**Jordan Basketball Management System**

Shadip Kumar Joshi

B.Sc. (Hons.) Computing, Softwarica College of IT and E-commerce, Coventry University

ST5007CEM Web Development

Hari Saran Shrestha

August 10, 2023

**Acknowledgement**

I would like to express my heartfelt gratitude to all those who contributed to the successful completion of this website named Jordan Basketball Management System. Special thanks go to my module leader Hari Saran Shrestha for his valuable guidance, encouragement, and support throughout this project.

Additionally, I extend my appreciation to my peers and seniors for their constructive feedback and insightful discussions. Lastly, I am indebted to the developers and individuals who generously shared their knowledge and resources, enabling me to enhance my understanding and deliver this work.

This report would not have been possible without the collective efforts and cooperation of all involved. Thank you for being an essential part of this endeavour.

**Table of Contents**

[Jordan Basketball Management System 5](#_Toc140791007)

[Introduction 5](#_Toc140791008)

[Keywords 7](#_Toc140791009)

[Aim 8](#_Toc140791010)

[Objectives 8](#_Toc140791011)

[Problem Statement 10](#_Toc140791012)

[Problem 10](#_Toc140791013)

[Solution 12](#_Toc140791014)

[Features and Functionalities 14](#_Toc140791015)

[Functional Requirements 15](#_Toc140791016)

[Non-functional Requirements 17](#_Toc140791017)

[Scope 19](#_Toc140791018)

[Development Methodology 20](#_Toc140791019)

[Methodology 20](#_Toc140791020)

[Tools and Technologies 22](#_Toc140791021)

[Conceptual Diagram 23](#_Toc140791022)

[System Architecture 24](#_Toc140791023)

[Three Tier Architecture 24](#_Toc140791024)

[MVP Pattern 26](#_Toc140791025)

[Project Plan 28](#_Toc140791026)

[Milestones 28](#_Toc140791027)

[Figure 16 28](#_Toc140791028)

[Gantt Chart 29](#_Toc140791029)

[Figure 17 29](#_Toc140791030)

[Prototypes 30](#_Toc140791031)

[High Fidelity 30](#_Toc140791032)

[Figure 28 30](#_Toc140791033)

[Figure 29 31](#_Toc140791034)

[Figure 30 31](#_Toc140791035)

[Figure 31 32](#_Toc140791036)

[Figure 32 32](#_Toc140791037)

[Developed System 33](#_Toc140791038)

[Menus 33](#_Toc140791039)

[Figure 34 33](#_Toc140791040)

[Testing 34](#_Toc140791041)

[Figure 59 34](#_Toc140791042)

[Conclusion 40](#_Toc140791043)

[Reference 41](#_Toc140791044)

[Appendix 43](#_Toc140791045)

**Table of Figures**

[Figure 1 6](#_Toc140790988)

[Figure 2 7](#_Toc140790989)

[Figure 3 11](#_Toc140790990)

[Figure 4 13](#_Toc140790991)

[Figure 5 16](#_Toc140790992)

[Figure 6 19](#_Toc140790993)

[Figure 7 20](#_Toc140790994)

[Figure 12 23](#_Toc140790995)

[Figure 13 25](#_Toc140790996)

[Figure 14 27](#_Toc140790997)

[Figure 16 28](#_Toc140790998)

[Figure 17 29](#_Toc140790999)

[Figure 28 30](#_Toc140791000)

[Figure 29 31](#_Toc140791001)

[Figure 30 31](#_Toc140791002)

[Figure 31 32](#_Toc140791003)

[Figure 32 32](#_Toc140791004)

[Figure 34 33](#_Toc140791005)

[Figure 59 34](#_Toc140791006)

# 

# Jordan Basketball Management System

# Introduction

In the basketball world, Michael Jordan's name resonates strongly as an iconic representation of excellence and unparalleled talent. The Jordan Basketball Management System (JBMS) takes its name as a tribute to his impact and enduring legacy.

At its core, the JBMS serves as a comprehensive tool in the form of website that empowers players and coaches alike, enhancing their engagement with the sport and optimizing their basketball journey. This user-friendly digital application seamlessly facilitates a wide array of services viz searching basketball courts, reserving free courts, management of playgrounds, communication, etc at the convenience of one's fingertips.

Figure 1 shows the logo of this project along with an interface.

##### Figure 1

### Logo of the application with UI.



# Keywords

##### Figure 2

Keywords for Jordan Basketball Management System



# Aim

Jordan Basketball Management System aims to provide an online platform for easy booking of basketball courts for matches and efficient management for owners.

# Objectives

The objectives of “JBMS” are to:

* streamline court-booking process for players and enthusiasts.
* provide a user-friendly platform to check availability of courts in real-time and secure bookings effortlessly.
* deliver a seamless and immersive user-experience to cater to the player’s, coach’s, and court owner’s needs.
* simplify court management tasks like bookings, tournaments scheduling, etc for effective utilization of court facilities.
* provide valuable insights into court usage patterns, player preferences, and review feedback.
* fosters a vibrant basketball community to encourage social interaction among players, coaches, and enthusiasts, and facilitating team formation and friendly matches.
* establish a smooth communication channel between players and court owners to facilitate easy contact for inquiries, feedback, and addressing any concerns promptly.
* provides a review and rating system enabling users to share their experiences and feedback.
* optimize court usage, minimize downtime and maximize revenue for court owners through efficient scheduling and management.
* support and promote local basketball tournaments encouraging community engagement and providing a platform for players to showcase their talent.
* improve its features based on user feedback and technological advancements ensuring it remains at the forefront of basketball management systems.

# Problem Statement

## Problem

Players face various hurdles and problems to book basketball courts that meet their needs and requirements of infrastructures and time. Many courts still use traditional slow and cumbersome court booking procedures through pen and paper booking that leads to frustration among players and court owners.It lacks the ability to provide real-time updates on court availability, leading to miscommunications and missed opportunities for players. There is no provision of any data analytics, leaving players, coaches, and court owners without valuable insights to improve decision-making and court utilization and it even is prone to errors, document loss. Lack of a structured review and feedback system in this method, makes it difficult for players to share experiences and for potential users to make informed decisions. Due to lack of adequate security measures and data privacy policies concerns about the confidentiality of personal information may potentially arise. Additionally, it may lead to potential booking conflicts and inefficient coordination for its absence of effective communication channels between players and court owners. Figure 3 shows some problems in for basketball management system at present.

