FindMyNest

Mini Project Report

Submitted by

Anandhu Biju

Reg. No.: AJC19MCA-I013

In Partial fulfillment for the Award of the Degree of

INTEGRATED MASTER OF COMPUTER APPLICATIONS

(INMCA)

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 with ‘A’ grade. Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

2023-2024

DEPARTMENT OF COMPUTER APPLICATIONS

AMAL JYOTHI COLLEGE OF ENGINEERING

KANJIRAPPALLY

CERTIFICATE

This is to certify that the Project report, "**PROJECT NAME"** is the bona fide work of **STUDENT NAME (Regno: AJC00MCA-0000)** in partial fulfillment of the requirements for the award of the Degree of Integrated Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2023-24.

Ms. Sruthimol Kurian — **Internal Guide**

Ms. Meera Rose Mathew — **Coordinator**

Rev. Fr. Dr. Rubin Thottupurathu Jose

Head of the Department

DECLARATION

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

Date:23/10/2023 **ANANDHU BIJU**

KANJIRAPPALLY **Reg: AJC18MCA-I013**

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. It has been said that gratitude is the memory of the heart. I wish to express my sincere gratitude to our Manager **Rev. Fr. Dr. Mathew Paikatt** and Principal **Dr. Lillykutty Jacob** for providing good faculty for guidance.

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

ANANDHU BIJU

ABSTRACT

FindMyNest is a cutting-edge property management software that redefines the way users engage with the real estate market. It offers a user-friendly and intuitive platform for individuals seeking their ideal properties and property owners looking to manage their listings seamlessly. This innovative system prioritizes direct communication between property seekers and owners, facilitating property visits, negotiations, and secure transactions. Unlike traditional booking systems, FindMyNest doesn't focus on booking but emphasizes connecting users directly with property owners for personalized interactions.

The platform operates on a subscription-based model, allowing users to select from various subscription plans that cater to their specific needs, granting access to premium features and services. A standout feature is its robust electronic billing system, ensuring transparent and straightforward financial transactions. With 24/7 web-based accessibility, clients can engage with the platform anytime, ensuring unparalleled convenience in property management and exploration.

FindMyNest simplifies the property search process, offering comprehensive listings and in-depth property information. In summary, FindMyNest is a comprehensive property management solution, streamlining property transactions, enhancing transparency, and simplifying property-related activities for all stakeholders involved. It represents the future of property management, making property exploration and transactions as effortless as they should be. Whether you are a property seeker or owner, FindMyNest is the ultimate platform for all your real estate needs.

CONTENT

SL. NO	TOPIC	PAGE NO
1	INTRODUCTION	1
1.1	PROJECT OVERVIEW	2
1.2	PROJECT SPECIFICATION	3
2	SYSTEM STUDY	4
2.1	INTRODUCTION	5
2.2	EXISTING SYSTEM	5
2.3	DRAWBACKS OF EXISTING SYSTEM	7
2.4	PROPOSED SYSTEM	7
2.5	ADVANTAGES OF PROPOSED SYSTEM	8
3	REQUIREMENT ANALYSIS	9
3.1	FEASIBILITY STUDY	10
3.1.1	ECONOMICAL FEASIBILITY	10
3.1.2	TECHNICAL FEASIBILITY	11
3.1.3	BEHAVIORAL FEASIBILITY	11
3.1.4	FEASIBILITY STUDY QUESTIONNAIRE	12
3.2	SYSTEM SPECIFICATION	13
3.2.1	HARDWARE SPECIFICATION	13
3.2.2	SOFTWARE SPECIFICATION	13
3.3	SOFTWARE DESCRIPTION	14
3.3.1	PYTHON	14
3.3.2	SQLITE	14
4	SYSTEM DESIGN	15
4.1	INTRODUCTION	16
4.2	UML DIAGRAM	16
4.2.1	USE CASE DIAGRAM	17
4.2.2	SEQUENCE DIAGRAM	19
4.2.3	STATE CHART DIAGRAM	21
4.2.4	ACTIVITY DIAGRAM	23
4.2.5	CLASS DIAGRAM	25
4.2.6	OBJECT DIAGRAM	28
4.2.7	COMPONENT DIAGRAM	29

4.2.8	DEPLOYMENT DIAGRAM	30
4.3	USER INTERFACE DESIGN USING FIGMA	31
4.4	DATABASE DESIGN	33
5	SYSTEM TESTING	45
5.1	INTRODUCTION	46
5.2	TEST PLAN	47
5.2.1	UNIT TESTING	47
5.2.2	INTEGRATION TESTING	48
5.2.3	VALIDATION TESTING	48
5.2.4	USER ACCEPTANCE TESTING	48
5.2.5	AUTOMATION TESTING	49
5.2.6	SELENIUM TESTING	49
6	IMPLEMENTATION	58
6.1	INTRODUCTION	59
6.2	IMPLEMENTATION PROCEDURE	59
6.2.1	USER TRAINING	60
6.2.2	TRAINING ON APPLICATION SOFTWARE	60
6.2.3	SYSTEM MAINTENANCE	60
7	CONCLUSION & FUTURE SCOPE	61
7.1	CONCLUSION	62
7.2	FUTURE SCOPE	62
8	BIBLIOGRAPHY	64
9	APPENDIX	67
9.1	SAMPLE CODE	68
9.2	SCREEN SHOTS	76

List of Abbreviation

IDE	-	Integrated Development Environment
HTML	-	Hyper Text Markup Language.
CSS	-	Cascading Style Sheet
SQLite	-	Structured Query Language
UML	-	Unified Modeling Language
PYTHON	-	Hypertext Preprocessor
DJANGO	-	Frame Work

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

"FindMyNest" is an online property management software designed to revolutionize the real estate market. It works as a complete platform for both owners and owners. This application allows users to easily register online, search for ideal properties and make endless real estate transactions. The main goal of the project is to automate and simplify real estate management processes and promote direct and transparent communication between claimants and real estate owners. There are three main user roles in the system: administrators, property applicants and owners. Administrators can manage property information, make changes and delete records if necessary. Users can create accounts, manage their profiles and view a wide range of real estate information. They can safely add the property to their favorites or cart and continue trading real estate within the platform. "FindMyNest" stands out as a user-friendly and visually appealing platform that prioritizes convenience and business growth. An interactive web interface allows users to search for properties, view detailed property descriptions and add items to wish lists, simplifying the property selection process. Sales and purchases of real estate occur electronically in real time, which increases efficiency and transparency. Reports are generated quickly, reducing manual work, and user interactions and property lists are regularly analyzed to facilitate decision-making processes. The system includes chatbot technology that provides fast customer support service that makes it easy for users to get quick help and information. In addition, data visualization technology is used to better understand real estate, and Google Translator is integrated for multilingual support.

Technologically, "FindMyNest" is developed using HTML, CSS and JavaScript for the user interface, providing a visually appealing and responsive user experience. Python works as an efficient back-end software to manage user accounts, property information and transactions. A relational database management system and Sqlite is used to efficiently store and manage data, ensuring data integrity and accessibility.

In short, "FindMyNest" is a state-of-the-art real estate management solution that simplifies real estate transactions, facilitates direct communication, and improves user experience in the real estate market. It is a technological advancement that represents the future of property management.

1.2 PROJECT SPECIFICATION

The proposed system, "FindMyNest," is a web-based platform designed to streamline property management and real estate transactions. This system caters to the needs of property seekers, property owners, and administrators, making property exploration, listing management, and transactions efficient and user-friendly. Below are the project specifications

The system includes 2 modules:

Admin Module:

The administrator serves as the pivotal figure within the system, holding the highest level of authority. Only the admin has access to the exclusive admin panel, as they essentially own and operate the platform. This role boasts comprehensive control over the entire system, dictating its operations. Admins are required to log in to the system and are privy to all the vital information concerning registered users, property listings, and transactions. The administrator's responsibilities encompass managing property listings, overseeing property transactions, and maintaining the integrity of the system. Furthermore, admins are entrusted with the pivotal task of handling payments, ensuring secure and smooth transactions.

User Module:

The client module is engineered to cater to a diverse range of users who don the dual hat of property seekers and property owners. This versatile module accommodates users' interaction with the system. Clients, when acting as property seekers, are presented with an intuitive platform. They can peruse an extensive catalog of properties and conveniently engage with the system by selecting properties that pique their interest. Each user is bestowed with the opportunity to establish a unique profile, complete with options for password updates. For those keen to explore the properties listed within the system, new user registration is a requisite. Following successful registration, users can seamlessly log in using their chosen credentials. Once logged in, clients can access a comprehensive list of properties, each featuring detailed descriptions, images, and pricing. Clients are empowered to add properties of their choice to their cart and wishlist. Upon the completion of property selection, clients can seamlessly initiate property transactions, culminating in secure payments. The system also extends the courtesy of allowing clients to furnish reviews and ratings for properties they have engaged with, enhancing user engagement and feedback.

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

System analysis is a necessary step in the development of any software platform, and "FindMyNest" is no exception. This complex process requires careful data collection and review, with the ultimate goal of identifying problems and developing strategies to improve the system. Collaboration between system users and developers is essential during this problem solving journey. Every system development initiative begins with a thorough system analysis, a detailed examination of the existing infrastructure.

As for "FindMyNest", the current system is undergoing rigorous research. This step involves comprehensively understanding the challenges, isolating the relevant variables, dividing the system components and choosing the most feasible course of action. Systematic analysis requires an in-depth examination of procedures, facilitated by a variety of methods, including surveys and interviews. The information collected by these tools is then carefully checked to form a complete picture of the current system. With a deep understanding of the existing system, problem areas emerge and it is at this point that the designer assumes the role of a problem solver. Recommendations emerge as valuable aspects of this phase as possible ways to address the challenges facing the real estate industry. These recommendations are carefully checked, weighed against the current system, and the most suitable is selected for further processing.

"FindMyNest" remains committed to user-centricity, whereby users have the prerogative to accept or reject proposed solutions. Their input is invaluable and the proposal will be revised and modified based on their feedback. The process of data collection and analysis for subsequent system studies is consistent with the preliminary study, which is the cornerstone of significant system development. In the collaborative spirit of "FindMyNest", system users and developers work closely together to meet the challenges facing the real estate industry. A nuanced understanding of how the system works is achieved through various cost-effectiveness studies. This review, in turn, selects the most effective techniques for systematic research and analysis, reinforcing FindMyNest's commitment to continuous improvement and excellence in property management.

2.2 EXISTING SYSTEM

In this prevailing manual system, the traditional approach to real estate transactions typically involves physically visiting the property, dealing with various brokers, and completing transactions in person. Like purchasing paperwork manually, this traditional system can be cumbersome and time-consuming for property seekers and owners alike. In this dominant manual system, potential property seekers often find themselves moving from property to property in an attempt to find their ideal property. This process can be physically demanding and time consuming, reflecting the challenges of the paper business. Crowded property viewings and long, drawn-out processes are not uncommon, especially during times of high demand such as the school season. Just like the limitations faced by those without access to a quality writing market, not all places have the privilege of a well-organized and efficient real estate market. This situation calls for a change in property management, paving the way to a smooth and user-friendly platform like "FindMyNest".

2.2.1 NATURAL SYSTEM STUDIED

The "FindMyNest" project is rooted in the analysis of the existing natural system within the real estate and property management industry. This examination encompasses the dynamics of the real estate market, property transactions, property listings, and user interactions. It involves a comprehensive understanding of factors such as supply and demand, property pricing mechanisms, and challenges like the need for physical property visits and the presence of intermediaries. This study provides essential insights that serve as the foundation for "FindMyNest," a platform designed to streamline and enhance property management processes, making them more efficient, user-centric, and responsive to the evolving real estate landscap

2.2.2 DESIGNED SYSTEM STUDIED

This system is designed to address the challenges identified in the studied natural system. Key components of this proposed system include an advanced property management platform that allows owners to efficiently create, update and manage property directories. It offers real estate seekers an intuitive user interface that allows them to view listings in detail, add properties to their favorites or carts, and complete real estate transactions seamlessly. A user-centric approach with user review and rating functions is essential to increase engagement and provide valuable insights. The administration module allows administrators to control property listings, user accounts and financial transactions. In addition, user feedback mechanisms encourage review and modification of proposals, ensuring system development meets user requirements. The use of modern technology, such as data visualization and integration with tools such as Google Translate, further improves the user experience. "FindMyNest" is an innovative solution that aims to increase transparency, convenience and user satisfaction in property management and transactions. It provides a user-friendly and efficient platform for real estate research and transactions.

2.3 DRAWBACKS OF EXISTING SYSTEM

- Lack of Specialized Real Estate Platforms
- Limited Property Information
- Time-Consuming Property Searches
- Inaccessibility of Quality Properties
- Storage Facility Limitations
- Limited Payment Options
- User-Friendliness Challenges

2.4 PROPOSED SYSTEM

The proposed "FindMyNest" system is designed to address the limitations of the current property management system by focusing on providing an enhanced, user-friendly and growth-oriented experience to property seekers and owners. Important features of the proposed system include the introduction of modern online payment methods to facilitate secure and convenient real estate transactions. The system covers three main categories of users: managers, customers and property owners, creating a smooth and efficient process.

The main purpose of this software is to automate the recording and generation of reports, which greatly reduces the manual and time required to maintain real estate. Centralized data storage ensures simultaneous availability to all system users, which increases collaboration and efficiency. The system's intuitive user interface minimizes the need for special training, which simplifies historical data management and general operation. The proposed system simplifies routine tasks, which increases the efficiency of real estate management. Data storage of online real estate sales and transactions is easy, and this data can be seamlessly stored in structured databases.

In particular, the proposed system eliminates the need for property seekers to physically visit the property to view it. Instead, they can use the platform to search, explore and select the features they want. Owners acting as system administrators can complete property and transaction requests through an efficient and secure online system. User-centric design extends to developing a user-friendly website that allows real estate seekers to easily navigate real estate listings and complete transactions from the comfort of their own homes using mobile devices or computers. This online platform ensures 24/7 availability, making property management and mapping very accessible. The "FindMyNest" system is designed to improve the user experience, especially during times of increased demand for real estate. This eliminates the need for long waiting times or complicated property viewings, providing a convenient and efficient solution. Delivering property information directly to the user's device streamlines the property search process. "FindMyNest" also provides a diverse property listing that ensures a quality real estate experience for both the property seeker and the property owner.

2.5 ADVANTAGES OF PROPOSED SYSTEM

- Accessibility and User-Friendliness
- Comprehensive Property Listings
- Cost and Time Efficiency
- Enhanced User Support
- Simple Registration Process
- Multiple Payment Options
- Captcha in login page
- Chatbot for personal assistants
- Time and Travel Savings
- User Feedback and Ratings

CHAPTER 3
REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

This feasibility study is a vital step in determining the FindMyNest project's viability in terms of resources, time, and organizational goals. It aids in assessing prospective long-term advantages and opportunities. By carrying out this study, the developer will be able to determine whether the suggested system is workable and deserving of more investigation. The feasibility study looks at the system's effects on the business, its capacity to satisfy customer needs, and its resource-use effectiveness.

A system is of no real value to a corporation if it does not serve its goals. Despite the fact that this may seem obvious, many organizations create systems that do not support their goals, either because they lack a clear statement of these goals, because they fail to specify the system's business requirements, or because other organizational or political factors have an impact on the procurement of the system.The company may have trouble managing properties effectively if the FindMyNest system is not installed. Customers would have a difficult time locating properties, and transactions would be cumbersome and prone to mistakes. Customers may become unsatisfied as a result, and business growth may be slower than that of rivals who use online methods.

The manual labour and paperwork required by the present property management procedures lead to errors and delays. By offering a user-friendly online platform for property listings and communication between clients and brokers, a new system like FindMyNest can help. Time would be saved, accuracy would increase, and customers would be happy.By enhancing the company's competitiveness, raising customer happiness, and reducing time and expense spent on administrative activities, FindMyNest will directly benefit the company

3.1.1 Economical Feasibility

FindMyNest’s economic feasibility is a key factor in determining whether your project will be financially viable for your organization. It’s a comprehensive analysis of the costs associated with software development and the expected long-term returns and benefits that your system can bring to your organization. For example, you’ll need to consider the resources, time and effort needed for software development. You’ll also need to consider the cost of conducting a complete software investigation, including requirements elicitation, analysis, etc. The economic feasibility also includes hardware and software infrastructure costs, such as servers and licenses, your development team’s assembly and maintenance, training, etc. By thoroughly analyzing these costs and comparing them to the expected benefits, you can determine whether your FindMyNest project will bring you significant and sustainable returns in the long run.

The system was entirely developed using open-source software, and all necessary resources were readily available, which further reduced the development cost. This indicates that the system is economically feasible for development. Users only need to pay for internet connectivity in order to access the applications

3.1.2 Technical Feasibility

The technical feasibility of FindMyNest revolves around evaluating whether the project can be successfully implemented from a technological perspective. A key factor is the availability of skilled and proficient team members capable of handling tasks such as web development, database management, user interface design, and system integration. Their expertise is crucial in ensuring the system's smooth and effective development. Additionally, the choice of technology stack is of paramount importance. Opting for stable and established technologies, including well-known programming languages, frameworks, and databases, ensures reliability, security, and easier maintenance of the system.
Technical feasibility is important to ensure that the proposed system can be developed using the available resources and technology within the allocated time and budget. In the proposed system, the frontend is developed using HTML, CSS, and JavaScript. These technologies are widely used and have a large developer community, making it easier to find solutions and resolve issues. The most popular and well-established Python programming language is used in backend. The mongoDB tool is used for the administration over the web. The system is self-explanting and does not need any entire sophisticated training. Overall, the use of these technologies ensures technical feasibility and efficient development of the proposed system

3.1.3 Behavioral Feasibility

The feasibility of the "FindMyNest" project depends on its ability to address important behavioral aspects and ensure that the system meets the needs of its users effectively and without harm. The following questions are important to assess the feasibility of the behavior:

1. **Sufficient User Support**: A key aspect of the proposed system is its ability to provide strong user support. This includes prompt resolution of user questions, concerns and issues. The "FindMyNest" platform is designed to be user-centric and offers easy-to-use customer support channels to assist users. In particular, the inclusion of a chat and user-friendly interface ensures that users can navigate and use the system without special training.

2. **User Welfare:** The system design aims to improve the overall welfare of users, whether they are property seekers or property owners. The purpose of "FindMyNest" is to streamline real estate transactions, improve real estate management, and reduce the time and effort of real estate related activities. The well-being of users is central to the goals of the project, and no adverse effects are expected.

After careful consideration of these behavioral aspects, it was concluded that the "FindMyNest" project is behaviorally viable. The system prioritizes user-friendliness and support, making it accessible and useful for everyone involved. Users can use the "FindMyNest" platform without extensive training, which promotes a positive user experience and overall well-being.

3.1.4 Feasibility Study Questionnaire

1. How is property information collected and stored in your real estate farm?

   We maintain physical property files containingproperty details, photographs, and relevant documents.

2. How do potential buyers or tenants inquire about properties in your real estate farm?

   They usually visit our office, call us, or send inquiries through emails or letters.

3. What is the process for property viewings and visits?

   Interested clients schedule appointments, and our agents accompany them to the properties for viewings.

4. How are property transactions processed in the offline system?

   We handle transactions through physical paperwork, including agreements and payment receipts.

5. How do you manage client information and communication?

   We maintain client records in paper files and use phone calls and face-to-face meetings for communication.

6. What challenges do you face in the offline system's property management process?

   Limited reach to potential clients, manual paperwork, and longer transaction times are some challenges.

7. How do you handle property advertisements and marketing offline?

   We use printed brochures, banners, and advertisements in local newspapers.

8. How do you keep track of property availability and occupancy status?

   We manually update a property status board in our office based on available properties.

9. Do you receive feedback from clients about your services and property offerings?

   Yes, we often receive feedback during face-to-face interactions and through phone calls.

10. Do you receive feedback from clients about your services and property offerings?

    We are considering adopting an online platform to reach a wider audience and streamline our processes.

3.1 SYSTEM SPECIFICATION

3.2.1 Hardware Specification

Processor	-	Intel i5
RAM	-	RAM 4GB
Hard disk	-	1TB

3.2.2 Software Specification

Front End	-	HTML, CSS, JS, Bootstrap
Back End	-	Python
Database	-	SQlite
Client on PC	-	Windows 7 and above.
Technologies used	-	JS, HTML5, AJAX, J Query, Python, CSS, Google reCAPTCHA, Chatbot, Machine Learning

3.3 SOFTWARE DESCRIPTION

3.3.1 HTML

HTML, that is Hypertext Markup Language, is the basic language for creating web pages. It uses a set of essential elements to give structure and meaning to web content, allowing content to be organized in logical sections and include multimedia elements such as images and videos. In the "FindMyNest" project, HTML5 is used in the design of the homepage, and HTML validations are applied to ensure the quality of the code.

3.3.2 CSS

Cascading Style Sheets (CSS) plays a pivotal role in shaping the layout and visual styling of web content. While HTML defines content structure and meaning, CSS allows modification of various visual aspects, including fonts, colors, sizes, and spacing. It also facilitates the creation of visual effects, animations, and decorative features. The "FindMyNest" system incorporates CSS3 to style the website, introducing animations to enhance its visual appeal.

3.3.3 Python

Python, often referred to as "PHP: Hypertext Preprocessor," is a versatile server-side scripting language extensively employed for both general-purpose programming and web development tasks. It is utilized by a vast number of web servers and web pages and is characterized by its seamless integration with HTML source files, command-line interface convenience, and cross-platform compatibility. Python is integral to "FindMyNest" for executing server-side tasks and facilitating web development.

3.3.4 SQLite:

SQLite is a relational database management system (RDBMS) utilized for data storage, organization, and retrieval. As an open-source software, SQLite offers a wide range of applications, including use with various programming languages such as Java, PHP, and C++. "FindMyNest" leverages SQLite to manage and store property-related data, providing scalability, speed, and reliability. It further enhances data protection and accessibility through features like replication, high availability, and encryption.

CHAPTER 4
SYSTEM DESIGN

4.1 INTRODUCTION

The development process of any intended system or product begins with design. Creative process is planning the secret to an effective system is proper planning. The process uses different methods and concepts to define a process or system in sufficient detail to enable it the physical implementation is called the "design". One way to describe it is as a work process various methods and concepts to define a device, process or system in sufficient detail enable its physical realization. Regardless of the development paradigm used, software engineering forms the technical core of software engineering. Architectural the detail required to build a system or product is developed through system design. This the program also refined all the efficiency, performance, and levels of detail, as with any systematic technique. A user-oriented document is converted into a document for developers or database workers during the design phase. The two phases of system design are logical design and physical design.

4.2 UML DIAGRAM

A standardized language known as UML used for design purposes, The development, visualization and description of software systems are created by Object management group (OMG). The initial draft of the UML 1.0 standard was published in January 1997 and since then it has been widely used as a visual language for description and development. software systems. Unlike programming languages such as Java, C and COBOL, UML is used instead of diagrams to describe the flow and interaction of system components writing code Although UML is most often used to design software systems, it can also be applied to non-software systems such as the process flow of a manufacturing facility. With use UML diagrams in easy-to-use tools can be translated into code for multiple computers Language (languages. Thanks to successful standardization, OMG recognized UML as a standard and a close relationship with object system analysis and design.

