

FORMULA ONE FAN HUB

Formula One Fan Engagement System

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



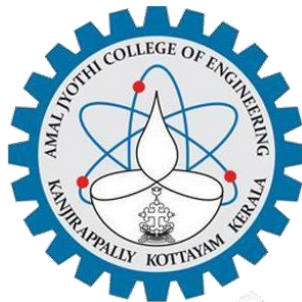
AMAL JYOTHI COLLEGE OF ENGINEERING

KANJIRAPPALLY

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

2022-2024

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

External Examiner

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 fulfillment 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: 01-05-2024

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 successfully. I wish to express my gratitude to our Director (Administration) **Rev. Fr. Dr. Roy Abraham Pazhayaparambil** and Principal **Dr. Lillykutty Jacob** for providing good faculty for guidance.

I extend my sincere gratitude to **Rev. Fr. Dr. Rubin Thottupurathu Jose**, our Head of the Department, for the significant support and assistance provided. I extend my whole hearted thanks to the project coordinator **Ms. Meera Rose Mathew** for her 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

In the dynamic world of Formula One, where speed, precision, and passion collide on the global stage, the "Formula One Fan Hub" emerges as a groundbreaking initiative poised to redefine fan engagement within the sport. This innovative platform represents a concerted effort to bridge the gap between fans, drivers, and teams, offering a comprehensive suite of features and functionalities designed to immerse enthusiasts in the rich tapestry of Formula One.

The Formula One Fan Hub is more than just a website; it is a digital ecosystem meticulously crafted to cater to the diverse needs and interests of Formula One aficionados worldwide. At its core lies a commitment to seamlessness and integration, seamlessly blending historical exploration, dynamic discussions, and convenient ticket booking into the fabric of individual team websites. The journey begins with the Home Page, a gateway to the world of Formula One, where fans can delve into the annals of Formula One history, explore team profiles, track driver standings, and stay abreast of the latest news and fixtures. This immersive experience sets the stage for deeper engagement, enticing fans to embark on a journey of discovery and connection.

User Functionality lies at the heart of the Formula One Fan Hub, empowering fans with the tools and resources they need to interact with the sport on their own terms. From seamless registration and login processes to streamlined ticket booking and interactive fan polls, every aspect of the user experience is meticulously designed to foster engagement and participation. The centrepiece of this functionality is the Open Forum, a vibrant community hub where fans, drivers, and teams converge to discuss the latest developments, share insights, and forge lasting connections. Meanwhile, Team functionalities empower teams to cultivate and nurture their fan communities, providing them with the tools they need to manage their profiles, create and moderate fan groups, and operate team stores seamlessly integrated within the platform. Admin functionalities serve as the backbone of the Formula One Fan Hub, providing administrators with the oversight and management tools they need to ensure the smooth operation of the platform. From user account management to content moderation, ticketing, and forum administration, every aspect of the platform is meticulously monitored and maintained to uphold the highest standards of quality and integrity. Looking to the future, the Formula One Fan Hub is committed to continuous improvement and innovation. The platform consistently evolves with cutting-edge technologies and advanced features to cater to the dynamic needs of Formula One fans.

CONTENT

SL. NO	TOPIC	PAGE NO
1	INTRODUCTION	1
1.1	PROJECT OVERVIEW	2
1.2	PROJECT SPECIFICATION	2
2	SYSTEM STUDY	5
2.1	INTRODUCTION	6
2.2	EXISTING SYSTEM	6
2.2.1	NATURAL SYSTEM STUDIED	6
2.2.2	DESIGNED SYSTEM STUDIED	7
2.3	DRAWBACKS OF EXISTING SYSTEM	7
2.4	PROPOSED SYSTEM	8
2.5	ADVANTAGES OF PROPOSED SYSTEM	8
3	REQUIREMENT ANALYSIS	9
3.1	FEASIBILITY STUDY	10
3.1.1	ECONOMICAL FEASIBILITY	11
3.1.2	TECHNICAL FEASIBILITY	11
3.1.3	BEHAVIORAL FEASIBILITY	11
3.1.4	FEASIBILITY STUDY QUESTIONNAIRE	12
3.2	SYSTEM SPECIFICATION	15
3.2.1	HARDWARE SPECIFICATION	15
3.2.2	SOFTWARE SPECIFICATION	15
3.3	SOFTWARE DESCRIPTION	15
3.3.1	ASP .NET CORE	15
3.3.2	REACT JS	15
3.3.3	MS SQL SERVER	15
4	SYSTEM DESIGN	17
4.1	INTRODUCTION	18
4.2	UML DIAGRAM	18
4.2.1	USE CASE DIAGRAM	19
4.2.2	SEQUENCE DIAGRAM	20
4.2.3	STATE CHART DIAGRAM	22

4.2.4	ACTIVITY DIAGRAM	23
4.2.5	CLASS DIAGRAM	24
4.2.6	OBJECT DIAGRAM	25
4.2.7	COMPONENT DIAGRAM	26
4.2.8	DEPLOYMENT DIAGRAM	27
4.3	USER INTERFACE DESIGN USING FIGMA	28
4.4	DATABASE DESIGN	30
4.5	TABLE DESIGN	33
5	SYSTEM TESTING	45
5.1	INTRODUCTION	46
5.2	TEST PLAN	46
5.2.1	UNIT TESTING	46
5.2.2	INTEGRATION TESTING	47
5.2.3	VALIDATION TESTING	47
5.2.4	USER ACCEPTANCE TESTING	48
5.2.5	AUTOMATION TESTING	48
5.2.6	SELENIUM TESTING	49
6	IMPLEMENTATION	67
6.1	INTRODUCTION	68
6.2	IMPLEMENTATION PROCEDURE	68
6.2.1	USER TRAINING	68
6.2.2	TRAINING ON THE WEBSITE	69
6.2.3	SYSTEM MAINTENANCE	69
6.2.4	HOSTING	69
7	CONCLUSION & FUTURE SCOPE	71
7.1	CONCLUSION	72
7.2	FUTURE SCOPE	72
8	BIBLIOGRAPHY	73
9	APPENDIX	75
9.1	SAMPLE CODE	76
9.2	SCREENSHOTS	78

List of Abbreviation

IDE	Integrated Development Environment
HTML	Hyper Text Markup Language.
CSS	Cascading Style Sheet
RDBMS	Relational Database Management System
UML	Unified Modeling Language
JS	Java Script
FK	Foreign key
PK	Primary key
F1	Formula One
ASP	Active Server Pages

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

The "Formula One Fan Hub" is an innovative digital platform revolutionizing fan engagement within the Formula One community. Dedicated to exploring, engaging, and elevating the Formula One experience, our platform seamlessly integrates historical exploration, dynamic discussions, and convenient race ticket booking into individual team websites. With a commitment to fairness, our innovative waiting list and ticket confirmation system ensure equal access to race tickets for all fans. Our interactive "Open Forum" facilitates lively interaction among fans, drivers, and teams, fostering a vibrant community atmosphere. Utilizing cutting-edge technologies such as React and .NET, our platform offers real-time data updates, enhancing the overall user experience. Addressing conventional limitations, our platform ingeniously integrates ticket booking into individual team websites, streamlining the process for fans. Moreover, we introduce an integrated store feature within the hub, allowing fans to purchase team merchandise directly. Teams can create and manage their fan groups, propelling a new era of fan-team connectivity within the Formula One landscape. With features like timely updates, interactive discussions, and advanced technologies, the "Formula One Fan Hub" redefines fan engagement and establishes itself as the ultimate destination for Formula One enthusiasts worldwide.

1.2 PROJECT SPECIFICATION

The "Formula One Fan Hub" revolutionizes fan engagement, offering a comprehensive platform for fans, teams, and administrators. With tailored modules, it enhances the Formula One experience for all users. From historical exploration to dynamic discussions and convenient ticket booking, it seamlessly integrates key functionalities. Admins oversee user management, content curation, and ticket operations, while fans enjoy streamlined registration, ticket booking, and open forum participation. Teams manage profiles, engage with fans, and run their stores, fostering a deeper connection. Through advanced features and user-centric design, the hub sets a new standard for Formula One interaction.

Admin Functionality:

- **User Management:** Admins have comprehensive control over user accounts, enabling them to oversee account verification and suspension processes. They ensure platform integrity by managing user permissions and addressing any issues promptly. Admins play a pivotal role in maintaining the security and functionality of the platform, implementing robust user management strategies to safeguard sensitive data and uphold user trust.

- **Content Management:** Admins are responsible for curating and updating various content sections, including Formula One history, team profiles, driver standings, news, and fixtures. They ensure the accuracy and relevance of information to provide users with up-to-date and engaging content.
- **Ticket Management:** Admins handle all aspects of ticket management, from setting ticket availability to managing bookings and waiting lists. They also maintain a detailed record of historical ticket sales data, enabling them to optimize fan access and enhance the overall ticket booking experience.
- **Open Forums:** Admins play a crucial role in overseeing the creation, moderation, and management of open forums. They ensure that discussions remain constructive and respectful, fostering a vibrant community where fans, drivers, and teams can interact and exchange ideas.

User Functionality:

- **Registration and Profile Management:** Fans have the convenience of registering and managing their profiles within the platform. They can set preferences, update personal information, and customize their Formula One experience according to their interests and preferences.
- **Ticket Booking:** Fans enjoy a seamless ticket booking experience, with the ability to browse, select, and book race tickets effortlessly. Features like waiting lists and ticket confirmation ensure fair access to tickets, enhancing user satisfaction and engagement.
- **Open Forum Participation:** Fans actively engage in discussions through the open forum feature, sharing insights, opinions, and experiences with fellow enthusiasts. The platform serves as a hub for meaningful interactions within the Formula One community, fostering camaraderie and collaboration among fans.

Team Functionality:

- **Team Profile Management:** Teams have full control over their profiles, allowing them to showcase team information, achievements, and merchandise available in the team store. They can update content regularly to keep fans informed and engaged.
- **Open Forum Management:** Teams take an active role in moderating and participating in open forums, directly engaging with fans and addressing inquiries, feedback, and discussions. This interaction fosters a stronger connection between teams and their fan base, promoting loyalty and support.
- **Team Store:** Teams manage their merchandise inventory and storefront within the platform, providing fans with a convenient shopping experience. Fans can browse products, make purchases, and checkout seamlessly, further enhancing their engagement with their favourite teams.

Technologies Used:

Frontend	: React JS, CSS, HTML
Backend	: ASP .NET Core
Database	: SQL Server
Payment Gateway Integration	: Razorpay

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

In an era characterized by technological advancements and evolving fan dynamics within the motorsport landscape, the "Formula One Fan Hub" emerges as a pivotal response to the changing paradigms of fan engagement. Positioned as an innovative digital platform, this project aims to redefine the interaction between Formula One enthusiasts, teams, and drivers. By harnessing the power of digital connectivity, the Formula One Fan Hub endeavors to revolutionize the fan experience, offering a comprehensive array of features to enhance engagement and connectivity within the Formula One community.

2.2 EXISTING SYSTEM

The established Formula One Fan Hub represents a pioneering initiative aimed at enhancing fan interaction and team connectivity within the Formula One ecosystem. Admins oversee the platform's functionality, ensuring seamless integration of historical exploration, dynamic discussions, and ticket booking services. The hub serves as a centralized platform for fans to explore Formula One history, access team information, and engage in real-time discussions with fellow enthusiasts, drivers, and teams. Through advanced features and an interactive "Open Forum," the Formula One Hub prioritizes user engagement and satisfaction. Security measures, including user authentication, safeguard sensitive data and ensure a secure browsing experience. By offering a user-friendly interface and personalized features such as fan polls and team merchandise stores, the Formula One Hub fosters a sense of community and loyalty among Formula One fans worldwide.

2.2.1 NATURAL SYSTEM STUDIED

The Formula One Fan Hub operates as a natural system, embodying a dynamic ecosystem where technological innovations, fan behaviors, and sporting trends converge seamlessly. At its core, the technological infrastructure, consisting of web servers and databases, serves as the backbone facilitating efficient fan engagement and team interactions. Fans, as active participants in this ecosystem, wield significant influence through their interactions and preferences, shaping the platform's response to the evolving dynamics of Formula One and motorsport as a whole. This holistic approach ensures a symbiotic relationship between cutting-edge technology, passionate fan engagement, and the ever-evolving landscape of Formula One.

2.2.2 DESIGNED SYSTEM STUDIED

The Formula One Fan Hub represents an intricately designed platform engineered to prioritize immersive fan experiences and seamless interactions for all stakeholders involved. Teams and administrators benefit from streamlined content management, intuitive navigation across historical data, and real-time updates on race fixtures and driver standings. Comprehensive insights into fan demographics contribute to strategic decision-making and targeted engagement efforts. The system ensures secure access through robust authentication mechanisms, safeguarding user data and interactions. Fans experience a personalized journey with features such as race ticket booking, interactive forums for lively discussions, and access to exclusive team merchandise. Advanced technologies like this further enhance the user experience, promising a vibrant online community that fosters deeper connections between fans, drivers, and Formula One teams.

2.3 DRAWBACKS OF EXISTING SYSTEM

- The current Formula One fan engagement platforms may lack sophisticated search and filtering capabilities, making it difficult for users to efficiently find specific historical information, team updates, or driver-related content.
- Scalability challenges might arise as the platform expands, potentially causing performance issues and slower response times. This could hinder the seamless exploration and interaction experience desired by users.
- Mobile optimization may be inadequate, disregarding the increasing trend of users accessing platforms through smartphones and tablets.
- Limited Interactivity: The existing systems may lack sufficient interactivity, hindering dynamic discussions and real-time engagement among fans, drivers, and teams.
- Fragmented Experience: Users may face a disjointed experience due to the absence of a unified platform integrating ticket booking, historical exploration, and community interaction seamlessly.
- Insufficient Personalization: The current systems may not offer personalized recommendations or tailored content based on user preferences and interests, limiting user engagement and satisfaction.

2.4 PROPOSED SYSTEM

The proposed system, "Formula One Fan Hub," revolutionizes fan engagement within the Formula One landscape by offering a comprehensive platform that amalgamates historical exploration, dynamic discussions, and seamless ticket booking functionalities. Unlike existing systems that often segregate these features across various platforms, our system integrates them into individual team websites, providing a one-stop destination for fans. Through a user-friendly interface, fans can delve into Formula One history, team profiles, driver standings, and latest news, fostering a deeper connection with the sport they love.

Moreover, the proposed system addresses key limitations in current systems by introducing unique solutions. By incorporating a waiting list and ticket confirmation system, it ensures fair access to race tickets, mitigating frustrations often encountered in ticket booking processes. Additionally, the integration of a team store feature directly within the platform enables fans to purchase merchandise seamlessly, enhancing their overall experience. The interactive "Open Forum" facilitates real-time interactions among fans, drivers, and teams, fostering a vibrant community and elevating fan engagement to new heights.

Furthermore, leveraging cutting-edge technologies like React and .NET, the system promises real-time data updates and innovative features such as face detection for driver identification. With continuous improvements such as notifications and responsive design, the system aims to provide an informative and adaptable user experience, ensuring it remains the ultimate destination for Formula One enthusiasts. By seamlessly merging historical exploration, dynamic discussions, and convenient ticket booking, the "Formula One Fan Hub" is poised to redefine fan engagement and become an indispensable tool for fans, teams, and drivers alike.

