# Hi Drive ( An online web application to hire drivers)

Akshat Singh
Student, Department of Computer Science
KIET Group of Institutions
Ghaziabad, India
akshat.2024cs1139@kiet.edu

Aman Verma
Student, Department of Computer Science
KIET Group of Institutions
Ghaziabad, India
aman.2024cs1128@kiet.edu

Anmol Patel
Student, Department of Computer Science
KIET Group of Institutions
Ghaziabad, India
anmol.2024cs1085@kiet.edu

Amit Kumar Singh Sanger
Professor, Department of Computer Science
KIET Group of Institutions
Ghaziabad, India
amit.sanger@kiet.edu

## ABSTRACT

**In India, many people nowadays prefer to book drivers for their rides instead of auto-ricksha or taxis. There are several applications available for this purpose, but they typically use a central server to store and manage data. The problem with this approach is that if the central server fails, the entire system stops working. Our idea is to create a driver booking system that uses a different method, called a server-based approach, to maintain the safety of passengers. Additionally, this system aim to monitor driver's behaviour using an accelerometer device. In our study, we have designed and built an intelligent server-based driver system that serves passengers by using local information. We implemented and tested this approach using a framework that runs on web browsers. The results of our simulations show that our approach can overcome the problems of the existing system and provide a more reliable and safer way for passengers to book drivers.**

## I. INTRODUCTION

In recent years, technology has significantly advanced across various industries, playing a crucial role in human commerce. In the realm of commerce, numerous applications and websites have emerged on the internet, simplifying our lives. Among these innovations, there are several platforms that offer on-demand driver services to customers whenever they require them, such as Drive4U, Hire4drive: Car Drivers and Cabs, Swift partners, and Hop-on demand driver. While these applications provide convenient and high-quality services to customers, there remain certain issues in the current system. The current system is not transparent in how it handles customer interactions. Problems include difficulty in locating the customer and the customer's inability to track the driver's location and estimated arrival time. These issues can result in various challenges for the customer.[1]

To address these concerns, this system intend to enhance and improve the existing system by implementing features that make it easier for customers to find drivers in their vicinity. This upgrade will reduce wait times and minimize the difficulties associated with pinpointing the driver's location.[1]

## II. OBJECTIVE

The objective of this project is to develop a web application for booking drivers for any purpose. [1] The designed system consists of:

- A client web application for customers.
- A web application for drivers.
- A server with a database.

## III. SCOPE OF PROJCT

In the current system, people often have trouble finding drivers for their own cars, so they end up renting a car for their travels. To solve this problem, we're creating an app called "HI! Drive" This app will allow customers who own a car but can't drive it themselves to easily find and hire drivers nearby. It's a convenient method for them to secure a driver whenever they require one.

## IV. MOTIVATION

Our system will be safer and easier to use. It will keep your data secure on the server. We'll also have tools to manage driver information and monitor their activities. The admin can check out statistics to understand how the drivers are driving. This system uses a smart method called the nearest neighbour algorithm, which makes it really good at finding drivers for you. It's a powerful way to search for the right driver.

## V. SYSTEM ANALYSIS

### *A. EXISTING SYSTEM:*

In the existing setup, customers can access a range of services, yet there are noticeable hurdles. Feedback from users highlights concerns about transparency, driver scheduling issues, inaccuracies in location tracking, and challenges with driver verification. Proposed system resolved these problems.

### *B. PROPOSED SYSTEM*

In our new system, you can easily book a driver from the comfort of your home by entering some information in an online web application. Once you start your journey with the driver, when it's over, you can pay conveniently with cash, card, or net banking. Our system does more than that too. It keeps an eye on where the car is and how fast it's going. It also keeps a record of all the drivers and listens to what customers say about their rides.[2] It even offers special deals to drivers and customers based on certain conditions. Our system mainly focuses on two things: booking a driver and making sure our customers are safe. We keep your details and order information so we can keep track of everything. We have set rules that help us do things automatically, which reduces the need for manual work and calculations. Here are some good things about our system:

- It's easy to use.
- It's fast.
- It saves time and effort.
- You can search for drivers that are nearby.

### *C. SYSTEM ARCHITECTURE*

The system architecture for this project is designed to provide a robust and scalable framework for the driver hiring web application. At its core, the architecture follows the principles of the MERN stack, comprising MongoDB, Express.js, React.js, and Node.js.

