

Software Development for Web Apps (Sessional)
Final Report
Course: CSE-410



BARI-WALA

Team Members

Md. Econozzaman Econ
ID: 1111034

Tamal Chakraborty
ID: 1111038





Md. Arafat Bin M. Rahman
ID: 1111041

Md. Saad Bin Kamal
ID: 1111044

Department of Computer Science and Engineering (CSE)
**Bangladesh Army International University of Science &
Technology (BAIUST)**

Sharatnagar, Syedpur 3501, Bangladesh

Team Members

Photo	Name	ID	Email	Contact NO.	Total Credit Passed	CGPA acquired
	Md. Econozzaman Econ	1111034	econoza- man@baiust.edu.bd	01533798331	138.75	3.95
	Tamal Chakraborty	1111038	tamal- cbcb@gmail.com	01838345075	138.75	3.84
	Md. Arafat Bin M. Rahman	1111041	m.rah- man@baiust.edu.bd	01780839976	138.75	3.27
	Md. Saad Bin Kamal	1111044	saad.ka- mal@baiust.edu.bd	01788694804	138.75	3.46

Executive Summary

In Bangladesh we frequently see news like robbery, scam, security breach and women harassment, but the culprit escapes often remaining no trace of him . So, it can be assumed that there is a communication gap between house owners and the tenants. So, we're representing a project named "BARI-WALA" to solve those issues.

The project is aimed to help the house owners in a great manner. Currently no such system is present in our country that holds all the records of the tenants. The system will also integrate a verification module for the provided information's by the tenant for authenticity. It will feature more facilities such as keeping track of the tenants if they are present in the house or not, rent payment details, registering new tenants and deleting old ones, complain section for the tenants.

Contents

Executive Summary	1
List of Figures	4
List of Tables	4
1 Introduction	5
1.1 Goals and Objectives of the project	5
1.2 Scope of the work	5
1.3 System overview	5
1.4 Terms, Acronyms, and Abbreviations Used	6
2 Project Management Plan	7
2.1 Project Organization	7
2.1.1 Individual Contribution to the project	7
2.2 Process Model Used	7
2.2.1 Rationale for choosing lifecycle model	8
2.3 Risk Analysis	9
2.4 Constraints to project implementation	9
2.5 Hardware and Software Resource (Tools/Language) Require- ments	10
2.5.1 Hardware Requirements:	10
2.5.2 Software Requirements:	10
2.6 Project Timeline and Schedule	10
2.7 Estimated Budget	11

2.8 Social/Cultural/Environmental impact of the project	11
3 Requirement Specifications	12
3.1 Use case diagram with Graphical and Textual Description . . .	12
3.2 Activity Diagram	13
3.3 Static model – class diagram	13
3.4 Dynamic model – sequence diagram	13
4 Architecture	14
4.1 Architectural model/style used	14
4.2 Technology, software, and hardware used	15
5 Design	16
5.1 Component level design following pattern	16
5.2 GUI (Graphical User Interface) design	16
6 Testing and sustainability plan	22
6.1 Requirements/specifications-based system level test cases . . .	22
6.2 Traceability of test cases to use cases	26
6.3 Techniques used for test generation	26
6.4 Assessment of the goodness of your test suite	26
6.5 Sustainability Plan	26
6.5.1 Scalability	26
6.5.2 Flexibility / Customization	26
Acknowledgement	26
References	28

List of Figures

List of Tables

2.1 Individual Contribution to the project	7
6.1 Requirements/specifications-based system level test cases (01)	22
6.2 Requirements/specifications-based system level test cases (02)	23
6.3 Requirements/specifications-based system level test cases (03)	24
6.4 Requirements/specifications-based system level test cases (04)	25
6.5 Traceability of test cases to use cases	26

Introduction

1.1 Goals and Objectives of the project

1. The owner will possess complete information about the tenants within the system.
2. The owner can also verify the tenants information.
3. The owner can apply penalty automatically for late payment.
4. Payment will be more convenient, as the owner won't have to personally collect it from the tenants.
5. The system will allow to track the entry and exit of tenants from the house, if needed.
6. The owner can review tenants' feedback about the service and take appropriate action if necessary.
7. Develop a system for tenants to submit maintenance requests and for house owners to efficiently track and address those requests.
8. Creating a UI (User Interface) and UX (User Experience) that is intuitive and enjoyable for users will be gratifying using HTML, CSS and JS.

1.2 Scope of the work

The potential clients for "Bariwala", particularly it's designed for property management are house owners. House owners and landlords looking for a platform to efficiently manage their rental properties, including tenant information, payments, and maintenance. It will revolutionize property management, streamlining tenant data, improving tenant-owner communication, and boosting overall efficiency. So, this project will help the house owners in a great manner.

1.3 System overview

The Bari-wala Project is a comprehensive web application designed to streamline PMS (property management system) for both owners and tenants. This platform serves as a centralized hub for property-related activities, enhancing communication, efficiency, and transparency in the management of rental properties.

1. Home Page:

The home page serves as the gateway to the Bari-wala platform, offering an intuitive interface for users to navigate through the system.

2. User Authentication:

The platform features a secure user authentication system, allowing both tenants and owners to log in to their respective sections.

3. Tenant Section:

Tenants are provided with a dedicated section where they can access personalized information and perform various tasks:

- a. View and update their personal information.
- b. Access payment history and details.
- c. Provide feedback on services or report issues.

4. Owner Section:

Owners have access to an extended set of functionalities, empowering them with comprehensive control over property management:

- a. Tenants Info: Review and manage detailed information about current and past tenants.
- b. Payments: Monitor and track rent payments, with the ability to view payment history.
- c. Unit Status: Track the entry and exit of tenants from the property, ensuring security and accountability.
- d. Tenants Feedback: Receive and respond to feedback from tenants, fostering improved service delivery.

1.4 Terms, Acronyms, and Abbreviations Used

Bari-wala: The name of the web application.

PMS: Property Management System.

UI: User Interface - The graphical layout of the application, including buttons, screens, and visual elements.

UX: User Experience

HTML: Hypertext Markup Language - The standard markup language for creating web pages and web applications.

CSS: Cascading Style Sheets - A style sheet language used for describing the look and formatting of a document written in HTML.

JS: JavaScript.