2.5 ADVANTAGES OF PROPOSED SYSTEM

- The Formula One Fan Hub revolutionizes fan engagement by offering a multifaceted platform that combines historical exploration, dynamic discussions, and convenient ticket booking, catering to the diverse interests and needs of Formula One enthusiasts.
- With a user-centric design, the Formula One Fan Hub prioritizes a seamless and efficient user experience, allowing fans to easily navigate through the platform, access information, and interact with fellow enthusiasts and teams effortlessly.
- Emphasizing the security of transactions, the hub provides robust payment options, instilling confidence in users regarding the safety and reliability of financial transactions, thereby facilitating hassle-free ticket bookings and merchandise purchases.

- The Formula One Fan Hub fosters inclusivity by catering to a broad spectrum of Formula One fans, including those interested in historical insights, current events, and race attendance. Its user-friendly interface and diverse features accommodate fans with varying levels of engagement and preferences.
- Simplifying the fan experience, the hub offers intuitive search and browsing functionalities, enabling fans to easily discover relevant content, engage in discussions, and access race-related information, enhancing overall satisfaction and enjoyment.

The Formula One Fan Hub offers a comprehensive and user-centric platform that enhances fan engagement, promotes inclusivity, and facilitates seamless interactions within the Formula One community.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

Ensuring the feasibility of the Formula One Fan Hub is vital to its successful implementation. A thorough feasibility study is conducted to assess its viability, considering factors such as resource availability, development costs, and potential business advantages. The study aims to determine if the project aligns with the organization's goals and if it should proceed to the requirements engineering and system development phases.

3.1.1 ECONOMIC FEASIBILITY

Evaluating the economic feasibility of the Formula One Fan Hub is crucial to understand its financial viability and potential profitability. Conducting a comprehensive cost-benefit analysis is essential, comparing the development, operational, and maintenance expenses against expected benefits and revenue streams. Initial development costs, ongoing operational expenses, and potential revenue from ticket sales and merchandise should be estimated to calculate projected return on investment (ROI) and break-even points. Additionally, market research should be conducted to gauge demand, competition, and growth potential within the Formula One fan community. By conducting a thorough economic feasibility analysis, informed decisions can be made to ensure the long-term financial success of the project.

3.1.2 TECHNICAL FEASIBILITY

Ensuring technical feasibility is paramount for the success of the Formula One Fan Hub. Firstly, assess the organization's technology infrastructure, including web hosting, databases, and server capacity, to support the platform's requirements. Skilled software developers are essential for building and maintaining the system efficiently. Seamless integration with third-party services like payment gateways and data providers is necessary. The platform's architecture should be scalable to accommodate increasing traffic and data as it grows. Robust security measures must be in place to protect user data and transactions. Mobile-friendliness and responsive performance across various devices are imperative. Compliance with data protection regulations is non-negotiable. Lastly, evaluate the financial resources needed for development, hosting, maintenance, and support.

3.1.3 BEHAVIORAL FEASIBILITY

Assessing behavioral feasibility is crucial to determine the integration of the Formula One Fan Hub into existing operations. Ensure the availability of necessary resources, including human capital, technology, and infrastructure, within the project's time frame and budget. Evaluate the

skill set of existing staff and identify any need for additional training or hiring. Consider how well the new system will integrate with current processes, such as managing fan engagement and team communications. Plan for change management strategies to address potential resistance to organizational changes. Ensure third-party services or suppliers can support operational requirements effectively. Compliance with legal and regulatory aspects, including those related to fan engagement and data protection, must be verified. Gather user feedback to gauge acceptance and identify areas for improvement. Analyze operational costs to ensure alignment with the organization's budget.

3.1.4 FEASIBILITY STUDY QUESTIONNAIRE

1. Project Overview?

The “Formula One Fan Hub” project aims to create an innovative platform that enhances the fan experience in the world of Formula One. This platform will allow fans to explore the sport's history, engage in discussions, and seamlessly book race tickets. The integration of ticket booking into individual team websites sets this project apart, making it easier for enthusiasts to connect with the Formula One world.

2. To what extend the system is proposed for?

The Formula One Fan Hub aims to provide a comprehensive solution for Formula One fans. It covers aspects such as exploring Formula One history, engaging in discussions, and booking race tickets. The system is designed to provide to the needs of fans, drivers, and teams, offering a cohesive and interactive experience.

3. Specify the Viewers/Public which is to be involved in the System?

The Formula One Fan Hub will involve the following categories of viewers and users:

- Registered Formula One Fans
- Administrators/Content Managers
- Team Management
- Delivery Company

4. List the Modules included in your System?

Here is a concise list of modules included in The Formula One Fan Hub:

1. User Registration and Login: The system should allow users to register and log in securely, providing access to personalized features and content.
2. Team Profile and Management: Each Formula One team should have a dedicated profile section where users can explore team history, achievements, and updates.
3. Ticket Booking and Management: The platform should facilitate easy booking and management of race tickets, with features like cancellation and ticket confirmation systems.

4. Discussion Forums: An interactive "Open Forum" should be implemented to encourage real-time discussions among fans, drivers, and teams, fostering a vibrant community atmosphere.
5. E-commerce Integration: A team store feature should allow fans to purchase team merchandise directly from the platform, enhancing fan engagement and revenue generation opportunities.
6. Admin Panel: An administrative dashboard is essential for managing content, users, ticketing, forums, and other aspects of the platform efficiently.
7. Responsive Design: Ensure the platform is accessible and user-friendly across various devices and screen sizes, providing a seamless experience for fans.

5. Identify the users in your project?

Fans: These are the primary users of the platform. They use the hub to explore Formula One history, engage in discussions on the Open Forum, purchase race tickets, browse team merchandise, and interact with drivers and teams.

Teams: Formula One teams utilize the hub to manage their profiles, share team information, achievements, and merchandise in the Team Store. They also participate in the Open Forum, interacting with fans and managing discussions relevant to their team.

Admins: Administrators play a crucial role in managing the platform. They oversee content management, ticket management, user management, and forum moderation. Admins ensure the smooth functioning of the hub and address any issues that arise.

6. Who owns the system?

This is a sole idea of Nithin Jose, young talented student of Amal Jyothi College of Engineering. He has the ownership of its idea and its web application.

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

The Formula One Fan Hub is related to the Formula One industry, specifically catering to fans, drivers, and Formula One teams. It aims to provide a unique and engaging experience within this sporting domain.

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

- a. Mostly Online Sources
- b. Sarosh Hataria, Ahura Racing Academy, Coimbatore
- c. 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 Formula One Fan Hub project?

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

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

It includes exploring Formula One 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. 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 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 Formula One 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 Formula One Fan Hub 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 or above
RAM	-	4GB
Hard disk	-	256GB SSD

3.2.2 Software Specification

Front End	-	React JS, HTML5, CSS, Bootstrap, React JS
Back End	-	ASP .NET Core
Database	-	SQL Server
Client on PC	-	Windows 7 and above.

3.3 SOFTWARE DESCRIPTION

3.3.1 Asp .Net Core

ASP.NET Core serves as the backbone of the Formula One Fan Hub, offering a robust and efficient framework for web development. Leveraging the power of C# and the ASP.NET Core MVC architecture, this platform delivers high-performance, scalable, and secure solutions. With features like built-in dependency injection and middleware support, ASP.NET Core streamlines development and ensures code maintainability. Its cross-platform compatibility allows seamless deployment across various hosting environments, while its integration with Entity Framework Core simplifies database operations.

3.3.2 React JS

ReactJS is a JavaScript library for developing UIs, at a declarative level. Facebook is the company behind this. It lets the programmers use, create, and display the UI elements in an interactive and dynamic way much easier. Integrated component- first system in React allows for the building of reusable user interface components while simultaneously improving code structure and maintenance. React offers a better way to optimize performance to the user through its virtual DOM and one-way data binding. React is getting more popular as a framework of choice for building up modern designs, usable on all devices, and scalable web applications.

3.3.3 MS SQL Server

SQL server serves as the database management system for the Formula One Fan Hub, providing a reliable and scalable solution for data storage and retrieval. With its robust transactional

capabilities and support for ACID properties, SQL Server ensures data integrity and consistency. The platform's advanced query optimization and indexing features optimize performance, even with large datasets. SQL Server's comprehensive security features safeguard sensitive information, while its administrative tools simplify database management tasks. Its seamless integration with ASP.NET Core via Entity Framework Core enables efficient data access and manipulation, empowering the Formula One Connection Hub to deliver a seamless user experience.

CHAPTER 4

SYSTEM DESIGN

4.1 INTRODUCTION

It is a critical step in process of application development, working a pivotal role in molding the architecture and functionality. It encompasses the creation of a structured blueprint that outlines everything within the application will interact and collaborate to meet specific objectives. This process involves making crucial decisions regarding database architecture, server configuration, and overall system architecture to ensure scalability, efficiency, and robustness. Additionally, system design takes into consideration factors such as user experience and data management to create a mature and effective application. It requires understanding of the application's requirements and proficiency in various technologies and programming paradigms to construct a robust foundation for the development process.

4.2 UML DIAGRAM

It stands as a foundational tool in software engineering, renowned for its role in visually representing complex systems and processes. It provides a standardized set of graphical notations that facilitate the clear depiction of diverse parts of a system's structure. Originating from the collaboration of industry experts, UML has gained widespread acceptance and adoption in both academia and industry. It serves as a powerful communication tool, enabling stakeholders, including developers, designers, and clients, to attain a shared understanding of system architecture, design, and functionality. UML diagrams transcend language barriers and ensuring a consistent means of conveying intricate software concepts, ultimately enhancing the performance of the development process.

- Use case diagram
- Sequence diagram
- State diagram
- Activity diagram
- Class diagram
- Object diagram
- Component diagram
- Deployment diagram

4.2.1 USE CASE DIAGRAM

Use Case Diagrams, a cornerstone in software engineering, serve as a look into of the interactions between a system and other entities. At their core, they provide a structured means of identifying and defining the various functionalities a system offers and how these functionalities are accessed by different actors or entities. Actors, representing users, systems, or external entities, are depicted along with the specific use cases they engage with. Associations between actors and use cases elucidate the nature of these interactions, clarifying the roles and responsibilities of each entity within the system. This detailed visual representation not only enhances communication among stakeholders but also provides a clear blueprint for system functionality, laying the foundation for the subsequent stages of the development process. Overall, UseCase Diagrams play a pivotal role in aligning development efforts with user expectations, ensuring that the resulting software system fulfills its intended purpose effectively and efficiently.

- **Actor Definition:** Clearly define and label all actors involved in the system. Actors are the external entities interacting with the system.
- **Use Case Naming:** Use descriptive names for use cases to accurately convey the functionality they represent.
- **Association Lines:** Use solid lines to represent associations between actors and use cases. This signifies the interaction between entities.
- **System Boundary:** Draw a box around the system to indicate its scope and boundaries. This defines what is inside the system and what is outside.
- **Include and Extend Relationships:** Use "include" relationships to represent common functionalities shared among multiple use cases. Use "extend" relationships to show optional or extended functionalities.

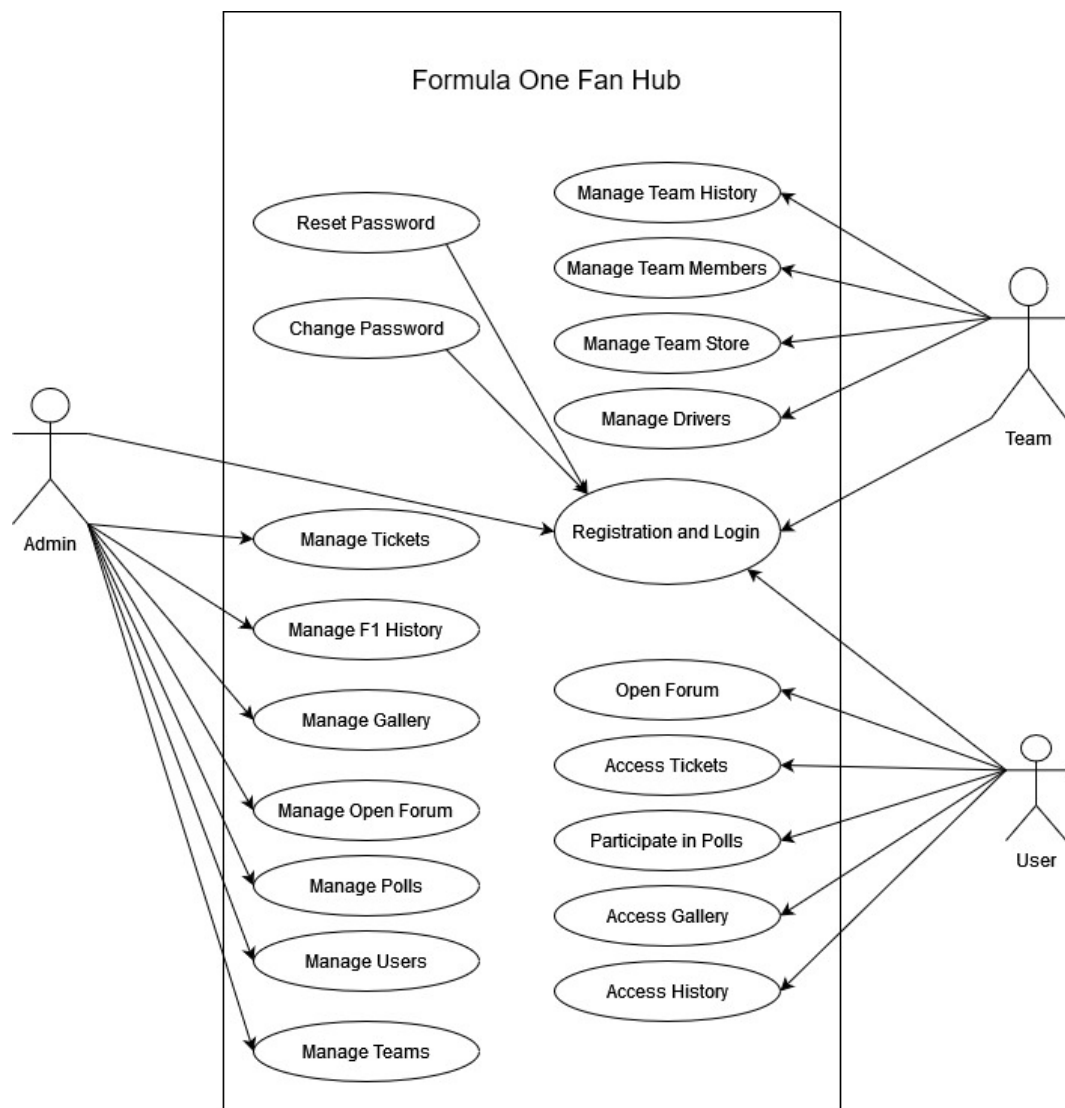


Figure 1: Use Case diagram

4.2.2 SEQUENCE DIAGRAM

Sequence Diagrams stand as dynamic models in software engineering, portraying the chronological flow of interactions between or components within a system. They spotlight the order in which messages are exchanged, revealing the behaviour of the system over time. Actors and objects are represented along a vertical axis, with arrows indicating the sequence of messages and their direction. Lifelines, extending vertically from actors or objects, illustrate their existence over the duration of the interaction. These diagrams serve as vital tool for visualizing system behaviour, understanding temporal aspects of the process. Through Sequence Diagrams, stakeholders gain valuable insights into how different elements collaborate to achieve specific functionalities, facilitating more effective communication among development teams and stakeholders alike. This detailed representation not only aids in detecting potential bottlenecks or inefficiencies but also provides a foundation for refining system performance in the later stages of software development.