The nine different UML diagrams are:

- Class Diagram
- Object Diagram
- Use Case Diagram
- Sequence Diagram
- Activity Diagram
- State chart Diagram
- Deployment Diagram
- Component Diagram

4.2.1 USE CASE DIAGRAM

A use case diagram is a graphical representation that illustrates the relationships between them functions of various system components. It is a method of defining, outlining and organizing system requirements, such as the need for a website to order goods and services posting Use case diagrams are one of the tools of the popular modeling language UML describes the structure and operation of systems and objects seen in the real world. Tasks which constitute the objectives of the system can range from defining general needs to acceptance hardware design, testing and troubleshooting software currently in development; create online help resources and perform tasks related to customer service. For example, in the context of a business scenario, use cases might include responding to a customer service requests, product order processing, product listing maintenance and tracking payment processing. Typically, a use case diagram has four important parts

- A wall that separates the system being studied from its environment.
- Actors are often participants in the system, identified by their role.
- Actors inside and outside the system implement use cases, which are expert roles.
- Discussions and interactions between participants and use cases

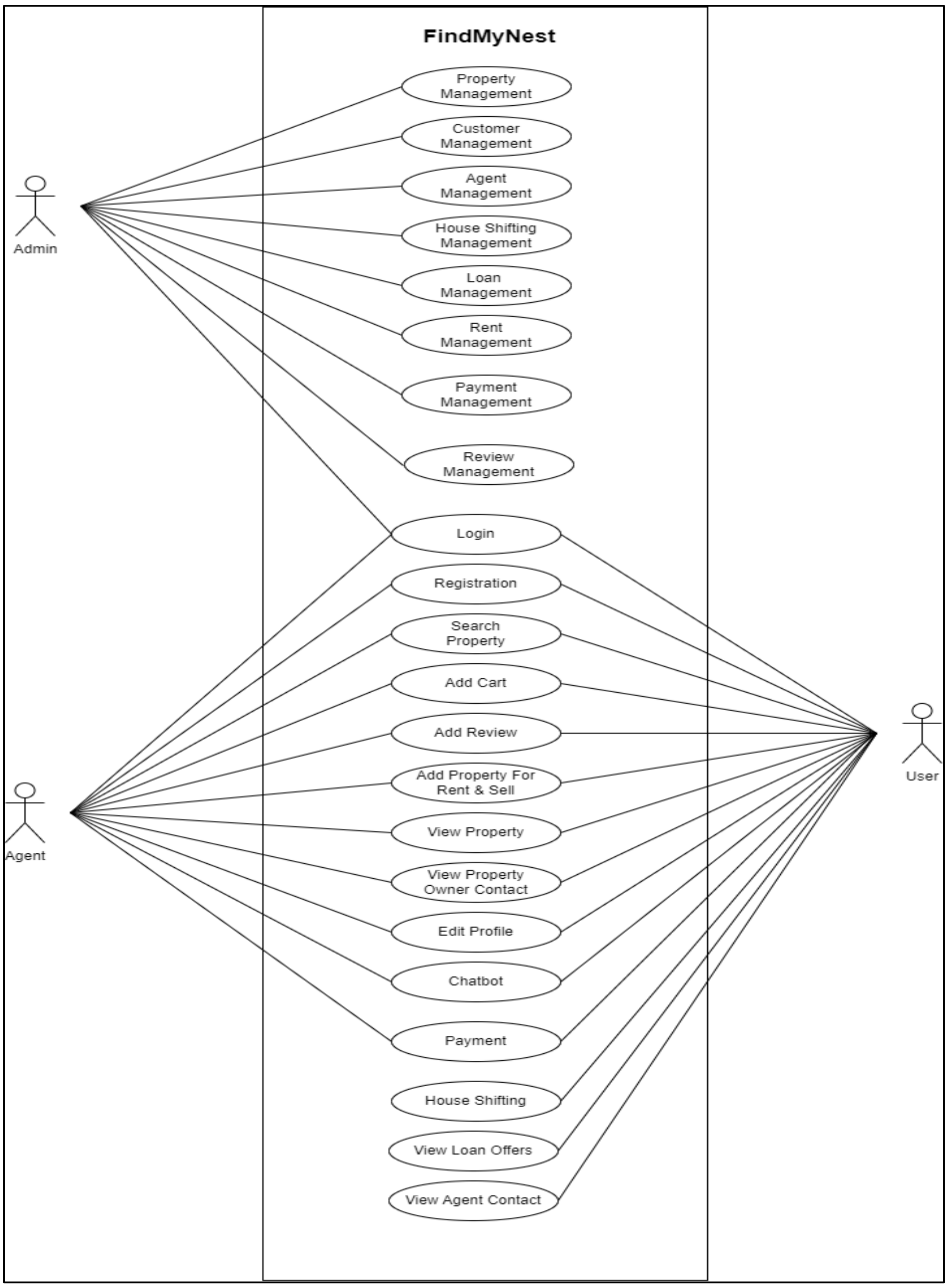


Figure 1: Use Case Diagram of Admin, User and Agent

4.2.1 SEQUENCE DIAGRAM

A sequence diagram is a powerful visual tool that effectively portrays the interaction between various objects or components in a precise sequence or order. This diagram is also commonly known as an event diagram or event scenario. The primary objective of a sequence diagram is to provide a clear and detailed representation of how different objects within a system collaborate and execute their functions in a specific order. Software developers, as well as business professionals, often leverage sequence diagrams to document, analyze, and comprehend the intricate dynamics and operational requirements of both newly designed systems and pre-existing ones. These diagrams play a vital role in enhancing communication, facilitating system design, and ensuring a comprehensive understanding of system behavior and interaction

Sequence diagrams in UML are a powerful tool for visualizing the interactions between objects in a system. To create effective sequence diagrams, it's essential to understand the key notations and elements used:

I. **Actors:** Actors represent roles or external entities that interact with the system being modeled. These can include human users, other systems, or any external subjects. Actors are always positioned outside the system's scope in the UML diagram. In sequence diagrams, actors are depicted using a stick person notation. Multiple actors can be present in a single sequence diagram, each representing a distinct role or entity.

II. **Lifelines:** Lifelines are named elements that signify individual participants in the sequence diagram. Each instance involved in the interaction is represented by a lifeline. Lifelines are typically located at the top of the sequence diagram, and they serve as a visual reference for the objects or participants in the interaction.

III. **Messages:** Messages in a sequence diagram represent communication between objects. Messages are presented in a sequential order on the lifelines, demonstrating the flow of interactions. Messages are typically represented using arrow notations. They are at the core of a sequence diagram and play a crucial role in illustrating how objects collaborate. Messages can fall into several categories, including:

- **Synchronous messages**: These messages denote interactions where the sender waits for a response from the receiver.
- **Asynchronous messages:** These messages indicate interactions where the sender does not wait for an immediate response.

- **Create message**: Used to illustrate the creation of a new object.
- **Delete message:** Indicates the destruction or deletion of an object.
- **Self-Message:** Represents interactions within the same object.
- **Reply Message:** Depicts responses or replies to messages.
- **Found Message:** Used to show a message found later during the interaction.
- **Lost Message:** Demonstrates a message that was lost during the interaction.

IV. **Guards:** Guards are used in UML sequence diagrams to model conditions or constraints. They come into play when there's a need to control the flow of messages based on specific conditions. Guards are crucial in conveying any constraints or rules governing the system or a particular process. They provide valuable information to software developers about the conditions that must be met for certain interactions to occur.

Uses of sequence diagrams –

- Used to model and visualize the logic behind a sophisticated function, operation or procedure.
- They are also used to show details of UML use case diagrams.
- Used to understand the detailed functionality of current or futuresystems.
- Visualize how messages and tasks move between objects or components in a system

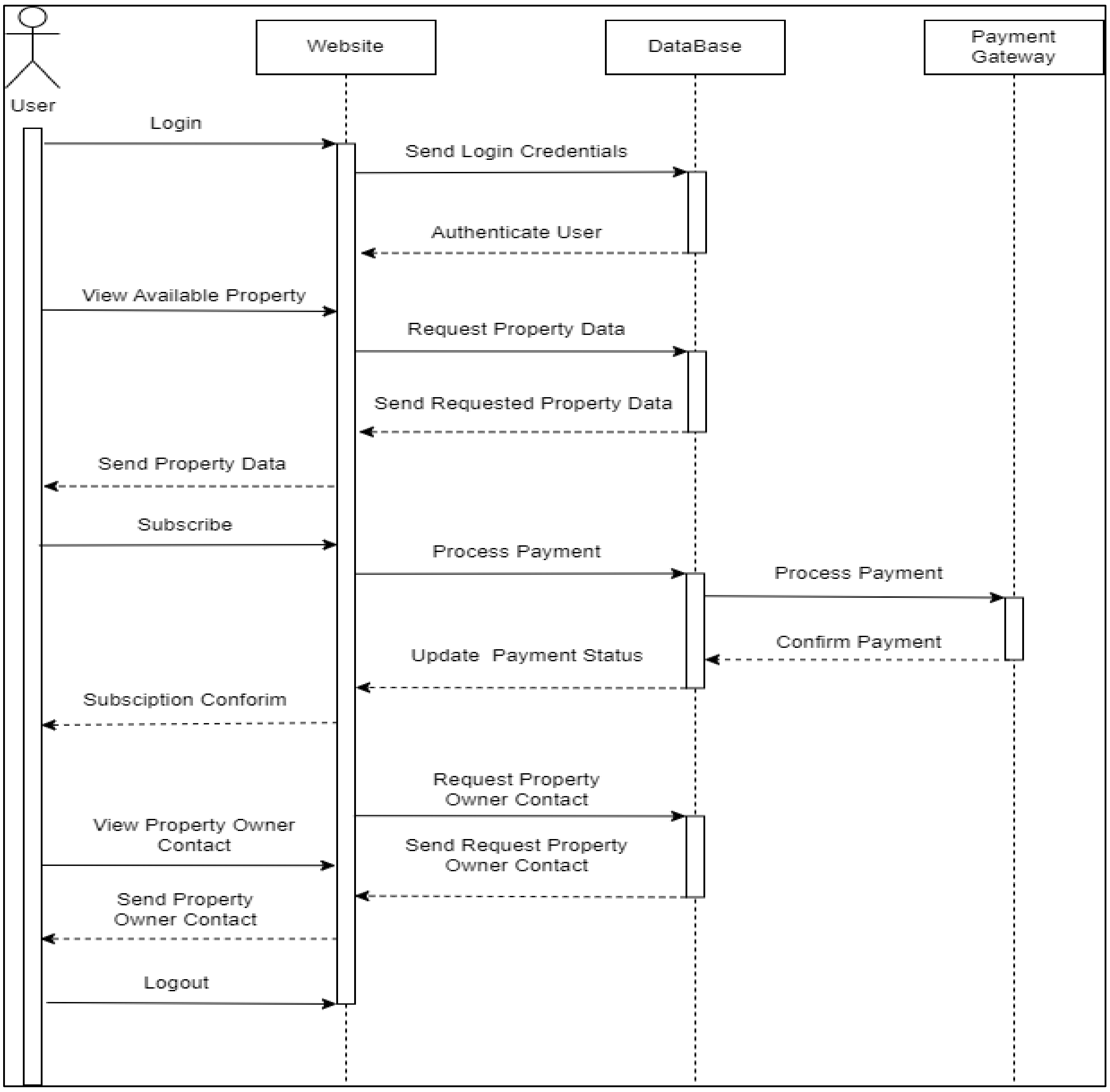


Figure 2: Sequence Diagram of FindMyNest

4.2.2 State Chart Diagram

A state machine diagram, also known as a state diagram or state transition diagram, is a visual representation that shows the sequence of states that an object passes through in a system. It provides a comprehensive view of the behavioral aspects of a software system and can be applied to model the behavior of a class, subsystem, package, or the entire system itself. State machine diagrams are particularly effective in modeling the interaction and cooperation between external entities and the system, especially in event-driven systems where object state plays a critical role. These diagrams define different states for a system component, and each object or component is associated with a specific state.

Notations of a State Machine Diagram:
The notations used in a State Machine Diagram include the following elements:

- **Initial State:** This element represents the starting point of the system's state machine and is depicted by a black filled circle.
- **Final State:** The final state signifies the endpoint of the system's state machine and is illustrated by a filled circle enclosed within another circle.
- **Decision Box:** This diamond-shaped element symbolizes decision points in the state machine, where choices are made based on evaluated guards or conditions.
- **Transition:** Transitions in a state machine diagram signify a change of control from one state to another due to the occurrence of a specific event. Transitions are represented by arrows labeled with the event that triggers the change.
- **State Box:** The state box represents the conditions or circumstances of a particular object or component within a class at a specific moment in time. It is depicted as a rectangle with rounded corners.

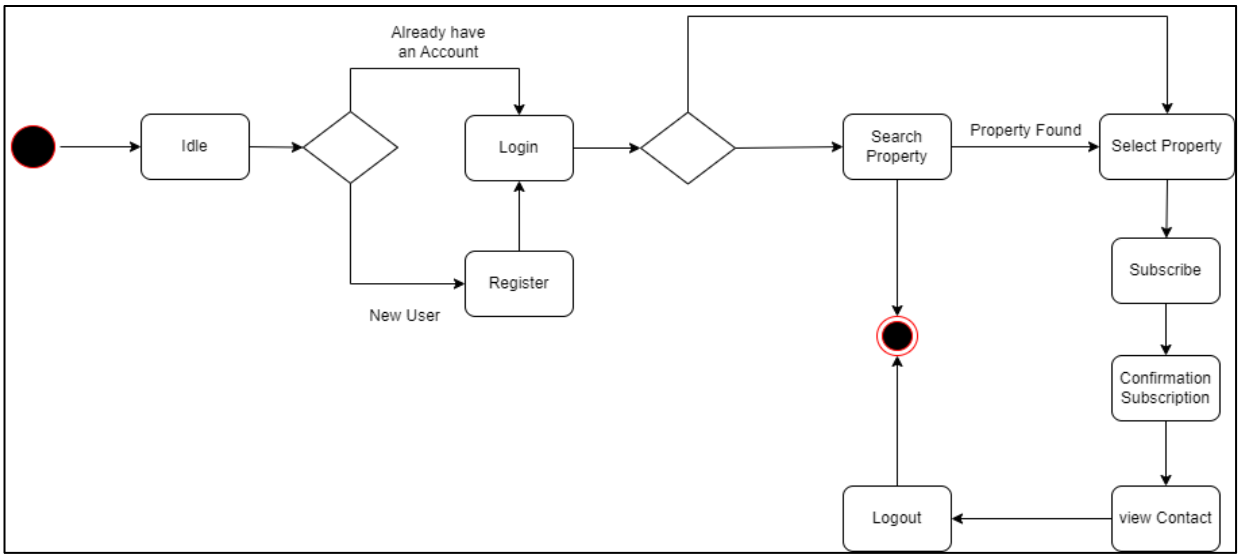


Figure 3: State Chart Diagram of FindMyNest

4.2.2 Activity Diagram

An Activity Diagram in UML serves as a visual tool to depict the flow of control within a system, with a focus on the workflow between activities rather than the detailed implementation. It is especially useful for illustrating the sequence of activities, both concurrent and sequential, and emphasizes the order and conditions that dictate the flow. Activity Diagrams are often likened to object-oriented flowcharts and are composed of a series of actions or operations to model the behavioral aspects of a system.

Components of an Activity Diagram:

a) **Initial Node:** This node marks the starting point of the activity diagram, indicating where the process begins.

b) **Activity States:** Activity states represent the various activities involved in a process and are typically depicted by rounded rectangles.

c) **Decision Nodes:** Decision nodes serve as points in the activity diagram where the flow can diverge into different paths based on specific conditions.

d) **Fork Nodes:** Fork nodes indicate parallel processing of activities, allowing multiple actions to occur concurrently.

e) **Join Nodes:** Join nodes are used to merge parallel processing flows back into a single flow, consolidating the outcomes of parallel activities.

f) **Final Nodes:** Final nodes signal the conclusion of an activity diagram, marking the endpoint of the control and object flows.

g) **Control Flows:** Control flows are represented by arrows connecting various nodes and depict the order in which activities are performed.

h) **Action States:** Action states represent specific actions or operations performed during an activity and are represented by rectangles with rounded edges.

i) **Swimlanes:** Swimlanes are used to group activities according to their responsible parties or departments, simplifying the visualization of the flow of work.

Notation of an Activity Diagram:

The notation used in an Activity Diagram includes the following elements:

1. **Initial State:** This denotes the initial stage or starting point of a sequence of actions.
2. **Final State:** The final state signifies the point where all control flows and object flows reach their conclusion.

3. **Decision Box:** Decision boxes ensure that the control or object flow follows only one path based on evaluated conditions.

4. **Action Box:** Action boxes represent a set of actions that need to be performed during the activity and are typically depicted as rectangles.

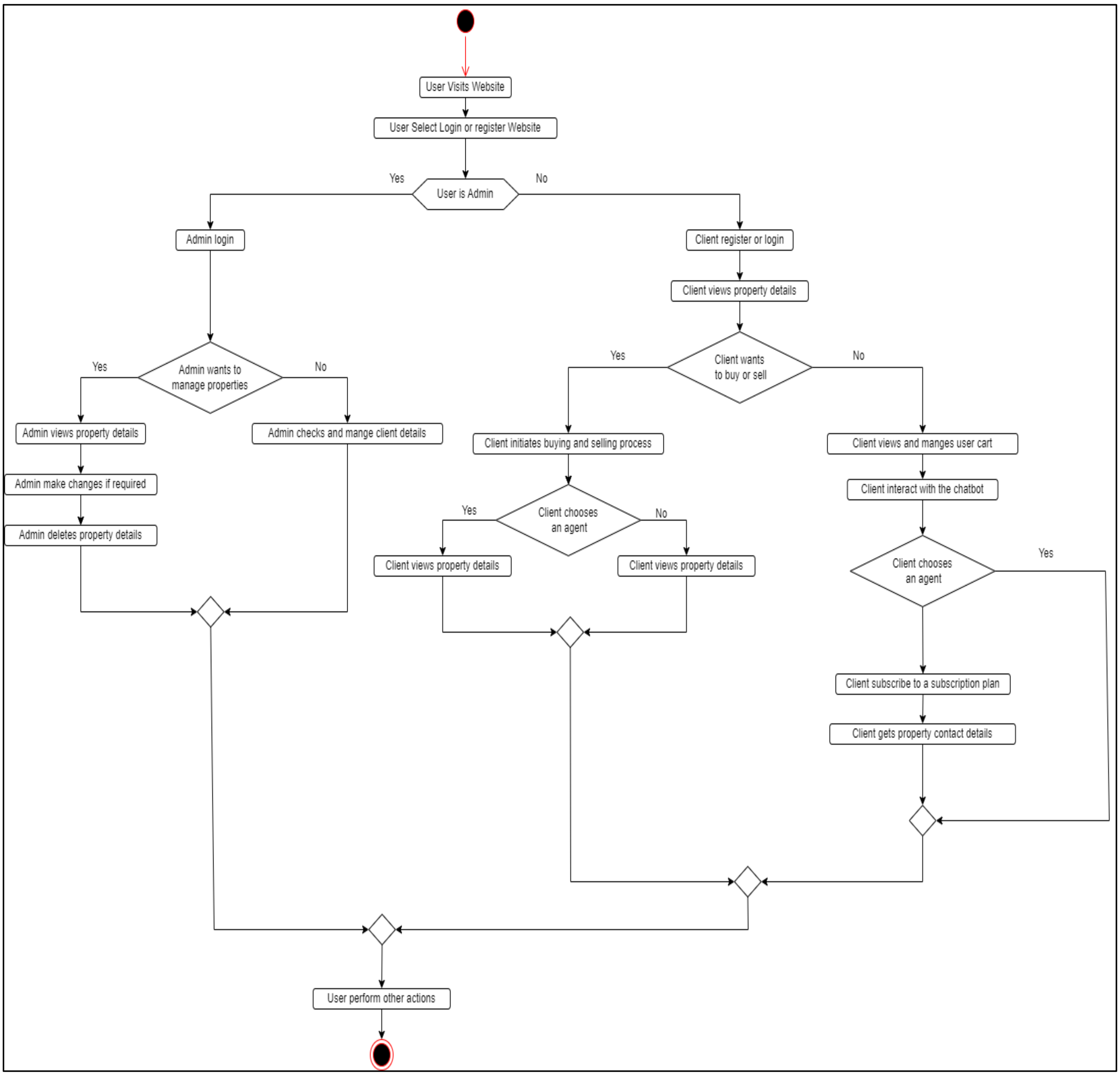

User Visits Website
User Select Login or register Website
Yes
User is Admin
No
Admin login
Client register or login
Client views property details
Yes
Admin wants to manage properties
No
Yes
Client wants to buy or sell
No
Admin views property details
Admin checks and mange client details
Client initiates buying and selling process
Client views and manges user cart
Admin make changes if required
Client interact with the chatbot
Admin deletes property details
Yes
Client chooses an agent
No
Client views property details
Client views property details
Client chooses an agent
Yes
Client subscribe to a subscription plan
Client gets property contact details
User perform other actions

Figure 4: Activity diagram of FindMyNest

4.2.3 Class Diagram

The static view of the program is shown in the class diagram. The many product types of the system contains and possible connections between them. Despite the fact that the class can too derived from other classes, a class is only as good as its objects. Class diagrams are used to describe, describe and document the various parts of the system. They are also used to create executable software code. A block diagram gives a general picture of the architecture of a software system by showing system classes, relationships, properties and functions. to support by creating programs for a specific domain, it helps to gather information about the class names, properties and characteristics

A Class Diagram in UML is a visual representation of the structure and organization of classes in a system, emphasizing their attributes, operations, and relationships. It consists of three main sections:

1. **Upper Section:** This section encompasses the name of the class, which represents a category of objects sharing the same attributes, operations, relationships, and semantics. When representing a class, certain rules should be followed:

    - Capitalize the initial letter of the class name.
    - Place the class name in the center of the upper section.
    - The class name must be written in bold format.
    - If the class is abstract, its name should be written in italics format.

2. **Middle Section:** The middle section of the class diagram deals with the attributes of the class. Attributes describe the qualities or characteristics of the class. Attributes in a class diagram have specific characteristics, such as:

    - Attributes are written along with their visibility factors, which can be public (+), private (-), protected (#), or package (~).
    - Visibility factors illustrate the accessibility of an attribute within the class.
    - Attributes should have meaningful names that explain their usage within the class.

3. **Lower Section:** The lower section of the class diagram contains methods or operations that define how the class interacts with data or performs specific actions. Methods are represented as a list, with each method written on a single line.

Relationships:

In UML, relationships between classes are classified into three main types:

- **Dependency:** A dependency is a semantic relationship between two or more classes where a change in one class may result in changes in another class. It represents a weaker relationship that indicates one class relies on another class, but not necessarily in an inheritance or association manner.

- **Generalization:** Generalization is a relationship between a parent class (superclass) and a child class (subclass). In this relationship, the child class inherits attributes, operations, and characteristics from the parent class. It signifies an "is-a" relationship where the child class is a specialized version of the parent class.

- **Association:** Association represents a static or physical connection between two or more objects, describing how many objects are involved in the relationship and their roles within that relationship. Associations can also include multiplicities to specify the number of instances participating in the relationship.