Project Management Plan

2.1 Project Organization

2.1.1 Individual Contribution to the project

Individual Contribution to the project are shown in Table-2.1

Member Name	Requirement Specification	Planning	Designing	Model Building	User Interface	Testing	Deployment
Md. Econozzaman Econ	✓	✓	✓	✓	✓		
Tamal Chakraborty	✓	✓	✓		✓		
Md. Arafat Bin M. Rahman	✓	✓	✓		✓		
Md. Saad Bin Kamal	✓	✓	✓		✓		

Table 2.1: Individual Contribution to the project

2.2 Process Model Used

We have adopted the waterfall model as our chosen process model, which is characterized by:

1. The waterfall model is a sequential and linear software development approach that progresses through defined phases.
2. The process initiates with requirements gathering and analysis, establishing clear project objectives and scope.
3. System design follows, meticulously planning the overall architecture and detailed specifications.

4. Implementation is the subsequent phase, where the system is constructed based on the established design.
5. Comprehensive testing is conducted to identify and rectify defects or issues within the completed system.
6. Upon successful testing, the system is deployed, and maintenance activities are initiated to address post-deployment concerns or updates. The waterfall model is characterized by its rigid and non-iterative nature, where each phase is a prerequisite for the next.

2.2.1 Rationale for choosing lifecycle model

Selecting a lifecycle model in software development is a critical decision that significantly influences the project's success. Here are some common rationales for choosing specific lifecycle models:

1. **Project Requirements:** The nature and stability of project requirements play a crucial role in selecting a lifecycle model.
2. **Flexibility and Adaptability:** Some projects may require a high degree of flexibility to accommodate changes in requirements or technology. In such cases, iterative and Agile models are preferred because they allow for regular adjustments and enhancements based on feedback from users or stakeholders.
3. **Project Size and Complexity:** The size and complexity of the project influence the choice of a lifecycle model.
4. **Client Involvement:** The level of client or stakeholder involvement can impact the choice of a lifecycle model.
5. **Budget and Time Constraints:** Budget and time constraints are significant considerations. Predictive models like the Waterfall are suitable with a fixed budget and a well-defined timeline.
6. **Risk Tolerance:** The tolerance for risk and uncertainty influences the choice of a lifecycle model.
7. **Regulatory Compliance:** In industries with strict regulatory requirements, such as healthcare or finance, a lifecycle model that emphasizes documentation, traceability, and rigorous testing, such as the V-Model, may be preferred to ensure compliance with industry standards.
8. **Team Experience and Skillset:** The experience and skillset of the development team are important factors.
9. **Customer Expectations:** Understanding customer expectations and preferences is essential.
10. **Organizational Culture:** The existing organizational culture and processes can influence the choice of a lifecycle model.

2.3 Risk Analysis

Risk analysis in the context of software development involves identifying, assessing, and mitigating potential risks that could impact the success of a project. The process of risk analysis typically involves the following key steps:

1. Risk Identification:

Objective: Identify potential risks that could affect the project.

2. Risk Assessment:

Objective: Evaluate the likelihood and impact of each identified risk.

3. Risk Prioritization:

Objective: Prioritize risks based on their severity and potential impact on the project.

4. Risk Mitigation Planning:

Objective: Develop strategies and plans to mitigate or address identified risks.

5. Risk Monitoring and Control:

Objective: Continuously monitor identified risks and implement control measures as necessary.

6. Contingency Planning:

Objective: Develop contingency plans for high-priority risks to minimize the impact if they occur.

7. Communication and Documentation:

Objective: Maintain open communication about identified risks and mitigation strategies.

8. Review and Update:

Objective: Regularly review and update the risk analysis throughout the project life cycle.

2.4 Constraints to project implementation

Schedule Constraints:

1. Project Deadline
2. Time-sensitive Activities
3. Milestones

Budget Constraints:

1. Fixed Budget
2. Resource Allocation
3. Cost Monitoring

Software & Hardware Constraints

1. Software Dependencies
2. Compatibility
3. Hardware Limitations

2.5 Hardware and Software Resource (Tools/Language) Requirements

2.5.1 Hardware Requirements:

1. Computer:

Recommended: A multi-core processor (e.g., Intel Core i5 or AMD Ryzen) and at least 8GB of RAM.

2. Storage:

Recommended: A solid-state drive (SSD) for faster read/write speeds.

3. Display

2.5.2 Software Requirements:

1. Text Editor or Integrated Development Environment (IDE)
2. Web Browser
3. Version Control
4. Browser Developer Tools
5. Database Design Tool
6. Code Linters and Formatters

2.6 Project Timeline and Schedule

1. Project Management Plan(1 week)
2. Requirement Specifications(2 week)
3. Architecture Determining(1 week)
4. Front-End Development(1-2 weeks)
5. Back-End Development(2-3 weeks)
6. Testing(4-5 week)
7. Deployment(1 week)

2.7 Estimated Budget

Hardware cost = 100000 Tk.

Software cost = 2000 Tk.

Development cost = 10000 Tk.

Testing cost = 10000 Tk.

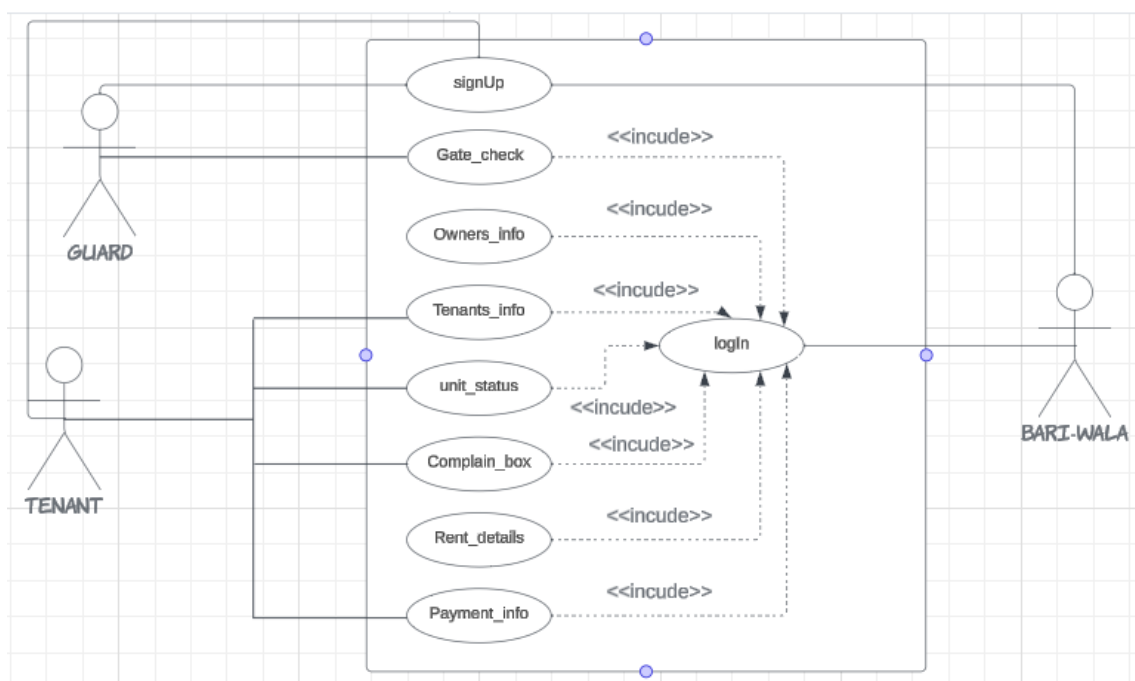
Total Estimated Budget = $(100000+2000+10000+10000)$ Tk. = 122000 Tk.

