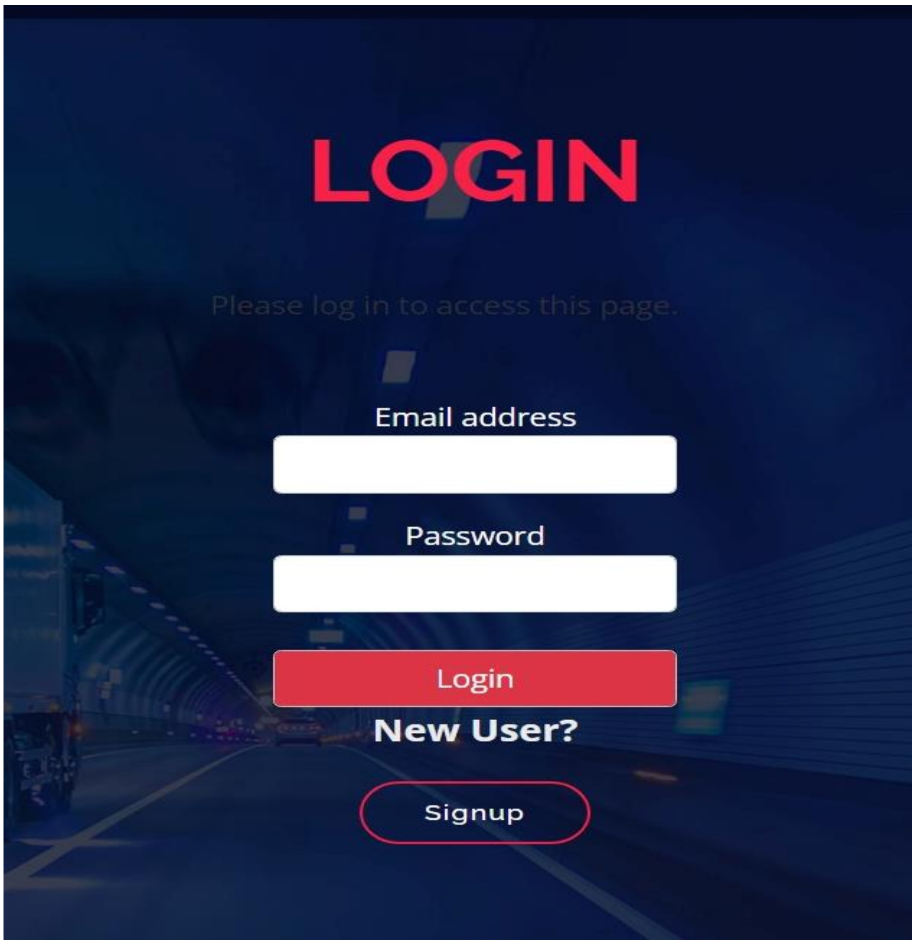
```
Microsoft Windows [Version 10.0.19045.2846]
(c) Microsoft Corporation. All rights reserved.

E:\project####\transport management>python main.py
 * Serving Flask app 'main'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (windowsapi)
 * Debugger is active!
 * Debugger PIN: 110-368-766
```

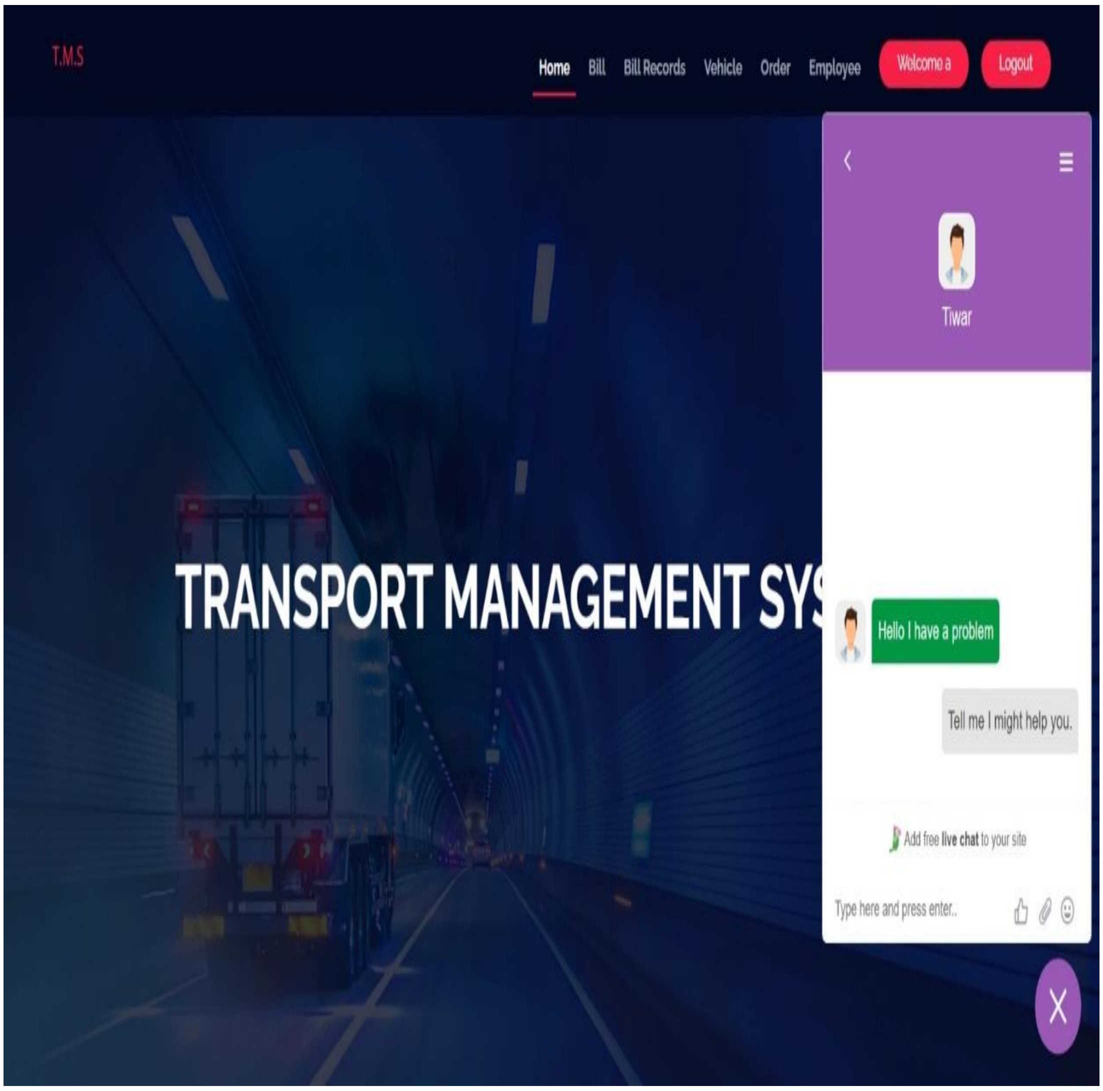
6. After clicking and opening link in web browser, you can use your web application



