FORMULA ONE
FAN HUB

Mini Project Report

Submitted by

NITHIN JOSE

Reg. No.: AJC22MCA-2069

In Partial fulfillment for the Award of the Degree of

MASTER OF COMPUTER APPLICATIONS
(MCA TWO YEAR)
[Accredited by NBA]

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

AMAL JYOTHI COLLEGE OF ENGINEERING

KANJIRAPPALLY

[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE, Accredited by NAAC. Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

2023-2024

DEPARTMENT OF COMPUTER APPLICATIONS

AMAL JYOTHI COLLEGE OF ENGINEERING

KANJIRAPPALLY

CERTIFICATE

This is to certify that the Project report, "**FORMULA ONE FAN HUB"** is the bona fide work of **NITHIN JOSE (Regno: AJC22MCA-2069)** in partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2023-24.

Ms. Gloriya Mathew
Internal Guide

Ms. Meera Rose Mathew
Coordinator

Rev. Fr. Dr. Rubin Thottupurathu Jose

Head of the Department

DECLARATION

I hereby declare that the project report **"FORMULA ONE FAN HUB"** is a bona fide work done at Amal Jyothi College of Engineering, towards the partial fulfilment of the requirements for the award of the Master of Computer Applications (MCA) from APJ Abdul Kalam Technological University, during the academic year 2023-2024.

Date: 01-12-23 **NITHIN JOSE**

KANJIRAPPALLY **Reg: AJC22MCA-2069**

ACKNOWLEDGEMENT

First and foremost, I thank God almighty for his eternal love and protection throughout the project. I take this opportunity to express my gratitude to all who helped me in completing this project. It has been said that gratitude is the memory of the heart. I wish to express my sincere gratitude to our Manager **Rev. Fr. Dr. Mathew Paikatt** and Principal **Dr. Lillykutty Jacob** for providing good faculty for guidance.

I owe a great depth of gratitude towards our Head of the Department **Rev.Fr.Dr. Rubin Thottupurathu Jose** for helping us. I extend my whole hearted thanks to the project coordinator **Ms. Meera Rose Mathew** for his valuable suggestions and for overwhelming concern and guidance from the beginning to the end of the project. I would also express sincere gratitude to my guide **Ms. Gloriya Mathew** for her inspiration and helping hand.

I thank our beloved teachers for their cooperation and suggestions that helped me throughout the project. I express my thanks to all my friends and classmates for their interest, dedication, and encouragement shown towards the project. I convey my hearty thanks to my family for the moral support, suggestions, and encouragement to make this venture a success.

NITHIN JOSE

ABSTRACT

.

The "Formula 1 Connection Hub" stands as a transformative venture set to redefine the Formula 1 fan landscape. This innovative platform goes beyond conventional fan engagement by seamlessly integrating ticket booking into each team's website, simplifying the process for enthusiasts. The project addresses an existing void in fan-team interaction, providing a space for fans to express their hopes and wishes directly to the teams. With a front-end driven by React and a robust ASP .NET Core back-end, the hub offers modules covering F1 and team history, driver and team member details, a dynamic gallery, and user functionalities such as registration, login, and an intuitive ticket booking system.

Comprising a rich array of features, the hub becomes a holistic fan experience. Users can explore the sport's extensive history, participate in fan polls for their favourite drivers, and actively contribute to open discussions through the platform's forum. The administrative side ensures smooth operation by allowing content management, including editing F1 and team history, managing gallery images, and overseeing user roles. Ticket management functionalities enable setting availability, viewing bookings, managing waiting lists, and handling cancellations, while the open forum management tools ensure a vibrant and respectful community. The "Formula 1 Connection Hub" is poised to elevate fan involvement, creating a dynamic space for exploration, engagement, and connection within the world of Formula 1.

CONTENT

List of Abbreviation

IDE	-	Integrated Development Environment
HTML	-	Hyper Text Markup Language.
CSS	-	Cascading Style Sheet
SQL	-	Structured Query Language
UML	-	Unified Modelling Language
RDBMS	-	Relational Database Management System
F1	-	Formula One

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

Formula One Fan Hub emerges as a cutting-edge online platform, uniting ardent Formula 1 enthusiasts, team aficionados, and motorsport experts within a dynamic virtual space. Comprising five key modules, the Admin module orchestrates the platform's seamless operation, while the Fan module empowers users to explore, engage, and elevate their Formula 1 experience. Registered Fans can delve into the sport's rich history, participate in lively discussions, and seamlessly book race tickets. Team Modules provide a dedicated space for team-centric content, including driver details, team history, and an immersive gallery experience.

This innovative hub aims to be the ultimate destination for Formula 1 enthusiasts, fostering a global community that seamlessly combines passion, knowledge, and interactive engagement. With features such as real-time fan polls, historical insights, and integrated ticket booking, Formula One Fan Hub aspires to redefine the fan experience in the high-speed world of Formula One.

1.2 PROJECT SPECIFICATION

Formula One Fan Hub is a visionary online platform designed to create a unified space for Formula 1 enthusiasts, team supporters, and motorsport experts. The project's objective is to establish an immersive digital ecosystem that serves as the central hub for Formula One-related activities, historical exploration, and interactive fan engagement.

1. Admin Module:

The centralized administrative module ensures smooth system management, overseeing user roles, content updates, and ticket management. It provides a comprehensive view of the platform's operation and ensures a seamless experience for all users.

2. Fans Module:

Fans can actively participate in the Formula 1 community, exploring historical insights, engaging in dynamic discussions, and effortlessly booking race tickets. The module aims to enhance the fan experience by providing a user-friendly interface for seamless interaction with the thrilling world of Formula 1.

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

The Formula One Fan Hub is an innovative online platform curated for Formula 1 enthusiasts, providing a comprehensive and engaging digital space. This dynamic hub, featuring modules such as Admin, Fan, and Team, aims to redefine the fan experience by seamlessly integrating historical exploration, interactive discussions, and ticket booking functionalities. With a user-friendly interface, advanced features, and a global vision, the Formula One Fan Hub aspires to become the go-to destination for Formula 1 enthusiasts, fostering a thriving community of passionate fans.

2.2 EXISTING SYSTEM

In the existing system of the Formula One project, enthusiasts and fans face challenges when trying to access relevant information, purchase tickets, or engage in discussions. Traditional methods often involve cumbersome processes, such as physically visiting race venues or relying on manual ticket booking systems. The search for historical information, team details, or race updates might require extensive time and effort. Additionally, participating in discussions or polls may be limited to local interactions. The Formula One project recognizes the need for a more streamlined and user-friendly system. As technology advances, the project aims to leverage online platforms, providing fans with easy access to comprehensive information, seamless ticket booking experiences, and vibrant online communities.

2.2.1 NATURAL SYSTEM STUDIED

The existing Formula 1 fan experience involves traditional methods like attending races in person or engaging with offline resources. These methods, while providing a certain level of engagement, lack the efficiency and accessibility required in today's fast-paced world. Offline systems often necessitate physical presence for ticket purchases, limiting fan interaction and making the process time-consuming. The Formula One Fan Hub addresses these drawbacks by seamlessly integrating ticket booking into team websites, offering a user-friendly interface, and bridging the gap between fans and teams.

2.2.2 DESIGNED SYSTEM STUDIED

The Formula One Fan Hub, designed as a comprehensive online ecosystem, transcends the limitations of the existing fan experience. The Admin module oversees the platform's operations, ensuring smooth user interactions, content management, and ticket logistics. The Fan module empowers users to explore the rich history of Formula 1, engage in vibrant discussions, and seamlessly book race tickets. Additionally, Team Modules provide dedicated spaces for each Formula 1 team, offering detailed insights into drivers, team history, and captivating galleries. The meticulously designed system not only caters to fan needs but also enhances the overall Formula 1 experience.

2.3 DRAWBACKS OF EXISTING SYSTEM

- Limited accessibility and convenience for fans
- Time-consuming processes for ticket purchases
- Insufficient online presence and engagement opportunities
- Lack of a centralized platform for fan-team interaction.

2.4 PROPOSED SYSTEM

The Formula One Fan Hub proposes a revolutionary shift by offering a centralized and user-friendly platform for Formula 1 enthusiasts. Fans can now explore, engage, and elevate their Formula 1 experience seamlessly through a dedicated online hub. The proposed system rectifies the drawbacks of the existing approach by providing a one-stop digital destination for historical insights, interactive discussions, and hassle-free ticket bookings. This system not only meets the demands of today's technologically advanced era but also envisions a global community of Formula 1 enthusiasts actively participating in the sport's narrative.

2.5 ADVANTAGES OF PROPOSED SYSTEM

- Enhanced accessibility and convenience for fans worldwide
- Streamlined and efficient ticket booking process
- Centralized platform for fan-team interaction and engagement
- Seamless integration of historical exploration and interactive discussions

CHAPTER 3
REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

A feasibility study is a comprehensive analysis of a proposed project that takes into account all critical aspects of the project to determine the likelihood of its success. It is a crucial step in the project development process as it evaluates the practicality and viability of a proposed plan or project. The purpose of a feasibility study is to identify potential issues and problems that could arise during the project's execution and assess the feasibility of the proposed system.

The feasibility study for the proposed system comprises an analysis of three critical aspects of feasibility, including technical, economic, and behavioural. The technical feasibility aspect examines the ability of the proposed system to function effectively in the existing technological environment. It evaluates the availability of necessary hardware, software, and infrastructure required for the system's development and operation. It also assesses the feasibility of incorporating the latest technological advancements in the proposed system.

3.1.1 Economical Feasibility

- **Cost-Benefit Analysis**: Evaluate whether the costs associated with software development, hardware and software acquisition, feasibility study, and other related expenses are justified by the expected financial gains.
- **Long-Term Financial Benefits**: Determine if the FAN HUB: FORMULA ONE platform is capable of generating long-term financial benefits for the organization through increased fan engagement, ticket sales, and merchandise purchases.
- **Cost of Software Investigation**: Consider the expenses incurred in conducting a thorough software investigation, including requirements elicitation and analysis.
- **Hardware and Software Costs**: Calculate the estimated costs of acquiring and maintaining the necessary hardware and software components, including licensing fees.
- **Development Team Costs**: Assess the expenses associated with the software development team, including salaries, training, and other personnel-related costs.

3.1.2 Technical Feasibility

- **Assessment of Current Resources**: Evaluate the existing hardware and software infrastructure to determine if it can support the development and operation of the FAN HUB: FORMULA ONE platform.
- **Technology Stability**: Ensure that the chosen technologies (React JS for frontend and .NET for backend) are stable and well-established, with a track record of reliability.

- **Team Capabilities**: Analyze the technical skills and capabilities of the software development team members to ensure they have the expertise required for the project.
- **User Base**: Verify that the selected technologies have a substantial user base, allowing for support and consultation if issues arise during development or operation.

3.1.3 Operational Feasibility:

- **Priority of User Requirements**: Determine whether the identified problems in user requirements are of high priority and significance to the Formula 1 fan community.
- **Acceptance of Proposed Solution**: Assess whether the solution proposed by the software development team is acceptable to the potential users and stakeholders.
- **User Adaptation**: Analyze whether Formula 1 fans will adapt to the new platform and its features, considering potential resistance to change.
- **Organizational Satisfaction**: Ensure that the organization is satisfied with the proposed alternative solutions and their alignment with business goals.

.

3.1.4 Feasibility Study Questionnaire

1. Project Overview?

The project is an online book store web application designed to facilitate the purchase and sale of books over the internet. It provides a platform for users to explore a vast catalog of books, including various genres and formats. The system allows customers to create accounts, browse books, add them to the shopping cart and make purchases. It also offers administrative control for managing users and sellers.

2.To what extent the system is proposed for?

The system is proposed as an all-encompassing platform catering to Formula One enthusiasts. It covers functionalities such as race information, ticket booking, user discussions, and polls. The platform aims to engage both fans and contributors, offering a complete Formula 1 experience online.

3. Specify the Viewers/Public involved in the System?

The viewers/public involved in the system include:

- **Fans/Viewers:** Individuals interested in accessing information, participating in discussions, and staying updated on Formula One events.
- **Administrators:** System administrators responsible for managing the website, user interactions, and content.

.

4. List the Modules included in your System?

The system is divided into two main modules:

- **Admin:** Responsible for overall website management and content moderation.
- **Fan/Viewer:** Individuals accessing information, participating in discussions, and booking tickets.

5. Identify the users in your project?

The users in the project are categorized into two main roles:

- **Admins:** Responsible for overall website management and moderation.
- **Fans/Viewers:** Individuals interested in Formula 1 events, discussions, and ticket booking.

6. Who owns the system?

Ownership of the system may reside with the organization or team responsible for developing the Formula One project. It could be owned by project stakeholders, developers, or a separate business entity.

7. System is related to which firm/industry/organization?

The system is related to the Formula One industry, providing an online platform for fans, teams, and drivers.

8. Details of the person that you have contacted for data collection?

- Sarosh Hataria, Ahura Racing Academy, Coimbatore
- Trusted internet resources and official site of individual teams as well as f1.com

9. Questionnaire to collect details about the project?

1. What inspired you to create the Fan Hub: Formula One project?

To enhance the fan experience and create a unique platform for Formula 1 enthusiasts.

2. Can you explain the key features and functionalities of the platform in more detail?

It includes exploring F1 history, engaging in discussions, and booking race tickets, all integrated with team websites.

3. What technologies are you using for the frontend and backend of the project?

React JS for frontend and .NET for the backend.

4. How do you envision fans benefiting from this platform compared to traditional Formula One websites?

Fans can explore history, engage with teams, book tickets and cancel or upgrade it all in one place, simplifying their experience.

5. Can you describe the user registration and login process?

Users register by providing personal details, including their full name, email, age. For ticket booking, users need to provide details of their Aadhar card or driving license, date of birth, and mobile number. Data will be verified at the Race time. For login, users enter their registered username and password. The system verifies their credentials, and essential data are provided.

6. How will users be able to browse and select race tickets? Can they also cancel or upgrade their tickets?

Users can browse and select race tickets by choosing from available options based on seating categories and prices. They can also cancel tickets within a specified timeframe and, upgrade to preferred categories from the waiting list when tickets become available.

7. What kind of content will be available in the open forum, and how will discussions be moderated?

Discussions will cover general F1 topics. Moderation ensures respectful and relevant discussions.

8. What administrative functions will be available for content management, ticket management, and user management?

Admins can edit content, manage ticket availability, view bookings, manage users, and moderate the forum.

9. How do you plan to ensure data security and privacy for users of the Fan Hub: Formula One platform?

Security measures include encryption, access controls, and regular security audits to protect user data.

10. Will the Fan Hub: Formula One platform offers a mobile app in addition to the web version?

Yes, we are planning to develop a mobile app for both iOS and Android platforms to provide users with a convenient and accessible mobile experience alongside the web version.

3.2 SYSTEM SPECIFICATION

3.2.1 Hardware Specification

Processor	-	Intel core i3
RAM	-	8 GB
Hard disk	-	512 GB

3.2.2 Software Specification

Front End	-	REACT JS, HTML, CSS, Bootstrap, Material UI
Back End	-	Dotnet Core 7
Database	-	SQL Server
Client on PC	-	Windows 7 and above.
Technologies used	-	C#, JS, HTML5, jQuery, CSS

3.3 SOFTWARE DESCRIPTION

3.3.1 React

React is a powerful JavaScript library for building user interfaces, chosen for its efficiency, flexibility, and the ability to create dynamic and interactive components. It allows for the creation of reusable UI components, making the development process more modular and maintainable. Reacts virtual DOM ensures efficient updates, enhancing the overall performance of web applications. Its popularity and extensive community support make it an excellent choice for developing the frontend of projects like Formula One Fan Hub.

3.3.2 .NET 7 and SQL Server

.NET 7, in conjunction with SQL Server, forms a robust backend solution for the Formula One Fan Hub project. .NET 7 is the latest version of the .NET framework, offering enhanced performance, improved language features, and increased developer productivity. It supports modern application development and provides a seamless integration with various platforms.

SQL Server, a relational database management system developed by Microsoft, ensures efficient and secure data storage and retrieval. Its scalability and reliability make it well-suited for handling the complex data requirements of Formula One Fan Hub. The combination of .NET 7 and SQL Server provides a powerful and cohesive backend infrastructure, supporting the project's functionality and ensuring data integrity.

CHAPTER 4

SYSTEM DESIGN

4.1 INTRODUCTION

The process of System Design entails the development of a comprehensive and, structured blueprint aimed at ensuring the optimal functionality of a given system in accordance with the specific requirements and expectations of its users. This entails the selection of constituent components of the system as well as determining their compatibility and integration with one another. System design involves the process of determining the essential requirements of a system and subsequently transforming them into tangible outcomes. The process of system development entails the endeavour of implementing enhancements to a pre-existing system. This implies the utilization of manifold approaches to enhance the efficiency and functionality of the system. It is imperative to implement a comprehensive strategy to effectively manage the requirements and aesthetics involved in ensuring seamless operability.

