****

**COMSATS University Islamabad**

**Abbottabad, Pakistan**

**Residential Society Management System**

***By***

**Abdullah Ludin FA21-BSE-016**

**Muhammad Danyal Khan FA21-BSE-111**

**Osman Ghani Granday FA21-BSE-024**

***Supervisor***Hifza Ali

***Bachelor of Science in Software Engineering (2021-2025)***

**The candidate confirms that the work submitted is their own and appropriate  
 credit has been given where reference has been made to the work of others.**

****

**COMSATS University, Islamabad Pakistan**

**Residential Society Management System**

**A project presented to**

**COMSATS University Islamabad**

**Abbottabad, Pakistan**

**In partial fulfillment**

**of the requirement for the degree of**

***Bachelor of Science in Software Engineering (2021-2025)***

**By**

**Abdullah Ludin FA21-BSE-016**

**Muhammad Danyal Khan FA21-BSE-111**

**Osman Ghani Granday FA21-BSE-024**

**DECLARATION**

We hereby declare that this software, neither whole nor as a part, has been copied out from any source. It is further declared that we have developed this software and accompanied report entirely based on our personal efforts. If any part of this project is proved to be copied out from any source or found to be reproduction of some other. We will stand by the consequences. No Portion of the work presented has been submitted of any application for any other degree or qualification of this or any other university or institute of learning.

Muhammad Danyal Khan

---------------------------

Osman Ghani Granday

---------------------------

Abdullah Ludin

---------------------------

**CERTIFICATE OF APPROVAL**

It is to certify that the final year project of BS (SE) “Residential Society Management System” was developed by **Osman Ghani Granday(****CIIT/ FA21-BSE-024)**, **Muhammad Danyal Khan (CIIT/ FA21-BSE-111)** and **Abdullah Ludin (CIIT/ FA21-BSE-016)** under the supervision of “Hifza Ali” and that in (their/his/her) opinion; it is fully adequate, in scope and quality for the degree of Bachelors of Science in Software Engineering.

---------------------------------------

**Supervisor**

---------------------------------------

**External Examiner**

---------------------------------------

**Head of Department**

**(Department of Computer Science)**

**EXECUTIVE SUMMARY**

The Residential Society Management System (RSMS) is a comprehensive digital solution designed to transform how residential communities operate and interact. By combining web and mobile platforms, RSMS provides a centralized hub for residents, staff, and administrators to manage all aspects of society life efficiently. The system addresses common pain points in traditional residential management, including delayed issue resolution, poor communication channels, and lack of engagement opportunities.

RSMS offers a complete suite of features tailored to modern community living. Residents can report maintenance issues with photos and track resolution progress in real time, while staff receive instant notifications auto form the system based on category to address problems. The system includes robust event management tools, allowing both administrators and residents to organize activities with built-in voting systems (upvotes and devotes) to gauge interest. A dedicated marketplace section enables secure property listings with admin verification and direct communication between buyers and sellers.

What sets RSMS apart is its focus on transparency and community building. The platform provides feedback mechanisms where residents can rate services and view performance analytics, fostering trust between residents and management. Real-time notifications through multiple channels ensure everyone stays informed, whether through in-app alerts or emails.

Developed using the MERN stack for web interfaces and React Native for mobile application, RSMS delivers a seamless experience across all devices. The architecture supports future expansions. By replacing outdated manual processes with an integrated digital platform, RSMS elevates the standard of residential community management, creating more connected, efficient, and satisfying living environments for all stakeholders.

**ACKNOWLEDGEMENT**

All praise is to Almighty Allah who bestowed upon us a minute portion of His boundless knowledge by virtue of which we were able to accomplish this challenging task.

We are greatly indebted to our project supervisor “Hifza Ali”. Without her personal supervision, advice and valuable guidance, completion of this project would have been doubtful. We are deeply indebted to them for their encouragement and continual help during this work.

And we are also thankful to our parents and family who have been a constant source of encouragement for us and brought us the values of honesty & hard work.