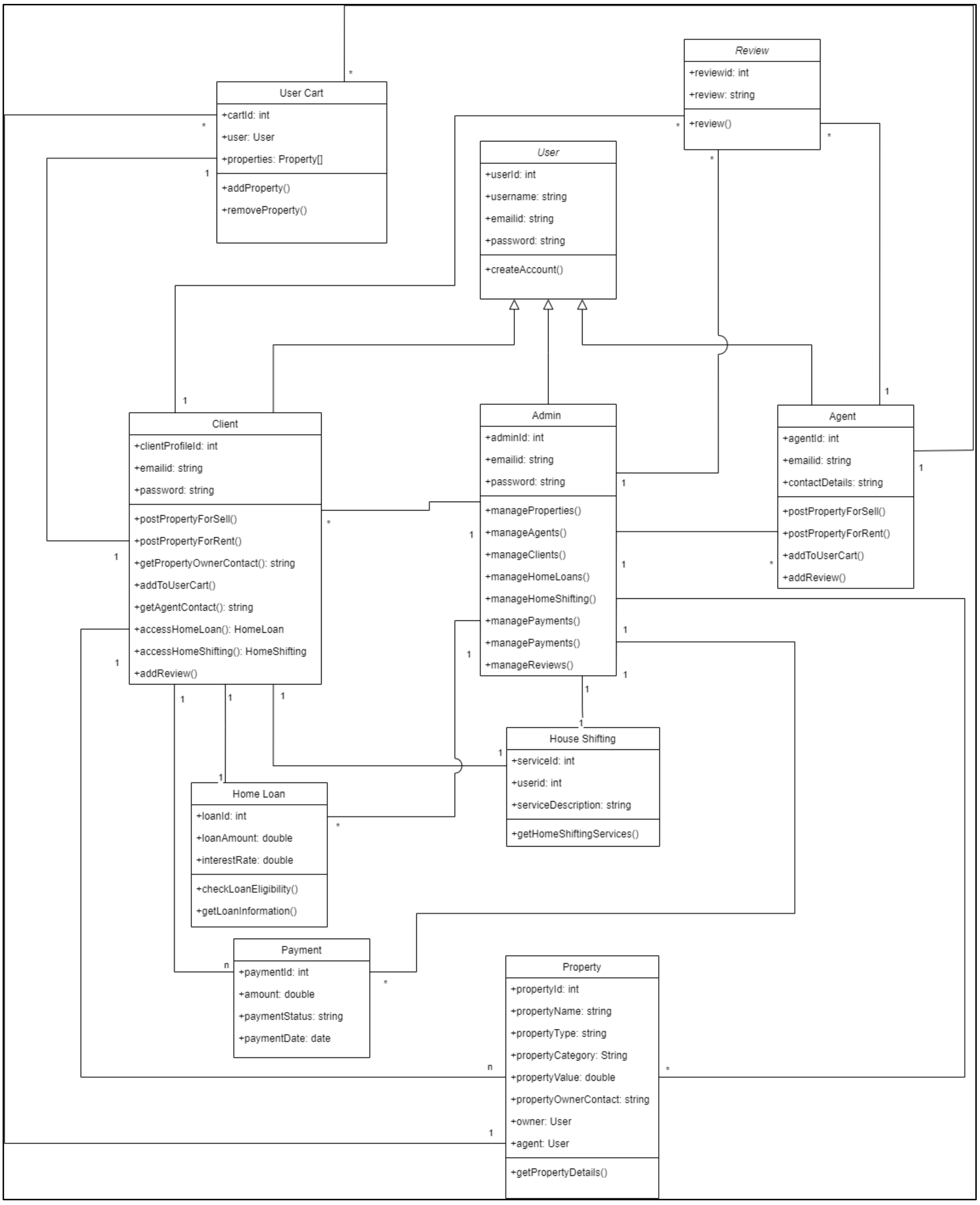


Figure 5 : Class Diagram of FindMyNest

4.2.4 Object Diagram

Object diagrams are based on the class diagram because they are based on the class diagram. They describe class diagram representation by object representation. Objects provide a static representation of the object system at a given moment. Object and class diagrams are similar in a way, but the class diagram provides a conceptual overview of the system and provides a visual representation of a specific function of the system

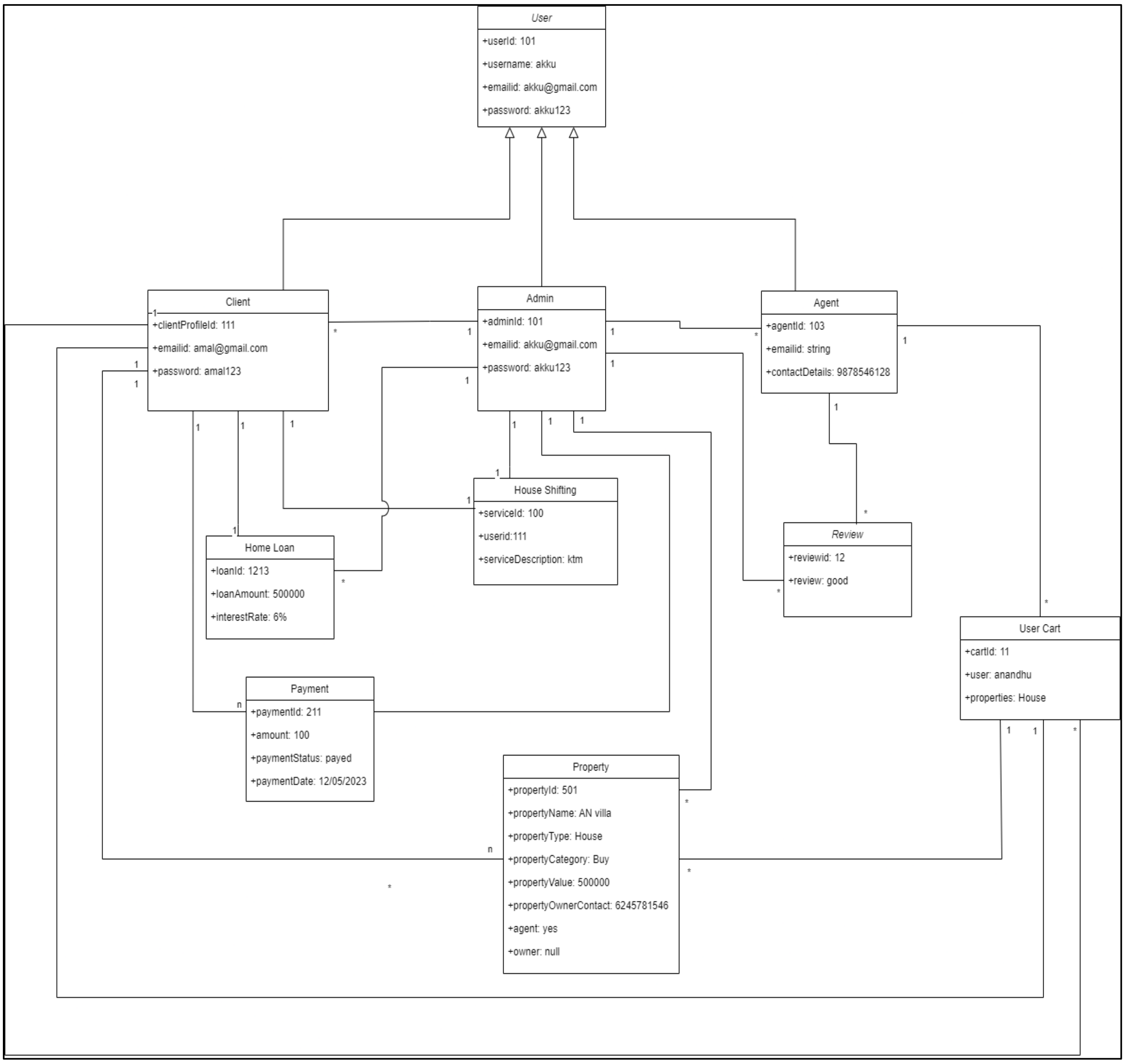


Figure 6 : Object Diagram of FindMyNest

4.2.5 Component Diagram

There are smaller components to make a complex object system more understandable separated using a component diagram. It simulates the physical view of the system, including this internal node executables, files, libraries, etc. It describes existing connections and hierarchies between system components. It helps to create a working system. individual, replaceable and executable system unit is called a component. Component application information is hidden, so a user interface is required to perform the function. This acts as a "black box" where the provided and required interfaces explain its behavior

Notation of a Component Diagram

1. A component
2. A node

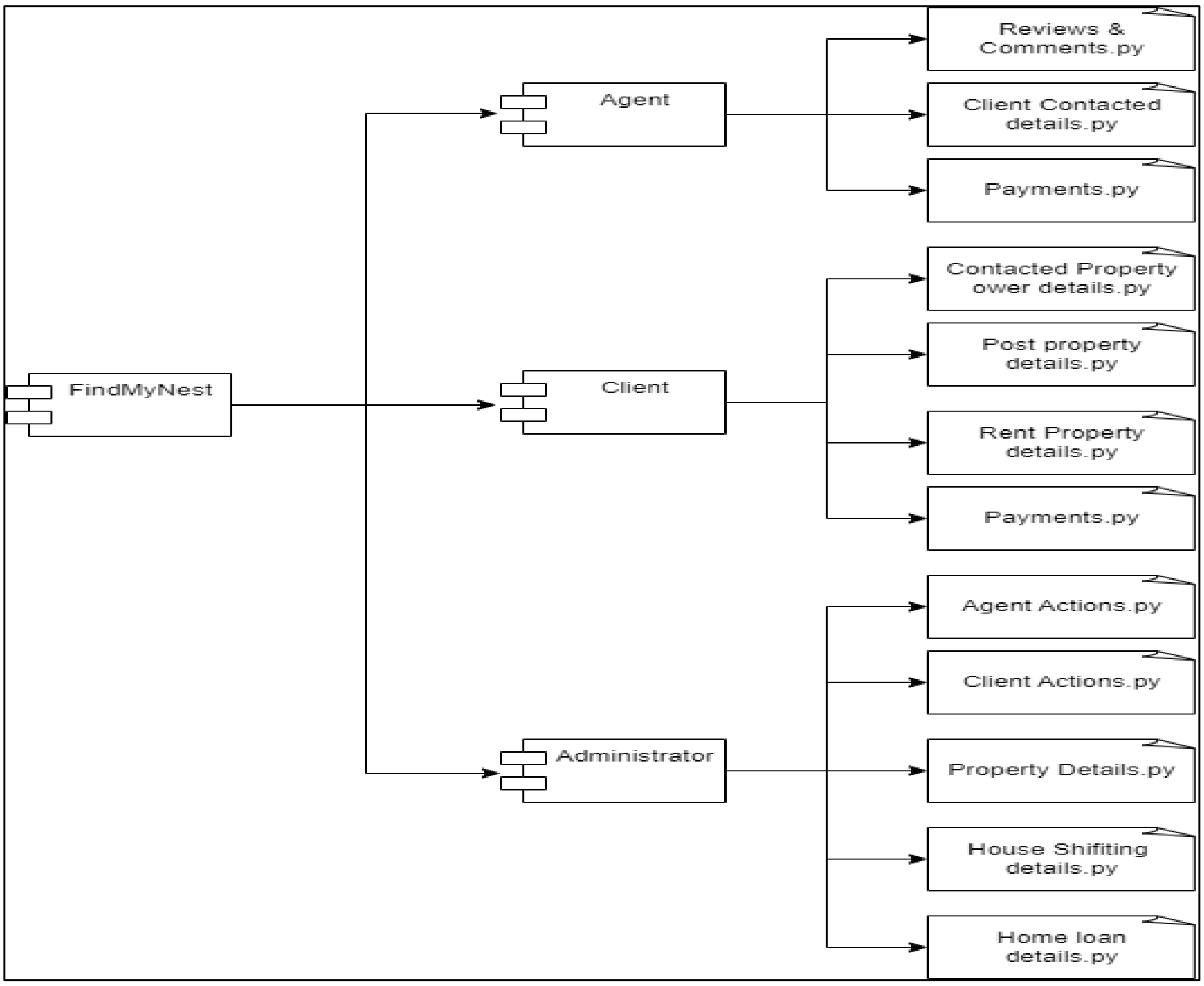


Figure 7 : Component Diagram of FindMyNest

4.2.8 Deployment Diagram

An deployment diagram is a type of UML diagram that shows the physical hardware or system the architecture in which applications are deployed. This gives a static view system implementation and includes nodes and their relationships. The diagram shows how software components are distributed and deployed on physical hardware or nodes. Diagram is useful for mapping a software architecture designed to a physical system architecture where the software is launched. Communication channels are used to show relationships between people

Notation of Deployment diagram

The deployment diagram consists of the following notations:

1. A component
2. An artifact
3. An interface
4. A nod

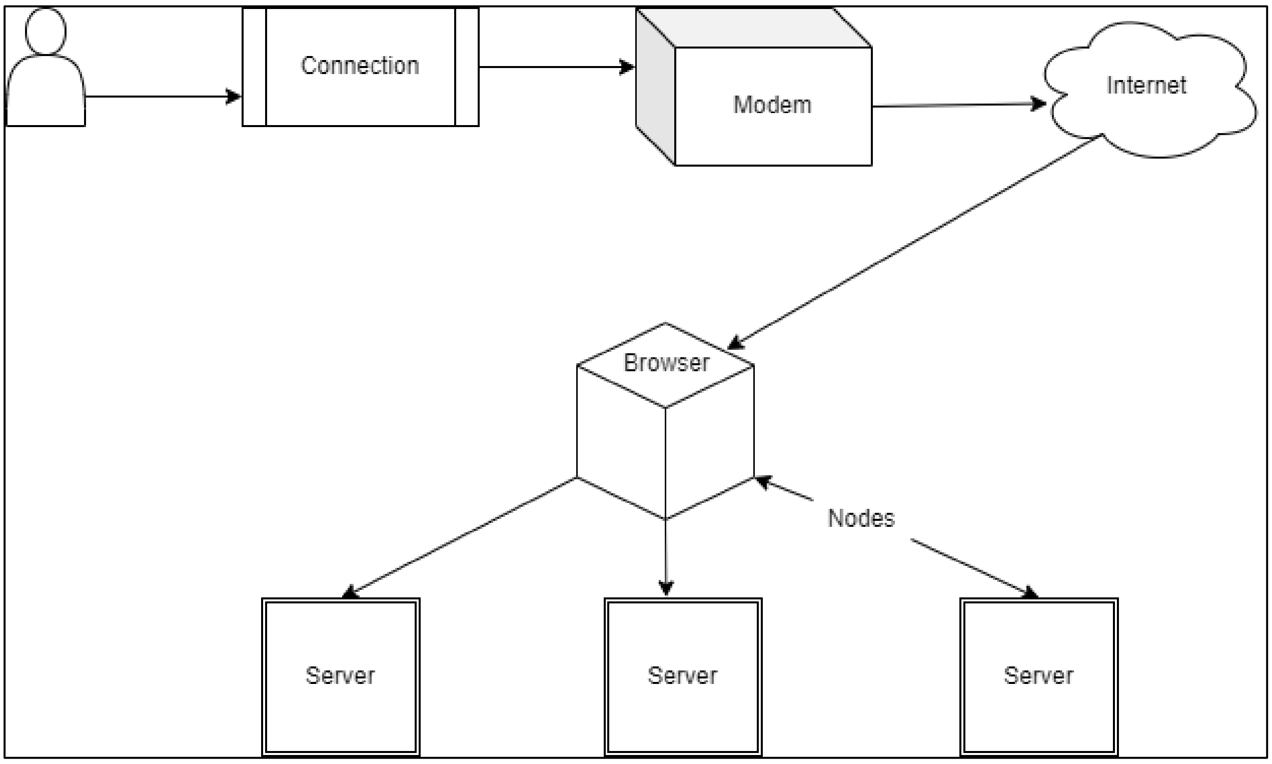


Figure 8 : Deployment Diagram of FindMyNest

4.3 USER INTERFACE DESIGN USING FIGMA

Form Name: User Registration

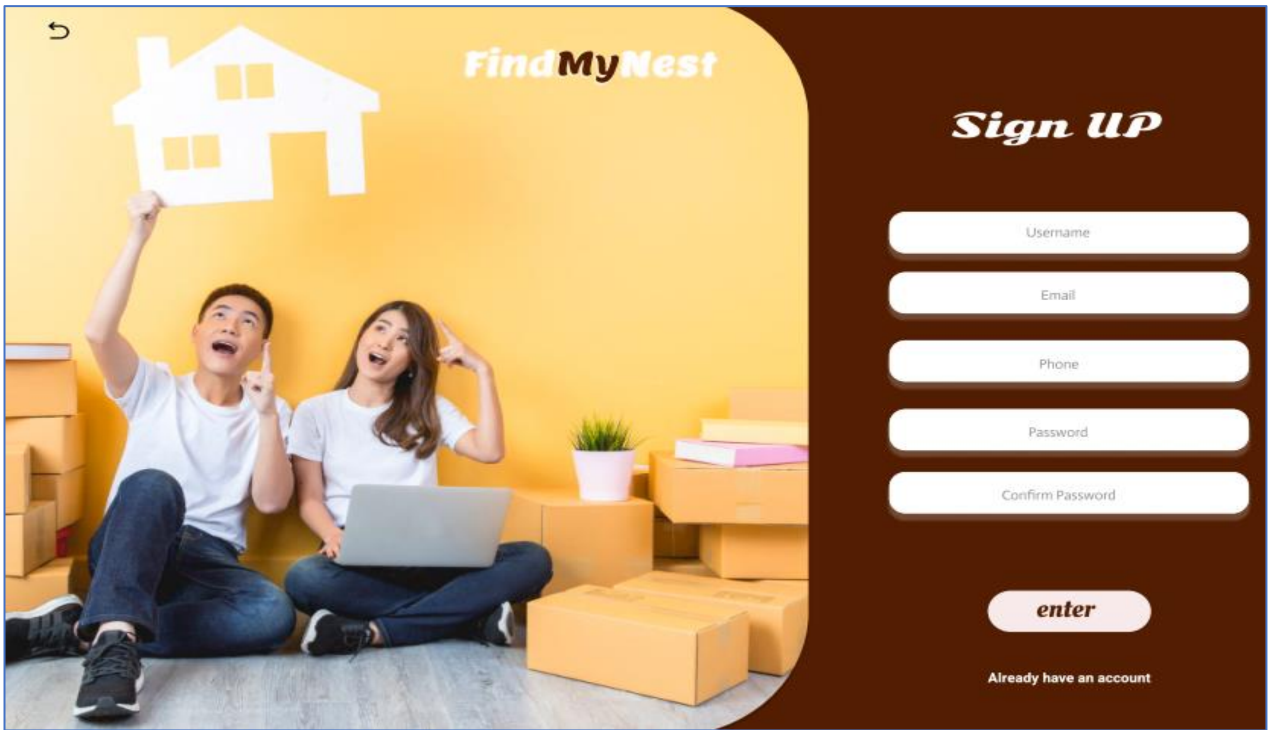


Figure 9 : Prototype of Registration Page

Form Name: User Login

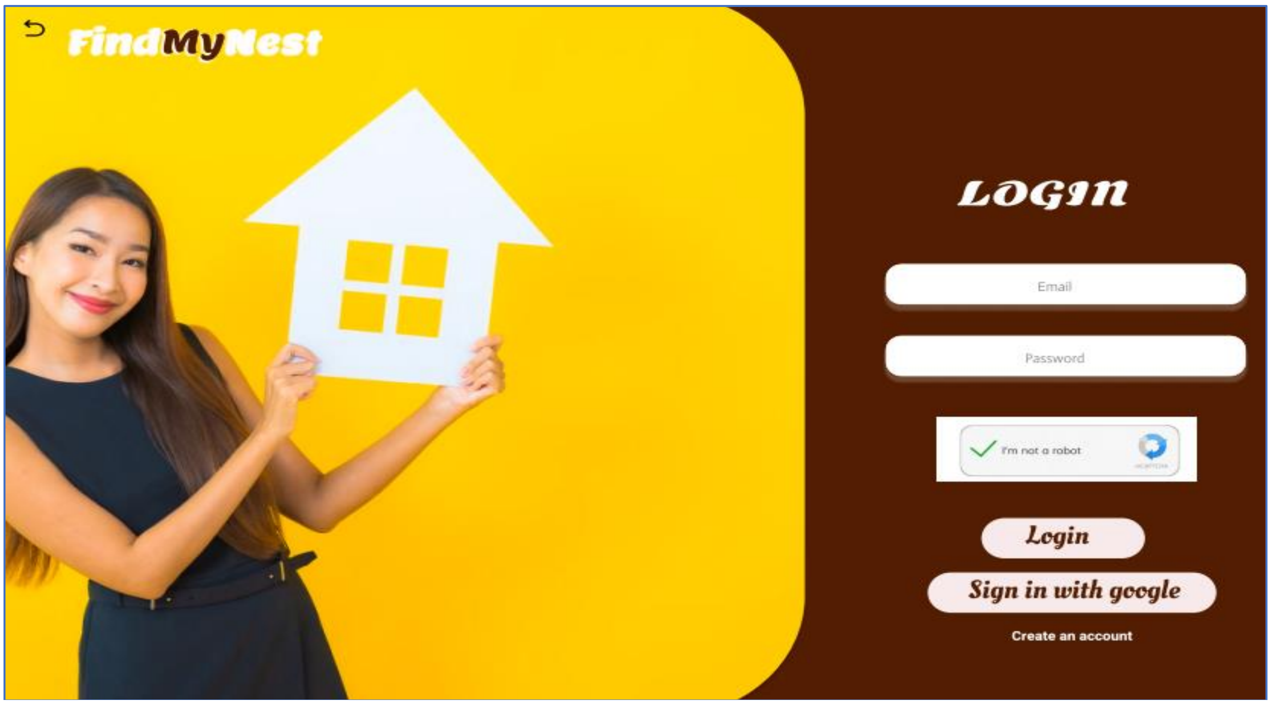


Figure 10 : Prototype of Login Page

Form Name: Post Property

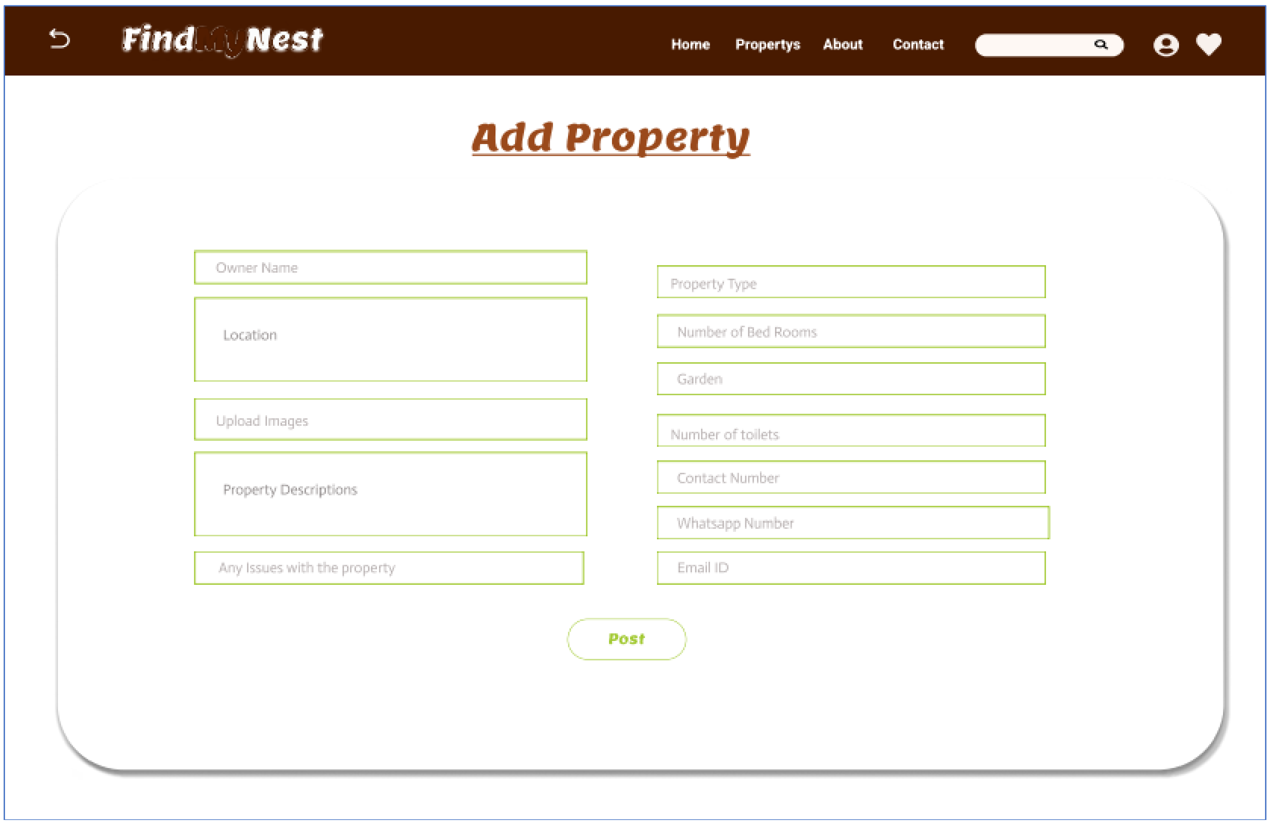
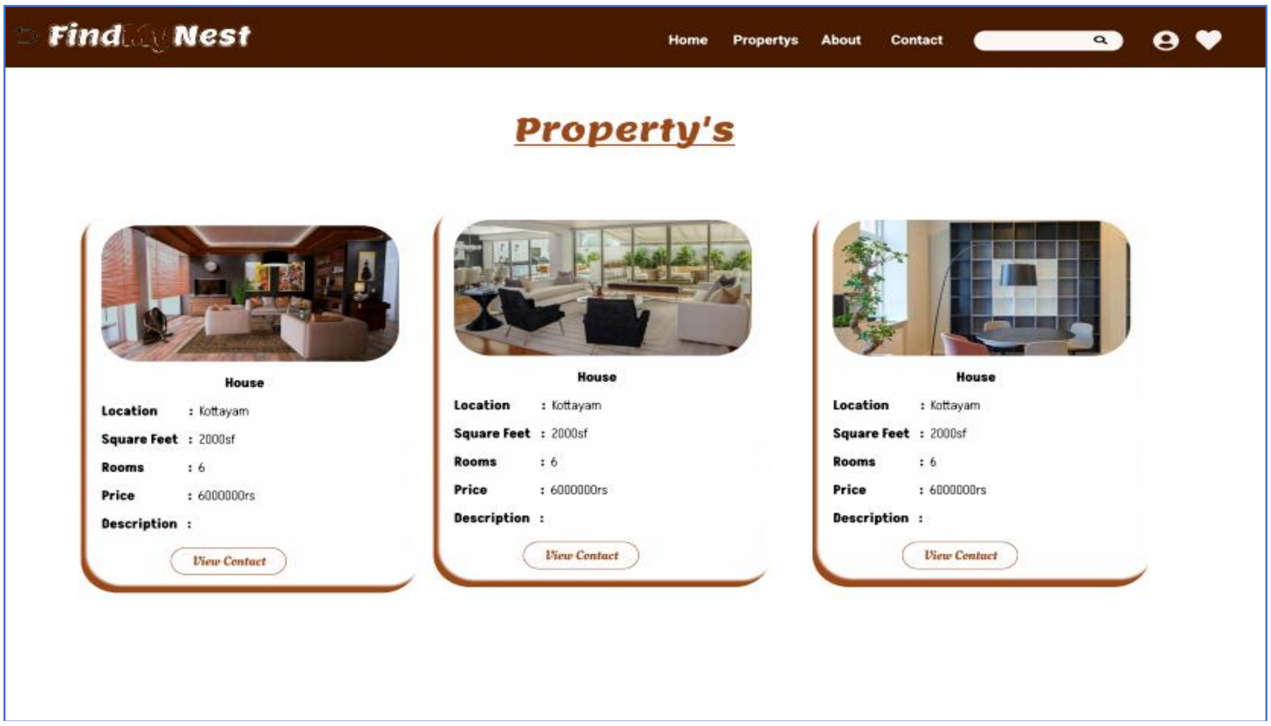


Figure 11 : Prototype of Post Property Page

Form Name: Property List



House

Location	: Kottayam
Square Feet	: 2000sf
Rooms	: 6
Price	: 6000000rs
Description	:

House

Location	: Kottayam
Square Feet	: 2000sf
Rooms	: 6
Price	: 6000000rs
Description	:

House

Location	: Kottayam
Square Feet	: 2000sf
Rooms	: 6
Price	: 6000000rs
Description	:

Figure 12 : Prototype of Property List Page

4DATABASE DESIGN

A organized system called a database allows users to retrieve information quickly and easily and has the capacity to retain information. The main objective of any database is its data, which need protection. The database design process is divided into two steps. In order to create a database that as clearly as possible satisfies these objectives, the user requests must first be identified. This stage, called as Information Level Design, is carried out independently from any specific DBMS.UML is a popular language utilized for modeling systems and objects found in the real world, and one of the tools it employs is use case diagrams to represent their creation and operation. The system architecture is paralleled by a database design.