7. Click on sign in and you will see this window. Add your email address and password to login to the system.



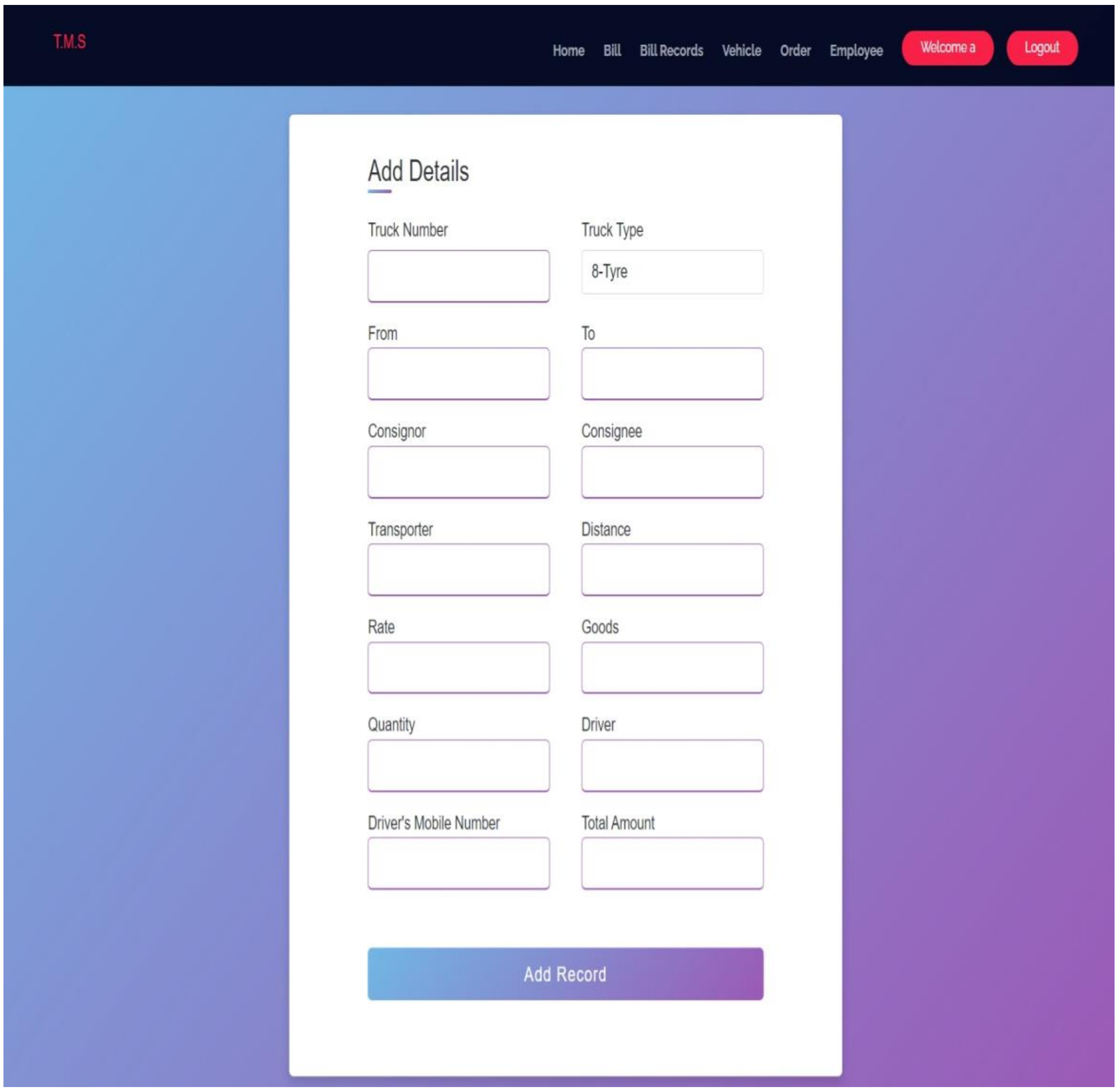
8. You will see the chat symbol in the bottom right of the homepage. Admin can click on it and start a conversation with the employees (drivers). This is important because employees may have queries and so they can write their queries here and admin can give his response and solve their queries.