On the client-side (frontend), React.js is utilized to build dynamic and responsive user interfaces, providing an intuitive experience for users interacting with the application. State management is facilitated by Redux, ensuring centralized and efficient data handling.[6]

On the server-side (backend), Node.js with Express.js serves as the foundation for building RESTful APIs.[10] These APIs enable seamless communication between the frontend and backend, handling user requests, authentication, and data processing. JSON Web Tokens (JWT) are employed for secure user authentication, ensuring the protection of sensitive user data.

For data storage, MongoDB, a NoSQL document database, is utilized to store user information, driver details, hiring requests, and other relevant data. The schema-less nature of MongoDB allows for flexibility in data modelling, accommodating the evolving needs of the application.[7]

Additionally, the architecture includes mechanisms for real-time communication and geolocation functionalities. WebSocket technology enables real-time updates and notifications, enhancing the user experience. Geolocation algorithms and services are integrated to facilitate location-based features such as driver tracking and route optimization.

Deployment and hosting are carried out on cloud platforms such as Heroku or AWS, ensuring scalability, reliability, and accessibility. Continuous integration and deployment (CI/CD) pipelines are employed to automate the deployment process, streamlining development workflows and ensuring efficient delivery of updates and changes.

Overall, the system architecture provides a solid foundation for building a feature-rich and scalable driver hiring web application. By leveraging the strengths of the MERN stack and incorporating modern technologies and best practices, the architecture enables the development of a high-performance and user-friendly application that meets the needs of both users and administrators.

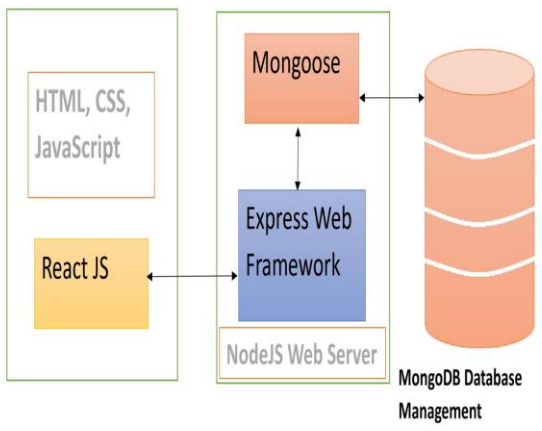

Figure: 5.1 (System Architecture)

## VI. DESIGN

### *A. INPUT DESIGN*

In this web application, the input design focuses on creating intuitive and user-friendly interfaces for users to interact with the system. The input design encompasses various elements, including forms, fields, buttons, and other interactive components, aimed at facilitating smooth data entry and interaction.

1. Form Layout: The input design begins with structuring form layouts in a logical and organized manner, ensuring ease of navigation and comprehension for users. Forms should be divided into sections or steps, with clear headings and labels to guide users through the data entry process.

2. Input Fields: Each input field should be appropriately labeled to indicate the type of information required. Use placeholders or default values where applicable to provide additional context or examples. Employ input validation techniques to ensure the accuracy and completeness of user inputs, minimizing errors and streamlining data entry.

3. Dropdowns and Selections: Utilize dropdown menus, checkboxes, radio buttons, and other selection controls where users need to choose from predefined options. Group related options together

and provide descriptive labels to enhance clarity and usability.

4. Date and Time Pickers: Implement date pickers and time selectors for inputting date and time information. Ensure consistency in date formats and provide visual cues to indicate valid date ranges or time slots.

5. Buttons and Controls: Design buttons and controls with clear labels and appropriate styling to convey their purpose and functionality. Use contrasting colours or visual cues to distinguish primary actions (e.g., submit, save) from secondary actions (e.g., cancel, reset).

6. Error Handling: Incorporate error messages and feedback mechanisms to notify users of any input errors or validation failures. Provide descriptive error messages that indicate the nature of the error and suggest corrective actions.

7. Mobile Responsiveness: Design input elements to be responsive and adaptable to different screen sizes and devices. Optimize input layouts for mobile devices, prioritizing usability and ease of interaction on smaller screens.

By focusing on input design, the driver hiring web application can offer a seamless and user-friendly experience, enhancing user satisfaction and productivity. A well-designed input interface streamlines data entry processes, reduces user errors, and improves overall usability, contributing to the success and effectiveness of the application.

## *B. DATABASE DESIGN*

The database design is crucial for efficiently storing and managing data related to users, drivers, hiring requests, and other system entities. Here's a detailed outline for the database design:

1. User Collection:

Attributes:

- userId (Primary Key): Unique identifier for the user.
- username: User's username or email address.
- password: Encrypted password for user authentication.
- role: Role of the user (e.g., customer, admin).
- Other user-related attributes such as name, email, phone number, etc.