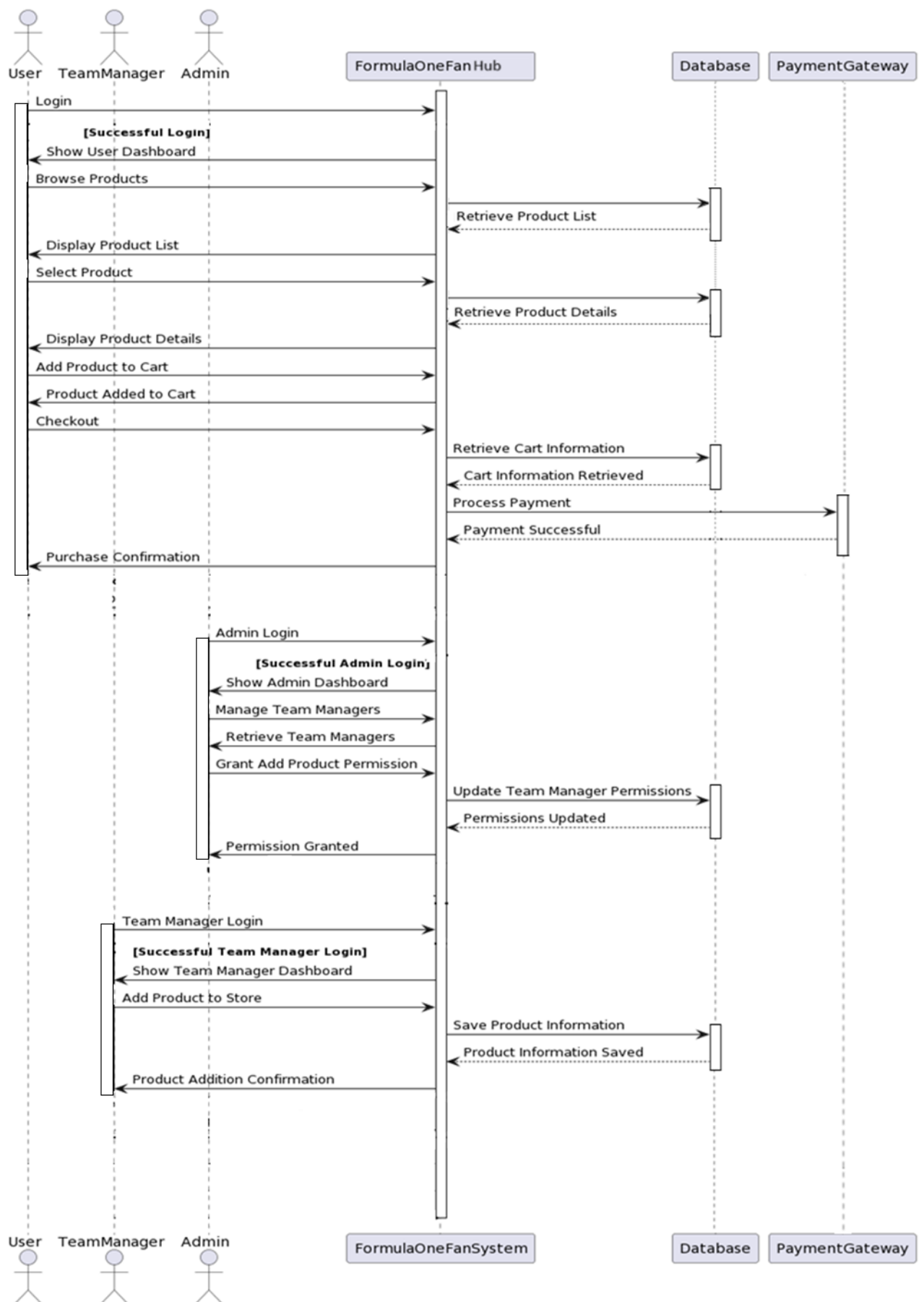


Figure 2: Sequence diagram

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.

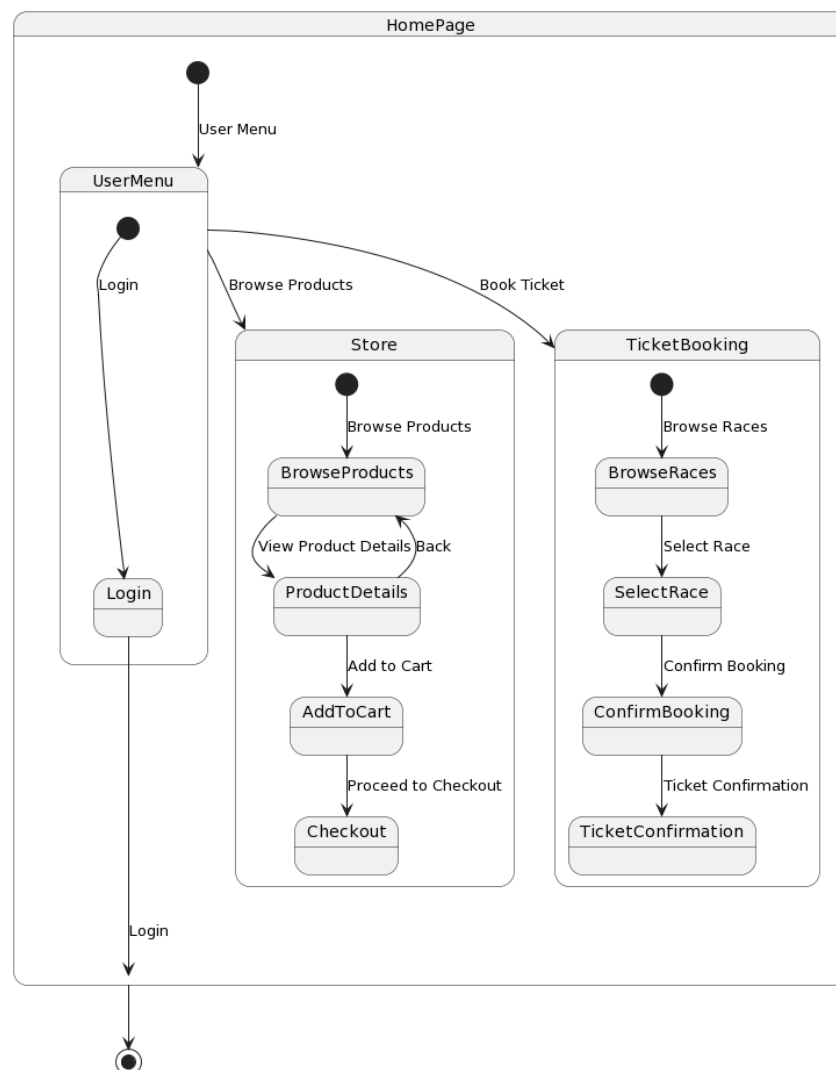


Figure 3: State Chart diagram

4.2.4 ACTIVITY DIAGRAM

It is a visual representation within UML that guidelines the way, the activities and actions in a process, flow. It employs various symbols to depict tasks, decision points, concurrency, and control flows. Rectangles signify activities or tasks, while diamonds represent decision points, allowing for conditional branching. Arrows indicate the flow of control from one activity to another. Forks and joins denote concurrency, where multiple activities can occur simultaneously or in parallel. Swimlane segregate activities based on the responsible entity, facilitating clarity in complex processes. Initial and final nodes mark the commencement and completion points of the activity. Decision nodes use guards to determine the path taken based on conditions. Synchronization bars enable the coordination of parallel activities. Control flows direct the sequence of actions, while object flows depict the flow of objects between activities. Activity Diagrams serve as invaluable tools for understanding, modeling, and analyzing complex workflows in systems and processes. They offer a structured visual representation that aids in effective communication and system development.

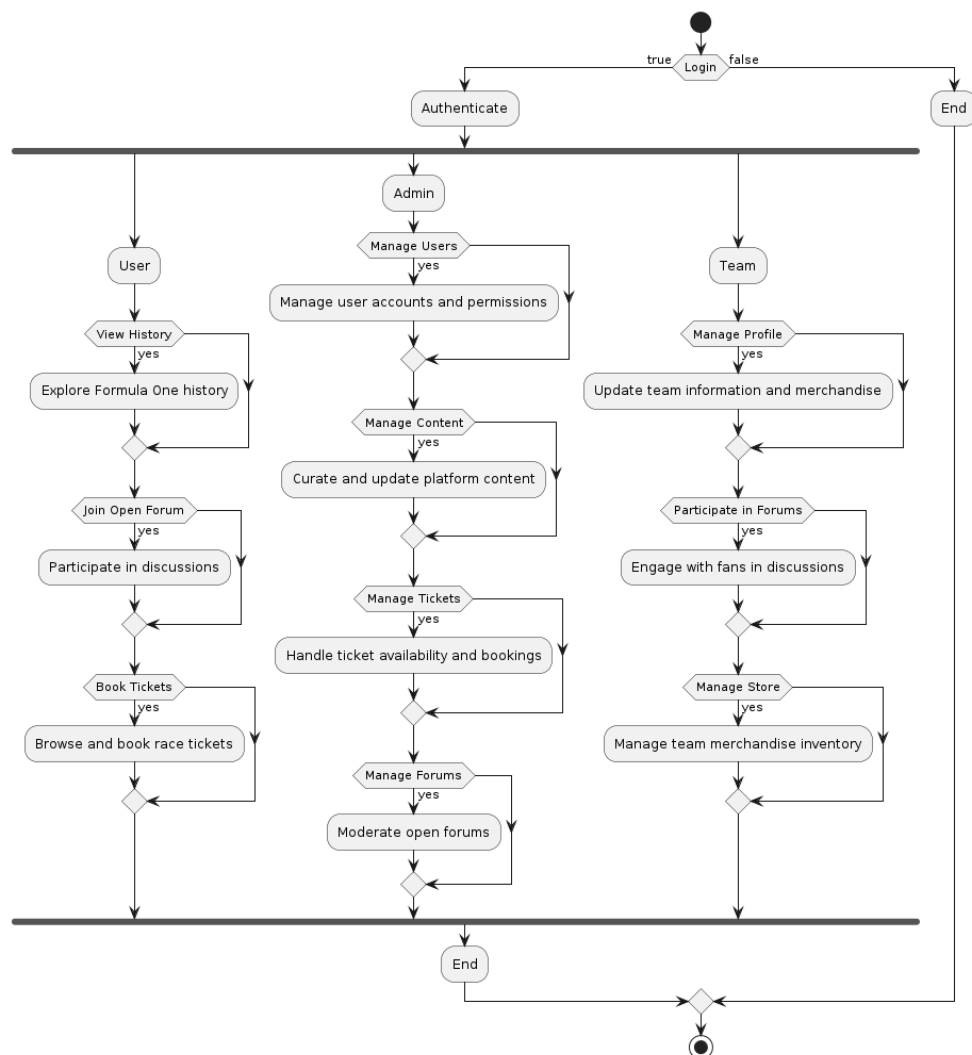


Figure 4: Activity diagram

4.2.5 CLASS DIAGRAM

It is a fundamental tool in UML used to visually show the structure of a system focusing on classes, attributes, methods, and relationships. Classes, depicted as rectangles, encapsulate data and behavior within a system. Associations between classes indicate relationships, showcasing how they interact. Multiplicity notations specify the cardinality of associations. Inheritance is denoted by an arrow indicating the subclass inheriting from a super-class. Stereotypes provide additional information about a class's role or purpose. Dependencies highlight the reliance of one class on another. Association classes facilitate additional information about associations. Packages group related classes together, aiding in system organization. Class Diagrams play a pivotal role in system design, aiding in conceptualizing and planning software architectures.

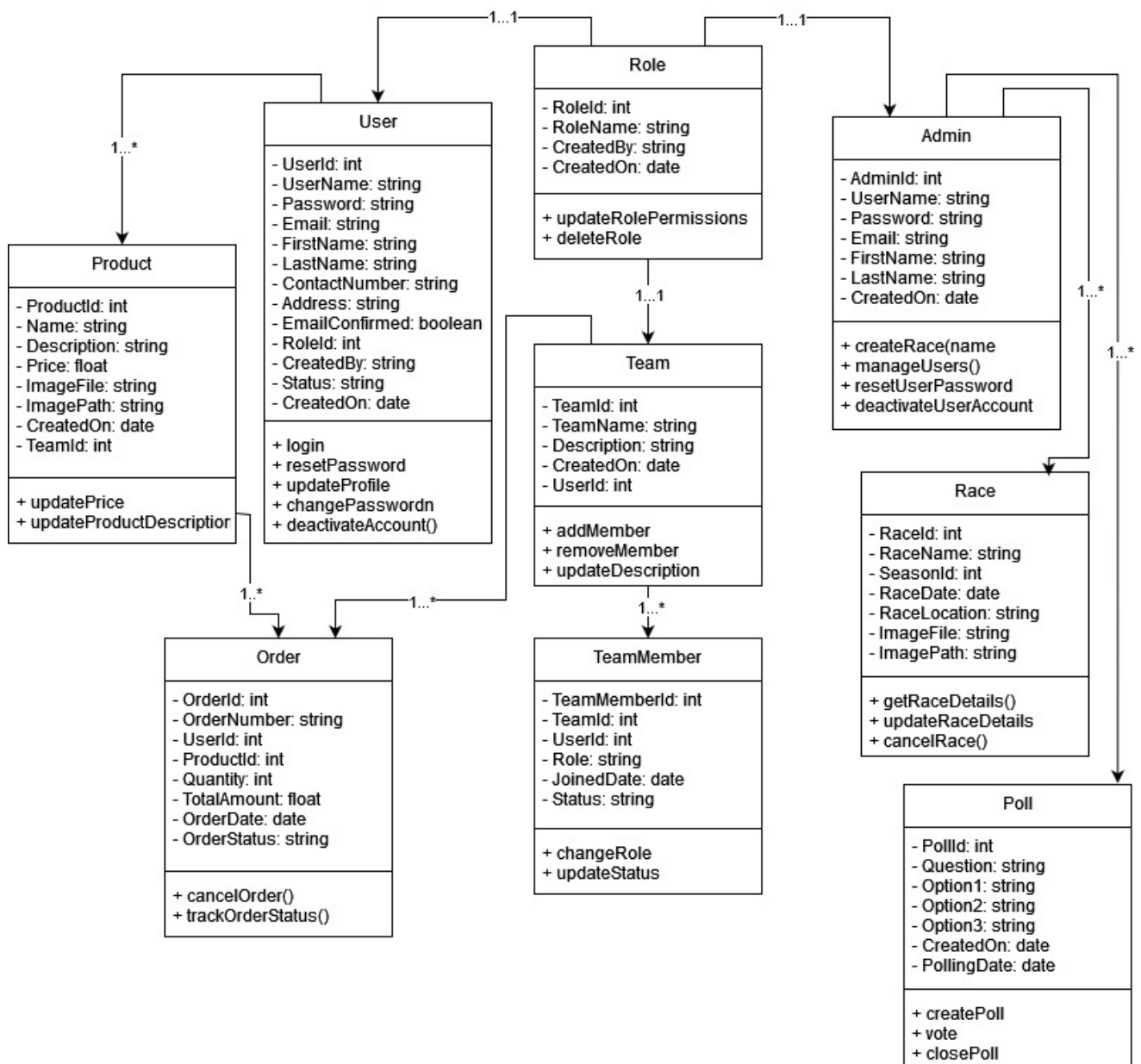


Figure 5: Class diagram

4.2.6 OBJECT DIAGRAM

UML, provides a quick look of a system at a specific time, displaying the instances of classes and their relationships. Objects, represented as rectangles, showcase the state and behavior of specific instances. Links between objects depict associations, highlighting how they interact. Multiplicity notations indicate the number of instances involved in associations. The object's state is displayed through attributes and their corresponding values. Object Diagrams offer a detailed view of runtime interactions, aiding in system understanding and testing. They focus on real-world instances, providing a tangible representation of class relationships. While similar to Class Diagrams, Object Diagrams emphasize concrete instances rather than class definitions.

