Daffo House Rent

Submitted By

Student Name	Student ID
Raiyan Jahan	0242310005101599

House Rental System Using CSS HTML and Python Flask Web Interface

This Report Presented in Partial Fulfillment of the course **CSE315: Software Engineering in the Computer Science and Engineering Department**

DAFFODIL INTERNATIONAL UNIVERSITY

Dhaka, Bangladesh

December 14, 2025

We hereby declare that this lab project has been done by us under the supervision of **Pranto Protim Choudhury**, **Lecturer**, Department of Computer Science and Engineering, Daffodil International University. We also declare that neither this project nor any part of this project has been submitted elsewhere as lab projects.

Submitted To:

Pranto Protim Choudhury
Lecturer
Department of Computer Science and Engineering
Daffodil International University

Submitted by

Raiyan Jahan
ID:0242310005101599
Dept. of CSE, DIU

Table of Contents

Project Report: Daffo House Rent

Project Title: Daffo House Rent

University: Daffodil International University (DIU)

Target Area: Daffodil Smart City (Ashulia), Dhanmondi, Sobhanbag, and greater Dhaka.

Team Members:

0. User Stories :

User stories describe the requirements from the perspective of the end-user.

1. **Student (General):** "As a DIU student, I want to search for housing specifically near the Smart City campus so that I can minimize my daily commute."
2. **Student (Budget Conscious):** "As a student, I want to filter properties by 'Bachelor Mess' or 'Seat' and set a maximum price limit so I can find accommodation within my monthly budget."
3. **Landlord:** "As a local homeowner in Ashulia, I want to list my empty flat with photos and rent details so that university students can easily find and contact me."

1. Process Model Chosen:

Agile Methodology (Iterative Model)

We chose the Agile Iterative Model for the development of Daffo House Rent.

- **Reasoning:** Student housing needs change rapidly, and user feedback is critical. Agile allows us to develop the core features (Search, Login) first and then iterate to add advanced features (Filters, Profile Management) based on testing by the team (Samira and Adiba).

- **Implementation:** The project was broken down into sprints:
    - *Sprint 1:* Basic UI and Database Setup.
    - *Sprint 2:* User Authentication and Property Listing.
    - *Sprint 3:* Advanced Search Filters and Profile Dashboard.

2.Functional Requirements:

Identifier	**Req 1**
Title	User Registration
Requirement	This feature allows new users to register and create an account on the platform.
Business Rule (if required)	Users must provide a valid email address for account registration.

Identifier	**Req 2**
Title	Search Functionality
Requirement	Enables users to search for information using specified criteria.
Business Rule (if required)	Search results must be based on the user's specified criteria.

Identifier	**Req 3**
Title	Map Integration
Requirement	Incorporates maps to show property locations or other relevant geographical data.
Business Rule (if required)	Maps must accurately represent property locations.

Identifier	**Req 4**
Title	Accommodation Details

Requirement	Provides detailed information about accommodations like hotels or rental homes.
Business Rule (if required)	Property details must include descriptions, amenities, pricing, and availability.

Identifier	**Req 5**
Title	Two-Way Communication
Requirement	Facilitates and secures communication between users.
Business Rule (if required)	Communication must be secure and real-time.

Identifier	**Req 6**
Title	Notification
Requirement	Sends alerts or notifications to users related to their preferences or important updates.
Business Rule (if required)	Notifications must be timely and relevant.

Identifier	**Req 7**
Title	User Profile
Requirement	Allows users to create and customize their profile on the platform.
Business Rule (if required)	User profiles must be customizable and accessible from any device.

Identifier	**Req 8**
Title	Real-Time Availability
Requirement	Ensures the information about the availability of services or products is current.
Business Rule (if required)	Availability updates must be accurate and immediate.

Identifier	**Req 9**
Title	Filter Customization
Requirement	Enables users to apply custom filters based on their preferences for filtering search results.
Business Rule (if required)	Customized filters must reflect user preferences accurately.

3.Non-Functional Requirements:

Performance

The web application should be responsive and provide fast loading times to ensure a smooth user experience. It should be able to handle many concurrent users and process requests efficiently

Scalability

The application should be designed to accommodate increasing user demands and be scalable to support many banquet events, attendees, and venues. It should handle essential data and increase user traffic without performance degradation

Security

The application should prioritize the security and confidentiality of user data, payment information, and communication logs. It should implement robust security measures, including encryption, secure authentication, and access controls, to protect against unauthorized access, data breaches, and potential vulnerabilities

Reliability

The application should be highly reliable, with minimal downtime and disruptions. It should have backup and recovery mechanisms to ensure data integrity and availability in case of system failures or unforeseen incidents. The application should be compatible with different web browsers,

User-Friendly Interface

The application should have a user-friendly interface with intuitive navigation and clear instructions. It should be easy for event organizers and attendees, with a consistent and visually appealing design that enhances the overall user experience.