2. Driver Collection:

Attributes:

- driverId (Primary Key): Unique identifier for the driver.
- userId (Foreign Key): Reference to the corresponding user in the User collection.
- availability: Availability status of the driver (e.g., available, busy).

3. Hiring Request Collection:

Attributes:

- requestId (Primary Key): Unique identifier for the hiring request.
- userId (Foreign Key): Reference to the user making the request.
- pickupLocation: Location from which the ride will be requested.
- destination: Destination location of the ride.
- status: Status of the hiring request (e.g., pending, accepted, completed).
- driverId (Foreign Key): Reference to the driver accepting the request (if applicable).

4. Feedback Collection:

Attributes:

- feedbackId (Primary Key): Unique identifier for the feedback.
- userId (Foreign Key): Reference to the user providing the feedback.
- driverId (Foreign Key): Reference to the driver receiving the feedback.
- rating: Rating provided by the user for the driver's service (e.g., 1-5 stars).
- comment: Optional comment or review provided by the user.

This database design utilizes a relational approach with MongoDB, utilizing references (foreign keys) to establish relationships between collections.[8] It allows for efficient querying and retrieval of data while maintaining data integrity. Additionally, consider indexing fields used frequently in queries for improved performance and implementing data

validation and security measures to prevent data inconsistencies and unauthorized access.[9]

### C. *TOKEN BASED AUTHORIZATION*

JSON Web Token, or JWT, is a compact, self-contained data structure used for securely transmitting information between parties in a web application. JWTs are employed to authenticate and authorize users in web applications, allowing them to access certain resources or perform specific actions based on their identity. JWTs consist of three sections: a header, a payload, and a signature. The header defines the algorithm and token type, the

payload contains user-related information, and the signature ensures the token's integrity. the client stores the JWT and includes it in subsequent requests to protected resources. The server, upon receiving the token, verifies its signature and checks if the user has the necessary permissions to access the resource.

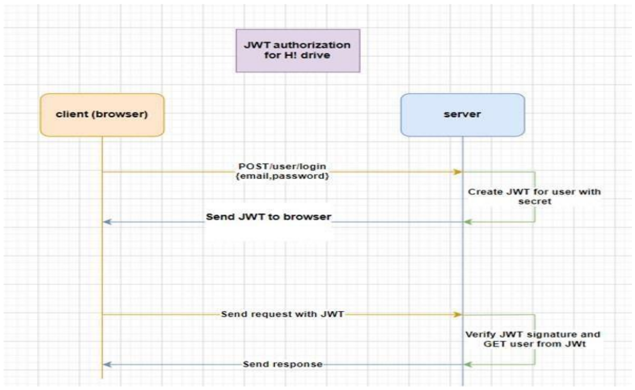
JWT authorization for H! drive
client (browser)
server
POST/user/login (email,password)
Create JWT for user with secret
Send JWT to browser
Send request with JWT
Verify JWT signature and GET user from JWt
Send response

Figure: 6.1 (JWT authentication)

## VII. DIAGRAMS

### A. *USE CASE*

A use case is like a specific task or action performed within an application, and it can involve one or more individuals or processes. This task can be as simple as a single action or as complex as a multi-step process. In application development, a use case could be represented by a complete workflow, a single action in that workflow, a series of screens and interactions, an activity that can be initiated under certain conditions, or even a new interface for a specific part of the application.

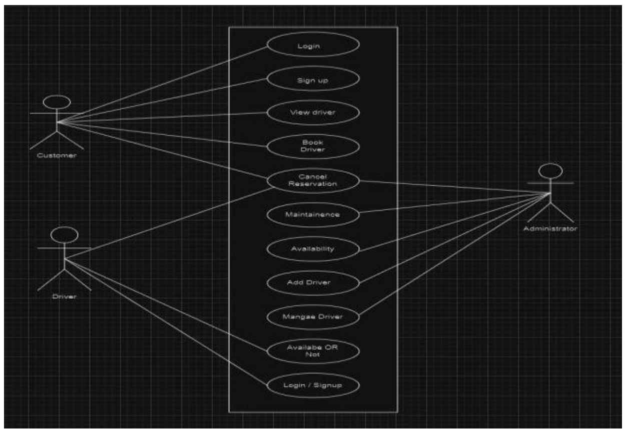

Figure: 7.1 (use-case diagram)

### B. *WORK FLOW*

A workflow diagram is a graphical illustration of a business process, typically created using a flowchart. It employs commonly recognized symbols to outline the specific tasks required to execute a process, while also identifying the individuals accountable for each task.