The database's data layout seeks to achieve the two objectives listed below

- Data independence
- Data Integrity

4.4.1 Relational Database Management System (RDBMS)

RDBMS stands for Relational Database Management System. All modern database management systems like SQL, MS SQL Server, IBM DB2, ORACLE, My-SQL, and Microsoft Access are based on RDBMS. It is called Relational Database Management System (RDBMS) because it is based on the relational model introduced by E.F. Codd. Data is represented in terms of tuples (rows) in RDBMS.A relational database is the most commonly used database. It contains several tables, and each table has its primary key. Due to a collection of an organized set of tables, data can be accessed easily in RDBMS.

Relations, Domains & Attributes

A table is a data structure representing a relation between entities, where the rows in a table, commonly referred to as tuples, are composed of an ordered set of n elements. These elements are organized into columns, also known as attributes, which define the characteristics of the entities. Relationships between tables in a database are established to ensure both referential and entity relationship integrity. To define the domain of a data attribute, a set of atomic values is specified, typically drawn from a particular data type. Assigning a name to the domain aids in interpreting its values and is a common practice

Relationships

- Table relationships are established using Key. The two main keys of prime importance are Primary Key & Foreign Key. Entity Integrity and Referential Integrity Relationships can be established with these keys.
- Entity Integrity enforces that no Primary Key can have null values.
- Referential Integrity enforces that no Primary Key can have null values.
- Referential Integrity for each distinct Foreign Key value, there must exist a matching Primary Key value in the same domain. Other key are Super Key and Candidate Keys

4.4.2 Normalization

During the initial phase, data is grouped together in a basic manner to minimize the impact of future changes on data structures. The process of normalizing data structures formally aims to enhance data integrity and reduce duplication by organizing it into logical and efficient groupings. Using the normalization technique, superfluous fields are removed and a huge table is divided into several smaller ones. Anomalies in insertion, deletion, and updating are also prevented by using it. Keys and relationships are two notions used in the standard form of data modelling. In a table, In a table, each row is distinguished by a unique key, which may either be a primary key or a foreign key. A primary key is a single or multiple elements that act as an exclusive identifier for the particular row in the table, in a table that serves as a means of distinguishing between records from the same table. A column in a table known as a foreign key is used to uniquely identify records from other tables. Up to the third normal form, all tables have been normalized. It means placing things in their natural form, as the name suggests. By using normalization, the application developer aims to establish a coherent arrangement of the data into appropriate tables and columns, where names may be quickly related to the data by the user. By removing recurring groups from the data, normalization prevents data redundancy, which puts a heavy strain on the computer's resources.

These consist of

- Select appropriate table and column names.
- Normalize the data.
- Choose the proper name for the data

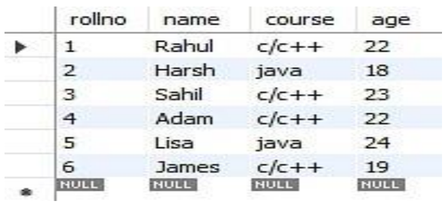
First Normal Form(1NF)

- A table is considered to be in the First Normal Form if it satisfies the requirement of atomicity, which means that the table's values cannot be divided into smaller, more granular parts, and each attribute or column contains only a single value. In other words, each column should hold only one piece of information, and there should be no repeating groups or arrays of values within a single row.

- The concept of atomicity in this context denotes that a cell within a database cannot contain more than one value, and must exclusively store a single-valued attribute.

- The first normal form in database normalization places limitations on attributes that have multiple values, are composed of multiple sub-attributes, or contain a combination of both. Therefore, in order to satisfy the conditions of the first normal form., these attributes need to be modified or removed, a relation must not have multi-valued or composite attributes, and any such attributes must be split into individual attributes to form atomic values.

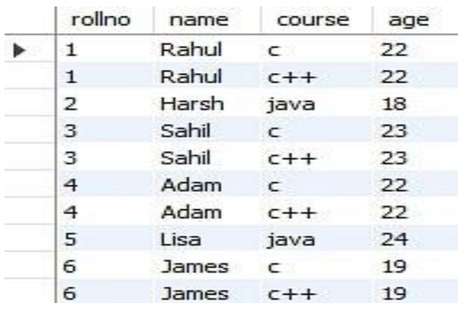
E.g., The table contains information pertaining to students, including their roll number, name, course of study, and age.



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

Figure 13 : Table not in 1NF

The table containing the records of students displays a violation of the First Normal Form due to the presence of two distinct values in the course column. To ensure compliance with the First Normal Form, the table has been modified resulting in the formation of a new table.



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

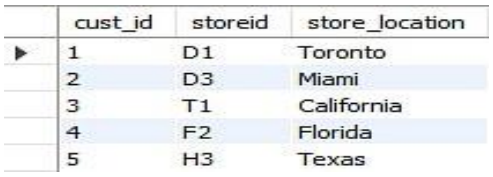
Figure 14 : Table in 1NF

The First Normal Form is applied to achieve atomicity in the system, which ensures that each column has unique values. By following this normalization process, the data is organized into individual atomic values, eliminating any redundancies or repeating groups. As a result, data integrity is maintained, and the system can efficiently handle complex data manipulations and updates.

Second Normal Form(2NF)

To meet requirements of Second Normal Form, a table must satisfy the criteria of First Normal Form as a prerequisite. Furthermore, the table must not exhibit partial dependency, which refers to a scenario where a non-prime attribute is dependent on a proper subset of the candidate key. In other words, the table should have no attributes that are determined by only a portion of the primary key.

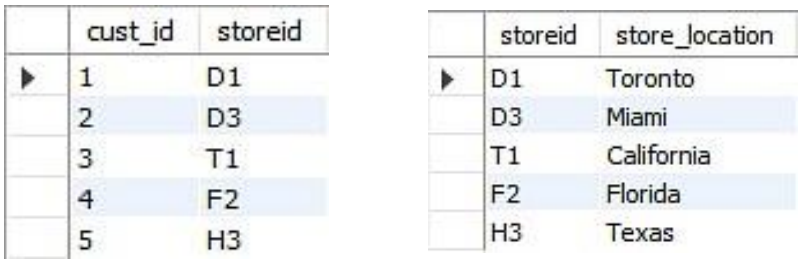
Now understand the Second Normal Form with the help of an example.
Consider the table Location:



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

Figure 15 : Table not in 2NF

The Location table currently has a composite primary key consisting of cust id and storied, and its non-key attribute is store location. However, the store location attribute is directly determined by the storied attribute, which is part of the primary key. As a result, this table does not meet the requirements of second normal form. To address this issue and ensure second normal form is met, it is necessary to separate the Location table into two separate tables. This will result in the creation of two distinct tables that accurately represent the relevant data and relationships: one for customer IDs and store IDs, and another for store IDs and their respective locations.:



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

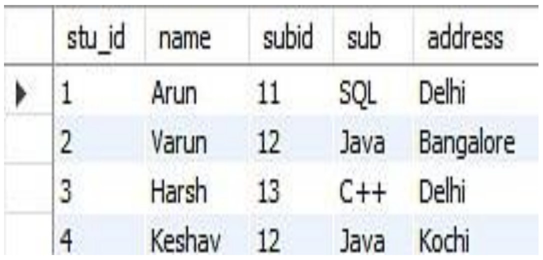
	storeid	store_location
▶	D1	Toronto
	D3	Miami
	T1	California
	F2	Florida
	H3	Texas

Figure 16 : Table in 2NF

After eliminating the partial functional dependency in the location table, it can be observed that the column storing the location is now entirely dependent on the primary key of the same table. In other words, the location column is fully determined by the primary key, thereby ensuring greater data consistency and integrity in the tabl

Third Normal Form

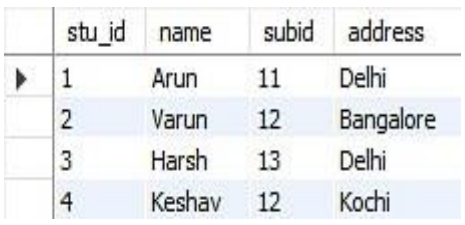
A table must meet the requirements of Second Normal Form in order to be considered in Third Normal Form, and also fulfill two additional conditions. The second condition states that non- prime attributes should not rely on non-prime characteristics that are not a part of the candidate key within the same table, thus avoiding transitive dependencies. A transitive dependency arises when A → C indirectly, due to A → B and B → C, where B is not functionally dependent on A. The main objective of achieving Third Normal Form is to reduce data redundancy and guarantee data integrity.
For instance, let's consider a student table that includes columns like student ID, student name, subject ID, subject name, and address of the student. To comply with the requirements for Third Normal Form, this table must first meet the standards of Second Normal Form and then ensure that there are no transitive dependencies between non-prime attributes.



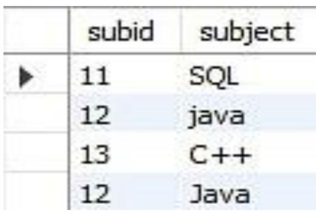
stu_id	name	subid	sub	address
1	Arun	11	SQL	Delhi
2	Varun	12	Java	Bangalore
3	Harsh	13	C++	Delhi
4	Keshav	12	Java	Kochi

Figure 17 : Table not in 3NF

Now to change the table to the third normal form, you need to divide the table as shown below: Based on the given student table, it can be observed that the stu_id attribute determines the sub_id attribute, and the sub_id attribute determines the subject (sub). This implies that there is a transitive functional dependency between stu_id and sub. As a result, the table does not satisfy the criteria for the third normal form. To adhere to the third normal form, the table must be divided into separate tables where each table represents a unique entity or relationship. In this case, the table can be divided into two tables: one for the student-subject relationship (stu_id and sub_id), and another for the subject information (sub_id and sub):



	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

Figure 18 : Table in 3NF

The two tables illustrate that the non-key attributes are completely determined by and reliant on the primary key. In the first table, the columns for name, sub_id, and addresses are all exclusively linked to the stu_id. Likewise, the second table demonstrates that the sub column is entirely dependent on the sub_id.

4.4.3 Sanitization

Data Sanitization involves the secure and permanent erasure of sensitive data from datasets and media to guarantee that no residual data can be recovered even through extensive forensic analysis. Data sanitization has a wide range of applications but is mainly used for clearing out end-of-life electronic devices or for the sharing and use of large datasets that contain sensitive information. The main strategies for erasing personal data from devices are physical destruction, cryptographic erasure, and data erasure. While the term data sanitization may lead some to believe that it only includes data on electronic media, the term also broadly covers physical media, such as paper copies. These data types are termed soft for electronic files and hard for physical media paper copies. Data sanitization methods are also applied for the cleaning of sensitive data, such as through heuristic-based methods, machine-learning based methods, and k-source anonymity

4.4.4 Indexing

Indexing is used to optimize the performance of a database by minimizing the number of disk accesses required when a query is processed. The index is a type of data structure. It is used to locate and access the data in a database table quickly.

- Primary Index − Primary index is defined on an ordered data file. The data file is ordered on a key field. The key field is generally the primary key of the relation.
- Secondary Index − Secondary index may be generated from a field which is a candidate key and has a unique value in every record, or a non-key with duplicate values.

- Clustering Index − Clustering index is defined on an ordered data file. The data file is ordered on a non-key field.

4.5 TABLE DESIGN

1.tbl_login

Primary key: **login_id**

No:	Fieldname	Datatype	Key Constraints	Description
1	login _id	int(11)	Primary Key	Unique User Identifier
2	username	varchar(100)	Not Null	User's Username
3	first_name	varchar(100)	Not Null	User's First Name
4	last_name	varchar(100)	Not Null	User's Last Name
5	email	varchar(100)	Not Null	User's Email Address
6	phone_no	varchar(12)	Not Null	User's Phone Number
7	user_type	PositiveSmallIntegerField	Not Null	User Type (e.g., User, Admin)
9	login_ status	boolean	Not null	User status
10	login_ last_login	datetime	Not null	Last Logined

2.tbl_register

Primary key: **reg_id**

Foreign key: **login_id** references table **tbl_login**

No:	Fieldname	Datatype	Key Constraints	Description
1	reg_id	int(11)	Primary Key	Primary Key
2	login_id_	int(11)	Foreign Key	References table tbl_login
3	profile_pic	varchar(255)	Not Null	Profile Picture
4	country	varchar(15)	Not Null	User's Country
5	state	varchar(15)	Not Null	User's State

6	city	varchar(15)	Not null	User's City
7	pin_code	varchar(6)	Not null	User's Pin Code

3.tbl_category

Primary key: **tbl_property**

Foreign key: **login_id** references table **tbl_login**

No:	Fieldname	Datatype	Key Constraints	Description
1	property_id	int(11)	Primary Key	Unique Property Identifier
2	login _id	int(11)	Foreign Key	References table tbl_login
3	thumbnail	varchar(255)	Not Null	Property Thumbnail
4	video	varchar(255)	-	Property Video (if available)
5	owner_name	varchar(30)	Not Null	Owner's Name
6	address	varchar(100)	Not Null	Property Address
7	Town	varchar(100)	Not Null	Town
8	zipcode	int(11)	Not Null	ZIP Code
9	description	varchar(500)	-	Property Description
10	floor_plan	varchar(255)	-	Floor Plan of the Property
11	whatsapp_no	int(11)	Not Null	WhatsApp Number
12	nearby_place	varchar(255)	-	Nearby Places of Interest
13	features	varchar(255)	-	Property Features
14	price	int(11)	-	Property Price
15	area	varchar(100)	-	Property Area
16	property_type	varchar(40)	-	Property Type (e.g., House, Apartment)
17	status	varchar(40)	-	Property Status (e.g., For Sale)

18	bedrooms	varchar(40)	-	Number of Bedrooms
19	bathrooms	varchar(40)	-	Number of Bathrooms
20	rooms	varchar(40)	-	Number of Rooms
21	state	varchar(40)	-	State
22	garage	varchar(40)	-	Garage Availability
23	major_road	varchar(40)	-	Distance to Major Road
24	near_hospital	varchar(40)	-	Distance to Nearest Hospital
25	near_supermarket	varchar(40)	-	Distance to Nearest Supermarket
26	bulding_age	varchar(40)	-	Building Age
27	floor	varchar(40)	-	Floor
28	floor_no	varchar(40)	-	Floor Number
29	last_renovated	varchar(40)	-	Last Renovated Year
30	created_at	datetime	-	Creation Timestamp
31	modified_at	datetime	-	Modification Timestamp
32	view_count	int(11)	-	View Count (Number of Views)
33	active	boolean	-	Property Status (Active or Inactive)
34	days_to_sell	int(11)	-	Days to Sell
35	property_tips	text	-	Property Tips
36	sale_duration_tips	text	-	Sale Duration Tips

4.tbl_ tbl_image

Primary key: **image_id**

Foreign key: **property_id** references table **tbl_property**

No:	Fieldname	Datatype	Key Constraints	Description
1	image_id	int(11)	Primary Key	Unique Image Identifier
2	property_id	int(11)	Foreign Key	References table tbl_property
3	images	varchar(255)	-	Property Images
1	image_id	int(11)	Primary Key	Unique Image Identifier

5. tbl_propertyview

Primary key: **propertyview _id**

Foreign key: **property_id references table tbl_property**

Foreign key: **login_id references table tbl_login**

No:	Fieldname	Datatype	Key Constraints	Description
1	propertyview _id	int(11)	Primary Key	Unique View Identifier
2	property_id	int(11)	Foreign Key	References table tbl_property
3	login _id	int(11)	Foreign Key	References table tbl_login
4	timestamp	datetime	-	View Timestamp

6. tbl_feedback

Primary key: **feedback_id**

Foreign key: **property_id** references table **tbl_property**

Foreign key: **login_id** references table **tbl_login**

No:	Fieldname	Datatype	Key Constraints	Description
1	feedback_id	int(11)	Primary Key	Unique Feedback Identifier
2	login_id	int(11)	Foreign Key	References table tbl_login
3	property_id	int(11)	Foreign Key	References table tbl_property
4	first_name	varchar(100)	Not Null	First Name

5	email	varchar(100)	-	Email Address
6	message	text	-	Feedback Message
7	comment_date	datetime	-	Comment Timestamp
8	likes	ManyToMany(tbl_login)	-	Users Who Liked the Feedback

7.tbl_wishlist

Primary key: **wishlist_id**

Foreign key: **property_id** references table **tbl_property**

Foreign key: **login_id** references table **tbl_login**

No:	Fieldname	Datatype	Key Constraints	Description
1	wishlist_id	int(11)	Primary Key	Unique Wishlist Identifier
2	login_id	int(11)	Foreign Key	References table tbl_login
3	property_id	int(11)	Foreign Key	References table tbl_property
4	created_at	datetime	-	Creation Timestamp

8. tbl_subscription

Primary key: **subscription_id**

No:	Fieldname	Datatype	Key Constraints	Description
1	subscription_id	int(11)	Primary Key	Unique Subscription Identifier
2	sub_type	varchar(40)	-	Subscription Type
3	price	decimal(8,2)	-	Subscription Price
4	validity	varchar(40)	-	Subscription Validity Duration
5	features	varchar(255)	-	Subscription Features

9. tbl_payment

Primary key**: payment_id**

Foreign key: **login_id references table tbl_login**

No:	Fieldname	Datatype	Key Constraints	Description
1	payment_id	int(11)	Primary Key	Unique Payment Identifier
2	login_id	int(11)	Foreign Key	References table tbl_login
3	razorpay_order_id	varchar(255)	-	Razorpay Order ID
4	payment_id	varchar(255)	-	Razorpay Payment ID
5	amount	decimal(8,2)	-	Payment Amount
6	currency	varchar(3)	-	Currency Code (e.g., "INR")
7	timestamp	datetime	-	Timestamp of the Payment
8	payment_status	varchar(20)	-	Payment Status

CHAPTER 5
SYSTEM TESTING

5.1 INTRODUCTION

Software Testing is the most common way of executing programming in a controlled way, to respond to the inquiry - Does the product act as determined? Programming testing is much of the time utilized in relationship with the term's confirmation and approval. Approval is the checking or testing of things, incorporates programming, for conformance and consistency with a related detail. Programming testing is only one sort of confirmation, which additionally utilizes strategies like surveys, investigation, reviews, and walkthroughs. Approval is the most common way of making sure that what has been indicated is what the client really cared about.

Different exercises which are frequently connected with programming testing are static examination and dynamic investigation. Static examination explores the source code of programming, searching for issues and assembling measurements without really executing the code. Dynamic examination takes a gander at the way of behaving of programming while it is executing, to give data, for example, execution follows, timing profiles, and test inclusion data. Testing is a bunch of movement that can be arranged in cutting edge and directed efficiently. Testing starts at the module level and work towards the mix of whole PCs based framework. Nothing is finished without testing, as its imperative outcome of the framework testing targets, there are a few principles that can act as testing goals. They are:

Testing is a process of executing a program with the intent of finding an error.

- A successful test is one that uncovers an undiscovered error.

On the off chance that a testing is led effectively as per the targets as expressed above, it would uncover mistakes in the product. Likewise testing show that the product capability seems, by all accounts, to be working as indicated by the particular, that exhibition necessity seems to have been met. There are three ways to test program.

- For correctness
- For implementation efficiency
- For computational complexity

Test for correctness is supposed to verify that a program does exactly what it was designed to do. This is much more difficult than it may at first appear, especially for large programs

5.2 TEST PLAN

A test plan defines the steps necessary to carry out different testing approaches, as well as the tasks that must be accomplished. The task of generating computer programmers, documentation, and data structures falls under the purview of software developers. Individual testing is required to make sure that every part of the programmed operates as planned. An independent test group (ITG) is ready to give developers feedback and spot any potential problems. The test strategy must include a quantification of the testing objectives. These goals may include measurements like the mean time to failure, the cost of fixing errors, the density of remaining errors, the frequency of recurrence, and the number of hours needed for regression testing. The testing levels could include:

- Unit testing
- Integration Testing
- Validation Testing or System Testing
- Output Testing or User Acceptance Testing
- Automation Testing
- Selenium Testing

5.2.1 Unit Testing

Unit testing centers confirmation exertion around the littlest unit of programming plan - the product part or module. Involving the part level plan portrayal as an aide, significant control ways are tried to reveal mistakes inside the limit of the module. The overall intricacy of tests and uncovered scope laid out for unit testing. The unit testing is white-box situated, and step can be led in lined up for different parts. The measured point of interaction is tried to guarantee that data appropriately streams into and out of the program unit under test. The nearby information structure is inspected to guarantee that information put away briefly keeps up with its honesty during all means in a calculation's execution. Limit conditions are tried to guarantee that all assertions in a module have been executed something like once. At long last, all mistake dealing with ways are tried. Trial of information stream across a module point of interaction are expected before some other test is started. On the off chance that information don't enter and exit appropriately, any remaining tests are unsettled. Specific testing of execution ways is a fundamental errand during the unit test. Great plan directs that blunder conditions be expected and mistake dealing with ways set up to reroute or neatly end handling when a mistake happens. Limit testing is the last undertaking of unit testing step. Programming frequently falls flat at its limits. Unit testing was finished in Sell-Delicate

Framework by regarding every module as discrete substance and testing every single one of them with a wide range of test inputs. A few imperfections in the inward rationale of the modules were found and were corrected. Subsequent to coding every module is tried and run separately. All pointless code were eliminated and guaranteed that all modules are working, and gives the normal outcome.

5.2.2 Integration Testing

Integration testing is orderly procedure for developing the program structure while simultaneously directing tests to reveal blunders related with connecting. The goal is to take unit tried parts and construct a program structure that has been directed by plan. The whole program is tried as entirety. Rectification is troublesome in light of the fact that detachment of causes is confounded by immense breadth of whole program. When these blunders are revised, new ones show up and the cycle go on in an apparently unending circle. In the wake of performing unit testing in the Framework every one of the modules were coordinated to test for any irregularities in the connection points. Besides contrasts in program structures were eliminated and an extraordinary program structure was developed

5.2.3 Validation Testing or System Testing

This is the last move toward testing. In this the whole framework was tried in general with all structures, code, modules and class modules. This type of testing is famously known as Discovery testing or Framework tests. Black Box testing technique centers around the practical prerequisites of the product. That is, Black Box testing empowers the computer programmer to infer sets of info conditions that will completely practice all utilitarian prerequisites for a program. Black Box testing endeavors to track down mistakes in the accompanying classes; erroneous or missing capabilities, interface blunders, mistakes in information designs or outside information access, execution mistakes and instatement blunders and end mistakes.