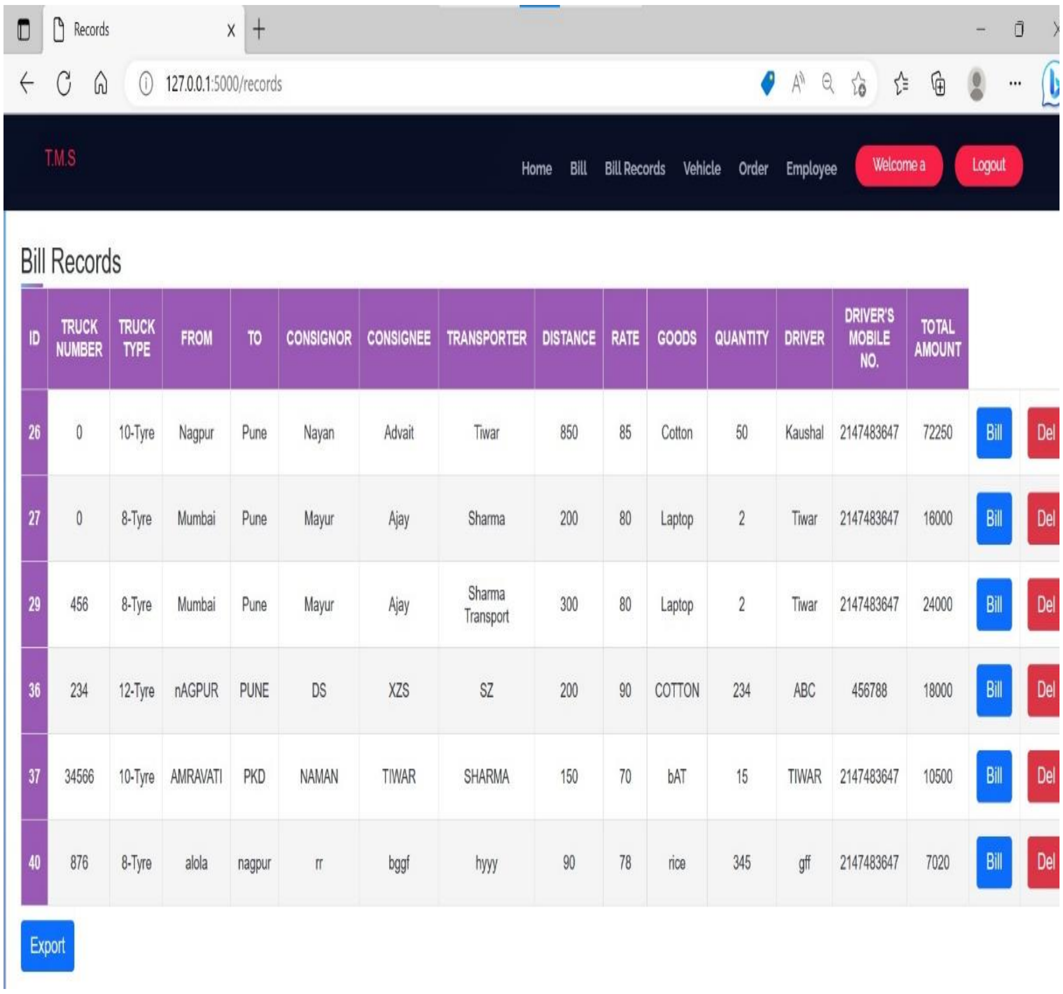
Hello I have a problem

Tell me I might help you.

9. After clicking on Bill you will see the following page. You can add bill details in the fields and save it to the database.



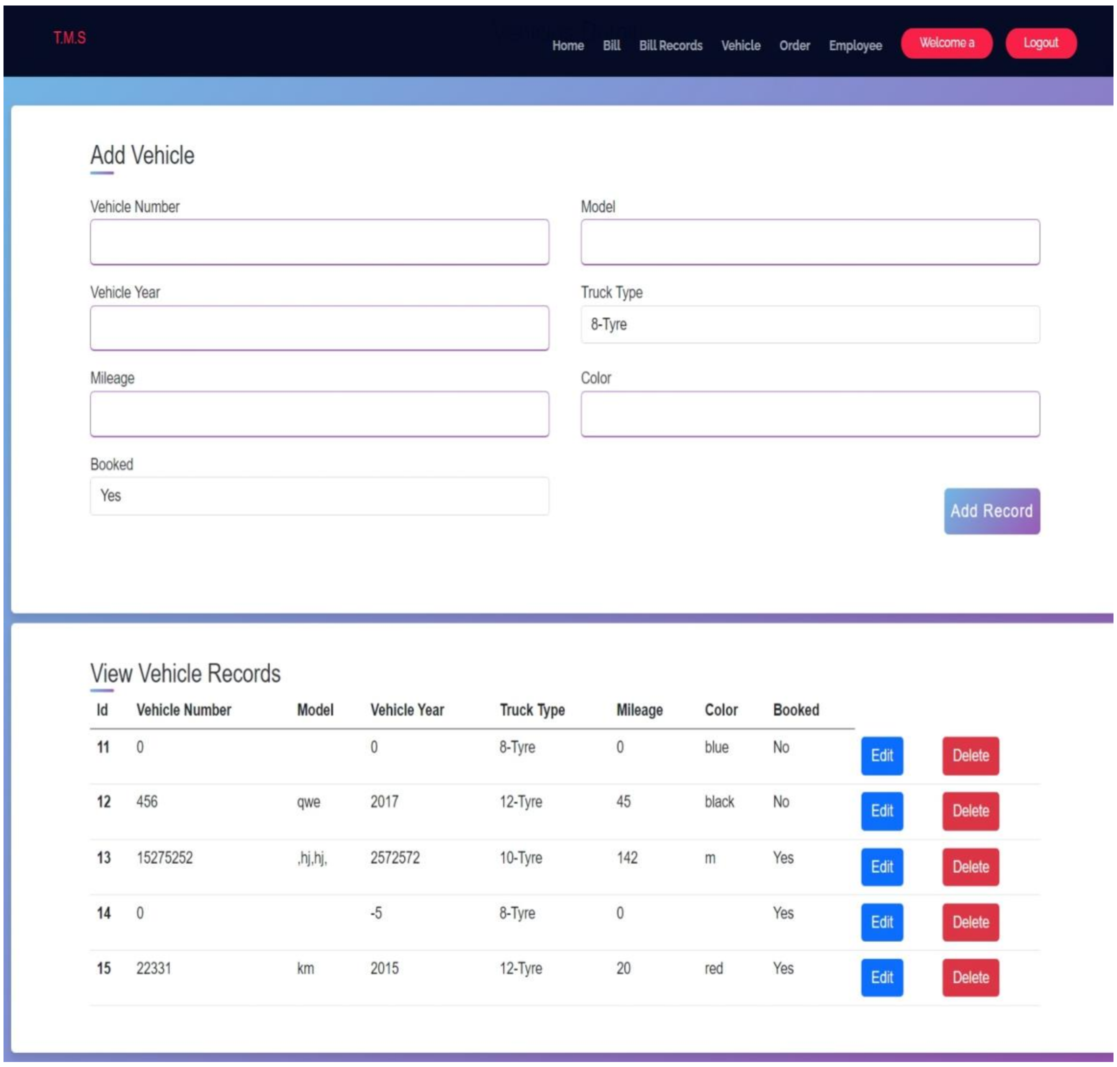
10. After clicking “Bill Records” you will see the following page. You can print bill by clicking on the Bill button and delete the record by clicking the delete button. If you want to get the complete record in ‘csv’ file you can click on “Import”.



Bill Records

ID	TRUCK NUMBER	TRUCK TYPE	FROM	TO	CONSIGNOR	CONSIGNEE	TRANSPORTER	DISTANCE	RATE	GOODS	QUANTITY	DRIVER	DRIVER'S MOBILE NO.	TOTAL AMOUNT		
26	0	10-Tyre	Nagpur	Pune	Nayan	Advait	Tiwar	850	85	Cotton	50	Kaushal	2147483647	72250	Bill	Del
27	0	8-Tyre	Mumbai	Pune	Mayur	Ajay	Sharma	200	80	Laptop	2	Tiwar	2147483647	16000	Bill	Del
29	456	8-Tyre	Mumbai	Pune	Mayur	Ajay	Sharma Transport	300	80	Laptop	2	Tiwar	2147483647	24000	Bill	Del
36	234	12-Tyre	nAGPUR	PUNE	DS	XZS	SZ	200	90	COTTON	234	ABC	456788	18000	Bill	Del
37	34566	10-Tyre	AMRAVATI	PKD	NAMAN	TIWAR	SHARMA	150	70	bAT	15	TIWAR	2147483647	10500	Bill	Del
40	876	8-Tyre	alola	nagpur	rr	bggf	hyyy	90	78	rice	345	gff	2147483647	7020	Bill	Del

11. After selecting "Vehicle' you will see the following page. You can add vehicle details and also edit and delete the record.