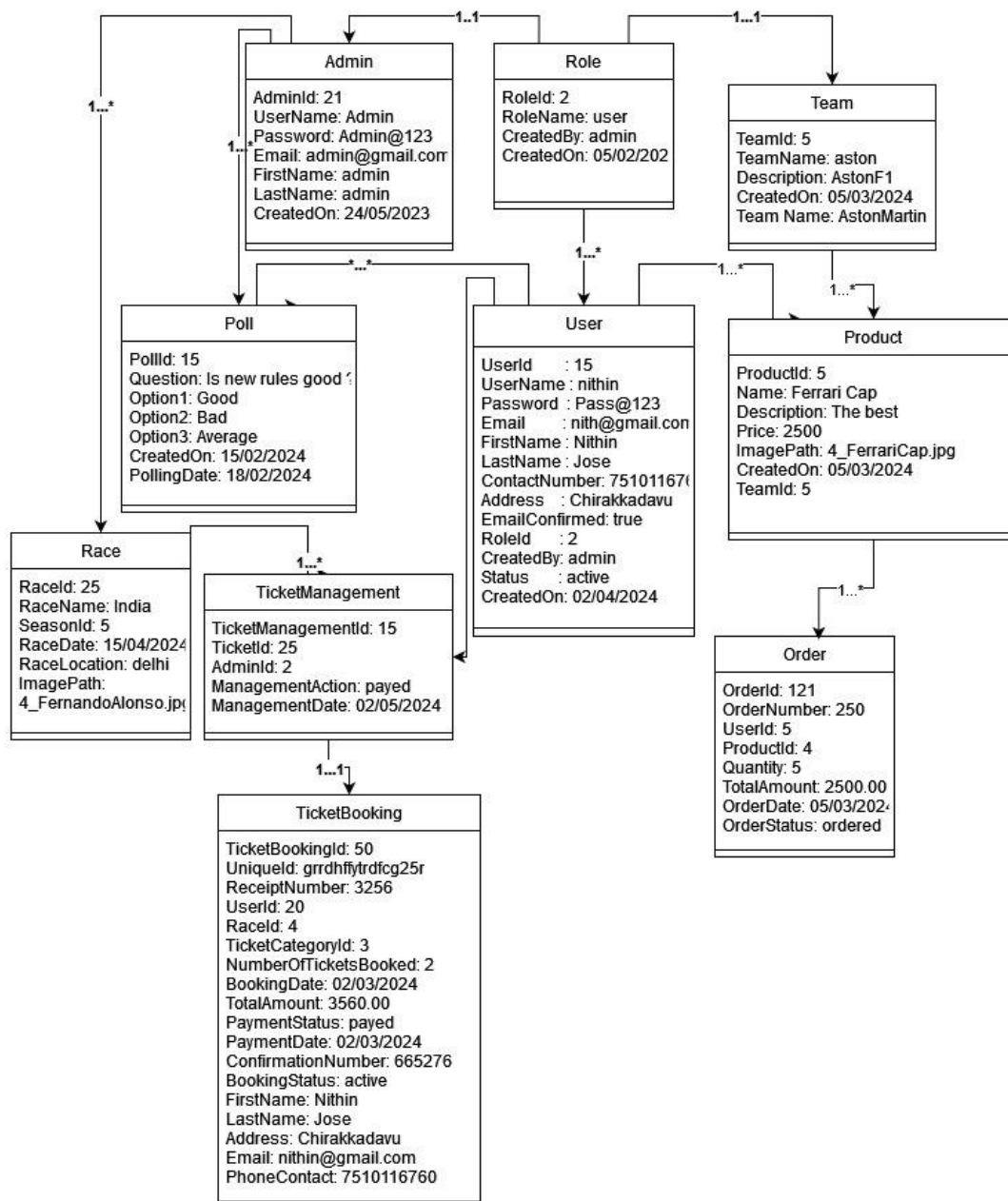


Figure 6: Object diagram

4.2.7 COMPONENT DIAGRAM

A Component Diagram, a vital aspect of UML, offers a visual representation of a system's architecture by showcasing the high-level components and their connections. Components, depicted as rectangles, encapsulate modules, classes, or even entire systems. Dependencies between components are displayed through arrows, signifying the reliance of one component on another. Interfaces, represented by a small circle, outline the services a component offers or requires. Connectors link interfaces to denote the required or provided services. Ports, depicted as small squares, serve as connection points between a component and its interfaces. Stereotypes provide additional information about the role or purpose of a component. Deployment nodes indicate the physical location or environment in which components are deployed. Component Diagrams are instrumental in system design, aiding in the organization and visualization of system architecture.

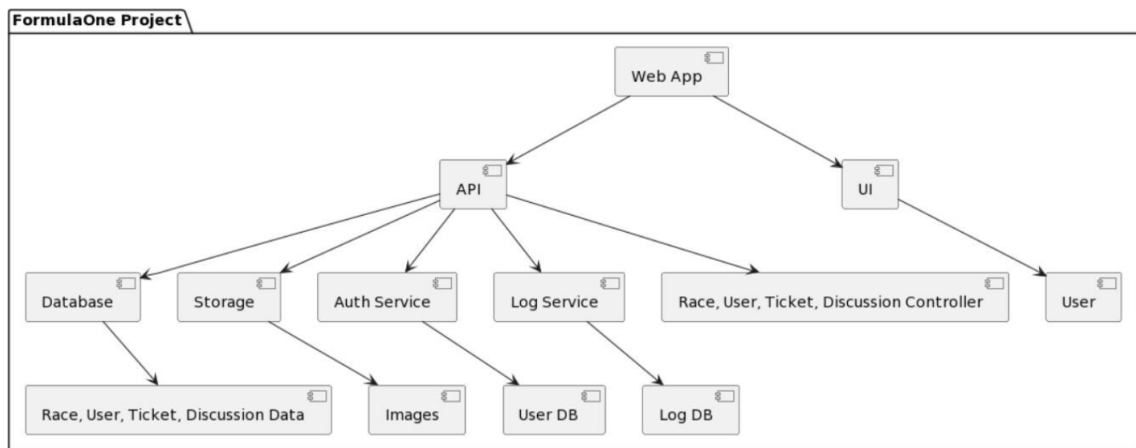


Figure 7: Component diagram

4.2.8 DEPLOYMENT DIAGRAM

It is a crucial facet of UML, provides a visual representation of the physical architecture, showcasing hardware nodes and components. Nodes, representing hardware entities like servers or devices, are depicted as rectangles. Artifacts, denoted by rectangles with a folded corner, represent software components or files deployed on nodes. Associations between nodes and artifacts indicate the deployment of software on specific hardware. Dependencies illustrate the reliance of one node on another. Communication paths, shown as dashed lines, represent network connections between nodes. Stereotypes provide additional information about the role or purpose of nodes and artifacts. Deployment Diagrams are instrumental in system planning, aiding in the visualization and organization of hardware and software components. They emphasize the allocation of software modules to specific hardware nodes, ensuring efficient utilization of resources.

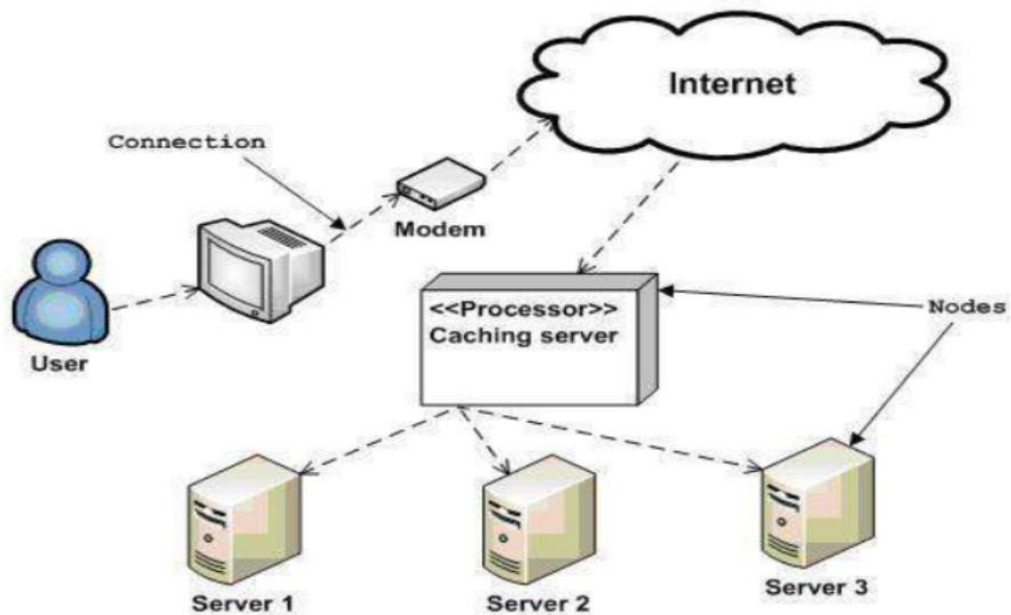


Figure 8: Deployment diagram

4.2.9 COLLABORATION DIAGRAM

It is a crucial facet of UML, provides a visual representation of the physical architecture, showcasing hardware nodes and components. Nodes, representing hardware entities like servers or devices, are depicted as rectangles. Artifacts, denoted by rectangles with a folded corner, represent software components or files deployed on nodes. Associations between nodes and artifacts indicate the deployment of software on specific hardware. Dependencies illustrate the reliance of one node on another. Communication paths, shown as dashed lines, represent network connections between nodes. Stereotypes provide additional information about the role or purpose of nodes and artifacts.

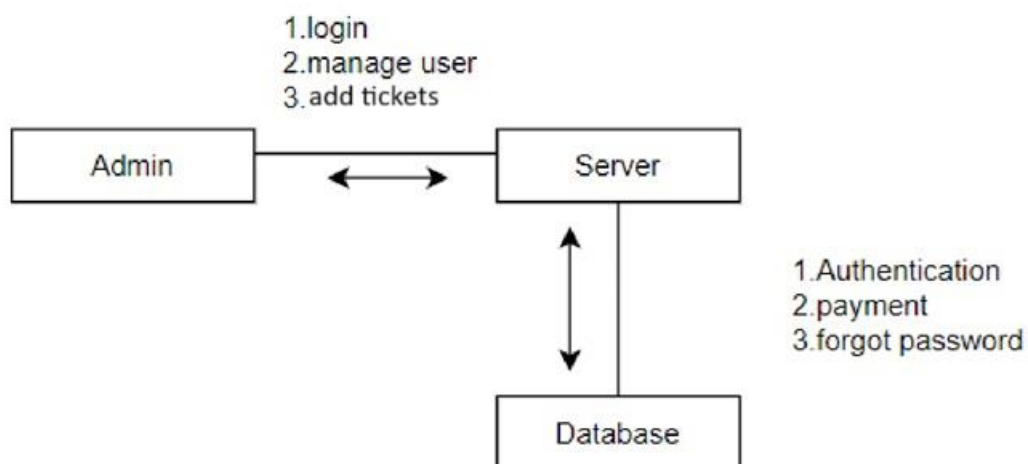
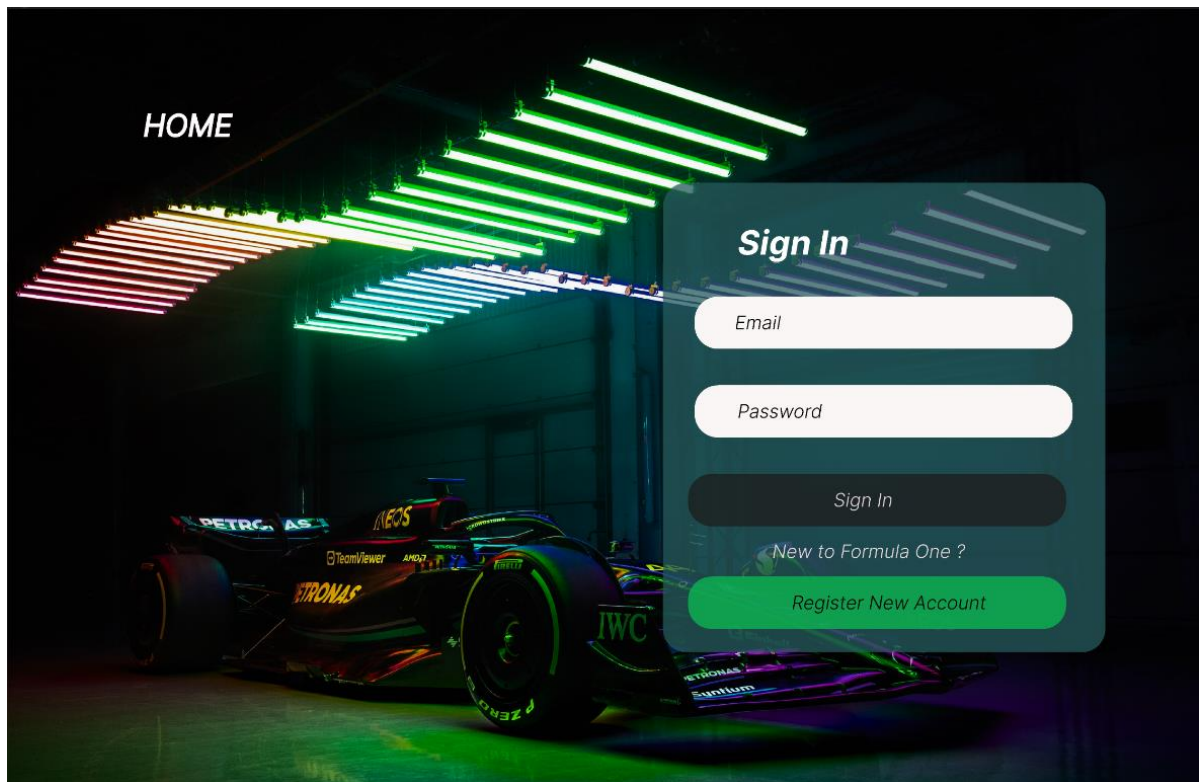