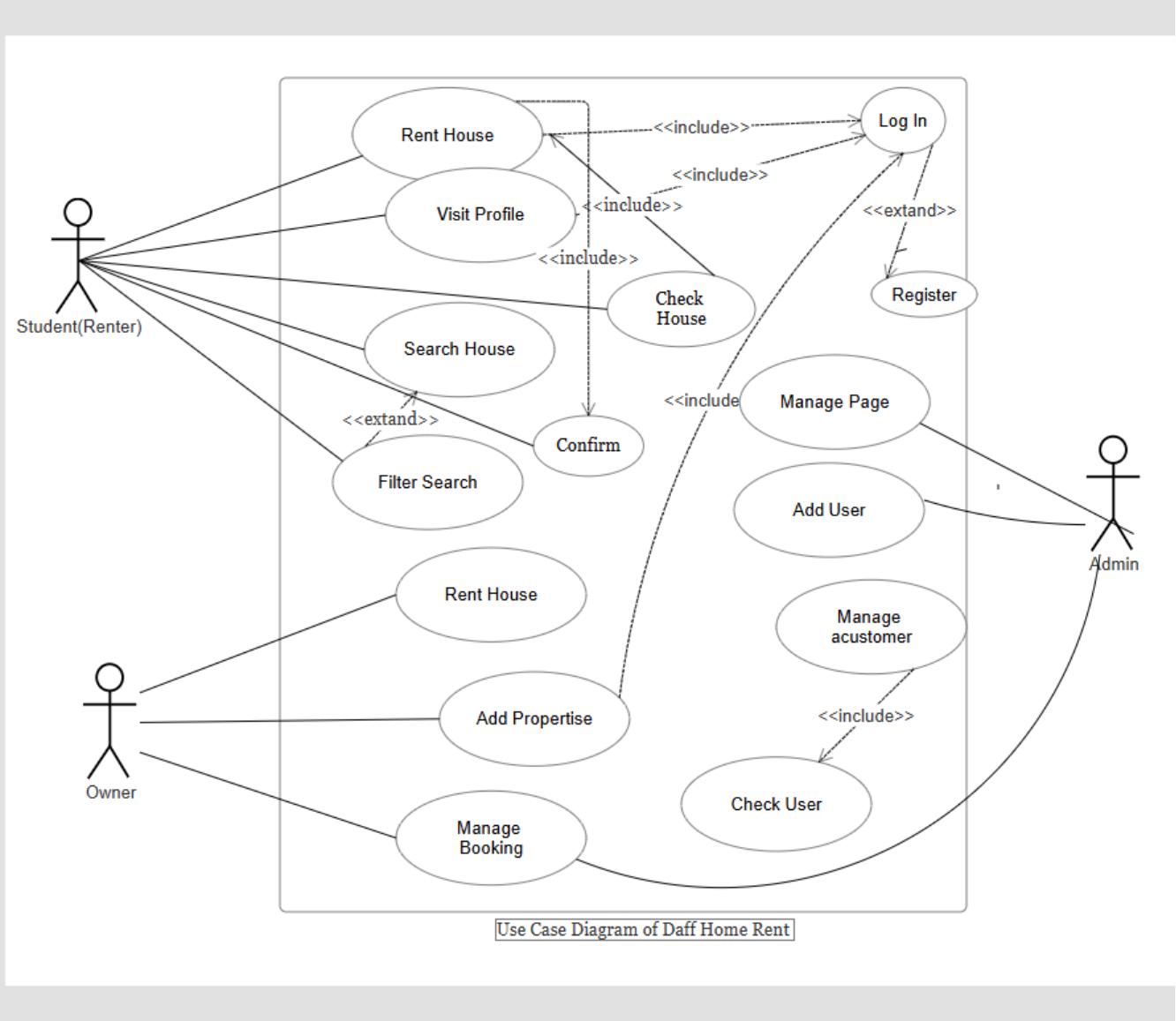
Accessibility

The application should be accessible to users with disabilities, complying with relevant accessibility standards and guidelines. It should support assistive technologies and provide options for font size adjustments, color contrasts, and alternative text for images. backup and recovery mechanisms to ensure data integrity and availability in case of system failures or unforeseen incidents.

Compatibility

The application should be compatible with different web browsers, operating systems, and devices, ensuring that users can access and use the application seamlessly regardless of their preferred platform. It should be easy for event organizers and attendees, with a consistent and visually appealing design that enhances the overall user experience.

4.Use Case Diagram:



Use Case Diagram of Daff Home Rent

5.Use Cases Descriptions

Buy/Rent House

Use Case Name:	Rent House
ID:	UC-1
Actors Involved:	Student

Brief Description	The “Rent House” use case allows customers to purchase or rent a residence from the available listings.
Trigger:	A Student indicates that they want to rent a house, hostel, or flat.
Pre-Conditions	The Student is logged into their account and has sufficient funds for the purchase.
Post-Conditions	The system marks the house and records the transaction.
Normal Flow of Events:	1. Student searches for available houses using specified criteria and views detailed information and images of the desired house. 2. Student selects a house to buy or rent. 3. The system verifies the customer’s credentials and funds. 4. The system confirms the transaction and updates the house status to “Owned” or “Rented.” 5. A confirmation message is sent to the customer.
Business Rules	BR-1: Only users with valid login credentials can search and view house details. BR-2: The house status must be updated in real-time to avoid double bookings or sales. BR-3: All transactions must be logged with a timestamp and customer details for auditing purposes. BR-4: Specific roles (e.g., Admin) can override certain business rules under special circumstances.

Register

Use Case Name:	Register
ID:	UC-2
Actors Involved:	Student, Owner
Brief Description	The “Register” use case allows new customers and owners to create accounts in the system.
Trigger:	A new user (customer or owner) wants to create an account.

Pre-Conditions	None.
Post-Conditions	A new user account is created, and the user can log in to the system.
Normal Flow of Events:	1. User accesses the registration page. 2. User fills out the registration form with necessary details 3. User submits the registration form. 4. The system validates the input details. 5. The system creates a new user account. 6. A confirmation email is sent to the user.
Business Rules	BR-1: The email address provided must be unique and not already associated with an existing account. BR-2: Passwords must meet the minimum-security requirements.

Check House

Use Case Name:	Check House
ID:	UC-3
Actors Involved:	Student
Brief Description	The "Check House" use case shows house details for rent
Trigger:	A Student wants to view details of a house.
Pre-Conditions	The Student logged into their account.
Post-Conditions	The Student views the details of the selected house.
Normal Flow of Events:	1. Students search for houses using specified criteria. 2. The student selects a house from the search results. 3. The system displays detailed information and images of the selected house.

Confirm

Use Case Name:	Confirm
ID:	UC-5
Actors Involved:	Student
Brief Description	The “Confirm” use case allows Students to confirm their booking or purchase.

Trigger:	A Student wants to confirm a booking or purchase.
Pre-Conditions	The student is logged into their account and has made a booking or purchase request.
Post-Conditions	The booking or purchase is confirmed, and the system updates the status accordingly.

Rent House

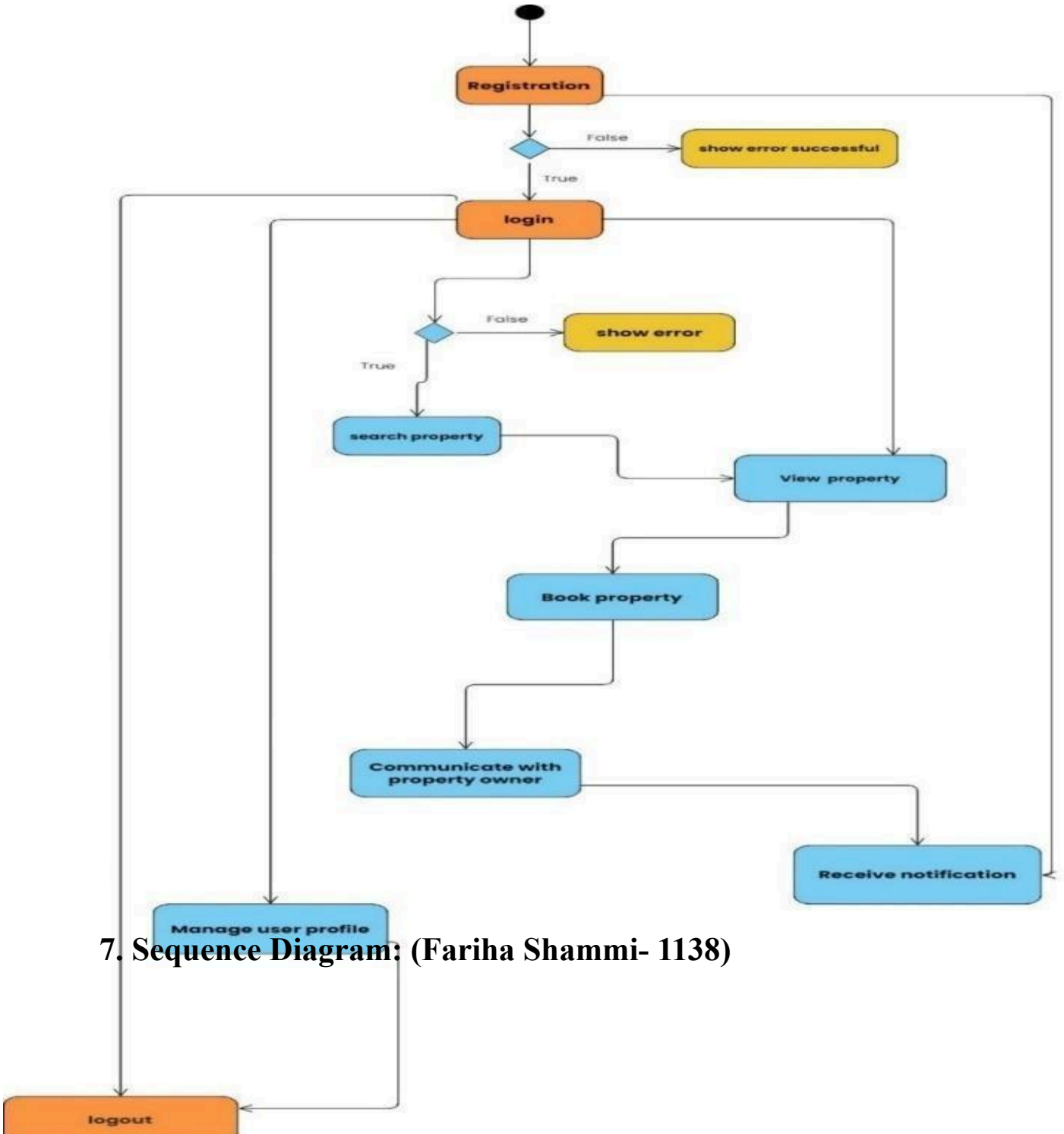
Use Case Name:	Rent House
ID:	UC-6
Actors Involved:	Owner
Brief Description	The “Rent House” use case allows owners to list their houses for rent.
Trigger:	An owner wants to list a house for rent.
Pre-Conditions	The owner is logged into their account.
Post-Conditions	The house is listed as available for rent in the system.
Normal Flow of Events:	1. Owner navigates to the “Rent House” page. 2. Owner enters details about the house (e.g., location, price, description, photos). 3. Owner submits the listing. 4. The system verifies the details and adds the house to the available listings. 5. A confirmation message is sent to the owner.
Business Rules	BR-1: Only logged-in owners can list houses for rent. BR-2: Listings must include all required details and meet the platform’s quality standards. BR-3: Owners must agree to the terms and conditions of the rental listing.