Muhammad Danyal Khan

---------------------------

Osman Ghani Granday

---------------------------

Abdullah Ludin

---------------------------

**ABBREVIATIONS**

|  |  |
| --- | --- |
| **SRS** | Software Require Specification |
| **PC** | Personal Computer |
| **SDD** | Software Design Description |
| **Vs** **Code** | Visual Studio Code |
| **API** | Application Programming Interface |
| **IDE** | Integrated Development Environment |
| **UML** | Unified Modeling Language |
| **SQL** | Structured Query language |
| **HCI** | Human Computer Interaction |
| **DBMS** | Database Management Systems |
| **OP** | Operating System |
| **Q & A** | Question and Answer |

**TABLE OF CONTENTS**

[**1. Introduction** 1](#_Toc207281922)

[1.1. Brief 1](#_Toc207281923)

[1.2. Relevance to Course Modules 1](#_Toc207281924)

[1.3. Project Background 1](#_Toc207281925)

[1.4. Literature Review 2](#_Toc207281926)

[1.5. Analysis from Literature Review 2](#_Toc207281927)

[1.6. Methodology and Software Lifecycle for This Project 3](#_Toc207281928)

[1.6.1. Rationale for Agile 3](#_Toc207281934)

[**2. Problem Definition** 4](#_Toc207281935)

[2.1. Problem Statement 4](#_Toc207281940)

[2.2. Deliverable and Development Requirements 4](#_Toc207281941)

[**3. Requirement Analysis** 5](#_Toc207281942)

[3.1. Use Cases Diagram(s) 5](#_Toc207281947)

[3.2. Detailed Use Cases 9](#_Toc207281948)

[3.3. Functional Requirements 21](#_Toc207281949)

[3.4. Non-Functional Requirements 21](#_Toc207281950)

[**4. Design and Architecture** 23](#_Toc207281951)

[4.1. Data Representation 23](#_Toc207281952)

[4.2. Process Flow/Representation 24](#_Toc207281960)

[4.3. Design Models: 32](#_Toc207281961)

[**5. Implementation** 42](#_Toc207281962)

[5.1. Algorithm 42](#_Toc207281963)

[5.2. External APIs 44](#_Toc207281964)

[5.3. User Interface: 45](#_Toc207281965)

[**6. Testing and Evaluation:** 59](#_Toc207281966)

[6.1. Manual Testing 59](#_Toc207281967)

[6.1.1. System Testing 59](#_Toc207281968)

[6.1.2. Unit Testing 59](#_Toc207281969)

[6.1.3. Functional Testing 64](#_Toc207281970)

[6.2. Integration Testing 67](#_Toc207281971)

[**7. Conclusion and Future Work** 69](#_Toc207281972)

[7.1. Conclusion 69](#_Toc207281973)

[7.2. Future Work 69](#_Toc207281974)

[**8.** **References** 71](#_Toc207281975)