##### Figure 3

Problem statement in basketball management system

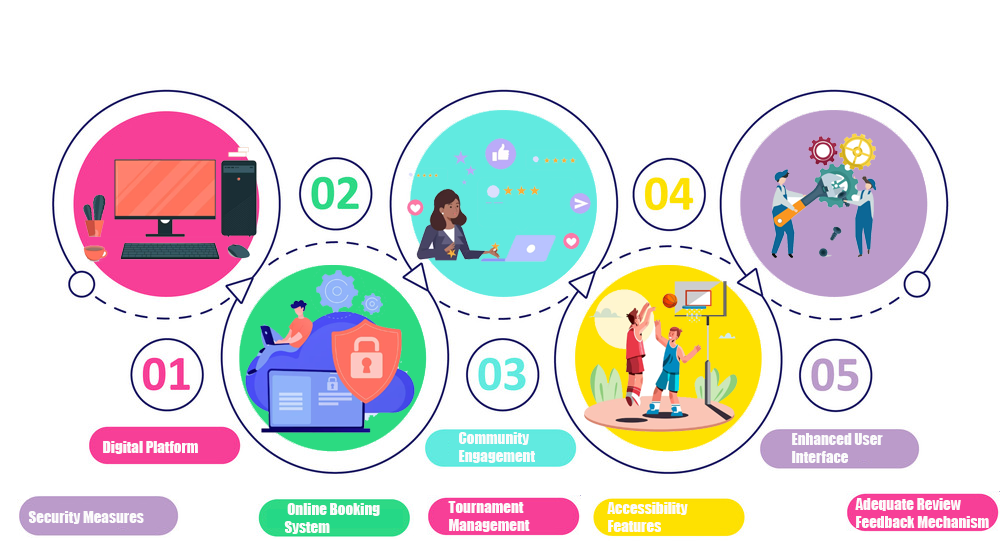


## Solution

To address the problems suffered by players in court booking, this JBMS website provides a user-friendly and intuitive online booking platform that allows players to check real-time court availability, make reservations, and manage bookings seamlessly. Implementing a digital system eliminates the slow and cumbersome nature of pen and paper booking. It also incorporates data analytics into its systems to gather insights on court utilization, player preferences, and peak booking times to optimize court allocation and improve the overall user experience. To eliminate miscommunications and enhances coordination it implements a centralized communication system within digital platforms to enable seamless and direct communication between players and court owners, thereby fostering social engagement. Additionally, it has a structured review and feedback system to encourage players to share their experiences and rate court facilities. This information helps potential users make informed decisions. Finally, it incorporates a robust security protocols and data privacy measures to safeguard users' personal information and ensure confidentiality. Figure 4 depicts some solutions for basketball management system.

##### Figure 4

Solutions to problems in basket management system.



# Features and Functionalities

The features of this applications are:

* **User Registration**: New users can register easily by providing essential details, such as name, email, and password.
* **Player Profile Creation**: Players can create personalized profiles, showcasing their achievements, skills, preferred positions, and basketball history.
* **Secure User Login**: The app employs robust security measures to ensure secure user login, protecting user data and preventing unauthorized access.
* **Password Reset**: Users can initiate a password reset process in case they forget their password, receiving a password reset link via email for account recovery.
* **Court Owner Profile Creation**: Court owners can create detailed profiles for their venues, including court images, facilities, rates, and available time slots.
* **Edit Profile:** Users can easily edit and update their profiles, enabling them to keep their information current and relevant.
* **Account Verification:** To ensure the authenticity of user accounts, the app may implement a verification process, such as email verification or mobile number verification.
* **User Dashboard**: Each user is provided with a personalized dashboard that displays relevant information, such as upcoming bookings, training schedules, and recent activity.
* **Booking History**: Users can access their booking history, which includes past and upcoming court reservations, allowing them to track their basketball activities.
* **Multi-Platform Accessibility**: JBMS is accessible across multiple platforms, including web browsers, smartphones, and tablets, ensuring users can use the app from any device with internet connectivity.

## Functional Requirements

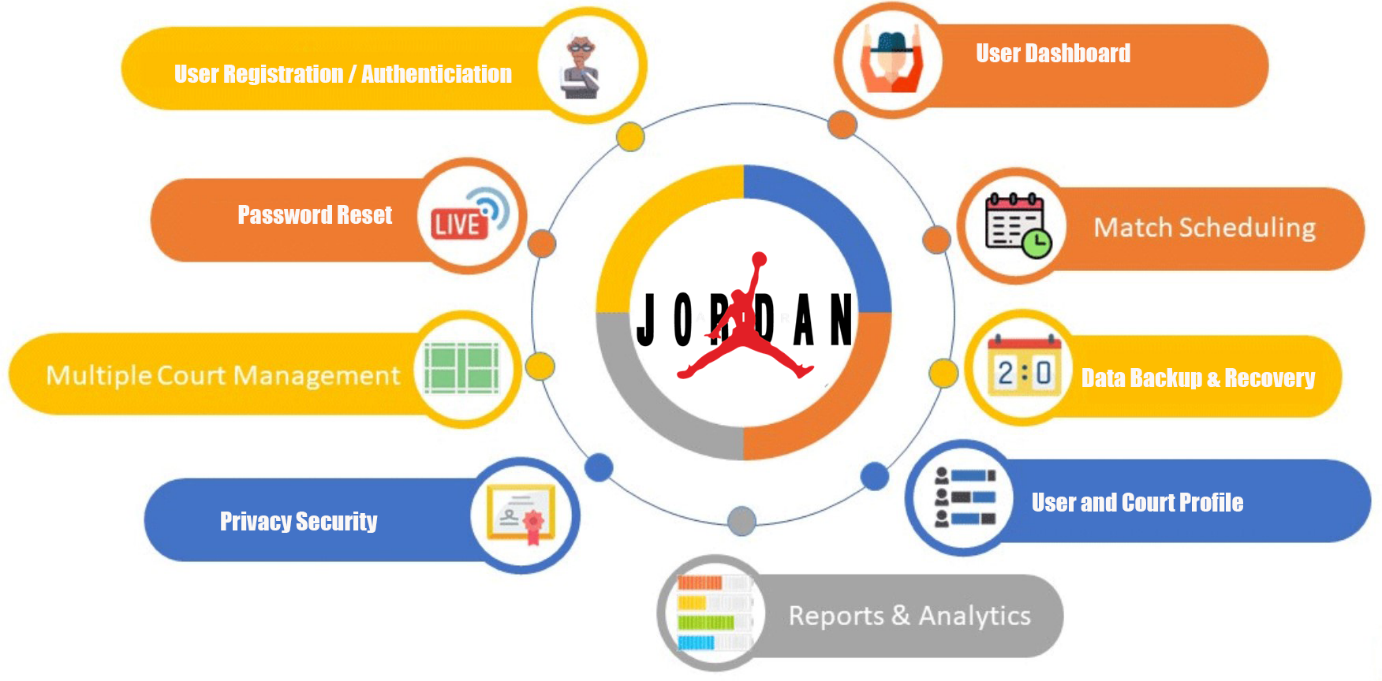
Functional requirements, detail the actions the software needs to perform and describe its capabilities in a comprehensive manner, are as follows:

* **User Registration**: The system should allow new users to register by providing their name, email, and password.
* **Authentication:** User registration should be verified through email or mobile number verification to ensure authenticity.
* **Player Profile Management**: Users should be able to create and edit personalized player profiles.
* **Court Owner Profile Management:** Court owners should have the ability to create and modify detailed profiles for their venues. Court profiles should include court images, facilities, rates, and available time slots.
* **Password Reset**: Users should be able to initiate a password reset process if they forget their password through email.
* **Account Deletion:** Users should have the option to delete their account if they wish to discontinue using the JBMS.
* **Privacy Security**: The software should employ robust security measures to protect user data, including encryption of sensitive information and secure transmission of data.