4.2 UML DIAGRAM

When people make computer programs using something called object-oriented programming, they have to do things in a certain way. Software engineers use a special language called Unified Modelling Language. This is how most people describe how someone makes a computer program. In Object-Oriented Programming, we treat difficult steps as things we can use called objects. Everything can be put into one of these groups. The issue is how we communicate with and direct them, even when they are capable of doing things.

The following nine diagrams are part of UML.

UML include the following diagrams:

- Class diagram
- Object diagram
- Use case diagram
- Sequence diagram
- Activity diagram
- Collaboration Diagram
- Deployment diagram
- Component diagram

4.2.1 USE CASE DIAGRAM

A use case diagram may be a graphical delineation that appears how clients and other outside on-screen characters associated with a systems inside components. A utilize case diagram's essential work is to perceive, layout, and orchestrate a system's utilitarian needs as seen through the eyes of its clients. The Unified Modelling Language (UML), a standard language for modelling actual things and systems, is frequently used to construct use case diagrams.

Use cases can be utilized to achieve an assortment of framework objectives, counting setting fundamental prerequisites, confirming equipment plans, testing and investigating program, creating online offer assistance references, or performing client bolster obligations. Customer support, product obtaining, catalogue overhauling, and payment processing are as it were a couple of illustrations of use cases within the setting of item deals.

The system boundaries, actors, use cases, and their connections together make up a use case diagram. The system boundary establishes the system's boundaries in reference to its surroundings. Actors are often defined depending on the roles they play and reflect the people or systems that interact with the system. The precise activities or behaviors that actors carry out within or close to the system is known as use cases. Finally, the graphic shows the connections between actors and use cases as well as the use cases themselves.

Use case diagrams are graphical representations used to capture the functional requirements of a system. When drawing a use case diagram, it is important to follow these guidelines to ensure an efficient and effective diagram:

- Choose descriptive names for use cases that accurately reflect the functionalities they perform.
- Assign appropriate names to actors to help identify their roles in the system.
- Ensure that relationships and dependencies are clearly depicted in the diagram.
- Avoid including every possible relationship, as the main goal is to identify the essential requirements.
- Use notes when necessary to clarify important points.

By following these guidelines, we can create a clear and concise use case diagram that accurately represents the functional requirements of the system.

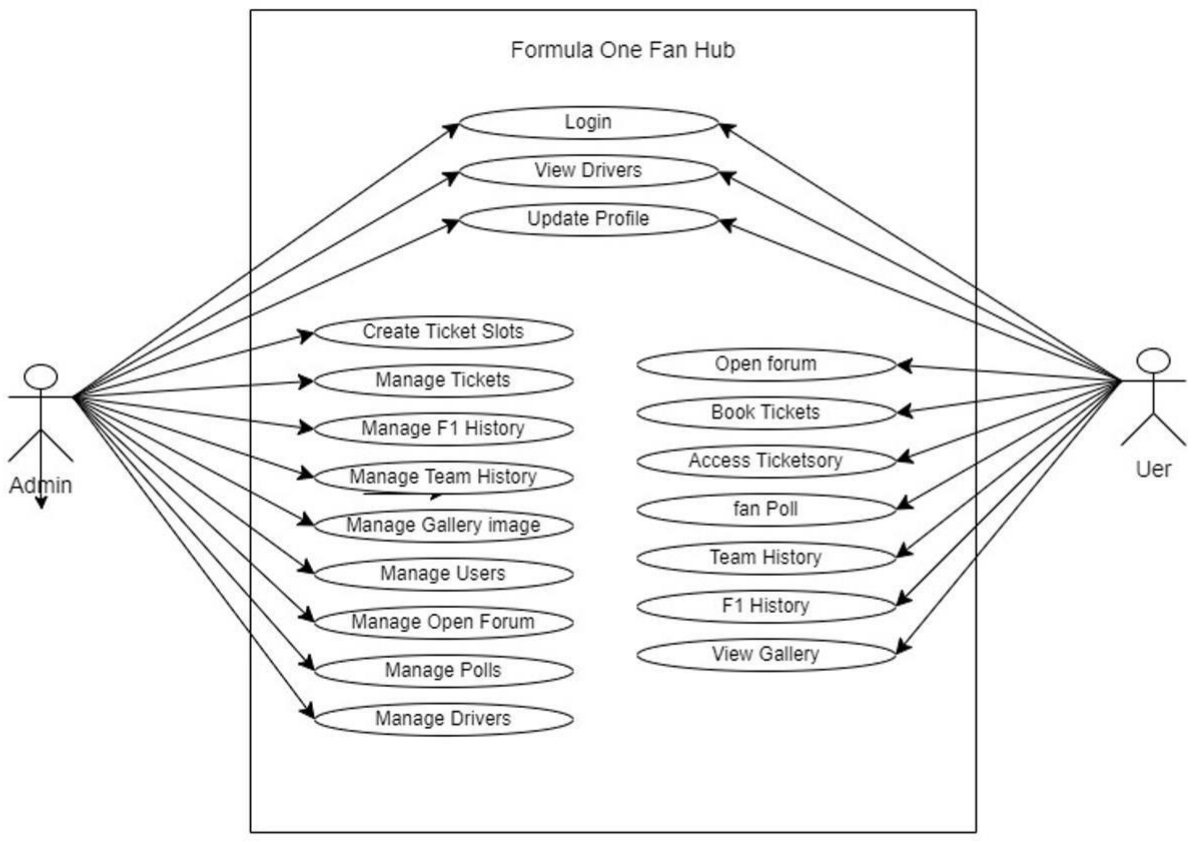


Fig.1 Use Case Diagram

4.2.2 SEQUENCE DIAGRAM

This diagram shows how things work together in order. A sequence diagram is another way to show information, and it's sometimes called event diagrams or event scenarios. Sequence maps show what actions the system's parts do and when they do them. Businesspeople and software developers use plans and explanations to help them understand both current and future systems.
Sequence Diagram Notations:

Actors – In a UML diagram, an actor is like a character who talks with the system and things in it. This means that we don't include actors in the UML diagram because they're not part of the system being modelled. We use actors who play different roles, like people and other things outside of a computer system. We use a drawing of a person with a stick figure to represent an actor in a UML diagram. We can show different characters in a sequence drawing.

Lifelines – A lifeline is a symbol representing a person in a picture of steps. Each part of a sequence diagram is shown as a lifeline. The parts that show the steps in a sequence are at the top of a sequence diagram and are called lifeline elements.

Messages – Objects talk to each other by sending messages. These messages are shown in a on the lifeline, in the order they are received. We use arrows to show messages. The most important parts of a sequence diagram are the lines that connect things and the notes that explain what's happening.

Guards – Objects talk to each other by sending messages. These messages are shown in a list on the lifeline, in the order they are received. We use arrows to show messages. The most important parts of a sequence diagram are the lines that connect things and the notes that explain what's happening.

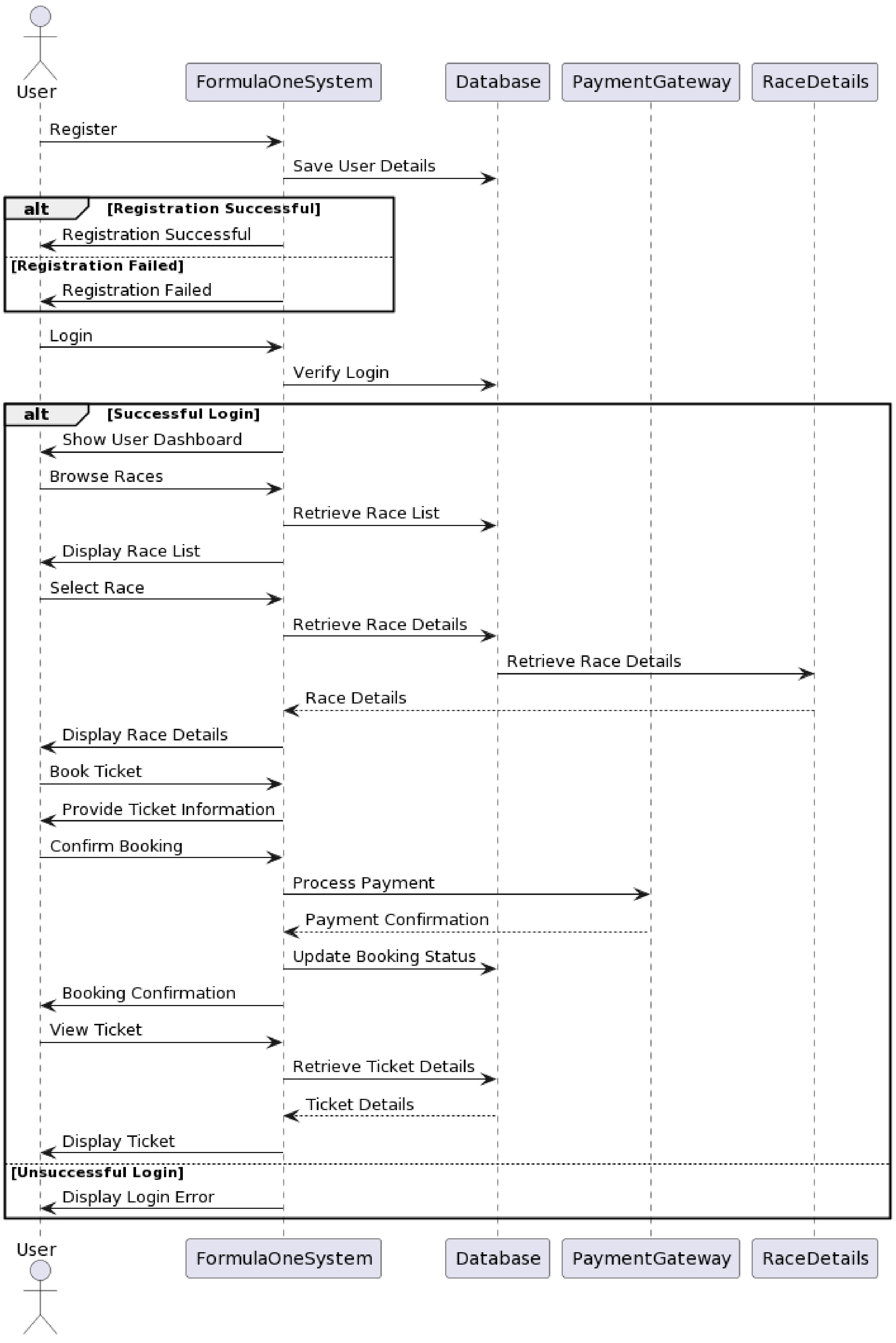


Fig 1. sequence diagram

4.2.3 State Chart Diagram

A State Chart Diagram, a fundamental component of UML, provides a visual representation of an object's lifecycle states and the transitions between them. It depicts the dynamic behavior of an entity in response to events, showcasing how it transitions from one state to another. Each state represents a distinct phase in the object's existence, while transitions illustrate the conditions triggering state changes. Initial and final states mark the commencement and termination of the object's lifecycle. Orthogonal regions allow for concurrent states, capturing multiple aspects of the object's behavior simultaneously. Hierarchical states enable the representation of complex behaviors in a structured manner. Entry and exit actions depict activities occurring upon entering or leaving a state. Moreover, guard conditions ensure that transitions occur only under specified circumstances. State Chart Diagrams play a crucial role in understanding and designing the dynamic behavior of systems, aiding in the development of robust and responsive software applications.

Key notations for State Chart Diagrams:

- Initial State: Represented by a filled circle, it signifies the starting point of the object's lifecycle.
- State: Depicted by rounded rectangles, states represent distinct phases in an object's existence.
- Transition Arrow: Arrows denote transitions between states, indicating the conditions triggering a change.
- Event: Events, triggers for state changes, are labeled on transition arrows.
- Guard Condition: Shown in square brackets, guard conditions specify criteria for a transition to occur.
- Final State: Represented by a circle within a larger circle, it indicates the end of the object's lifecycle.
- Concurrent State: Represented by parallel lines within a state, it signifies concurrent behaviors.
- Hierarchy: States can be nested within other states to represent complex behavior.
- Entry and Exit Actions: Actions occurring upon entering or leaving a state are labelled within the state.
- Transition Labels: Labels on transition arrows may indicate actions or operations that accompany the transition.

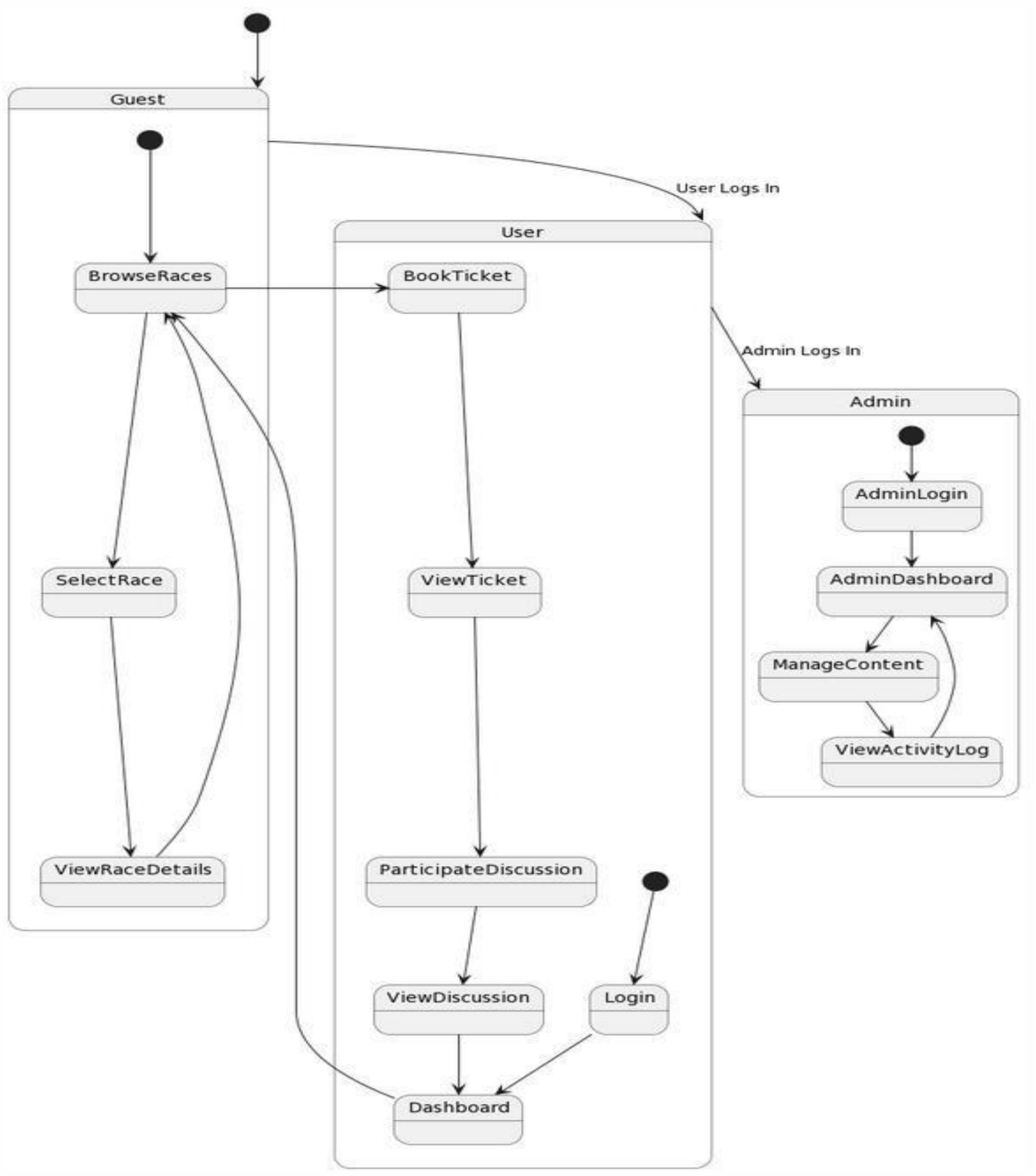


Fig 1. state chart diagram

4.2.4Activity Diagram

This diagram is a crucial UML diagram that depicts the dynamic aspects of a system. It functions as a flowchart, illustrating the progression from one activity to another. The actions within the system can be described using system operations, and the control flow shows how one action connects to the next. This flow can involve concurrency, parallelism, or branching. An activity diagram is a behavioural diagram, meaning it shows how a system behaves.

