

Cloud Based Cab Booking Service

A major project report submitted in partial fulfillment of the requirement
for the award of degree of

Bachelor of Technology

in

Computer Science & Engineering / Information Technology

Submitted by

Khushi (201238)

Nikita Sehgal (201277)

Under the guidance & supervision of

Dr. Anita



**Department of Computer Science & Engineering and
Information Technology**

**Jaypee University of Information Technology, Wagnaghat,
Solan - 173234 (India)**

Certificate

This is to certify that the work reported in the B-Tech. project entitled “**Cloud Based Cab Booking Service**” submitted by **Khushi** and **Nikita Sehgal** at **Jaypee University of Information Technology, Wagnaghat, India**, is a bonafide record of their original work carried out under my supervision. This work has not been submitted elsewhere for any other degree or diploma.

Dr.Anita

Computer Science Engineering

Jaypee University of Information Technology

Dated:

Candidate's Declaration

We hereby declare that the work presented in this report entitled '**Cloud Based Cab Booking Service**' in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science & Engineering / Information Technology** submitted in the Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat is an authentic record of our own work carried out over a period from August 2023 to May 2024 under the supervision of **Dr. Anita**(Assistant Professor(SG), Department of Computer Science & Engineering and Information Technology).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Student Name: Nikita Sehgal

Roll No.: 201277

Student Name: Khushi

Roll No.: 201238

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Supervisor Name: Dr. Anita

Designation: Assistant Professor(SG)

Department: CS and IT

Dated:

Acknowledgement

We would like to express my deepest appreciation to Dr Anita for helping us throughout the project and without whom this project would have been a very difficult task. We are highly indebted to ma'am for her guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in doing project. She consistently motivated and guided us towards the completion of the project. We would like to express our gratitude towards my parents & members of JUIT for their kind co-operation and encouragement which helped me in doing this project. Our thanks and appreciations also go to our colleagues who have helped us out with their abilities in developing the project.

Table Of Contents

Topic	Page Number
Certificate	i
Declaration	ii
Acknowledgements	iii
Table of Contents	iv
List of Tables	vi
List of Figures	vii
List of Abbreviations, Symbols or Nomenclature	ix
Abstract	x
Chapter 1: Introduction	1
1.1 Introduction	1
1.2 Problem Statement	2
1.3 Objectives	2
1.4 Significance and Motivation of the Project Work	3
1.5 Organization of Project Report	5
Chapter 2: Literature Survey	6
2.1 Overview of Relevant Literature	6
2.2 Key Gaps in the Literature	8
Chapter 3: System Development	12
3.1 Requirements and Analysis	12
3.1.1 Functional Requirements	12
3.1.2 Non- Functional Requirements	13
3.1.3 System Constraints	14
3.2 Project Design and Architecture	18
3.2.1 Activity Diagram	22
3.2.2 Block Diagram	22
3.3 Data Preparation	23
3.4 Implementation	29

3.4.1 Backend Implementation	29
3.4.2 Algorithm For Booking Process	37
3.4.3 Frontend Implementation	39
3.4.4 Deployment on AWS EC2	40
3.5 Key Challenges	45
Chapter 4: Testing	47
4.1 Testing Strategy	47
4.2 Test Cases and Outcomes	48
Chapter 5: Results and Evaluation	52
5.1 Results	52
5.2 Comparison With Existing Solutions	56
Chapter 6: Conclusions and Future Scope	59
6.1 Conclusion	59
6.2 Future Scope	60
REFERENCES	63
PLAGIARISM REPORT	

List of Tables

S.No.	Table Name	Page No.
Table 1	Tabular form of Literature Review	9
Table 2	`account`	25
Table 3	`route`	26
Table 4	`order`	26
Table 5	`car_details`	27
Table 6	`car`	27
Table 7	`order_has_car`	27
Table 8	`language`	28
Table 9	`account_has_order`	28
Table 10	`car_has_language`	28

List of Figures

S.No.	Name Of Figure	Page No.
Figure 1	Growth Of Cab Booking Industry	7
Figure 2	Project Design	18
Figure 3	MVC Architecture	20
Figure 4	Activity diagram	22
Figure 5	Block Diagram	23
Figure 6	Database Model	25
Figure 7	Database Connection Initialization	30
Figure 8	MySQL Database Configuration for Taxi	30
Figure 9	Database In Apache Netbeans	30
Figure 10	Implementation Of Md5	31
Figure 11	Data Validation Code	32
Figure 12	Pagination Code	33
Figure 13	CommandContainer.java Code	34
Figure 14	Tomcat Server Configuration	38
Figure 15	Multiple Language Supporting File	39
Figure 16	Creating An Instance	40
Figure 17	Allocating Elastic IP Address	40
Figure 18	Connecting Local Machine With Ubuntu On Cloud	41

Figure 19	Connecting To FileZilla	42
Figure 20	Creating A War File	42
Figure 21	Connecting To Mysql	43
Figure 22	Databases On Cloud	44
Figure 23	Setting Inbound Rules	44
Figure 24	Testing Files	50
Figure 25	Testing Results	51
Figure 26	Homepage	53
Figure 27	Homepage With Different Language	53
Figure 28	Signup page	54
Figure 29	Login Page	54
Figure 30	Booking Page	55
Figure 31	Order Detail Page	55
Figure 32	Account History With Filters	56
Figure 33	Booked Car Details	56
Figure 34	Cab	58

List of Abbreviations, Symbols or Nomenclature

ABBREVIATION	DEFINITION
MD5	Message Digest Algorithm 5
MVC	Model-view controller
AWS	Amazon Web Services
EC2	Elastic Compute Cloud
Java EE	Java Platform, Enterprise Edition
CRM	Customer relationship management
GPS	Global Positioning System
JDBC	Java Database Connectivity

Abstract

The Cloud-Based Cab Booking Service is a shining example of dependability and innovation in the taxi rental industry. Through the use of state-of-the-art technology, we offer a seamless and efficient platform for taxi and car rental services across India. We have a large fleet of cars, ranging in price from luxury to budget taxis, and we are experts at providing corporate entities with online taxi booking services. Our taxis are fully licensed and insured, and we prioritize hassle-free experiences by offering competitive and personalized rates for both intra- and inter-city travel. Our experienced, polite, and reliable taxi drivers receive intensive training in handling malfunctions and prioritize the safety of their passengers. For unanticipated events, we have a backup plan in place. Overall, our Cloud-Based Cab Booking Service revolutionizes the market by skillfully fusing state-of-the-art technology, a diverse and immaculate fleet, and an unwavering dedication to providing the highest caliber of customer service. The Cloud-Based Cab Booking Service project seeks to transform the conventional taxi booking experience through the use of HTML, CSS, Java EE Servlets, and AWS infrastructure. The system uses the MVC[14] pattern to put the needs of the user first, providing secure platform with user authentication, efficient order and car management, and multilingual support. Scalability is guaranteed by the AWS deployment, and project dependability is increased by thorough testing procedures and problem documentation.

Chapter -1

Introduction

1.1 Introduction

The taxi business is not an exception to the rule that having an online presence is crucial in the current digital era. Thanks to user-friendly websites and app-based systems, it's commonplace to book a taxi conveniently whenever and wherever you want these days. These platforms offer enhanced safety and transparency during our travels, along with accessibility. Even the biggest automakers have had to review their approaches due to the disruption caused by the increase in online taxi reservations. These systems have completely redesigned urban transportation, making it safer and more efficient, with features like real-time tracking, wait time estimation, and a customer experience focus. Welcome to the modern era, where transportation and technology live in harmony and have completely transformed how we get around. Our platform is methodically structured using the MVC[14] pattern, emphasizing efficiency and best practices. This is especially true in the quickly changing field of web development. Our main goal is to provide a flawless taxi booking experience that is characterized by ease of use and simplicity in each and every user interaction. Thorough on-site testing guarantees a dependable platform prior to implementation on the scalable and dependable AWS infrastructure, providing a revolutionary approach to modernize and enhance the taxi reservation procedure. The Model-View-Controller (MVC[14]) pattern is followed in this project, which provides a scalable and well-organized taxi service management solution. The comprehensive testing suite has been created during the incubation period so that our product is fully ready before launching it on AWS which accelerate the process of scaling and helps to use reliable services at every stage. Taxi booking problem is now solved with the novel technology of AWS. Therefore, more convenient and steady way of reaching taxi from booking to until the arrival of the customers has been provided. Moving to the Model-View-Controller (MVC) approach could be a small but significant step to be taken to improve the quality of service nowadays when both

technologies and transportation markets are forever expanding and taking on new demands every day.

1.2 Problem Statement

Entering the ride sharing market can be quite challenging, with established giants like Ola and Uber already dominating the industry. One of the issues with taxi services both in urban and rural areas is their limited availability of cars and outdated reservation systems. Another hurdle is establishing a platform that fosters a connection between drivers and riders. Today's consumers expect rides to be safe personalized and environmentally friendly which poses a challenge for newcomers trying to meet these expectations. Safety is a concern, for users making it crucial for any transportation service to prioritize creating a sense of security.

The objectives, features, technology stack, constraints, timetable, important stakeholders, and success criteria for developing a Java-based taxi booking system are succinctly summarized in this problem statement. The project team refers to it as a roadmap throughout the entire development lifecycle.

1.3 Objectives

The Cloud Based Cab Booking Service Web Application aims to achieve the following objectives:

1. **Efficient Order Management:** Provide customers with the ability to choose the type of vehicle, the number of passengers, and the beginning and ending points of their trip to make placing taxi orders easier.
2. **Scalability using AWS EC2 and MVC[14]:** Use the Model-View-Controller (MVC[14]) architecture to organize code effectively. To meet increasing user demand and maintain system efficiency, leverage AWS EC2 to provide scalable and flexible infrastructure.

3. Fleet Coordination: Assign as many available vehicles as soon as possible, taking into account variables such as distance, capacity, and category, to ensure dependable and timely service.

4. Loyalty Program: Establish a plan where customers obtain savings based on their previous purchases.

5. Multilingual Support: You can better serve a diverse user base and enhance the application's usability and accessibility by providing support for multiple languages.

In order to guarantee transaction security and improve user experience, the system expedites the booking procedure and incorporates safe payment gateways. Furthermore, the system continuously gathers user feedback to enhance service quality while optimizing driver management and routing for effective service delivery. It is imperative to take into account scalability, performance, and administrative management to guarantee that the system can accommodate high user and booking volumes while adhering to regulatory requirements.

1.4 Significance And Motivation Of The Project Work

The Taxi Service Web Application holds significant value in the following aspects:

- **Enhanced User Experience:** Users benefit from a user-friendly interface for order placement and transparent pricing. The platform is a combination of magnificent views and useful architecture. It will be easy to book a flight or search the information because every feature on our platform is simple and functioning. The application minimalizes the number of efforts that the user has to do to make his opinion through the tons of options. Whether the needed transport type is one or if they should do it for all the other passengers and which points they will pick them up from and drop them off to has been defined. We proceed by establishing firstly the simplicity and ease of use, which makes the whole process attractive and looking forward to even the most demanding client.

- The transparency principle is considered to be cornerstones of the platform, therefore, the end users deserve to have the prices that are not deceitful and consistent. The transparency gives the consumer a deeper understanding of the product before making the purchase by the corner if the right cushion he or she is looking for. The capacity to declare zero hidden charges as well as to assure consistency and freedom will be used by us to ensure our customers agree and agree with us. Also, on the quality and price of our products, consumers are able to follow their desires with an open hesitation which gives them the responsibility for their purchasing decisions from the very beginning.
- Business Optimization: Taxi service providers can efficiently manage their fleet, analyze order statistics, and improve overall service quality. It will assist customers to work more systematically and ultimately, the clients will get superior services. On top of that, transport service agent can combine use of optimized allocation of assets and time management tools to turn down idling time. There is huge cost saving and this also results in increased efficiency. Smart analytic tools can analyze reams of statistical data, detecting patterns and assisting organizations to use data for business use. Consumption feedback of customers, who use aggregated and evaluative ratings on their website, may help providers to improve service quality and to maintain their client's loyalty. This tool is a very helpful in ensuring that producers are fully equipped in the areas of business automation as well as in the skills and information service quality, and service delivery.
- Technological Innovation: By incorporating Java EE Servlets and JSP, this project embraces modern web development practices, showcasing the capabilities of these technologies. The Java EE Servlet provides the backbone in the backend brought about by our light-weight implementation, mostly focusing on receiving and processing data, as well as managing application logic. They take a holistic approach to developing web applications that are capable of scaling and performing efficiently in complex environments, which is achieved through the use of MVC design pattern that enables us to implement modular and structured solutions. Besides, JSP empowers technically to accomplish dynamic webpage and presentation of our web app by directly incorporating the Java program into HTML. This can facilitate the architecture of the web pages that

reacts to user information, and program logic. As a result, it is more satisfying and engaging to the user as it gives them an interactive way of interacting with the application. By integrating Servlets and JSP carefully, we prove a stronghold of technological innovation and ensure a modern web app, which offers many features .

1.5 Organization Of Project Report

This report is divided into 6 chapters:

- i. Chapter 1 covered the introduction and central idea of the project design. The goals and techniques of the project are included in this chapter.
- ii. Chapter 2 provides a review of the literature, which includes several scholarly articles on taxi reservations that we used to compare our results with those of other researchers.
- iii Chapter 3 covered the system development, code snippets and algorithms, hardware and software configuration, front-end and back-end system capabilities.
- iv. Chapter 4 offers an explanation of testing resources and techniques along with illustrations of testing.
- v. Chapter 5 covers the presentation and interpretation of the data in addition to the results and evaluations.
- vi. The project's outcome and future scope are outlined in Chapter 6.

Chapter -2

Literature Survey

2.1 Overview Of Relevant Literature

The literature review encompasses a broad spectrum of taxi booking systems, delving into various technologies and methodologies and offering valuable perspectives from numerous investigations. There are multiple themes that represent different operational or technological aspects of this investigation.

In the world of technology, the cab booking system that makes use of the Google Maps API, Dart, and Flutter is particularly notable because it uses real-time data and modern application development. Nevertheless, excluding dynamic elements like surge pricing and relying solely on static fare estimates has drawbacks [1]. Furthermore, several papers discuss the integration of GPS and mapping services, which is very beneficial for route optimization, effective dispatching, and location tracking. One study makes use of Dijkstra's algorithm to emphasize the necessity of improving scheduling effectiveness [2].[3][10][5]. Southeast Asian ride-hailing applications use Geographic Information Systems , indicating favorable user feedback [4]. Additional technological approaches that are employed in various systems include Java, Firebase, VB.net, GPRS Modem, Android-compatible devices, and [5][6][10]. In the context of dynamic taxi ridesharing, machine learning and IoT are investigated, with potential advantages like lower fares and higher driver earnings [10]. A/B testing and CRM techniques investigate the two primary factors that customers consider when choosing a taxi service: price and reliability [8]. Price is found to be a crucial factor in an Analytical Hierarchy Process analysis of car rentals [11]. Ultimately, an Integrated Development Environment (IDE), PHP, CSS, and HTML are used to create an ambulance booking application [12].

When creating a car booking system, operational factors emphasize user adoption and market viability [2]. A case study examining customer loyalty to ride-hailing taxi services emphasizes the significance of technology and financial gain. [7]. Research is being done to learn more about the elements that consumers value most when choosing a call taxi service provider, with an emphasis on the importance of socioeconomic and service attribute variables [8].

Limitations and challenges that are acknowledged include privacy concerns about GPS data and accessibility issues for users running specific operating systems. The planned automated taxi booking system depends on the availability of parking spaces set aside for that purpose, so infrastructure development is required. [3][5][6].