Figure 5 depicts some functional requirements of a proper basketball management system.

##### Figure 5

Functional Requirements of Basketball Management System.



## Non-functional Requirements

Non-functional requirements, which define how the system should perform, are as follows:

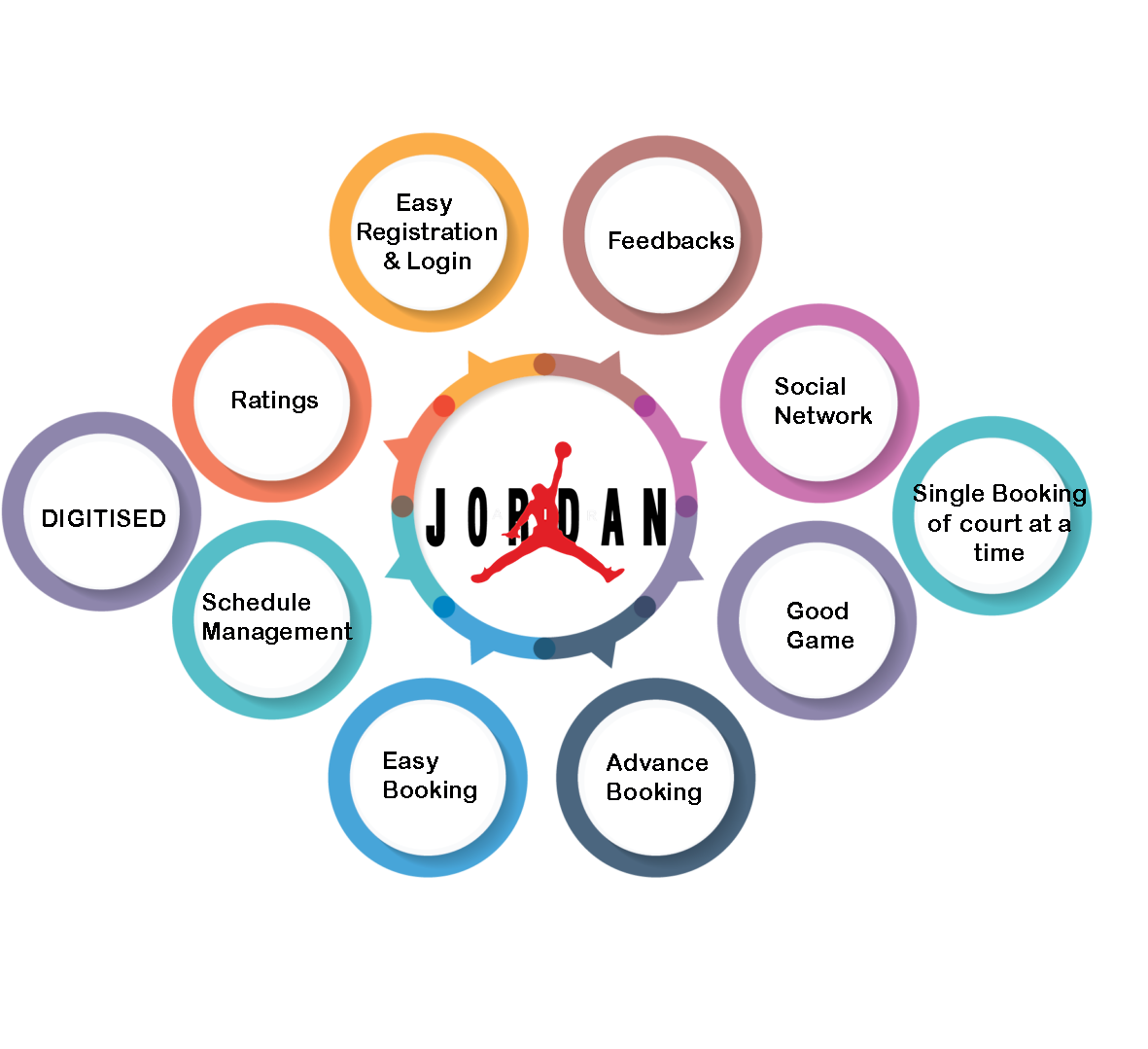
* **Usability:** JBMS should have an intuitive and user-friendly graphical user interface (GUI), ensuring that users can easily navigate and interact with the application.
* **Security**: The system should implement robust security measures to safeguard user data and ensure secure login and logout mechanisms. Users should only have access to their own credentials and data.
* **Performance**: Transitions between different interfaces and actions triggered by buttons should be smooth, with response times not exceeding 10 seconds.
* **Data Changeability**: The system should efficiently handle modifications to the database, ensuring that updates and changes are processed swiftly without any data inconsistencies.
* **Capacity**: JBMS should be capable of handling concurrent usage by a minimum of 200 users without significant degradation in performance, ensuring smooth operations during peak usage times.
* **Scalability**: The application should be designed with scalability in mind, allowing for future upgrades and expansions to accommodate growing user demands and additional features.
* **Portability**: JBMS should be compatible with various operating systems and mobile devices, enabling users to access and utilize the application seamlessly on their preferred platforms.
* **Compatibility**: The system's requirements should not be overly demanding, allowing it to run smoothly on different hardware configurations without compromising performance.
* **Connectivity**: The software should maintain a reliable connection to the internet, ensuring continuous accessibility and functionality even in low-bandwidth environments.

# Scope

The scope of this program is defined by its aim, objectives, problem statements, features, functional requirements, and non-functional requirements outlined earlier, as depicted in Figure 6.

##### Figure 6

Scope of Jordan Basketball Management System



# Development Methodology

Development methodology pertains to the process known as the Software Development Life Cycle (SDLC), which outlines the sequential stages involved in creating software from its inception to completion.

## Methodology

The Modern Waterfall Model, also referred to as the Iterative Model, is employed for the development of this software. It is preferred due to its systematic and easily understandable phases, making it suitable for individuals new to the IT field. As novice web developers in the realm of Software Engineering, this method is well-suited for us. Figure 7 illustrates all the stages in this model.