**LIST OF FIGURES**

Fig 3.1 ER Diagram 16

Fig 3.1 Use Case Diagram 17

Fig 3.3 Use Case Diagram 18

Fig 3.4 Use Case Diagram 18

Fig 3.5 Use Case Diagram 19

Fig 4.1.1 ER Diagram 34

Fig 4.2.1 DFD Level 0 34

Fig 4.2.2 DFD Level 1 34

Fig 4.2.3 DFD Level 2 User Management 37

Fig 4.2.4 DFD Level 2 Announcement Management 38

Fig 4.2.5 DFD Level 2 Event Management 39

Fig 4.2.6 DFD Level 2 Issue Management 40

Fig 4.2.7 DFD Level 2 Task Management 41

Fig 4.2.6 DFD Level 2 Marketplace 42

Fig 4.3.1 Generic Login 43

Fig 4.3.2 SuperAdmin Login/Signup 43

Fig 4.3.3 Create SocietyAdmin 44

Fig 4.3.4 Add Notice 44

Fig 4.3.5 Manager Users 45

Fig 4.3.6 Update Issue Status 45

Fig 4.3.7 Property Verification 46

Fig 4.3.8 Generic Logout 46

Fig 4.3.9 Resident Registeration 47

Fig 4.3.10 Resident Dashboard 47

Fig 4.3.11 View Announcements 48

Fig 4.3.12 View and create Events 48

Fig 4.3.13 Resident and Marketplace 49

Fig 4.3.14 Personalized Settings 49

Fig 4.3.15 Staff Task Management 50

Fig 4.3.16 Visitor Marketplace 50

**LIST OF TABLES**

Table 3.1 System Analysis with proposed project 13

Table 3.1 Fully Dressed Use Case Register 20

Table 3.2 Fully Dressed Use Case Login 20

Table 3.3 Create System Admin 22

Table 3.4 Register 22

Table 3.5 Login 23

Table 3.6 Add Notices 23

Table 3.7 Manage Users 23

Table 3.8 View Dashboard 24

Table 3.9 Approve Events 24

Table 3.10 Issue Reports 24

Table 3.11 Verify Property Listings 25

Table 3.12 Resident Register 25

Table 3.13 Resident Login 26

Table 3.14 Resident Dashboard 26

Table 3.15 View Announcements 27

Table 3.16 Report Issues 27

Table 3.17 Add Events 28

Table 3.18 Add Listings 28

Table 3.19 Personalized Settings 29

Table 3.20 Staff Login 29

Table 3.21 Task Assigned to Staff 30

Table 3.22 Update Task Status by Staff 31

Table 3.23 Browse and View Properites 31

Table 5.1 External APIs used in Project 54

Table 6.1 UTC Login as SuperAdmin 58

Table 6.2 UTC Login as System Manager 58

Table 6.3 UTC Login as Resident 59

Table 6.4 UTC Login as Staff 59

Table 6.5 UTC Register as Admin and Resident 60

Table 6.6 UTC Report an Issue by Resident 60

Table 6.7 UTC Property Listing 61

Table 6.8 UTC Event Management 61

Table 6.9 FTC Case Login with different Roles 62

Table 6.10 FTC Registeration for Residents and System Managers 63

Table 6.11 FTC Issue Reporting 63

Table 6.12 FTC Property Listing 64

Table 6.13 FTC Event Creation by Resident 64

Table 6.14 Integeration Test Cases 65

# **1. Introduction**

The Residential Society Management System is about bringing collaborative platforms for residents and administrators, with a focus on ease of use and enhanced features like issue resolution, event management, announcements, property listings and more.

## 1.1. Brief

The Residential Society Management System (RSMS) is a web and mobile application designed to automate and streamline operations in residential societies. It integrates functionalities like issue tracking, event management, property listings, and announcements into a unified platform. Developed using the MERN stack (web) and React Native Expo (mobile), RSMS addresses inefficiencies in manual processes by providing real-time communication, role-based access (SuperAdmin, SystemAdmin, Resident, Staff), and scalable architecture. Key outcomes include improved transparency, reduced administrative overhead, and enhanced resident engagement.

## 1.2. Relevance to Course Modules

Software Requirement Engineering: SRS documentation (use cases, functional/non-functional requirements).

Web Technologies: MERN stack (MongoDB, Express.js, React, Node.js) for web development.

Database Systems I: MongoDB for real-time data storage and retrieval.

Software Design & Architecture (SDA): Three Tier architecture for modularity.

Introduction to Software Engineering: Agile methodology for iterative development.

## 1.3. Project Background

Traditional residential societies rely on manual processes for issue resolution, event coordination, and communication, leading to delays and inefficiencies. RSMS modernizes these operations by digitizing workflows, enabling real-time updates, and fostering community interaction through a centralized platform.

## 1.4. Literature Review

The following table 1.1 below shows the related system analysis to our RSMS system.

Table 1.1 related System Analysis with proposed related project solution

|  |  |  |
| --- | --- | --- |
| Application Name | Weakness | Proposed Project Solution |
| * RSMS | The app doesn’t support auto billing/payments from Users. | Our app contains Role based access control with proper authentication and authorization. Our app also supports Issue Reporting, Event Management, profile customization, and announcements for Resident via Staff handling by Admins |
| * MyGate | This app only focuses on Residents and no platform for Staff, just admin and Resident Interaction. | Our app contains Role based access control with proper authentication and authorization. Our app also supports Issue Reporting, Event Management, profile customization, and announcements for Resident via Staff handling by Admins |

## 1.5. Analysis from Literature Review

Unlike commercial tools with limited customization, RSMS offers tailored features (e.g., role-specific dashboards, marketplace integration) while prioritizing affordability and ease of deployment for small to mid-sized societies. While current systems focus narrowly on security.

RSMS focuses on resident engagement to make the community purpose more environmentally friendly.

## 1.6. Methodology and Software Lifecycle for This Project

An agile iterative methodology is being used for our project.