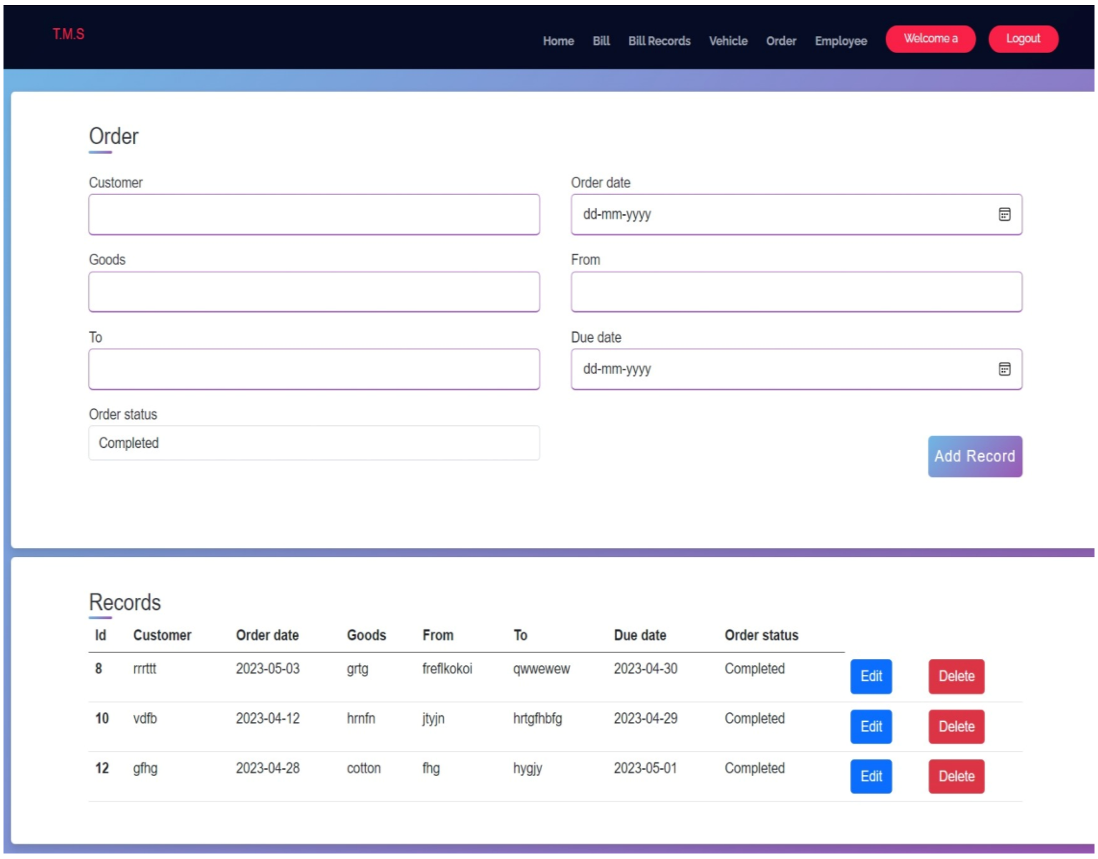
Add Vehicle

Vehicle Number	Model
Vehicle Year	Truck Type: 8-Tyre
Mileage	Color
Booked: Yes	

View Vehicle Records

Id	Vehicle Number	Model	Vehicle Year	Truck Type	Mileage	Color	Booked
11	0		0	8-Tyre	0	blue	No
12	456	qwe	2017	12-Tyre	45	black	No
13	15275252	,hj,hj,	2572572	10-Tyre	142	m	Yes
14	0		-5	8-Tyre	0		Yes
15	22331	km	2015	12-Tyre	20	red	Yes

12. After selecting “Order’ you will see the following page. You can add order details and also edit and delete the record.