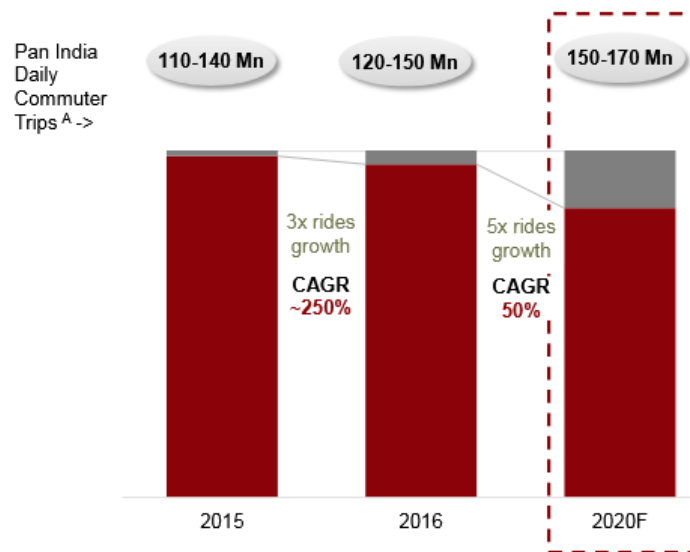


Figure 1: Growth Of Cab Booking Industry[13]

2.2 Key Gaps In Literature

A review of the literature reveals a number of noteworthy gaps that present worthwhile directions for further study and advancement in the field of cloud-based taxi booking services. First of all, most previous research employs static fare estimates, omitting the dynamic character of variables like surge pricing. To improve the accuracy of fare estimates, research on efficient surge management systems and dynamic pricing models is therefore desperately needed [1].

Secondly, there is a clear need for improvement as the accessibility is restricted to particular operating systems. Subsequent investigations ought to concentrate on remedies that guarantee all-encompassing user accessibility and accommodate a wider range of hardware and operating systems [6].

The necessity for privacy-preserving technology research is highlighted by the persistent privacy concerns surrounding GPS data. Strong solutions will ensure that user information is secure when utilizing location-based services. [3][10].

Infrastructure dependencies pose a significant challenge to the proposed automated taxi booking system, even in the case where assigned parking spaces are available. Future research projects should focus on solutions that don't rely on particular parking infrastructure to reduce reliance and ensure flexibility in use [5].

Closing the known research gap on drivers' labor rights for ride-hailing services is imperative. To support the rights and protections of drivers employed in the gig economy, extensive research is needed [4].

To increase the application of dynamic ridesharing systems, more research should focus on validation in different urban settings. Considering variables such as traffic patterns and cultural differences can help optimize these systems for a variety of scenarios [10].

Subsequent studies ought to specifically tackle privacy issues and perform a comprehensive examination of constraints and possible prejudices in survey approaches. This will ensure the transparency and integrity of the research process [1][3][8][9].

Integrating cutting-edge technologies like artificial intelligence (AI) and machine learning into dynamic taxi ridesharing systems is a challenging task. Further investigation into scalable and efficient integration strategies is required in order to fully realize the potential benefits of these technologies [10].

Particularly in the area of ride-hailing service customer loyalty, the proposal for comparative studies between developed and developing nations offers a promising direction for future research. These studies can offer nuanced perspectives on the factors influencing consumer behavior across various cultural and economic contexts [7].

Lastly, methodological transparency in research papers needs to be improved, especially when it comes to studies that focus on customer priorities. By ensuring the validity and repeatability of results, this will promote credibility in the field [8][11].

Table 1 : Tabular form of Literature Review

S. no	Paper Title	Journ al/con ferenc e	Tools/Techniques	Results	Limitations
1.	Customer-Friendly Cab Booking System[1]	2023	ASO) techniques. API Integration	This paper explores the dynamic landscape of cab and taxi services in response to the surging demand for convenient and safe transportation options	Concerns about data privacy and security. Regulatory and safety challenges . Potential price instability and conflicts. - Complex implementation
2.	Factors influencing customer's loyalty towards	2023	Discusses a system for taxi service analytics and visualization, using RNN and java	Conducts a study in Vietnam using PLS-SEM to analyze ride-hailing loyalty	Limited generalizability due to a focus on a developing country.

	ride-hailing taxi services[2]				
3.	The Planning Process for USIM Students' Car Booking(2023)[3]	2023	Global Positioning System (GPS)	adopts a Waterfall approach and incorporates authentication and authorization mechanisms for security	Market Competition ,Market Viability, User Adoption ,Data Privacy
4.	Taxi Intelligent Dispatch System Based On GPS[4]	2023	Integrated Development Environment (IDE), MQTT	The improved algorithm reduces scheduling complexity and enhances efficiency while considering real-time traffic data, optimizing taxi dispatching.	The paper lacks specific details on limitations but suggests potential challenges, including widespread GPS phone adoption and privacy concerns.
5.	Ride-hailing applications in Southeast Asia.[5]	2022	GIS[15] (Geographic Information Systems)	Indicates that RHA users hold favorable views of RHA compared to public transportation , reflecting negative attitudes of the general SEA population	Notes the need for more research on labor protections for RHA drivers. Calls for exploration of regulatory frameworks
6.	An Automated Taxi Booking and Scheduling System.[6]	2022	AspectJ (AOP framework)	The simulation demonstrates significant time and fuel savings, along with reduced congestion, when compared to the random taxi roaming model.	The proposed system assumes the availability of designated parkings, which may require substantial infrastructure development
7.	Factors Influencing Customer's Loyalty towards Ride-hailing Taxi Services – A case Study of Vietnam.[7]	2021	Partial Least Squares - Structural Equation Modeling (PLS-SEM)	This study explores factors influencing customer loyalty towards ride-hailing taxi services in Vietnam, focusing on technology and economic value	Suggests the need for further research to explore differences between developing and developed countries
8.	Intelligent Cab Service System.[8]	2021	Java Agent DEvelopment Framework (JADE)	The study addresses the limitations of the current car booking system via Telegram,	Market Viability User Adoption Market Competition

				which is not optimized for transportation services.	Data Privacy
9.	Understanding Customer Priorities for Selection of Call Taxi Service Provider.[9]	2021	A/B Testing,CRM	The study found that socioeconomic and service attribute factors influenced people's choice of taxi service in Bhubaneswar, with an emphasis on reliability and cost.	The paper lacks details on the survey methodology and potential biases, focuses on a specific city and taxi services, and does not address potential privacy concerns
10	Car Rentals Knowledge and Customer Choice.[10]	2020	Analytical Hierarchy Process (AHP)	This paper attempts to identify factors chosen by a customer while choosing car rental services using the AHP methodology	Suggests the study could be extended to capture linguistic judgment using fuzzy multi-criteria decision-making methods.
11	Ambuitem: Ambulance Booking Application for Emergency Health Response, Blood Inventory[11]	2020	DOI (Digital Object Identifier) , Integrated Development Environment (IDE)	This paper presents an ambulance booking application allowing users to request assistance, track ambulances, and select the best option based on cost and distance.	Doesn't address traffic and road conditions, potentially affecting response times. Efficiency depends on ambulance availability
12	Smart Vehicle Management using Cost-effective Approach[12]		MQTT, Continuous Integration/Continuous Deployment (CI/CD)	This paper discusses a carpooling system facilitating ride-sharing between car owners and passengers	Challenges include limited vehicle diversity. Scalability concerns and safety issues.

Chapter -3

System Deployment

3.1 Requirements And Analysis

3.1.1 Functional Requirements

- **User Roles and Responsibilities:** The system defines the Administrator and Customer roles in detail by giving each group particular rights and responsibilities. Determining the range of actions appropriate for each application position is part of this.
- **Authentication and Authorization:** The project uses a strong user login procedure to guarantee a secure environment. This includes authentication and authorization. Detailed explanations of permissions and access restrictions help protect the system by outlining the functionalities that are available to various user roles.
- **Booking Management:** This section's functional requirements deal with how passengers make taxi reservations. Administrators have access to every tool required for effective management and processing of reservation requests. When there are updates to trips, booking confirmations, payment receipts, or other pertinent information, the system ought to promptly notify users and drivers. Push notifications, SMS, and email can all be used to send notifications.
- **Taxi Fleet Management:** Administrators can monitor the fleet by using the system's features for viewing, adding, and removing taxis. Moreover, logic is set up to adjust the taxi fleet's availability dynamically based on the volume of reservations received.

- **User Dashboard:** The user dashboard is a crucial component that offers administrators and customers a range of features and a comprehensive data display. This ensures a user-friendly and efficient interface. To accommodate a large user base, the system must support multiple currencies and languages. Localizing user interfaces, date/time formats, and currency symbols are all included in this.

3.1.2 Non-Functional Requirements

- **Performance:** Acceptable response times for critical operations and the system's ability to handle several user interactions at once are outlined in the non-functional requirements of the system. With minimal performance degradation, the system ought to be able to accommodate a sizable number of concurrent users and reservations.
- **Security:** Two of the techniques covered are user authentication and data protection. Encryption standards and access controls are designed to safeguard the confidentiality and integrity of sensitive data. Mechanisms for gracefully handling errors and informing users of them should be built into the system.
- **Reliability:** Clearly defined availability and uptime are expected for the system. Furthermore, procedures for backup and recovery are offered to ensure data availability and integrity in the event of unanticipated circumstances. Adequate backup and recovery procedures should be in place in order to ensure data integrity at all times.
- **Usability:** To guarantee a seamless and simple user experience, the requirements for the user interface are specified in detail. Accessibility considerations, which take into account a variety of user requirements, ensure an inclusive and user-friendly design.

3.1.3 System Constraints

The project operates within the parameters of a well-defined technology stack that consists of HTML, CSS, AWS (EC2), Linux, Gitbash, SSH, Java, Tomcat, MySQL, JDBC, and Java EE Servlets. These technologies were chosen for the development of web applications because they are reliable, widely used, and compatible.

- **Java and Java EE Servlets:** When Java and Java EE Servlets are used, applications can gain from a dependable and expandable server-side environment as well as effective management of user requests and responses. With Servlets, developers can effectively manage session data, engage with databases, and implement the system's backend logic. Additionally, integrated support for features like session management, security, and scalability—all of which Java EE Servlets offer—is necessary for a robust taxi booking platform. All things considered, programmers can create a scalable, secure, and high-performance taxi booking system that meets the needs of administrators and users by utilizing Java and Java EE Servlets.
- **HTML and CSS:** The user interface is organized and styled using HTML and CSS throughout the project, producing a responsive and aesthetically beautiful design that enhances the user experience. CSS is needed for styled HTML elements in order to improve the booking system's usability and appearance. To create a unified and visually appealing user interface, developers can alter the appearance of text, colors, fonts, buttons, and layout elements using CSS.
- **Tomcat:** The Apache Tomcat web server is used to deploy and manage Java Servlets. It is an excellent choice for hosting the application because it conforms with Java EE specifications. Customers and the taxi booking platform can both rely on a reliable connection thanks to Apache Tomcat's integrated HTTP server. By listening for incoming HTTP requests, forwarding them to the appropriate servlets or JSP pages, and responding with the necessary data, it makes sure that user interaction with the application is seamless. The taxi booking system's overall

scalability, dependability, and performance are enhanced by integrating Apache Tomcat, enabling speedy processing of booking requests and an ideal client experience.

- **Network Bandwidth:** The amount of network bandwidth that is available for FileZilla file transfers between your local computer and the EC2 instance affects deployment speed. When connecting to external services, like payment gateways or mapping APIs for route computation, the system needs enough network bandwidth to ensure consistent communication. For the system to efficiently handle several requests at once and ensure low latency and uninterrupted service for users—even during periods of peak demand—it needs a robust network architecture and enough bandwidth.
- **MySQL:** MySQL is the name of the relational database management system. MySQL facilitates efficient data management, retrieval, and storage. It provides a dependable and efficient way to handle the application's database requirements. Developers can easily establish connections between the Java application server and the MySQL database server thanks to MySQL's compatibility with Java through JDBC (Java Database Connectivity). This makes it possible to efficiently access and manipulate data from Java code, which makes it easier to integrate application functionality and business logic into the taxi booking system.
- **JDBC:** This technology facilitates data exchange by connecting the Java program to the MySQL database. Data consistency and integrity are guaranteed throughout intricate operations involving numerous database interactions thanks to JDBC's transaction support. For example, JDBC allows developers to run a sequence of database operations within a transaction context when a user submits a booking request. This guarantees that all changes are either committed simultaneously or rolled back in the event of an error, protecting the integrity of the system's data.

- **FileZilla:** The widely used FileZilla FTP (File Transfer Protocol) client can be used to move files between the local computer and the EC2 instance. It offers a graphical user interface to make file management more efficient. When FileZilla is integrated with a Java-based taxi booking system, the file management features of the system are improved, making tasks like deployment, backup, and maintenance more efficient. Developers may guarantee the dependability, security, and scalability of the taxi booking system while reducing administrative overhead and operational complexity by utilizing FileZilla's features for safe file transfer and user-friendliness.
- **Git Bash:** On Windows operating systems, Git Bash is a command-line interface (CLI) tool that offers a Unix-like environment. It enables developers to work with Git repositories and execute native Unix/Linux commands on Windows PCs. It comes with the Bash shell and the Git command-line tools in addition to other Unix utilities.
- **Ubuntu:** The operating system selected for the EC2 instance is Ubuntu. Popular Linux distribution Ubuntu is well-known for its dependability, security, and user-friendliness. The taxi booking system is shielded from outside attacks and weaknesses by Ubuntu's security enhancements. To guarantee that the system is always safe from the newest security threats, Ubuntu receives security updates and patches on a regular basis. It is also simpler for developers to apply best practices for safeguarding the system and minimizing potential security risks thanks to Ubuntu's robust community support and thorough documentation.
- **SSH (Secure Shell):** To establish a remote connection with Linux servers, including EC2 instances, one must utilize the Secure Shell protocol. Through a secure SSH connection, developers can update applications, manage server configurations, and troubleshoot problems. Remote server management is a major application of SSH in taxi reservation systems. Without requiring physical access to the server, system

administrators can safely connect via SSH to the server hosting the Java application and carry out standard maintenance operations, track system performance, and troubleshoot issues. Sensitive data, including administrative commands and login credentials, are shielded from prying eyes and illegal access thanks to SSH's encrypted communication.

- AWS (EC2): As the application expands, this cloud deployment option provides scalability, dependability, and efficient resource management. Elastic Compute Cloud, or Amazon EC2, is a crucial AWS service that can be used since it provides resizable compute capacity in the cloud. The Java application server, which powers the taxi booking system, can be hosted on EC2 instances, giving you the flexibility to scale server resources up or down in response to demand. By doing this, it is guaranteed that the system can manage traffic variations and continue to operate at peak efficiency without overloading its resources.

The Java application server constitutes the dynamic and swift feature in terms of handling traffic variations in the important tasks for building the reservation system on Amazon EC2. EC2 scaling makes possible a quick change of server resources, which are used on the go and do not create extra resource consumption. Its reliability, which is supported by Amazon's resilient infrastructure, means that you can count on the service to be available all the time. Furthermore, EC2's resource management units, together with its option for inferring the utilization of computing power, is structured so to add efficiency. Similarly, its flexible units for configurations make it possible to customize computing needs. Through this mode of operation, the system is not only practicing the highest standard of performance but, also, savings in cost and disk space addressing the incessant demand make it the core cornerstone of a resilient and responsive system architecture.