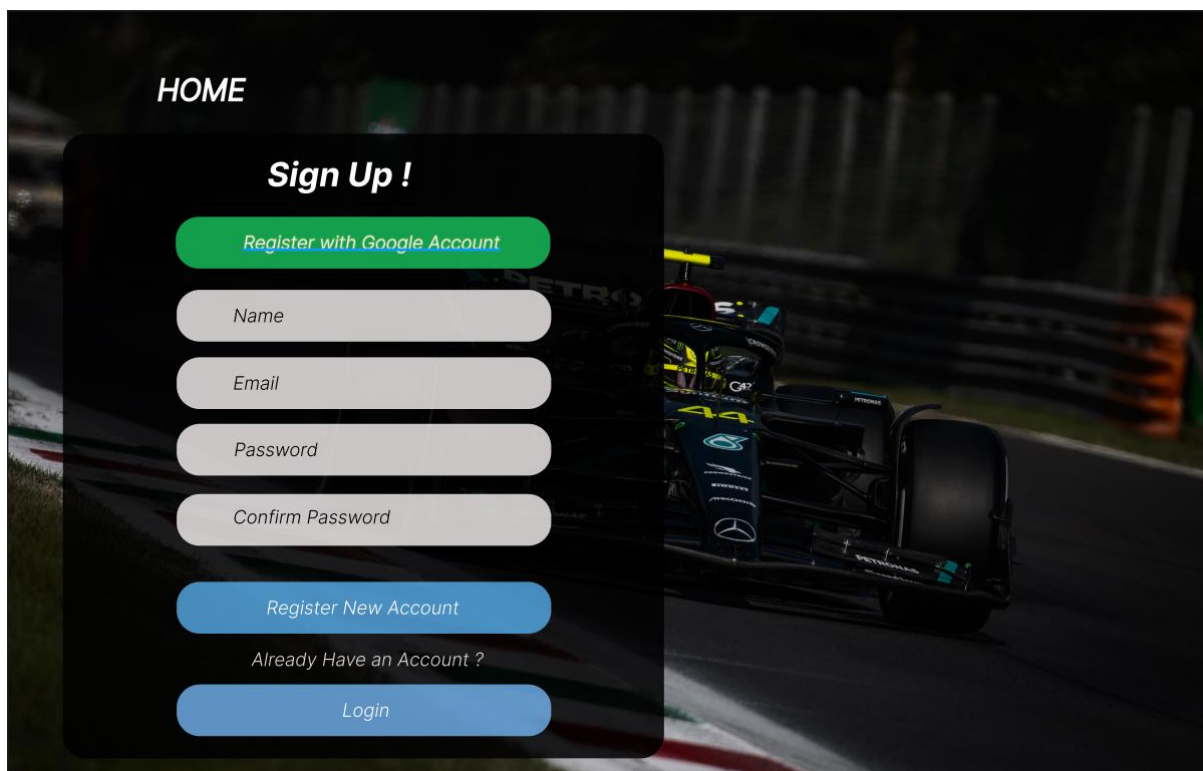
Fig 9: Collaboration Diagram

4.3 USER INTERFACE DESIGN USING FIGMA

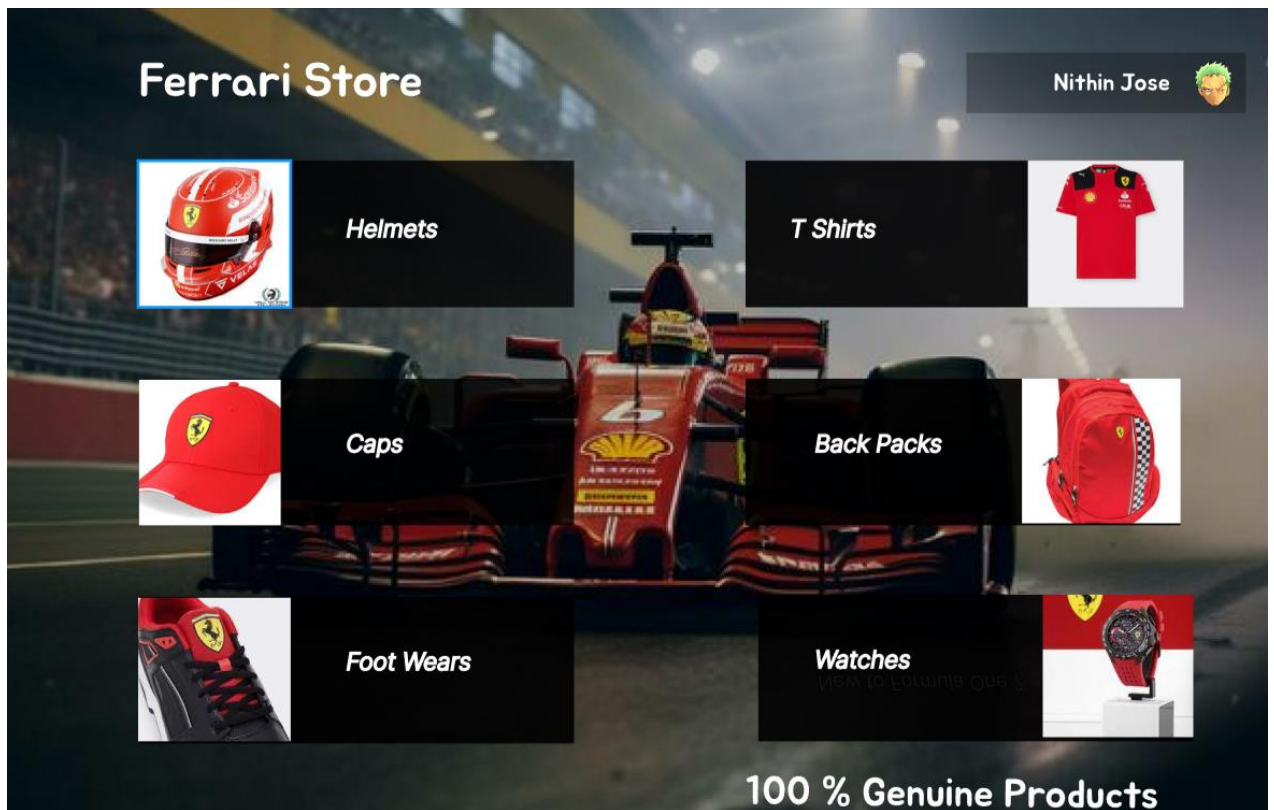
Home Page



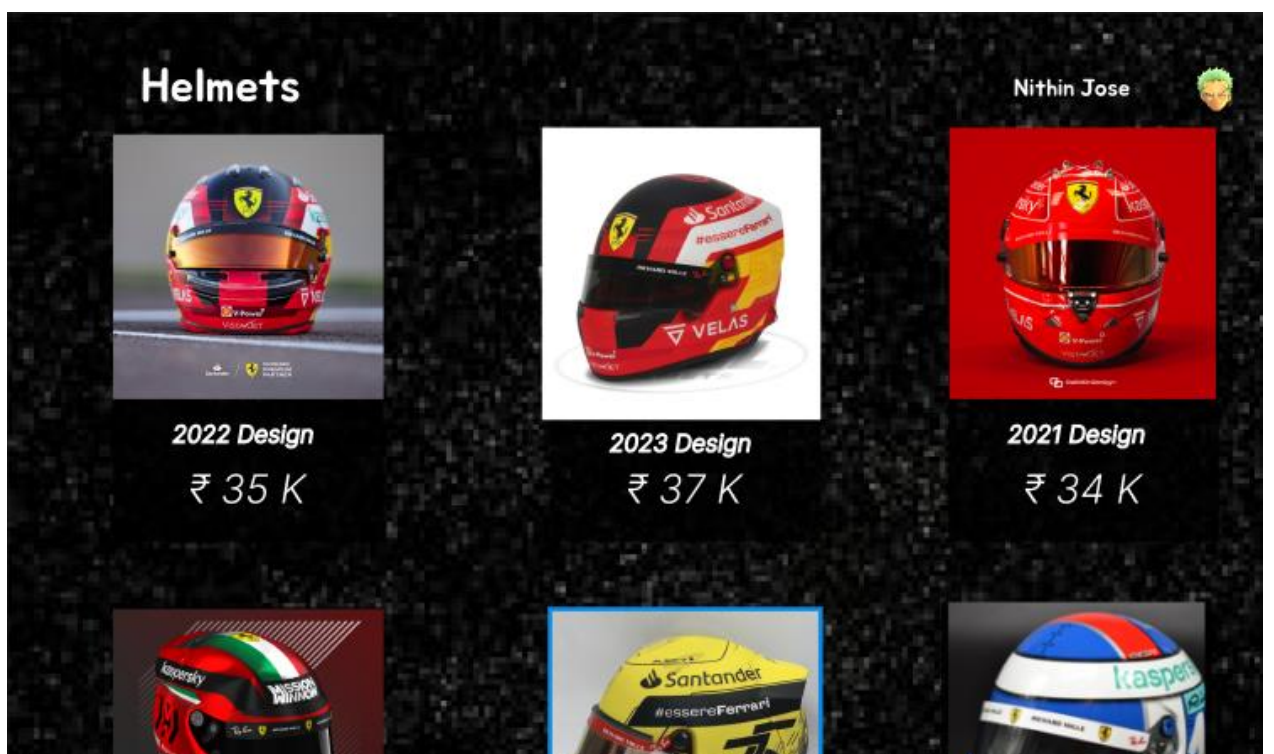
Registration Form



User Store



User Store Products Page



4.4 DATABASE DESIGN

Database Design is an important component in the realm of management and software development. It works by the thoughtful and systematic organization of data to ensure efficient storage, retrieval, and manipulation. A well-designed database serves as the backbone of applications, enabling them to handle large volumes of information with speed and accuracy. This process encompasses defining the structure, relationships, and constraints of data entities, optimizing for performance and scalability. Effective database design is pivotal in minimizing redundancy, ensuring data integrity, and providing a foundation for robust data analytics. It involves a deep understanding of business requirements and user needs, translating them into a coherent and logical data model. The goal of a sound database design is to create a reliable, scalable, and maintainable system that supports the organization's objectives and facilitates seamless information flow.

4.4.1 RELATIONAL DATABASE MANAGEMENT SYSTEM (RDBMS)

Data is represented via a relational model as a set of relationships. Every relationship can be thought of as a dataset or table of values. Tables are referred to as relations, rows as tuples, and column headers as attributes in relational model nomenclature. Tables make up a relational file, and each one has its own name. A set of linked values is represented by a row in the chart.

4.4.2 NORMALIZATION

Normalization is a database design technique used in relational database management systems (RDBMS) to eliminate data redundancy and improve data integrity. It involves organizing data in a way that reduces data duplication and ensures that relationships between tables are defined and maintained. The primary goals of normalization are:

- **Minimizing Data Redundancy:** By breaking down data into separate tables and ensuring that each piece of data is stored only once, normalization helps reduce the chances of data inconsistencies or errors.
- **Ensuring Data Integrity:** Normalization enforces the rules of referential integrity, which means that relationships between tables are well-defined and maintained. This ensures that data remains accurate and consistent.

By applying normalization principles, designers can create efficient and reliable database structures that support data integrity and ease data maintenance and manipulation.

There are several normal forms (NF) that define specific rules and requirements for achieving progressively higher levels of normalization. Here are the most common normal forms, from First Normal Form (1NF) to Fifth Normal Form (5NF):

First Normal Form (1NF)

If a table's atomicity is 1, then it is in first normal form. Atomicity prevents a single cell from having multiple values. It can only consist of one single valued attribute. The multivalued attribute, composite attribute are not allowed by the first normal form. Example: The table of students' records below includes details on each student's age, course, course number, and roll number. You can see that the course column in the students record table has two values. As a result, it deviates from the First Normal Form.

rollno	name	course	age
1	Rahul	c/c++	22
2	Harsh	java	18
3	Sahil	c/c++	23
4	Adam	c/c++	22
5	Lisa	java	24
6	James	c/c++	19
NULL	NULL	NULL	NULL

Second Normal Form (2NF)

The initial prerequisite for a table to meet the criteria of the Second Normal Form is its compliance with the First Normal Form. Additionally, the table should not exhibit partial dependency, implying that any subset of the candidate key should not determine a non-prime attribute. To illustrate the concept of the Second Normal Form, consider an example where the table needs to be divided into two separate parts. Below, provides an example of tables in the Second Normal Form:

cust_id	storeid	store_location
1	D1	Toronto
2	D3	Miami
3	T1	California
4	F2	Florida
5	H3	Texas

cust_id	storeid
1	D1
2	D3
3	T1
4	F2
5	H3

storeid	store_location
D1	Toronto
D3	Miami
T1	California
F2	Florida
H3	Texas

The column store location is completely dependent on the main key of that table, storied, as you have removed the partial functional reliance from the location table.

Third Normal Form (3NF)

Before a table can conform to the Third Normal Form (3NF), it must initially adhere to the Second Normal Form (2NF). One of the key requirements for 2NF is the absence of transitive dependencies for non-prime attributes. This implies that non-prime attributes, which are not part of the candidate key, should not be functionally dependent on other non-prime attributes. A transitive dependency occurs when a non-prime attribute is functionally dependent on another non-prime attribute, which, in turn, is functionally dependent on the candidate key. In simpler terms, if attribute A determines attribute B, and attribute B determines attribute C (where attribute B is not a candidate key itself), then there exists a transitive dependency. For instance, consider a student table with attributes such as student ID (stu_id), name, subject ID (sub_id), and subject. In this table, both sub_id and subject are

determined by stu_id. This creates a transitive dependency since subject depends indirectly on stu_id through sub_id. To normalize the table to 3NF, we need to split it into separate tables to remove the transitive dependency. This involves creating a table for students (with stu_id and name as attributes) and another table for subjects (with sub_id and subject as attributes), connecting them using appropriate foreign keys.

Here's an example of how to split the table:

	stu_id	name	subid	sub	address
▶	1	Arun	11	SQL	Delhi
	2	Varun	12	Java	Bangalore
	3	Harsh	13	C++	Delhi
	4	Keshav	12	Java	Kochi

	stu_id	name	subid	address
▶	1	Arun	11	Delhi
	2	Varun	12	Bangalore
	3	Harsh	13	Delhi
	4	Keshav	12	Kochi

	subid	subject
▶	11	SQL
	12	java
	13	C++
	12	Java

All non-key attributes are now completely functioning and solely rely on the primary key, as you can see in both tables. Name, sub_id, and addresses are the only fields in the first table that depend on stu_id. The sub solely depends on sub_id in the second table.

4.4.3 SANITIZATION

Data sanitization 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 in .NET Core is a fundamental database optimization technique. It involves creating data structures, or indexes, on specific fields to expedite data retrieval. These indexes act as signposts that enable the database to swiftly locate and fetch relevant data, especially in tables with substantial amounts of information. .NET offers automatic index creation for primary keys, unique fields, and foreign keys. Additionally, developers can define custom indexes for fields frequently used in filtering, sorting, or searching. The choice of proper indexing plays a pivotal role in improving query performance, resulting in more responsive and scalable web applications. It's essential to consider application-specific query patterns and create indexes accordingly, as well as to understand the capabilities and limitations of the chosen database backend. Effective indexing is a cornerstone of efficient database operations in .NET, contributing to enhanced application performance.