Manage House

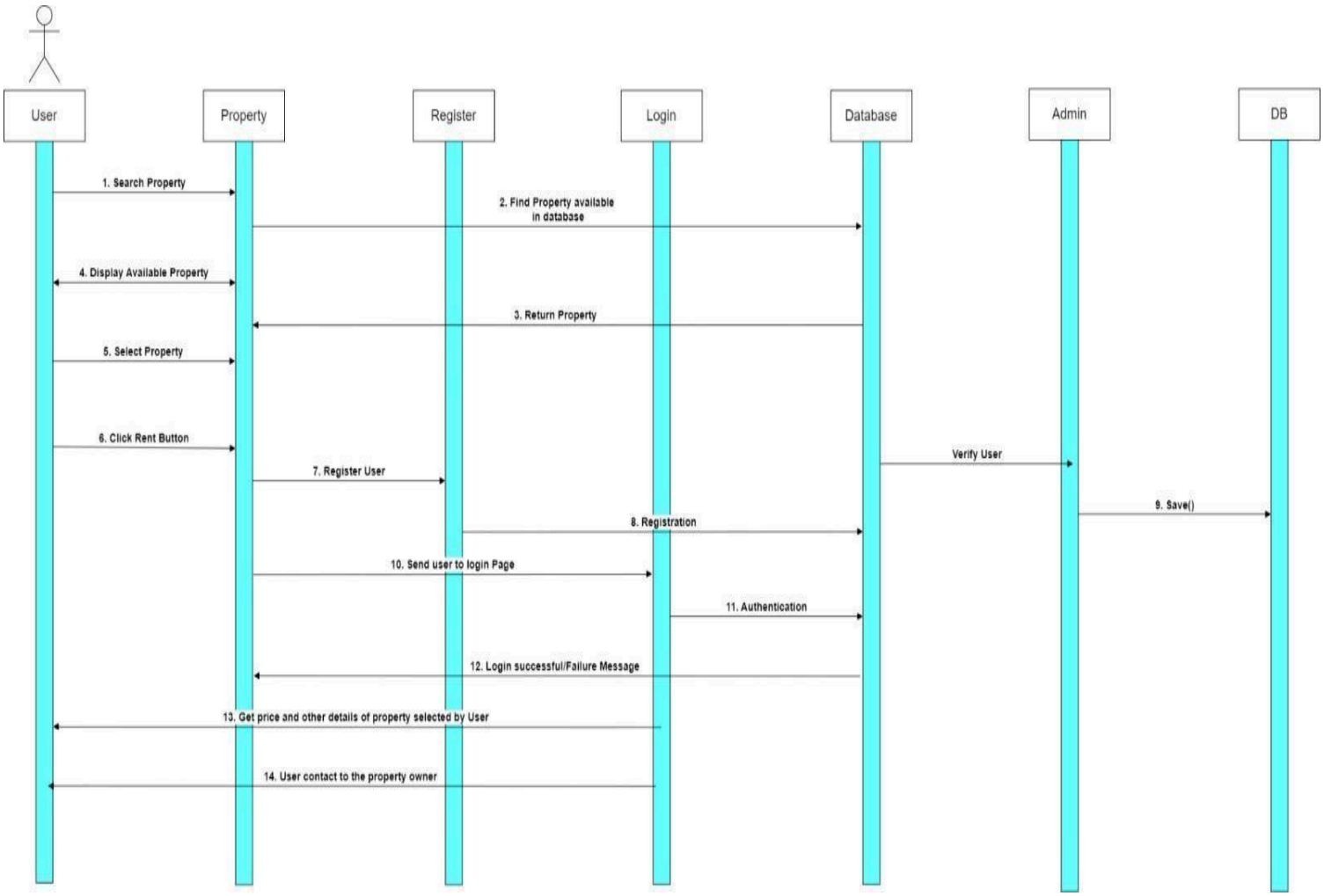
Use Case Name:	Manage House

ID:	UC-7
Actors Involved:	Owner
Brief Description	The "Manage House" use case allows owners to update details of their listed houses.
Trigger:	An owner wants to update the details of a listed house.
Pre-Conditions	The owner is logged into their account and has a house listed.
Post-Conditions	The house details are updated in the system.
Normal Flow of Events:	1. Owner navigates to the "Manage House" page. 2. Owner selects the house they want to update. 3. Owner updates the details of the house. 4. Owner submits the changes. 5. The system verifies the updated details and updates the listing. 6. A confirmation message is sent to the owner.
Business Rules	BR-1: Only logged-in owners can manage their house listings. BR-2: Updated details must meet the platform's quality standards and be verified. BR-3: Owners must confirm changes before they are applied.

6.Activity Diagram:



7. Sequence Diagram: (Fariha Shammi- 1138)


User
Property
Register
Login
Database
Admin
DB
1. Search Property
2. Find Property available in database
4. Display Available Property
3. Return Property
5. Select Property
6. Click Rent Button
Verify User
7. Register User
9. Save()
8. Registration
10. Send user to login Page
11. Authentication
12. Login successful/Failure Message
13. Get price and other details of property selected by User
14. User contact to the property owner

8.UML

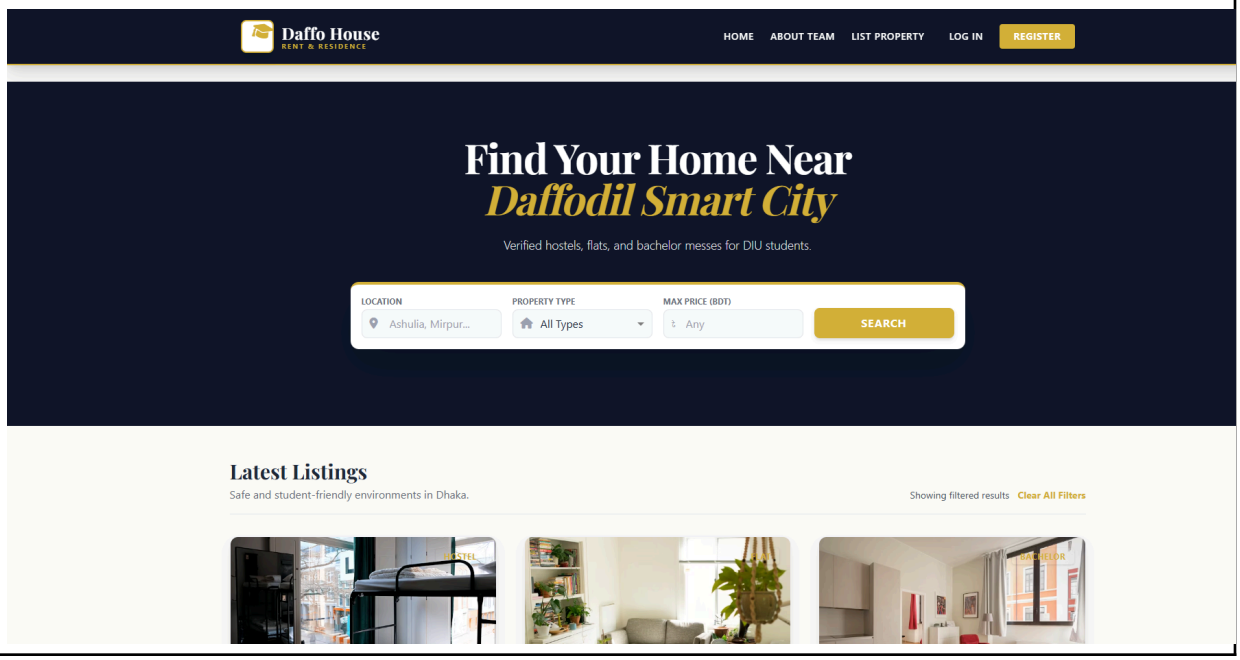
Screen: <1 of 7> **Link from screen:** Home Page **Link to screen:** Home Page	**Screen Description:** The Screen home page show the main page of project where user can search the properties and see the recent listed properties.