5.2.4 Output Testing or User Acceptance Testing

The system considered is tested for user acceptance; here it should satisfy the firm’s need. The software should keep in touch with perspective system; user at the time of developing and making changes whenever required. This done with respect to the following points:

- Input Screen Designs
- Output Screen Designs

The above testing is done taking various kinds of test data. Preparation of test data plays a vital role in the system testing. After preparing the test data, the system under study is tested using that test data. While testing the system by which test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use

5.2.5 Automation Testing

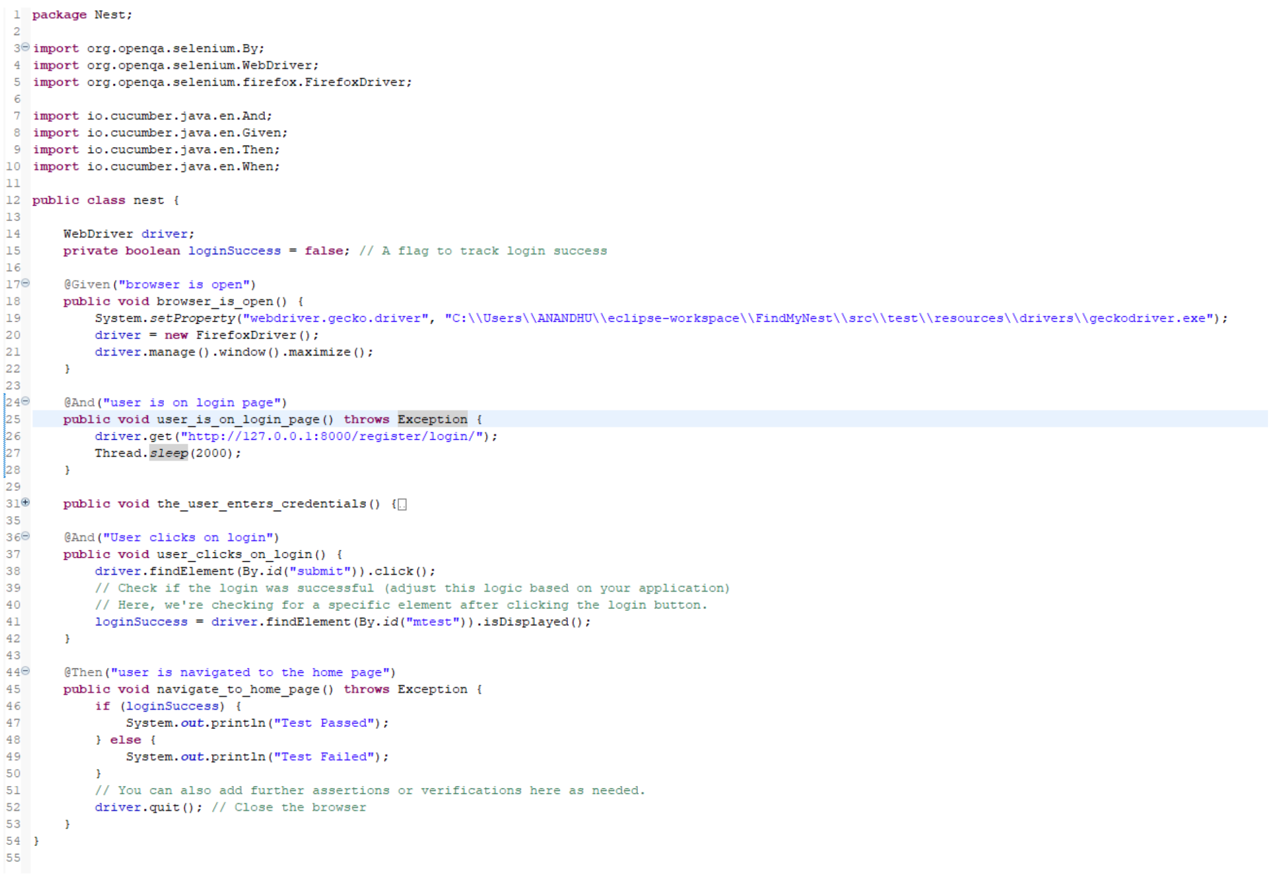
An experiment suite is executed utilizing specific robotized testing programming devices as a component of the product testing method known as mechanization testing. The test stages are fastidiously completed by a human performing manual testing while situated before a PC. Moreover, the robotization testing programming might produce intensive test reports, think about expected and genuine discoveries, and enter test information into the Framework Under Test. Programming test computerization requires huge monetary and material data sources. Rehashed execution of a similar test suite will be vital during ensuing improvement cycles. This test suite can be recorded and replayed depending on the situation utilizing a test robotization device. No further human contribution is required once the test suite has been mechanized

5.2.6 Selenium Testing

Selenium is a widely used open-source automation testing suite for web UI. Originally developed by Jason Huggins in 2004, Selenium supports automation across different browsers, platforms, and programming languages. It can be deployed on Windows, Linux, Solaris, Macintosh, and even supports mobile OS such as iOS, Windows Mobile, and Android. Selenium supports various programming languages through drivers specific to each language including C#, Java, Perl, PHP, Python, and Ruby. Selenium Web Driver is the most popular with Java and C#. Test scripts can be coded in any supported language and run directly in modern web browsers such as Internet Explorer, Mozilla Firefox, Google Chrome, and Safari. Selenium is not only used for functional tests but can also be integrated with automation test tools like Maven, Jenkins, and Docker for continuous testing. Furthermore, it can be integrated with TestNG and JUnit for managing test cases and generating reports

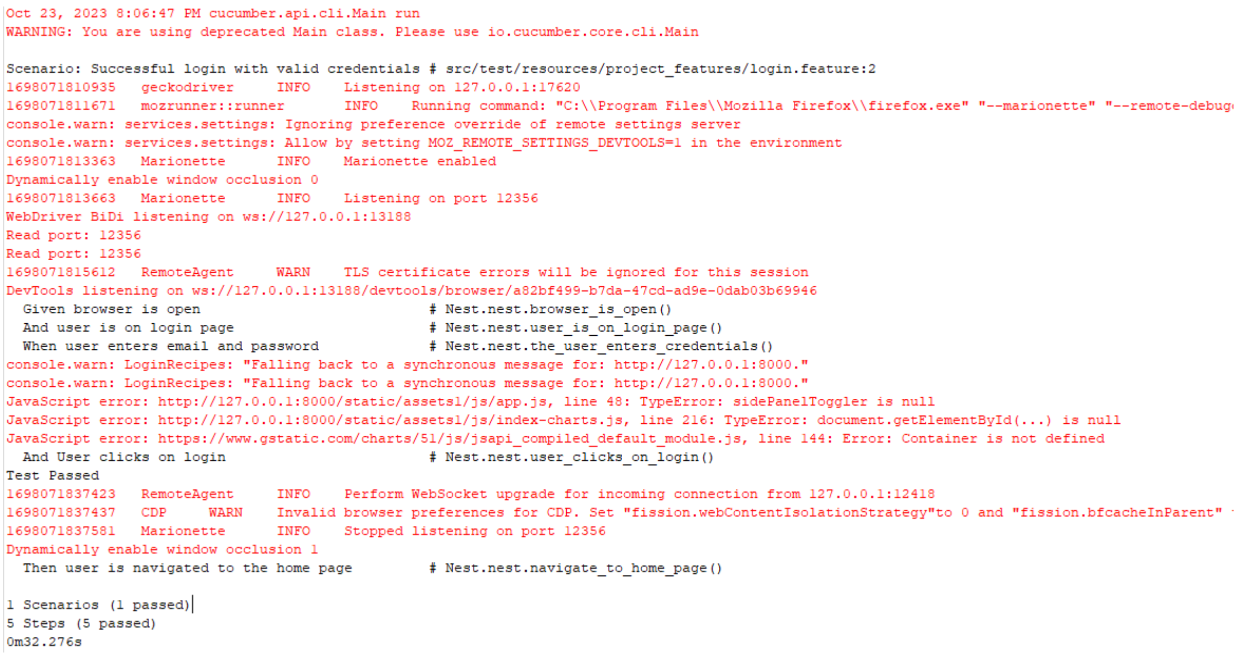
Test Case 1

Code



```java
package Nest;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

import io.cucumber.java.en.And;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;

public class nest {

    WebDriver driver;
    private boolean loginSuccess = false; // A flag to track login success

    @Given("browser is open")
    public void browser_is_open() {
        System.setProperty("webdriver.gecko.driver", "C:\\Users\\ANANDHU\\eclipse-workspace\\FindMyNest\\src\\test\\resources\\drivers\\geckodriver.exe");
        driver = new FirefoxDriver();
        driver.manage().window().maximize();
    }

    @And("user is on login page")
    public void user_is_on_login_page() throws Exception {
        driver.get("http://127.0.0.1:8000/register/login/");
        Thread.sleep(2000);
    }

    public void the_user_enters_credentials() {

    @And("User clicks on login")
    public void user_clicks_on_login() {
        driver.findElement(By.id("submit")).click();
        // Check if the login was successful (adjust this logic based on your application)
        // Here, we're checking for a specific element after clicking the login button.
        loginSuccess = driver.findElement(By.id("mtest")).isDisplayed();
    }

    @Then("user is navigated to the home page")
    public void navigate_to_home_page() throws Exception {
        if (loginSuccess) {
            System.out.println("Test Passed");
        } else {
            System.out.println("Test Failed");
        }
        // You can also add further assertions or verifications here as needed.
        driver.quit(); // Close the browser
    }
}
```

Output



```
Oct 23, 2023 8:06:47 PM cucumber.api.cli.Main run
WARNING: You are using deprecated Main class. Please use io.cucumber.core.cli.Main

Scenario: Successful login with valid credentials # src/test/resources/project_features/login.feature:2
1698071810935   geckodriver     INFO    Listening on 127.0.0.1:17620
1698071811671   mozrunner::runner       INFO    Running command: "C:\\Program Files\\Mozilla Firefox\\firefox.exe" "--marionette" "--remote-debug
console.warn: services.settings: Ignoring preference override of remote settings server
console.warn: services.settings: Allow by setting MOZ_REMOTE_SETTINGS_DEVTOOLS=1 in the environment
1698071813363   Marionette      INFO    Marionette enabled
Dynamically enable window occlusion 0
1698071813663   Marionette      INFO    Listening on port 12356
WebDriver BiDi listening on ws://127.0.0.1:13188
Read port: 12356
Read port: 12356
1698071815612   RemoteAgent     WARN    TLS certificate errors will be ignored for this session
DevTools listening on ws://127.0.0.1:13188/devtools/browser/a82bf499-b7da-47cd-ad9e-0dab03b69946
  Given browser is open                             # Nest.nest.browser_is_open()
  And user is on login page                         # Nest.nest.user_is_on_login_page()
  When user enters email and password               # Nest.nest.the_user_enters_credentials()
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
JavaScript error: http://127.0.0.1:8000/static/assets1/js/app.js, line 48: TypeError: sidePanelToggler is null
JavaScript error: http://127.0.0.1:8000/static/assets1/js/index-charts.js, line 216: TypeError: document.getElementById(...) is null
JavaScript error: https://www.gstatic.com/charts/51/js/jsapi_compiled_default_module.js, line 144: Error: Container is not defined
  And User clicks on login                          # Nest.nest.user_clicks_on_login()
Test Passed
1698071837423   RemoteAgent     INFO    Perform WebSocket upgrade for incoming connection from 127.0.0.1:12418
1698071837437   CDP     WARN    Invalid browser preferences for CDP. Set "fission.webContentIsolationStrategy"to 0 and "fission.bfcacheInParent"
1698071837581   Marionette      INFO    Stopped listening on port 12356
Dynamically enable window occlusion 1
  Then user is navigated to the home page           # Nest.nest.navigate_to_home_page()

1 Scenarios (1 passed)
5 Steps (5 passed)
0m32.276s
```

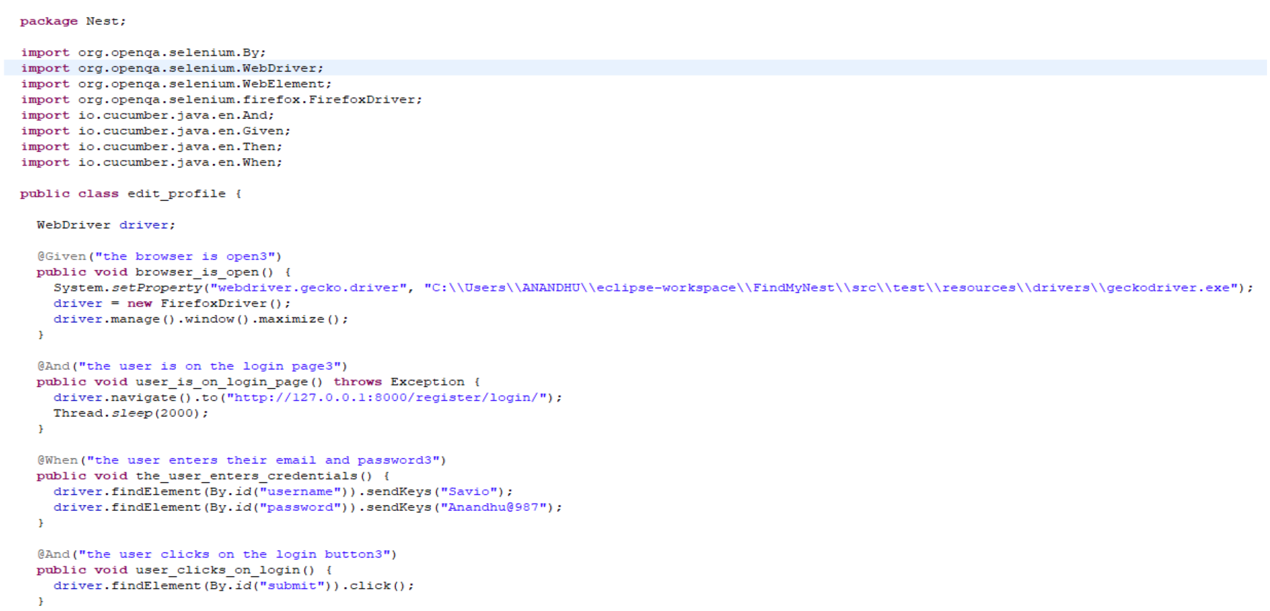
Test Report

Test Case 1

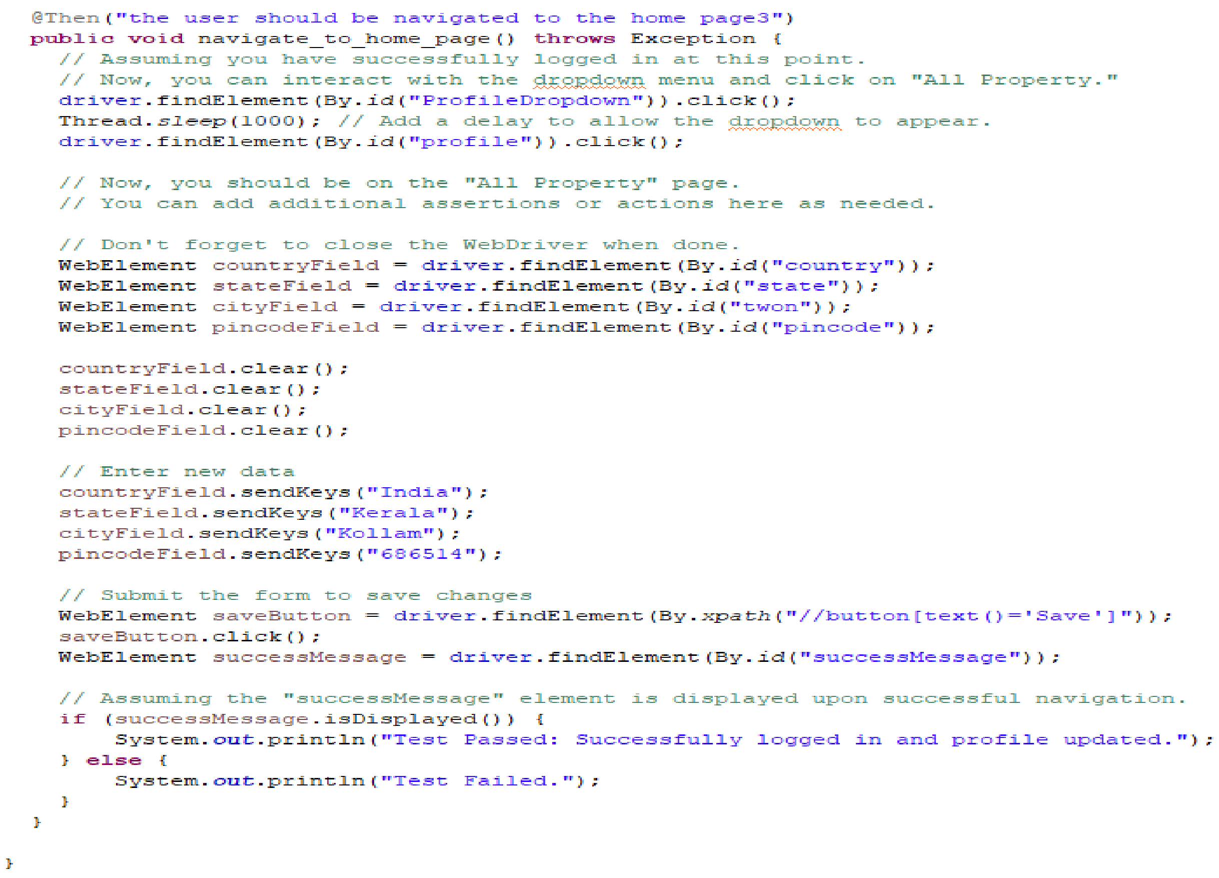
Project Name:FindMyNest					
Login Test Case					
Test Case ID: 1			**Test Designed By:** Anandhu Biju		
Test Priority(Low/Medium/High): High			**Test Designed Date:** 23/10/2023		
Module Name: Login Page			**Test Executed By :** Ms. Sruthimol Kurian		
Test Title : Verify login with valid Username and Password			**Test Execution Date:** 23/10/2023		
Description: Test the Login Page					
Pre-Condition : User has valid username and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Navigation to Login Page		Login Page should be displayed	Login page displayed	Pass
2	Provide Valid Username	User Name: Anandhu	User should be able to Login	User Logged in and navigated to the dashboard with records	Pass
3	Provide Valid Password	Password: Anandhu@987			
4	Click on Sign In button				
Post-Condition: User is validated with database and successfully login into account. The Account session details are logged in database					

Test Case 2:

Code

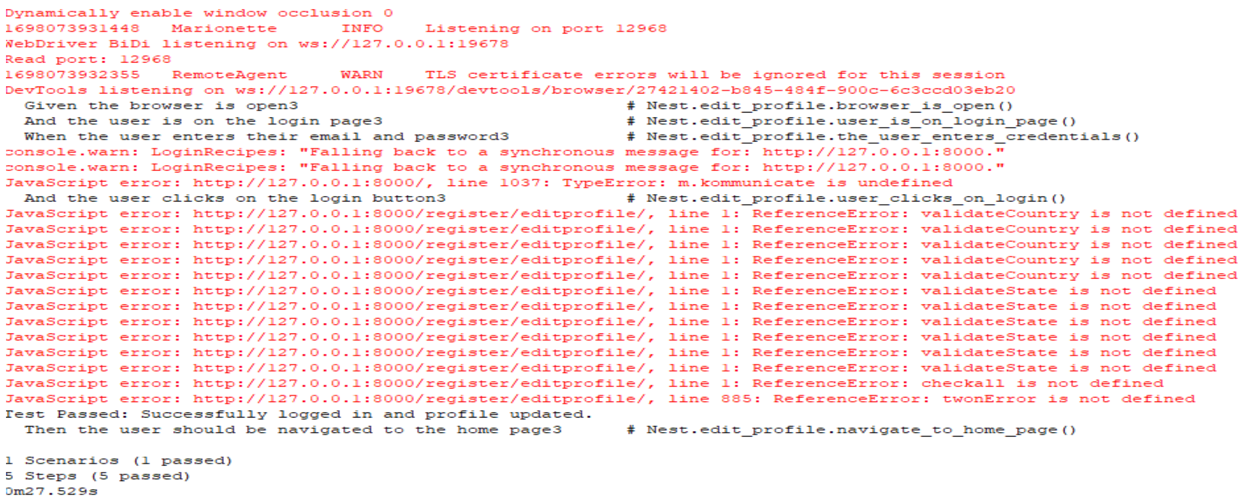


```java
package Nest;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import io.cucumber.java.en.And;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;

public class edit_profile {

  WebDriver driver;

  @Given("the browser is open3")
  public void browser_is_open() {
    System.setProperty("webdriver.gecko.driver", "C:\\Users\\ANANDHU\\eclipse-workspace\\FindMyNest\\src\\test\\resources\\drivers\\geckodriver.exe");
    driver = new FirefoxDriver();
    driver.manage().window().maximize();
  }

  @And("the user is on the login page3")
  public void user_is_on_login_page() throws Exception {
    driver.navigate().to("http://127.0.0.1:8000/register/login/");
    Thread.sleep(2000);
  }

  @When("the user enters their email and password3")
  public void the_user_enters_credentials() {
    driver.findElement(By.id("username")).sendKeys("Savio");
    driver.findElement(By.id("password")).sendKeys("Anandhu@987");
  }

  @And("the user clicks on the login button3")
  public void user_clicks_on_login() {
    driver.findElement(By.id("submit")).click();
  }
```



```java
  @Then("the user should be navigated to the home page3")
  public void navigate_to_home_page() throws Exception {
    // Assuming you have successfully logged in at this point.
    // Now, you can interact with the dropdown menu and click on "All Property."
    driver.findElement(By.id("ProfileDropdown")).click();
    Thread.sleep(1000); // Add a delay to allow the dropdown to appear.
    driver.findElement(By.id("profile")).click();

    // Now, you should be on the "All Property" page.
    // You can add additional assertions or actions here as needed.

    // Don't forget to close the WebDriver when done.
    WebElement countryField = driver.findElement(By.id("country"));
    WebElement stateField = driver.findElement(By.id("state"));
    WebElement cityField = driver.findElement(By.id("twon"));
    WebElement pincodeField = driver.findElement(By.id("pincode"));

    countryField.clear();
    stateField.clear();
    cityField.clear();
    pincodeField.clear();

    // Enter new data
    countryField.sendKeys("India");
    stateField.sendKeys("Kerala");
    cityField.sendKeys("Kollam");
    pincodeField.sendKeys("686514");

    // Submit the form to save changes
    WebElement saveButton = driver.findElement(By.xpath("//button[text()='Save']"));
    saveButton.click();
    WebElement successMessage = driver.findElement(By.id("successMessage"));

    // Assuming the "successMessage" element is displayed upon successful navigation.
    if (successMessage.isDisplayed()) {
        System.out.println("Test Passed: Successfully logged in and profile updated.");
    } else {
        System.out.println("Test Failed.");
    }
  }

}
```

Output



```
Dynamically enable window occlusion 0
1698073931448   Marionette      INFO    Listening on port 12968
WebDriver BiDi listening on ws://127.0.0.1:19678
Read port: 12968
1698073932355   RemoteAgent     WARN    TLS certificate errors will be ignored for this session
DevTools listening on ws://127.0.0.1:19678/devtools/browser/27421402-b845-484f-900c-6c3ccd03eb20
  Given the browser is open3                            # Nest.edit_profile.browser_is_open()
  And the user is on the login page3                    # Nest.edit_profile.user_is_on_login_page()
  When the user enters their email and password3        # Nest.edit_profile.the_user_enters_credentials()
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
JavaScript error: http://127.0.0.1:8000/, line 1037: TypeError: m.kommunicate is undefined
  And the user clicks on the login button3              # Nest.edit_profile.user_clicks_on_login()
JavaScript error: http://127.0.0.1:8000/register/editprofile/, line 1: ReferenceError: validateCountry is not defined
JavaScript error: http://127.0.0.1:8000/register/editprofile/, line 1: ReferenceError: validateCountry is not defined
JavaScript error: http://127.0.0.1:8000/register/editprofile/, line 1: ReferenceError: validateCountry is not defined
JavaScript error: http://127.0.0.1:8000/register/editprofile/, line 1: ReferenceError: validateCountry is not defined
JavaScript error: http://127.0.0.1:8000/register/editprofile/, line 1: ReferenceError: validateCountry is not defined
JavaScript error: http://127.0.0.1:8000/register/editprofile/, line 1: ReferenceError: validateState is not defined
JavaScript error: http://127.0.0.1:8000/register/editprofile/, line 1: ReferenceError: validateState is not defined
JavaScript error: http://127.0.0.1:8000/register/editprofile/, line 1: ReferenceError: validateState is not defined
JavaScript error: http://127.0.0.1:8000/register/editprofile/, line 1: ReferenceError: validateState is not defined
JavaScript error: http://127.0.0.1:8000/register/editprofile/, line 1: ReferenceError: validateState is not defined
JavaScript error: http://127.0.0.1:8000/register/editprofile/, line 1: ReferenceError: validateState is not defined
JavaScript error: http://127.0.0.1:8000/register/editprofile/, line 1: ReferenceError: checkall is not defined
JavaScript error: http://127.0.0.1:8000/register/editprofile/, line 885: ReferenceError: twonError is not defined
Test Passed: Successfully logged in and profile updated.
  Then the user should be navigated to the home page3       # Nest.edit_profile.navigate_to_home_page()

1 Scenarios (1 passed)
5 Steps (5 passed)
0m27.529s
```