4.5 TABLE DESIGN

1. CartItem

Primary key: CartItemId

Foreign key: ProductId references table Product

Foreign Key: UserId references table User

No	Field name	Datatype	Key Constraints	Description of the field
1	CartItemId	Int	Primary Key	Unique identifier for Cart Item
2	ProductId	Int	Foreign Key	Identifier of the product
3	UserId	Int	Foreign Key	Identifier of the user
4	Quantity	Int	Not Null	Quantity of the product in the cart
5	Price	Decimal	Not Null	Price of the product
6	Timestamp	DateTime	Not Null	Date and time of the cart item
7	Status	String	Not Null	Status of the cart item
8	Size	String	Not Null	Size of the product

2. DeliveryCompany

Primary key: DeliveryCompanyId

No	Field name	Datatype	Key Constraints	Description of the field
1	DeliveryCompanyId	Int	Primary Key	Unique identifier for Delivery Company
2	UniqueName	String	Not Null	Unique name of the delivery company
3	Password	String	Not Null	Password for the delivery company
4	Email	String	Not Null	Email of the delivery company
5	CompanyName	String	Not Null	Name of the delivery company
6	ContactNumber	String	Not Null	Contact number of the delivery company
7	CompanyStatus	String	Not Null	Status of the delivery company
8	Address	String	Not Null	Address of the delivery company
9	ImagePath	String	Not Null	File path of the image representing the delivery company
10	CreatedOn	DateTime	Not Null	Date and time when the delivery company was created

3. Comment

Primary key: CommentId

Foreign key: UserId references table User

Foreign key: TopicId references table Topic

No	Field name	Datatype	Key Constraints	Description of the field
1	CommentId	Int	Primary Key	Unique identifier for Comment
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

4. Corner

Primary key: CornerId

Foreign key: RaceId references table Race

No	Field name	Datatype	Key Constraints	Description of the field
1	CornerId	Int	Primary Key	Unique identifier for Corner
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

5. Driver

Primary key: DriverId

Foreign key: TeamId Ref references table Team

No	Field name	Datatype	Key Constraints	Description of the field
1	DriverId	Int	Primary Key	Unique identifier for Driver
2	TeamIdRef	Int	Foreign Key	Identifier of the team
3	Name	String	Not Null	Name of the driver
4	Dob	DateTime	Not Null	Date of birth of the driver
5	Description	String	Not Null	Description of the driver
6	ImagePath	String	Not Null	File path of the image representing the driver
7	CreatedOn	DateTime	Not Null	Date and time when the driver was created
8	UpdatedOn	DateTime	Not Null	Date and time when the driver was last updated

6. F1History

Primary key: HistoryId

No	Field name	Datatype	Key Constraints	Description of the field
1	HistoryId	Int	Primary Key	Unique identifier for History
2	Heading	String	Not Null	Heading of the F1 history
3	Paragraph	String	Not Null	Paragraph describing the F1 history

7. Gallery

Primary key: ImageId

No	Field name	Datatype	Key Constraints	Description of the field
1	ImageId	Int	Primary Key	Unique identifier for Image
2	ImageUrl	String	Not Null	URL of the image in the gallery
3	Caption	String	Not Null	Caption for the image
4	UniqueName	String	Not Null	Unique name of the image
5	IsActive	String	Not Null	Flag indicating if the image is active or not

8. Order

Primary key: OrderId

Foreign key: DeliveryCompanyId references table DeliveryCompany

Foreign key: UserId references table User

No	Field name	Datatype	Key Constraints	Description of the field
1	OrderId	Int	Primary Key	Unique identifier for Order
2	UniqueId	String	Not Null	Unique identifier of the order
3	OrderDate	DateTime	Not Null	Date and time of the order
4	UserId	String	Foreign key	Identifier of the user who placed the order
5	Name	String	Not Null	Name of the customer
6	Email	String	Not Null	Email of the customer
7	PhoneNumber	String	Not Null	Phone number of the customer
8	Address	String	Not Null	Address for delivery
9	OrderStatus	String	Not Null	Status of the order
10	ShippingDate	DateTime	Not Null	Date and time of shipping
11	PaymentNumberRazor	String	Not Null	Payment number from Razorpay
12	PaymentStatus	String	Not Null	Status of the payment
13	PaymentDate	DateTime	Not Null	Date and time of payment
14	OrderIdRazor	String	Not Null	Order ID from Razorpay
15	OrderTotalAmount	Decimal	Not Null	Total amount of the order
16	DeliveryCompanyId	Int	Foreign Key	Identifier of the delivery company associated with the order

9. OrderedItem

Primary key: OrderedItemId

Foreign key: ProductId references table Product

Foreign key: OrderId references table Order

No	Field name	Datatype	Key Constraints	Description of the field
1	OrderedItemId	Int	Primary Key	Unique identifier for OrderedItem
2	ProductId	Int	Foreign Key	Identifier of the product
3	Quantity	Int	Not Null	Quantity of the ordered item
4	Price	Decimal	Not Null	Price per unit of the ordered item
5	DiscountPrice	Decimal	Not Null	Discounted price per unit of the ordered item
6	FinalPrice	Decimal	Not Null	Final price per unit of the ordered item
7	OrderId	Int	Foreign Key	Identifier of the associated order

10. Poll

Primary key: PollId

No	Field name	Datatype	Key Constraints	Description of the field
1	PollId	Int	Primary Key	Unique identifier for Poll
2	Question	String	Not Null	Question of 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
6	CreatedOn	DateTime	Not Null	Date and time when the poll was created
7	PollingDate	DateTime	Not Null	Date of the polling

11. Product

Primary key: ProductId

Foreign key: TeamId references table Team

Foreign key: ProductCategoryId references table ProductCategory

No	Field name	Datatype	Key Constraints	Description of the field
1	ProductId	Int	Primary Key	Unique identifier for Product
2	ProductName	String	Not Null	Name of the product
3	Description	String	Not Null	Description of the product
4	Price	Decimal	Not Null	Price of the product
5	TeamId	Int	Foreign Key	Identifier of the team associated with the product
6	ProductCategoryId	Int	Foreign Key	Identifier of the product category
7	StockQuantity	Int	Not Null	Quantity available in stock
8	ImagePath1	String	Not Null	Path to the first uploaded image on the server
9	ImagePath2	String	Not Null	Path to the second uploaded image on the server
10	ImagePath3	String	Not Null	Path to the third uploaded image on the server
11	ImagePath4	String	Not Null	Path to the fourth uploaded image on the server
12	IsActive	Bool	Not Null	Indicates whether the product is active for sale
13	UniqueName	String	Not Null	Unique name for the product
14	DiscountAmount	Decimal	Not Null	Discount amount in currency

12. ProductCategory

Primary key: ProductCategoryId

No	Field name	Datatype	Key Constraints	Description of the field
1	ProductCategoryId	Int	Primary Key	Unique identifier for Product Category
2	PCategoryName	String	Not Null	Name of the product category
3	ImagePath	String	Not Null	File path of the image representing the product category
4	CreatedOn	DateTime	Not Null	Date and time when the product category was created
5	UpdatedOn	DateTime	Not Null	Date and time when the product category was last updated
6	UniqueName	String	Not Null	Unique name of the product category

13. Race

Primary key: RaceId

Foreign key: SeasonId references table Season

No	Field name	Datatype	Key Constraints	Description of the field
1	RaceId	Int	Primary Key	Unique identifier for Race
2	UniqueRaceName	String	Not Null	Unique name of the race
3	RaceName	String	Not Null	Name of the race
4	SeasonId	Int	Foreign Key	Identifier of the associated season
5	RaceDate	DateTime	Not Null	Date of the race
6	RaceLocation	String	Not Null	Location of the race
7	ImagePath	String	Not Null	File path of the image representing the race

14. Role

Primary key: RoleId

No	Field name	Datatype	Key Constraints	Description of the field
1	RoleId	Int	Primary Key	Unique identifier for Role
2	RoleName	String	Not Null	Name of the role
3	CreatedBy	String	Not Null	Creator of the role
4	CreatedOn	DateTime	Not Null	Date and time when the role was created

15. Season

Primary key: SeasonId

No	Field name	Datatype	Key Constraints	Description of the field
1	SeasonId	Int	Primary Key	Unique identifier for Season
2	UniqueSeasonName	String	Not Null	Unique name of the season
3	Year	Int	Not Null	Year of the season
4	Champion	String	Not Null	Champion of the season
5	ImagePath	String	Not Null	File path of the image representing the season

16. TeamHistory

Primary key: HistoryId

Primary key: TeamId references table Team

No	Field name	Datatype	Key Constraints	Description of the field
1	HistoryId	Int	Primary Key	Unique identifier for History
2	Heading	String	Not Null	Heading of the team history
3	Paragraph	String	Not Null	Paragraph describing the team history
4	TeamId	Int	Foreign Key	Identifier of the team

17. SoldItem

Primary key: SoldItemId

Foreign key: OrderId references table Order

Foreign key: ProductId references table Product

Foreign key: TeamId references table Team

No	Field name	Datatype	Key Constraints	Description of the field
1	SoldItemId	Int	Primary Key	Unique identifier for SoldItem
2	OrderId	Int	Foreign Key	Identifier of the order
3	ProductId	Int	Foreign Key	Identifier of the product
4	TeamId	Int	Foreign Key	Identifier of the team
5	Quantity	Int	Not Null	Quantity of sold items
6	PricePerItem	Decimal	Not Null	Price per item
7	TotalPrice	Decimal	Not Null	Total price of sold items
8	SoldDate	DateTime	Not Null	Date and time of sale
9	Status	String	Not Null	Status of the sold item

18. Team

Primary key: TeamId

No	Field name	Datatype	Key Constraints	Description of the field
1	TeamId	Int	Primary Key	Unique identifier for Team
2	userName	String	Not Null	Username of the team
3	Name	String	Not Null	Name of the team
4	Password	String	Not Null	Password of the team
5	Email	String	Not Null	Email of the team
6	PhoneNumber	String	Not Null	Phone number of the team
7	Address1	String	Not Null	Address line 1 of the team
8	Address2	String	Not Null	Address line 2 of the team
9	Address3	String	Not Null	Address line 3 of the team
10	Country	String	Not Null	Country of the team
11	Status	String	Not Null	Status of the team
12	TeamPrincipal	String	Not Null	Principal of the team
13	TechnicalChief	String	Not Null	Technical chief of the team
14	EngineSupplier	String	Not Null	Engine supplier of the team
15	Chassis	String	Not Null	Chassis of the team
16	ImagePath	String	Not Null	File path of the image representing the team
17	CreatedOn	DateTime	Not Null	Date and time when the team was created
18	UpdatedOn	DateTime	Not Null	Date and time when the team was last updated

19. TicketBooking

Primary key: TeamId

Primary key: UserId references table User

Primary key: SeasonId references table Season

Primary key: RaceId references table Race

Primary key: CornerId references table Corner

Primary key: TicketCategoryId references table TicketCategory

No	Field name	Datatype	Key Constraints	Description of the field
1	TicketBookingId	Int	Primary Key	Unique identifier for Ticket Booking
2	UniqueId	String	Not Null	Unique identifier for the ticket booking
3	ReceiptNumber	String	Not Null	Receipt number
4	UserId	Int	Foreign Key	Identifier of the user
5	SeasonId	Int	Foreign Key	Identifier of the season
6	RaceId	Int	Foreign Key	Identifier of the race
7	CornerId	Int	Foreign Key	Identifier of the corner
8	TicketCategoryId	Int	Foreign Key	Identifier of the ticket category
9	NumberOfTicketsBooked	Int	Not Null	Number of tickets booked
10	BookingDate	DateTime	Not Null	Date and time of booking
11	TotalAmount	Decimal	Not Null	Total amount of booking
12	PaymentStatus	String	Not Null	Payment status
13	PaymentDate	DateTime	Not Null	Date and time of payment
14	ConfirmationNumber	String	Not Null	Confirmation number
15	BookingStatus	String	Not Null	Booking status
16	FirstName	String	Not Null	First name of the booker
17	LastName	String	Not Null	Last name of the booker
18	Address	String	Not Null	Address of the booker
19	Email	String	Not Null	Email of the booker
20	PhoneContact	String	Not Null	Phone contact of the booker

20. TicketCategory

Primary key: TicketCategoryId

No	Field name	Datatype	Key Constraints	Description of the field
1	TicketCategoryId	Int	Primary Key	Unique identifier for Ticket Category
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	ImagePath	String	Not Null	File path of the image representing the ticket category

21. Topic

Primary key: TopicId

Foreign key: TeamId references table Team

No	Field name	Datatype	Key Constraints	Description of the field
1	TopicId	Int	Primary Key	Unique identifier for Topic
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 was created
5	TeamId	Int	Foreign Key	Identifier of the team who created the topic

22. WishList

Primary key: WishListId

Foreign key: ProductId references table Product

Foreign key: UserId references table User

No	Field name	Datatype	Key Constraints	Description of the field
1	WishListId	Int	Primary Key	Unique identifier for WishList
2	ProductId	Int	Foreign Key	Identifier of the product in the wishlist
3	UserId	Int	Foreign Key	Identifier of the user who created the wishlist

23. User

Primary key: UserId

Foreign key: RoleId references table Role

No	Field name	Datatype	Key Constraints	Description of the field
1	UserId	Int	Primary Key	Unique identifier for User
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 the user's email is confirmed
10	RoleId	Int	Foreign Key	Identifier of the user's role
11	CreatedBy	String	Not Null	Creator of the user
12	Status	String	Not Null	Status of the user
13	CreatedOn	DateTime	Not Null	Date and time when the user was created

24. Vote

Primary key: VoteId

Foreign key: UserId references table User

Foreign key: PollId references table Poll

No	Field name	Datatype	Key Constraints	Description of the field
1	VoteId	Int	Primary Key	Unique identifier for Vote
2	UserId	Int	Foreign Key	Identifier of the user who voted
3	PollId	Int	Foreign Key	Identifier of the poll
4	OptionSelected	Int	Not Null	Selected option in the poll

CHAPTER 5

SYSTEM TESTING

5.1 INTRODUCTION

Software Testing is the methodical process of executing a program to find any possible bugs or problems. A well-designed test case has a high probability of identifying problems that were previously overlooked. When a test uncovers an error that was previously unknown, it is deemed successful. A test can identify software defects if it runs according to plan and achieves its goals. The test indicates whether the computer program is running at its best and according to its planned functionality. A computer program can be evaluated using three main methods: computational complexity analysis, correctness evaluation, and implementation efficiency evaluation.

5.2 TEST PLAN

A test plan is a thorough document that delineates the strategy, scope, objectives, resources, schedule, and expected outcomes for a specific testing endeavor. It functions as a guiding framework for carrying out testing activities, guaranteeing that every facet of the testing process is methodically organized and executed. Additionally, the test plan establishes the roles and responsibilities of team members, outlines the required testing environment, and sets forth the criteria for the successful completion of testing activities. This document plays a pivotal role in ensuring that the testing phase is conducted in a structured and effective manner, ultimately contributing to the overall success of the project.

The testing levels include:

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

Overall, defining the scope, identifying the test environment, defining the test cases, generating the test scripts, running the tests, recording the results, analyzing the results, and reporting the results are all steps in the Selenium test plan creation process. By doing the following actions, you can make sure that your testing is thorough and efficient and that you are able to spot and fix any issues before they become serious ones.

5.2.1 UNIT TESTING

Unit testing serves as a crucial foundation for the entire software testing process, where the focus lies on isolating and scrutinizing individual units of code. The significance of Unit Testing cannot be overstated, as it acts as a vanguard against potential discrepancies or errors early in the development cycle. This proactive approach not only fortifies the integrity and reliability of the

software but also lays the groundwork for subsequent testing phases, thereby fostering a robust and dependable software solution. This meticulous process ensures that each unit functions reliably and adheres precisely to its defined behaviour. By subjecting individual code units to rigorous scrutiny, any discrepancies or errors are identified and rectified early in the development cycle, bolstering the overall integrity and reliability of the software.

5.2.2 INTEGRATION TESTING

It stands as a pivotal phase in the software testing process, dedicated to scrutinizing the interactions and interfaces among diverse modules or components in a software system. Primary objective is ascertaining that individual units of code seamlessly converge to create a unified and functional system. In stark contrast to unit testing, which assesses individual units in isolation, integration testing delves into the interplay between these units, with a keen eye for any disparities, communication glitches, or integration hurdles. By subjecting the integrated components to rigorous testing, development teams aim to affirm that these elements function cohesively, addressing any potential issues before deployment. This systematic evaluation is instrumental in ensuring that the software operates as an integrated whole, free from any unforeseen conflicts or errors that may arise from the convergence of individual modules.

5.2.3 VALIDATION TESTING OR SYSTEM TESTING

It places end-users at the forefront of evaluation, ensuring that the software aligns precisely with their anticipated needs and expectations. This phase stands distinct from other testing methodologies, as its primary objective is to authenticate that the software, in its final form, serves its intended purpose seamlessly within the real-world scenarios it was designed for. As a culmination of the testing process, Validation Testing carries the responsibility of confirming that the software not only meets the defined technical specifications but also delivers genuine value to its users. It does so by scrutinizing the software against the backdrop of actual usage, thereby fortifying its readiness for deployment. Moreover, in Validation Testing, user stories and acceptance criteria form the cornerstone of assessment. Stakeholders' expectations are meticulously validated, ensuring that every specified requirement is met. These testing techniques are applied during validation testing to ensure comprehensive coverage of the software's functionality. For instance, software engineers can design input conditions that address various program requirements, facilitating thorough testing. Additionally, beta testing, a common practice in this phase, involves a select group of end-users testing the software in a live environment, providing invaluable feedback that can inform potential refinements.