##### Figure 7

Stages of waterfall method [(Singhal, 2021).](#water)

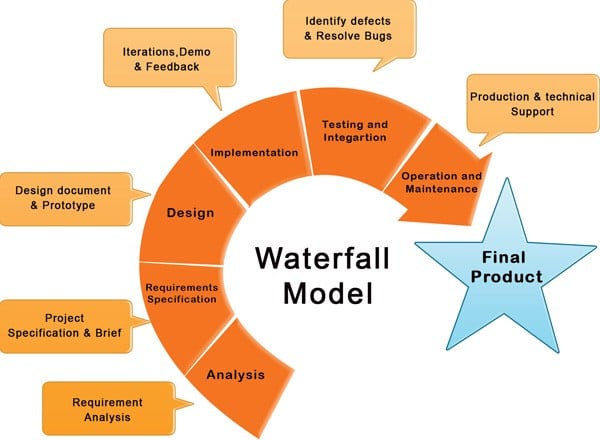
Diagram

Description automatically generated

The developing began by conducting a 'Feasibility study' to assess the practicality of the software through discussions and research. Following that, the 'Requirement Analysis and Specification' phase involved gathering and analyzing requirements, leading to the creation of a Software Requirement Specification (SRS) document. This document detailed functional and non-functional requirements, objectives, features, problem statements, scope, and other software specifics. Next, a structured 'Design' was iteratively finalized using an object-oriented approach, considering various objects in both the problem domain and solution domain, along with their relationships. The design was then implemented into source code using the Java programming language during the 'Development' phase, resulting in a functional software that underwent unit testing and incremental integration in the subsequent 'Integration and System-testing' phase. With the completion of testing, the software was ready for release to users and was regularly maintained during the 'Deployment and Maintenance' stage [*(Nitika, 2018).*](#niti)Throughout the process, the various phases were revisited as needed to accommodate new user requirements and feedback that arose during development [*(Singhal, 2021).*](#water)Figure 8 provides an overview of all the phases of the waterfall SDLC, along with their respective characteristics.

##### Figure 8

*Phases and their characteristics of Waterfall Model* [*(Nitika, 2018).*](#niti)

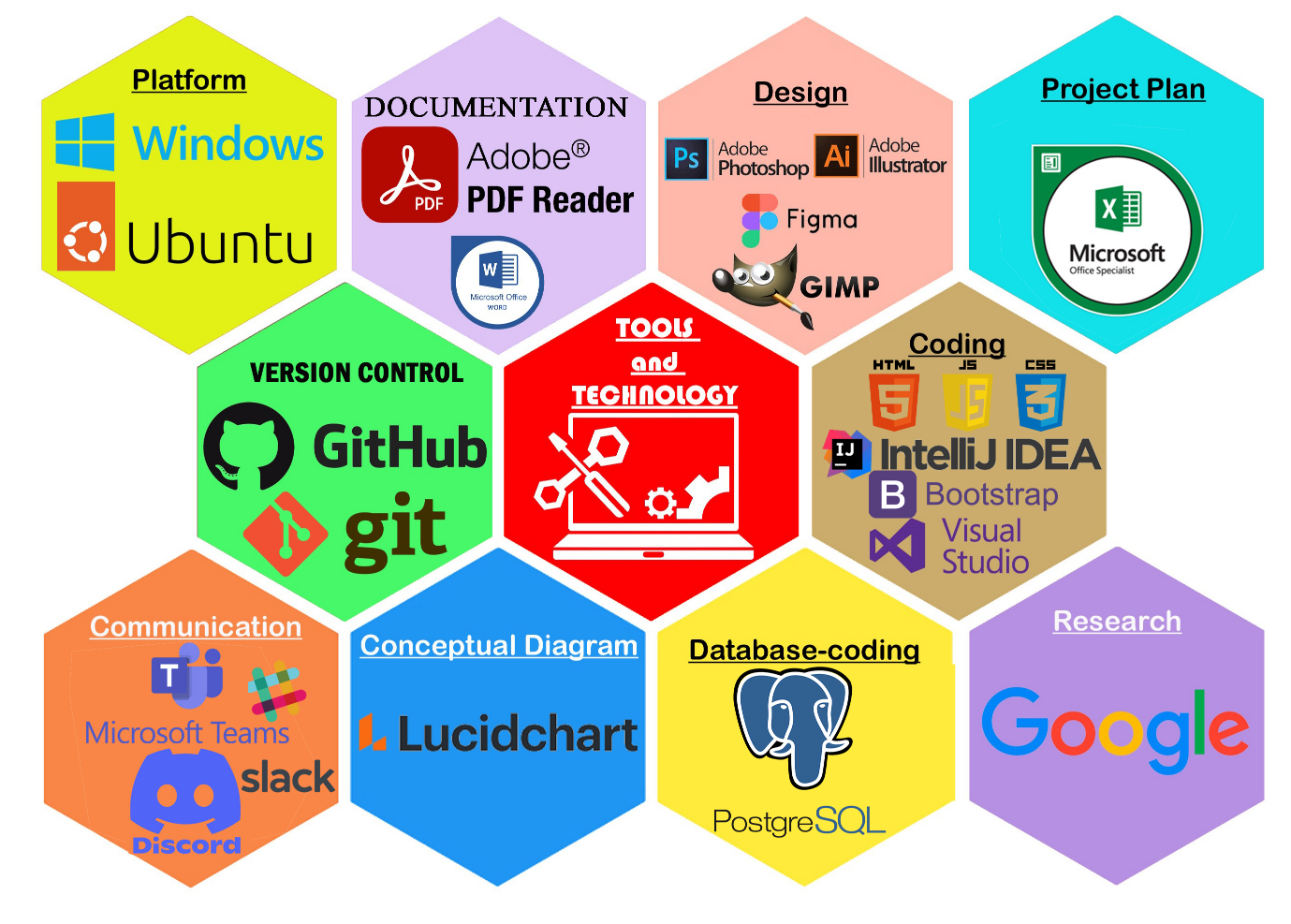


## Tools and Technologies

Various tools and technologies were used in the entire SDLC as shown in figure 9.

##### Figure 9

Tools and Technologies used in JBMS

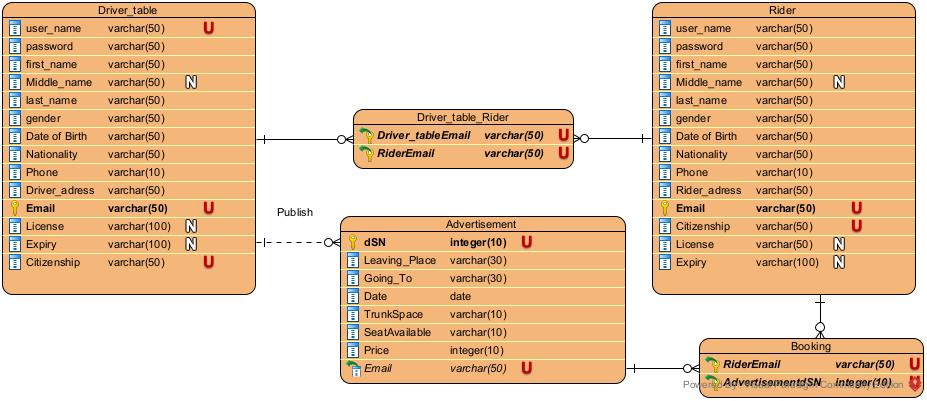
****

## Conceptual Diagram

Entity-Relationship Diagram explains the relationship between entities in a system diagrammatically. Figure 10 demonstrates ER diagram between entities in this project.