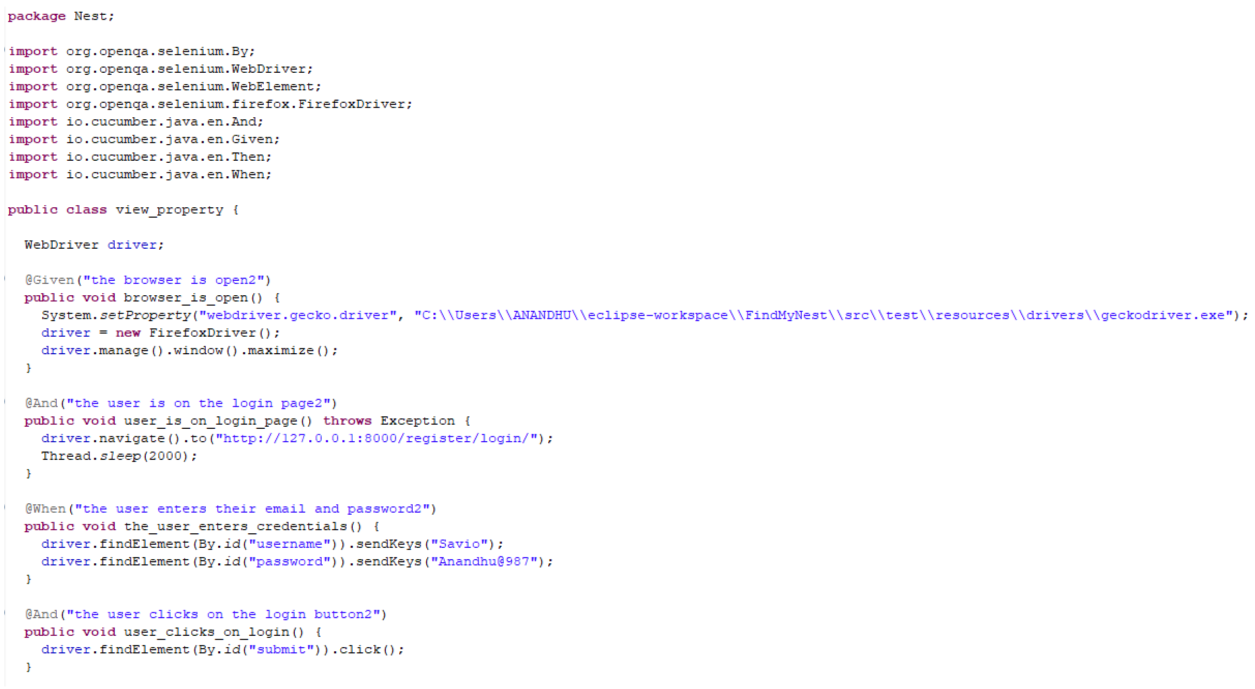
Test report

Test Case 2

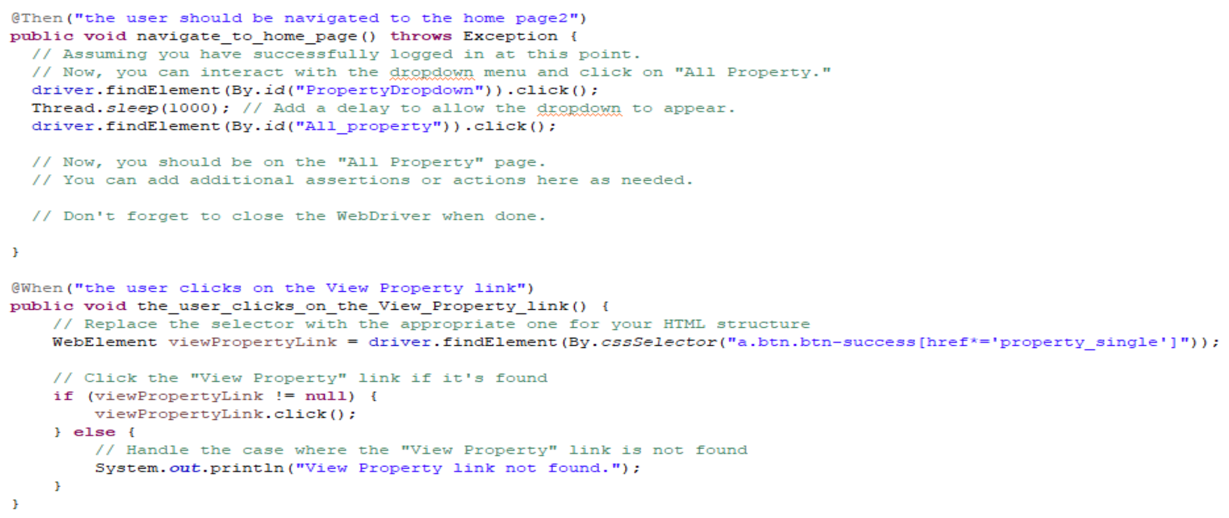
Project Name:FindMyNest					
Profile Update Test Case					
Test Case ID: 2			**Test Designed By:** Anandhu Biju		
Test Priority(Low/Medium/High): High			**Test Designed Date:** 23/10/2023		
Module Name: Update profile			**Test Executed By :** Ms. Sruthimol Kurian		
Test Title : Verify update profile			**Test Execution Date:** 23/10/2023		
Description: Testing the update profile page					
Pre-Condition : User has valid username and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Navigation to profile Page		Profile page should be displayed	Profile page displayed	Pass
2	Provide valid country	Country: India	User should be able to update profile	Profile updated and message should be displayed.	Pass
3	Provide State	State: Kerala			
4	Provide town	town: Kollam			
Post-Condition: User is inserted into the database and successfully registered. Profile details updated and inserted successfully into database					

Test Case 3:

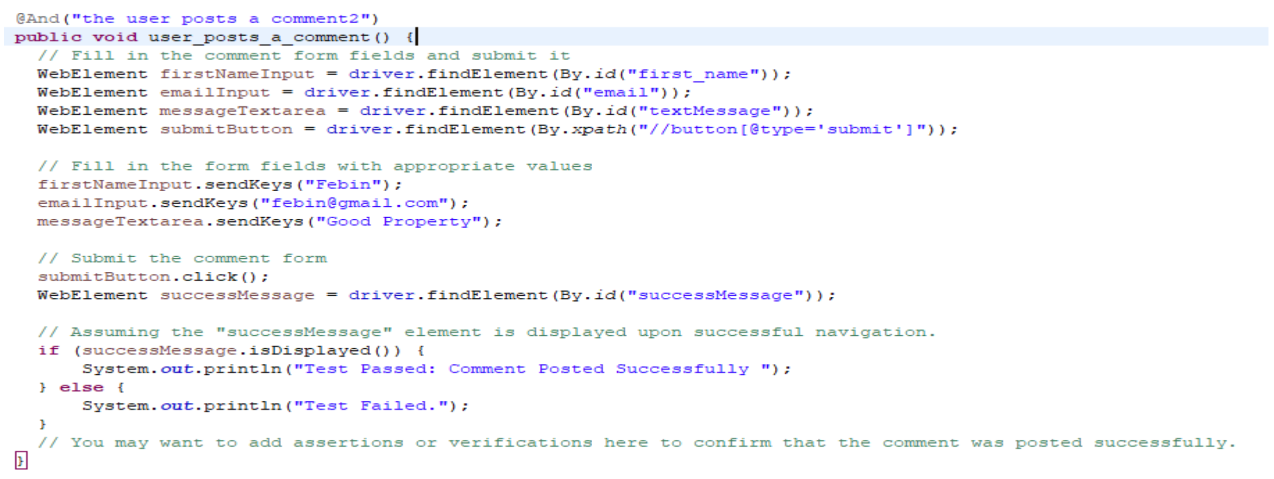
Code



```
package Nest;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import io.cucumber.java.en.And;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;

public class view_property {

  WebDriver driver;

  @Given("the browser is open2")
  public void browser_is_open() {
    System.setProperty("webdriver.gecko.driver", "C:\\Users\\ANANDHU\\eclipse-workspace\\FindMyNest\\src\\test\\resources\\drivers\\geckodriver.exe");
    driver = new FirefoxDriver();
    driver.manage().window().maximize();
  }

  @And("the user is on the login page2")
  public void user_is_on_login_page() throws Exception {
    driver.navigate().to("http://127.0.0.1:8000/register/login/");
    Thread.sleep(2000);
  }

  @When("the user enters their email and password2")
  public void the_user_enters_credentials() {
    driver.findElement(By.id("username")).sendKeys("Savio");
    driver.findElement(By.id("password")).sendKeys("Anandhu@987");
  }

  @And("the user clicks on the login button2")
  public void user_clicks_on_login() {
    driver.findElement(By.id("submit")).click();
  }
```

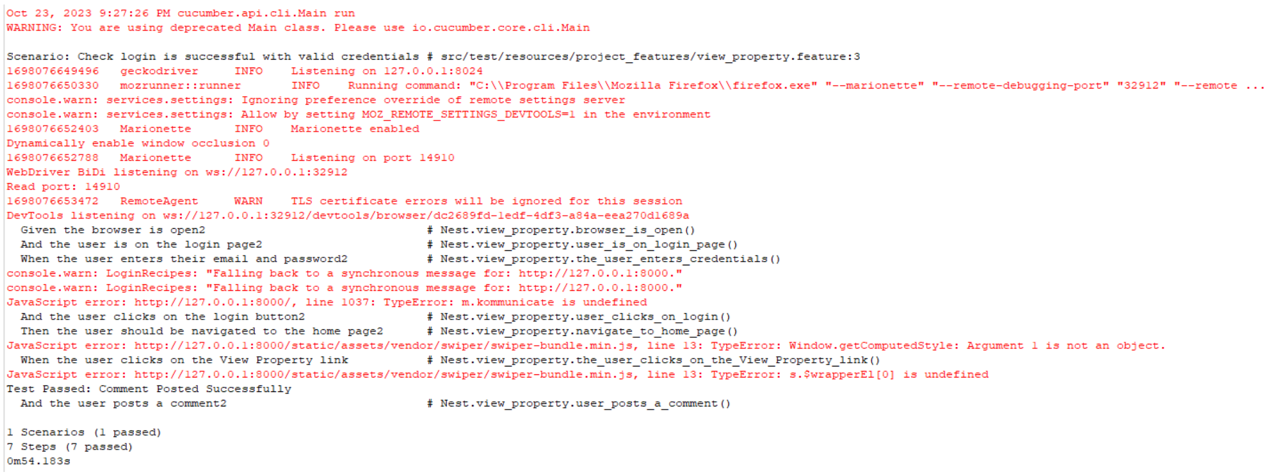


```
  @Then("the user should be navigated to the home page2")
  public void navigate_to_home_page() throws Exception {
    // Assuming you have successfully logged in at this point.
    // Now, you can interact with the dropdown menu and click on "All Property."
    driver.findElement(By.id("PropertyDropdown")).click();
    Thread.sleep(1000); // Add a delay to allow the dropdown to appear.
    driver.findElement(By.id("All_property")).click();

    // Now, you should be on the "All Property" page.
    // You can add additional assertions or actions here as needed.

    // Don't forget to close the WebDriver when done.

  }

  @When("the user clicks on the View Property link")
  public void the_user_clicks_on_the_View_Property_link() {
      // Replace the selector with the appropriate one for your HTML structure
      WebElement viewPropertyLink = driver.findElement(By.cssSelector("a.btn.btn-success[href*='property_single']"));

      // Click the "View Property" link if it's found
      if (viewPropertyLink != null) {
          viewPropertyLink.click();
      } else {
          // Handle the case where the "View Property" link is not found
          System.out.println("View Property link not found.");
      }
  }
```



```
 @And("the user posts a comment2")
 public void user_posts_a_comment() {
   // Fill in the comment form fields and submit it
   WebElement firstNameInput = driver.findElement(By.id("first_name"));
   WebElement emailInput = driver.findElement(By.id("email"));
   WebElement messageTextarea = driver.findElement(By.id("textMessage"));
   WebElement submitButton = driver.findElement(By.xpath("//button[@type='submit']"));

   // Fill in the form fields with appropriate values
   firstNameInput.sendKeys("Febin");
   emailInput.sendKeys("febin@gmail.com");
   messageTextarea.sendKeys("Good Property");

   // Submit the comment form
   submitButton.click();
   WebElement successMessage = driver.findElement(By.id("successMessage"));

   // Assuming the "successMessage" element is displayed upon successful navigation.
   if (successMessage.isDisplayed()) {
       System.out.println("Test Passed: Comment Posted Successfully ");
   } else {
       System.out.println("Test Failed.");
   }
   // You may want to add assertions or verifications here to confirm that the comment was posted successfully.
 }
```

Output



```
Oct 23, 2023 9:27:26 PM cucumber.api.cli.Main run
WARNING: You are using deprecated Main class. Please use io.cucumber.core.cli.Main

Scenario: Check login is successful with valid credentials # src/test/resources/project_features/view_property.feature:3
1698076649496   geckodriver     INFO    Listening on 127.0.0.1:8024
1698076650330   mozrunner::runner       INFO    Running command: "C:\\Program Files\\Mozilla Firefox\\firefox.exe" "--marionette" "--remote-debugging-port" "32912" "--remote ...
console.warn: services.settings: Ignoring preference override of remote settings server
console.warn: services.settings: Allow by setting MOZ_REMOTE_SETTINGS_DEVTOOLS=1 in the environment
1698076652403   Marionette      INFO    Marionette enabled
Dynamically enable window occlusion 0
1698076652788   Marionette      INFO    Listening on port 14910
WebDriver BiDi listening on ws://127.0.0.1:32912
Read port: 14910
1698076653472   RemoteAgent     WARN    TLS certificate errors will be ignored for this session
DevTools listening on ws://127.0.0.1:32912/devtools/browser/dc2689fd-1edf-4df3-a84a-eea270d1689a
  Given the browser is open2                               # Nest.view_property.browser_is_open()
  And the user is on the login page2                       # Nest.view_property.user_is_on_login_page()
  When the user enters their email and password2           # Nest.view_property.the_user_enters_credentials()
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
JavaScript error: http://127.0.0.1:8000/, line 1037: TypeError: m.kommunicate is undefined
  And the user clicks on the login button2                 # Nest.view_property.user_clicks_on_login()
  Then the user should be navigated to the home page2      # Nest.view_property.navigate_to_home_page()
JavaScript error: http://127.0.0.1:8000/static/assets/vendor/swiper/swiper-bundle.min.js, line 13: TypeError: Window.getComputedStyle: Argument 1 is not an object.
  When the user clicks on the View Property link           # Nest.view_property.the_user_clicks_on_the_View_Property_link()
JavaScript error: http://127.0.0.1:8000/static/assets/vendor/swiper/swiper-bundle.min.js, line 13: TypeError: s.$wrapperEl[0] is undefined
Test Passed: Comment Posted Successfully
  And the user posts a comment2                            # Nest.view_property.user_posts_a_comment()

1 Scenarios (1 passed)
7 Steps (7 passed)
0m54.183s
```

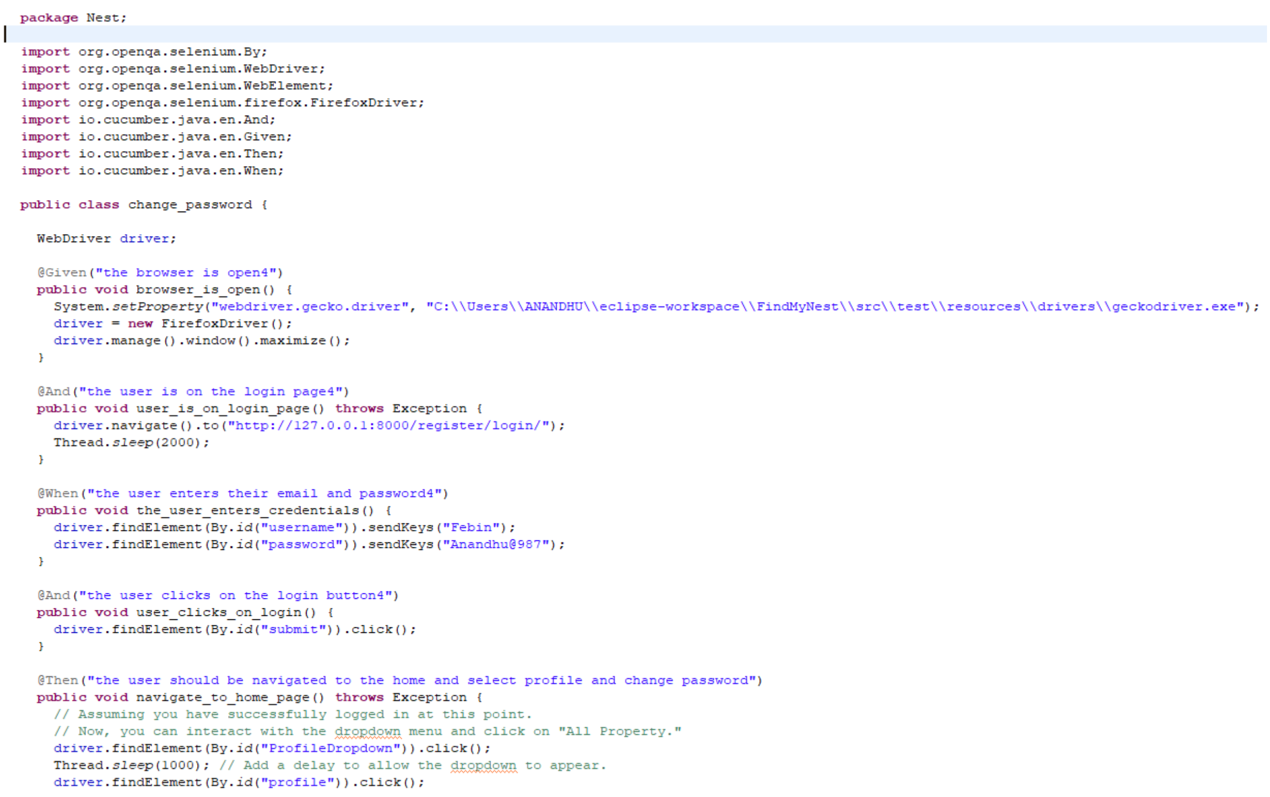
Test report

Test Case 3

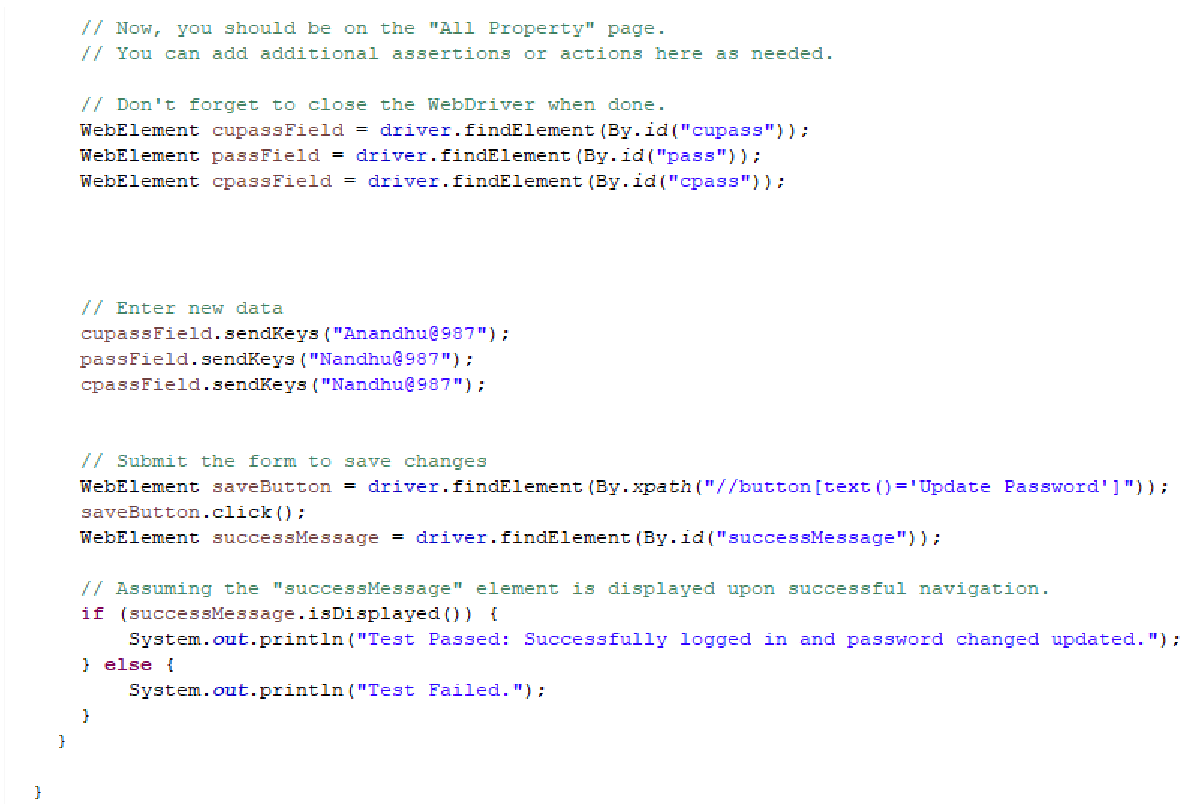
Project Name:FindMyNest					
Post CommentTest Case					
Test Case ID: 3			**Test Designed By:** Anandhu Biju		
Test Priority(Low/Medium/High): High			**Test Designed Date:** 23/10/2023		
Module Name: Update profile			**Test Executed By :** Ms. Sruthimol Kurian		
Test Title : Verify Post Comment			**Test Execution Date:** 23/10/2023		
Description: Testing the Post Comment page					
Pre-Condition : User has valid username and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Navigation to Property Single page		Property Single page should be displayed	Property Single page displayed	Pass
2	Provide valid first_name	first_name: Febin	User should be able to post comment	Comment posted and comment displayed.	Pass
3	Provide email	email: febin@gmail.com			
4	Provide textMessage	textMessage: Good Property			
Post-Condition: Comment posted and inserted successfully into database					

Test Case 4:

Code

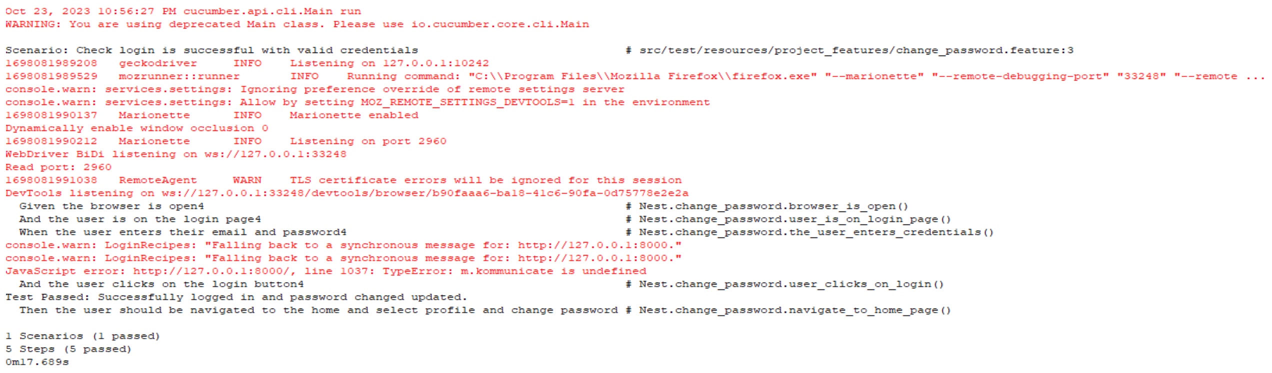


```
package Nest;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import io.cucumber.java.en.And;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;

public class change_password {

  WebDriver driver;

  @Given("the browser is open4")
  public void browser_is_open() {
    System.setProperty("webdriver.gecko.driver", "C:\\Users\\ANANDHU\\eclipse-workspace\\FindMyNest\\src\\test\\resources\\drivers\\geckodriver.exe");
    driver = new FirefoxDriver();
    driver.manage().window().maximize();
  }

  @And("the user is on the login page4")
  public void user_is_on_login_page() throws Exception {
    driver.navigate().to("http://127.0.0.1:8000/register/login/");
    Thread.sleep(2000);
  }

  @When("the user enters their email and password4")
  public void the_user_enters_credentials() {
    driver.findElement(By.id("username")).sendKeys("Febin");
    driver.findElement(By.id("password")).sendKeys("Anandhu@987");
  }

  @And("the user clicks on the login button4")
  public void user_clicks_on_login() {
    driver.findElement(By.id("submit")).click();
  }

  @Then("the user should be navigated to the home and select profile and change password")
  public void navigate_to_home_page() throws Exception {
    // Assuming you have successfully logged in at this point.
    // Now, you can interact with the dropdown menu and click on "All Property."
    driver.findElement(By.id("ProfileDropdown")).click();
    Thread.sleep(1000); // Add a delay to allow the dropdown to appear.
    driver.findElement(By.id("profile")).click();
```



```
      // Now, you should be on the "All Property" page.
      // You can add additional assertions or actions here as needed.

      // Don't forget to close the WebDriver when done.
      WebElement cupassField = driver.findElement(By.id("cupass"));
      WebElement passField = driver.findElement(By.id("pass"));
      WebElement cpassField = driver.findElement(By.id("cpass"));



      // Enter new data
      cupassField.sendKeys("Anandhu@987");
      passField.sendKeys("Nandhu@987");
      cpassField.sendKeys("Nandhu@987");


      // Submit the form to save changes
      WebElement saveButton = driver.findElement(By.xpath("//button[text()='Update Password']"));
      saveButton.click();
      WebElement successMessage = driver.findElement(By.id("successMessage"));

      // Assuming the "successMessage" element is displayed upon successful navigation.
      if (successMessage.isDisplayed()) {
          System.out.println("Test Passed: Successfully logged in and password changed updated.");
      } else {
          System.out.println("Test Failed.");
      }
    }

  }
```

Output



```
Oct 23, 2023 10:56:27 PM cucumber.api.cli.Main run
WARNING: You are using deprecated Main class. Please use io.cucumber.core.cli.Main

Scenario: Check login is successful with valid credentials                    # src/test/resources/project_features/change_password.feature:3
1698081989208   geckodriver     INFO    Listening on 127.0.0.1:10242
1698081989529   mozrunner::runner       INFO    Running command: "C:\\Program Files\\Mozilla Firefox\\firefox.exe" "--marionette" "--remote-debugging-port" "33248" "--remote ...
console.warn: services.settings: Ignoring preference override of remote settings server
console.warn: services.settings: Allow by setting MOZ_REMOTE_SETTINGS_DEVTOOLS=1 in the environment
1698081990137   Marionette      INFO    Marionette enabled
Dynamically enable window occlusion 0
1698081990212   Marionette      INFO    Listening on port 2960
WebDriver BiDi listening on ws://127.0.0.1:33248
Read port: 2960
1698081991038   RemoteAgent     WARN    TLS certificate errors will be ignored for this session
DevTools listening on ws://127.0.0.1:33248/devtools/browser/b90faaa6-ba18-41c6-90fa-0d75778e2e2a
  Given the browser is open4                                                  # Nest.change_password.browser_is_open()
  And the user is on the login page4                                          # Nest.change_password.user_is_on_login_page()
  When the user enters their email and password4                              # Nest.change_password.the_user_enters_credentials()
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
JavaScript error: http://127.0.0.1:8000/, line 1037: TypeError: m.kommunicate is undefined
  And the user clicks on the login button4                                    # Nest.change_password.user_clicks_on_login()
Test Passed: Successfully logged in and password changed updated.
  Then the user should be navigated to the home and select profile and change password # Nest.change_password.navigate_to_home_page()

1 Scenarios (1 passed)
5 Steps (5 passed)
0m17.689s
```

Test report

Test Case 4

Project Name: FindMyNest	
Password Change Test Case	
Test Case ID: 4	**Test Designed By:** Anandhu Biju
Test Priority(Low/Medium/High): High	**Test Designed Date:** 23/10/2023
Module Name: Change password	**Test Executed By :** Ms. Sruthimol Kurian
Test Title : Verify change passwsord	**Test Execution Date:** 23/10/2023
Description: Testing change password page	
Pre-Condition : User has valid username and password	

Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Navigation to profile page		Profile page should be displayed	Profile page displayed	Pass
2	Provide old password	Anandhu@987	User should be able Change Password	Password Must Change	Pass
4	Provide new password	Nandhu@987			
5	Provide confirm password	Jans@123			

Post-Condition: Password Change and inserted successfully into database	

CHAPTER 6
IMPLEMENTATION

6.1INTRODUCTION

Implementation is the phase of the task where the hypothetical plan is transformed into a functioning framework. It very well may be viewed as the most urgent stage in accomplishing a fruitful new framework acquiring the client’s certainty that the new framework will work and will be compelling and exact. It is basically worried about client preparing and documentation. Change for the most part happens about a similar time the client is being prepared or later. Execution essentially implies meeting another framework plan into activity, which is the most common way of changing over another reconsidered framework plan into a functional one. At this stage the principal responsibility, the best disturbance and the significant effect on the current framework movements to the client office. In the event that the execution isn't painstakingly arranged or controlled, it can make disarray and disarray.