3.2 Project Design And Architecture

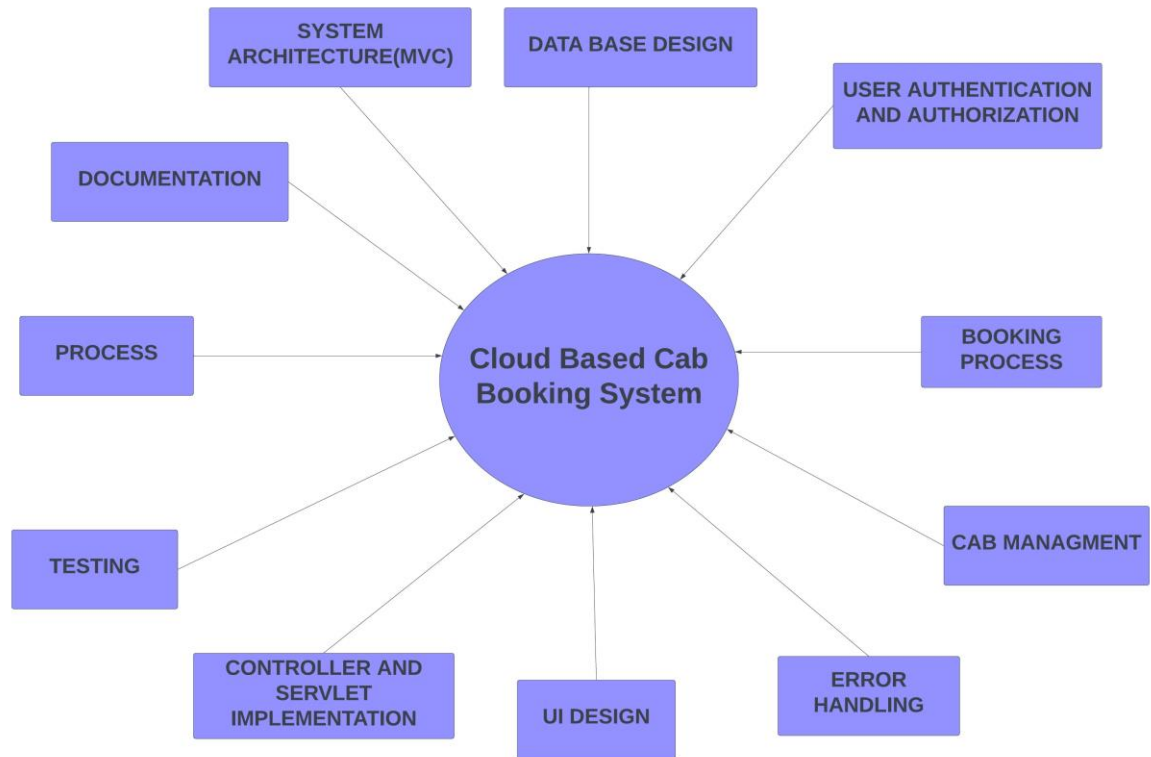


Figure 2: Project Design

The following modules make up the cloud-based taxi booking service as shown in Figure 2. Each module is linked to a specific system feature or capability. An overview of each module's contents is provided below:

- **System Architecture:** The project's design and architecture adhere to the Model-View-Controller (MVC[14]) pattern as shown in Figure 3, a well-liked architectural paradigm for web applications. This is a deliberate choice made with the goal of building a structured and modular architecture to enhance code reuse, maintainability, and scalability.

- **Model:** The Model component houses the data and business logic. Entities like Account, Car, Order, and Route represent various facets of the taxi service. The Model ensures a clear division of responsibilities and establishes a seamless communication channel with the MySQL database via the use of the JDBC (Java Database Connectivity) API. The efficacy and integrity of data management are enhanced by this design principle.
- **View:** The View component is in charge of rendering the user interface. A dynamic and adaptable presentation layer is offered by JavaServer Pages (JSP) and the JavaServer Pages Standard Tag Library (JSTL). The multilingual support built into the View allows users to interact with the application in their own language
- **Controller:** The data flow between the model and view is controlled by Java EE Servlets, a stand-in for controllers. They are necessary to initiate pertinent business logic in the Model, respond to user requests, and update the View. Java EE Servlets also manage important functions like session management, user authentication, and authorization. This dual function ensures data integrity and a safe, personalized user experience.

The View seamlessly incorporates multilingual support while maintaining user inclusivity and accessibility. The technology stack consists of Java, Java EE Servlets, Tomcat, MySQL, JDBC, HTML, and CSS in addition to AWS (EC2). Reliability, compatibility, and industry best practices were taken into consideration when making these decisions.

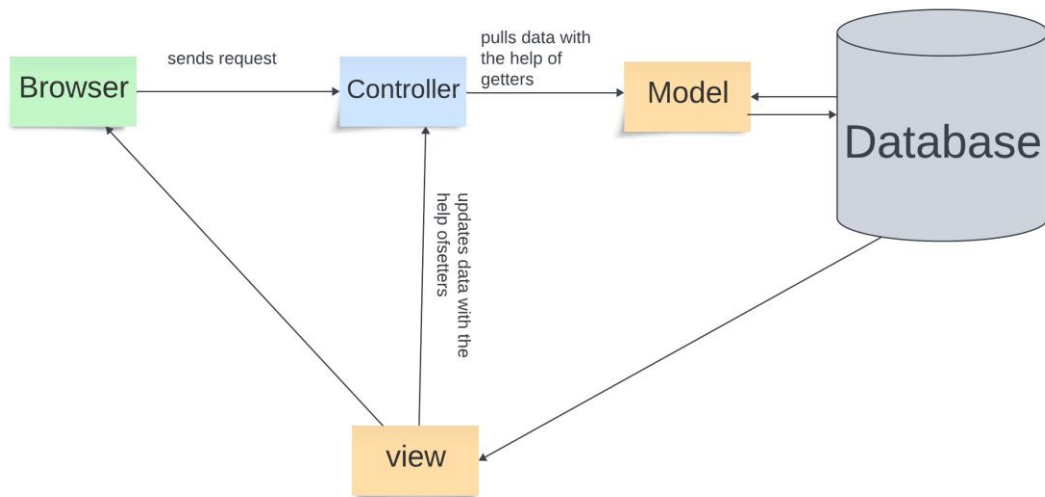


Figure 3: MVC[14] Architecture

- **Database Design:** The Database Design module's goal is to efficiently set up the database so that data can be stored and retrieved. Entity-Relationship Modeling (ERD) is used to find entities and relationships.
- **User Authentication and Authorization:** The module provides for user authentication and authorization protocol and guarantees safe accessibility of the application. Hashing algorithms for securing password, Admin and customer access controls, as well as authentication and login processes.
- **Booking Process:** This module guarantees that authorized users log in using a user authentication process. Password security, user registration and login, as well as Admin and Customer roles employ hashing functions.
- **Cab Management:** The fleet management of taxis monitors current information and statuses. This platform has two features: an interface to manage and list taxis among other functions, as well as dynamic taxi availability update system that keeps track of the number of bookings received.

- **Error Handling:** The purpose of this module is to identify, document, and correct any errors as they occur during the running of the program. It ensures a robust system reaction where there are cases of incorrect input and system failures.

- **UI Design:** The purpose of the UI Design module is to design an attractive and easily understandable interface. It is a set of practices that aim at enhancing accessibility through how the components are made, assembled, and offered.

- **Controller and Servlet Information:** The data flow controller in this module is Java EE Servlets. As a result of user requests, it dials for appropriate business logic in the model and the view is then updated. Moreover, user authentication, authorization, and session management are handled by servlets.

- **Testing:** The Testing module includes unit testing for individual components, integration testing for the system as a whole, and user acceptability testing for the client and admin interfaces. It ensures that the software functions as intended and locates and resolves any bugs.

3.2.1 Activity diagram:

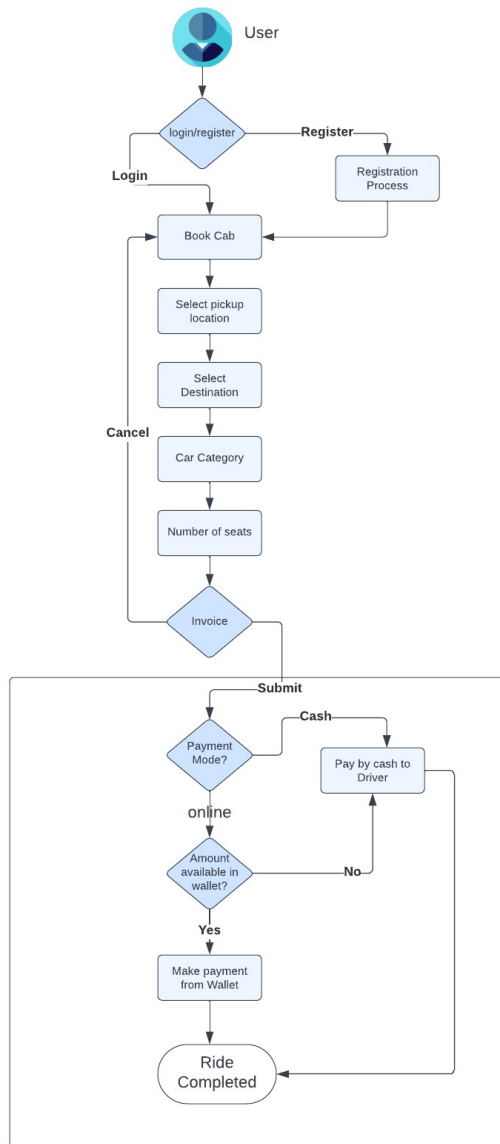


Figure 4: Activity diagram

3.2.2 Block Diagram

An extensive architecture for a cloud-based taxi booking service is displayed in the block diagram as shown in Figure 5. The Client Application is at the core of the User Interface, which offers an easy-to-use interface for taxi reservations. Improved multilingual support for a worldwide user base and Google Maps integration allow for seamless location

services and route planning within the application. Security mechanisms are integrated to protect user data and transactions. The Cloud Server, at the center of the system, is in charge of handling user requests and carrying out crucial application logic. These consist of Cab Fleet Management, which is in charge of managing the taxi fleet, Booking Management, which handles all booking-related matters, and User Authentication and Authorization, which maintains user roles and guarantees safe login. The Cloud Server is linked to a MySQL database that holds user data, booking history, and taxi specifics. The incorporation of External APIs enhances the application's functionality. Two examples of security measures that fortify the system's communication channels are SSL and encryption. All things considered, with a focus on scalability, security, and user experience, this architecture builds a robust and networked structure that enables efficient taxi booking services.

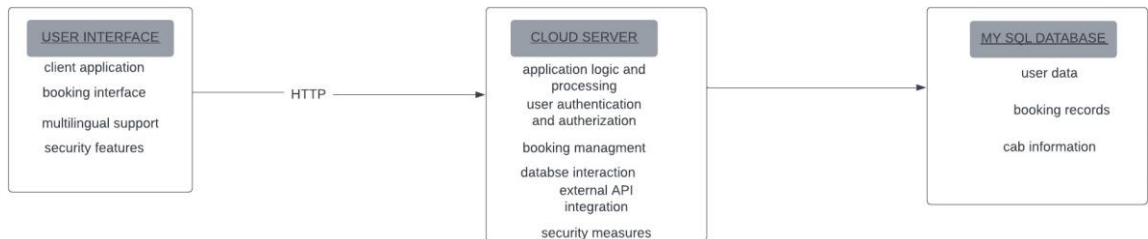


Figure 5: Block Diagram

3.3 Data Preparation

The data preparation process for the Taxi Service project involved setting up the database schema, populating tables with meaningful data, and establishing relationships between entities. The MySQL script provided in the project description outlines the creation of tables such as `account`, `route`, `order`, `car_details`, `car`, `order_has_car`, and more.

- User Accounts: User accounts were created for both administrators and clients using the `account` table, specifying details such as login, email, password, phone number, discount, and role.
- Language Support: Language entries were added to the `language` table, supporting multiple languages.
- Car Details: Different car categories were defined in the `car_details` table, specifying the category and the number of seats.
- Cars: Fleet details were populated in the `car` table, including the status of each car (to_order, in_run, inactive) and the corresponding `car_details_id` for categorization.
- Car Descriptions in Multiple Languages: Descriptions for each car were provided and stored in the `car_has_language` table.
- Routes: Various routes were defined in the `route` table, specifying departure and arrival locations along with the corresponding distances.
- Orders: Test orders were generated and stored in the `order` table, capturing details such as the route (`route_id`), price, number of passengers, and creation date.
- Establishing Relationships: To establish relationships between entities, entries were made in junction tables:
 - Entries in `account_has_order` linked user accounts with specific orders.
 - Entries in `order_has_car` established associations between orders and cars.

This detailed data preparation ensures that the database is populated with diverse and realistic data representative of various scenarios. It lays a solid foundation for testing the Taxi Service application's functionality, user interactions, and system responses across different use cases. The relationships between entities are crucial for testing the seamless flow of data and interactions within the application. Regular reviews and updates to the dataset will be essential to adapt to evolving project requirements.

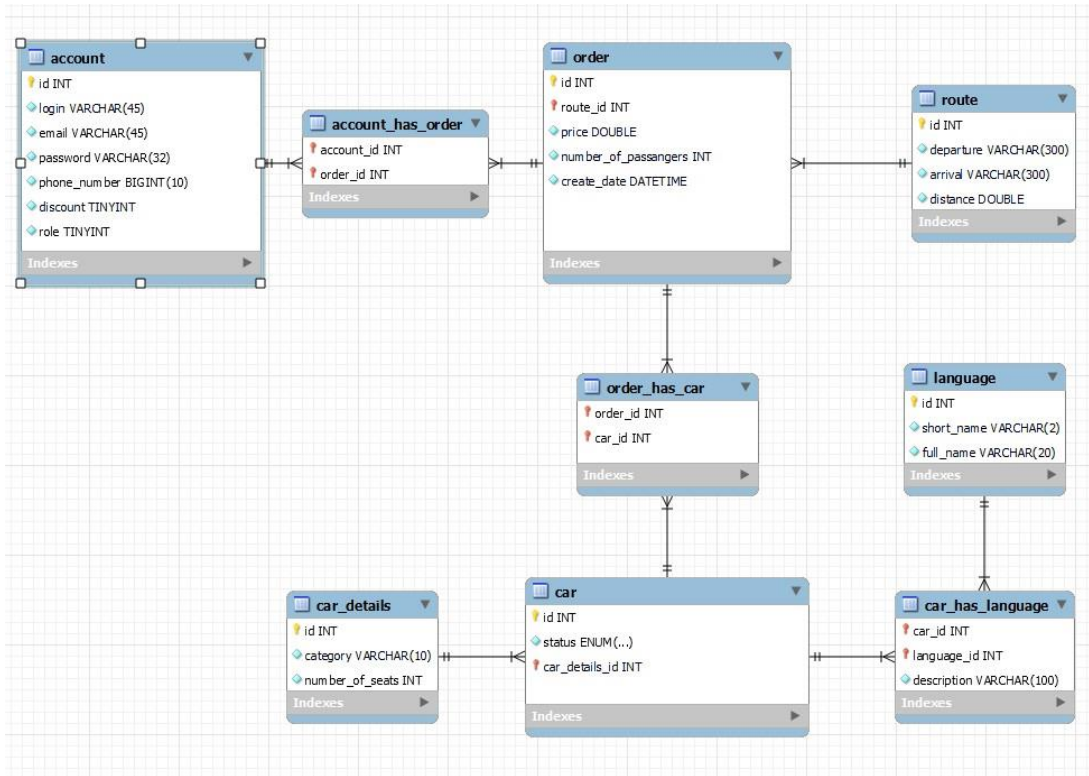


Figure 6: Database Model

Table 2: `account`

Column	Data Type	Description
`id`	INT	Unique identifier for the user account
`login`	VARCHAR(45)	User login username
`email`	VARCHAR(45)	User email address
`password`	VARCHAR(32)	User password (hashed)
`phone_number`	BIGINT(10)	User phone number
`discount`	TINYINT	Discount percentage for the
`role`	TINYINT	User role (e.g., admin, regular user)

Table 3: `route`

Column	Data Type	Description
`id`	INT	Unique identifier for the route
`departure`	VARCHAR(300)	Departure location
`arrival`	VARCHAR(300)	Arrival location
`distance`	DOUBLE	Distance of the route in kilometers

Table 4: `order`

Column	Data Type	Description
`id`	INT	Unique identifier for the order
`route_id`	INT	Foreign key referencing `route` table
`price`	DOUBLE	Total price of the order
number_of_passengers`	INT	Number of passengers for the order
`create_date`	DATETIME	Order creation date

Table 5: `car_details`

Column	Data Type	Description
`id`	INT	Unique identifier for the car details
`category`	VARCHAR(10)	Car category (e.g., economic, comfort)
`number_of_seats`	INT	Number of seats in the car

Table 6: `car`

Column	Data Type	Description
`id`	INT	Unique identifier for the car
`status`	ENUM	Car status (to_order, in_run, inactive)
`car_details_id`	INT	Foreign key referencing `car_details` table

Table 7: `order_has_car`

Column	Data Type	Description
`order_id`	INT	Foreign key referencing `order` table
`car_id`	INT	Foreign key referencing `car` table

Table 8: `language`

Column	Data Type	Description
`id`	INT	Unique identifier for the language
`short_name`	VARCHAR(2)	Short name code for the language
`full_name`	VARCHAR(20)	Full name of the language

Table 9: `account_has_order`

Column	Data Type	Description
`account_id`	INT	Foreign key referencing `account` table
`order_id`	INT	Foreign key referencing `order` table

- `car_has_language` Table

Table 10: `car_has_language`

Column	Data Type	Description
`car_id`	INT	Foreign key referencing `car` table
`language_id`	INT	Foreign key referencing `language` table

3.4 Implementation

3.4.1 Backend Implementation:

-Language and Framework:

Utilized Java and Java EE Servlets for the backend implementation. Model-View-Controller (MVC[14]) architecture is used to create a structured and modular design. The implementation of the Model-View-Controller (MVC[14]) architectural pattern makes use of a structured file structure. The Model is represented by Java classes such as `Order.java`, `Car.java`, and `User.java`, encapsulating data and business logic. Serving as controllers, Java EE Servlets like `OrderServlet.java` and `UserServlet.java` manage data flow and communication between the Model and View. JSP pages, such as `order.jsp` and `userProfile.jsp`, function as the View, dynamically rendering content based on processed data. The `web.xml` file acts as the deployment descriptor, configuring the web application, while additional files like `style.css` and `script.js` enhance the user interface. The culmination of these components is packaged into the `taxi-service.war` file, creating a deployable web archive for seamless execution on Apache Tomcat. This structured file organization ensures a modular and organized implementation of the MVC[14] pattern, facilitating the development and maintenance of the Java web application.