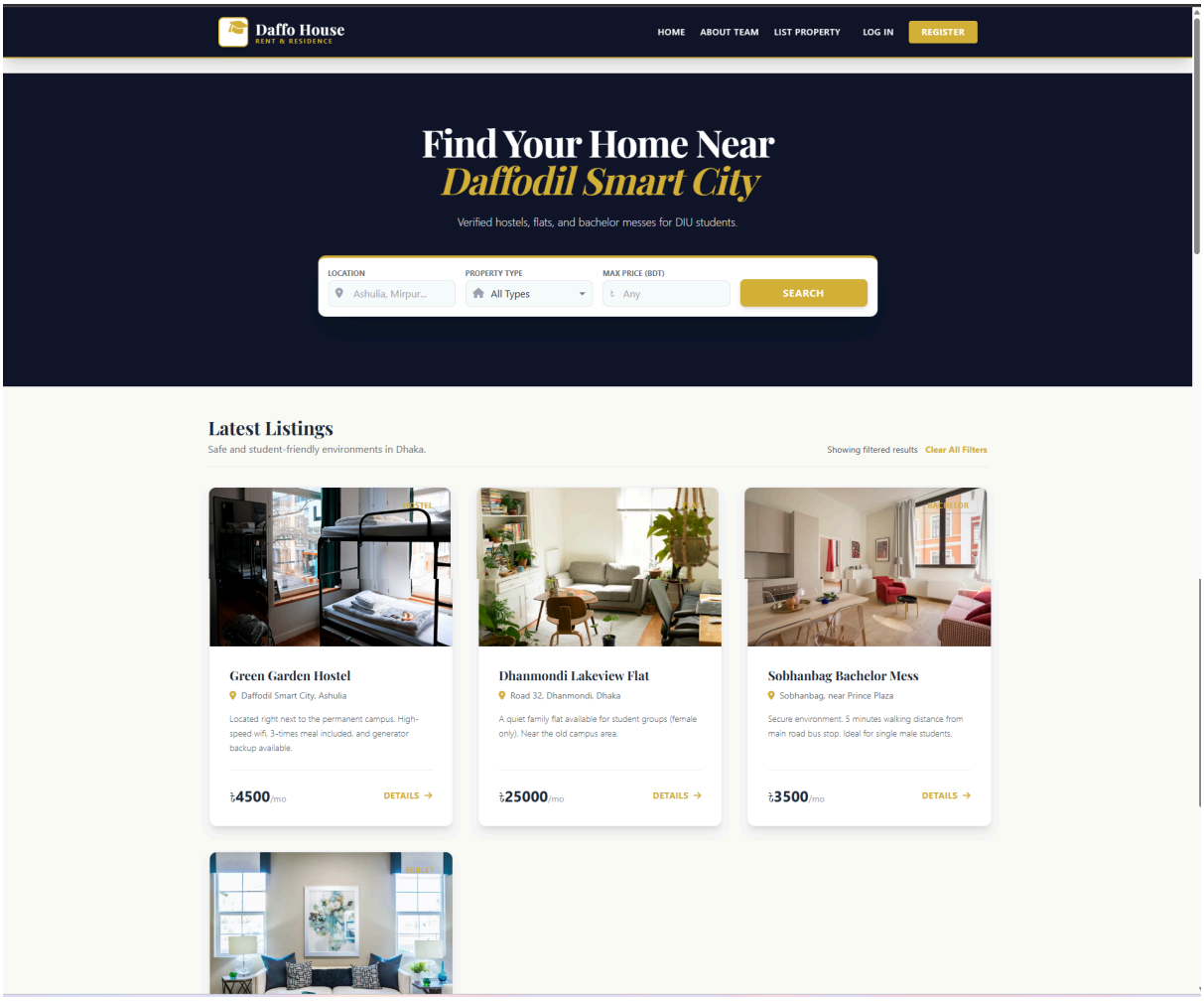
Functionality/Interactivity:

The screen will display the complete home page to the user.

- Where user have to login or register themselves.
- Students can see all the property listing.

Screen Design:





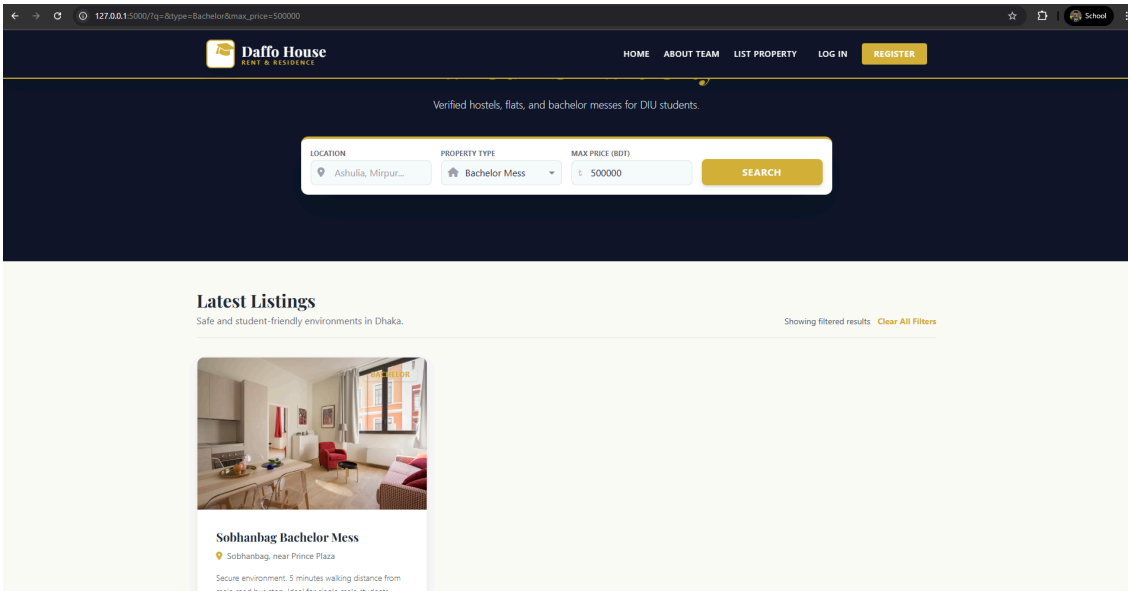
Background:	**Audio: None**
Colour scheme: grey, white.	**Video: None**

Screen: <2 of 7> **Link from screen:** Home Page **Link to screen:** Search Property	**Screen Description:** The Screen Get Started (Search Properties) show us the listed properties where using filters user can see the properties.

Functionality/Interactivity:

The screen shows the property listed and user can also filter the property according to Amenities, Price high to low.