Execution incorporates that multitude of exercises that occur to change over from the current framework to the new framework. The new framework might be an absolutely new, supplanting a current manual or computerized framework or it could be a change to a current framework. Legitimate execution is fundamental to give a dependable framework to meet association necessities. The most common way of placing the created framework in genuine use is called framework execution. This incorporates that multitude of exercises that happen to change over from the old framework to the new framework. The framework can be carried out solely after through testing is finished and assuming that it is viewed as working as per the details. The framework work force actually looks at the practicality of the framework. The more perplexing the framework being executed, the more elaborate will be the framework examination and plan exertion expected to carry out the three principal perspectives: instruction and preparing, framework testing and changeover.

6.2 IMPLEMENTATION PROCEDURES

Implementation of software refers to the last establishment of the bundle in its genuine climate, as per the general inclination of the expected purposes and the activity of the framework. In numerous associations somebody who won't be working it, will commission the product improvement project. In the underlying stage individuals uncertainty about the product yet we need to guarantee that the opposition doesn't develop, as one needs to ensure that: The dynamic client should know about the advantages of utilizing the new system. Their trust in the software is developed. Legitimate direction is granted to the client with the goal that he is agreeable in utilizing the application. Before going ahead and viewing the system, the user must know that for viewing the result, the server program should be running in the server. If the server object is not up running on the server, the actual process won’t take place

6.2.1 User Training

The goal of user training is to give users the confidence to test and alter computer-based systems in order to finally accomplish the intended goals. Training becomes more important as systems get more complicated. As part of the user training process, participants are exposed to a variety of crucial tasks like data entering, handling error alerts, querying databases, calling up routines to generate reports, among others

6.2.2 Training on the Application Software

The new application software requires users to first complete a basic computer literacy training course before utilising it. They should be shown how to access help resources, traverse the software panels, correct entry errors, and update data that has already been entered. Specific topics pertaining to the user group and their function in using the system or its components should also be covered in the training. The training for the programme should be adapted to the requirements of various user groups and hierarchical levels

6.2.3 System Maintenance

The software maintenance phase is an essential part of the software development life cycle and occurs after a software product has been successfully implemented and is performing useful work. Maintenance is necessary to ensure that the system remains adaptable to changes in the system environment. While finding mistakes is a part of software maintenance, it is not the only task involved. There are many other activities involved in maintenance, such as updating documentation, fixing bugs, enhancing existing features, and adding new features. Effective maintenance can help ensure that the software remains functional, reliable, and efficient over time.

CHAPTER 7
CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

FindMyNest is a multifaceted platform that has revolutionized the way individuals discover and manage rental properties. This comprehensive solution offers a rich set of features, catering to both property owners and prospective tenants. Through its intuitive interface and advanced search capabilities, FindMyNest simplifies the property search process, ensuring that users can easily find the ideal rental property to meet their specific needs. For property owners, FindMyNest provides an efficient and user-friendly platform to list their properties. It enables property owners to showcase their offerings to a wide audience, enhancing their property's visibility in the competitive rental market. The platform also streamlines the rental application and transaction processes, providing a secure environment for both owners and tenants. One of the standout features of FindMyNest is its location-based searching and personalized recommendations. This ensures that users can find properties in areas that are most suitable for their preferences and needs. Moreover, the inclusion of a chatbot feature offers instant assistance to users, enhancing the overall user experience. The order tracking system is a valuable addition, allowing tenants to monitor the status of their rental applications and providing transparency in the application process. FindMyNest goes above and beyond by incorporating a translator, making it accessible to a diverse user base, and a fertilizer recommendation module to assist users in maintaining their rented properties.

In conclusion, FindMyNest is a comprehensive and user-centric platform that has transformed the way we search for and manage rental properties. With its array of features and commitment to user satisfaction, it sets a new standard in the rental property market, making it a valuable tool for both property owners and tenants alike.

7.2 FUTURE SCOPE

In the near future, FindMyNest is set to expand its horizons with a host of exciting developments. These may include personalized property recommendations, a seamless booking and payment system, user reviews for transparency, and a mobile app for convenient access on the go. Enhanced property management tools for owners and wider geographic coverage are also on the radar, all fortified with robust security measures.

Sustainability features and immersive virtual property tours could make their debut, and potential collaborations with real estate agencies may result in an even broader array of listings. Moreover, the pipeline includes plans for a community forum, AI chat support, and a payout module for owners, ensuring a comprehensive and user-friendly experience for all.

FindMyNest envisions an exciting future in the real estate landscape. In addition to the aforementioned enhancements, the platform is exploring AI-powered predictive analytics to provide users with insights into property trends, helping them make informed decisions. An augmented reality (AR) feature may allow users to virtually stage and customize properties, making it easier to visualize their dream homes.

.

CHAPTER 8
BIBLIOGRAPHY

REFERENCES:

- Gary B. Shelly, Harry J. Rosenblatt, *"System Analysis and Design",* 2009
- Roger S Pressman, *"Software Engineering"*, 1994.
- PankajJalote, *"Software engineering: a precise approach",* 2006
- EEE Std 1016 Recommended Practice for Software Design Descriptions

WEBSITES:

- https:/stackoverflow.com/
- www.w3schools.com
- www.agilemodeling.com/artifacts/useCaseDiagram.html
-

CHAPTER 9

APPENDIX

9.1 Sample Code

Views.py

```
from django.db.models import Count
from django.contrib.auth.models import User
from django.shortcuts import render,redirect
from django.contrib.auth import login as auth_login ,authenticate,update_session_auth_hash
from django.contrib.auth import authenticate, login as auth_login
from django.utils.crypto import get_random_string
from django.core.mail import send_mail
from .manager import UserManager
from django.shortcuts import render, get_object_or_404
from django.contrib  import messages,auth
from .models import CustomUser,UserProfile
from FindMyNestApp.models import Subscription
from Customer.models import Property
from FindMyNestApp.models import Payment
from django.contrib.auth import get_user_model
from django.http import HttpResponseRedirect
from django.urls import reverse
import json
import requests
import matplotlib.pyplot as plt
from social_django.utils import psa
from django.core.exceptions import ObjectDoesNotExist
from django.contrib.auth.decorators import login_required
from django.http import JsonResponse
from django.http import HttpResponse
from django.views.decorators.csrf import csrf_exempt

User = get_user_model()


def login(request):
    # Check if the user is already authenticated
    if request.user.is_authenticated:
        if request.user.user_type == CustomUser.ADMIN:
            return redirect(reverse('admindashboard'))
        elif request.user.user_type == CustomUser.CUSTOMER:
            return redirect(reverse('index'))
        else:
            return redirect('/')

    if request.method == 'POST':
        captcha_token = request.POST.get("g-recaptcha-response")
        cap_url = "https://www.google.com/recaptcha/api/siteverify"
        cap_secret = "6LcJj44nAAAAADDjTqz0n5e7UM5HRFzMtC54swC3"
        cap_data = {"secret": cap_secret, "response": captcha_token}

        cap_server_response = requests.post(url=cap_url, data=cap_data)
        cap_json = json.loads(cap_server_response.text)

        if not cap_json['success']:
            return HttpResponseRedirect(reverse('login') + '?alert=invalid_captcha')
        username = request.POST.get('username')
        password = request.POST.get('password')

        if username and password:
            user = authenticate(request, username=username, password=password)

            if user is not None:
                auth_login(request, user)

                if request.user.user_type == CustomUser.ADMIN:
                    return redirect(reverse('admindashboard'))
                elif request.user.user_type == CustomUser.CUSTOMER:
                    return redirect(reverse('index'))
                else:
                    return redirect('/')
            else:
                return HttpResponseRedirect(reverse('login') + '?alert=invalid_credentials')
        else:
            return HttpResponseRedirect(reverse('login') + '?alert=fill_fields')
    return render(request, 'accounts/google/login.html')

def register(request):
    if request.method == 'POST':
        username = request.POST.get('username', None)
        first_name = request.POST.get('first_name', None)
        last_name = request.POST.get('last_name', None)
        email = request.POST.get('email', None)
        phone = request.POST.get('phone', None)
        password = request.POST.get('password', None)
        cpassword = request.POST.get('confirmPassword', None)

        if username and first_name and last_name and email and phone and password:

            if User.objects.filter(username=username).exists():
                return HttpResponseRedirect(reverse('register') + '?alert=username_is_already_registered')

            elif User.objects.filter(email=email).exists():
                return HttpResponseRedirect(reverse('register') + '?alert=email_is_already_registered')

            elif User.objects.filter(phone_no=phone).exists():
                return HttpResponseRedirect(reverse('register') + '?alert=phone_no_is_already_registered')

            elif password != cpassword:
                return HttpResponseRedirect(reverse('register') + '?alert=passwords_do_not_match')



            else:
                user = User(username=username, first_name=first_name, last_name=last_name, email=email,
phone_no=phone)
                user.set_password(password)  # Set the password securely
                user.save()

                user_profile = UserProfile(user=user)
                user_profile.save()
            return HttpResponseRedirect(reverse('login') + '?alert=registered')
    return render(request, 'register.html')

def addproperty(request):
    if request.method == 'POST':
        form = PropertyForm(request.POST, request.FILES)
        if form.is_valid():
            property_instance = form.save(commit=False)
            property_instance.user = request.user

            selected_features = form.cleaned_data['features']
            features_str = ', '.join(selected_features)
            property_instance.features = features_str

            selected_nearby_place = form.cleaned_data['nearby_place']
            nearby_place_str = ', '.join(selected_nearby_place)
            property_instance.nearby_place = nearby_place_str

            distance_to_major_road = input_and_average(form.cleaned_data['major_road'])
            distance_to_supermarket = input_and_average(form.cleaned_data['near_supermarket'])
            property_age = input_and_average(form.cleaned_data['bulding_age'])

            last_renovation_years_ago_input = form.cleaned_data['last_renovated']
            last_renovation_years_ago_parts = last_renovation_years_ago_input.split('-')
            last_renovation_years_ago = (int(last_renovation_years_ago_parts[0]) +
int(last_renovation_years_ago_parts[1])) // 2

            property_type = form.cleaned_data['property_type']

            if property_type == 'Apartment':
                floor_value = form.cleaned_data['floor_no']
            else:
                floor_value = form.cleaned_data['floor']

            if property_type in ['Commercial', 'Garage']:
                num_bedrooms = 0
            else:
                num_bedrooms = form.cleaned_data['bedrooms']

            furnished = 1 if 'Furnished' in selected_features else 0
            air_conditioner = 1 if 'Air Condition' in selected_features else 0
            parking = 1 if 'Parking' in selected_features else 0
            water_available = 1 if 'Well(Water Availability)' in selected_features else 0

            user_input_dict = {
                'property_type': property_type,
                'num_bedrooms': num_bedrooms,
                'num_bathrooms': form.cleaned_data['bathrooms'],
                'furnished': furnished,
                'air_conditioner': air_conditioner,
                'parking': parking,
                'last_renovation_years_ago': last_renovation_years_ago,
                'water_available': water_available,
                'distance_to_major_road': [distance_to_major_road],
                'distance_to_supermarket': [distance_to_supermarket],
                'price': form.cleaned_data['price'],
                'property_age': [property_age],
                'total_rooms': form.cleaned_data['rooms'],
                'floor': floor_value
            }

            user_input_df = pd.DataFrame(user_input_dict)

            for col, le in label_encoders.items():
                user_input_df[col] = le.transform(user_input_df[col])

            sale_duration_prediction = model.predict(user_input_df)

            property_instance.save()
            property_id = property_instance.id

            images = request.FILES.getlist('images')
            for image in images:
                Image.objects.create(property=property_instance, images=image)

            # Initialize property_tips
            property_tips = []

            # Save the exact number of days to sell
            exact_days_to_sell = int(sale_duration_prediction[0])
            property_instance.days_to_sell = exact_days_to_sell

            # Based on your business logic, you can generate property_tips
            property_tips = []

            if exact_days_to_sell > 30:
                property_tips = analyze_property_details(user_input_df.iloc[0])

            # Save property_tips
            property_instance.property_tips = ", ".join(property_tips)  # Join with commas

            sale_duration_tips = []

            # Generate sale_duration_tips
            if exact_days_to_sell <= 30:
                sale_duration_tips.append("Your property is likely to sell quickly. Ensure it's well-maintained for
a smooth sale.")
            elif exact_days_to_sell <= 60:
                sale_duration_tips.append("Your property may take a couple of months to sell. Consider staging
and effective marketing.")
            else:
                sale_duration_tips.append("Your property may take some time to sell. Focus on competitive
pricing and marketing strategies")

            # Set the sale_duration_tips field
            property_instance.sale_duration_tips = ", ".join(sale_duration_tips)  # Join with commas

            property_instance.save()

            # Redirect to the property details page with tips as URL parameters
```

```
            return redirect(reverse('property_single', args=[property_id]))
    else:
        form = PropertyForm()

    context = {
        'form': form,
    }
    return render(request, 'addproperty.html', context)


def analyze_property_details(property_details):
    tips = []

    # Example: Analyze property details like 'furnished,' 'air_conditioner,' 'parking,' etc.
    if property_details['furnished'] == 0:
        tips.append("Consider furnishing the property to attract more buyers/renters.")
    if property_details['air_conditioner'] == 0:
        tips.append("Adding air conditioning can make the property more appealing.")
    if property_details['parking'] == 0:
        tips.append("Providing parking can be a significant selling point.")
    if property_details['water_available'] == 0:
        tips.append("Ensure water availability for the property, as it is essential.")
    # Add tips related to renovation based on the last renovation years
    if property_details['last_renovation_years_ago'] >= 5:
        tips.append("Consider renovating the property if it hasn't been renovated in the last 5 years.")

    return tips


def add_wishlist(request, property_id):
    # Get the Property object based on the property_id
    property = get_object_or_404(Property, id=property_id)

    # Create a Wishlist object for the current user and the property
    if request.user.is_authenticated:
        wishlist, created = Wishlist.objects.get_or_create(user=request.user, property=property)
        if created:
            message = f'The property "{property.address}" has been added to your wishlist.'
        else:
            message = f'The property "{property.address}" is already in your wishlist.'
    else:
        # Handle the case where the user is not authenticated
        message = 'You need to be logged in to add properties to your wishlist.'

    # You can pass the message to your template or use it as needed
    # For now, we'll just redirect back to the propertylist view
    return redirect('propertylist')

def delete_wishlist(request, property_id):
    wishlist_item = get_object_or_404(Wishlist, property_id=property_id, user=request.user)
    wishlist_item.delete()
    return redirect('propertylist')

def wishlist_view(request):
    if request.user.is_authenticated:
        # Retrieve the wishlist items for the logged-in user
        wishlist_items = Wishlist.objects.filter(user=request.user)
        # Extract the properties from the wishlist items
        wishlist_properties = [item.property for item in wishlist_items]
        return render(request, 'wishlist.html', {'wishlist_properties': wishlist_properties})
    else:
        # Handle the case when the user is not logged in
        # You can redirect them to the login page or display a message
        return render(request, 'wishlist.html', {'wishlist_properties': None})


def payment(request, sub_id):
    # Use get_object_or_404 to get the Subscription object based on sub_id
        # Retrieve subscription features from a specific Subscription instance
    # You may want to retrieve a specific subscription
    subscriptions = Subscription.objects.all()

    sub_type = Subscription.objects.get(pk=sub_id)

    if sub_type:
        features_str = sub_type.features  # Get the 'features' field as a string
        features = features_str.split(',') if features_str else []
    else:
        features = []

    # Get unique property types
    property_types = Property.objects.values_list('property_type', flat=True).distinct()

    # For Razorpay integration
    currency = 'INR'
    amount = sub_type.price  # Get the subscription price
    amount_in_paise = int(amount * 100)  # Convert to paise

    # Create a Razorpay Order
    razorpay_order = razorpay_client.order.create(dict(
        amount=amount_in_paise,
        currency=currency,
        payment_capture='0'
    ))

    # Order ID of the newly created order
    razorpay_order_id = razorpay_order['id']
    callback_url = '/paymenthandler/'  # Define your callback URL here


    payment = Payment.objects.create(
        user=request.user,
        razorpay_order_id=razorpay_order_id,
        payment_id="",
        amount=amount,
        currency=currency,
        payment_status=Payment.PaymentStatusChoices.PENDING,
    )
    payment.sub_type.add(sub_type)
    # Prepare the context data
    context = {
        'user': request.user,
        'property_types': property_types,
        'razorpay_order_id': razorpay_order_id,
        'razorpay_merchant_key': settings.RAZOR_KEY_ID,
        'razorpay_amount': amount_in_paise,
        'currency': currency,
        'amount': amount_in_paise / 100,
        'callback_url': callback_url,
        'subscriptions': subscriptions,
        'features': features,  # Add features to the context
        'sub_type': sub_type,
    }

    return render(request, 'Payment.html', context)




def add_subscription(request):
    if request.method == 'POST':
        form = SubscriptionForm(request.POST)
        if form.is_valid():
            sub_type = form.cleaned_data['sub_type']
            price = form.cleaned_data['price']
            validity = form.cleaned_data['validity']
            features_list = form.cleaned_data['features'].split(',')
            features_csv = ','.join(features_list)


            if Subscription.objects.filter(
                sub_type=sub_type,
                price=price,
                validity=validity,
                features=features_csv
            ).exists():

                form.add_error(None, "A subscription with the same data already exists.")
            else:

                subscription = form.save(commit=False)
                subscription.features = features_csv
                subscription.save()
                return redirect('admindashboard')
    else:
        form = SubscriptionForm()

    return render(request, 'add_subscription.html', {'form': form})
 # Replace 'search.html' with your template

razorpay_client = razorpay.Client(
    auth=(settings.RAZOR_KEY_ID, settings.RAZOR_KEY_SECRET))

@csrf_exempt
```

```
def paymenthandler(request):
    # only accept POST request.
    if request.method == "POST":
        try:
            # get the required parameters from post request.
            payment_id = request.POST.get('razorpay_payment_id', '')
            razorpay_order_id = request.POST.get('razorpay_order_id', '')
            signature = request.POST.get('razorpay_signature', '')
            params_dict = {
                'razorpay_order_id': razorpay_order_id,
                'razorpay_payment_id': payment_id,
                'razorpay_signature': signature
            }

            # verify the payment signature.
            result = razorpay_client.utility.verify_payment_signature(params_dict)
            payment = Payment.objects.get(razorpay_order_id=razorpay_order_id)
            if result is not None:
                payment = Payment.objects.get(razorpay_order_id=razorpay_order_id)
                amount = int(payment.amount * 100)  # Convert Decimal to paise
                try:
                    # capture the payment
                    razorpay_client.payment.capture(payment_id, amount)
                    payment = Payment.objects.get(razorpay_order_id=razorpay_order_id)

                    # Update the order with payment ID and change status to "Successful"
                    payment.payment_id = payment_id
                    payment.payment_status = Payment.PaymentStatusChoices.SUCCESSFUL
                    payment.save()

                    # Send the welcome email with PDF invoice
                    send_welcome_email(payment.user.username, payment.sub_type, payment.amount,
payment.user.email, payment,)

                    # render success page on successful capture of payment
                    return render(request, 'index.html')
                except:
                    # if there is an error while capturing payment.
                    payment.payment_status = Payment.PaymentStatusChoices.FAILED
                    return render(request, 'paymentfail.html')
            else:
                # if signature verification fails.
                payment.payment_status = Payment.PaymentStatusChoices.FAILED
                return render(request, 'paymentfail.html')
        except:
            # if we don't find the required parameters in POST data
            return HttpResponseBadRequest()
    else:
        # if other than POST request is made.
        return HttpResponseBadRequest()



def send_welcome_email(username, sub_type, amount, email, payment):
    subject = 'Welcome to FindMyNest'
    message = f"Hello {username},\n\n"
    message += f"Welcome to FindMyNest, your platform for finding your dream property. We are excited
to have you join us!\n\n"

    # Retrieve the associated subscription object
    subscription = sub_type.first()  # Assuming sub_type is a ManyToMany field

    if subscription:
        message += f"You have subscribed to the {subscription.sub_type} plan, which is valid for
{subscription.validity}.\n\n"

    message += "Please feel free to contact the property owner for more information or to schedule a
viewing of the property.\n\n"
    message += "Thank you for choosing FindMyNest. We wish you the best in your property search!\n\n"
    message += "Warm regards,\nThe FindMyNest Team\n\n"

    from_email = 'findmynest.info@gmail.com'  # Replace with your actual email
    recipient_list = [email]

    # Create a PDF invoice and attach it to the email
    pdf_invoice = generate_pdf_invoice(username, sub_type, amount, payment)
    email = EmailMessage(subject, message, from_email, recipient_list)
    email.attach(f"{username}_invoice.pdf", pdf_invoice, 'application/pdf')

    # Send the email
    email.send()


def generate_pdf_invoice(username, sub_type, amount, payment):
    # Create a PDF document using xhtml2pdf
    template_path = "invoice1.html"  # Replace with the path to your HTML template
    response = HttpResponse(content_type='application/pdf')
    response['Content-Disposition'] = f'attachment; filename="{username}_invoice.pdf"'

    template = get_template(template_path)
    context = {
        'username': username,
        'sub_type': sub_type,
        'amount': amount,
        'payment': payment # Format the timestamp as desired
        }
    html = template.render(context)

    pdf_buffer = io.BytesIO()
    pisa.pisaDocument(io.BytesIO(html.encode("UTF-8")), pdf_buffer)

    pdf_content = pdf_buffer.getvalue()
    pdf_buffer.close()

    return pdf_content



@login_required
```

```
def search_property(request):
    query = request.GET.get('query')
    print("Query:", query)


    if query:
        properties = Property.objects.filter(
            Q(property_type__icontains=query) | Q(Town__icontains=query) | Q(state__icontains=query)
        )
    else:

        properties = []
    property_data = []

    for property in properties:

        property_dict = {
            'id': property.pk,
            'thumbnail': property.thumbnail.url,
            'address':  property.address,
            'property_type':  property.property_type,
            'bathrooms':  property.bathrooms,
            'bedrooms':  property.bedrooms,
            'price':  property.price,
            'area':  property.area,
        }
        property_data.append(property_dict)

    return JsonResponse({'property': property_data})
```