RSMS fosters the benefits of agile methodology by focusing on user’s needs and priorities.



### 1.6.1. Rationale for Agile

The Agile methodology was selected for RSMS development due to its proven efficiency in delivering working software incrementally. Agile's adaptive nature allows for continuous improvement throughout the project lifecycle, accommodating changing requirements even in later stages. This approach significantly reduces project failure risks while enhancing team productivity and stakeholder satisfaction. The iterative cycles enable regular feedback incorporation, ensuring the final product aligns closely with user needs and market demands.

# **2. Problem Definition**

RSMS solves the needs like lack in communication, and lack of feedback and promotes the ease of platform for all kinds of stakeholders.



## 2.1. Problem Statement

Manual society management systems suffer from:

Delayed issue resolution.

Fragmented communication (notices, events) means they use paper or posters source of usage for better reach and communication.

Lack of transparency in task allocation.

RSMS solves these by automating workflows and providing a unified interface.

## 2.2. Deliverable and Development Requirements

The Residential Society Management System (RSMS) will be developed as a comprehensive platform with core functional modules. The system will enable user account creation and role-based access management. Residents can report community issues which will be visible only to administrators. The event management module allows residents to propose events, which then require approval from the event manager before becoming active. Issues are automatically assigned to dedicated staff members based on the residential block where the problem was reported, with each block having its own assigned maintenance team. Real-time notifications will keep stakeholders updated on issue status changes. A dedicated marketplace will feature verified property listings with direct contact options. The technical implementation will utilize React Native for mobile development, the MERN stack (MongoDB, Express, React, Node.js) for web applications, and MongoDB as the primary database management system.

# **3. Requirement Analysis**

Requirement Analysis clearly shows your system behavior via visualizing diagrams that help developers and even normal people/clients to understand your system.  
It helps you by following the consistent architecture throughout the development process.



## 3.1. Use Cases Diagram(s)

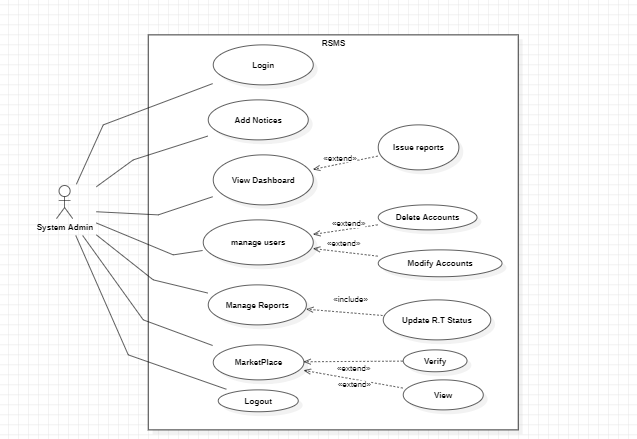
The following figures 3.1 below show the use case diagram for the RSMS SocietyAdmin.

A diagram of a software system

Description automatically generated

*Figure 3.1 Use Case Diagram for Super Admin*

The following figures 3.2 below shows the use case diagram for the RSMS Society Admin.



*Figure 3.2 Use Case Diagram for System Admin*

The following figures 3.3 below shows the use case diagram for the RSMS Society Admin.

A diagram of a diagram

AI-generated content may be incorrect.

*Figure 3.3 Use Case Diagram for resident*

A diagram of a software system

Description automatically generated with medium confidence

*Figure 3.4 Use Case Diagram for Staff*

The following figures 3.5 below shows the use case diagram for the RSMS Society Admin.

A diagram of a marketplace

Description automatically generated

*Figure 3.5 Use Case Diagram for Visitor*

## 3.2. Detailed Use Cases

Detailed use cases help our system develop in a more professional manner by implementing each feature in different categories, so to help the developer team implement the system in a more accurate way.