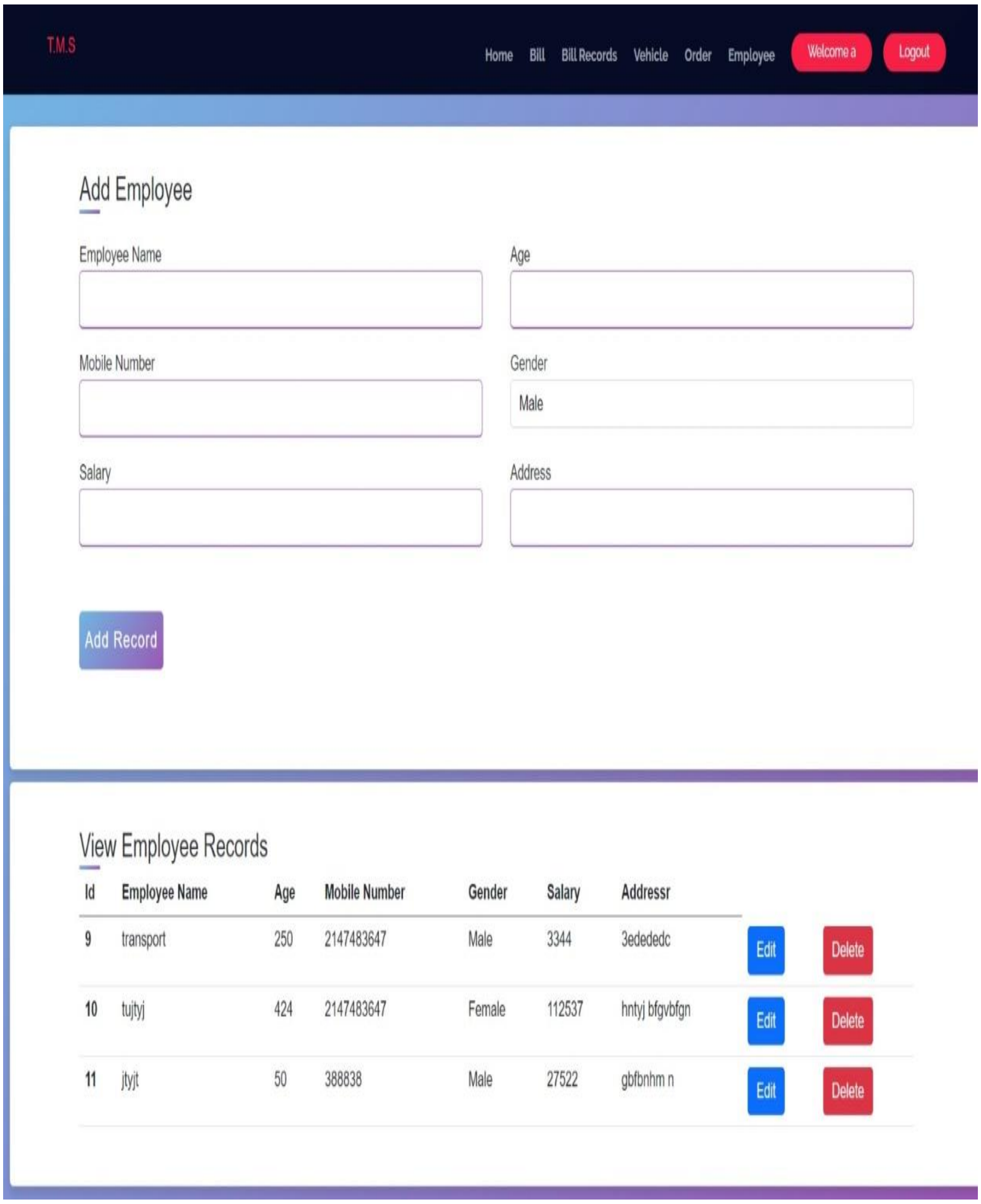
Order

Customer	Order date: dd-mm-yyyy
Goods	From
To	Due date: dd-mm-yyyy
Order status: Completed	

Records

Id	Customer	Order date	Goods	From	To	Due date	Order status		
8	rrrttt	2023-05-03	grtg	freflkokoi	qwwewew	2023-04-30	Completed	Edit	Delete
10	vdfb	2023-04-12	hrnfn	jtyjn	hrtgfhbfg	2023-04-29	Completed	Edit	Delete
12	gfhg	2023-04-28	cotton	fhg	hygjy	2023-05-01	Completed	Edit	Delete

13. After selecting "Employee' you will see the following page. You can add employee details and also edit and delete the record.



Add Employee

Employee Name	Age
Mobile Number	Gender: Male
Salary	Address

View Employee Records

Id	Employee Name	Age	Mobile Number	Gender	Salary	Addressr		
9	transport	250	2147483647	Male	3344	3edededc	Edit	Delete
10	tujtyj	424	2147483647	Female	112537	hntyj bfgvbfgn	Edit	Delete
11	jtyjt	50	388838	Male	27522	gbfbnhm n	Edit	Delete

14. You can log out of the system after clicking the “Logout” button in the navigation bar.

REFERENCES

[1] Miguel Grinberg, "Building a Simple CRUD Application with Flask and SQLAlchemy".

[2] Miguel Grinberg, "Flask Mega-Tutorial Part I: Hello, World!" .

[3] Corey Schafer, "Flask Web Development with Python Tutorial".

[4] Miguel Grinberg, "The Flask Mega-Tutorial Part VII: Error Handling".

[5] Julian Nash, “"Flask by Example – Project Setup".

[6] Dan Bader, "Building a Simple Flask Application" by Real Python.

[7] Twilio, "Build a CRUD Web App With Python and Flask - Part One".

[8] Ezequiel Delpero, "Creating a Simple Web Application Using Flask".

[9] Full Stack Python, "Building a Flask Web App with SQLite3 and Flask-SQLAlchemy".

[10] Shalabh Aggarwal, "Flask Web Development Cookbook".

[11] Flask documentation, " Flask Blueprints".

[12] Flask-JWT-Extended documentation, "Flask-JWT-Extended"

[13] AWS documentation, "Deploying a Flask Application to Elastic Beanstalk".

[14] Flask-Admin documentation, "Flask-Admin".

[15] Flask-RESTful documentation, "Flask-RESTful".

[16] Flask-Security documentation, "Flask-Security".

[17] Flask documentation, "Testing Flask Applications".

[18] Flask-Caching documentation, "Flask-Caching".

[19] Flask-Script documentation, "Flask-Script".

[20] Flask-Script documentation, "Flask-Script"

[21] Scotch.io, "Flask and AngularJS".

[22] Barzan Mozafari, Carlo Curino, and Samuel Madden, "Benchmarking MySQL and PostgreSQL on the Amazon Cloud".

[23] Tao Yang, Ramesh Illikkal, and Brian Cooper , "A Scalable and Highly Available MySQL Database Service.

[24] Andrew Morgan, Tomas Ulin,"MySQL Cluster: A Real-Time Open Source Distributed Database" .

[25] Eman AlQuraishi, Maha Alqallaf, "Performance Evaluation of MySQL and PostgreSQL Database Management Systems in a Web Server Environment".

[26] S. Srinivasan, V. Jayakumar, "A Comparative Study of MySQL and Oracle Databases".

[27] S. S. Anandhi, S. S. Sujatha, N. Nandhini, "A Comparative Study of MySQL and MS SQL Server for a Hospital Management System".

[28] Abhinav Singh, Kirti Gupta ,"A Cloud-Based Inventory Management System using MySQL Database".

[29] Cheikh Ahmadou Bamba Mbacké, Bamba Gueye ,"Design and Implementation of a Web-Based Project Management System using PHP and MySQL".

[30] Wenbing Zhao, Rong Zhang, Guoyin Wang , "A MySQL-Based Knowledge Management System for Software Development".

[31] Stanley E. Griffis, Thomas J. Goldsby, “Transportation Management Systems: An Exploration of Progress and Future Prospects”, Journal of Transportation Management

[32] https://www.techtarget.com/searcherp/definition/transportation-management-system-TMS

[33] https://mercurygate.com/blog-posts/how-to-know-when-the-limitations-of-tms-are-hurting-more-than-helping/#:~:text=Users%20Cannot%20Access%20Data%20in,data%20can%20have%20disastrous%20consequences.

[34] https://balloonone.com/blog/2019/07/23/the-future-of-tms/

[35] Abid Jamil, Muhammad Arif, Akhlaq Ahmad, “Software Testing Techniques: A Literature Review”, 2016 6th International Conference on Information and Communication Technology for The Muslim World.

[36] Jonas Kjaer Rask, Frederik Palludan Madsen, Nick Battle, Hugo Daniel Macedo, “Visual Studio Code VDM Support”, The 18th Overture Workshop, 07/12/2020

[37] https://www.javatpoint.com/uml-deployment-diagram

DISSEMINATION OF WORK

Title - Survey of Particle Swarm Optimization Algorithm to optimize energy in modern data centers

Author – Nayan Sonkalyari, Advait Patil, Tiear Zade, Kaushal Sharma, Dr. A. S. Manekar.