- Database Interaction:

Use Java Database Connectivity (JDBC) to establish connections to the MySQL database as shown in Figure 7 , Figure 8 and Figure 9 .


```

private DBManager() {
    try {
        Context initContext = new InitialContext();
        Context envContext = (Context) initContext.lookup(
            name: "java:/comp/env");
        dataSource = (DataSource) envContext.lookup(name: "jdbc/Taxi");
    } catch (NamingException e) {
        LOGGER.error(message: "Cannot init DBManager", t: e);
    }
}
}

```

Figure 7: Database Connection Initialization

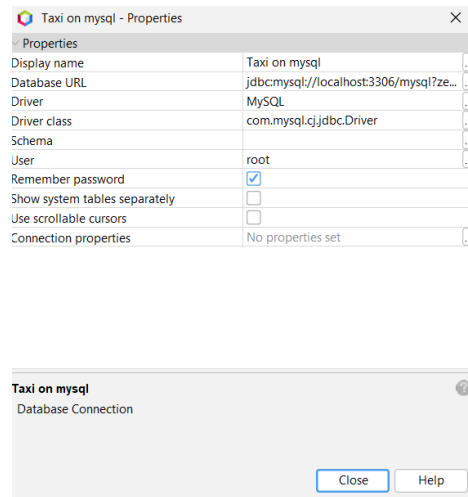


Figure 8: MySQL Database Configuration for Taxi Service

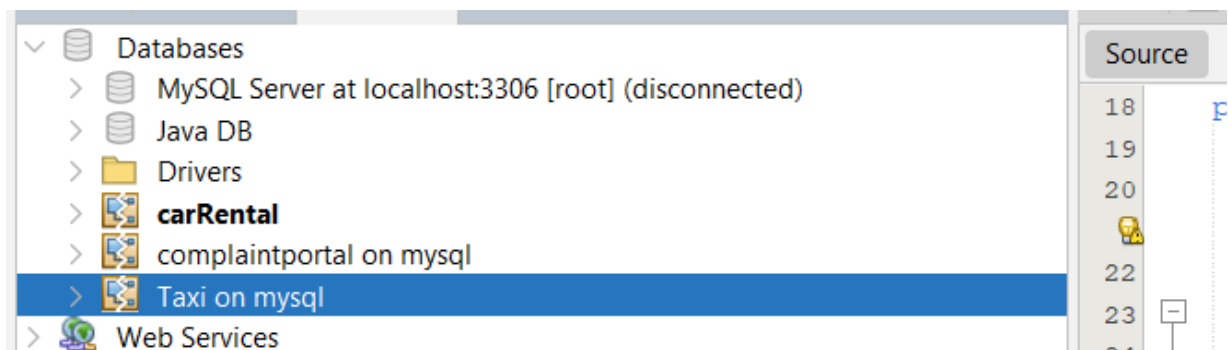


Figure 9: Database in Apache Netbeans

- User Authentication and Authorization:

Firstly, develop secure user registration and login functionalities. Then implement authentication and authorization mechanisms using Java EE Servlets. Lastly, ensure password security through MD5 hashing algorithm as shown in Figure 10.

```
public class PasswordEncoder {  
    private PasswordEncoder() {  
    }  
    public static String encode(String password) {  
        return DigestUtils.md5Hex(data: password);  
    }  
}
```

Figure 10: Implementation Of Md5

- Data Validation:

In the taxi booking system, the `DataValidator` class ensures the accuracy and security of user inputs by validating login details, phone numbers, emails, and passwords against predefined patterns as shown in Figure 9. This proactive validation not only prevents erroneous or malicious inputs but also enhances the overall user experience. By maintaining strict adherence to input requirements, the system fortifies itself against vulnerabilities, ensuring a secure and reliable platform for taxi bookings.

```

7 public class DataValidator {
8     private static final String LOGIN_PATTERN = "([a-zA-Z0-9_]){5,}";
9     private static final String PHONE_PATTERN = "[0-9]{10}";
10    private static final String EMAIL_PATTERN = "[\\w-\\@]+(\\.\\w+)*@[\\w-]+(\\.\\w+)*\\.([a-z]{2,})";
11    private static final String PASSWORD_PATTERN = "([a-zA-Z0-9_]){4,}";
12
13    public static boolean checkLoginData(String login, String phoneNumber, String email, String password) {
14        if (!matchPattern(data: login, currentPattern: LOGIN_PATTERN)) {
15            return false;
16        }
17        if (!matchPattern(data: phoneNumber, currentPattern: PHONE_PATTERN)) {
18            return false;
19        }
20        if (!matchPattern(data: email, currentPattern: EMAIL_PATTERN)) {
21            return false;
22        }
23        return matchPattern(data: password, currentPattern: PASSWORD_PATTERN);
24    }
25
26    private static boolean matchPattern(String data, String currentPattern) {
27        Pattern pattern = Pattern.compile(regex: currentPattern);
28        Matcher matcher = pattern.matcher(input: data);
29
30        return matcher.matches();
31    }
32 }

```

Figure 11: Data validation code

-Pagination

The `Pagination` class in the `com.epam.taxi.utils` package offers a flexible and dependable solution for adding pagination functionality to web applications. It uses a configurable constant to set the number of records per page and dynamically calculates the total number of pages based on the size of the entity list. Its `createPagination` method adds pertinent pagination attributes to the provided `HttpServletRequest`, thereby preventing `ArrayIndexOutOfBoundsException` scenarios. Its `getPageList` function also returns a subset of entities that are unique to the specified page. Developers can quickly implement pagination with this class, which will improve user experience by simplifying the navigation of large datasets.

```

/**
 * Method which returns list with specific number of records.
 *
 * @param entityList List of entities that we want to display on our page.
 * @param pageNumber The page number on which the user stopped.
 * @return List of records for specific page.
 */
private static List<? extends Entity> getPageList(List<? extends Entity> entityList, int pageNumber) {
    int startPoint = (pageNumber - 1) * RECORDS_PER_PAGE;
    int endPoint = pageNumber * RECORDS_PER_PAGE;

    //Insurance from ArrayIndexOutOfBoundsException
    if (endPoint >= entityList.size()) {
        endPoint = entityList.size();
    }

    return entityList.subList(fromIndex: startPoint, toIndex: endPoint);
}

```

Figure 12: Pagination Code

-CommandContainer.java:

The `CommandContainer` class houses all commands in the taxi reservation system centrally, making it simpler to manage and allow scheduled access to command instances. It does this by using a static initialization block to fill a {Map` structure with examples of different commands, divided into common, client-specific, and admin-specific commands. Every one of these commands has a specific purpose in the application. Using Apache Log4j, the class generates debug logs that show successful container initialization as well as trace logs that show the number of commands that were initialised. These logs provide useful information about the initialization process. The `getCommand` method allows for the retrieval of specific commands solely by name, preserving modularity and flexibility in command handling.

```

public class CommandContainer {
    private static final Logger LOGGER = Logger.getLogger(clazz: CommandContainer.class);

    private static Map<String, Command> commands = new TreeMap<>();

    static {
        //Common commands
        commands.put(key: "login", new LoginCommand());
        commands.put(key: "logout", new LogoutCommand());
        commands.put(key: "noCommand", new NoCommand());
        commands.put(key: "changeLanguage", new ChangeLanguageCommand());

        //Client commands
        commands.put(key: "registration", new RegistrationCommand());
        commands.put(key: "checkOrder", new CheckOrderCommand());
        commands.put(key: "createOrder", new CreateOrderCommand());
        commands.put(key: "analogOrder", new AnalogOrderCommand());
        commands.put(key: "getCarInfo", new GetCarInfoCommand());

        //Admin commands
        commands.put(key: "getOrdersList", new GetOrdersListCommand());
        commands.put(key: "getCarsList", new GetCarsListCommand());
        commands.put(key: "changeCarStatus", new ChangeCarStatusCommand());

        LOGGER.debug(message: "Command container was successfully initialized");
        LOGGER.trace("Number of commands " + commands.size());
    }
}

```

Figure 13: CommandContainer.java

-CommandFilter.java

1. doFilter Method:

Then, the ICMP method, instead of continuing with its journey, intercepts it and hides it from the final intended receiver, before it reaches them. Through this utilization, a general servlet request will firstly be converted into a HttpServletRequest, which, in turn, would enable you to get all the HTTP specific functions at your disposal. Afterward, it goes on to send a message of 'user authorized' result to the backend by using the 'isAuthorized' method. The method will let the request's task execution and processing through the filter chain, either if permitted or not, and users can then use the existing filterChain. doFilter(request, servletResponse). If the user does not have the necessary credentials for the method, it will not operate and the user will be redirected back to the initial app page.

2. isAuthorized Method:

This is an example of how check is utilized to make sure an user's role is toxic whether they want to execute the command to deter if they are allowed to do so. It accepts HttpServletRequest web-request object as its input, provides several methods to handle session data, and enables connectivity with other layers of middleware such as application components, business logic, and database. The method proceeds to the command parameter of request and returns the account of the user through session.

It then proceeds with several checks:- Next, the algorithm rumbles the following: When the command that was intended to be used is a null, so the agent returns of overwhelmingly true meaning unauthorized access instances (page refusal). If the user is not authenticated and the command belongs to the entranceCommand array, then it gives back a positive message and entrance in the commands such as login, register etc. While the user is not recognized if not equal to the elements in entranceCommand array and there is no command in the entranceCommand array, that returns false and the linkage to main page is realized. Next, if the authentication of the user is successful and the command matches with these commands array then, it will return true and consequently the user will perform the normal operation. If the user(admin) is the true flag of the establishCommand array and it is included with the admin-specific commands then it is exhibited and provides the admin to access the specific commands. Instead of using non client commands the user if named a client and commands are in clientCommand array, information is returned which will give access to client commands. The validation function is supposed to be in the form of a conditions statement. The condition is to be met for the function to return as true, signifying the authorized access. However, if none of these conditions is met, the function returns as false, implying the unauthorized access.

-FrontController.java:

FrontController.java is the backbone or the core who is investigating the flow of the request by the dispatcher. It is used as the root for all requests which are forwarded through the "/controller/regex" URL stemplens. After the receipt of an order, process() method that captures the name of the command and then Command object from CommandContainer utilizing this command name, which gets hold of it. A decoupled structure like this helps

modularization and makes it easy to change the logic controller by making him the centre of the execution. Completing the command execution, the servlet apprises the Path object and uses redirect flag to determine how to move the user either to the next views or some of the available resources in the system as deemed fit. The logging extension specifically aids via the Apache Log4j log messages on what commands were run and the URL addresses references, which can be utilized for debugging and performance analysis. Not only does the FrontController servlet implement these common patterns that belongs to the design of MVC but it also ensures a security for app navigation and access to data. It has a very high scalability and extendability level of the app.

-SessionListener.java:

Usually, the first thing people mention about the tag is the other executive power side, which is the ability to set the timeline for the events that are responsible for starting and stopping sessions. If I submit “an init parameter” to servlet container, you may start saying that ba servlet Container is the one which makes it possible to the session to set information, which is, for example, the session identifier in session identification. As such, this synchronization approach enables the threads to stop any sync session no matter how interconnected it is. In this technique, classes of the listening session involve the interval between the first class plays. This other stored information is allowed to be used in these session. Allocating a log facility for automation sessions is in principle possible with Log4j implementation, which allows to acquire the desired segment of the session data (drawn from the general information on automation process) for further employment. Finally HttpSessionLifecycleListener will accomplish initializing by defining the entities with HttpSessionListener interface. It then treats the event in a proper way but only connected to the servlet. This implies, therefore, the action of that system that is completely flexible and the respondent almost immediately to all the applications immediately. The other parts of application management are quite unexceptional and it is not so important for a sessionListreener to take a closer look such that the site being implemented is entirely up and running which is often a top thing that any web site owner wants to see run.

- Booking Process:

logic for customers to submit taxi booking requests.

3.4.2 Algorithm For Booking Process:

Step 1. User registration and login functionality.

Step 2. Clients initiate the booking process, providing details like departure, destination, passengers, and car category.

Step 3. System queries available cars based on location, category, and availability status. Assign an available taxi to the order, considering factors like proximity and category.

Step 4. If no suitable cars, provide alternative options for client confirmation.

Step 5. Calculate the trip cost using Manhattan distance and consider additional factors.

The Manhattan distance formula, also known as the L1 norm or taxicab distance, calculates the distance between two points in a grid-based system (like city blocks). For two points (x_1, y_1) and (x_2, y_2) , the Manhattan distance (D) is given by:

$$D = |X_2 - X_1| + |Y_2 - Y_1|$$

Step 6. Check if the client is eligible for the loyalty program discount (every second ride gets a 20% discount).

Step 7. Notify the client about the assigned taxi, estimated arrival time, and calculated trip cost.

Step 8. Confirm user acceptance of the details.

Step 9. Notify the assigned driver about the booking, providing user and location details.

Step 10. Initiate the ride as the driver confirms availability.

Step 11. Process the payment based on the calculated trip cost using secure payment gateways.

Step 12. Mark the ride as completed after reaching the destination.

Step 13. Allow users to provide feedback and ratings for the driver.

- Testing:

JUnit is a widely adopted framework for unit testing in Java, playing a crucial role in ensuring the reliability and correctness of software applications. In the context of our taxi

booking service project, JUnit facilitates the creation of dedicated test classes containing methods annotated with `@Test`, each representing a specific test case. These methods utilize JUnit's assertion methods to validate that the actual output of a particular unit or method aligns with the expected output. Annotations such as `@Before` and `@After` are employed for setup and teardown operations, ensuring a consistent testing environment. Test runners provided by JUnit execute these tests, and the framework generates detailed reports highlighting successes and failures.