5.2.3 OUTPUT TESTING OR USER ACCEPTANCE TESTING

It also known as Results Validation, is a critical phase in the software testing process. Its primary focus is to verify the correctness and accuracy of the output generated by a software application. The goal is to make sure that the system produces the expected results for a given set of inputs and conditions.

Key aspects of Output Testing include:

- **Comparison with Expected Results:** This phase involves comparing the actual output of the software with the expected or predefined results.
- **Test Case Design:** Test cases are designed to cover various scenarios and conditions to thoroughly evaluate the accuracy of the output.
- **Validation Criteria:** The criteria for validating the output are typically defined during the requirements and design phase of the software development process.
- **Regression Testing:** Output Testing often includes regression testing to ensure that changes or updates to the software do not affect the correctness of the output.
- **Data Integrity:** It verifies that data is processed and displayed correctly, without any corruption or loss.
- **Precision and Completeness:** Output Testing assesses not only the precision of the results but also their completeness in addressing the requirements.
- **Error Handling:** It evaluates how the system handles errors or exceptions and ensures that appropriate error messages are displayed.

5.2.4 AUTOMATION TESTING

It stands as a cornerstone in the testing process, harnessing power of automated tools and scripts to meticulously execute test cases. In stark contrast to manual testing, which hinges on human intervention, automation testing brings forth a streamlined approach, employing software to conduct repetitive, intricate, and time-consuming tests. This methodology not only heightens operational efficiency but also significantly diminishes the likelihood of human error, ensuring precise and reliable results. Moreover, it empowers thorough testing across a diverse array of scenarios and configurations, from browser compatibility to load and performance assessments. This approach significantly reduces manual effort and enhances efficiency in the testing process. However, manual testing, performed by individuals seated in front of a computer, is still essential for certain test stages. By automating the testing process, organizations can realize a myriad of benefits. It enables the seamless execution of regression tests, providing confidence that existing functionalities remain intact after each round of enhancements or modifications. Furthermore, automation facilitates the concurrent execution of multiple tests, thereby expediting the overall

testing cycle. This approach is particularly invaluable in environments characterized by rapid development and frequent software updates, such as Agile and DevOps setups.

5.2.5 SELENIUM TESTING

Selenium is an extensively adopted open-source framework essential for automating web browsers, particularly in software testing. Its primary function is to automate web applications for testing purposes, although it's also adept at handling web-based administrative tasks. Selenium's flexibility arises from its capacity to replicate real-user interactions with web browsers, making it indispensable for testing web applications under real-world scenarios. One of Selenium's core strengths lie in its support for multiple programming languages such as Java, C#, Python, and Ruby. This versatility empowers testers to write test scripts in their preferred language, enhancing productivity and ease of adoption. Testing done using Selenium testing tool is usually referred to as Selenium testing. Moreover, Selenium's extensibility allows integration with other testing frameworks, tools, and technologies. It can be combined with testing frameworks like TestNG or JUnit for advanced test management and reporting. It integrates well with popular build tools, source control systems, and defect tracking systems, enhancing its usability and effectiveness in software testing processes. Selenium offers different variations, with Selenium WebDriver and Selenium Grid being the most notable. WebDriver directly interacts with the web browser, leveraging its native compatibility for automation. Conversely, Selenium Grid enables parallel test execution across different machines and browsers, significantly expediting the testing process. Utilizing Selenium for automated testing reduces manual effort, particularly in repetitive testing scenarios, while also bolstering test accuracy by minimizing human error. This results in more dependable, efficient, and expedited testing cycles, crucial in agile and DevOps environments where rapid product releases are imperative.

Test Case 1

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: 15-04-2024

Module Name: Login

Test Executed By: Ms. Gloriya Mathew

Test Title: Login Test

Test Execution Date: 15-04-2024

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:**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: 15-04-2024		
Module Name: Add New Topic			Test Executed By: Ms. Gloriya Mathew		
Test Title: Add Topic			Test Execution Date: 15-04-2024		
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:**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 {
    // Add explicit wait for the element with name "testHistoryHead"
    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 anyawait 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!
.Expected 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: 15-04-2024		
Module Name: Add New F1 History			Test Executed By: Ms. Gloriya Mathew		
Test Title: Add New F1History			Test Execution Date: 15-04-2024		
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:**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();
});

```

Screenshot



```

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-dbf4-42c2-954a-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!
.Expected 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: 15-04-2024		
Module Name: add new Team history			Test Executed By: Ms. Gloriya Mathew		
Test Title: Add New Team History			Test Execution Date: 15-04-2024		
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					

Test Case 5:**Code**

```
const { After, Given, When, Then } = require("cucumber");
const { Builder, By, until } = require("selenium-webdriver"); // Ensure correct import
statement
let driver;

Given ("I am on the login page ATC", async function () {
    driver = await new Builder().forBrowser("chrome").build();
});

When ("I enter valid credentials ATC", async function () {
    await driver.findElement(By.id("username")).sendKeys("nithinjose");
    await driver.findElement(By.id("password")).sendKeys("Manjadi@123");
});

When ("click on the login button ATC", async function () {
    try {
        const button = await driver.wait(
            until.elementLocated(By.id("testid")),
            5000
        );
        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 ATC", async function () {
    await driver.wait(until.urlIs("http://localhost:3000/UserHome"));
    console.log ("Redirected to the home page!");
});

When ("I click on the StoreElement in the UserNavbar ATC", async function () {
    try {
        const storeElement = await driver.findElement(By.name("StoreNavBar"));
        await storeElement.click();
        console.log ("Clicked on the StoreElement in UserNavbar!");
    }
});
```

```
    } catch (error) {  
        console.error("Error clicking on the StoreElement in UserNavbar:", error);  
    }  
});
```

When ('I click on the ExploreStore in the UserNavbar ATC', async function () {

```
    try {  
        const exploreStore = await driver.findElement(By.name("ExpStore"));  
        await exploreStore.click();  
        console.log ("Clicked on the ExploreStore in UserNavbar!");  
    } catch (error) {  
        console.error("Error clicking on the ExploreStore in UserNavbar:", error);  
    }  
});
```

When ('I click on a productCategory ATC', async function () {

```
    try {  
        // Find the category item element  
        const categoryItem = await driver.findElement(By.name("hello"));  
        // Click on the category item  
        await categoryItem.click();  
        console.log ("Clicked on a productCategory!");  
    } catch (error) {  
        console.error("Error clicking on a productCategory:", error);  
    }  
});
```

When ('I click on a product ATC', async function () {

```
    try {  
        // Find the product item element  
        const productItem = await driver.findElement(By.className("product-item"));  
        // Click on the product item  
        await productItem.click();  
        console.log ("Clicked on a product!");  
    } catch (error) {  
        console.error("Error clicking on a product:", error);  
    }  
}
```

```
});  
When ('I click on the Add to Cart button ATC', async function () {  
  try {  
    // Find the Add to Cart button element  
    const addToCartButton = await driver.findElement(By.className("add-to-cart-  
button"));  
    // Click on the Add to Cart button  
    await addToCartButton.click();  
    console.log ("Clicked on the Add to Cart button!");  
  } catch (error) {  
    console.error("Error clicking on the Add to Cart button:", error);  
  }  
});
```

Screenshot

```
PS E:\Project\testing\testing> npm test  
  
> testing@1.0.0 test  
> cucumber-js features/testAddToCart.feature  
  
DevTools listening on ws://127.0.0.1:57447/devtools/browser/b3e593e7-8064-458d-8a0a-b1f1e6c483fb  
..Login successful!  
.Redirected to the home page!  
.Clicked on the StoreElement in UserNavbar!  
.Clicked on the ExploreStore in UserNavbar!  
.Clicked on a productCategory!  
.Clicked on a product!  
.Clicked on the Add to Cart button!  
.  
  
1 scenario (1 passed)  
9 steps (9 passed)  
0m04.850s  
PS E:\Project\testing\testing> Created TensorFlow Lite XNNPACK delegate for CPU.
```


Test report

Test Case 5					
Project Name: Formula One Fan Hub					
Add Product To Cart					
Test Case ID: Test5			Test Designed By: Nithin Jose		
Test Priority (Low/Medium/High): High			Test Designed Date: 15-04-2024		
Module Name: add product to cart			Test Executed By: Ms. Gloriya Mathew		
Test Title: Add Product To cart			Test Execution Date: 15-04-2024		
Description: Testing adding a product to the cart					
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: nithinjose	User should be able to login	User navigated to the index page with successful login	Pass
3	Provide valid password	Password: Manjadi@123			
4	Click on the login button				
5	Navigate to the store main page		Main page of store is displayed	User navigated to the store page	Pass
6	Click on the first product category	Category Selected: Flask	Products are displayed for the selected category	User navigated to the products page	Pass
7	Click on the first product	Product Selected: Redbull Flask	Product details are displayed	User navigated to the product details page	Pass
8	Click on the Add to Cart button		Item added to cart and cart page is displayed	Cart page for the user is displayed with product added	Pass
Post-Condition: After successful login, user can add a new product to the cart					

CHAPTER 6

IMPLEMENTATION

6.1 INTRODUCTION

The implementation phase of a project is pivotal as it marks the transition from design to a functional system, ensuring its success. It involves converting the revised system design into an operational one, which demands gaining user confidence in its effectiveness and accuracy. User training and documentation are crucial aspects during this phase, alongside the conversion process. Implementing a new system, whether entirely new or replacing an existing one, requires meticulous planning to meet organizational needs. Poorly managed implementation can lead to confusion and disruption. Therefore, thorough testing is essential before system implementation. It involves significant effort in education and training, system testing, and changeover. Careful planning, assessing constraints, and designing changeover methods are vital in the implementation phase.

6.2 IMPLEMENTATION PROCEDURES

Software implementation is the process of installing the software in its actual environment and ensuring that it satisfies the intended use and operates as expected. In some organizations, the software development project may be commissioned by someone who will not be using the software themselves. During the initial stages, there may be doubts about the software, but it's important to ensure that resistance does not build up. This can be achieved by:

- Ensuring that active users are aware of the benefits of the new system, building their confidence in the software.
- Providing proper guidance to the users so that they are comfortable using the application.

Before viewing the system, users should know that the server program must be running on the server. Without the server object up and running, the intended process will not take place.

6.2.1 USER TRAINING

User training is a critical component of ensuring the effective utilization of any website. It involves imparting the necessary knowledge and skills to end-users, enabling them to navigate and utilize the website efficiently. This training equips users with a comprehensive understanding of the website's features, functions, and capabilities. Through hands-on sessions and guided tutorials, users learn how to perform tasks, customize settings, and troubleshoot common issues. Moreover, user training fosters confidence and proficiency, empowering individuals to maximize their productivity while using the application. Regular updates and refresher sessions further enhance user competence, ensuring they stay abreast of new features and functionalities.

6.2.2 TRAINING ON THE WEBSITE

Training on the website is a structured program designed to familiarize individuals with the intricacies and functionalities of a specific website. It encompasses a range of topics, from basic navigation to advanced features, tailored to meet the diverse needs of users. Trainers may also provide supplemental resources such as user manuals or online guides for reference. By the end of the training, participants are equipped with skills and knowledge required to proficiently utilize the website in their respective contexts.

6.2.3 SYSTEM MAINTENANCE

System maintenance is a crucial aspect of ensuring the seamless operation and longevity of any website application. It encompasses a series of tasks aimed at monitoring, optimizing, and troubleshooting the underlying infrastructure on which the application runs. This includes activities such as regular performance monitoring, and data backups. Additionally, system maintenance involves identifying and rectifying any potential vulnerabilities or inefficiencies that may impede the website's performance. Proactive maintenance measures contribute to a stable and secure environment, minimizing the risk of unexpected downtime or data loss.

6.2.4 HOSTING

The project seamlessly operates with its frontend hosted on Firebase and backend, including .NET, database, and storage, on Azure services. This setup ensures robust performance and scalability, offering users a hassle-free experience without worrying about infrastructure complexities. By utilizing Firebase for frontend hosting and Azure services for backend support, the project excels in the digital landscape.

Procedure for hosting a backend on Azure

- Step 1. Fetch backend code from GitHub repository.
- Step 2. Set up environment variables and dependencies.
- Step 3. Deploy the latest commit.
- Step 4. Copy the backend API URL for integration with frontend.

Procedure for hosting a website on Firebase

- Step 1: Replace local backend URL with Azure URL in the website code.
- Step 2: Run 'npm run build' to build the app.
- Step 3: Upload the generated build files to Firebase.
- Step 4: Verify deployment and finalize setup.

Hosted Link:

<https://formulaonefanhub.web.app/>

Screenshots:

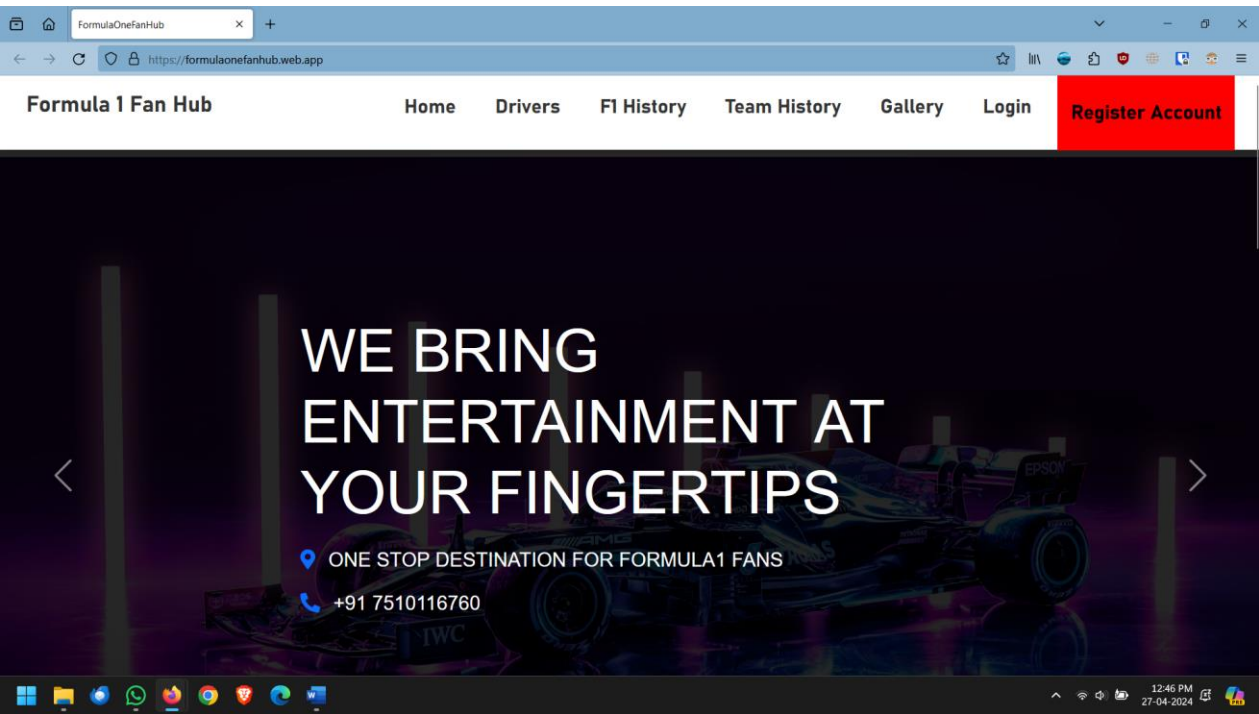


Fig: homepage

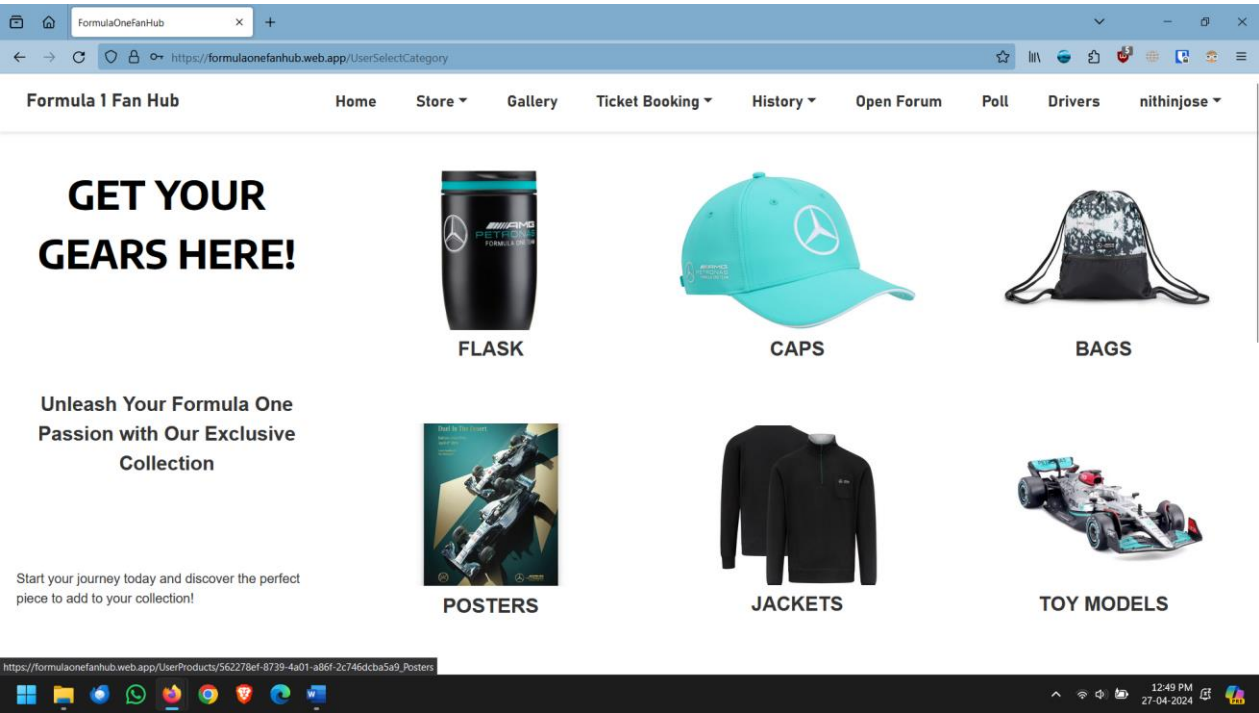


Fig: User Store Page

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

The Formula One Fan Hub revolutionizes fan engagement by seamlessly integrating historical exploration, dynamic discussions, and streamlined ticket booking. With modules catering to users, teams, and administrators, it offers a comprehensive platform for interaction and convenience. As the project evolves, the implementation of responsive design will further enhance the user experience, ensuring adaptability and informativeness. This hub stands as the premier destination, uniting fans with their passion for Formula One in an unparalleled manner.

7.2 FUTURE SCOPE

The Formula One Fan Hub project holds vast potential for future expansion and innovation. Integration of cutting-edge technologies like artificial intelligence and machine learning can elevate user experiences by providing personalized race recommendations and predictive insights. Enhancing community engagement features with live streaming events, virtual team meetups, and user-driven content creation can foster a vibrant and interactive Formula One community. Moreover, considering global expansion opportunities, incorporating multilingual support and strategic partnerships with race circuits, sponsors, and broadcasters can broaden the platform's international presence and appeal. Exploring the integration of virtual reality (VR) or augmented reality (AR) for immersive race experiences could further elevate the platform's uniqueness and competitiveness in the Formula One ecosystem. Overall, the future of the Formula One Fan Hub promises exciting avenues for the technological innovation, community building, and global expansion.

CHAPTER 8

BIBLIOGRAPHY

REFERENCES:

1. Jalote, P. (Year of publication). "Software Engineering: A Precise Approach." Publisher.
2. Shelly, G. B., & Rosenblatt, H. J. (2009). "System Analysis and Design." Publisher.
3. Schwaber, K., & Beedle, M. (2008). "Agile Software Development with Scrum." Pearson.
4. Pressman, R. S. (Year of publication). "Software Engineering." Publisher.
5. IEEE. (Year of publication). "IEEE Std 1016 Recommended Practice for Software Design Descriptions."

WEBSITES:

- <https://www.w3schools.com/>
- <https://chat.openai.com/>
- <https://react.dev/learn>
- <https://dotnet.microsoft.com/en-us/learn>
- <https://mui.com/material-ui/>

CHAPTER 9

APPENDIX

9.1 Sample Code

Index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <App />
);
reportWebVitals();
```

RazorPayService.cs

```
using System.Net.Http.Headers;
using System.Text;
using System.Text.Json;
using Azure.Core;
using FormulaOneFanHub.API.DTO;
namespace FormulaOneFanHub.API.Services
{
    public class RazorPayService
    {
        private readonly IConfiguration _configuration;
        public RazorPayService(IConfiguration configuration)
        {
            _configuration = configuration;
        }
        public RazorPayOrderResponseResult CreateOrder(decimal amount, string receipt, string notes)
        {
            var client = new HttpClient();
            var requestContent = BuildOrderRequest(amount, receipt, notes);
```

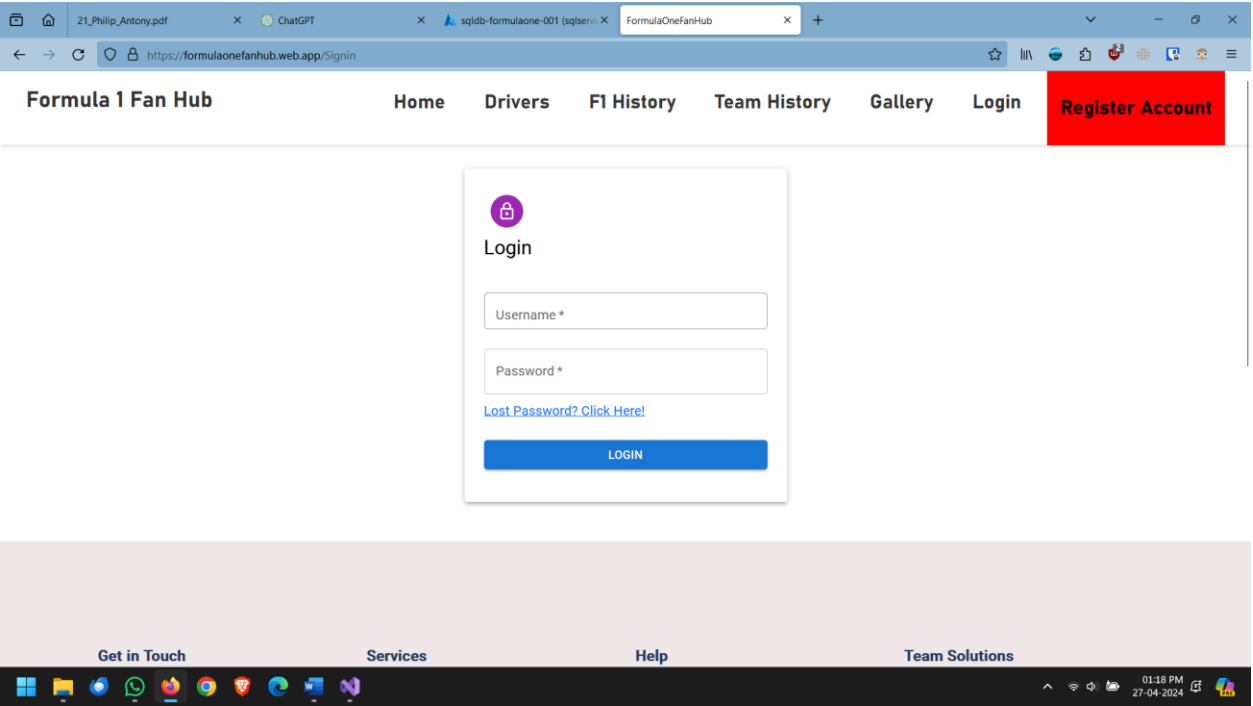
```
var razorKey = _configuration["RazorPay:Key"];
var razorKeySecret = _configuration["RazorPay:KeySecret"];
client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Basic",
    Convert.ToBase64String(Encoding.UTF8.GetBytes(razorKey + ":" +
razorKeySecret)));
var response = client.PostAsync("https://api.razorpay.com/v1/orders",
requestContent).GetAwaiter().GetResult();
var responseContent = response.Content.ReadAsStringAsync().GetAwaiter().GetResult();
RazorPayOrderResponseResult razorPayOrderResponse = null;
if (response.IsSuccessStatusCode)
{
    var razorPayOrderSuccessResponse =
JsonSerializer.Deserialize<RazorPayOrderSucessResponse>(responseContent);
    razorPayOrderResponse = new RazorPayOrderResponseResult
    {
        IsSuccess = true,
        SuccessResponse = razorPayOrderSuccessResponse
    };
}
else
{
    var razorPayOrderFailureResponse =
JsonSerializer.Deserialize<RazorPayOrderFailureResponse>(responseContent);
    razorPayOrderResponse = new RazorPayOrderResponseResult
    {
        IsSuccess = false,
        ErrorResponse = razorPayOrderFailureResponse
    };
}
return razorPayOrderResponse;
}

private HttpContent? BuildOrderRequest(decimal amount, string receipt, string notes)
{
    var requestContent = new StringContent(JsonSerializer.Serialize(new
```

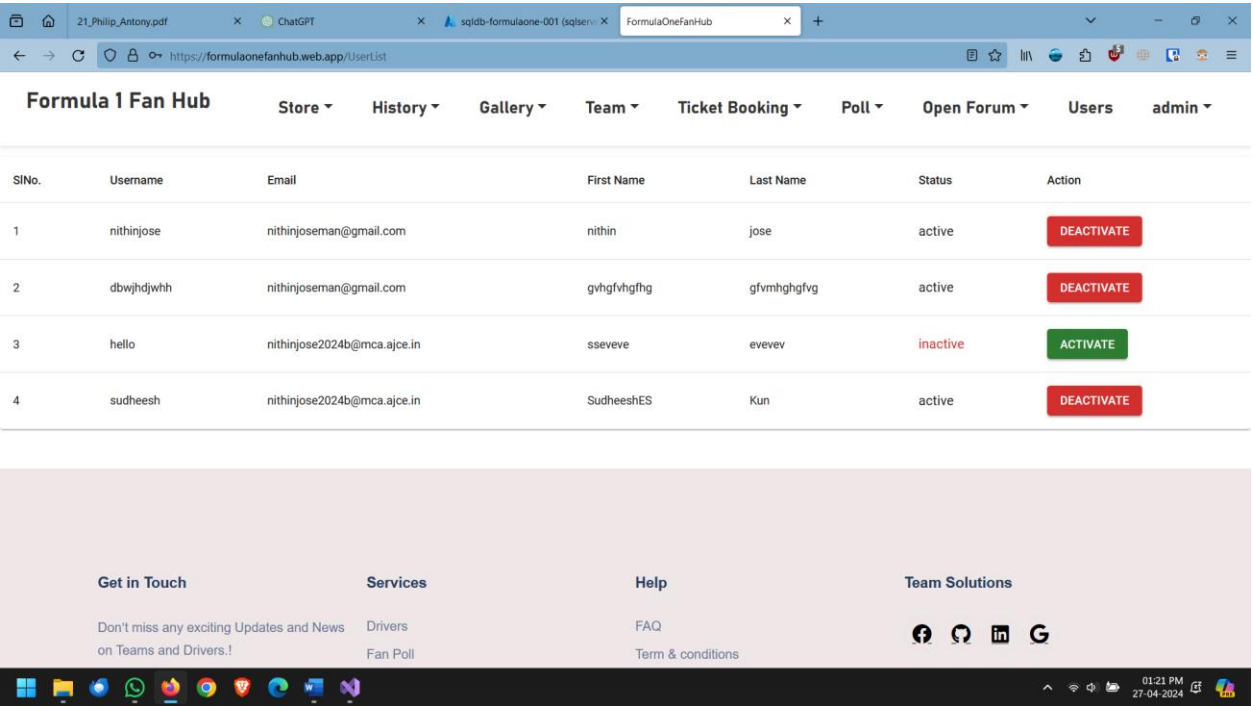
```
{
    amount = amount,
    currency = "INR",
    receipt = receipt,
    notes = new
    {
        key1 = notes,
        key2 = "value2"
    }
}), Encoding.UTF8, "application/json");
return requestContent;
}
}
}
```

9.2 Screen Shots

Login Page



View Users



Add New Delivery Company

Formula 1 Fan Hub

Store History Gallery Team Ticket Booking Poll Open Forum Users admin

Add Delivery Company

Email

Company Name

Contact Number

Address

UPLOAD IMAGE

SUBMIT

Listing the Products

Formula 1 Fan Hub

ferrari

Product Name	Description	Price	Stock Quantity	Product Category	Status	Actions
Ferrari Flask	the best in the market	2000	0	Flask	Active	<button>MANAGE STOCK</button> <button>EDIT</button>
Ferrari Cap	The best caps for a fashion trend	200	190	Caps	Active	<button>MANAGE STOCK</button> <button>EDIT</button>

HOME

VIEW PROFILE

Driver

Team History

OpenForum

Store

Get in Touch

Don't miss any exciting Updates and News on Teams and Drivers!

Services

Drivers

Fan Poll

Ticket Booking

Help

FAQ

Term & conditions

Security and Privacy

Team Solutions

f o i n g

Viewing Selling History

HOME

VIEW PROFILE

Driver

Team History

OpenForum

Store

Generate Selling Report

Start Date

01 / 03 / 2024

End Date

27 / 04 / 2024

GENERATE REPORT


DOWNLOAD PDF


Selling Report

Product Name	Quantity	Price Per Item	Total Price	Sold Date
Ferrari Flask	5	2000	10000	20/4/2024, 7:52:06 am
Ferrari Cap	1	200	200	20/4/2024, 10:15:04 am
Ferrari Flask	1	2000	2000	20/4/2024, 10:17:08 am
Ferrari Cap	1	200	200	20/4/2024, 10:17:08 am
Ferrari Cap	8	200	1600	20/4/2024, 10:19:41 am
Total Amount			14000	

Buying page

HomeStoreGalleryTicket BookingHistoryOpen ForumPollDriversnithinjose





FERRARI CAP

₹200

Stock Available: 190

Select Quantity:

Genuine product from:

Ferrari Sauber

Assured Benefits !

- Authenticity
- Quality Assurance
- Exclusive Access

Description

+

Add to Cart

Grab Now !

Shipping Address

Manjadiyil

Total Amount

₹200

Edit Address

Buy Now

Shipping Details

Free Shipping

Return Policy: 7 Days

Your item will be shipped on the next business day after your order and will reach you within 7 days.

An authorized delivery person will be contacting you using the contact details provided during checkout.