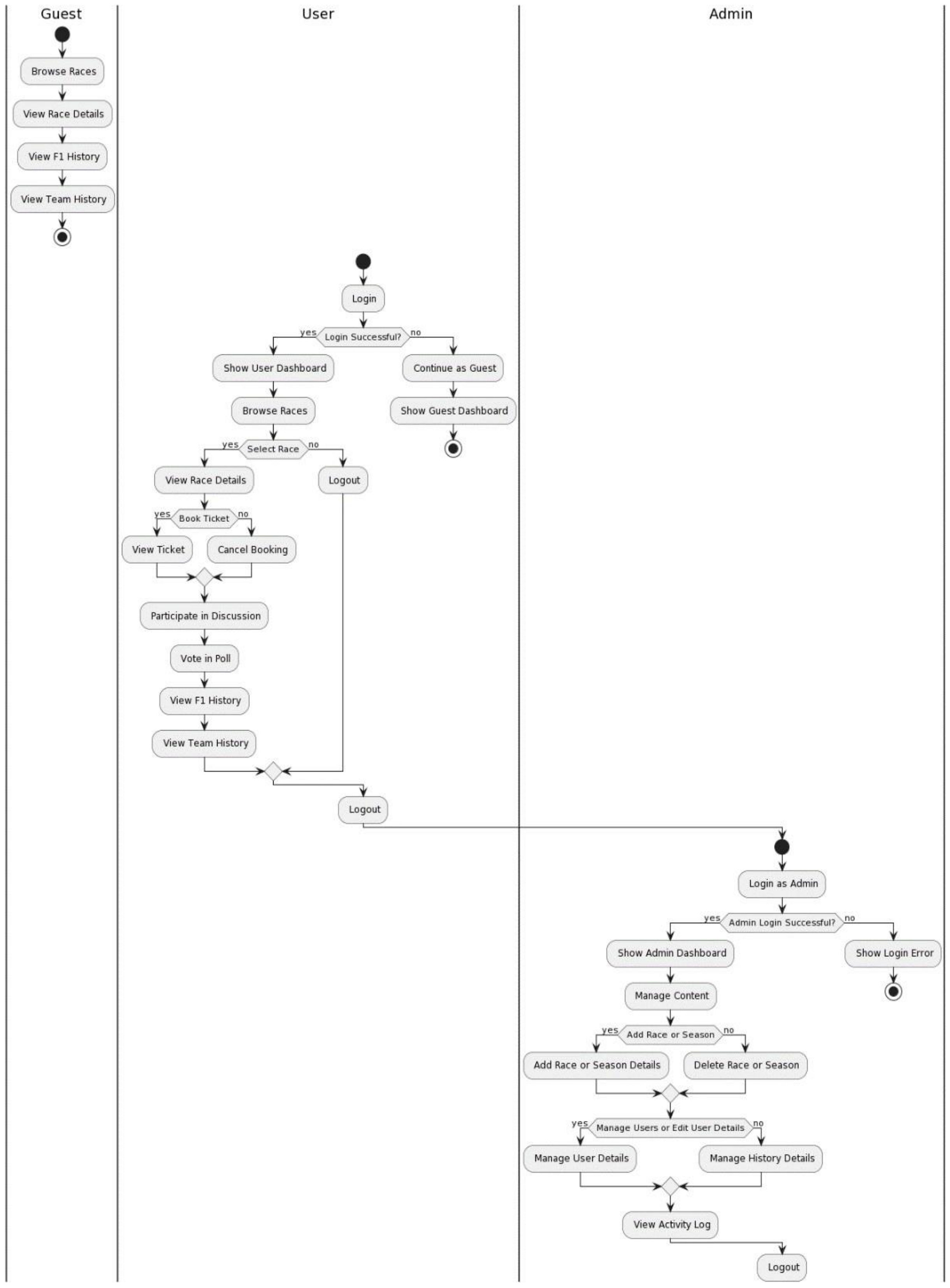


Fig 1. Activity diagram

4.2.5 Class Diagram

The class diagram is a fundamental component of object-oriented modeling and serves as the primary means of conceptual modeling for the structure of an application. Additionally, class diagrams can be used for detailed modeling that can be translated into programming code. They can also be employed for data modeling purposes.

Class diagrams are a crucial component of UML used to represent classes, objects, interfaces, and their relationships and attributes in a system. Some important components of a class diagram are:

• **Class:** It is a blueprint or template for creating objects and is represented as a rectangle with the class name, attributes, and methods.

• **Interface:** It is a collection of abstract methods that specify a contract between a class and the outside world. It is represented as a circle with the interface name inside.

• **Object:** It is an instance of a class with state and behavior. It is represented as a rectangle with the object name inside.

• **Association:** It is a relationship between two classes that represents a connection or link and is represented as a line with optional directionality, multiplicity, and role names.

• **Aggregation:** It is a part-whole relationship where the whole (aggregator) is composed of parts (aggregates) and is represented as a diamond shape on the aggregator side.

• **Composition:** It is a stronger form of aggregation where the parts cannot exist without the whole and is represented as a filled diamond shape on the aggregator side.

• **Inheritance:** It is a relationship between a superclass and its subclasses that represents an "is-a" relationship and is represented as a line with an open arrowhead pointing from the subclass to the superclass.

• **Dependency:** It is a relationship where a change in one class may affect the other class and is represented as a dashed line with an arrowhead pointing from the dependent class to the independent class.

• **Multiplicity:** It represents the number of instances of a class that can be associated with another class and is represented as a range of values near the association or aggregation line.

Class diagrams are essential in designing and modeling object-oriented software systems as they provide a visual representation of the system's structure, its functionality, and the relationships between its objects. They facilitate software development, maintenance, and improve communication among team members.

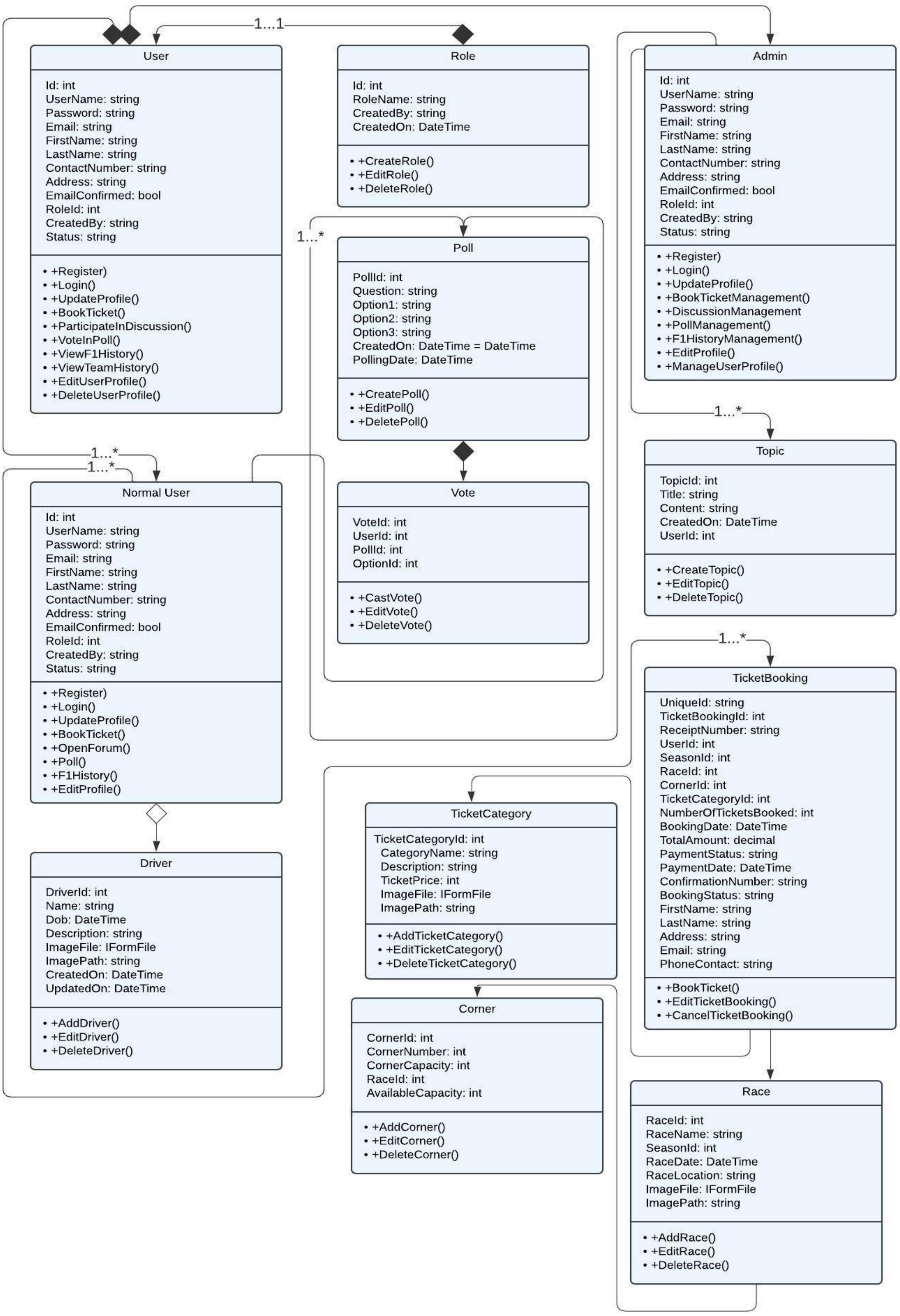


Fig 1. Class Diagram

4.2.6 Object Diagram

These diagrams need class diagrams, which are made from the source of class diagrams. An object diagram shows a real example of a class diagram. Class diagrams and object diagrams have similar main ideas. Object diagrams show what a system looks like at a single moment. It's like a picture of the system. Object diagrams let you see a bunch of stuff and how they're connected.

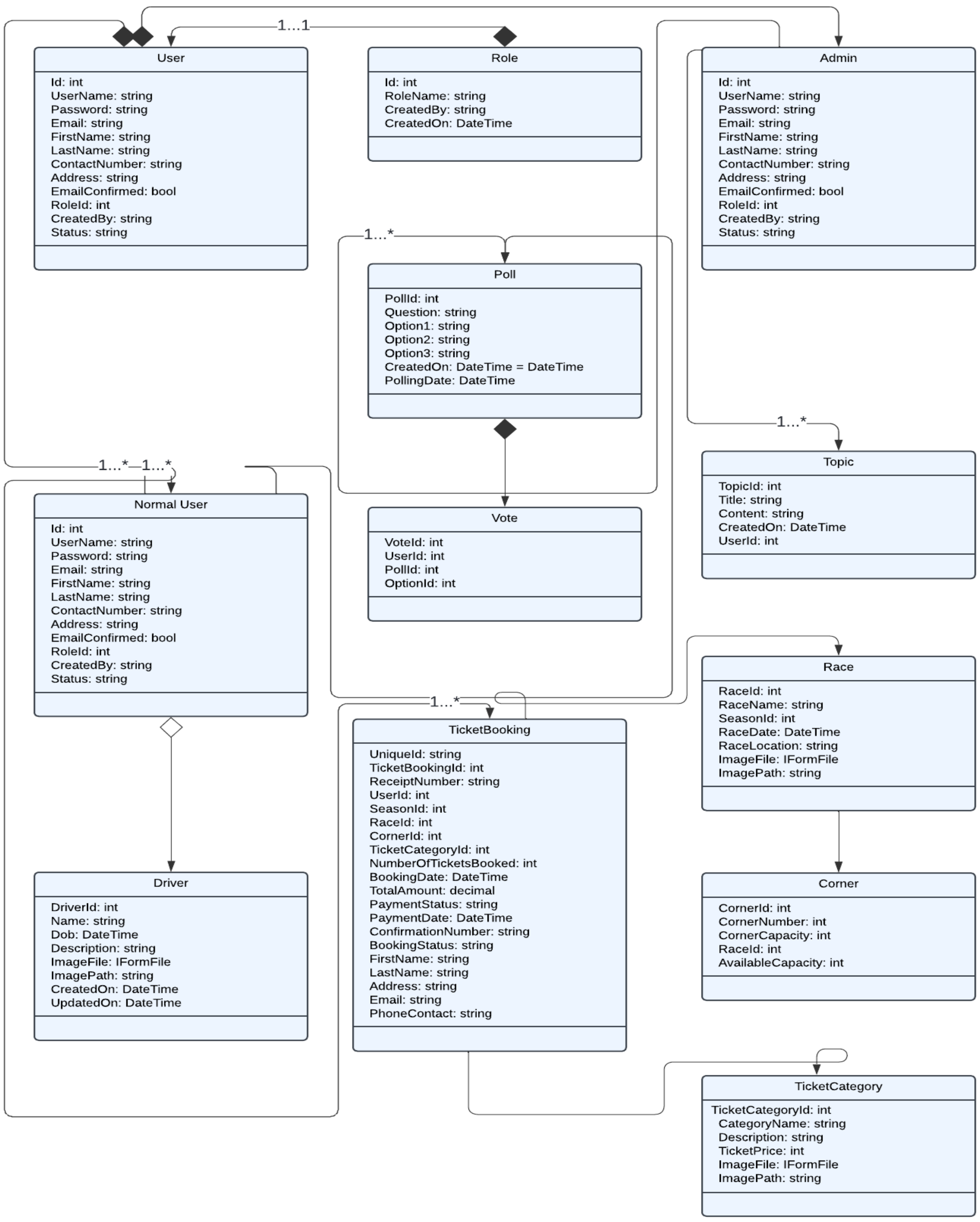


Fig 1. Object Diagram

4.2.7 Component Diagram

There are many different types of component diagrams with different features and characteristics. Component diagrams show the different parts of a system. There are things like programs, books, and papers that are kept in a computer that we call nodes. We use component diagrams to display how different parts of a system connect and are set up. These drawings can help make things that can work.

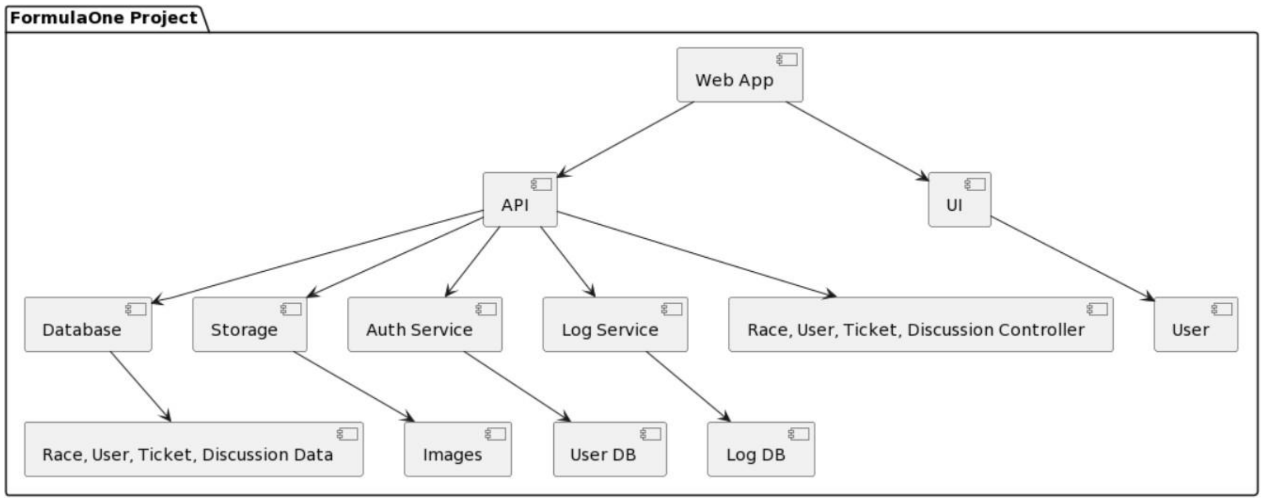


Fig 1. Component Diagram

4.2.8 Deployment Diagram

There are many different types of component diagrams with different features and characteristics. Component diagrams show the different parts of a system. There are things like programs, books, and papers that are kept in a computer that we call nodes. We use component diagrams to display how different parts of a system connect and are set up. These drawings can help make things that can work.