By incorporating JUnit into our development process, we systematically verify the functionality of individual components, promoting code robustness and facilitating early detection of potential issues. This approach significantly contributes to the overall quality and stability of the taxi booking service application.

- Tomcat Server:

Configure Apache Tomcat as the web server for deploying and running Java Servlets as shown in Figure 14.

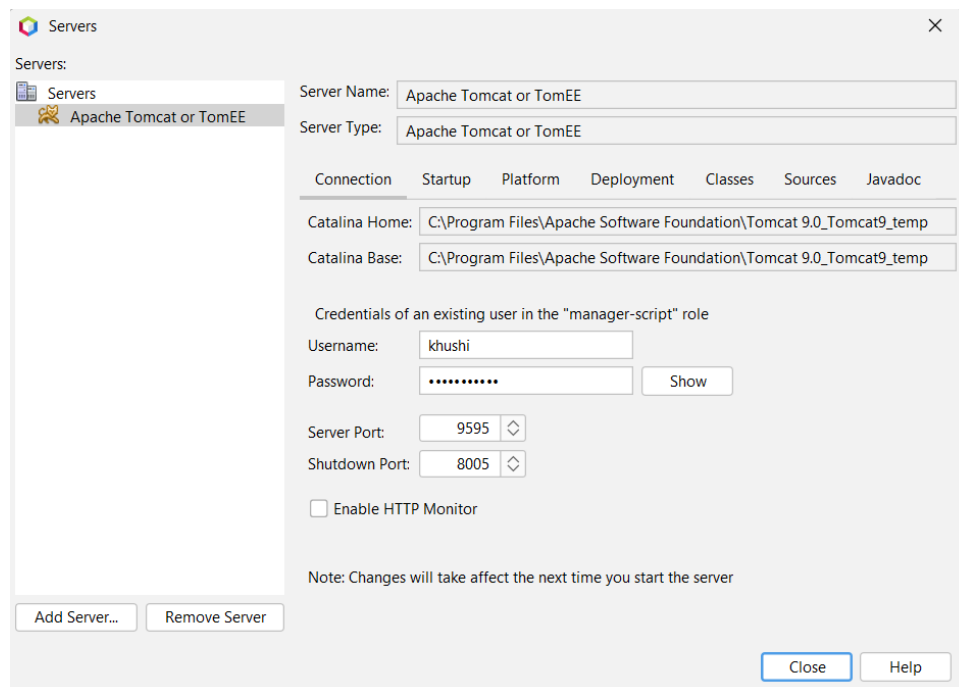


Figure 14: Tomcat Sever Configuration

3.4.3 Frontend Implementation:

- UI Design:

Use HTML and CSS for structuring and styling the user interface. Implement JavaServer Pages (JSP) and JavaServer Pages Standard Tag Library (JSTL) for dynamic and responsive presentation.

- Multilingual Support:

Provide multilingual support by localizing the web app so that users can interact with it in the language of their choice. Multilingual support is achieved by creating separate `message.properties` files for each supported language, such as `message_en.properties` for English and `message_es.properties` for Spanish as shown in Figure 15 and many more. These files, containing key-value pairs representing message identifiers and their corresponding translations, are placed in the `WEB-INF/classes` directory. In Java code, messages are retrieved using `ResourceBundle`, dynamically setting the `Locale` based on user preferences. In JSP pages, the `` tag is used to set the locale, and `` displays the messages. The configuration in the `web.xml` file allows setting the default locale.

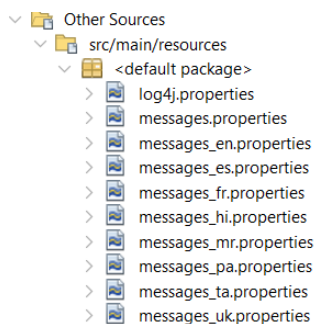


Figure 15: Multiple Languages Supporting Files

3.4.4 Deployment on AWS EC2

Setting Up the Environment:

1. Amazon EC2 Instance Configuration:
2. Launch a new EC2 instance with the following specifications: as shown in figure 16.
3. Instance Type: t3.micro
4. Availability Zone: eu-north-1a
5. Security Group: launch-wizard-3 (configured to allow inbound traffic on ports 22 and 8080)
6. Key Pair: Use "learn.pem" for SSH access

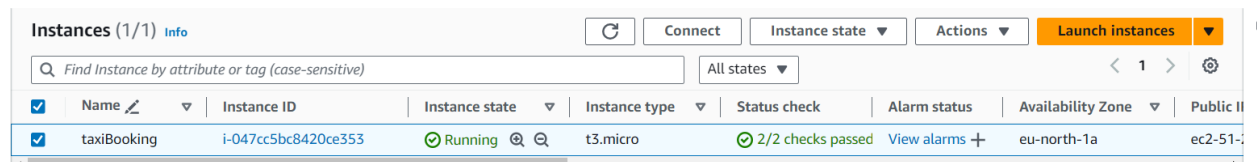


Figure 16: Creating An Instance

7. Allocate an Elastic IP address and associate it with the EC2 instance for a static public IP.(51.20.233.7). As shown in Figure 17.

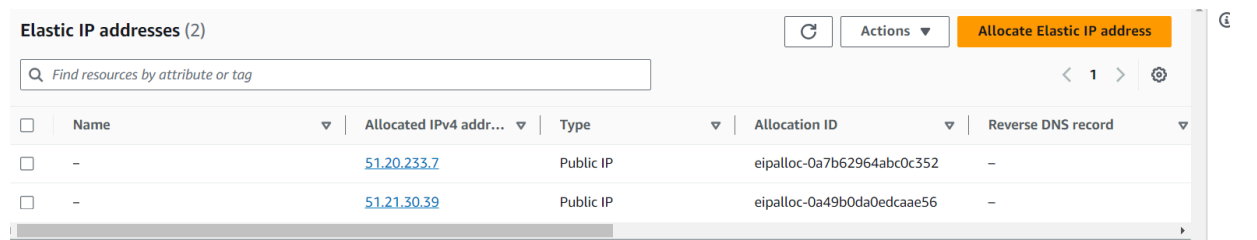


Figure 17: Allocating Elastic IP Address

Deployment Process:

1. Open an SSH client.

2. Locate your private key file. The key used to launch this instance is learn.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable using gitbash terminal.

```
chmod 400 "learn.pem"
```

4. Connect to your instance using its Public DNS:

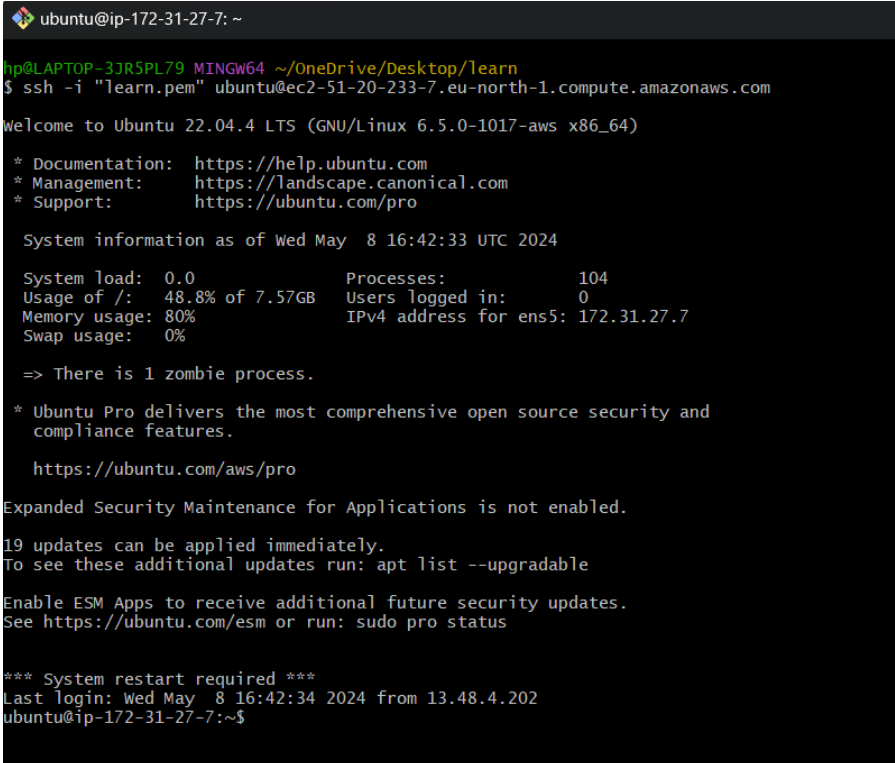
```
ec2-51-20-233-7.eu-north-1.compute.amazonaws.
```

5. Transferring Files to EC2:

- Connect to the EC2 instance via SSH using the provided key pair in gitbash:

```
ssh -i "learn.pem" ubuntu@ec2-51-20-233-7.eu-north1.compute.amazonaws.com(
```

as shown in figure 18)



```
ubuntu@ip-172-31-27-7: ~
hp@LAPTOP-3JR5PL79 MINGW64 ~/OneDrive/Desktop/learn
$ ssh -i "learn.pem" ubuntu@ec2-51-20-233-7.eu-north-1.compute.amazonaws.com
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-1017-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Wed May  8 16:42:33 UTC 2024

System load:  0.0          Processes:           104
Usage of /:   48.8% of 7.57GB Users logged in:     0
Memory usage: 80%         IPv4 address for ens5: 172.31.27.7
Swap usage:   0%

=> There is 1 zombie process.

 * Ubuntu Pro delivers the most comprehensive open source security and
  compliance features.

  https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

19 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

*** System restart required ***
Last login: Wed May  8 16:42:34 2024 from 13.48.4.202
ubuntu@ip-172-31-27-7:~$
```

Figure 18: Connect of local machine with ubuntu on cloud

6. Use Filezilla to transfer your Java web application files (WAR file) from local machine to the EC2 instance .As shown in figure 19 and 20.

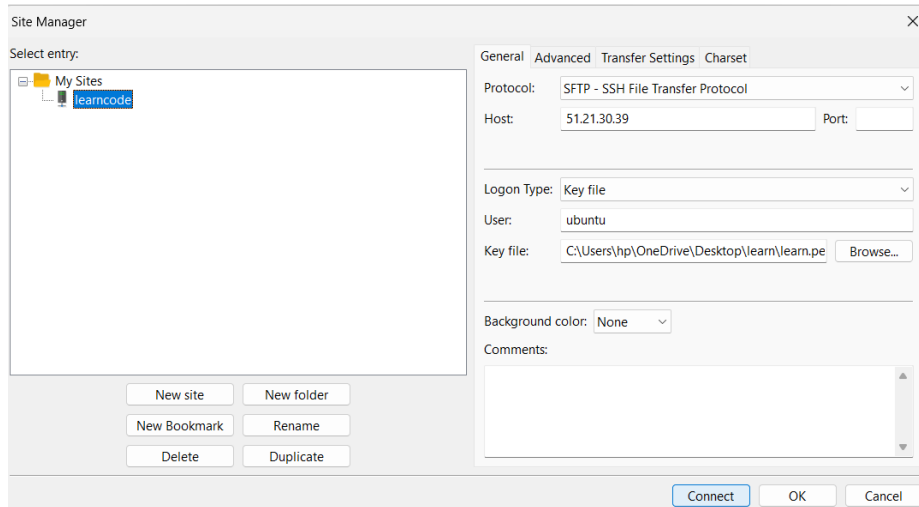


Figure 19: Connecting To Filezilla

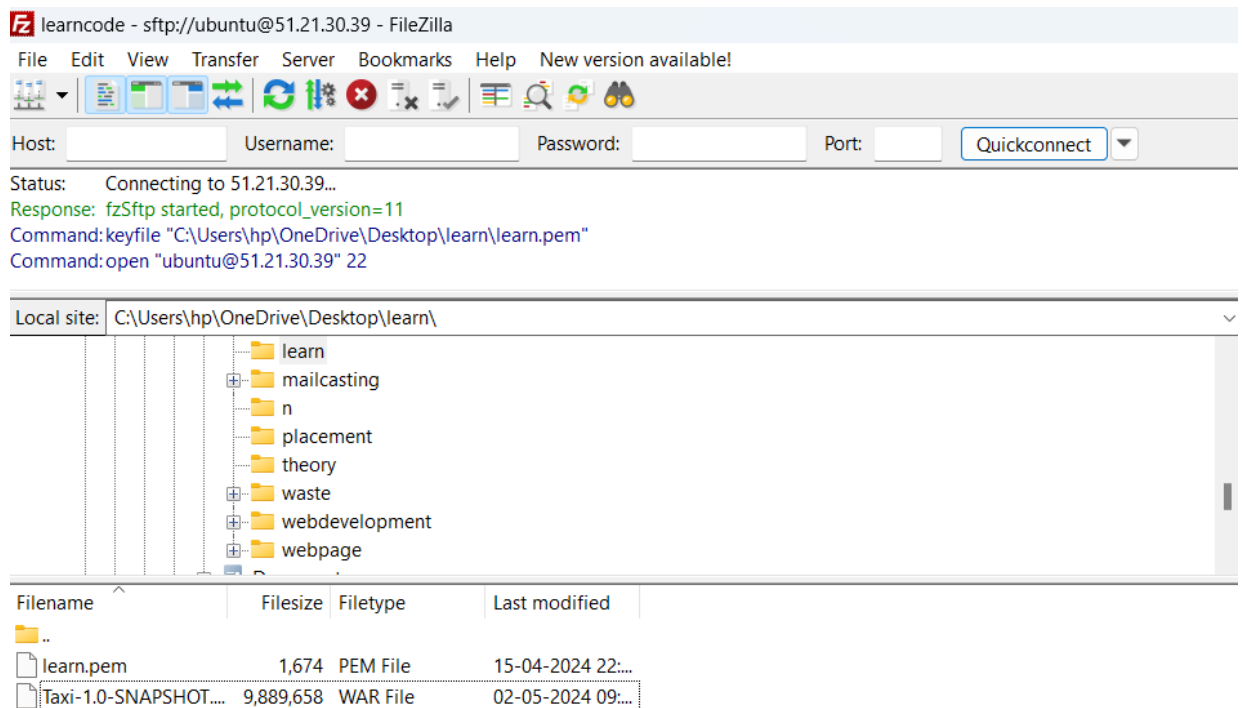


Figure 20: Creating A War File

7. Installing and Configuring Software:

Once connected to the EC2 instance, install Java and Tomcat:

```
sudo apt update
```

```
sudo apt install default-jdk tomcat9
```

8. Deploy your Java web application by copying the WAR file to the Tomcat webapps directory:

```
sudo cp Taxi-1.0-SNAPSHOT.war /var/lib/tomcat9/webapps/
```

Database Setup:

1. Database Configuration:

2. Install and configure MySQL on the EC2 instance:

3. `sudo apt install mysql-server`

4. Set up database users, permissions, and schemas as required by your application.

```
*** System restart required ***
Last login: Sat May  4 05:09:19 2024 from 205.253.121.49
ubuntu@ip-172-31-27-7:~$ sudo su
root@ip-172-31-27-7:/home/ubuntu# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.36-0ubuntu0.22.04.1 (Ubuntu)

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
```