2.8 Social/Cultural/Environmental impact of the project

- 1. Improved Living Conditions:** If the property management system helps streamline operations, it could contribute to better maintenance and living conditions for tenants.
- 2. Enhanced Tenant-Owner Relations:** Improved communication and transparency between tenants and owners may foster better relationships and a sense of community.
- 3. Access to Information:** The system may empower tenants with access to relevant information about their lease, property rules, and maintenance schedules.
- 4. Efficient Issue Resolution:** A system that facilitates prompt response to maintenance requests can positively impact the quality of life for tenants.
- 5. Increased Financial Transparency:** Automatic payment systems and transparent penalty structures may lead to better financial planning for both tenants and property owners.
- 6. Reduced Paper Usage:** If the system promotes digital communication and documentation, it could lead to a reduction in paper usage, contributing to environmental sustainability.

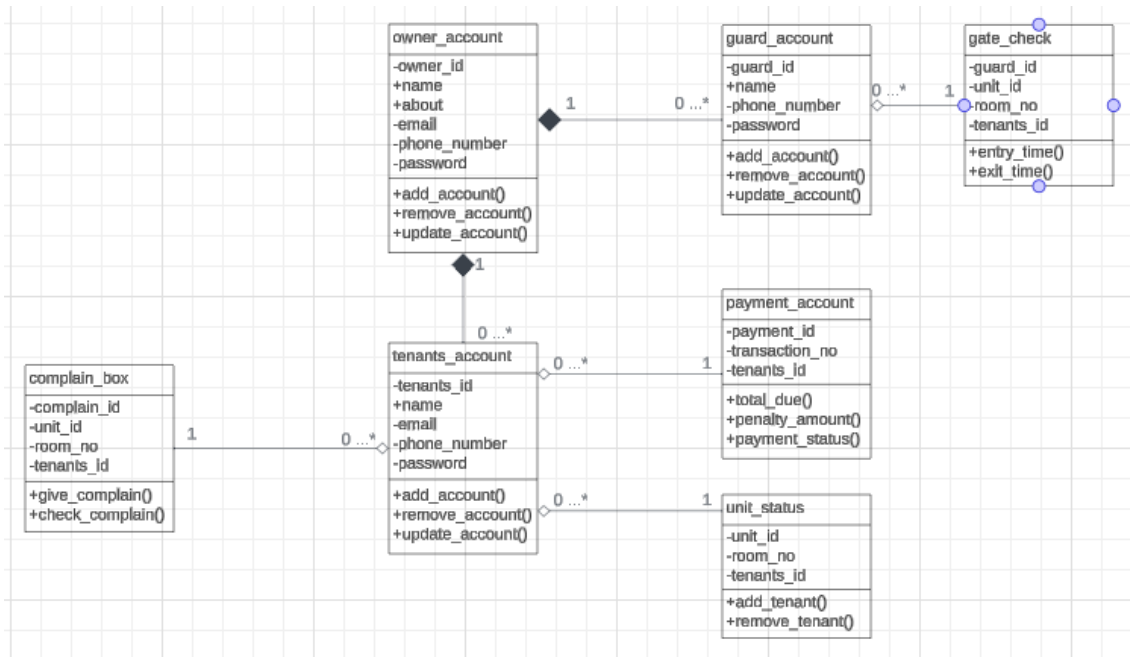
Requirement Specifications

3.1 Use case diagram with Graphical and Textual Description



3.2 Activity Diagram

3.3 Static model – class diagram



3.4 Dynamic model – sequence diagram

Architecture

4.1 Architectural model/style used

We've adopted the Layered Architecture. Layered Architecture is an architectural style that organizes a system into layers, each responsible for a specific set of functions. Here are characteristics of Layered Architecture:

- 1. Modularity:** Layered Architecture promotes modularity by organizing the system into distinct layers, each handling a specific aspect of functionality. This enhances maintainability and makes it easier to update or replace individual layers.
- 2. Separation of Concerns:** Each layer in the architecture has a specific responsibility, leading to a clear separation of concerns. This separation simplifies development, debugging, and modification of the system.
- 3. Abstraction:** Layers abstract the underlying complexity by presenting a simplified interface to the layers above. This abstraction allows developers to focus on specific functionalities without needing to understand the entire system.
- 4. Scalability:** Layered architectures are often scalable, allowing for the addition of new layers or the duplication of existing layers to accommodate growing requirements or changes in the system.
- 5. Flexibility:** The architecture is flexible as layers can be modified or replaced independently without affecting other layers, as long as the interfaces remain consistent.
- 6. Communication:** Communication between layers occurs in a predefined and well-defined manner. Typically, each layer communicates with adjacent layers, promoting a structured flow of information.
- 7. Reusability:** Components within each layer are often designed for reusability, as they can be employed in various contexts within the same layer or across different layers.
- 8. Ease of Testing:** Layered Architecture facilitates testing, as each layer can be tested independently. This allows for more effective unit testing and simplifies the identification of issues within specific layers.
- 9. Security:** Security measures can be implemented at different layers, providing a layered defense against potential threats. Access controls and validation checks can be enforced at specific layers.
- 10. Portability:** Layers can often be ported to different environments or platforms with minimal modifications, making the architecture suitable for applications that need to run in diverse settings.