Screen Design:

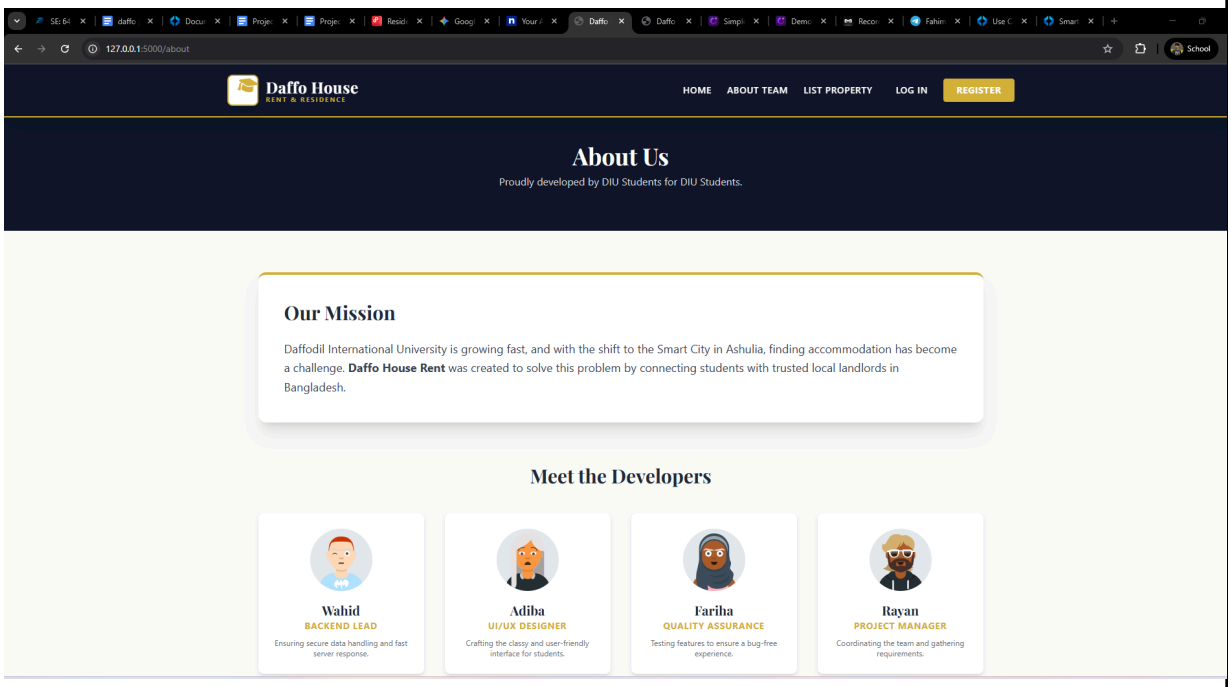


Background:	**Audio: None**

Colour scheme: grey, silver, white.	**Video: None**
Text attributes: Inter, Avenir, Helvetica, Arial, sans-serif.	**Still images: No**

Screen: <3 of 7> **Link from screen:** Home Page **Link to screen:** About Page	**Screen Description:** The Screen About Us show about the project, vision of project, functionality of project and what project offer.
Functionality/Interactivity: This screen displays the details about our website.	

Screen Design:



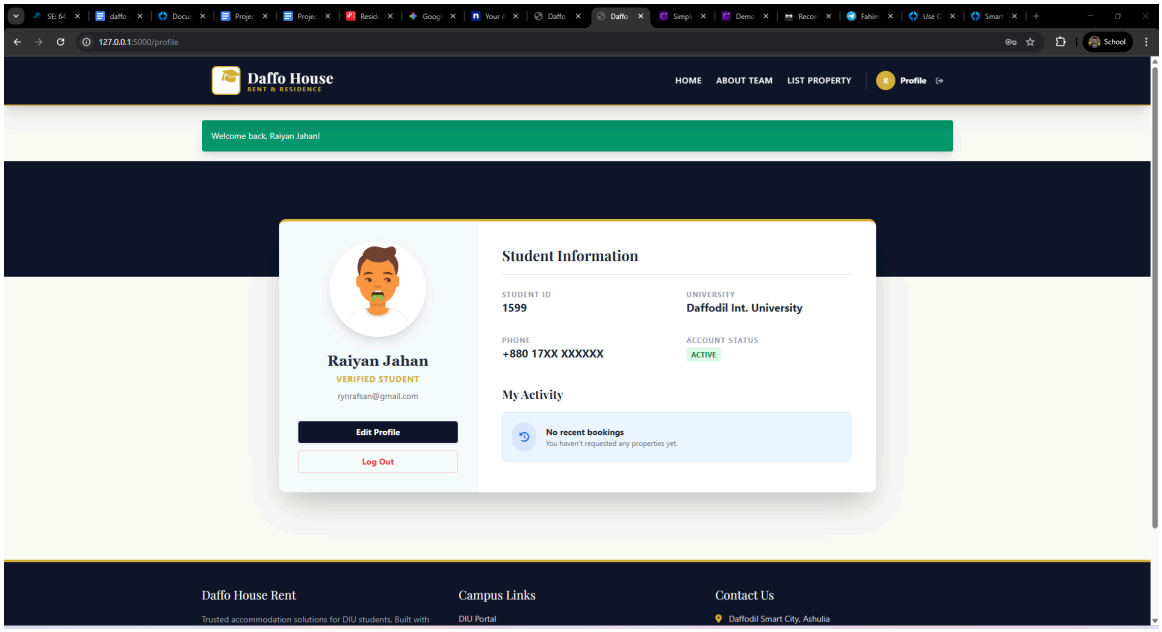
Background:	**Audio: None**
Colour scheme: grey, silver, black.	**Video: None**
Text attributes: Inter, Arial, sans-serif.	**Still images: No**

Screen: <4 of 7> **Link from screen:** Home, About Page **Link to screen:** Profile Page	**Screen Description:** The Screen Profile show the profile of user either user can update the properties, create listing, delete account, sign-out and show listed properties.

Functionality/Interactivity:

In this screen user can update their profile and they can also create listing to their property if they want but this screen is only displayed when user will be login to their account.

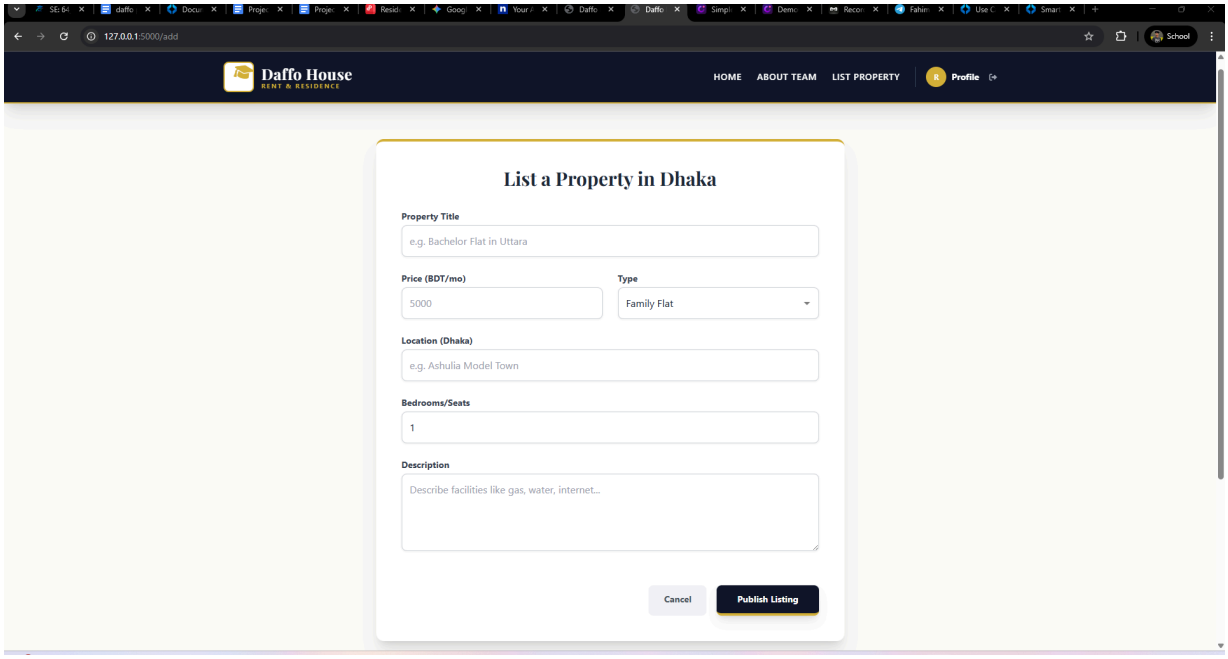
Screen Design:



Background:	**Audio: None**
Colour scheme: grey, white, blue, green, red.	**Video: None**
Text attributes: Inter, Avenir, Helvetica, Arial, sans-serif.	**Still images: No**

Screen: <5 of 7> **Link from screen:** Profile Page **Link to screen:** Create Listing Property	**Screen Description:** The Screen Create listing properties show the form where user can add new property on application.
Functionality/Interactivity: This screen displayed a form for create listing, they can fill the form and upload their flat / hostel images. After create listing, it will display on the home page and display the details according to they fill the form.	

Screen Design:



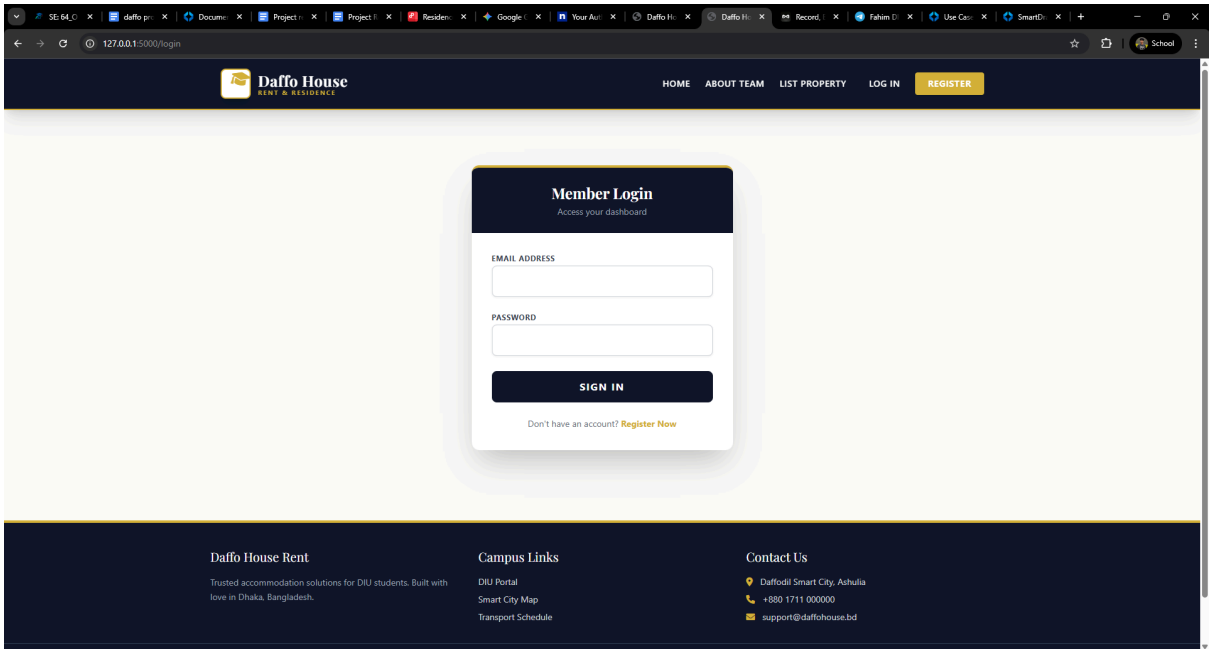
Background:	**Audio: None**
Colour scheme: grey, silver, red, green, black, white.	**Video: None**

Screen: <6 of 7> **Link from screen:** Home, About Page **Link to screen:** Sign In	**Screen Description:** The Screen Sign-In show the options to sign in for registered (signup) user either user can sign in through google.

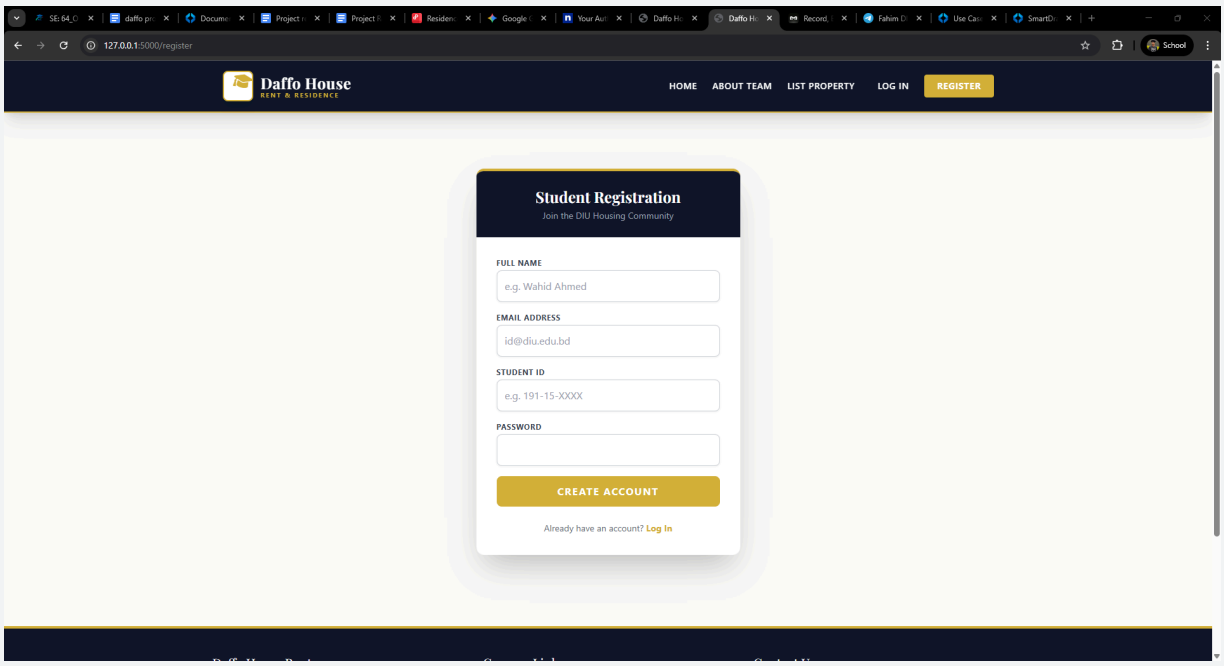
Functionality/Interactivity:

Use inputs data to create listing. If he/she is a new user, user chooses the sign-up button or they can also connect with their google account by click on the button.