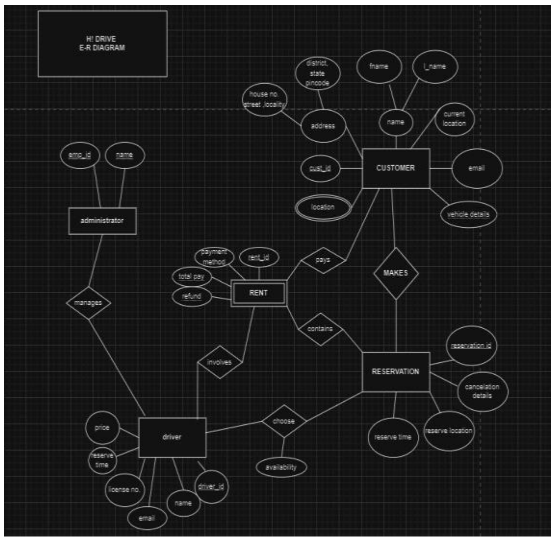

Figure: 7.2 (E-R diagram)

## VIII. FURURE SCOPE

Our proposed system has several key features: Location and Speed Tracking: We can monitor the car's location and how fast it's [2] moving.

- Driver Database: We keep a record of all the drivers and collect feedback from customers about them. Special Offers: We provide tailored offers to both drivers and customers based on specific conditions. Focus on Safety: Our primary goal is to make booking a driver easy and ensure our customers' safety.
- Technology: This system utilises Google Maps, sensors, and web services to achieve our goals. Sensors help us evaluate a driver's behaviour.
- Security: The system is highly secure to protect data on our servers.
- Driver Management: This system offers tools to manage driver information and monitor their activities.
- Statistics: Admins can access driver statistics to analyse their driving patterns.

- Powerful Search: Our system uses a sophisticated algorithm (nearest neighbour) to efficiently find the right driver.

In summary, our system is designed to be secure, user friendly, and focused on ensuring safety while making it easy for customers to book a driver.

## IX. CONCLUSION

The driver hiring web application offers several clear benefits to both customer and drivers. Employers can streamline their hiring process, reduce administrative overhead, and access a larger pool of qualified drivers. Drivers, on the other hand, benefit from increased job opportunities, improved working conditions, and better pay rates. The platform's ability to match drivers with compatible employers enhances job satisfaction and overall efficiency in the industry. And, it is essential to acknowledge potential challenges in the development and adoption of such a system. Ensuring the security and privacy of user data, addressing potential biases in driver selection, and adapting to changing regulations and technological advancements are among the key challenges.

## X. ACKNOWLEDGEMENT

We would like to extend our sincere appreciation to all the individuals who have supported us, both directly and indirectly, throughout the course of this thesis. We want to begin by extending our sincere thanks to Professor Amit Kumar Sanger sir, who has been my supervisor. Their guidance and mentorship have been invaluable, and we consider them the finest advisor we could have had. Prof. Amit's advice, encouragement, and constructive critiques have not only been instrumental in generating innovative ideas but have also served as a constant source of inspiration throughout this dissertation. Their unwavering confidence in our abilities has been a driving force behind our success. Working with Prof. Amit sir over the past year has been a privilege, and we are immensely grateful for their support not only during the project but also in various aspects of our academic journey. Their teachings and insights have not only enhanced our thesis work but have also contributed to our growth as a responsible and skilled professional. We are profoundly thankful for their mentorship, which has played a pivotal role in the successful completion of this work.

## REFERENCES

[1] DriveMyCar Android Application - IRE Journals https://www.irejournals.com/formatedpaper/1702686.pdf

[2] Online Driver Booking Service - IJRES https://www.ijres.org/papers/Volume-9/Issue-7/Series-13/F09072427.pdf

[3] https://book.olacabs.com/rental-cabs

[4] www.driveu.in/Driver/Pune

[5] www.089drivers.com/

[6] React. (2021). React Documentation. Retrieved from https://reactjs.org/docs/getting-started.html

[7] MongoDB University. (2021). MongoDB University. Retrieved from https://university.mongodb.com/

[8] Dey, A., Natale, M. D., Iovanna, P., & Vergari, A. (2020). A comprehensive study of MongoDB. Journal of Systems and Software, 168, 110683.

[9] Suleiman, H., Qabajeh, L., Kandeel, W., & Jararweh, Y. (2019). MongoDB Atlas: A Cloud-based Database Service. In 2019 5th International Conference on Computing Sciences (ICCS) (pp. 79-84). IEEE.

[10] Node.js Foundation. (2021). Node.js Documentation. Retrieved from https://nodejs.org/en/docs/