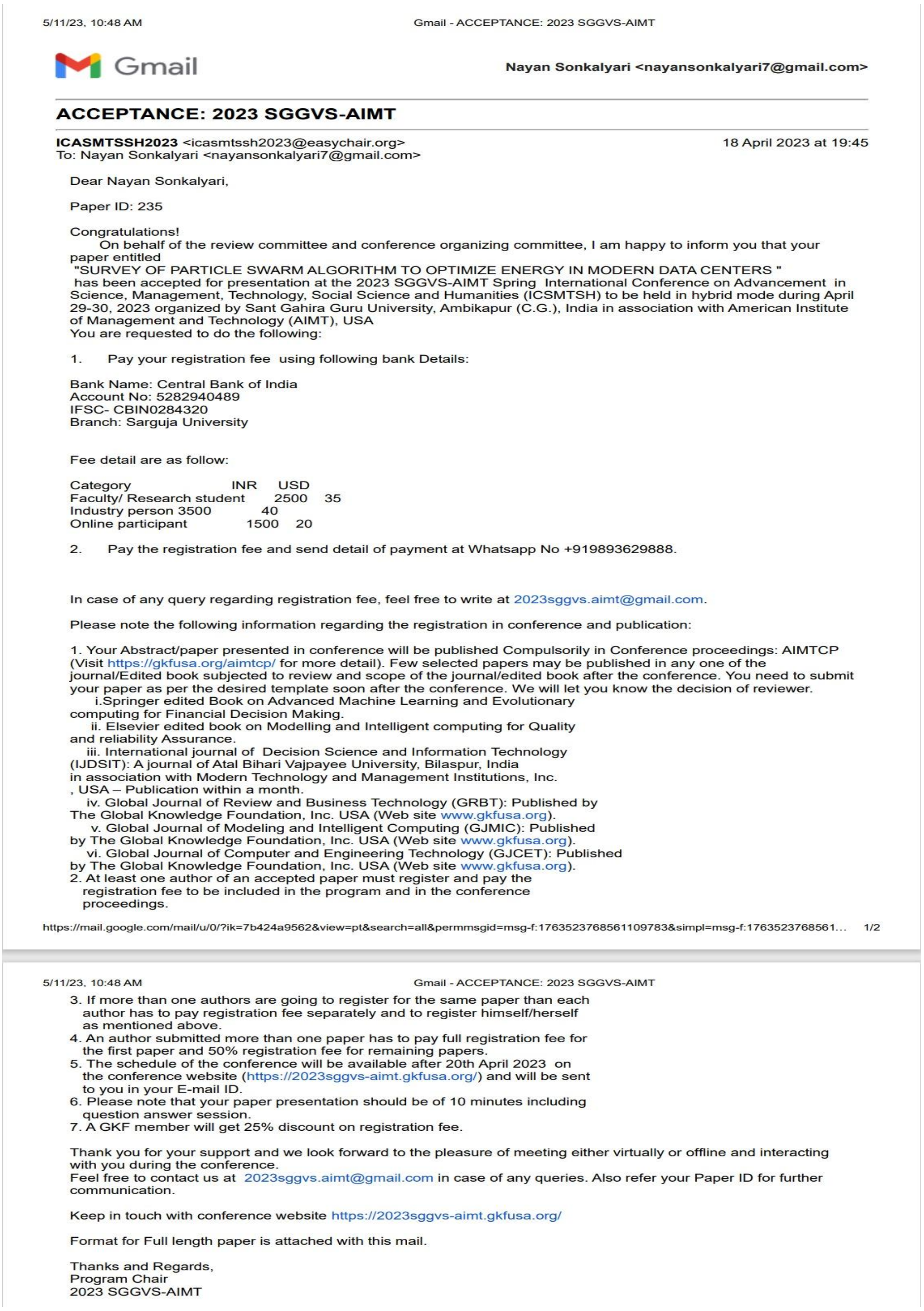
Publisher - SGGVS-AIMT Spring International Conference on Advancement in Science, Management, Technology, Social Science and Humanities (ICSMTSH)

ISSN No -

DOI -

Issue –

Acceptance Letter



5/11/23, 10:48 AM Gmail - ACCEPTANCE: 2023 SGGVS-AIMT

Gmail

Nayan Sonkalyari <nayansonkalyari7@gmail.com>

ACCEPTANCE: 2023 SGGVS-AIMT

ICASMTSSH2023 <icasmtssh2023@easychair.org> 18 April 2023 at 19:45
To: Nayan Sonkalyari <nayansonkalyari7@gmail.com>

Dear Nayan Sonkalyari,

Paper ID: 235

Congratulations!
On behalf of the review committee and conference organizing committee, I am happy to inform you that your paper entitled
"SURVEY OF PARTICLE SWARM ALGORITHM TO OPTIMIZE ENERGY IN MODERN DATA CENTERS "
has been accepted for presentation at the 2023 SGGVS-AIMT Spring International Conference on Advancement in Science, Management, Technology, Social Science and Humanities (ICSMTSH) to be held in hybrid mode during April 29-30, 2023 organized by Sant Gahira Guru University, Ambikapur (C.G.), India in association with American Institute of Management and Technology (AIMT), USA
You are requested to do the following:

1. Pay your registration fee using following bank Details:

Bank Name: Central Bank of India
Account No: 5282940489
IFSC- CBIN0284320
Branch: Sarguja University

Fee detail are as follow:

Category	INR	USD
Faculty/ Research student	2500	35
Industry person	3500	40
Online participant	1500	20

2. Pay the registration fee and send detail of payment at Whatsapp No +919893629888.

In case of any query regarding registration fee, feel free to write at 2023sggvs.aimt@gmail.com.

Please note the following information regarding the registration in conference and publication:

1. Your Abstract/paper presented in conference will be published Compulsorily in Conference proceedings: AIMTCP (Visit https://gkfusa.org/aimtcp/ for more detail). Few selected papers may be published in any one of the journal/Edited book subjected to review and scope of the journal/edited book after the conference. You need to submit your paper as per the desired template soon after the conference. We will let you know the decision of reviewer.
i.Springer edited Book on Advanced Machine Learning and Evolutionary computing for Financial Decision Making.
ii. Elsevier edited book on Modelling and Intelligent computing for Quality and reliability Assurance.
iii. International journal of Decision Science and Information Technology (IJDSIT): A journal of Atal Bihari Vajpayee University, Bilaspur, India in association with Modern Technology and Management Institutions, Inc. , USA – Publication within a month.
iv. Global Journal of Review and Business Technology (GRBT): Published by The Global Knowledge Foundation, Inc. USA (Web site www.gkfusa.org).
v. Global Journal of Modeling and Intelligent Computing (GJMIC): Published by The Global Knowledge Foundation, Inc. USA (Web site www.gkfusa.org).
vi. Global Journal of Computer and Engineering Technology (GJCET): Published by The Global Knowledge Foundation, Inc. USA (Web site www.gkfusa.org).
2. At least one author of an accepted paper must register and pay the registration fee to be included in the program and in the conference proceedings.

https://mail.google.com/mail/u/0/?ik=7b424a9562&view=pt&search=all&permmsgid=msg-f:1763523768561109783&simpl=msg-f:1763523768561… 1/2

5/11/23, 10:48 AM Gmail - ACCEPTANCE: 2023 SGGVS-AIMT

3. If more than one authors are going to register for the same paper than each author has to pay registration fee separately and to register himself/herself as mentioned above.
4. An author submitted more than one paper has to pay full registration fee for the first paper and 50% registration fee for remaining papers.
5. The schedule of the conference will be available after 20th April 2023 on the conference website (https://2023sggvs-aimt.gkfusa.org/) and will be sent to you in your E-mail ID.
6. Please note that your paper presentation should be of 10 minutes including question answer session.
7. A GKF member will get 25% discount on registration fee.