4.2 Technology, software, and hardware used

1. Technology Used:

Frontend: [HTML,CSS,JavaScript]

Backend: [jQuery,PHP]

Database: [MySQL]

2. Software Used:

Operating System: [Windows, Linux]

Integrated Development Environment (IDE): [Visual Studio Code]

Version Control: [GitHub]

Web Browser: [Google Chrome]

3. Hardware:

PC:

Windows Edition: [Windows 11 Pro]

Processor: [Ryzen 5 5600G]

Ram: [16 GB]

SSD: [500 GB]

Graphics Card: [RTX 3070i]

Motherboard: [ASUS Tuf Gaming E550m E-Wifi]

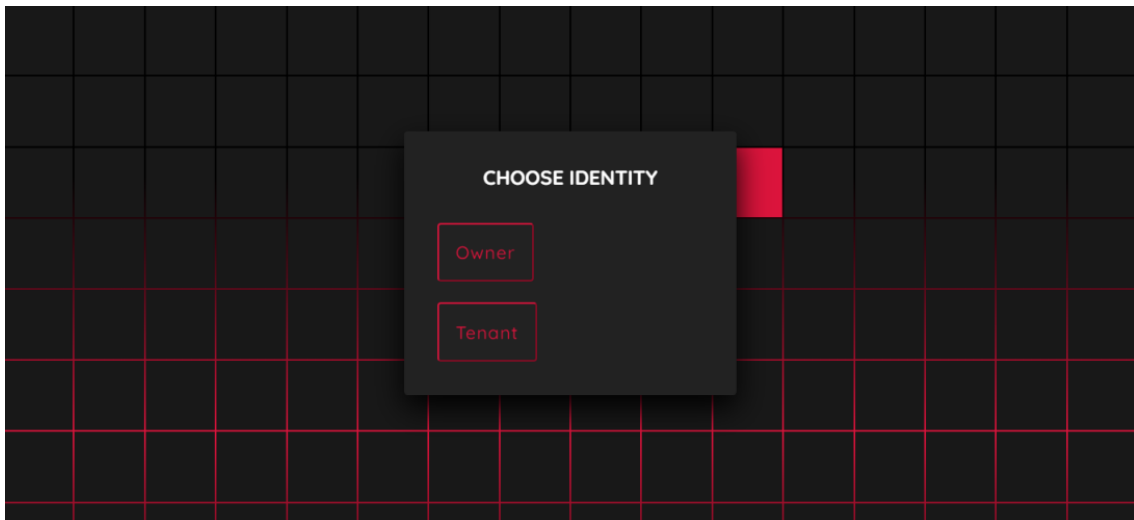
System Type: [64-Bit Operating System, x64 Based Processor]

Design

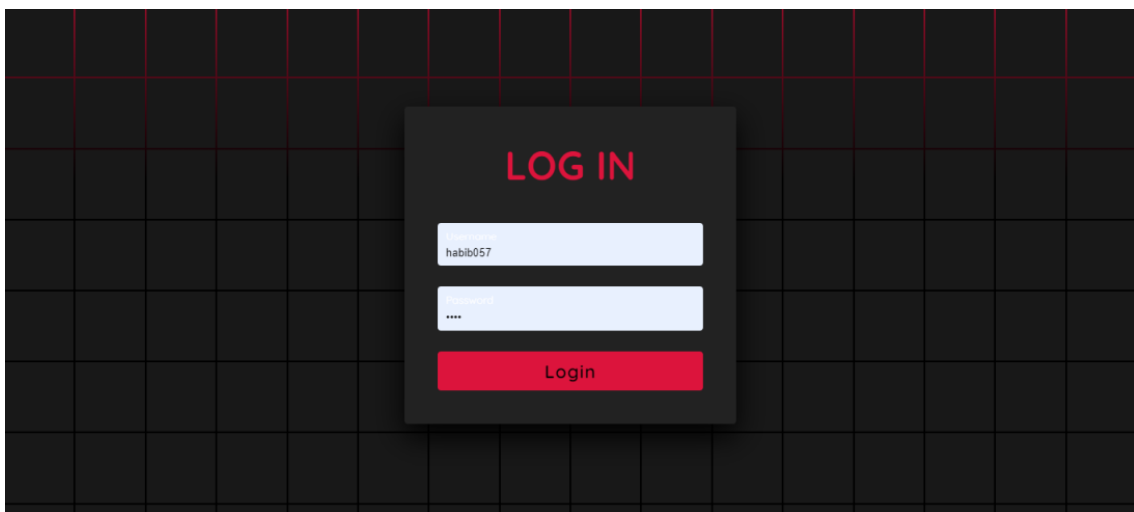
5.1 Component level design following pattern

5.2 GUI (Graphical User Interface) design

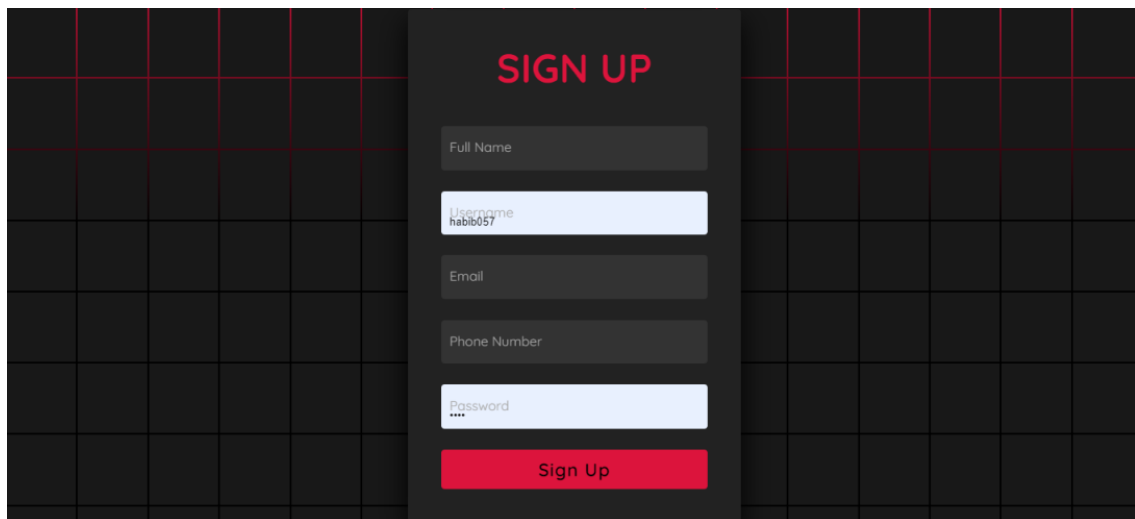
Identification:



Owner Login:

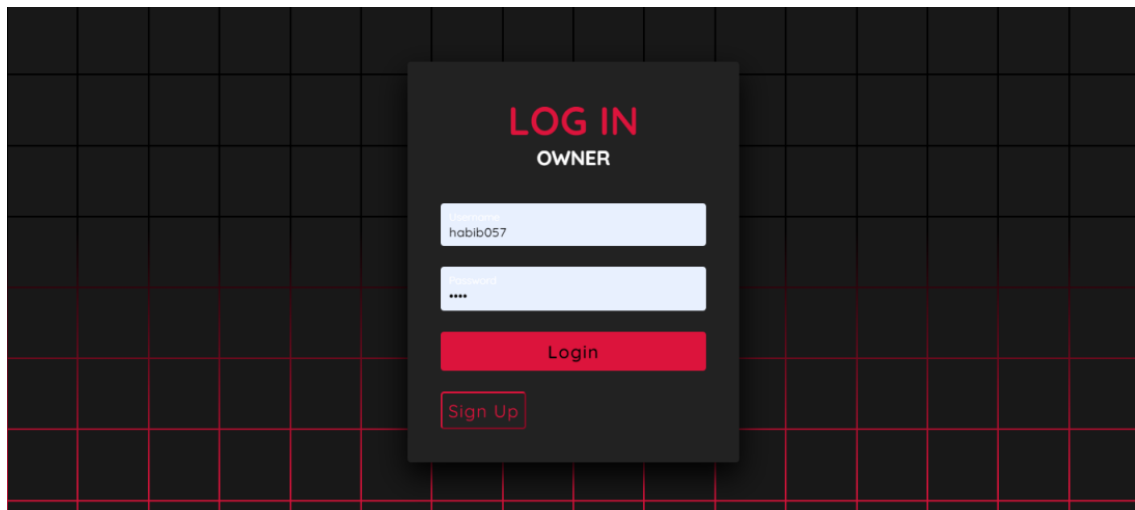


Owner SignUp:



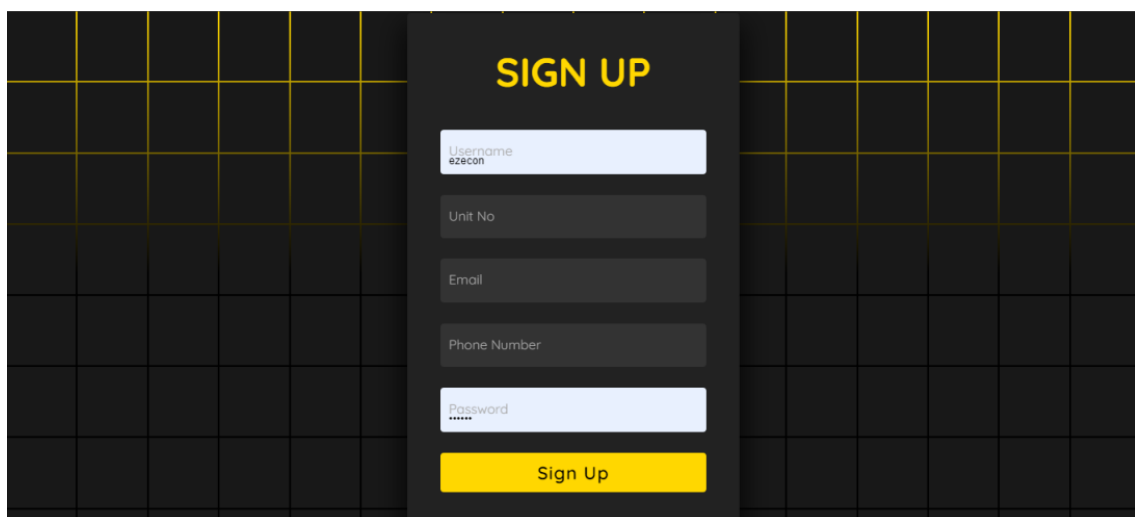
A dark-themed user registration form titled "SIGN UP" in red. The form is centered on a dark background with a red grid pattern. It contains five input fields: "Full Name", "Username" (with the placeholder "habib057"), "Email", "Phone Number", and "Password" (with three asterisks). A red "Sign Up" button is at the bottom.

Tenants Login:



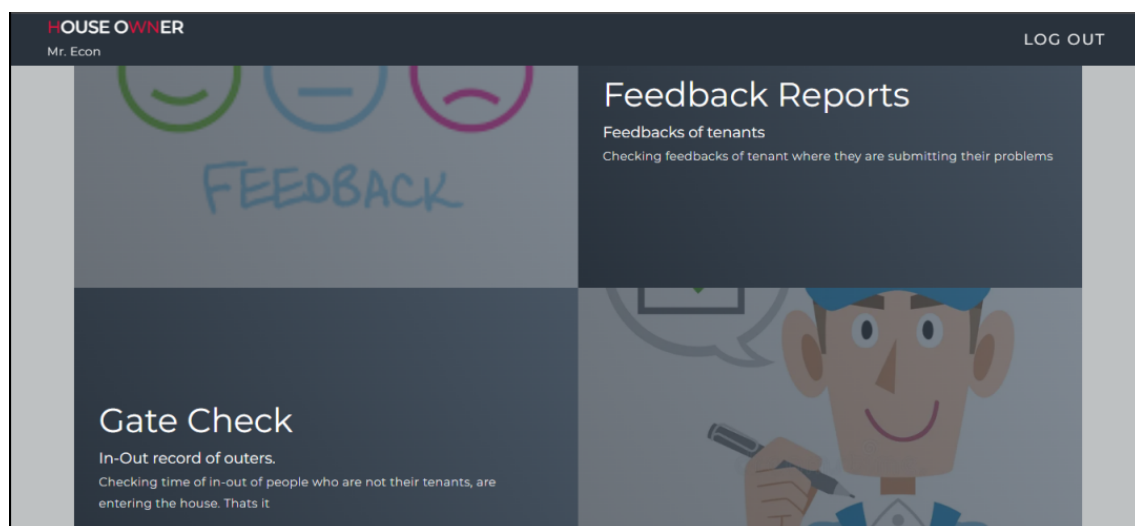
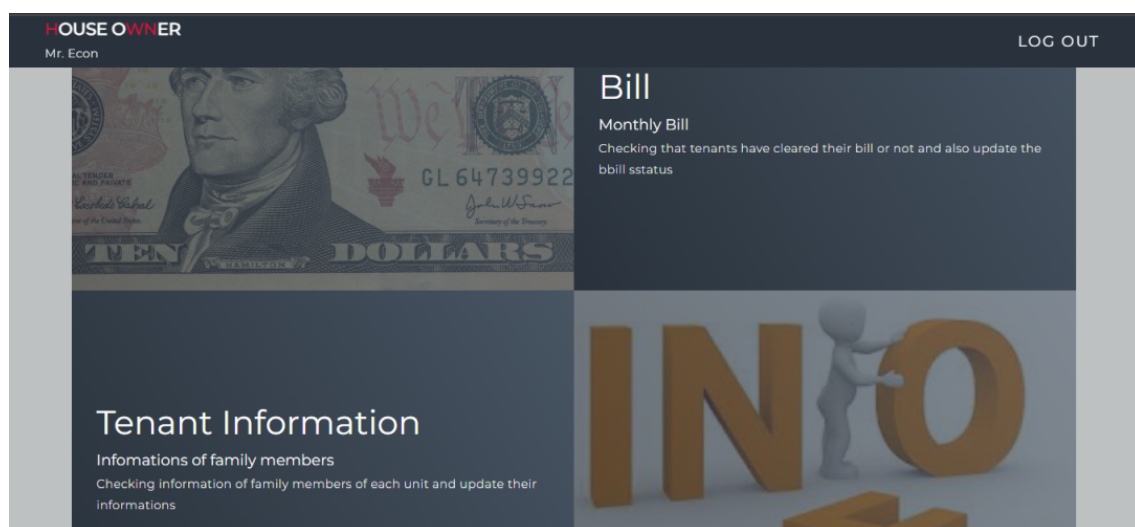
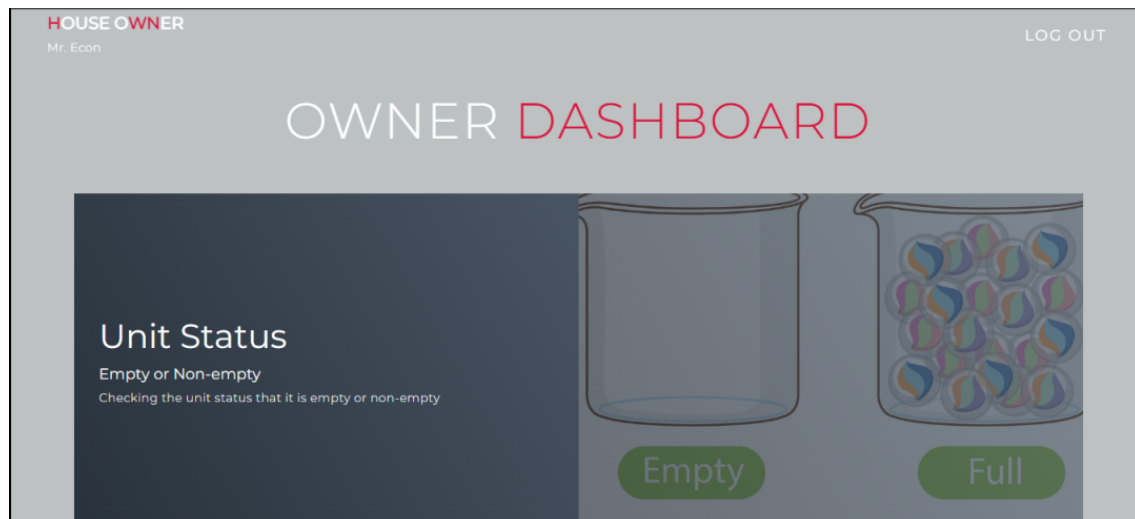
A dark-themed login form titled "LOG IN" and "OWNER" in red. The form is centered on a dark background with a red grid pattern. It contains two input fields: "Username" (with the placeholder "habib057") and "Password" (with four asterisks). There is a red "Login" button and a smaller, outlined "Sign Up" button below it.