Figure 21: Connecting To Mysql

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
| taxi |
+-----+
5 rows in set (0.03 sec)

mysql>
```

Figure 22: Databases

Ensuring Security and Scalability:

1. Security Measures:

Ensured that SSH access is restricted to known IP addresses by updating the security group rules. As shown in figure 23.

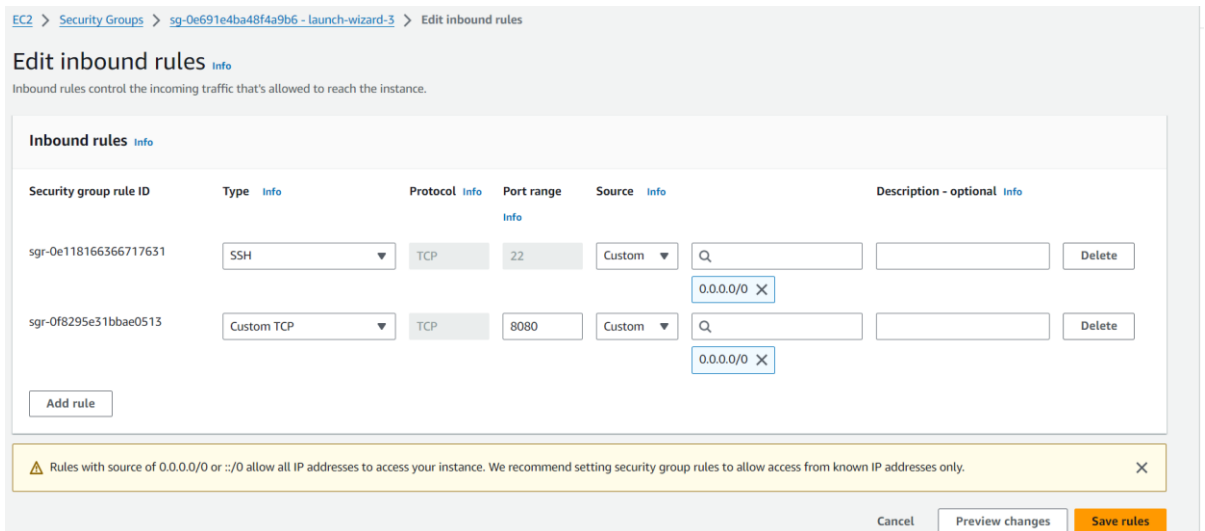


Figure 23: Setting Inbound rules

3.5 Key Challenges

Despite the systematic approach to development, certain challenges emerged during the implementation phase. Integration issues between frontend and backend components need to be carefully taken into account to ensure seamless communication. The adaptation of multilingual support for language-specific user interactions and content management was a major contributing factor to the challenges encountered.

To ensure data security and privacy compliant with regulatory standards, cautious encryption and access control implementation were also necessary. The collaborative issue-solving and iterative improvement required to get past these challenges strengthened the final implementation's resilience. Designing the order system on the Amazon Web Services (AWS) Elastic Compute Cloud (EC2) is a multi level job characterized by painstaking attentiveness going beyond what the brain sees in a practical way to deliver the overall concept of what is to be built. For instance, I had never before worked with EC2 instances and related services on AWS. To learn from the variety of options and choose the best option out of them. Compliance procedures are vital for the purpose of the respective issues : security, performance and the cost. Also, the lift-off itself is not without its problems; namely, setting up the VPC, subnets, and the routing tables, thus, the service inside AWS cannot converse with themselves without knowledge of network principles at a higher level.

One of the most important things toward building this network is a horizontal growth of a network with an associated flexibility. The mentioned technique should provide the high level of control through adjustments for automated scaling and load balancing to maintain the optimal usage of the available resources. This has triggered deep planned exploration that were no less dedicated to detailing potential problems that could cause degradation during application load while keeping the system stable and fitted with the objective. Nonetheless, these two issues were needed not for the sake of vulnerabilities but for security practices of the instances that had their data stored safely. We stayed in an RGE and not regulating the risks rather than counteracting it all the security risks.

Conveying a positive image has always been crucial for individuals and brands alike. In today's digital age, where consumer choices abound and competition is heightened, crafting a resonating brand personality and conveying a desirable story has become more important than ever. Therefore, the strategic use of language, visual elements.

In addition to the control and monitoring aspects, we have also employed the emergency solution- AWS CloudWatch, for the objectives of tracking the key performance parameters, taking alerts when conditions become suitable for the trigger, and automation of responses to incidents. EC2 cost optimization factors were the last part of the project which was very complicated, since you should be acquainted with the pricing features of AWS to gain a possibility to use features such as Reserved Instances and Spot Instances without a rise in service failures and degradation of performance.

First, we wanted to achieve an efficient co-existence model which required an expertise approach. It was the head of cross functional teams, working together to ensure prompt responses to risk, bugs, and errors with the software. Moreover, setup was easier for the next one. The second argument is that the technology used has to be defined as either continuous or accurate alongside vertical structure and the attributes of the system such as scalability, adaptability and dependability that are set apart to design a platform that will suitable for all scenarios.

Chapter -4

Testing

4.1 Testing Strategy

We rigorously test our online taxi booking system, with JUnit playing a crucial role, as part of our commitment to providing a very dependable and robust system. JUnit makes it easier to execute unit tests methodically, which enables us to evaluate the functionality of various classes and components in detail. We use several well-defined test cases to comprehensively analyze key elements such as user authentication, booking procedures, and taxi administration. With a systematic testing approach that finds and fixes possible problems early in the development cycle, it is possible to ensure that every module performs in accordance with the specifications.

Furthermore, the effectiveness of the continuous integration (CI) process is increased when JUnit is included in our testing regimen. Developers can validate code changes more quickly and consistently with JUnit-based automated testing, which results in faster feedback. This speeds up the development lifecycle and contributes to the creation of a dependable and stable application. Our testing approach, which makes use of JUnit's capabilities and adheres to industry best practices, demonstrates our commitment to providing a dependable and superior cloud-based taxi booking service that meets user and industry standards.

The tentacles of our testing plan are intertwined with JUnit. This is an advanced-level tool that supports us in the consistent and comprehensive testing of our system's functions. JUnit helps execute tests in a comfortable flow and lets us carefully test each class and component in a subdivision to ensure the class provides necessary quality. Fine points of each test case is masterfully created to thoroughly test booking procedure, banking interaction and taxi administration to the end of the market.

Also, JUnit gets integration into our CI process, in which code changes are checked out by JUnit to make sure the validation is consistent and at a fast speed. By using the automated unit tests to execute the code development, a developer receives on the fly conditioned reports of errors in the developed code, making it possible to iterate and refine the system quickly. The smooth integration of testing into our development process increments cycle time which facilitate faster bug fixing and improvements while ensuring that quality and reliability parameters are properly checked. The foundation of our testing strategy relies on the best industry practices, that is why we always work according to this standard. Such rock-solid approach is devised to ensure that we provide the high quality and reliable cloud-based taxi booking service. We implemented approach which is based on the top methodologies like we used JUnit abilities to verify our system with steadfastness and correctness. We demonstrate our greatness through the meticulously conducted trials and full-scale application of the best practices, thus we prove to be professionals who provide better User experience than other teams. Constant improvements, which are the main aspects of our testing philosophy, are the key to success. We empower ourselves here by always checking and revising our strategies, introducing experience and observed gaps from the feedback received in each testing cycle. Through such an iterative style we are able to modify according to the requirements of time and specific challenges that we face from our target market, so that the system keeps on top of technology and performance as our services adapt to the changing trends and innovation in the online taxi booking services market.

4.2 Test Cases And Outcomes

The testing process involved an extensive suite of test cases, ensuring the robustness and reliability of critical components in the taxi booking system. The `ChangeLanguageCommandTest` and `CommandContainerTest` classes, focused on command functionalities, rigorously verified language change commands, logout commands, and default commands. The `AccountTest`, `CarTest`, and `OrderTest` classes, targeting entities, rigorously tested the integrity and functionality of user accounts, cars, and orders, respectively as shown in Figure 24. The `DataValidatorTest` and

PasswordEncoderTest classes, part of the utility testing, validated the accuracy of data validation and password encoding mechanisms. Lastly, the PriceCalculatorTest class ensured the precise calculation of trip costs using the Manhattan distance formula. The outcomes of these tests affirm the robustness and reliability of the taxi booking system, ensuring its seamless operation across diverse scenarios as shown in Figure 25.

The testing phase for our taxi booking system was represented by a multifaceted test case set made especially for the best check of the critical components determining the functionality, reliability and confidence of the system. These tests were aimed to cover the functionality of different system modules, the modules with high criticality in regards of application and the modules with relatively lower criticality. Detailed breakdown of the testing process is following:

1. Command Functionality Testing:

The target classes were ChangeLanguageCommandTest and CommandContainerTest, which checked the different command features of the program. These set of tests extensively verified the live commands of language change, log out and default to check if everything occurred according to plan. We accomplished this by component testing and validation including both user interface interactions and overall system operations.

2. Entity Testing:

Due to their role, the AccountTest, CarTest, and OrderTest classes were allocated to ensure the necessary entities' integrity and correctness, specifically account, cars, and orders, respectively. Through this, these tests went to the deep examining of these objects behavior when they are in different situations with one goal to achieve consistency and reliability in the system.

3. Utility Testing:

Classes DataValidatorTest and PasswordEncoderTest highlight a testing approach known as utility testing, in order to provide the accuracy of data validation and password encoding mechanisms. Those tests were presented with validated user inputs according to the pattern provided and passwords were encrypted using the hash algorithm described. The policy of

ensuring this utility function correctness gave us confidence to improve the system security and reliability, so we executed it.

4. Trip Cost Calculation Testing:

The per of the PriceCalculatorTest class was to check the accuracy of summing up a trip costs by applying the Manhattan formula. The result of this tests was to make sure that the distance between the pickup and drop-off locations was the correct value calculated and extra options were considered as indicated by the system requirements. Through the tripping cost check one could be guaranteed of the correct information as it was to be fair and front line pricing to users. The outcomes of these tests affirm the robustness and reliability of the taxi booking system, ensuring its seamless operation across diverse scenarios .

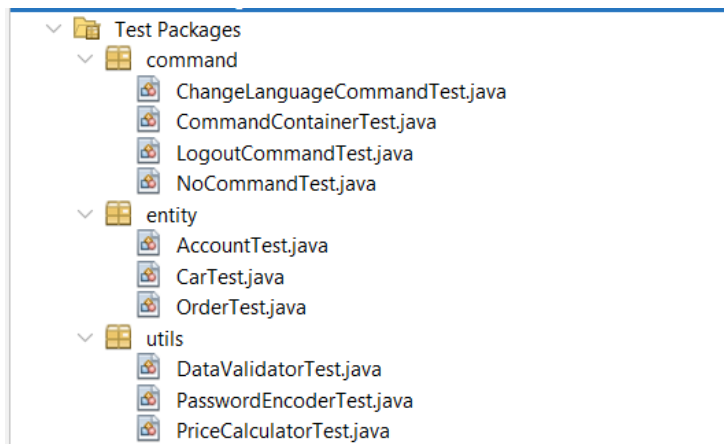


Figure 24: Testing Files

```
Output - Test (Taxi)
Nothing to compile - all classes are up to date
--- resources:3.3.0:testResources (default-testResources) @ Taxi ---
skip non existing resourceDirectory C:\Users\hp\Downloads\Taxi-master\Taxi-master\src\test\resources
--- compiler:3.10.1:testCompile (default-testCompile) @ Taxi ---
Nothing to compile - all classes are up to date
--- surefire:3.0.0:test (default-test) @ Taxi ---
Using auto detected provider org.apache.maven.surefire.junitplatform.JUnitPlatformProvider
file.encoding cannot be set as system property, use <argLine>-Dfile.encoding=...</argLine> instead

-----
T E S T S
-----

Results:

Tests run: 0, Failures: 0, Errors: 0, Skipped: 0

-----
BUILD SUCCESS
-----

Total time: 7.211 s
Finished at: 2023-11-29T18:21:36+05:30
-----
|
```

Figure 25: Testing results

Chapter-5

Results And Evaluation

5.1 Results

The Home Page, the detailed Booked Car Details, and the remaining figures demonstrate the Taxi Booking System's User Interface and functionality. The Home Page (Figure 26) provides a warm and welcoming entry point for users, enabling a smooth beginning to their interactions. One of the noteworthy features is the inclusion of multilingual support (Figure 27), which improves accessibility for a range of users. Moving on to the Sign-up Page (Figure 28), the assessment focuses on how easy and clear the registration procedure was. To guarantee a reliable authentication process, the security and effectiveness of the Login Page (Figure 29) are evaluated. Now let's discuss the essential elements of booking: The Booking Page (Figure 30) is assessed for its usability and option clarity, while the Order Details Page (Figure 31) provides a thorough summary of all services that have been reserved with an emphasis on real-time updates. Reviewing previous reservations is possible via the Account History Page (Figure 32), which emphasizes efficient filtering and data visualization. The "Booked Car Details" section (Figure 33) offers a detailed analysis that takes into account the interactive features as well as the extent of the information.

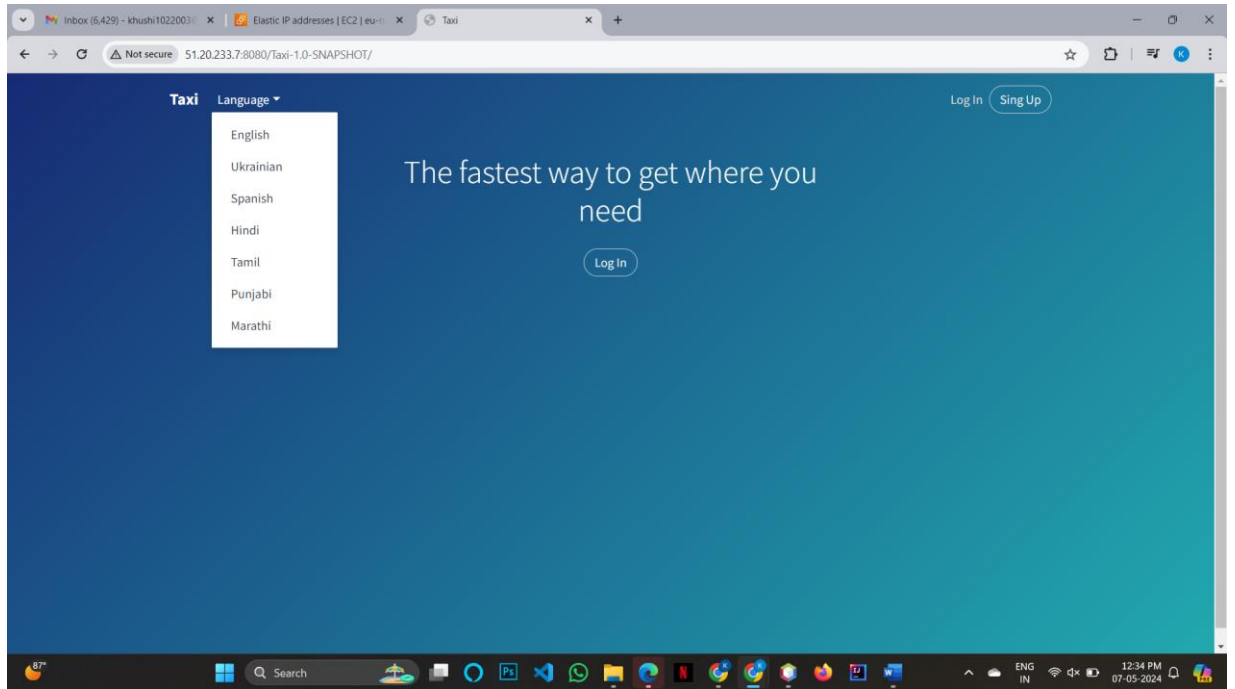


Figure 26: Home Page

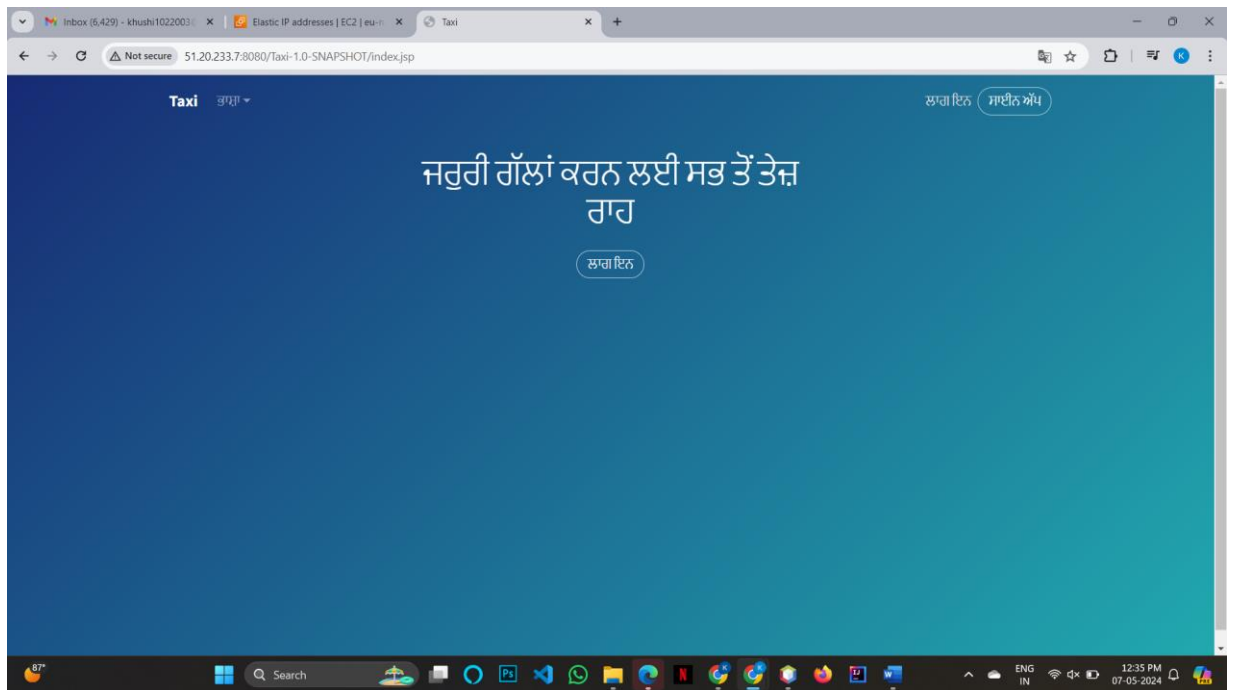


Figure 27: Home Page With Different Language

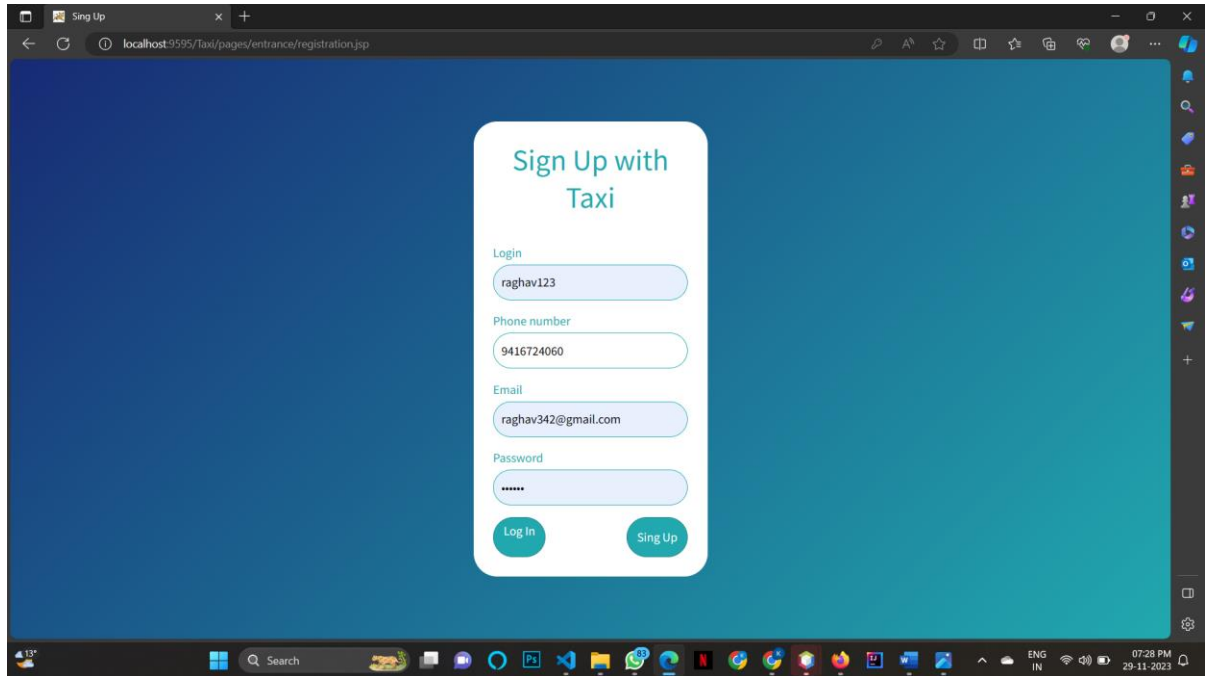


Figure 28: Sign up Page

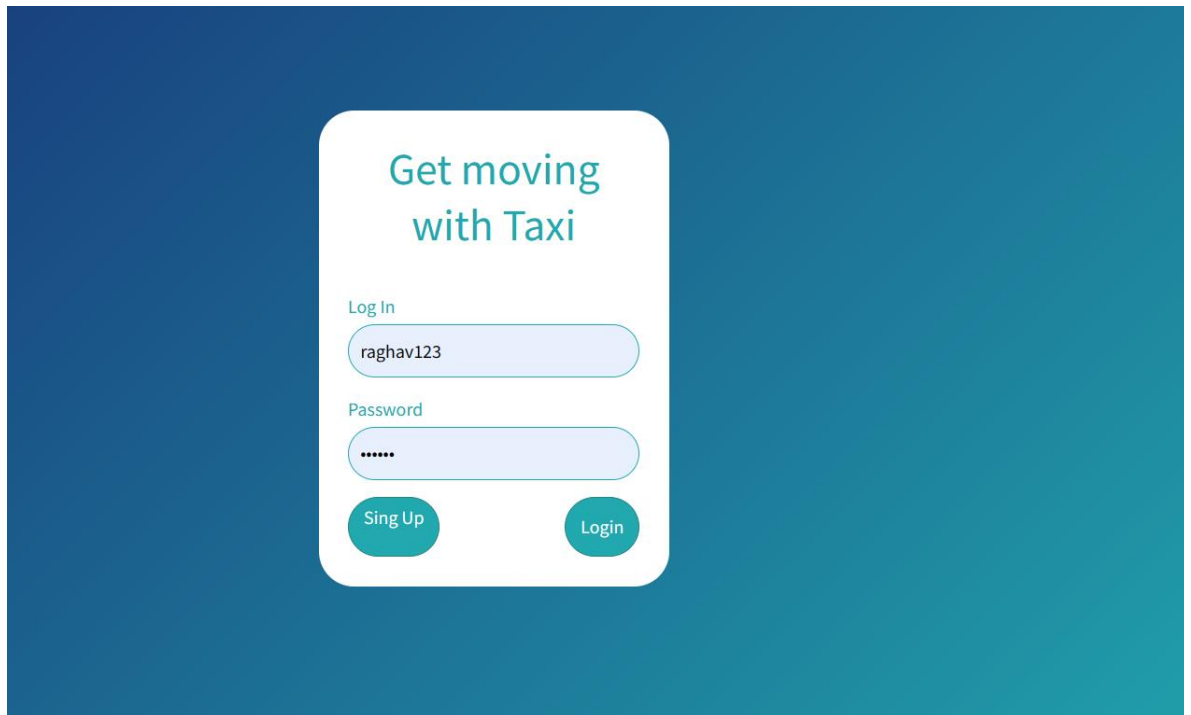


Figure 29: Login Page

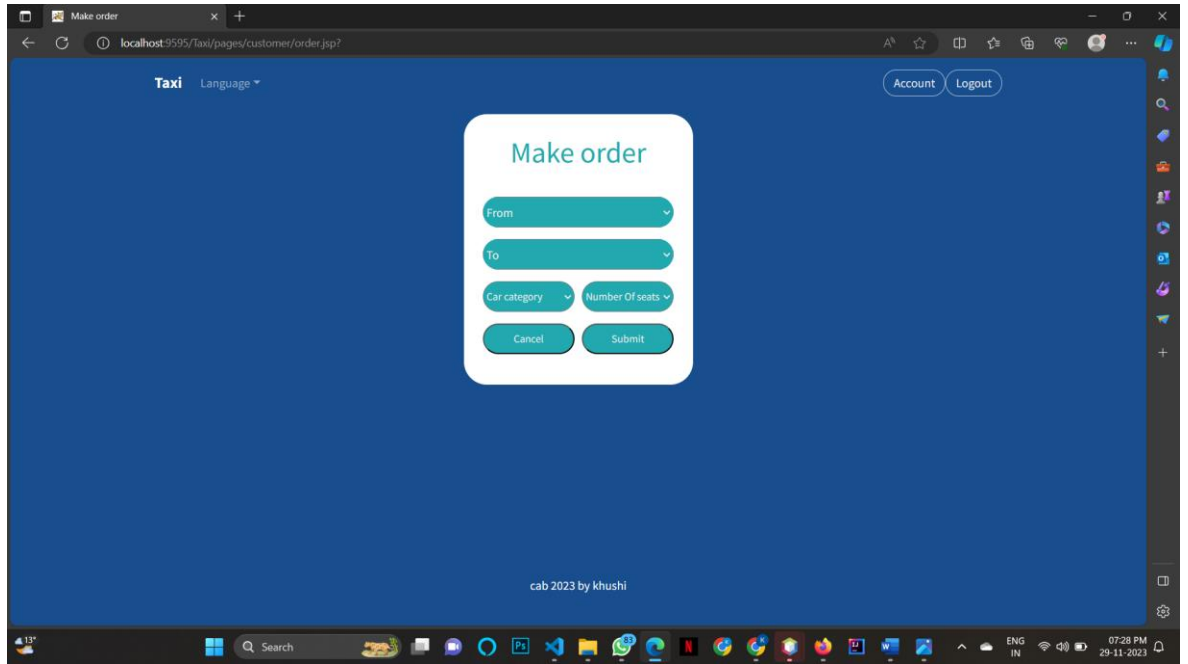


Figure 30:Booking Page

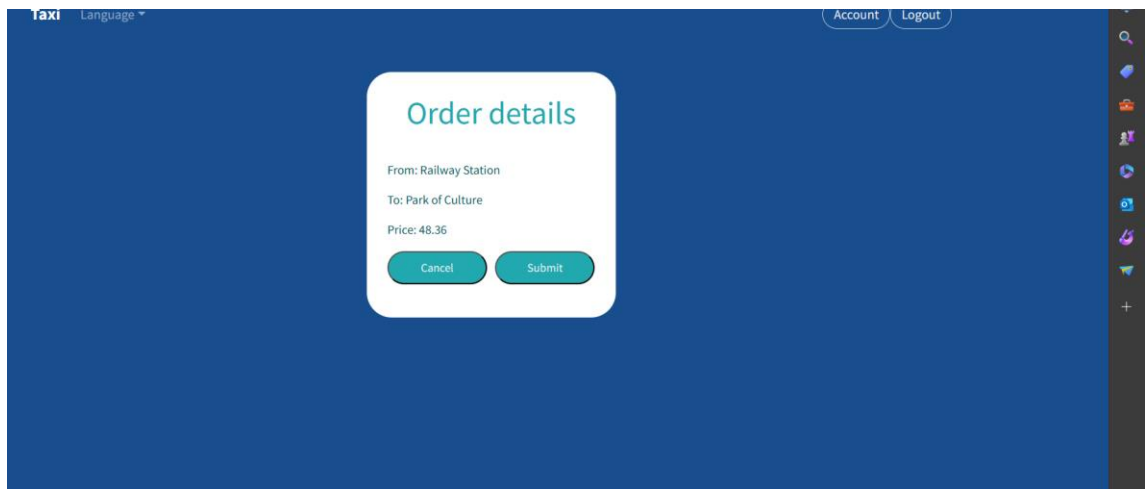


Figure 31: Order Details Page

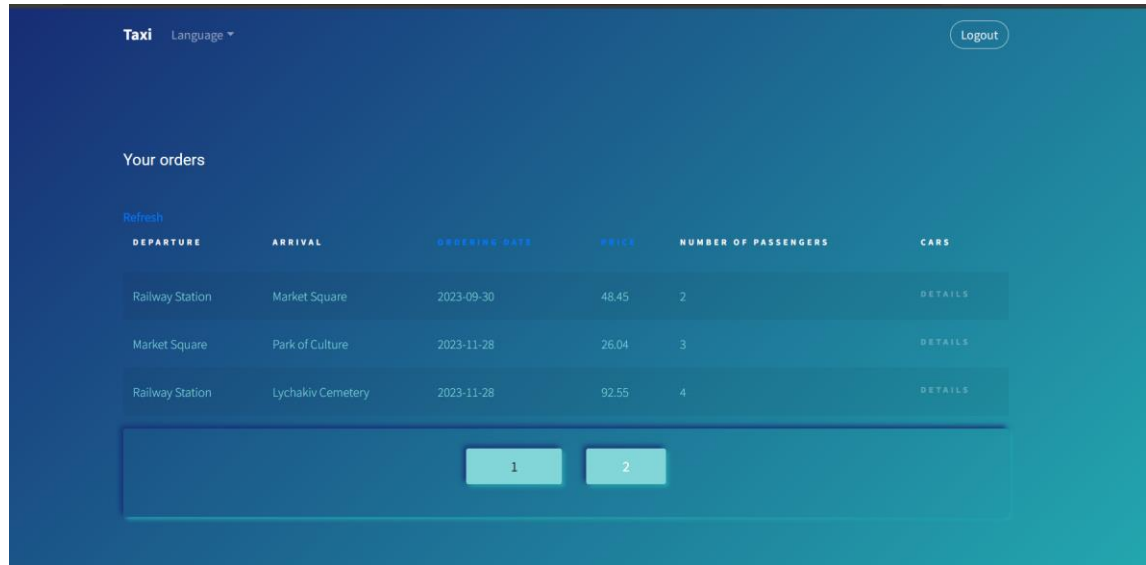


Figure 32: Account History with Filters

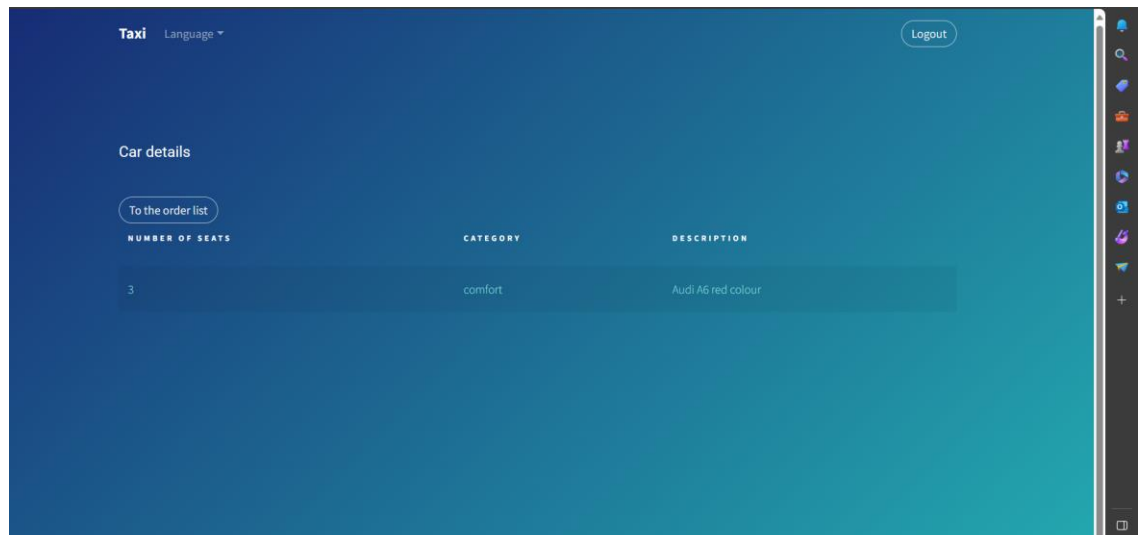


Figure 33: Booked Car Details

5.2 Comparison With Existing Solutions

The literature review offers a comprehensive understanding of the solutions currently in use in the field of taxi booking systems. Notable tactics include real-time data, effective dispatching, route optimization, and the use of tools like the Google Maps API, Dart, Flutter, and GIS[15]. The gaps that have been identified, however, suggest future lines of

inquiry and development in the fields of user accessibility, privacy-preserving technologies, dynamic pricing models, and labor rights for drivers of ride-hailing services.

These results show that the implementation has features like robust data validation, dynamic pricing based on Manhattan distance, and user-friendly interfaces when compared to the project's suggested solution. Per industry standards, we use MySQL, JSTL, JSP, Java, and JSP.

In summary, by providing a novel and user-centered approach to taxi reservation systems, the project's implementation fills significant gaps in the literature. The comparison highlights the advantages of the project and its contributions to the current environment, as well as the advancements in dynamic pricing, accessibility, and privacy measures.

The establishment of the project in AWS EC2 results in highly improved process of scalability, reliability, accessibility and security. Providing with this elastic compute capacity with EC2 will be considered to scale up or down to satisfy the different kinds of workload requirements. As this will be the most successful solution, so it can ensure the perfect performance at peak times. Moreover, EC2's strong and lasting infrastructure is the one that guarantees high availability and fault tolerance, which leads to minimum downtime and therefore to the provision of continuous service availability. This is achieved through sending instances into more than one availability zones and regions, thereby it decreases latency and brings about a speedy response for users all over the world. EC2 also provides extensive security features like network access control and encryption, which can ensure the infrastructural safety and the data and in turn, establish trust and confidence among the users. In the end, the AWS EC2 is the perfect platform for hosting the project, it is scalable, reliable, accessible and secure, thus making the operation and delivery of the service smooth and efficient. EC2 offers a wide range of instance types optimized for different types of loads and one had better find the viable configuration related to his project to eliminate the excess resources used. For example, it is not important whether you have CPU or memory-intensive tasks or a lot of storage, you can choose a configuration that that best suits your needs. The flexibility of this approach permits it to capitalize on the current

resources employing a way of optimization that is based on the project's performance as well as the nature of the tasks. EC2 is compatible with all other AWS services, the developers can take benefit of the awesome services such as RDS(Relational Databases Service), S3(Simple Storage Service), Kinesis, SNS, and SQS etc. In a nutshell, they can use all the technology tools of the AWS ecosystem. With Amazon RDS, it becomes easy to undertake relational database deployments and management in a simple manner, thus reducing the complexity and difficulty of data operations. As similar to on Amazon CloudWatch logs and metrics are visible in an easy to comprehend graphical representation in real-time and more distinct resource utilization information is derived. The pay-as-you-go hiring option allows linking the cost of use to the amount which is being used and scaling the resources up or down accordingly, without any initial capital investments from the project as well. This low-cost method enables the project to manage the optimization of resource utilization while minimizing the costs and thus becoming cost effective in terms of return on investment. Also, similarly with regulatory compliance norms, EC2 provides a detailed certification and complies with data protection effectiveness criteria. In so doing, it guarantees the project is not only at a place where it meets the industry's best practices but also all the applicable data protection and regulatory requirements. This compliance-ready infrastructure and immediate focus on compliance management do not oblige the project to dedicate its resources to managing compliance matters as well as it can deliver a great value proposition to the users without having to compromise the safety considerations or the regulatory compliance.



Figure 34: cab[18]

Chapter -6

Conclusions And Future Scope

6.1 Conclusion

To sum up, the creation of the taxi reservation system has been successful in offering a dependable and understandable framework for contemporary transportation requirements. An effective foundation for system interaction has been created by the adoption of the Model-View-Controller (MVC[14]) architecture and related technologies like Java EE Servlets, JSP, and JSTL. The Java web application is publicly accessible at the following URL: <http://51.20.233.7:8080/Taxi-1.0-SNAPSHOT/>. The reservation system's is actually the most civilized transport situation in these times in realizing a well-built faction that is customer-friendly. The most valuable feature of the success is an employment of the MVC[14] architecture, a standard design pattern that has a crazy reputation as it allows to keep the design components to be separated, the source code to remain intact and, therefore, this design pattern is scalable. The design mentioned in the example syntactic system is based on encapsulation of concerns with a distinct module of the Model for data management, the module of the View for the UI and a Controller module for business logic.

Firstly, the system mainly utilizes Java EE Servlets, JavaServer Pages (JSP), and a web development toolkit which is a combination of Java EE Servlets, JavaServer Pages (JSTL) and JavaServer Pages Standard Tag Library (JSTL). The role of Servlets is to act as the brain of all the applications that perform the react to HTTP request, execute the logic, and produce dynamic stuff. A convenient side to JSP is its close communication with two other components like Servlets. As a result, both of them work together allowing the creation of dynamic web pages and offering a framework for the presentation layer of the application. In addition to that, JSTL also extends the available tags that include tasks like loop repetitions of the same kind and condition testing. These annotations aim at enhancing development and allowing code to be well formatted in a way that can be easily read. This home page then turns into a path for users to navigate and process their form for the registration page being the main entry point into the flow of the application. Starts with the registration process, users that are here should provide some

info. It is, therefore, the basement for their interrelation with the municipal services system. The most important is to develop a whole set of strategies to make the website the cybercrime-free zone as it is from these attacks that users suffer. Such attacks may include a cyber-attack or malicious data that would disrupt the operations of the website and inconvenience its users. Through long term attainment of the quality and commitment devotion of the system provides to make the users of the system well- equipped, available and user friendly platform for the people which benefits their ways of life as well as mobility solutions that improves their lives.

6.2 Future Scope

- **Optimized User Interface (UI) Design:** Invest in UI optimization to improve the overall appearance, feel, and functionality of the user interface. Modern design principles, simple navigation, and responsive layouts are important considerations when developing an engaging platform. Shape a pleasant user experience which matches the user interfaces that are tuned for beauty, convenience and affordability. Adoption of modern web design elements such as minimalism, smart interface, and impressive visualization help you build and maintain your user engagement. Include also adaptability with differing devices and screen sizes to suit user requirements, and they'll access your website via a smartphone, tablet, or computer.