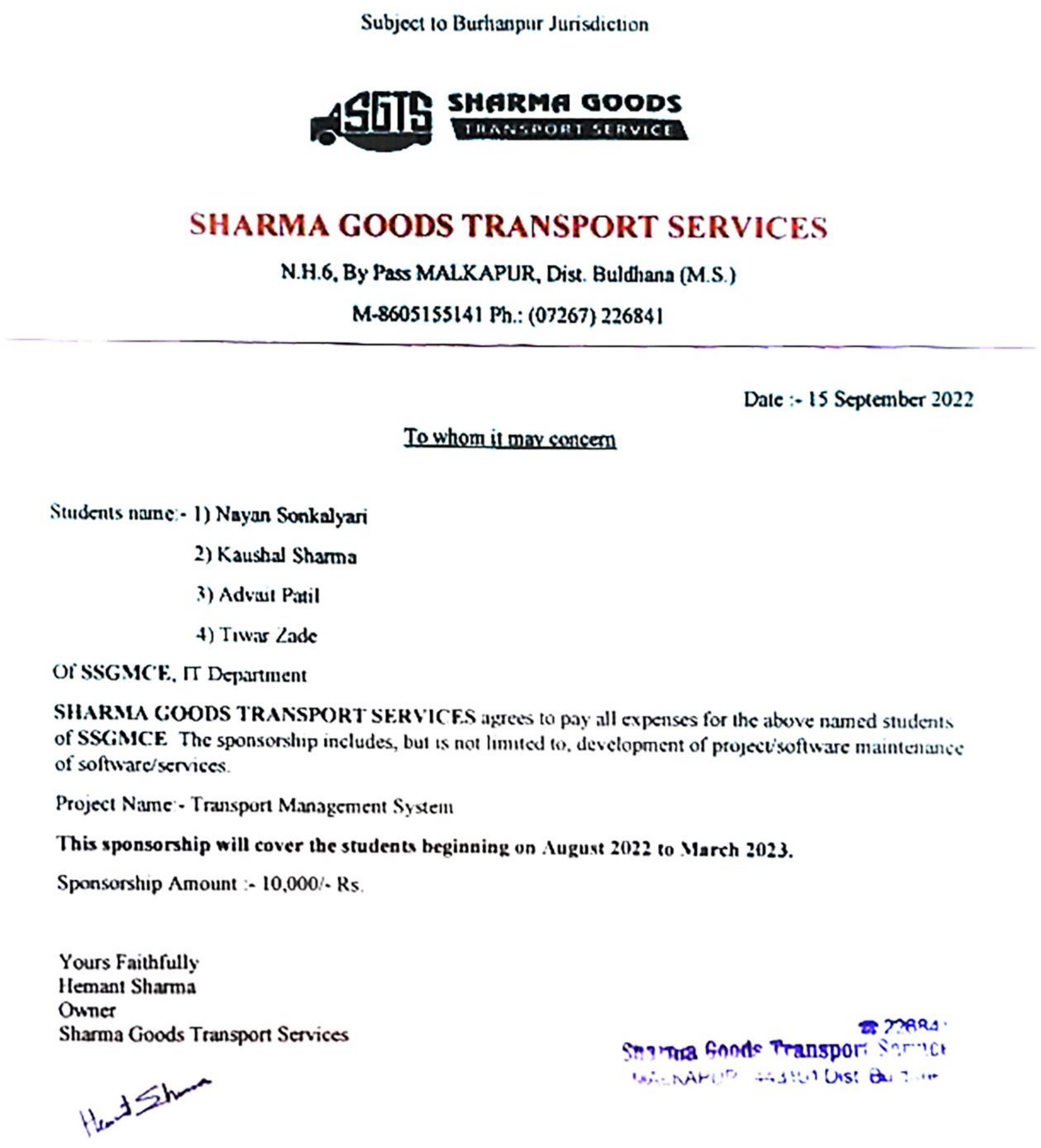
Thank you for your support and we look forward to the pleasure of meeting either virtually or offline and interacting with you during the conference.
Feel free to contact us at 2023sggvs.aimt@gmail.com in case of any queries. Also refer your Paper ID for further communication.

Keep in touch with conference website https://2023sggvs-aimt.gkfusa.org/

Format for Full length paper is attached with this mail.

Thanks and Regards,
Program Chair
2023 SGGVS-AIMT

Certificate of Sponsorship



Subject to Burhanpur Jurisdiction

SHARMA GOODS TRANSPORT SERVICE

SHARMA GOODS TRANSPORT SERVICES

N.H.6, By Pass MALKAPUR, Dist. Buldhana (M.S.)

M-8605155141 Ph.: (07267) 226841

Date :- 15 September 2022

To whom it may concern

Students name:- 1) Nayan Sonkalyari

2) Kaushal Sharma

3) Advait Patil

4) Tiwar Zade

Of SSGMCE, IT Department

SHARMA GOODS TRANSPORT SERVICES agrees to pay all expenses for the above named students of **SSGMCE**. The sponsorship includes, but is not limited to, development of project/software maintenance of software/services.

Project Name:- Transport Management System

This sponsorship will cover the students beginning on August 2022 to March 2023.