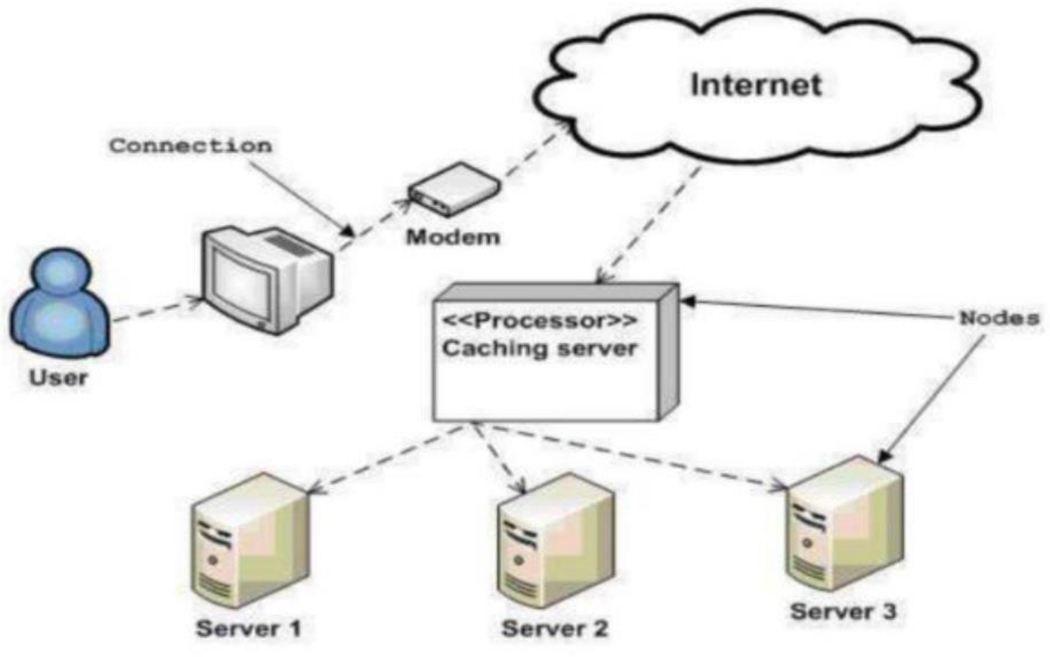


Fig 1. Deployment Diagram

4.3 USER INTERFACE DESIGN USING FIGMA

Form Name: User Registration

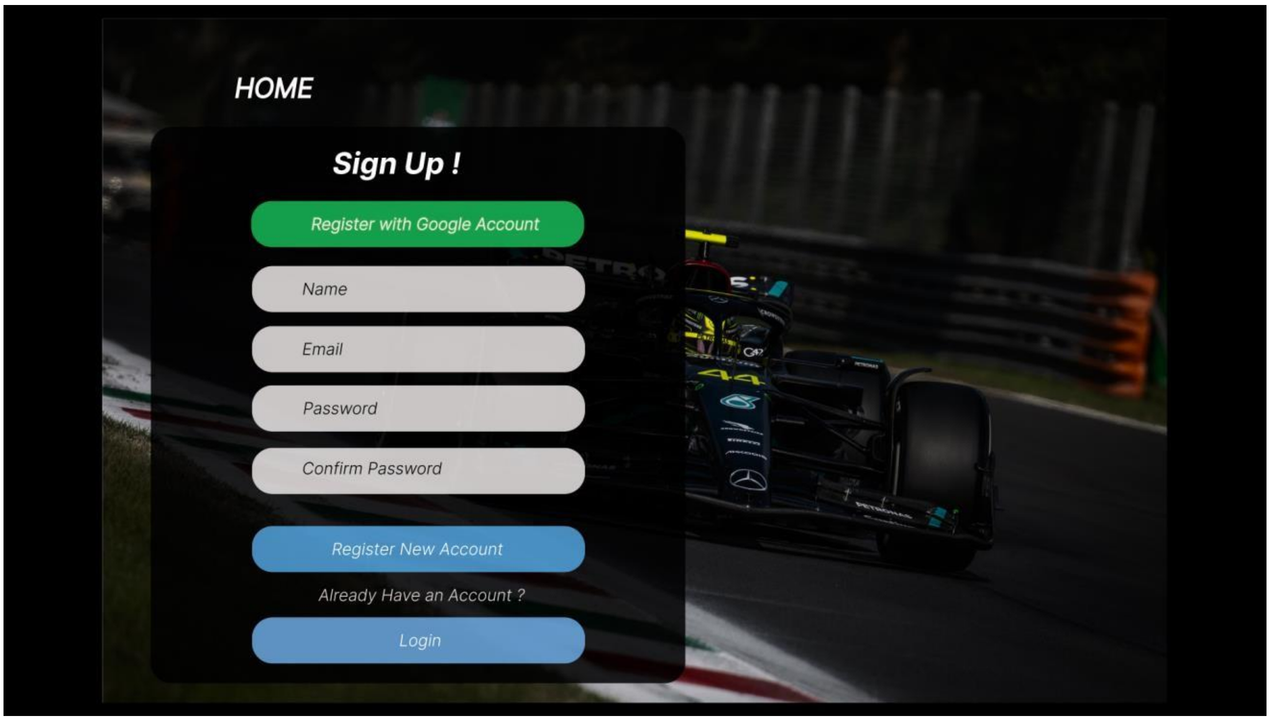


Fig 1. User Registration

Form Name: User Login

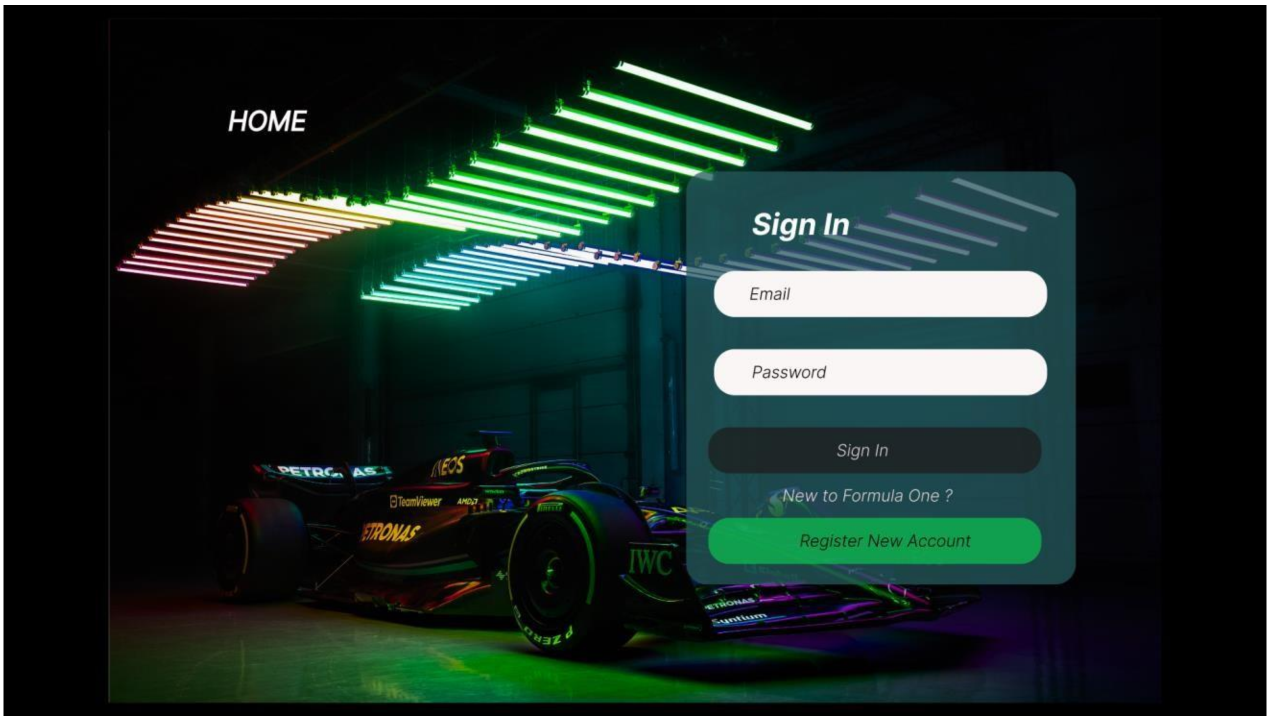


Fig 2. User Login

Form Name: Home page

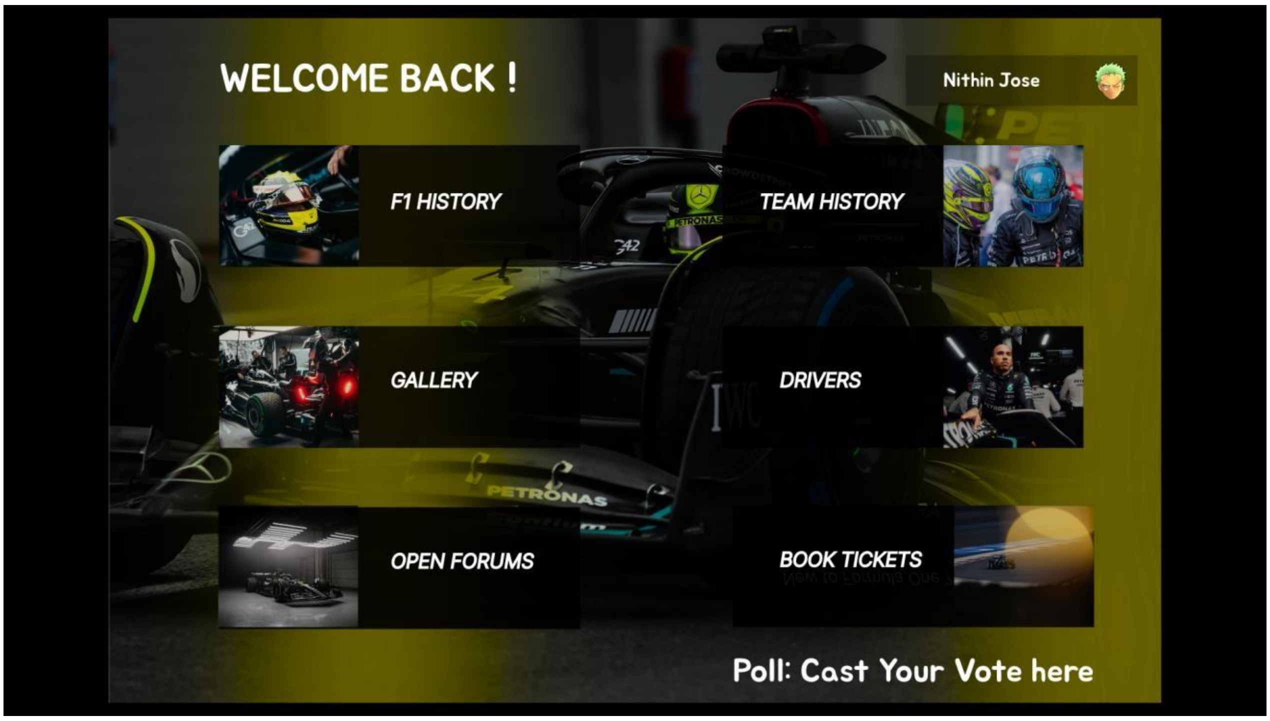


Fig 3 Home Page

Form Name: User HomePage

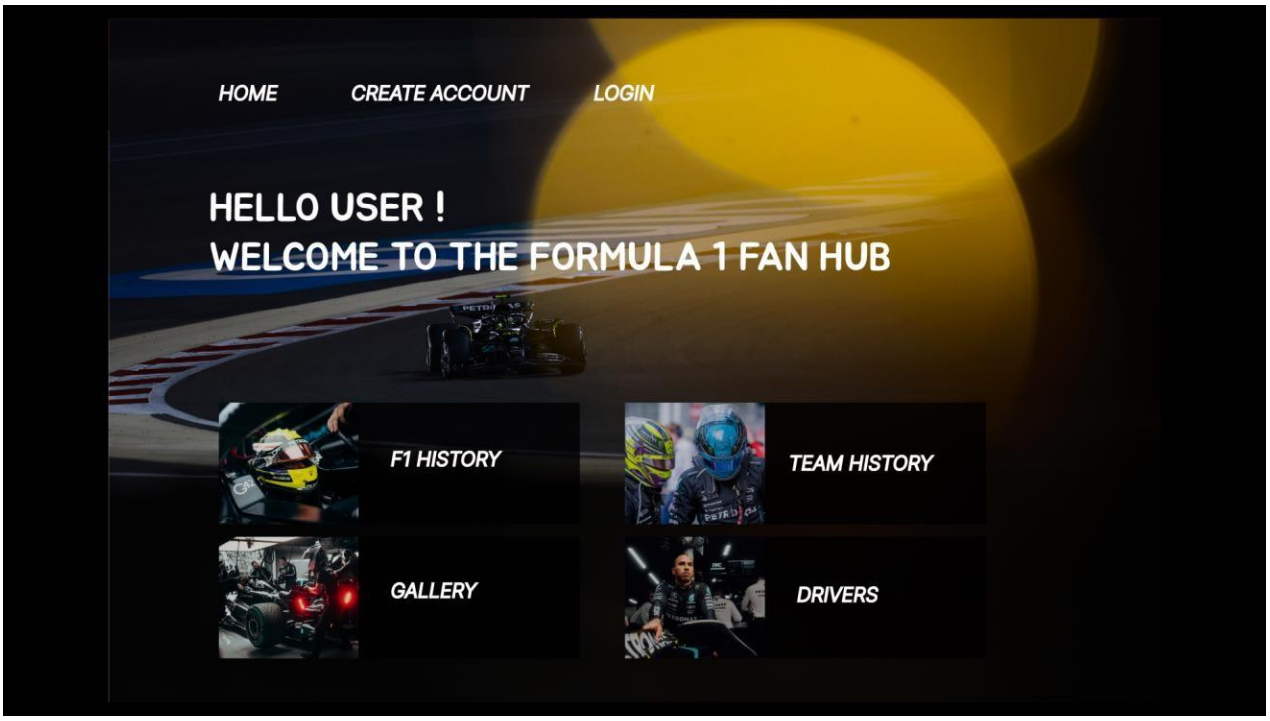


Fig 4 User Home

4.4 DATABASE DESIGN

A database is like a storage place where you can save information and then find it again easily. The reason for having a database is to store information. It’s important to keep the information safe. To make a database, you need to do two things. Jot down what the user wants so, that you can create a document that exactly meets their requirements. The creation of data is done in a different way for each database system and it’s called data-level creation.

4.4.1 Relational Database Management System (RDBMS)

The database is represented as a collection of relations in a relational paradigm. A table or file of records with values can be compared to each relation. In formal relational model terminology, a row is referred to as a tuple, a column heading is referred to as an attribute, and the table is referred to as a relation. Numerous tables, each with a unique name, make up a relational database. Each row in a tale represents a collection of related values, relations, domains, and attributes.

4.4.2 Normalization

Data are organized in a simple manner to minimize the impact of future changes on data structures. Normalization is a formal process that eliminates redundancy and promotes integrity by separating redundant fields and breaking up large tables into smaller ones. Its purpose is to avoid anomalies during insertion, deletion, and updating of data. Normal form in data modelling is based on two concepts, keys and relationships. A primary key uniquely identifies a row in a table, while a foreign key identifies a record from a different table. All tables are normalized up to the third normal form to achieve a sensible organization of data into proper tables and columns that can be easily correlated by the user. Normalization eliminates repeating groups of data, thereby avoiding data redundancy and reducing the burden on computer resources.

4.4.2.1 First Normal Form

According to the First Normal Form, each attribute's domain must only include atomic values, and each attribute's value in a tuple must be a single value from that domain. In other words, 1NF forbids using relationships as attribute values within tuples or relations within relations. Single atomic or indivisible values are the only attribute values that are permitted under 1NF. The data must first be entered into First Normal Form. This can be accomplished by separating data into tables of a similar type in each table. Each table is given a Primary Key or Foreign Key as per requirement of the project. In this we form new relations for each non-atomic attribute or nested relation. This eliminated repeating groups of data. A relation is said to be in first normal form if only if it satisfies the constraints that contain the primary key only.

4.4.2.2 Second Normal Form

No non-key attribute should be functionally dependent on a portion of the main key for relations where the primary key has several attributes, according to Second Normal Form. This involves breaking down each partial key into its dependent characteristics and setting up a new relation for each one. Keep the original primary key and any properties that are entirely dependent on it in your database. This procedure aids in removing data that depends only on a small portion of the key. If and only if a relation satisfies all the requirements for first normal form for the primary key and all of the non-primary key qualities of the relation are completely dependent on the primary key alone, then that relation is said to be in second normal form.

4.4.2.3 Third Normal Form

Relation should not have a non-key attribute that is functionally determined by another non-key attribute or by a collection of non-key attributes, according to the Third Normal Form. The primary key should not be transitively dependent, in other words. The non- key attributes that functionally determine other non-key attributes are decomposed in this way put up in relation. This procedure is used to eliminate anything not wholly dependent on the Primary Key. Only when a relation is in second normal form and, more importantly, when its non-key characteristics do not depend on those of other non- key attributes, is it considered to be in third normal form.