##### Figure 10

ERD of Unistay



# System Architecture

The structure and behavior of a good software are defined by its appropriate system architecture.

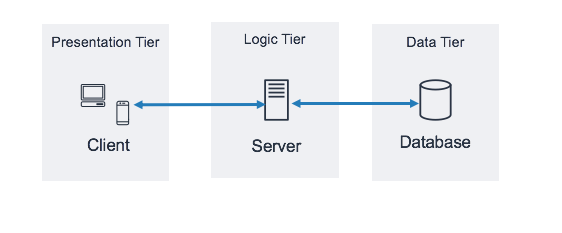
## Three Tier Architecture

The three-tier architecture, illustrated in Figure 11, is employed to organize an application into three separate computing tiers: the presentation tier, the application tier, and the data tier. This approach allows each tier to function independently on different infrastructures, providing benefits such as isolated development, updates, and scalability without affecting the other tiers [*(IBM, n.d.).*](#n)

The presentation tier, also known as the front-end, serves as the user interface and communication layer of the software. It facilitates user interactions, displaying information and collecting input. For this project, the presentation tier is developed using HTML, CSS, JavaScript and Bootstrap. The application tier, often referred to as the logic tier or middle tier, is the core of the application. It processes data received from the presentation tier and performs various operations such as data processing, addition, deletion, or modification. In this project, the application tier is developed using Java. The data tier, also known as the database tier or back-end, is responsible for storing and managing the processed information. PostgreSQL is utilized as the data tier for this software. One notable advantage of the three-tier architecture is improved security, as the presentation tier and data tier cannot communicate directly; all communication between these tiers must pass through the application tier [*(IBM, n.d.).*](#n)

##### Figure 11

3-Tier Architecture [(AWS Whitepaper, n.d.).](#j)



## MVC Pattern

The MVC (Model-view-controller) pattern simplifies complex application development by dividing it into manageable parts, separating the frontend and backend codes. This enables multiple developers to work on the software simultaneously without causing conflicts. The pattern consists of three components: the model, view, and controller, as depicted in figure 12. The model represents the backend, containing data and related logic. The view serves as the frontend or graphical user interface (GUI), displaying the data from the model to the user. The controller acts as the application's central control unit, handling user input and converting it into commands for the application. [*(Hernandez, 2021)*](#o).

##### Figure 8

MVC Pattern [mvc](#mvc)(Singh Bhui, 2022).

A diagram of a computer

Description automatically generated

## Software Quality

In my individual project, I emphasized software quality by incorporating both SOLID principles and Lean methodology.

### SOLID principles

By following SOLID principles, I ensured that my codebase was robust, maintainable, and easy to understand. The Single Responsibility principle allowed me to divide responsibilities among different modules, enhancing code readability and reducing the likelihood of errors. Implementing the Open/Closed principle allowed me to extend the software's functionality seamlessly without modifying existing code, reducing the risk of introducing bugs. I also adhered to the Liskov Substitution principle, ensuring that different components could be interchanged without compromising the overall system's integrity. Emphasizing Interface Segregation led me to create precise interfaces, promoting efficient communication between different parts of the application and minimizing the impact of changes. Lastly, applying the Dependency Inversion principle resulted in a loosely coupled architecture, making testing easier and enhancing the software's stability [*(Oloruntoba, 2021).*](#solid)

### Lean Methodology

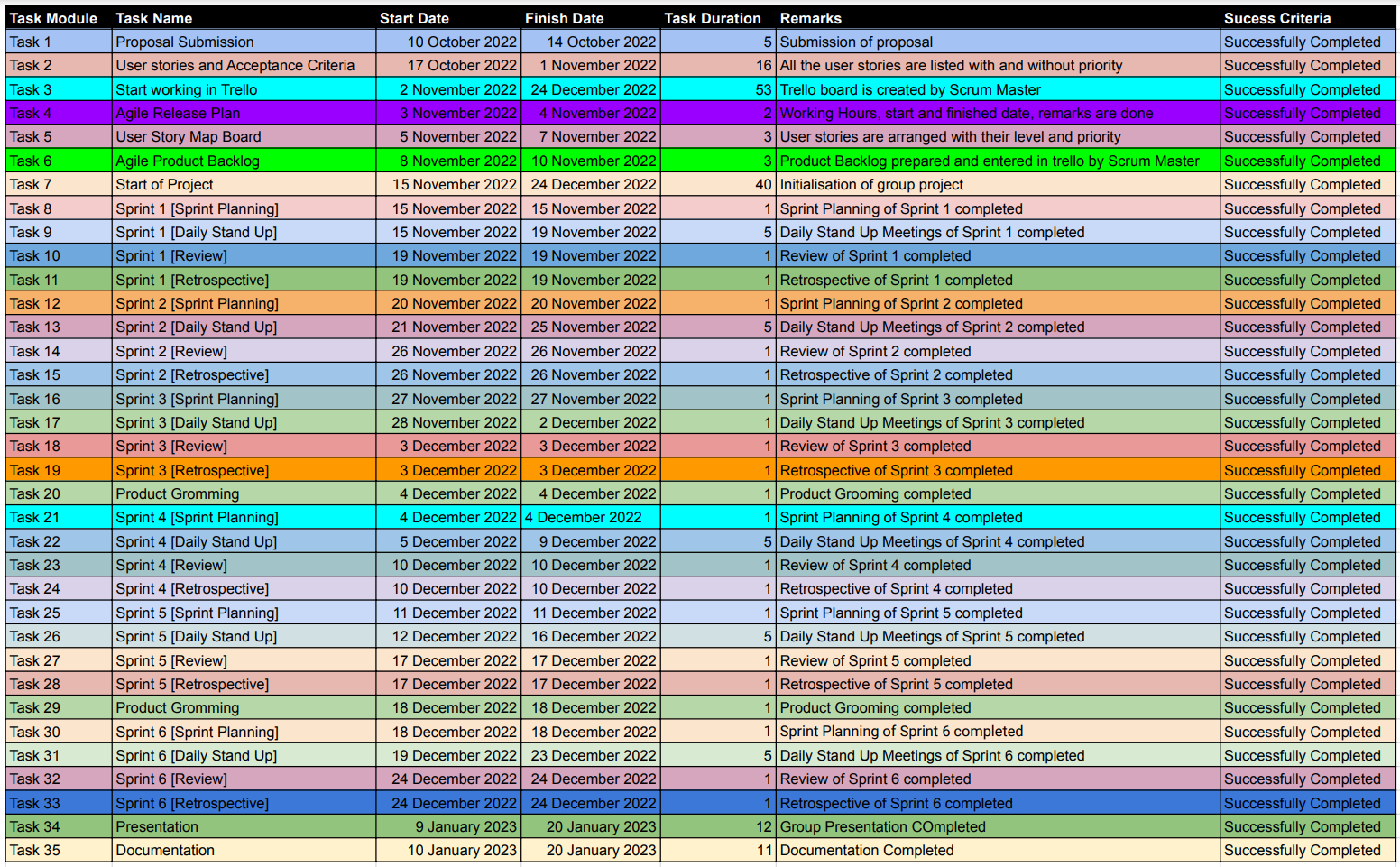
To further elevate software quality, I incorporated Lean methodology into my development process. Frequent iterations and a continuous improvement mindset allowed me to quickly identify and address any issues or changes in requirements. Regular feedback loops enabled timely adjustments, leading to a more refined and user-friendly end product. By prioritizing customer value, I focused on implementing the most critical features and eliminating unnecessary complexities, ensuring that the software aligned perfectly with the needs and preferences of end-users. Lean's emphasis on resource optimization and waste reduction streamlined my development efforts, maximizing productivity and minimizing delays [(Simplilearn, 2023).](#lean)