Screen Design:

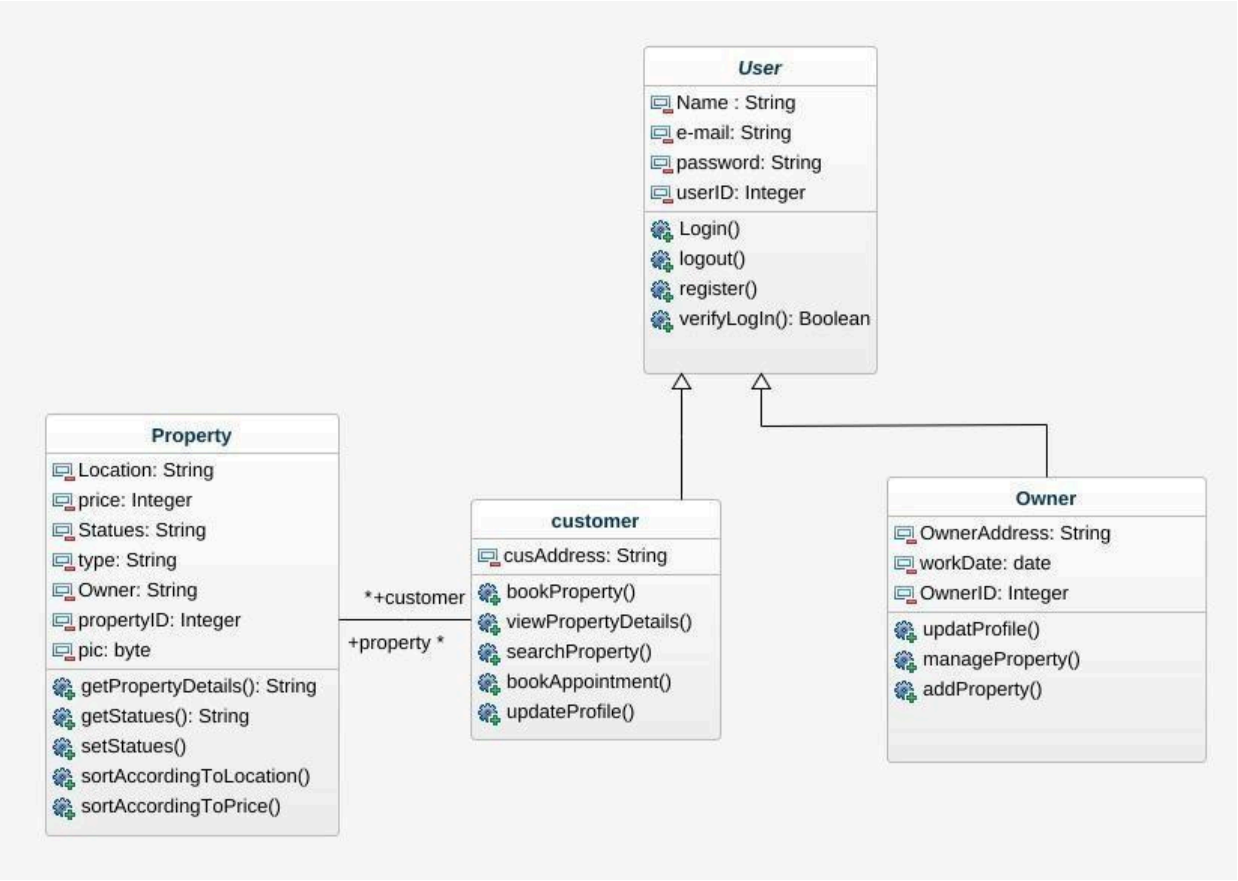


Screen: <7 of 7> **Link from screen:** Home, Page, Sign in Page **Link to screen:** Sign Up	**Screen Description:** The Screen Sign-Up show the options to registered (signup) for new user either user can sign Up through google.
Functionality/Interactivity: User Register himself by input the required credentials mention in screen, or they can sign up using their google account by clicking the button.	
Screen Design:	

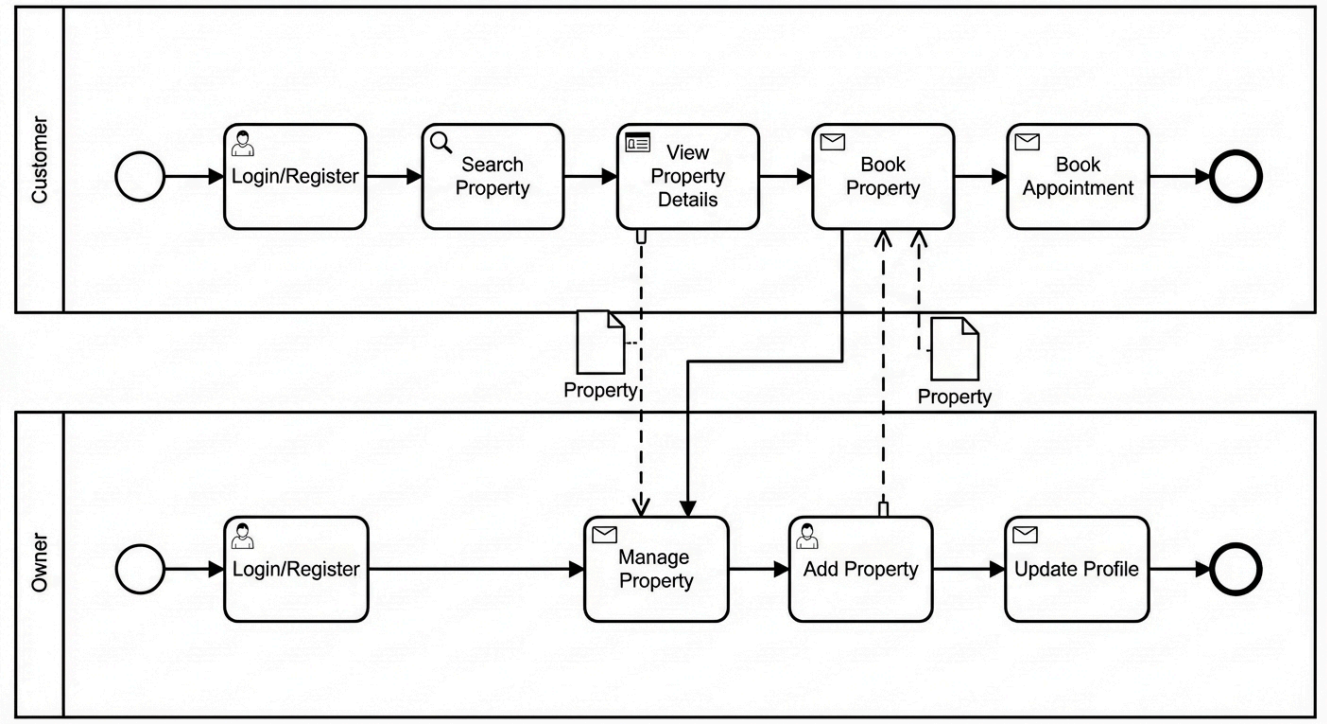


Background:	**Audio: None**
Colour scheme: grey, silver, red, blue.	**Video: None**
Text attributes: Inter, sans-serif.	**Still images: No**

9.Class Diagram:



10.Business Process Model Notation:



11.Software Maintenance:

Software maintenance for this property management system is an ongoing process carried out after deployment to ensure the system continues to function correctly, securely, and efficiently. Based on the completed unit testing and functional testing, all major modules such as login, profile management, property handling, image uploading, and authentication have been verified. However, regular maintenance is required to sustain system quality and user satisfaction.

Login and Authentication Maintenance:

The login module, including manual login and “Continue with Google” authentication, will be continuously monitored to ensure secure access. Maintenance activities include handling failed login attempts, updating authentication libraries, resolving issues related to invalid credentials, and ensuring compatibility with Google authentication updates. Security patches will be applied to protect user data and prevent unauthorized access.

Profile Management Maintenance:

The edit profile functionality will be maintained to ensure accurate updating of user information such as username, email, and password. Validation rules will be reviewed periodically to prevent empty or invalid inputs. Any issues related to profile update failures or confirmation messages will be corrected to maintain data integrity and user trust.