4.4.3 Sanitization

Data validation is a crucial aspect of web development, particularly when dealing with forms that require users to input personal information which is then sent to the database. Any data submitted in an invalid format has the potential to compromise the security of the DBMS. Therefore, to prevent hackers from gaining access to the database, it is essential to sanitize and filter all user- entered data before sending it to the database. By ensuring that user-entered data is properly formatted, meets certain criteria, and is free from malicious input, developers can prevent errors and security vulnerabilities in their applications.

4.4.4 Indexing

Indexing is a technique that enhances the speed of retrieving specific data from a database by creating an index table. This reduces the number of disks required to access data, and it works by storing a sorted copy of the Primary or Candidate Keys. The index takes a search key as input and efficiently returns a collection of matching records. When creating an index, the database management system creates a separate table that contains the values of the indexed column or columns and the location of the corresponding records. This table is sorted by the values of the indexed column or columns, which makes searching for specific data much faster. In addition to improving query performance, indexes can also help ensure data integrity by enforcing uniqueness constraints on columns that have been indexed. This means that the database management system will not allow duplicate values to be inserted into the indexed column.

4.5 TABLE DESIGN

1. Comment

Primary key: CommentId

Foreign Key: UserId references User,

TopicId references Topic

No	Field name	Datatype	Key Constraints	Description of the field
1	CommentId	int	Primary Key	Key for this table
2	Content	string	Not Null	Content of the comment
3	CreatedOn	DateTime	Not Null	Date and time when the comment is created
4	UpdatedOn	DateTime	Not Null	Date and time when the comment is updated
5	UserId	int	Foreign Key	Identifier of the user who made the comment
6	TopicId	int	Foreign Key	Identifier of the associated topic

2. Corner

Primary key: CornerId

Foreign Key: RaceId references Race

No	Field name	Datatype	Key Constraints	Description of the field
1	CornerId	int	Primary Key	Key for this table
2	CornerNumber	int	Not Null	Number assigned to the corner
3	CornerCapacity	int	Not Null	Capacity of the corner
4	RaceId	int	Foreign Key	Identifier of the associated race
5	AvailableCapacity	int	Not Null	Available capacity of the corner

3. Driver

Primary key: DriverId

No	Field name	Datatype	Key Constraints	Description of the field
1	DriverId	int	Primary Key	Key for this table
2	Name	string	Not Null	Name of the driver
3	Dob	DateTime	Not Null	Date of birth of the driver
4	Description	string	Not Null	Description of the driver
5	ImageFile	IFormFile	Not Null	Image file of the driver
6	ImagePath	string	Not Null	Path to the image file of the driver
7	CreatedOn	DateTime	Not Null	Date and time when the driver is created
8	UpdatedOn	DateTime	Not Null	Date and time when the driver is updated

4. ErrorLog

Primary key: Id

No	Fieldname	Datatype	Key Constraints	Description of the field
1	Id	int	Primary Key	Key for this table
2	Message	string	Not Null	Error message
3	Source	string	Not Null	Source of the error
4	StackTrace	string	Not Null	Stack trace information
5	TargetSite	string	Not Null	Target site of the error
6	InnerException	string	Not Null	Inner exception information
7	Data	string	Not Null	Additional data related to the error
8	HelpLink	string	Not Null	Help link for the error
9	HResult	string	Not Null	Result code of the error
10	Type	string	Not Null	Type of the error
11	User	string	Not Null	User related to the error
12	CreatedOn	DateTime	Not Null	Date and time when the error is logged

5. F1History

Primary key: HistoryId

No	Fieldname	Datatype	Key Constraints	Description of the field
1	HistoryId	int	Primary Key	Key for this table
2	Heading	string	Not Null	Heading of the F1 history
3	Paragraph	string	Not Null	Paragraph describing the F1 history

6. Gallery

Primary key: ImageId

No	Fieldname	Datatype	Key Constraints	Description of the field
1	ImageId	int	Primary Key	Key for this table
2	ImageUrl	string	Not Null	URL of the gallery image
3	Caption	string	Not Null	Caption for the gallery image

7. Role

Primary key: Id

No	Fieldname	Datatype	Key Constraints	Description of the field
1	Id	int	Primary Key	Key for this table
2	RoleName	string	Not Null	Name of the role
3	CreatedBy	string	Not Null	User who created the role
4	CreatedOn	DateTime	Not Null	Date and time when the role is created

8. Season

Primary key: SeasonId

No	Fieldname	Datatype	Key Constraints	Description of the field
1	SeasonId	int	Primary Key	Key for this table
2	Year	int	Not Null	Year of the season
3	Champion	string	Not Null	Champion of the season
4	ImageFile	IFormFile	Not Null	Image file of the season
5	ImagePath	string	Not Null	Path to the image file of the season

9. Poll

Primary key: PollId

No	Fieldname	Datatype	Key Constraints	Description of the field
1	PollId	int	Primary Key	Key for this table
2	Question	string	Not Null	Question for the poll
3	Option1	string	Not Null	Option 1 for the poll
4	Option2	string	Not Null	Option 2 for the poll
5	Option3	string	Not Null	Option 3 for the poll (Nullable)
6	CreatedOn	DateTime	Not Null	Date and time when the poll is created
7	PollingDate	DateTime	Not Null	Date when the poll is scheduled for polling

10. Race

Primary key: RaceId

Foreign Key: SeasonId references Season

No	Fieldname	Datatype	Key Constraints	Description of the field
1	RaceId	int	Primary Key	Key for this table
2	RaceName	string	Not Null	Name of the race
3	SeasonId	int	Foreign Key	Identifier of the associated season
4	RaceDate	DateTime	Not Null	Date of the race
5	RaceLocation	string	Not Null	Location of the race
6	ImageFile	IFormFile	Not Null	Image file of the race
7	ImagePath	string	Not Null	Path to the image file of the race

11. TeamHistory

Primary key: HistoryId

No	Fieldname	Datatype	Key Constraints	Description of the field
1	HistoryId	int	Primary Key	Key for this table
2	Heading	string	Not Null	Heading of the team history
3	Paragraph	string	Not Null	Paragraph describing the team history

12. TicketCategory

Primary key: TicketCategoryId

No	Fieldname	Datatype	Key Constraints	Description of the field
1	TicketCategoryId	int	Primary Key	Key for this table
2	CategoryName	string	Not Null	Name of the ticket category
3	Description	string	Not Null	Description of the ticket category
4	TicketPrice	int	Not Null	Price of the ticket category
5	ImageFile	IFormFile	Not Null	Image file of the ticket category
6	ImagePath	string	Not Null	Path to the image file of the ticket category

13. Topic

Primary key: TopicId

Foreign Key: UserId references User

No	Fieldname	Datatype	Key Constraints	Description of the field
1	TopicId	int	Primary Key	Key for this table
2	Title	string	Not Null	Title of the topic
3	Content	string	Not Null	Content of the topic
4	CreatedOn	DateTime	Not Null	Date and time when the topic is created
5	UserId	int	Foreign Key	Identifier of the user who created the topic

14. Vote

Primary key: VoteId

No	Fieldname	Datatype	Key Constraints	Description of the field
1	VoteId	int	Primary Key	Key for this table
2	UserId	int	Not Null	User ID associated with vote
3	PollId	int	Not Null	Poll ID associated with vote
4	OptionId	int	Not Null	Option ID chosen by the user

15. User

Primary key: Id

Foreign Key: RoleId references Role

No	Fieldname	Datatype	Key Constraints	Description of the field
1	Id	int	Primary Key	Key for this table
2	UserName	string	Not Null	User's username
3	Password	string	Not Null	User's password
4	Email	string	Not Null	User's email address
5	FirstName	string	Not Null	User's first name
6	LastName	string	Not Null	User's last name
7	ContactNumber	string	Not Null	User's contact number
8	Address	string	Not Null	User's address
9	EmailConfirmed	bool	Not Null	Indicates if email is confirmed
10	RoleId	int	Foreign Key	Foreign key to Role
11	CreatedBy	string	Not Null	Created by user
12	Status	string	Not Null	User's status
13	CreatedOn	DateTime	Not Null	Date and time of creation

16. TicketBooking

Primary key: TicketBookingId

Foreign Key: UserId references User,

RaceId references Race,

CornerId references Corner,

TicketCategoryId references TicketCategory

No	Fieldname	Data type	Key Constraints	Description of the field
1	TicketBookingId	int	Primary Key	Key for this table
2	UniqueId	string	Not Null	Unique identifier for the ticket booking
3	ReceiptNumber	string	Not Null	Receipt number associated with the booking
4	UserId	int	Foreign Key	Foreign key to User
5	SeasonId	int	Not Null	Season ID associated with the booking
6	RaceId	int	Foreign Key	Foreign key to Race
7	CornerId	int	Foreign Key	Foreign key to Corner
8	TicketCategoryId	int	Foreign Key	Foreign key to TicketCategory
9	NumberOfTickets Booked	int	Not Null	Number of tickets booked
10	BookingDate	DateTime	Not Null	Date and time of booking
11	TotalAmount	decimal	Not Null	Total amount for the booking
12	PaymentStatus	string	Not Null	Payment status (Paid/Unpaid)
13	PaymentDate	DateTime	Not Null	Date and time of payment
14	ConfirmationNum ber	string	Not Null	Confirmation number
15	BookingStatus	string	Not Null	Booking status (Confirmed/Cancelled)
16	FirstName	string	Not Null	First name of the person booking
17	LastName	string	Not Null	Last name of the person booking
18	Address	string	Not Null	Address of the person booking
19	Email	string	Not Null	Email of the person booking
20	PhoneContact	string	Not Null	Phone contact of the person booking

CHAPTER 5

SYSTEM TESTING

5.1 INTRODUCTION

An index is a way to make databases work faster. A table in a database can be connected to one or more lists. When you create an index, you can give it instructions on what to include by typing in a specific phrase or command. The field expression is usually just one field name like EMP_ID. One example is a list of employee identification numbers that are organized in order. This list is stored in an index that is created for the EMP_ID column in a table. The list shows where each value comes from.

5.2 TEST PLAN

A plan that tells us how we will test a particular project or product is called a test plan. Selenium is a tool that helps test web applications automatically, and many people use it. A test plan tells you what steps you need to follow to finish testing different ways. The plan for the test shows what needs to be done. Software developers make computer programs, documents that explain how they work, and structures to hold the information used by the program. Software developers create computer programs, instructions that tell the computer what to do, and systems to organize the information used by the program. Software creators should review every section of their program to confirm that it operates as intended. There is a team called "Independent Test Group" that helps fix issues that can happen when the person who makes something tests it on their own. The objective of testing should be clearly defined in measurable terms. So, the plan for testing needs to have details like how long it takes for something to break, how much it costs to find and fix problems, how often problems still happen after being fixed, and how much time is needed for each type of test. The levels of testing include:

- Unit testing
- Integration Testing
- Data validation Testing
- Output Testing

5.2.1 Unit Testing

Unit testing is a verification process that focuses on the smallest software design unit – the software component or module. Control paths are tested based on the component level design description to uncover errors within the module's boundary. The complexity of tests and scope of unit testing are determined based on the component. This type of testing is white-box oriented and can be conducted in parallel for multiple components. The modular interface is tested to ensure proper information flow into and out of the program unit under test. Local data structures are examined to maintain data integrity during algorithm execution. Boundary conditions are tested to ensure that all statements in a module have been executed at least once, and all error handling paths are tested. Before any other test is initiated, tests of data flow across a module interface are required. Selective testing of execution paths is important in unit testing, and error handling paths should be established to reroute or terminate processing when errors occur. Boundary testing is the final task of unit testing because software often fails at its boundaries. In Sell-Soft System, each module was treated as a separate entity, and a wide range of test inputs were used to test each one individually. Some flaws in the internal logic of the modules were found and corrected. After coding, each module was tested and run individually, and unnecessary code was removed to ensure all modules worked correctly and produced the expected results.

5.2.2 Integration Testing

Integration testing is when software parts that work together are tested together. This type of testing tries to find problems when different parts of the software work together. Integration testing checks how well different parts of a program share information with each other. Integration and testing are sometimes called I&T, string testing, or thread testing.

5.2.3 Validation Testing or System Testing

The final stage of software testing involves testing the entire system as a complete unit, including all forms, code, modules, and class modules. This type of testing is commonly referred to as Black Box testing or System testing. Black Box testing is centered on testing the functional requirements of the software, whereby the software engineer derives input conditions to fully exercise all functional requirements of a program. The goal of Black Box testing is to identify errors in various categories such as incorrect or missing functions, interface errors, errors in data structures or external data access, performance errors, initialization errors, and termination errors.

5.2.4 Output Testing or User Acceptance Testing

During the user acceptance testing phase, the system is evaluated to ensure it meets the needs of the organization. The software must remain relevant to the user and be adaptable to changes as needed.

The testing is conducted with respect to the input and output screen designs, using various types of test data. The preparation of test data is crucial to the success of the system testing. Once the test data is ready, the system is tested using the data, and any errors uncovered are corrected using the testing steps mentioned above. Any necessary corrections are noted for future use..

5.2.5 Automation Testing

A group of tests called a test case suite is run using special computer programs that can test software automatically. This way of testing is called automation testing. A person checks the test step by step by using a computer. The testing program can create detailed reports, check if the results match what was predicted, and add test information into the software being checked. The process of automating software testing necessitates a significant investment of financial and material resources. Multiple iterations of the aforementioned test will be necessary throughout the course of the project's development. This instrument possesses the capability to both document and playback a series of evaluations at any given point in time. Automating the test suite renders individuals unnecessary, thereby obviating the need for further intervention. This made more money from using test automation. Automation aims to reduce the number of tests done by humans but not to completely remove manual testing.

5.2.6 Selenium Testing

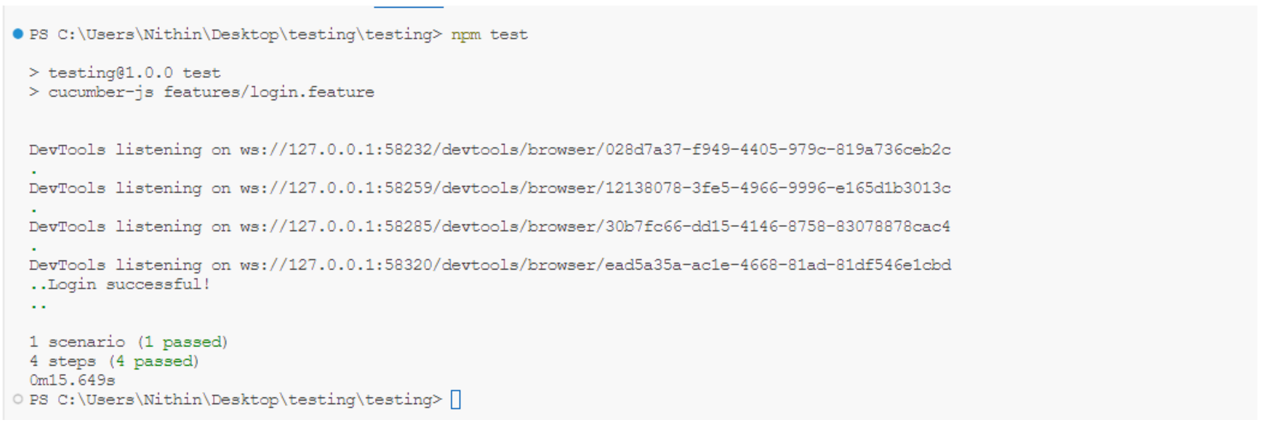
Selenium is a free tool that tests websites on different devices and web browsers. Selenium test scripts can be created using different programming languages like Java, C#, and Python. The testing process conducted utilizing the Selenium testing tool is commonly denoted as Selenium testing. Owing to the fact that Selenium comprises a variety of tools, it further involves the contributions of divergent developers. The following individuals are noteworthy for their substantial contributions to the Selenium project. Jason Huggins made Selenium in 2004. A person who designs things was making a website that needed to be checked often. He made a computer program called JavaScript to test their app automatically because doing it by hand was taking too long. He gave it the name "JavaScriptTestRunner".