# Project Plan

In software engineering, a progress chart is essential for effectively monitoring the time, effort, and advancement of software development. Gannt Chart wasutilized to document the progress plan for all phases of the modern waterfall SDLC, as depicted in figure 13. This chart allows for a visual representation of the project's timeline and tasks, helping the team stay organized and track their development milestones.

##### Figure 13

Milestone and Gannt Chart of Sahayatri carpool system



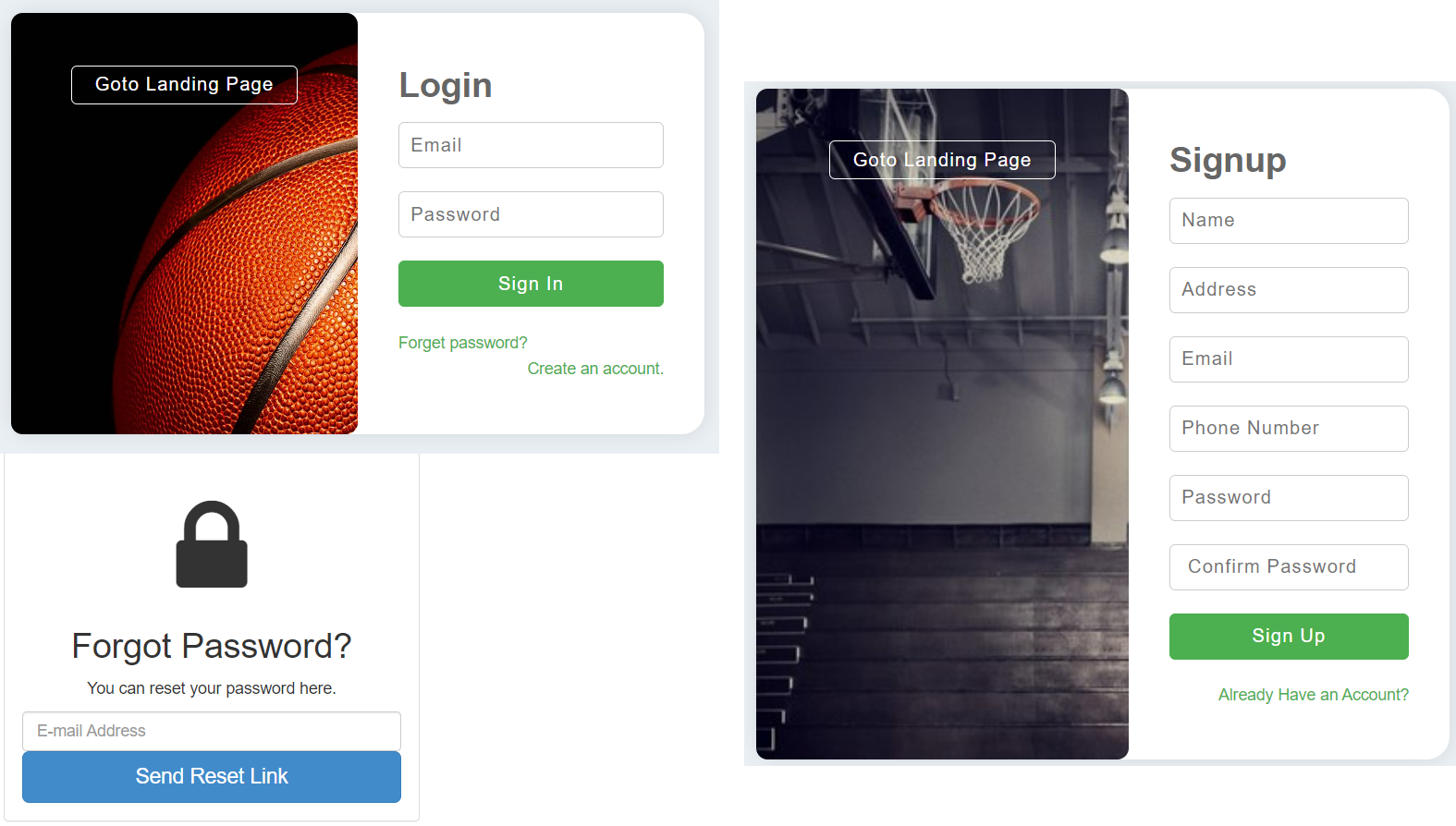
# Prototypes

Figures 14-16 illustrates several types of prototypes, which are early models, or releases of software built for testing a concept or process, in this project [*(Simplilearn, 2022).*](#r)