Property Management Maintenance:

Maintenance of the add, update, view, and delete property modules is critical to the system’s core functionality. The property database will be monitored to ensure accurate storage and retrieval of property details such as name, price, amenities, location, and description. Bugs related to property updates, deletion, or display on the home page will be resolved promptly. Performance optimization will be applied as the number of properties increases.

Image Upload Maintenance:
The image upload module will be maintained to ensure only valid image formats (JPEG, PNG) and file sizes (less than 10 MB) are accepted. Error handling mechanisms will be updated to clearly notify users of invalid uploads. Storage management and optimization will be performed to prevent server overload and ensure smooth image loading.
Logout and Session Management Maintenance:
The logout functionality will be maintained to ensure proper session termination and secure redirection to the home page. Session handling will be reviewed to prevent unauthorized access after logout and to maintain correct navigation behavior.
Database and System Performance Maintenance:
Regular database backups, performance monitoring, and optimization will be conducted to ensure data reliability and system stability. Preventive maintenance activities such as code refactoring, validation checks, and security updates will help reduce future system failures.

12.Software Testing:

https://docs.google.com/document/d/1vOxPFAn3083y-3ZxzFUaONEgUpeQ-mAAbCshA8-lSMQ/edit?usp=sharing

13.Cocomo:The Constructive Cost Model (COCOMO) is used to estimate the development effort, time, and manpower required for this Property Management System. The estimation is based on the actual modules implemented in the system, such as user login, Google authentication, profile management, property management (add, update, delete, view), image upload, and database operations.

Project Classification

Based on the characteristics of the developed system, the project is classified as an Organic type project because:

- The system is a web-based property management application
- Requirements are clearly defined and stable
- The project has moderate complexity
- Development was carried out by a small, experienced team
- The system does not involve complex real-time or hardware constraints

Size Estimation (KLOC)

The estimated size of the software is calculated from the implemented modules:

Module	Estimated Lines of Code
Login & Logout	1,200
Google Authentication	800
Edit Profile	900
Add / Update / Delete Property	2,800
Property Details Page	1,100
Image Upload & Validation	1,000
Database & Backend Logic	1,200
Total LOC	10,000 LOC ≈ 10 KLOC

Basic COCOMO Formula (Organic Mode)

- Effort (E)
  $$E = 2.4 \times (KLOC)^{1.05}$$
- Development Time (D)
  $$D = 2.5 \times (E)^{0.38}$$
- Average Team Size (P)
  $$P = \frac{E}{D}$$

COCOMO Calculation for This System

Using KLOC = 10:

- Effort (E)
  $$E = 2.4 \times (10)^{1.05} = 26.9 \text{ Person-Months}$$
- Development Time (D)
  $$D = 2.5 \times (26.9)^{0.38} \approx 8.7 \text{ Months}$$
- Average Team Size (P)

$$P = \frac{26.9}{8.7} \approx 3 \text{ Developers}$$

Cost Interpretation

The estimated results indicate that the development of this Property Management System required approximately 27 person-months of effort, with a development duration of about 9 months and an average team size of 3 developers. These values are realistic considering the number of implemented features such as authentication, property CRUD operations, image validation, and database integration.

14.Sofware Quality Assurance:

Software Quality Assurance (SQA) is a systematic process that ensures the software meets specified requirements and maintains high standards of quality throughout its development and deployment. For this Property Management System, SQA activities were implemented to guarantee the reliability, functionality, and usability of core modules, including login, Google authentication, profile management, property management, image upload, and session handling.

Objectives
The main objectives of SQA for this project were:
1. To ensure that all modules function correctly and meet user requirements.
2. To detect and correct defects early in the development lifecycle.
3. To verify compliance with functional specifications, coding standards, and security guidelines.
4. To provide confidence in system reliability, performance, and user satisfaction.

SQA Activities in the Project
1.Requirement Review
All functional requirements, such as login/logout, property CRUD operations, profile editing, and image uploading, were reviewed for completeness and clarity. This ensured that each module could be tested effectively and met user expectations.
2.Unit Testing
Each module was tested independently:
Login Module:Verified valid and invalid credentials, empty inputs, and Google authentication.
Profile Management:Tested updating with valid data and handling empty fields.
Property Management:Tested adding, updating, viewing, and deleting properties.
Image Upload:Verified correct formats (JPEG, PNG) and file size limits (<10 MB).
All unit tests passed successfully, ensuring individual modules function as expected.
3.Functional Testing
After unit testing, functional testing was performed to ensure modules worked together seamlessly. Tests

included:

- Login with correct navigation to the home page.
- Logout with proper session termination and redirection.
- Property details display and integration with the database.

Successful results confirmed that modules interacted correctly and fulfilled system requirements.
4.Performance and Usability Checks
Although primarily a functional project, attention was given to response times, proper navigation, and intuitive user interface design. Error messages and validation alerts were verified for clarity.
5.Corrective and Preventive Measures
Any identified issues during testing were corrected immediately. Preventive actions, such as input validation, secure session handling, and data integrity checks, were implemented to reduce future defects.

Tools and Standards

- Testing was conducted manually using predefined test cases for all modules.
- Standard SQA practices, such as code review, validation, and verification against specifications, were applied throughout development.
- All test cases were documented and results recorded to provide traceability and accountability.

15.Ethical, Security Issues: (Wahid Ahmed- 1650)

Ethical Issues
1.User Privacy and Confidentiality
The system handles sensitive user information such as usernames, passwords, email addresses, profile data, and property details. It is an ethical responsibility of the system to ensure that all personal and financial information is kept confidential and is not disclosed to unauthorized users. Any misuse or leakage of user data would violate ethical standards and user trust.
2.Data Protection and Responsible Data Handling
Users rely on the system to store their personal and property-related data securely. Ethically, the system must ensure that collected data is used only for its intended purpose and is not shared with third parties without user consent. Improper handling of user data may result in ethical misconduct and legal consequences.
3.Transparency in Authentication Methods
The system allows login through traditional credentials and Google authentication. It is ethically important to clearly inform users how their data is being accessed and stored when using third-party login services. Users should be aware of what information is retrieved from Google accounts.
4.Fair Access and Non-Discrimination
The system should provide equal access to all users without discrimination. Login, property addition, profile editing, and image uploads should function consistently for all valid users. Any bias in system behavior may raise ethical concerns.
5.Accuracy of Information Display
Property details such as price, location, and amenities are visible to users. Ethically, the system should ensure that the information displayed is accurate and not misleading. Incorrect or manipulated data can negatively impact user decisions and trust.

Security Issues

1.Authentication and Authorization Risks

Although login testing confirms valid and invalid credential handling, weak password policies or improper session handling may allow unauthorized access. The system must ensure strong authentication mechanisms to prevent brute-force attacks and unauthorized logins.

2.Password Security

Passwords entered during login and profile editing must be encrypted and stored securely using hashing techniques. Storing passwords in plain text poses a serious security threat and can lead to account compromise.

3.Third-Party Login Vulnerabilities

The “Continue with Google” feature introduces dependency on an external authentication service. If not implemented securely, it may expose the system to token misuse, session hijacking, or unauthorized access through compromised Google accounts.

4.Data Integrity and Access Control

Property management features such as add, update, and delete must be protected by proper authorization checks. Unauthorized users should not be able to modify or delete properties belonging to other users. Weak access control may result in data tampering.

5.File Upload Security Risks

The image upload functionality accepts JPEG and PNG formats under 10 MB. However, without proper validation and scanning, malicious files could be uploaded. This may lead to security threats such as malware injection or server compromise.

6.Session Management and Logout Security

Logout testing confirms redirection to the home page, but improper session termination may allow attackers to reuse session tokens. Secure session invalidation is necessary to prevent session hijacking.

7.Database Security

Since properties and user data are stored in a database, weak database security can expose sensitive information. SQL injection, unauthorized access, and data leakage are potential threats if proper security measures are not implemented.