Test Case 1: Login Test

Code

```
const { Given, When, Then } = require("cucumber");
const { Builder, By, until } = require("selenium-webdriver");
let driver;
Given("I am on the login page", async function () {
  driver = await new Builder().forBrowser("chrome").build();
  await driver.get("http://localhost:3000/Signin");
});
When("I enter valid credentials", async function () {
  await driver.findElement(By.name("username")).sendKeys("nithin");
  await driver.findElement(By.name("password")).sendKeys("Manjadi@123");
});
When("click on the login button", async function () {
  try {
    const button = await driver.wait(
      until.elementLocated(By.id("testid")),
      10000
    );
    await button.click();
    console.log("Login successful!");
  } catch (error) {
    console.error("Error clicking on the login button:", error);
  }
});
Then("I should be redirected to the home page", async function () {
  await driver.wait(until.urlIs("http://localhost:3000/UserHome"));
  await driver.quit();
});
```

Screenshot



```
PS C:\Users\Nithin\Desktop\testing\testing> npm test

  > testing@1.0.0 test
  > cucumber-js features/login.feature


  DevTools listening on ws://127.0.0.1:58232/devtools/browser/028d7a37-f949-4405-979c-819a736ceb2c
  .
  DevTools listening on ws://127.0.0.1:58259/devtools/browser/12138078-3fe5-4966-9996-e165d1b3013c
  .
  DevTools listening on ws://127.0.0.1:58285/devtools/browser/30b7fc66-dd15-4146-8758-83078878cac4
  .
  DevTools listening on ws://127.0.0.1:58320/devtools/browser/ead5a35a-ac1e-4668-81ad-81df546e1cbd
  ..Login successful!
  ..

  1 scenario (1 passed)
  4 steps (4 passed)
  0m15.649s
PS C:\Users\Nithin\Desktop\testing\testing>
```

Test Report

Test Case 1

Project Name: Formula One Fan Hub					
Login Test					
Test Case ID: Test1			**Test Designed By:** Nithin Jose		
Test Priority (Low/Medium/High): High			**Test Designed Date:** 01-12-23		
Module Name: Login			**Test Executed By:** Ms. Gloriya Mathew		
Test Title: Login Test			**Test Execution Date:** 01-12-23		
Description: Testing the Login Module					
Pre-Condition: User has valid username and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass/ Fail)**
1	Navigate to login page		login page should be displayed	Login page is displayed	Pass
2	Provide valid username	username: nithin	User should be able to login	User navigated to the index pages with the successful login	Pass
3	Provide valid password	Password: Manjadi@123			
4	Click on the Login button				
Post-Condition: User is logged in to the dashboard					

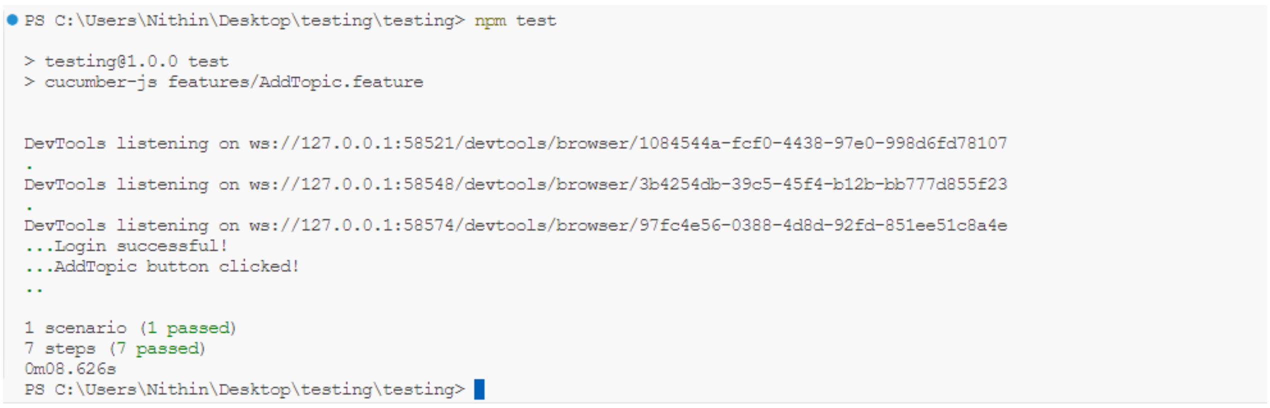
Test Case 2: Add a new Topic

Code:

```
const { Before, Given, When, Then } = require("cucumber");
const { Builder, By, until } = require("selenium-webdriver");
let driver;
Before(async function () {
   driver = await new Builder().forBrowser("chrome").build();
});
Given("I am on the admins login page", async function () {
   await driver.get("http://localhost:3000/Signin");
});
```

```
When("I enter valid admins credentials", async function () {
   await driver.findElement(By.name("username")).sendKeys("admin");
   await driver.findElement(By.name("password")).sendKeys("Admin@123");
});
When("click on the admins login button", async function () {
   try {
      const button = await driver.wait(
         until.elementLocated(By.id("testid")),
         10000
      );
      await button.click();
      console.log("Login successful!");
   } catch (error) {
      console.error("Error clicking on the login button:", error);
   }
});
Then("I should be redirected to the admins home page", async function () {
   await  driver.wait(until.urlIs("http://localhost:3000/AdminHome"));
   await driver.get("http://localhost:3000/AddTopic");
});
When("I enter Topic Details", async function () {
   await driver.findElement(By.name("testTopic")).sendKeys("test Topic");
   await driver.findElement(By.name("testContent")).sendKeys("test Content");
});
When("click on the AddTopic button", async function () {
   try {
      const button = await driver.wait(
         until.elementLocated(By.id("AddTopicButton")),
         10000
      );
      await button.click();
      console.log("AddTopic button clicked!");
   } catch (error) {
      console.error("Error clicking on the AddTopic button:", error);
   }
});
Then("I should be redirected to TopicList page", async function () {
   await driver.wait(until.urlIs("http://localhost:3000/TopicListAdmin"));
   await driver.quit();
});
```

Screenshot



```
● PS C:\Users\Nithin\Desktop\testing\testing> npm test

  > testing@1.0.0 test
  > cucumber-js features/AddTopic.feature


  DevTools listening on ws://127.0.0.1:58521/devtools/browser/1084544a-fcf0-4438-97e0-998d6fd78107
  .
  DevTools listening on ws://127.0.0.1:58548/devtools/browser/3b4254db-39c5-45f4-b12b-bb777d855f23
  .
  DevTools listening on ws://127.0.0.1:58574/devtools/browser/97fc4e56-0388-4d8d-92fd-851ee51c8a4e
  ...Login successful!
  ...AddTopic button clicked!
  ..

  1 scenario (1 passed)
  7 steps (7 passed)
  0m08.626s
  PS C:\Users\Nithin\Desktop\testing\testing>
```

Test Report

Test Case 2

Project Name: Formula One Fan Hub					
Add a new Topic					
Test Case ID: Test2			**Test Designed By:** Nithin Jose		
Test Priority (Low/Medium/High): High			**Test Designed Date:** 01-12-23		
Module Name: Add New Topic			**Test Executed By:** Ms. Gloriya Mathew		
Test Title: Add Topic			**Test Execution Date:** 01-12-23		
Description: Testing inserting a New Topic					
Pre-Condition: User has valid username and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass/ Fail)**
1	Navigate to login page		Login page should be displayed	Login page displayed	Pass
2	Provide valid username	Username: Admin	User should be able to login	User navigated to the index page with successful login	Pass
3	Provide valid password	Password: Admin@123			
4	Click on the login button				
5	Navigate to add Topic page		Add topic page should be displayed	User navigated to the add topic page	Pass
6	Provide valid topic data	Topic: test Topic	The topic data should be added in the database	User navigated to the topic List page with topic added message displayed	Pass
7	Provide valid content data	Content: test Content			
8	Click on the add topic button				
Post-Condition: After successful login, user can add a new topic					

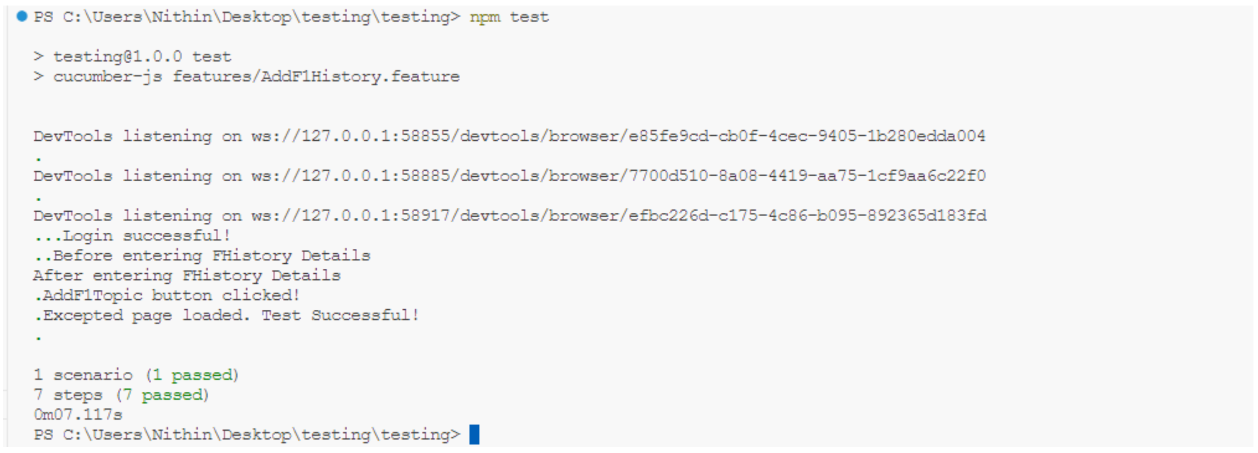
Test Case 3: Add new F1 history

Code:

```
const { Before, Given, When, Then } = require("cucumber");
const { Builder, By, until } = require("selenium-webdriver");
let driver;
Before(async function () {
   driver = await new Builder().forBrowser("chrome").build();
});

Given("I am on the admins login pageFOFH", async function () {
   await driver.get("http://localhost:3000/Signin");
});
When("I enter valid admins credentialsFOFH", async function () {
   await driver.findElement(By.name("username")).sendKeys("admin");
   await driver.findElement(By.name("password")).sendKeys("Admin@123");
});

When("click on the admins login buttonFOFH", async function () {
   try {
      const button = await driver.wait(
         until.elementLocated(By.id("testid")),
         10000
      );
      await button.click();
      console.log("Login successful!");
   } catch (error) {
      console.error("Error clicking on the login button:", error);
   }
});


Then("I should be redirected to the admins home pageFOFH", async function () {
   await driver.wait(until.urlIs("http://localhost:3000/AdminHome"));
   await driver.get("http://localhost:3000/AddF1History");
});
```

```
When("I enter FHistory Details", async function () {
   console.log("Before entering FHistory Details");
   try {
      const testHistoryHead = await
driver.wait(until.elementLocated(By.id("testHistoryHead")), 10000);
      await testHistoryHead.sendKeys("test Topic entered");
      // Add explicit wait for the element with name "testHistoryPara"
      const testHistoryPara = await
driver.wait(until.elementLocated(By.id("testHistoryPara")), 10000);
      // Clear existing text in case there is any
      await testHistoryPara.clear();
      // SendKeys after clearing
      await testHistoryPara.sendKeys("test Content entered");
      console.log("After entering FHistory Details");
   } catch (error) {
      console.error("Error entering FHistory Details:", error);
   }
});
When("click on the FHistory button", async function () {
   try {
      const button = await driver.wait(
         until.elementLocated(By.id("testFHistory")),
         10000
      );
      await button.click();
      console.log("AddTopic button clicked!");
   } catch (error) {
      console.error("Error clicking on the AddTopic button:", error);
   }
});
Then("I should be redirected to FHistoryList page", async function () {
   await driver.wait(until.urlIs("http://localhost:3000/F1HistoryList"));
   await driver.quit();
});
```

Screenshot



```
● PS C:\Users\Nithin\Desktop\testing\testing> npm test

> testing@1.0.0 test
> cucumber-js features/AddF1History.feature


DevTools listening on ws://127.0.0.1:58855/devtools/browser/e85fe9cd-cb0f-4cec-9405-1b280edda004
.
DevTools listening on ws://127.0.0.1:58885/devtools/browser/7700d510-8a08-4419-aa75-1cf9aa6c22f0
.
DevTools listening on ws://127.0.0.1:58917/devtools/browser/efbc226d-c175-4c86-b095-892365d183fd
...Login successful!
..Before entering FHistory Details
After entering FHistory Details
.AddF1Topic button clicked!
.Excepted page loaded. Test Successful!
.

1 scenario (1 passed)
7 steps (7 passed)
0m07.117s
PS C:\Users\Nithin\Desktop\testing\testing>
```

Test Report

Test Case 3

Project Name: Formula One Fan Hub					
Add New F1 History					
Test Case ID: Test3			**Test Designed By:** Nithin Jose		
Test Priority (Low/Medium/High): High			**Test Designed Date:** 01-12-23		
Module Name: Add New F1 History			**Test Executed By:** Ms. Gloriya Mathew		
Test Title: Add New F1History			**Test Execution Date:** 01-12-23		
Description: Testing adding a new F1 history data					
Pre-Condition: User has valid username and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass/ Fail)**
1	Navigate to login		Login Page should be displayed	Login page is displayed	Pass
2	Provide valid username	Username: Admin	User should be able to login	User navigated to the index page with successful login	Pass
3	Provide valid password	Password: Admin@123			
4	Click on the login button				
5	Navigate to add F1 history page		Add F1 history page should be displayed	User navigated to the add F1 history page	Pass
6	Provide valid heading data	Topic: test Topic entered	The F1 history data should be added in the database	User navigated to the F1 history List page with history data added message displayed	Pass
7	Provide valid paragraph data	Content: test Content			
8	Click on the add F1 history button				
Post-Condition: After successful login, user can add a new F1 history data					

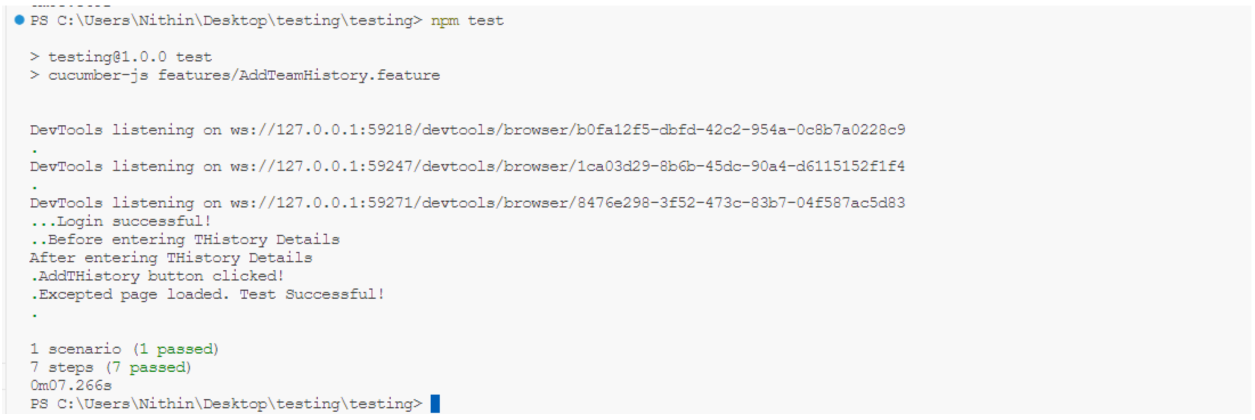
Test Case 4: Add new Team history

Code:

```
const { Before, Given, When, Then } = require("cucumber");
const { Builder, By, until } = require("selenium-webdriver");
let driver;
Before(async function () {
    driver = await new Builder().forBrowser("chrome").build();
});

Given("I am on the admins login pageFOFHT", async function () {
    await driver.get("http://localhost:3000/Signin");
});

When("I enter valid admins credentialsFOFHT", async function () {
    await driver.findElement(By.name("username")).sendKeys("admin");
    await driver.findElement(By.name("password")).sendKeys("Admin@123");
});

When("click on the admins login buttonFOFHT", async function () {
    try {
        const button = await driver.wait(
            until.elementLocated(By.id("testid")),
            10000
        );
        await button.click();
        console.log("Login successful!");
    } catch (error) {
        console.error("Error clicking on the login button:", error);
    }
});
Then("I should be redirected to the admins home pageFOFHT", async function () {
    await driver.wait(until.urlIs("http://localhost:3000/AdminHome"));
    await driver.get("http://localhost:3000/AddTeamHistory");
});
```

```
When("I enter THistory Details", async function () {
  console.log("Before entering THistory Details");
  try {
    const testHistoryHead = await
driver.wait(until.elementLocated(By.id("testTHistoryHead")), 10000);
    await testHistoryHead.sendKeys("test Team Topic entered");
    // Add explicit wait for the element with name "testHistoryPara"
    const testHistoryPara = await
driver.wait(until.elementLocated(By.id("testTHistoryPara")), 10000);
    await testHistoryPara.clear();
    // SendKeys after clearing
    await testHistoryPara.sendKeys("test Team Content entered");
    console.log("After entering THistory Details");
  } catch (error) {
    console.error("Error entering THistory Details:", error);
  }
});
When("click on the THistory button", async function () {
  try {
    const button = await driver.wait(
      until.elementLocated(By.id("testTHistory")),
      10000
    );
    await button.click();
    console.log("AddTHistory button clicked!");
  } catch (error) {
    console.error("Error clicking on the AddTHist button:", error);
  }
});
Then("I should be redirected to THistoryList page", async function () {
  await driver.wait(until.urlIs("http://localhost:3000/TeamHistoryList"));
  await driver.quit();
});
```