## High Fidelity

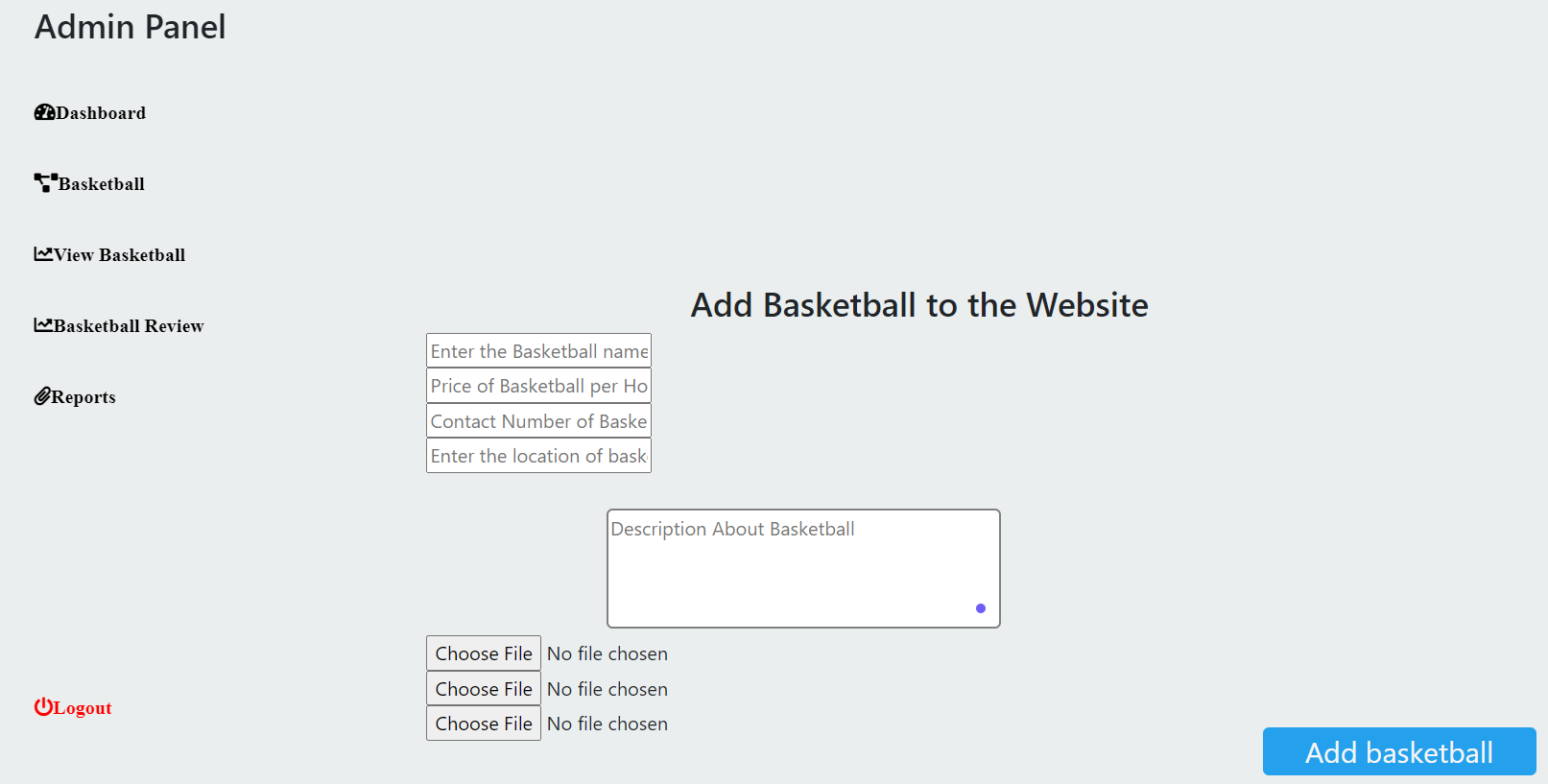
##### Figure 14

High fidelity prototype of Login, Signup and Forgot Password



##### Figure 15

Prototype of Admin Dashboard



##### Figure 16

High fidelity prototype of Landing Page

A basketball hoop and a basketball ball

Description automatically generated

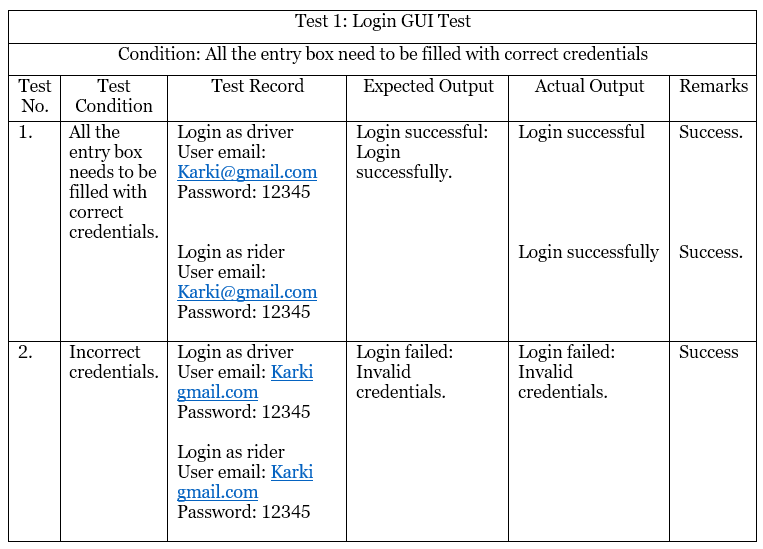
# Developed System

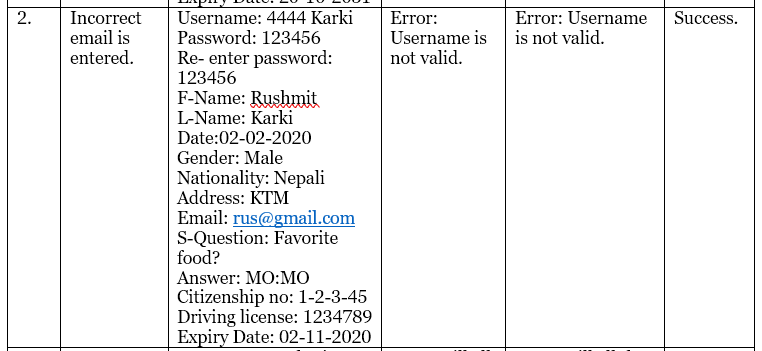
# Testing

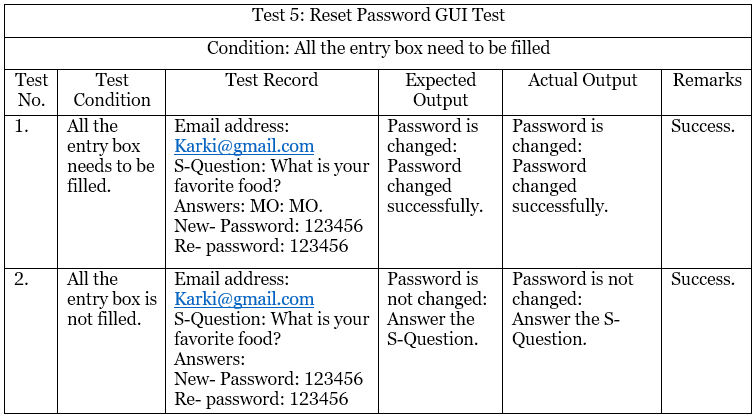
Black box testing was performed for system testing after the completion of coding to check the functionality of the project as shown in figure 57. Outputs were recorded to the test cases. Errors were corrected until all test cases were satisfied.