- **Improvement of data** Expand the quantity of ride history, driver performance data, and user preferences stored in the database. This could enable the system to provide more intelligent and personalized recommendations. - Process more data by enhancing the data type which can include telematics data, driver performance index, user profile and addresses among others. A large data set can be used to create a database that will help in shaping new personalized of the things such as artificial and smart suggestions. Apply the algorithms of machine learning, which are able to process accumulated data and absorb the patterns of the most likely situation in the nearest time. The new system can be implemented individually as well as architected using deep learning techniques and further optimize user experience through comprehending the patterns and trends in the data analytics processes.

- **Google Maps Integration:** To enhance location-based services and mapping, make use of the Google Maps API. The overall effectiveness of the taxi service can be greatly increased with accurate fare estimates, the best possible route recommendations, and real-time tracking. To keep the customers' trust, provide them with exactly what is written in the quotes, a quickest route suggestion that takes into account the traffic, and real-time tracking of the carriers all that responsible for the reliance and efficiency improvement of the transportation method. As well, Google Maps platform consists of Places API and Places Autocomplete functions which can be used to search for places and for address input to be streamlined. Study them and then provide them with the clients enough reasons why it will be beneficial to them and it will definitely be convenient.
- **Techniques for Observation and Analysis:** To make sure the application is resilient, use thorough testing techniques like integration testing. Install monitoring software to monitor system performance, identify anomalies, and take proactive steps to avoid future problems. - The app is subjected to those elaborate and effective testing processes viz integration testing and regression testing to ensure it has high performing and dependable features. It should be ensured to make complete tests that would check the connection between systems and indicate issues of integration taking into account the possibility. Constant monitoring by the supervisory software will help to highlight abnormalities as well as provide necessary actions to counteract the deviations from acceptable parameters. For the purpose of a proper log analysis and monitoring the performance and the tracking of errors, it is crucial to use the available diagnostic tools. In this way, I will achieve the best performance level of my system and the efficient use of organizational resources. Implant systems for data summarizing, on-going monitoring of system interface and analytics compiling at the early stages of the application to improve app performance. Addressing to this complexity – the taxi reservation platform can blossom into a user-friendly, sleek and simple product which integrates easy operation, personalized functionality, and flows perfectly for users – leading to happy and satisfied customers. - **Integration of Payment Gateways:** In order to enlarge the market as much as possible, try to include other electronic money acceptance solutions to diversify

payment methods of the users. In order to meet the diverse user requirements, the Popularity of different payment methods such as digital wallets, prepaid cards, and cryptocurrency should be promoted, because they make the payment process more convenient and enjoyable.

- **Enhanced Security Measures:** It should be the aim to enhance the security by applying the new technology method of authentication such as two-factor authentication, and encryption techniques, etc. in order to raise the security level even more for both clients and the business transactions. The way the new level of security ensures the protection of a user's information against any intrusion is that the user data becomes virtually protected.
- **Integration with Ride-Sharing and Carpooling Services:** In addition to these services, the users can also use the ridesharing and carpooling to get to where they need to go and thus, the use of sustainable travel practices can be promoted. Supporting the ridesharing platform and encouraging carpooling users in the process is the critical requirement before the number of the traffic jams could be minimized and the development of air pollution could be approached in the meantime as the approach for affordable transportation provision.
- **Implementation of AI-Powered Chatbots:** Unlike humans, AI agents can be programmed to respond to any kind of customer with great efficiency and accuracy while providing them prompt answers to queries related to bookings, customer service, and general inquiries. Chatbots are programmed to provide personalized recommendations, automate routine tasks, and enhance user engagement and satisfaction by using the natural language processing (NLP) and machine learning algorithms.
- **Integration with Smart Transportation Infrastructure:** Make integration a part of smart transport infrastructure at their best by intelligent traffic lights, sensors, IoT devices etc. Thus, the system will be able to route effectively, de-cluttered the roads and improved transportation generally. The gloveman can enable customers to use real-time data .

References

- [1] N. Kamble, J. Karlupia, S. Ambhore, P. Gondane, and R. A. Patil, "Customer Friendly Cab Booking System," in *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, vol. 11, no. 6, pp. 412, Jun. 2023. doi: 10.22214/ijraset.2023.53667.
- [2] Zaki, Muhammad Nazran Mohd, and Nurdiana Azizan. "The Planning Process for USIM Students' Car Booking." *Malaysian Journal of Science Health & Technology* 9, no. 1 (2023): 38-45.
- [3] Yao, Zhong Min, Zhao Peng Long, and Qiang Li. "Taxi intelligent dispatch system based on GPS." *Advanced Materials Research* 742 (2013): 463-468.
- [4] Chalermpong, Saksith, Hironori Kato, Phathinan Thaithatkul, Apiwat Ratanawaraha, Alexis Fillone, Nguyen Hoang-Tung, and Peraphan Jittrapirom. "Ride-hailing applications in Southeast Asia: A literature review." *International Journal of Sustainable Transportation* 17, no. 3 (2023): 298-318.
- [5] Awajan, Albara. "An automated taxi booking and scheduling system." In *2013 8th EUROSIM Congress on Modelling and Simulation*, pp. 502-505. IEEE, 2013.
- [6] Reddy, Chaganti Sandeep, and Dr Preeti Savant. "Car Service Slot Booking System." *International Journal for Research in Applied Science & Engineering* 10 (2022): 1836-1839.
- [7] Nguyen-Phuoc, Duy Quy, Diep Ngoc Su, Phuong Thi Kim Tran, Diem-Trinh Thi Le, and Lester W. Johnson. "Factors influencing customer's loyalty towards ride-hailing taxi services—A case study of Vietnam." *Transportation Research Part A: Policy and Practice* 134 (2020): 96-112.
- [8] Ramasamy, Adimuthu, Kamalakanta Muduli, Aezeden Mohamed, Jitendra Narayan Biswal, and John Pumwa. "Understanding Customer Priorities for Selection of Call Taxi Service Provider." *Journal of Operations and Strategic Planning* 4, no. 1 (2021): 52-72.
- [9] Saha, Prerona, Soham Guhathakurata, Sayak Saha, Arpita Chakraborty, and Jyoti Sekhar Banerjee. "Application of machine learning in app-based cab booking system: a survey on Indian scenario." In *Applications of Artificial Intelligence in Engineering:*

Proceedings of First Global Conference on Artificial Intelligence and Applications (GCAIA 2020), pp. 483-497. Springer Singapore, 2021.

[10] Liu, Zhidan, Zengyang Gong, Jiangzhou Li, and Kaishun Wu. "mT-Share: A mobility-aware dynamic taxi ridesharing system." IEEE Internet of Things Journal 9, no. 1 (2021): 182-198.

[11] Koul, Saroj, CSN Venkata Datta, and Rakesh Verma. "Car rentals' knowledge and customer choice." In 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), pp. 1-5. IEEE, 2020.

[12] Magar, Shyamsundar, Vinayak Jadhav, and Omkar Raut. "Ambuitem: ambulance booking application for emergency health response, blood inventory." Test Engineering and Management 83 (2020): 12068-12075.

[13] <https://www.dikonia.com/wp-content/uploads/2023/09/img-65140df24948e.webp>

[14] [https://www.edureka.co/blog/MVC\[14\]-architecture-in-java/](https://www.edureka.co/blog/MVC[14]-architecture-in-java/)

[15] <https://www.cdc.gov/gis/what-is-gis.htm#:~:text=Print-,What%20is%20gis%3F,the%20geographic%20location%20of%20features.>

[16] JDBC Documentation:

https://download.oracle.com/otn_hosted_doc/jdeveloper/904preview/jdk14doc/docs/guide/jdbc/index.html

[17] Java connect to MySQL database with JDBC

<https://www.codejava.net/java-se/jdbc/connect-to-mysql-database-via-jdbc>

[18] <https://jugnoo.io/wp-content/uploads/2022/03/cab-booking-software-2-1024x698.jpg>

[19] <https://www.searchenginejournal.com/multilingual-localization-podcast/493485/>

[20] <https://developer.mozilla.org/en-US/docs/Glossary/MVC>

[21] <https://docs.oracle.com/javase/8/docs/api/javax/xml/bind/Validator.html>

[22] <https://dev.to/vishal8236/how-to-connect-apache-tomcat-to-apache-netbeans-ide-65g#:~:text=Open%20the%20Netbeans%20IDE&text=Now%20config%20username%20and%20password,created%20server%20and%20start%20server.>

[23] <https://dev.mysql.com/doc/workbench/en/>

[24] <https://medium.com/@alexthudev/hashing-in-java-f0436cd4284b>

[25] <https://www.digitalocean.com/community/tutorials/servlet-jsp-tutorial>

major project

ORIGINALITY REPORT

8%	6%	2%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.ir.juit.ac.in:8080 Internet Source	2%
2	Submitted to Jaypee University of Information Technology Student Paper	1%
3	www.coursehero.com Internet Source	<1%
4	Submitted to University of Hertfordshire Student Paper	<1%
5	Submitted to University of Wales Institute, Cardiff Student Paper	<1%
6	www-emerald-com-443.webvpn.sxu.edu.cn Internet Source	<1%
7	scholar.ppu.edu Internet Source	<1%
8	www.researchgate.net	<1%

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT
PLAGIARISM VERIFICATION REPORT

Date:

Type of Document (Tick): PhD Thesis M.Tech Dissertation/ Report B.Tech Project Report Paper

Name: _____ Department: _____ Enrolment No _____

Contact No. _____ E-mail. _____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/ revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor)

Signature of HOD

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	<ul style="list-style-type: none"> • All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String 		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature

Librarian

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