Screenshort



```
PS C:\Users\Nithin\Desktop\testing\testing> npm test

> testing@1.0.0 test
> cucumber-js features/AddTeamHistory.feature


DevTools listening on ws://127.0.0.1:59218/devtools/browser/b0fa12f5-dbfd-42c2-954a-0c8b7a0228c9
.
DevTools listening on ws://127.0.0.1:59247/devtools/browser/1ca03d29-8b6b-45dc-90a4-d6115152f1f4
.
DevTools listening on ws://127.0.0.1:59271/devtools/browser/8476e298-3f52-473c-83b7-04f587ac5d83
...Login successful!
..Before entering THistory Details
After entering THistory Details
.AddTHistory button clicked!
.Excepted page loaded. Test Successful!
.

1 scenario (1 passed)
7 steps (7 passed)
0m07.266s
PS C:\Users\Nithin\Desktop\testing\testing>
```

Test Report

Test Case 4

Project Name: Formula One Fan Hub					
Add New Team History					
Test Case ID: Test4			**Test Designed By:** Nithin Jose		
Test Priority (Low/Medium/High): High			**Test Designed Date:** 01-12-23		
Module Name: add new Team history			**Test Executed By:** Ms. Gloriya Mathew		
Test Title: Add New Team History			**Test Execution Date:** 01-12-23		
Description: Testing adding a new Team history					
Pre-Condition: User has valid username and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass/ Fail)**
1	Navigate to login		Login page displayed	Login page is displayed	Pass
2	Provide valid username	Username: Admin	User should be able to login	User navigated to the index page with successful login	Pass
3	Provide valid password	Password: Admin@123			
4	Click on the login button				
5	Navigate to add team history page		Add team history page should be displayed	User navigated to the add team history page	Pass
6	Provide valid heading data	Topic: test Team Topic entered	The team history data should be added in the database	User navigated to the team history List page with history data added message displayed	Pass
7	Provide valid paragraph data	Content: test Team Content entered			
8	Click on the add team history button				
Post-Condition: After successful login, user can add a new team history data					

CHAPTER 6
IMPLEMENTATION

6.1 INTRODUCTION

The project's implementation phase is when the idea becomes a working system. Testing a new system, is very important because it makes sure that the system will work properly and be dependable and users can trust it to be correct. The prioritized goals of the company are teaching and giving information to users. Usually, people learn how to use something first before they change it. Implementation means putting a new system design into action. It's the process of making the new system work, after it has been changed. The people in charge of using the system have the most work to do, face the most problems, and have the most say on how things
work right now. Implementation means the process of changing from the old system to the new one and includes all the steps involved in it. " The new system might be very different, replace a manual or computer system, or be changed to work even better. We need to set up a good system that works well for our organization. System implementation means using the system after it has been built. The people in charge check if the system will work or not. Developing a comprehensive education and training system necessitates the implementation of three crucial components,
namely, system testing, transitioning to the new system, and conducting an evaluative examination to ensure its efficacy. Implementation of the system may engender challenges the amplify the complexity and difficulty of executing such tasks.

6.2 IMPLEMENTATION PROCEDURES

Software implementation refers to the systematic procedure of installing and configuring a software program while ensuring its intended operation and functionality. Occasionally, individuals desire the creation of a computer program intended for their specific needs, albeit they may not be the primary end-users of said program. Initially, there may exist an apprehension amongst certain individuals with regards to the computer program. It is imperative to prevent the progression of uncertainty. The person who is using the new system needs to know the benefits it
has. The person using the app gets clear directions to help them feel comfortable using it. In order to utilize the system, it is imperative for the user to possess knowledge regarding the pre-requisite that the server software needs to be operational and functional, for the results to be visible. If the server instance fails to function, the corresponding operation cannot be executed.

6.1.1 User Training

The purpose of user training is to get the user ready to test and modify the system. The people who will be involved must have faith in their ability to contribute to the goal and benefits anticipated from the computer-based system. Training is more necessary as systems get more complicated. The user learns how to enter data, handle error warnings, query the database, call up routines that will generate reports, and execute other important tasks through user training

6.1.2 Training on the Application Software

The user will need to receive the essential basic training on computer awareness after which the new application software will need to be taught to them. This will explain the fundamental principles of how to use the new system, including how the screens work, what kind of help is displayed on them, what kinds of errors are made while entering data, how each entry is validated, and how to change the date that was entered. Then, while imparting the program's training on the application, it should cover the information required by the particular user or group to operate the system or a certain component of the system.

6.1.3 System Maintenance

Maintenance is the enigma of system development. The maintenance phase of the software cycle is the time in which a software product performs useful work. After a system is successfully implemented, it should be maintained in a proper manner. System maintenance is an important aspect in the software development life cycle. The need for system maintenance is for it to make adaptable to the changes in the system environment. Software maintenance is of course, far more than "Finding Mistakes".

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

In conclusion, the Formula One Fan Hub project stands as a dedicated platform designed to cater to the dynamic world of Formula 1 enthusiasts. With a focus on providing a rich and immersive experience, the system encompasses three pivotal roles: User, Guest, and Admin, each contributing to the overall functionality.

For Users, the platform offers an interactive dashboard, allowing seamless navigation through races, team histories, and personalized F1 and team-related content. The booking feature enhances the fan experience by providing a streamlined process for securing tickets, while participation in discussions, polls, and exploring historical data adds layers of engagement. Guests, although with limited access, can still explore race information, fostering inclusivity and encouraging potential users to join the vibrant community. The Admin role plays a crucial part in maintaining the platform's integrity. Admins have the capability to manage content, ensuring that race details, user interactions, and discussions are moderated effectively. The system also empowers Admins to oversee user activities and contribute to the longevity of the community.

As we look ahead, the Formula One Fan Hub project has exciting prospects. Future enhancements may include a mobile application for on-the-go engagement, advanced analytics to deliver personalized content, and potential collaborations with Formula 1 entities for exclusive insights. Global expansion, educational initiatives, and sustainability efforts align with the project's vision for long-term growth.

By fostering innovation, embracing user feedback, and staying attuned to the evolving landscape of Formula 1 fandom, the Formula One Fan Hub is poised to evolve into an indispensable hub for enthusiasts, maintaining its status as a premier online destination for all things Formula 1

7.2 FUTURE SCOPE

The Formula One Fan Hub project holds significant potential for future expansion and enhancement, geared towards creating an immersive and inclusive experience for Formula 1 enthusiasts. With a foundation built on distinct roles for Users, Guests, and Admin, the platform is poised to evolve in several key areas.

The future scope of the Formula One Fan Hub project includes the possibility of introducing advanced features such as IoT-based enhancements for an enriched race-watching experience, community-building through dedicated forums, and the development of a mobile application to ensure accessibility on various devices. Integrating cutting-edge technologies like AI and ML will enable personalized recommendations, fostering deeper engagement and interaction within the fan community.

Global expansion is a strategic avenue, aiming to reach a wider audience of Formula 1 enthusiasts around the world. Collaborations with Formula 1 entities for exclusive insights and partnerships with educational institutions can further elevate the platform's credibility and offer unique opportunities for learning and engagement.

Real-time API integration, live grid position, real-time score distribution are additional aspects that could be explored, adding layers of innovation and entertainment to the Formula One Fan Hub experience. By staying committed to continuous innovation and responsiveness to user feedback, the project is positioned to thrive and remain a dynamic hub for Formula 1 enthusiasts globally.

.

CHAPTER 8
BIBLIOGRAPHY

REFERENCES:

- ... Gary B. Shelly, Harry J. Rosenblatt, “System Analysis and Design”, 2009.
- ... Pankaj Jalote, “Software engineering: a precise approach”, 2006.
- ... Roger S Pressman, “Software Engineering”, 1994.
- ... Martin Fowler, “UML Distilled: A Brief Guide to the Standard Object Modeling Language”, 2003.
- ... Sommerville, Ian, "Software Engineering", 10th Edition, 2015.
- ... Kendall Scott, "Systems Analysis and Design: An Object-Oriented Approach with UML", 2007.

WEBSITES:

- ... https://www.formula1.com/
-
- ... www.geeksforgeeks.com
- ... www.agilemodeling.com/artifacts/useCaseDiagram.html
- ... https://www.racefans.net/
- …https://www.espn.in/f1/
- … https://learn.microsoft.com/en-us/dotnet/

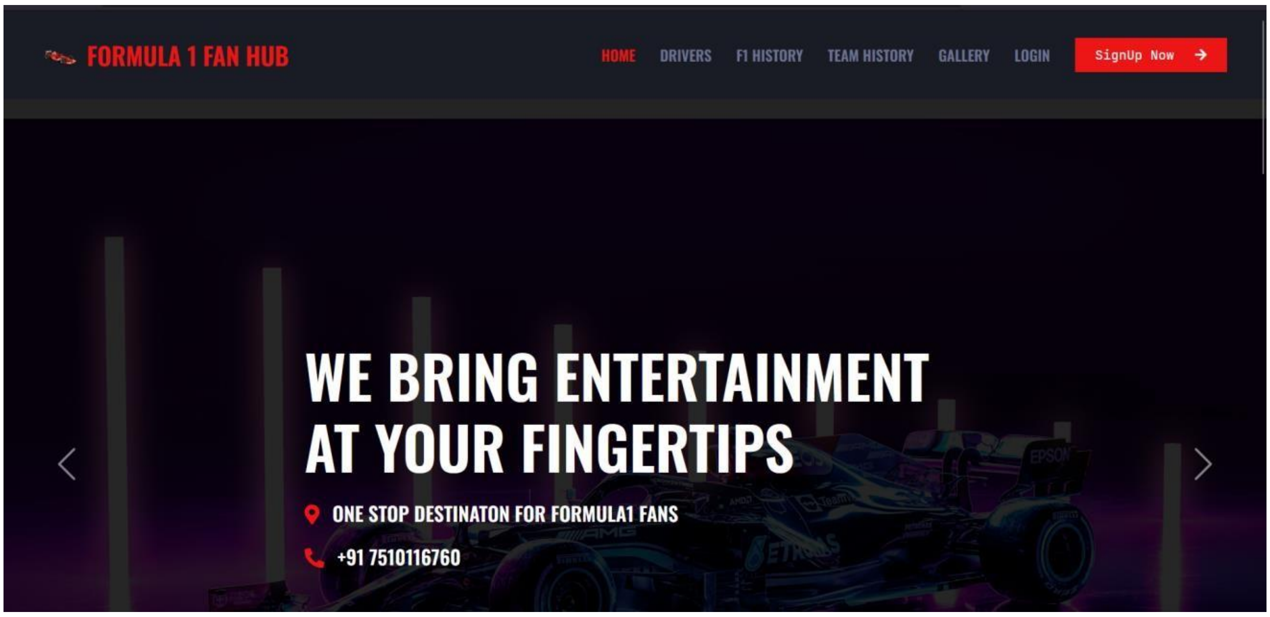
CHAPTER 9
APPENDIX

9.1 Sample Code

```
import React, { useEffect, useState } from 'react';
import { useNavigate } from 'react-router-dom';
import { toast } from 'react-toastify';
import UserNavbar from './UserNavbar';
import Footer from './Footer';
import jwt_decode from 'jwt-decode';
import { HomeCarousel } from './HomeCarousel';
import About from './About'; // assuming that About.jsx is in the same directory
import { Container, Typography, CssBaseline, Button, Box } from '@mui/material';

const AuthenticatedUserHome = () => {
  const navigate = useNavigate();
  const [userName, setUserName] = useState('');

  useEffect(() => {
    const token = localStorage.getItem('jwtToken');
    const decodedToken = jwt_decode(token);
    console.log('Decoded Token:', decodedToken);

    if (!token) {
      toast.error('Login and then access the Home page');
      navigate('/Signin');
    } else {
      const tokenPayload = jwt_decode(token);

      if (tokenPayload.RoleId === 'Admin') {
        navigate('/AdminHome');
      } else if (tokenPayload.RoleId === 'User') {
        setUserName(tokenPayload.userName);
      }
    }
  }, [navigate]);
  const handleLogout = () => {
    // Remove the JWT token from local storage
    localStorage.removeItem('jwtToken');
    toast.success('Logged out successfully');
    navigate('/Signin'); // Redirect to the login page
  };

  return (
    <Box>
      <CssBaseline />
      <UserNavbar />
      <Container maxWidth="xl">
        <Box mt={2}>
          <Typography variant="h4" align="center" gutterBottom>
            Hello {userName}
          </Typography>
          <Typography variant="h2" align="center" gutterBottom>
            Welcome Back to the Fan Hub
          </Typography>
        </Box>
      </Container>
      <HomeCarousel />
      <About />
      <Footer />
    </Box>
  );
};

export default AuthenticatedUserHome;
```

9.1Screen Shots:

home page:

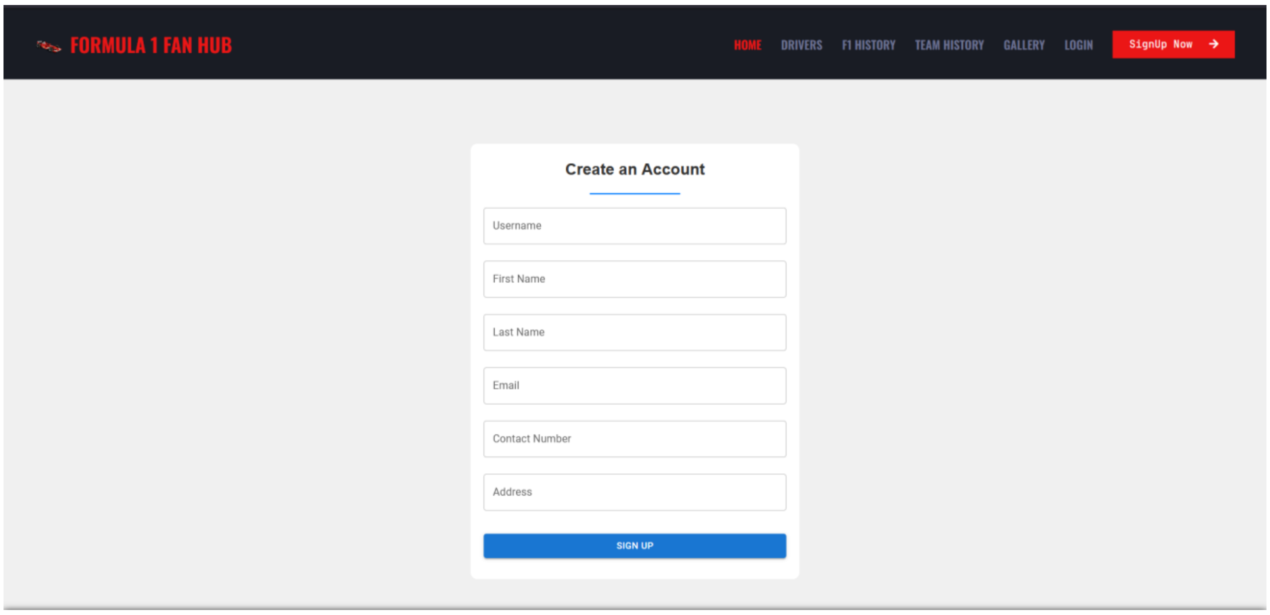


WE BRING ENTERTAINMENT AT YOUR FINGERTIPS

ONE STOP DESTINATON FOR FORMULA1 FANS

+91 7510116760

Registration Page:



Create an Account

Username

First Name

Last Name

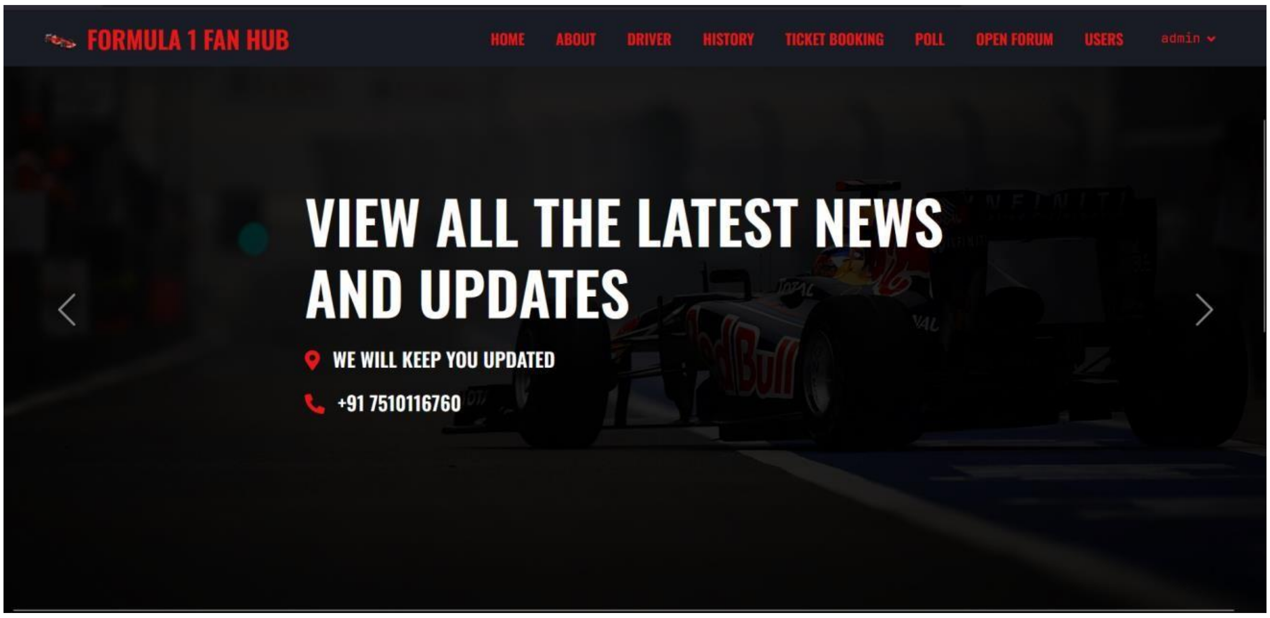
Email

Contact Number

Address

SIGN UP

Admin Home Page:

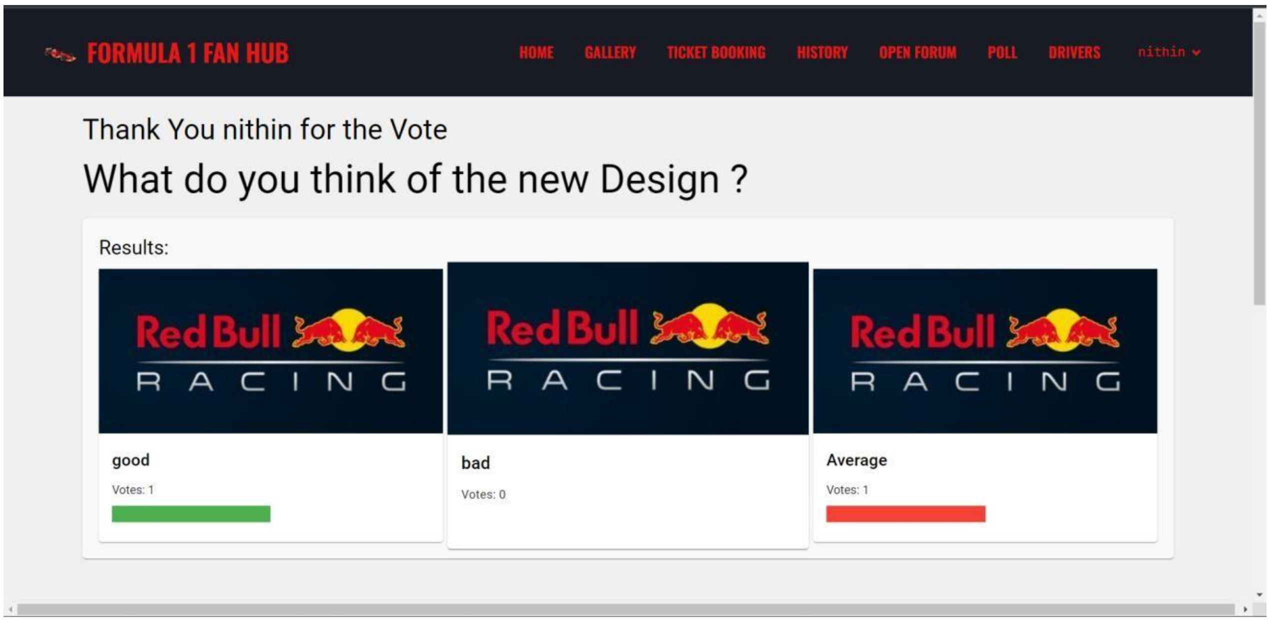


VIEW ALL THE LATEST NEWS AND UPDATES

WE WILL KEEP YOU UPDATED

+91 7510116760

User Polls:



Thank You nithin for the Vote

What do you think of the new Design ?

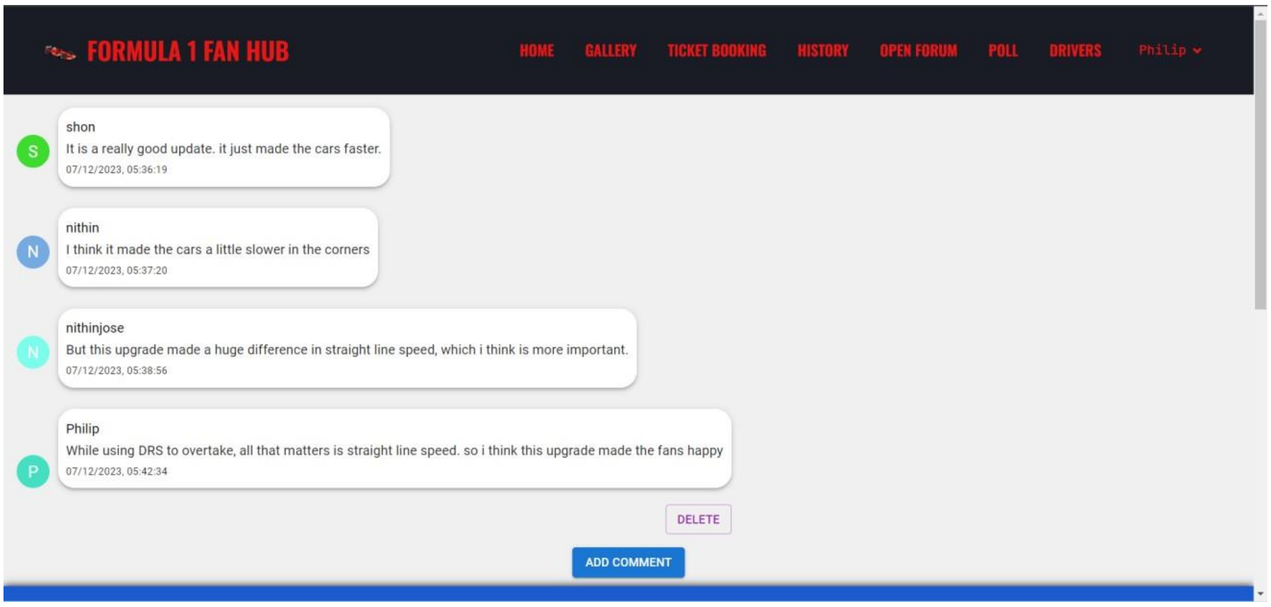
Results:

good
Votes: 1

bad
Votes: 0

Average
Votes: 1

Open Forums:



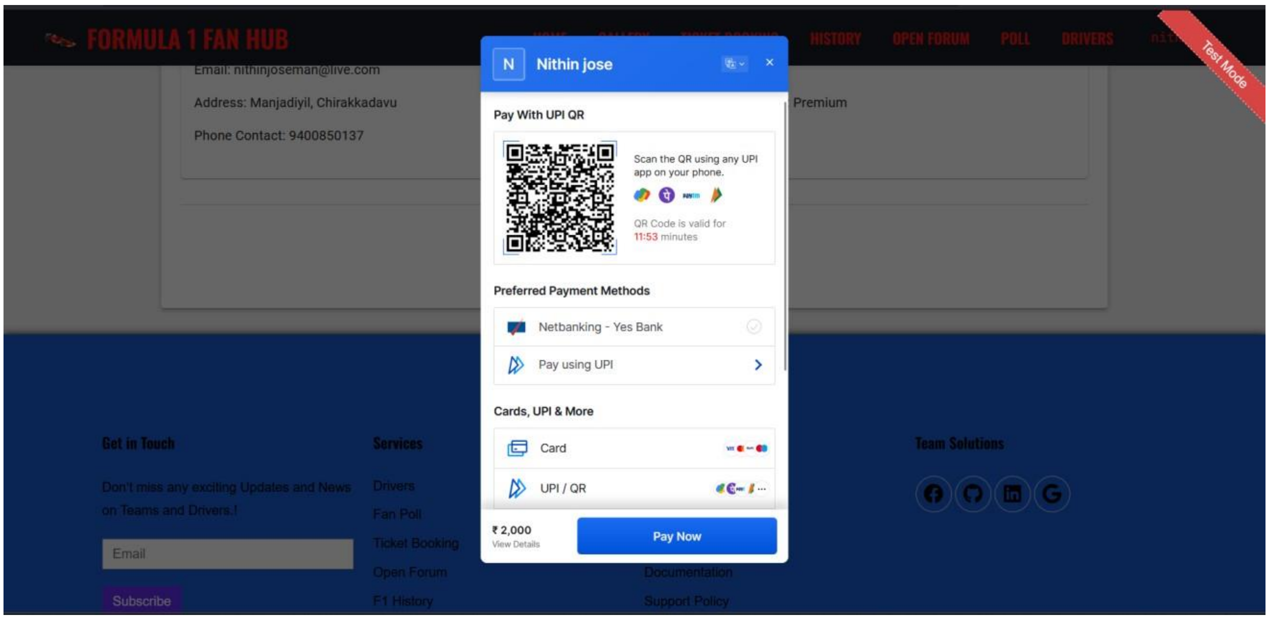
shon
It is a really good update. it just made the cars faster.
07/12/2023, 05:36:19

nithin
I think it made the cars a little slower in the corners
07/12/2023, 05:37:20

nithinjose
But this upgrade made a huge difference in straight line speed, which i think is more important.
07/12/2023, 05:38:56

Philip
While using DRS to overtake, all that matters is straight line speed. so i think this upgrade made the fans happy
07/12/2023, 05:42:34

Payment:



Email: nithinjoseman@live.com
Address: Manjadiyil, Chirakkadavu
Phone Contact: 9400850137
Premium

Nithin jose

Pay With UPI QR
Scan the QR using any UPI app on your phone.
QR Code is valid for 11:53 minutes

Preferred Payment Methods
Netbanking - Yes Bank
Pay using UPI

Cards, UPI & More
Card
UPI / QR

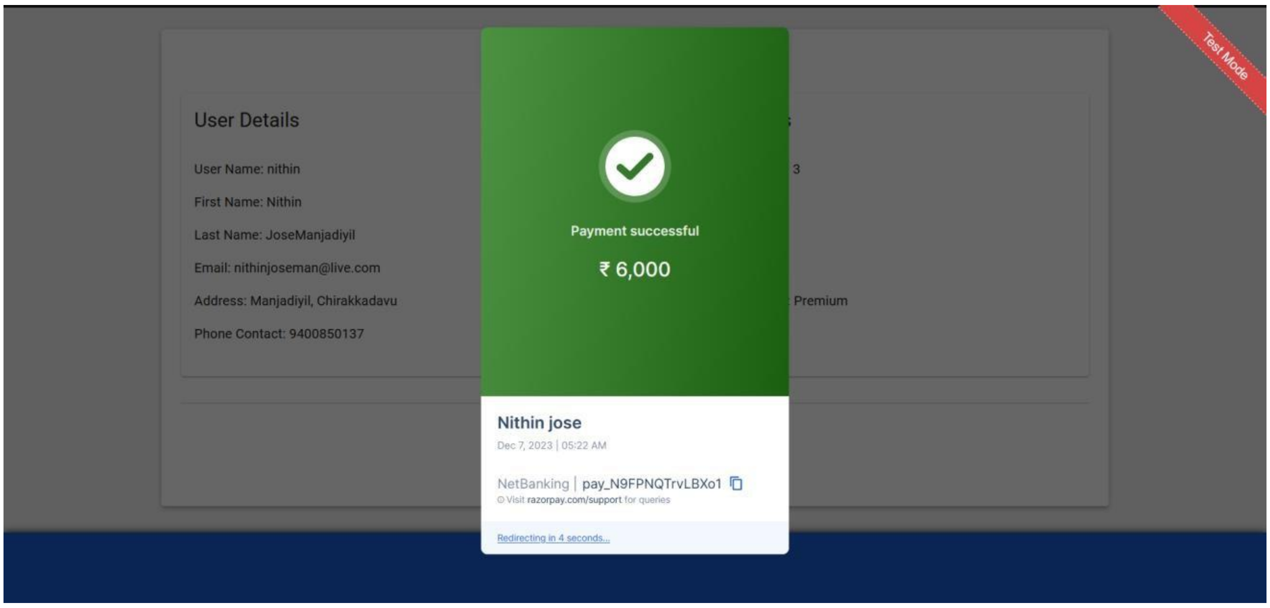
₹ 2,000
View Details

Get in Touch
Don't miss any exciting Updates and News on Teams and Drivers.!
Email
Subscribe

Services
Drivers
Fan Poll
Ticket Booking
Open Forum
F1 History

Documentation
Support Policy

Team Solutions



User Details

User Name: nithin

First Name: Nithin

Last Name: JoseManjadiyil

Email: nithinjoseman@live.com

Address: Manjadiyil, Chirakkadavu

Phone Contact: 9400850137

Payment successful

₹ 6,000

Nithin jose

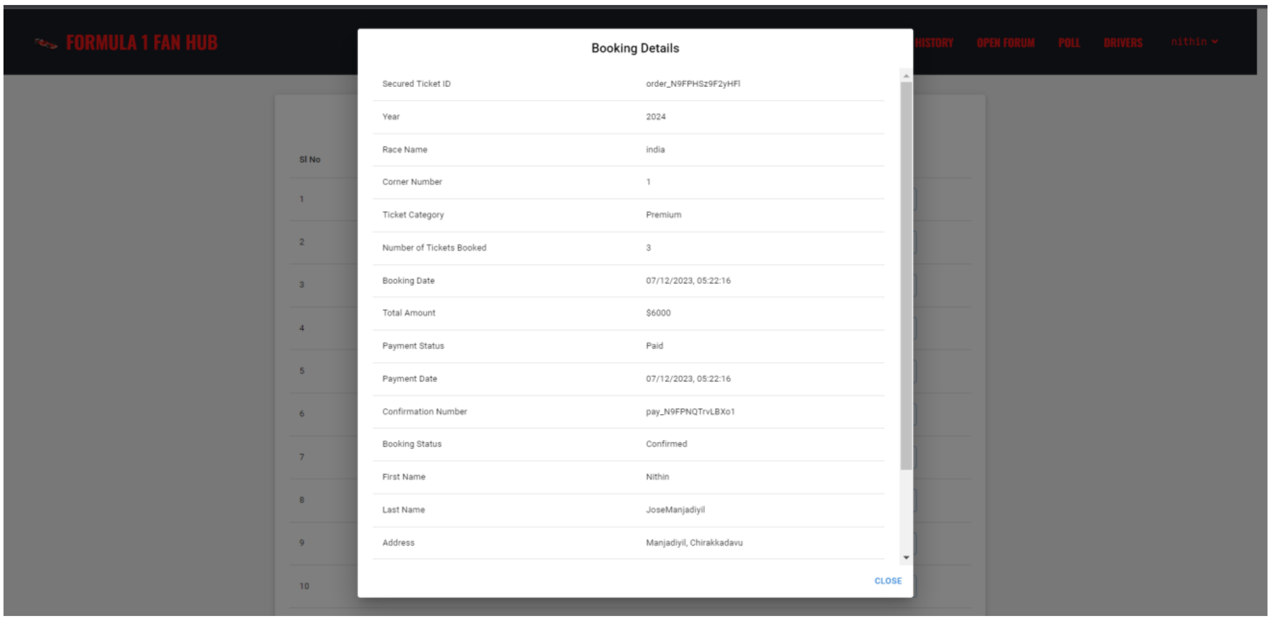
Dec 7, 2023 | 05:22 AM

NetBanking | pay_N9FPNQTrvLBXo1

Visit razorpay.com/support for queries

Redirecting in 4 seconds...

Ticket Details:



Booking Details

Secured Ticket ID	order_N9FPHSz9F2yHFl
Year	2024
Race Name	india
Corner Number	1
Ticket Category	Premium
Number of Tickets Booked	3
Booking Date	07/12/2023, 05:22:16
Total Amount	$6000
Payment Status	Paid
Payment Date	07/12/2023, 05:22:16
Confirmation Number	pay_N9FPNQTrvLBXo1
Booking Status	Confirmed
First Name	Nithin
Last Name	JoseManjadiyil
Address	Manjadiyil, Chirakkadavu