## Figure 15

Code Testing

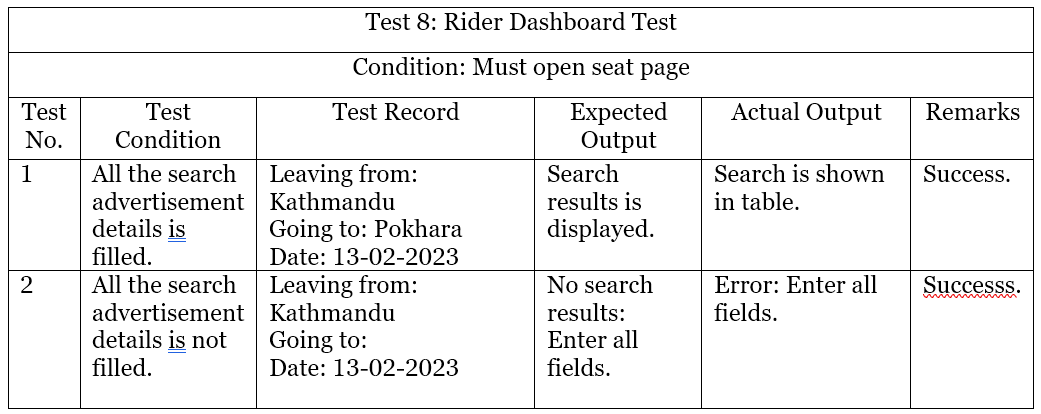






A white box with black text

Description automatically generated



# Conclusion

The final working product was engineered by the team as per the requirement stated by the user by integrating all the concepts learned in classroom from modules like Mathematics, Programming and Algorithm, Database, Architecture, Object-oriented Programming and Software Design and using all the available technologies to their fullest.

This project simplifies carpooling interaction between the driver and rider. Both the functional and non-functional requirements as stated for the software are fully satisfied. System testing has shown positive results. All the bugs noticed in the implementation phase were solved by the team with great teamwork. The GUIs can be systematically accessed through correct credentials and user can be worry-free about their privacy. Through good teamwork the GUI is built to be attractive, presentable, simple, user-friendly, and self-describable.

The team has gained precious experience of developing a software from scratch, increasing our experience in the IT field. The use of agile and scrum taught the team that many problems can be solved by putting more heads in use and simplicity in programming is the best. It has taught us to learn and grow through new challenges. Friendly interaction between user, teammates and instructor, reminds us of the importance of friendly social networks. The team pledges to be better in IT field through open-minded learning, taking new challenges to create software as per need incorporating logics and following the IT ethics.

# Reference

Abeythilake, U. (2022, June 12). *Agile Methodology*. Medium. Retrieved December 11, 2022, from <https://medium.com/@abeythilakeudara3/agile-methodology-106270809c99>

Agile Alliance. (2021, October 9). *Agile Manifesto for Software Development*. Agile Alliance |. Retrieved December 13, 2022, from <https://www.agilealliance.org/manifesto-download/>

AWS Whitepaper. (n.d.). *Three-tier architecture overview*. Amazon AWS. Retrieved December 11, 2022, from <https://docs.aws.amazon.com/whitepapers/latest/serverless-multi-tier-architectures-api-gateway-lambda/three-tier-architecture-overview.html>

Company, I.-. M. a. D., Trętowicz, M., Zalewski, A., & Turkowski, J. (2023, January 26). MVC MVP and MVVM architecture pattern – Introduction. *Mobile & Web App Development Company | USA, UK, Norway*. Retrieved July 10, 2023, from <https://itcraftapps.com/blog/mvc-mvp-and-mvvm-architecture-pattern-introduction/#:~:text=In%20MVP%2C%20the%20data%20from,which%20can%20support%20multiple%20Views>.

Enalean SAS. (n.d.). *Understanding Agile Scrum in 10 minutes*. Tuleap. Retrieved December 23, 2022, from <https://www.tuleap.org/agile/agile-scrum-in-10-minutes>

Global App Testing. (n.d.). *Best Practices for Agile Testing*. Global App Testing Retrieved December 13, 2022, from <https://www.globalapptesting.com/the-ultimate-guide-to-agile-testing>

Hoory, L., Bottorff, C., & Watts, R. (2022, March 27). *What Is A Burndown Chart? Everything You Need To Know*. Forbes Advisor. Retrieved December 13, 2022, from <https://www.forbes.com/advisor/business/what-is-a-burndown-chart/>

IBM. (n.d.). *What is three-tier architecture?* IBM. Retrieved December 24, 2022, from <https://www.ibm.com/topics/three-tier-architecture>

Lima, A. (2020). MVP vs MVVM: principais diferenças, vantagens e desvantagens. *Zup*. <https://www.zup.com.br/blog/mvp-vs-mvvm>

Nitika. (2018). *WHAT IS THE ROLE OF WATERFALL MODEL IN SDLC - Loginworks*. Loginworks. Retrieved 21 July 2023, from <https://www.loginworks.com/blogs/role-waterfall-model-sdlc/>.

Oloruntoba, S. (2021). SOLID: The First 5 Principles of Object Oriented Design. *DigitalOcean*. Retrieved July 20, 2023, from <https://www.digitalocean.com/community/conceptual-articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design>

Pm admin. (2021, June 13). *The Agile Journey: A Scrum overview*. PM Partners. Retrieved December 11, 2022, from <https://www.pm-partners.com.au/the-agile-journey-a-scrum-overview/>

Shoaib A. (n.d.). *You will get Hostel Management System (A complete Web Solution).* https://www.upwork.com. Retrieved July 9, 2023, from <https://www.upwork.com/services/product/development-it-hostel-management-system-a-complete-web-solution-1666480411133214720>

Simplilearn. (2022, November 9). *A Complete Guide to Prototyping In Design Thinking With Benefits & Examples*. Simplilearn.com. Retrieved November 7, 2022, from <https://www.simplilearn.com/prototyping-in-design-thinking-article>

Simplilearn. (2023, March 7). Fundamentals of lean methodology explained with examples. *Simplilearn.com*. Retrieved July 21, 2023, from <https://www.simplilearn.com/lean-methodology-article#:~:text=Lean%20is%20a%20continuous%20improvement,to%20customers%20with%20fewer%20resources>

Singh Bhui, J. (2022, October 6). *Understand MVC Architecture in 5 mins.* Crio. Retrieved July 21, 2023, from <https://www.crio.do/blog/understand-mvc-architecture/>

Singhal, P. (2021). *Software Engineering | Iterative Waterfall Model - GeeksforGeeks*. GeeksforGeeks. Retrieved 21 July 2023, from <https://www.geeksforgeeks.org/software-engineering-iterative-waterfall-model/>.

# Appendix

**Version Control**

**Github Link:** <https://github.com/220049/JORDAN-Basketball-Management-System>

**Youtube Video Link:**