Tenants SignUp:

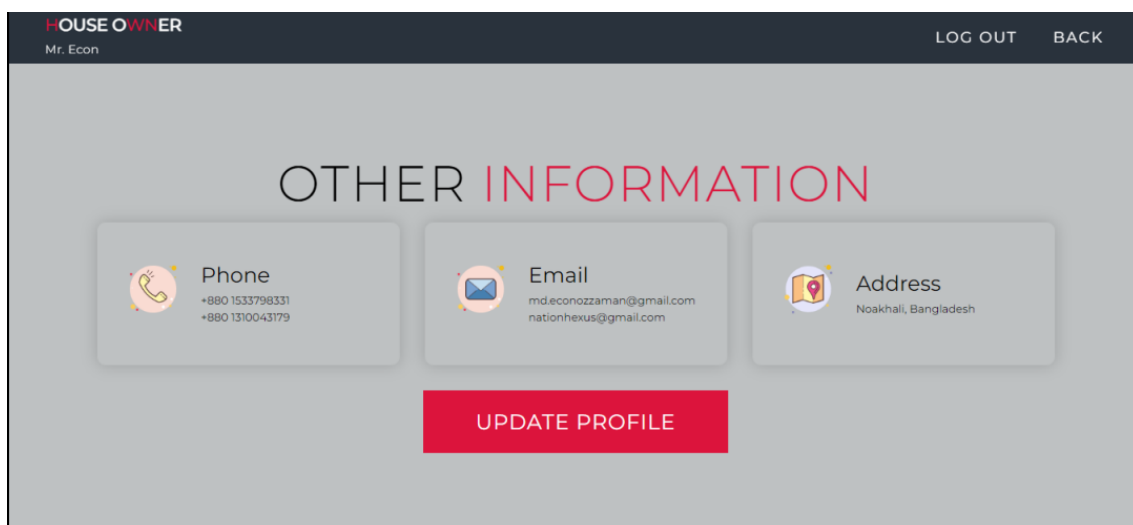


A dark-themed user registration form titled "SIGN UP" in yellow. The form is centered on a dark background with a yellow grid pattern. It contains five input fields: "Username" (with the placeholder "ezecon"), "Unit No", "Email", "Phone Number", and "Password" (with six asterisks). A yellow "Sign Up" button is at the bottom.

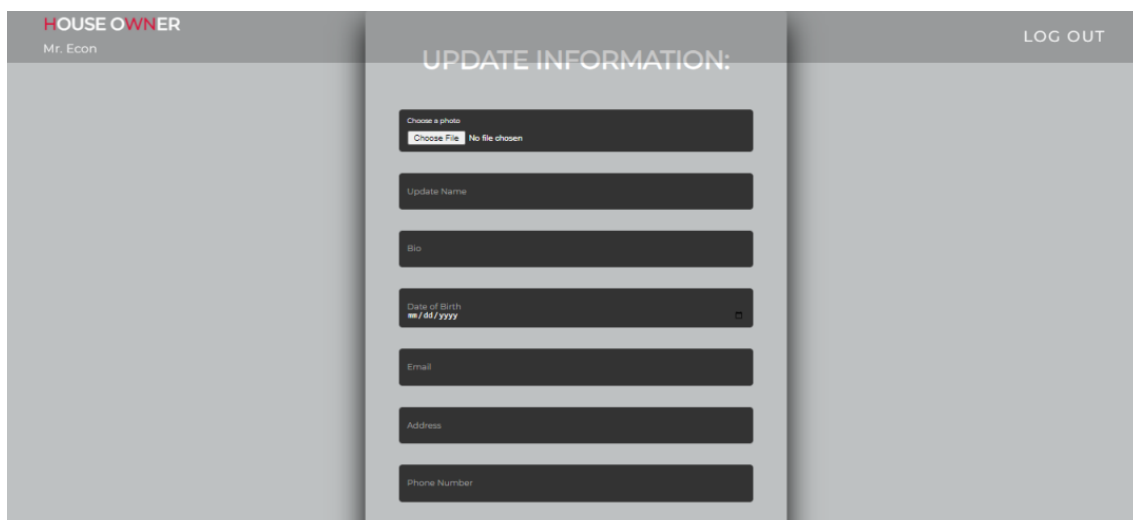
Owner Dash Board:



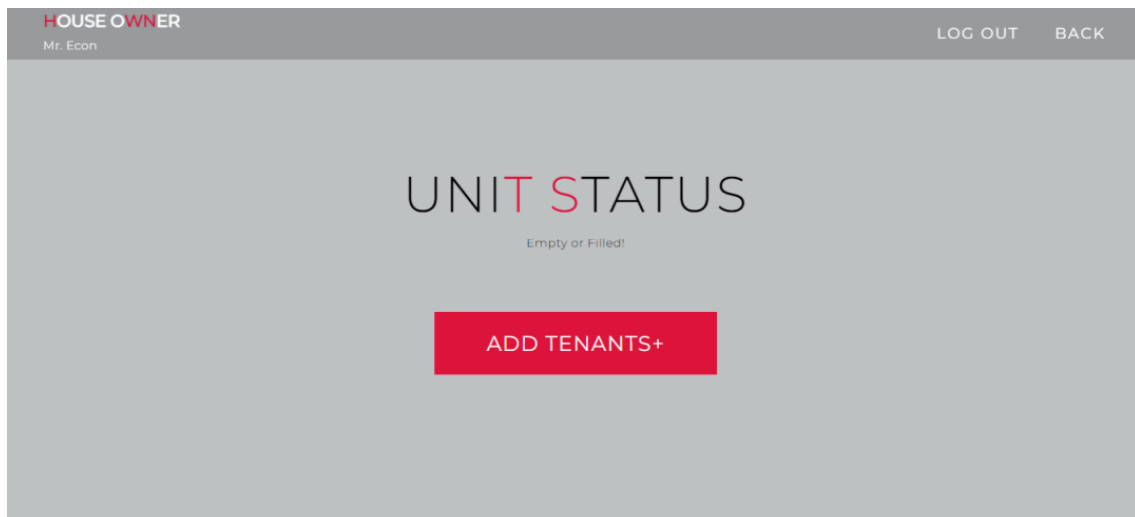
Owner Profile:



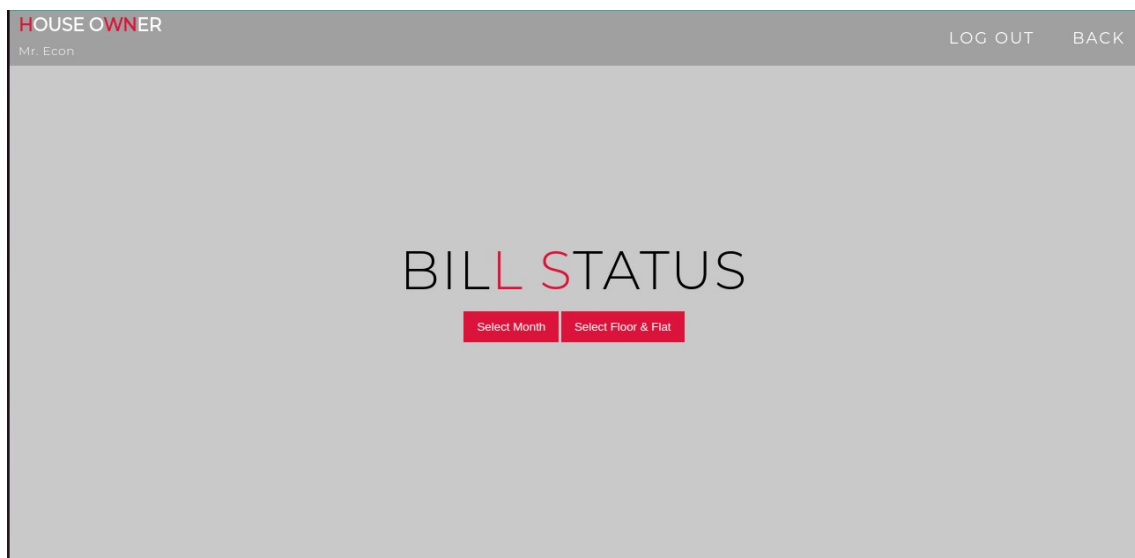
Update Owner Info.:



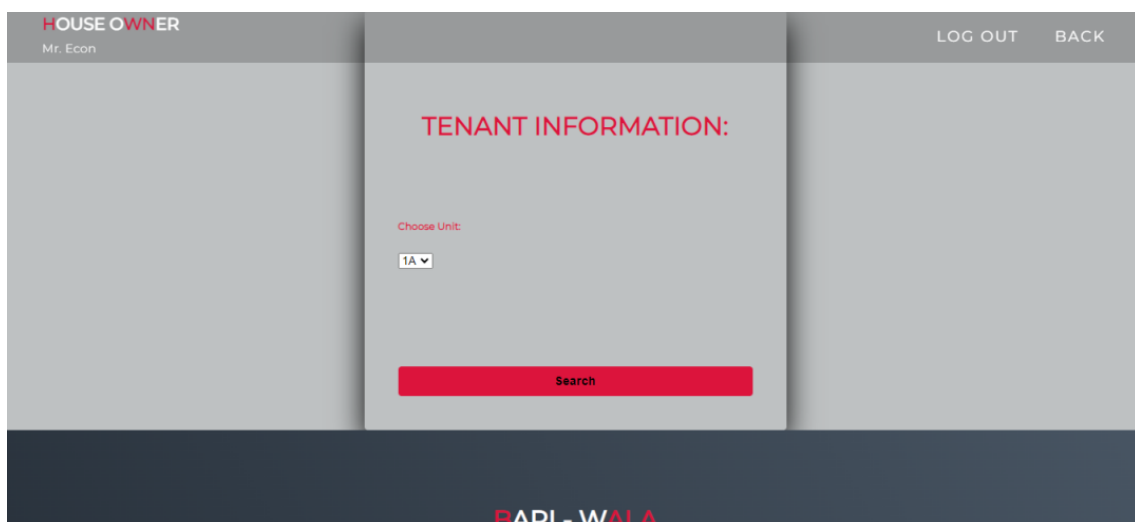
Unit Status:



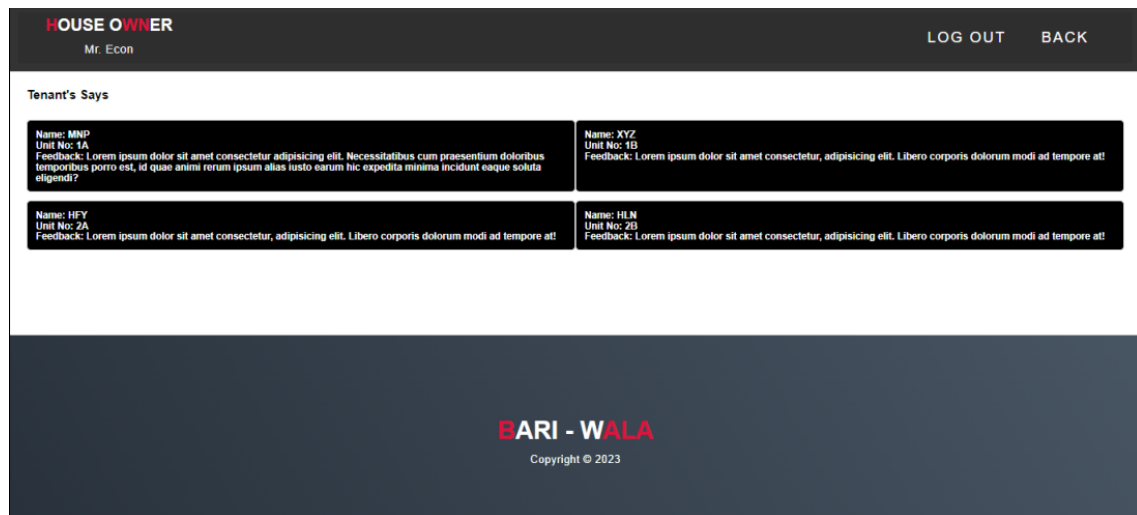
Bill Status:



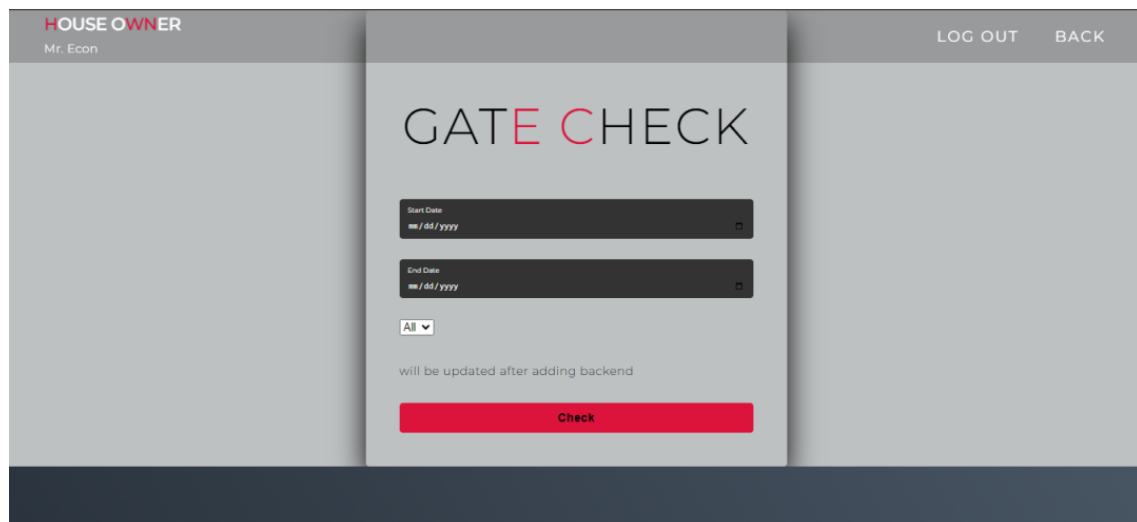
Tenants Info.:



Tenants Feedback:



Gate Check:



Testing and sustainability plan

6.1 Requirements/specifications-based system level test cases

An example table for Requirements/specifications-based system level test cases is given below:

Requirement ID	Requirement Statement	Must/ Want	Comment
R-Up-01	User can't upload image with invalid extension	Must	N/A
R-Up-02	Show error message if user upload image with invalid extension	Want	N/A
R-Gen-01	User can't generate caption without uploading image	Must	N/A
R- Gen-02	Show error message if user click generate caption option without uploading image	Want	N/A
R-Clean-01	For training the model with dataset, cleaning needs to be performed to the text dataset.	Must	N/A

Table 6.1: Requirements/specifications-based system level test cases (01)

Project Name	BARI-WALA					
Module Name	Upload					
Created By	Name 1					
Reviewed By	Name 1					
Date of Creation	15-02-21					
Date of Review	15-02-21					
Test Case ID	Scenario	Steps	Test Data	Expected Result	Actual Result	Status
TC-UP-01	Check upload image option with invalid file extension.	1. Go to the user interface 2. Click 'choose file' option 3. Upload file from PC's directory	File: new.pdf	Redirect to the initial state of user interface	Redirect to the initial state of user interface	Passed
TC-UP-02	-	-	-	-	-	Passed

Table 6.2: Requirements/specifications-based system level test cases (02)

Project Name	BARI-WALA					
Module Name	Text Cleaning					
Created By	Name 1					
Reviewed By	Name 1					
Date of Creation	15-02-21					
Date of Review	15-02-21					
Test Case ID	Scenario	Steps	Test Data	Expected Result	Actual Result	Status
TC-CL-01	-	-	-	-	-	Passed
TC-CL-02	-	-	-	-	-	Passed

Table 6.3: Requirements/specifications-based system level test cases (03)

Project Name	BARI-WALA					
Module Name	Generate Caption					
Created By	Name 1					
Reviewed By	Name 1					
Date of Creation	15-02-21					
Date of Review	15-02-21					
Test Case ID	Scenario	Steps	Test Data	Expected Result	Actual Result	Status
TC-GEN-01	-	-	-	-	-	Passed
TC-GEN-02	-	-	-	-	-	Passed

Table 6.4: Requirements/specifications-based system level test cases (04)

6.2 Traceability of test cases to use cases

Test Case ID	Requirement ID				
	R-Up-01	R-Up-02	R-Gen-01	R- Gen-02	R-Clean-01
TC-UP-01	✓				
TC-UP-02	✓				
TC-GEN-01	✓				
TC-GEN-02	✓				
TC-CL-01	✓				
TC-CL-02	✓				

Table 6.5: Traceability of test cases to use cases

6.3 Techniques used for test generation

6.4 Assessment of the goodness of your test suite

6.5 Sustainability Plan

6.5.1 Scalability

6.5.2 Flexibility / Customization

Acknowledgement

References