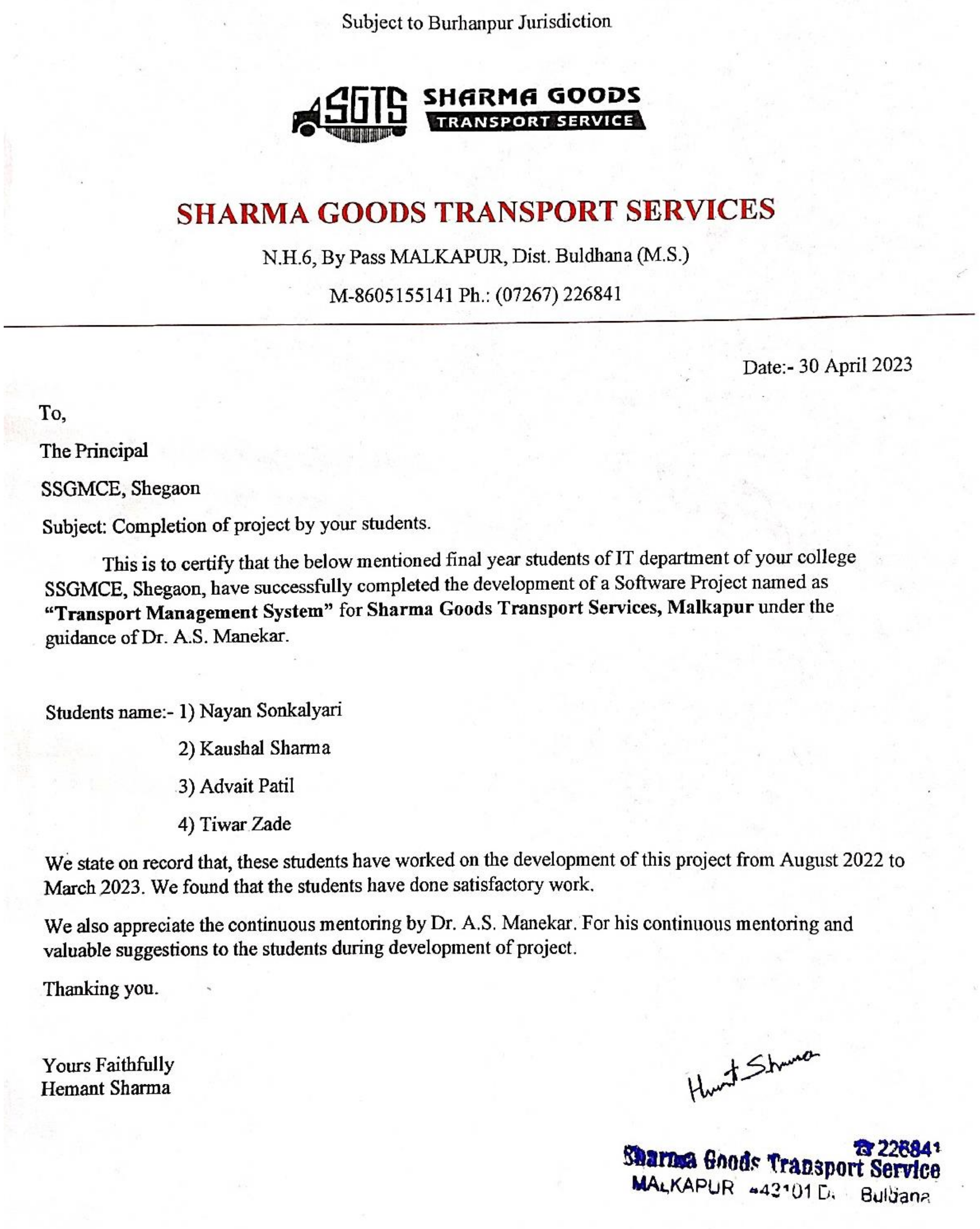
Sponsorship Amount :- 10,000/- Rs.

Yours Faithfully
Hemant Sharma
Owner
Sharma Goods Transport Services

☎ 22684[illegible]
Sharma Goods Transport Service
[illegible]KAPUR [illegible] Dist. Bu[illegible]

[illegible]

Certificate of Completion



Subject to Burhanpur Jurisdiction

SHARMA GOODS TRANSPORT SERVICE

SHARMA GOODS TRANSPORT SERVICES

N.H.6, By Pass MALKAPUR, Dist. Buldhana (M.S.)

M-8605155141 Ph.: (07267) 226841

Date:- 30 April 2023

To,

The Principal

SSGMCE, Shegaon

Subject: Completion of project by your students.

This is to certify that the below mentioned final year students of IT department of your college SSGMCE, Shegaon, have successfully completed the development of a Software Project named as **"Transport Management System"** for **Sharma Goods Transport Services, Malkapur** under the guidance of Dr. A.S. Manekar.

Students name:- 1) Nayan Sonkalyari

2) Kaushal Sharma

3) Advait Patil

4) Tiwar Zade

We state on record that, these students have worked on the development of this project from August 2022 to March 2023. We found that the students have done satisfactory work.

We also appreciate the continuous mentoring by Dr. A.S. Manekar. For his continuous mentoring and valuable suggestions to the students during development of project.

Thanking you.

Yours Faithfully
Hemant Sharma

Hmnt Sharma

Sharma Goods Transport Service ☎ 226841
MALKAPUR 443101 Dt. Buldana

Source code Listings

1. Open the Visual Studio Code for opening the project.
2. Open the folder in which web application files are stored.
3. Open the XAMPP Control panel and start Apache and MySQL module.
4. Click on the Terminal and select New Terminal
5. On the terminal, type

```
python main.py
```

You will get following output with link to open web application on local host

6. After clicking and opening link in web browser, you can use your web application
7. Click on sign in and you will see this window. Add your email address and password to login to the system.
8. You will see the chat symbol in the bottom right of the homepage. Admin can click on it and start a conversation with the employees (drivers). This is important because employees may have queries and so they can write their queries here and admin can give his response and solve their queries.
9. After clicking on Bill you will see the following page. You can add bill details in the fields and save it to the database.
10. After clicking "Bill Records" you will see the following page. You can print bill by clicking on the Bill button and delete the record by clicking the delete button. If you want to get the complete record in 'csv' file you can click on "Import".
11. After selecting "Vehicle' you will see the following page. You can add vehicle details and also edit and delete the record.
12. After selecting "Order' you will see the following page. You can add order details and also edit and delete the record.
13. After selecting "Employee' you will see the following page. You can add employee details and also edit and delete the record.
14. You can log out of the system after clicking the "Logout" button in the navigation bar.

INFORMATION OF MEMBERS

Name: Nayan Sonkalyari

Email: nayansonkalyari7@gmail.com

Mobile: 9307783470

Address: Sai Colony, Kudwa, Gondia(MS) - 441614

Name: Advait Patil

Email: patiladwait19@gmail.com

Mobile: 9763858134

Address: At Saywani, Post Channi, Ta Patur, Dis Akola

Name: Tiwar Zade

Email: tiwarzade111@gmail.com

Mobile: 9307587087

Address: Shyam Nagri 1,Pandharkawada

Name: Kaushal Sharma

Email: kaushalsharmakhs@gmail.com

Mobile: 8329821835

Address: Dwarka Nagar, Behind bus stand, Malkapur 443101 Dist: Buldhana