9.1 Screen Shots

Home Page

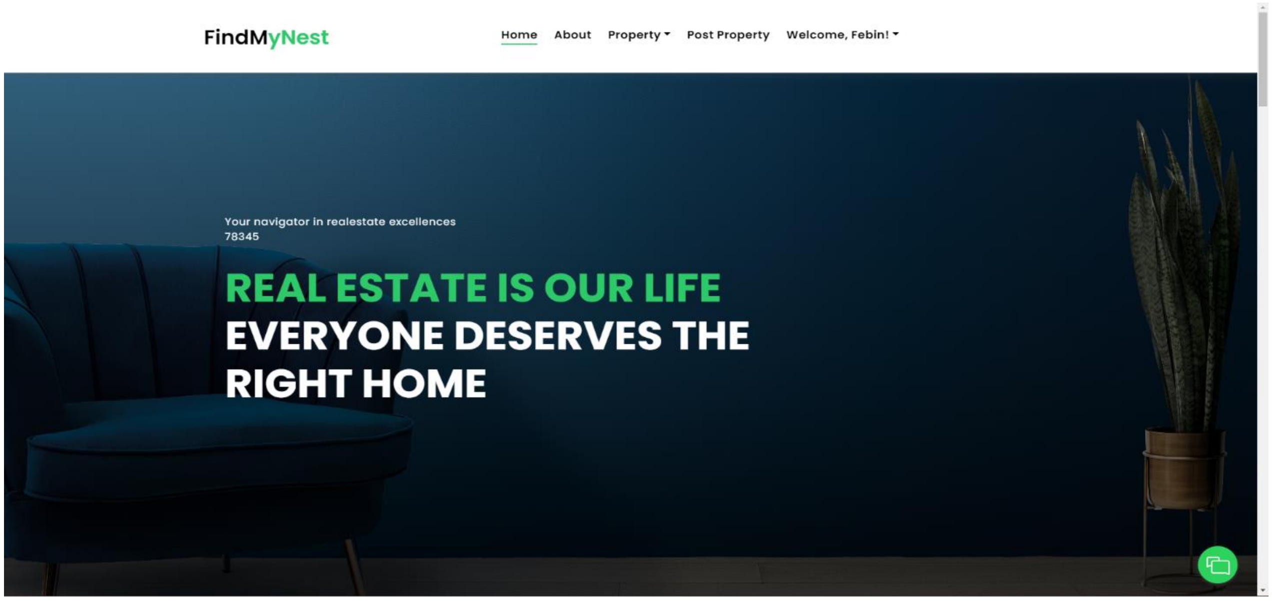
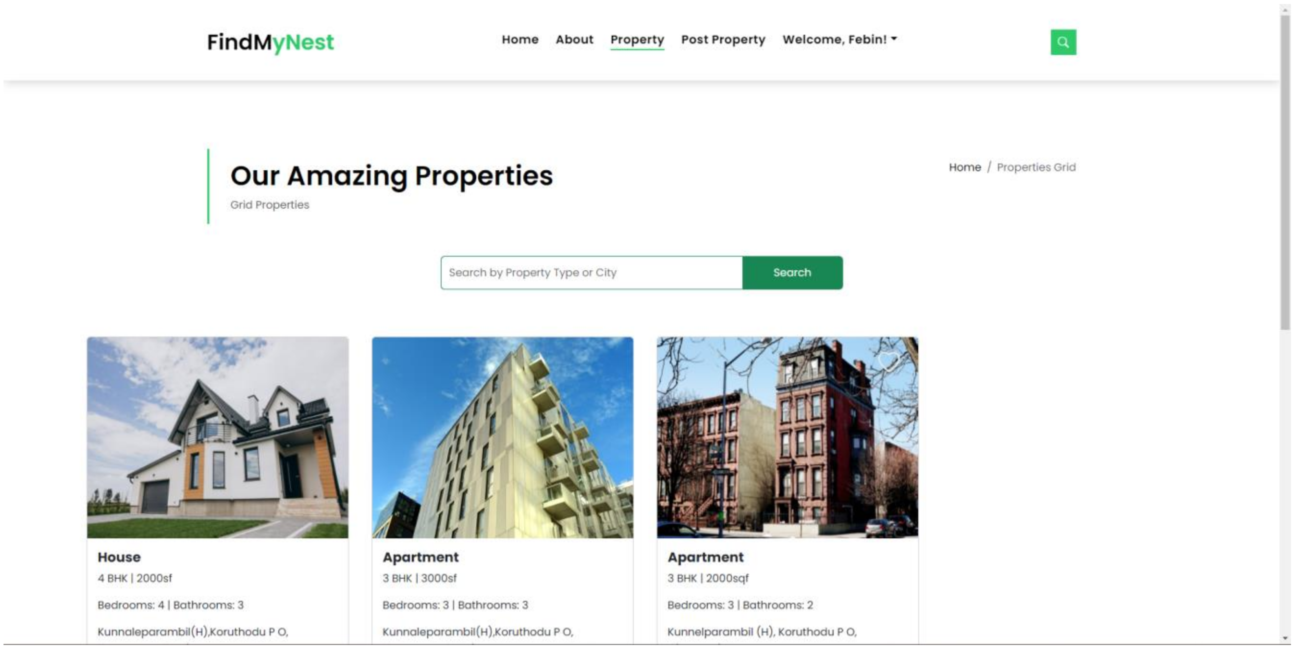


Figure 19: Home page

Property List Page



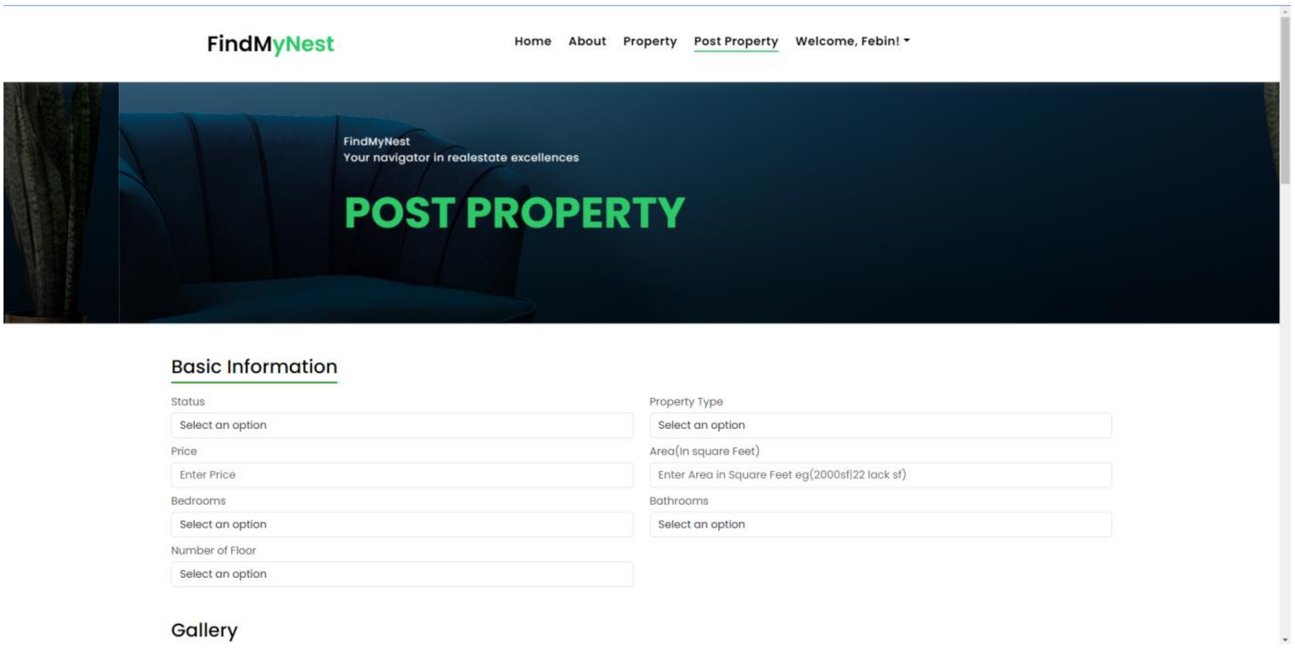
House
4 BHK | 2000sf
Bedrooms: 4 | Bathrooms: 3
Kunnaleparambil(H),Koruthodu P O,

Apartment
3 BHK | 3000sf
Bedrooms: 3 | Bathrooms: 3
Kunnaleparambil(H),Koruthodu P O,

Apartment
3 BHK | 2000sqf
Bedrooms: 3 | Bathrooms: 2
Kunnelparambil (H), Koruthodu P O,

Figure 20: Property List Page

Post Property Page



Basic Information

Status: Select an option	Property Type: Select an option
Price: Enter Price	Area(In square Feet): Enter Area in Square Feet eg(2000sf|22 lack sf)
Bedrooms: Select an option	Bathrooms: Select an option
Number of Floor: Select an option	

Gallery

Figure 21: Post Property

Registration Page

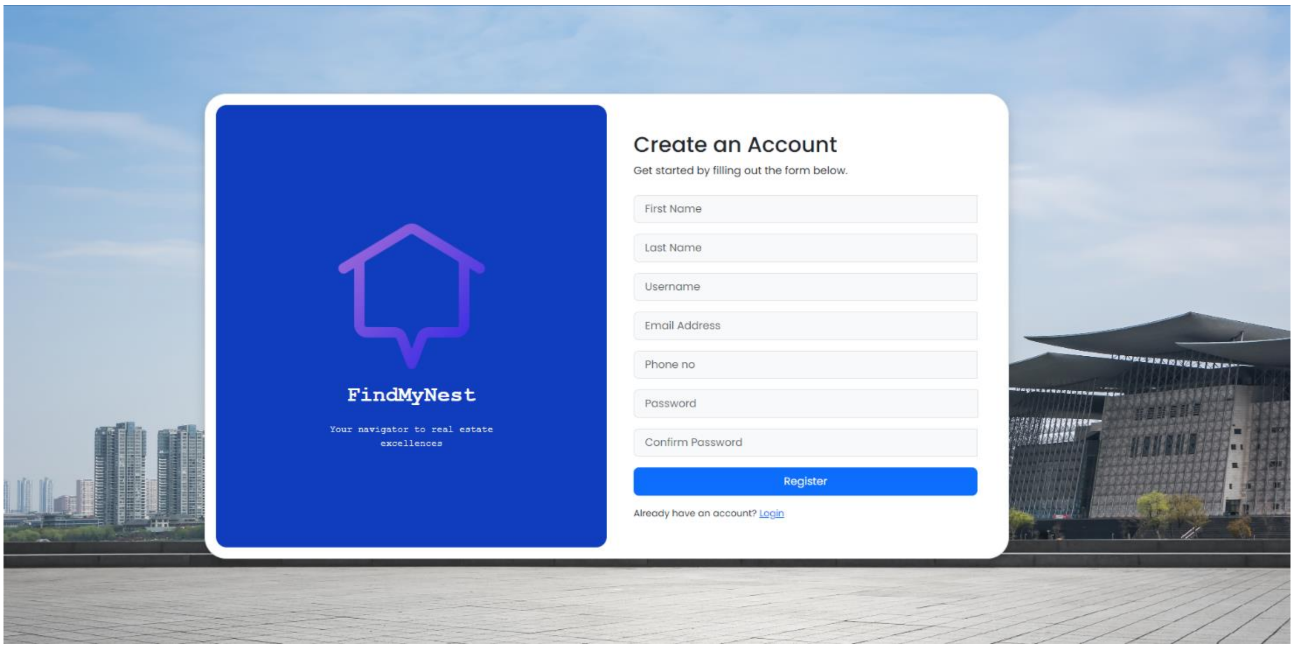

FindMyNest
Your navigator to real estate excellences
Create an Account
Get started by filling out the form below.
First Name
Last Name
Username
Email Address
Phone no
Password
Confirm Password
Register
Already have an account? Login

Figure 22: Registration page

Login Page

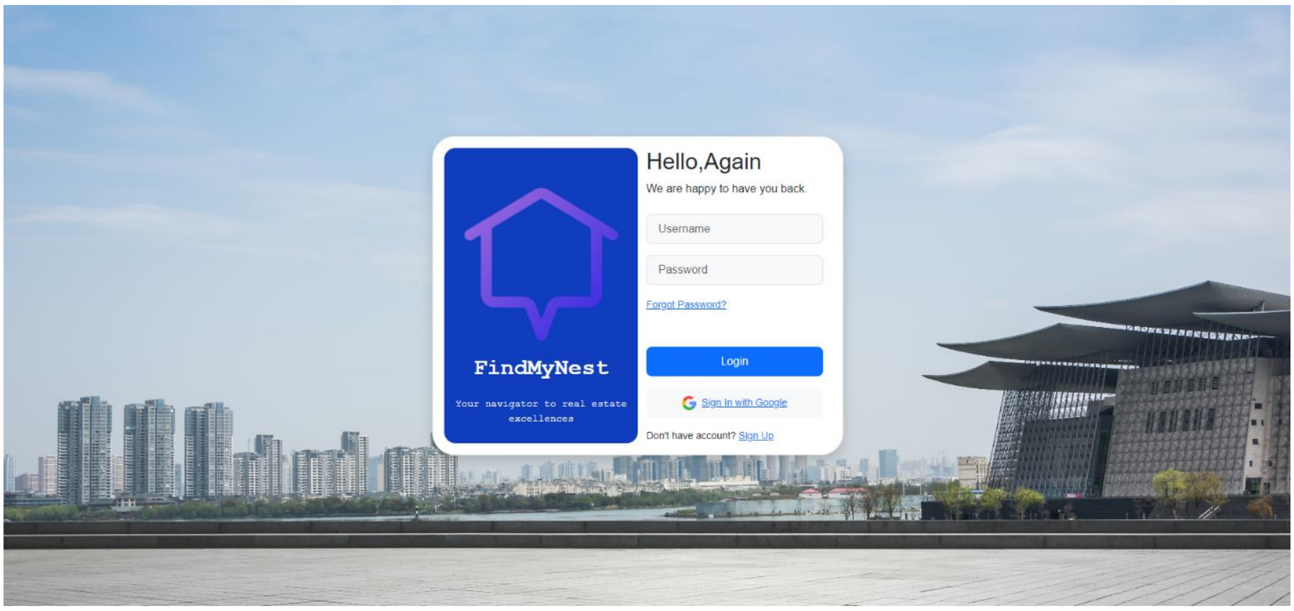
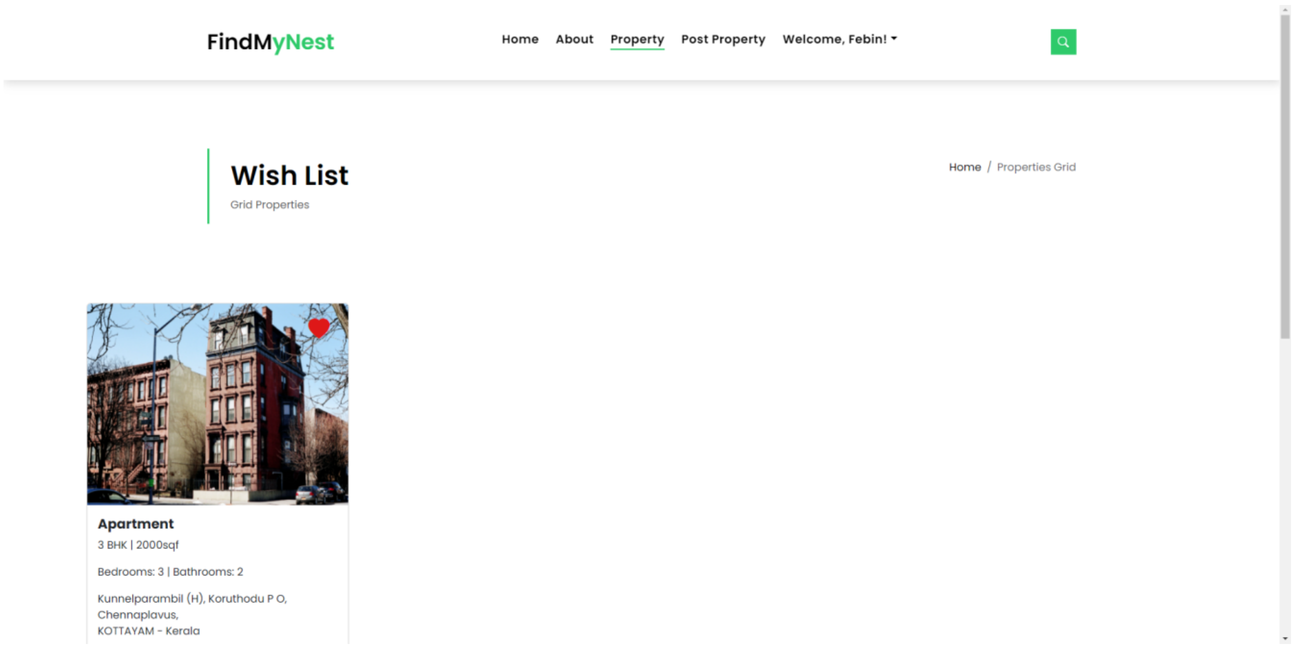

FindMyNest
Your navigator to real estate excellences
Hello,Again
We are happy to have you back.
Username
Password
Forgot Password?
Login
Sign In with Google
Don't have account? Sign Up

Figure 23: Login page

Wishlist Page



Wish List

Grid Properties

Apartment

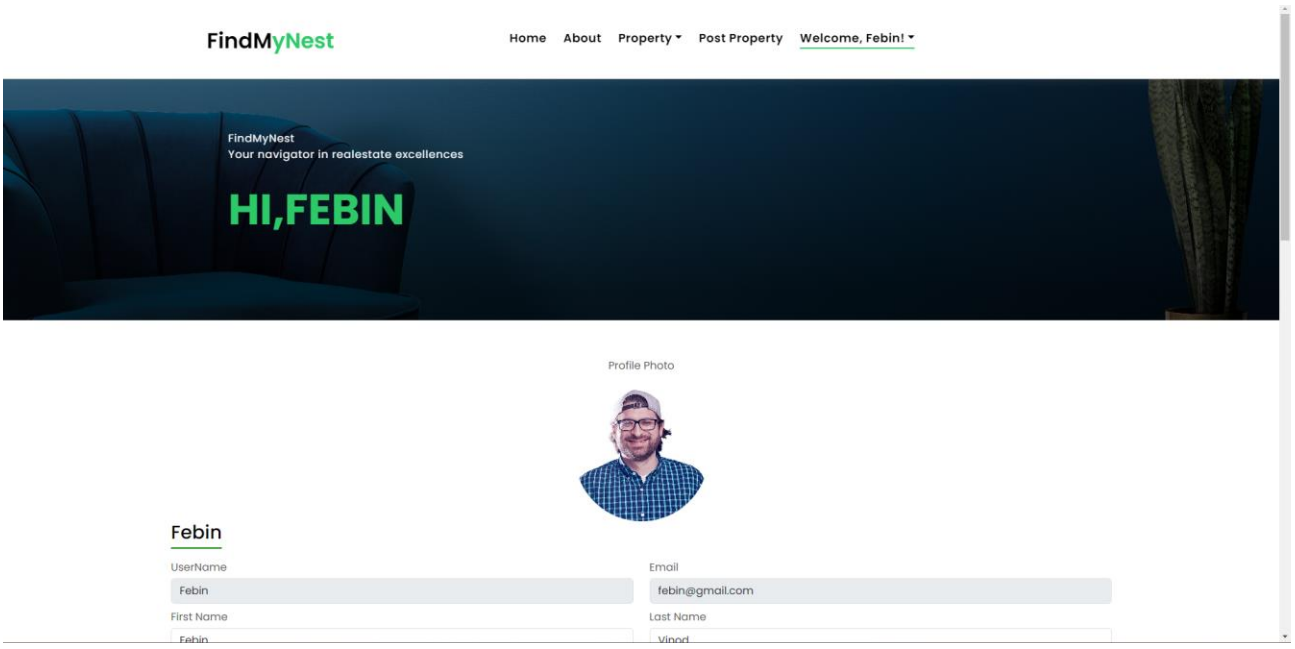
3 BHK | 2000sqf

Bedrooms: 3 | Bathrooms: 2

Kunnelparambil (H), Koruthodu P O, Chennaplavus, KOTTAYAM - Kerala

Figure 24: Wishlist Page

Profile

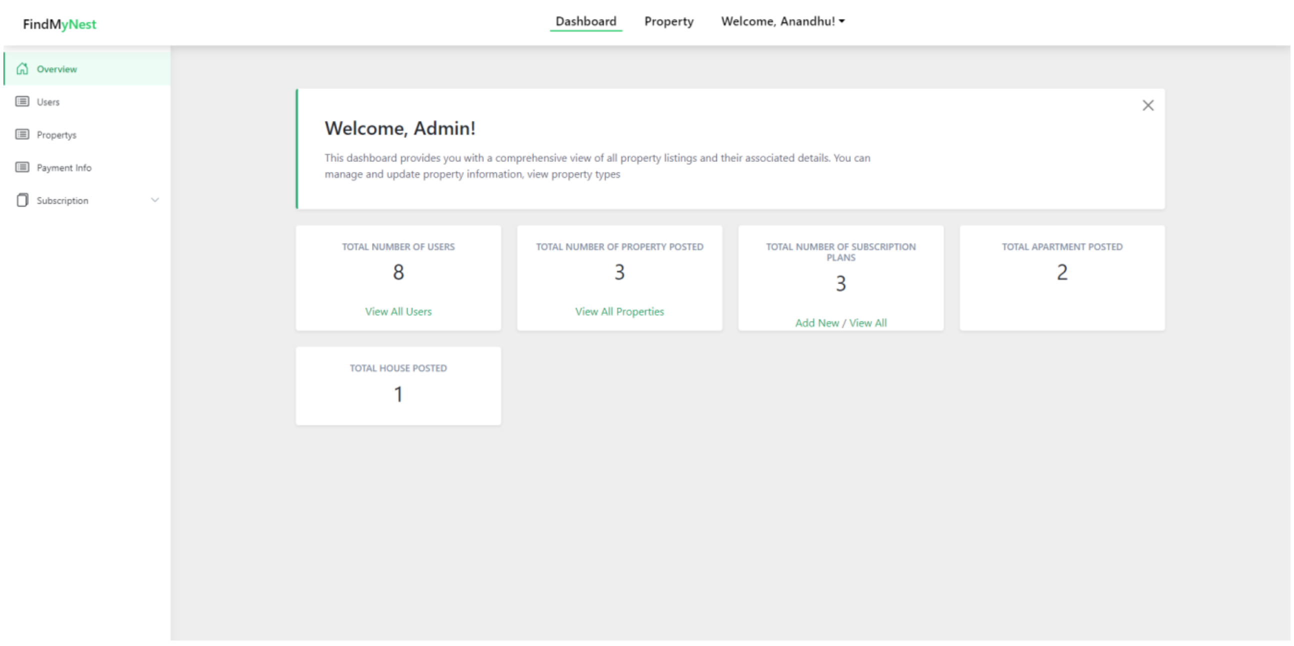


Febin

UserName	Email
Febin	febin@gmail.com
First Name	Last Name
Febin	Vinod

Figure 25: Profile Page

Admin Page


FindMyNest

Dashboard Property Welcome, Anandhu!

Overview
Users
Propertys
Payment Info
Subscription

Welcome, Admin!

This dashboard provides you with a comprehensive view of all property listings and their associated details. You can manage and update property information, view property types

TOTAL NUMBER OF USERS 8 View All Users

TOTAL NUMBER OF PROPERTY POSTED 3 View All Properties

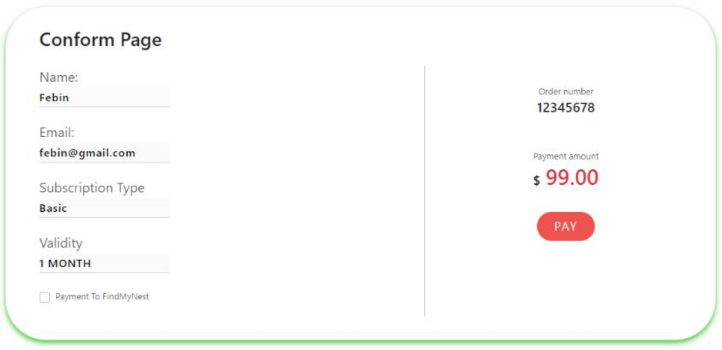
TOTAL NUMBER OF SUBSCRIPTION PLANS 3 Add New / View All

TOTAL APARTMENT POSTED 2

TOTAL HOUSE POSTED 1

Figure 26: Admin page

Payment Page


Conform Page

Name: Febin

Email: febin@gmail.com

Subscription Type Basic

Validity 1 MONTH

Payment To FindMyNest

Order number 12345678

Payment amount $ 99.00

PAY

Figure 27: Payment page