* + 1. **Fully Dressed Use Case: Register**

The following table 3.1 shows the fully dressed use case for Register:

*Table 3.1 Fully Dressed Use Case Register*

|  |  |
| --- | --- |
| Use Case Name | Register |
| Actor | SuperAdmin |
| Description | SuperAdmin registers on the system by providing necessary details. |
| Preconditions | |  | | --- | | The system must be accessible, and SuperAdmin must not already have an account. |  |  | | --- | |  | |
| |  | | --- | | Main Flow |  |  | | --- | |  | | 1. SuperAdmin navigates to the registration page. 2. SuperAdmin enters required details (e.g., name, email, password). 3. The system validates the input. 4. The system creates the account and displays a success message. |
| Post conditions | |  | | --- | | SuperAdmin account is successfully created. |  |  | | --- | |  | |
| Exceptions | - If the email is already registered, the system notifies the user. |

* + 1. **Fully Dressed Use Case: Login**

The following table 3.2 shows the fully dressed use case for Login:

*Table 3.2 Fully Dressed Use Case Login*

|  |  |
| --- | --- |
| Use Case Name | Login |
| Actor | SuperAdmin |
| Description | SuperAdmin logs into the system. |
| Preconditions | |  | | --- | | SuperAdmin must already have a registered account. |  |  | | --- | |  | |
| |  | | --- | | Main Flow |  |  | | --- | |  | | 1. SuperAdmin navigates to the login page. 2. SuperAdmin enters valid credentials (email and password). 3. The system authenticates the user. 4. SuperAdmin is granted access to the dashboard. |
| Post conditions | |  |  |  | | --- | --- | --- | | |  | | --- | | SuperAdmin is successfully logged in. |  |  | | --- | |  | |  |  | | --- | |  | |
| Exceptions | |  | | --- | | SuperAdmin is successfully logged in. |  |  | | --- | |  | |

* + 1. **Fully Dressed Use Case: Create System Admin**

The following table 3.3 shows the fully dressed use case for Create a System Admin:

*Table 3.3 Fully Dressed Use Case Create System Admin*

|  |  |
| --- | --- |
| Use Case Name | Create System Admin |
| Actor | SuperAdmin |
| Description | SuperAdmin creates a System Admin account. |
| Preconditions | |  | | --- | | The system must be accessible, and SuperAdmin must not already have an account. |  |  | | --- | |  | |
| |  | | --- | | Main Flow |  |  | | --- | |  | | 1. SuperAdmin navigates to the "Create System Admin" section. 2. SuperAdmin enters System Admin details (e.g., name, email, role permissions). 3. The system validates and saves the data. |
| Post conditions | |  | | --- | | System Admin account is successfully created. |  |  | | --- | |  | |
| Exceptions | - System error prevents the account creation. |

* + 1. **Fully Dressed Use Case: Logout**

The following table 3.4 shows the fully dressed use case for Logout:

*Table 3.4 Fully Dressed Use Case Logout*

|  |  |
| --- | --- |
| Use Case Name | Logout |
| Actor | SuperAdmin |
| Description | SuperAdmin logs out of the system |
| Preconditions | |  | | --- | | SuperAdmin must be logged in. |  |  | | --- | |  | |
| |  | | --- | | Main Flow |  |  | | --- | |  | | 1. SuperAdmin clicks the "Logout" button. 2. The system ends the session and redirects to the login page. |
| Post conditions | |  | | --- | | SuperAdmin is successfully logged out, and the session is terminated. | |
| Exceptions | - System error prevents logging out; the user remains logged in. |

* + 1. **Fully Dressed Use Case: Login**

The following table 3.5 shows the fully dressed use case for Login:

*Table 3.5 Fully Dressed Use Case Login*

|  |  |
| --- | --- |
| Use Case Name | Login |
| Actor | SystemAdmin |
| Description | SystemAdmin logs in the system |
| Preconditions | |  | | --- | | SystemAdmin must be registered/created first. |  |  | | --- | |  | |
| |  | | --- | | Main Flow |  |  | | --- | |  | | 1. SystemAdmin clicks the "Login" button. 2. The SystemAdmin enters the credentials like email and password to continue to dashboard. 3. The SystemAdmin successfully logged in to the system |
| Post conditions | |  | | --- | | SystemAdmin is successfully Logged in. | |
| Exceptions | - System error prevents logging in; the user remains logged out. |

* + 1. **Fully Dressed Use Case: Add Notices**

The following table 3.6 shows the fully dressed use case for Add Notices:

*Table 3.6 Fully Dressed Use Case Add Notices*

|  |  |
| --- | --- |
| Use Case Name | Add Notices |
| Actor | SystemAdmin |
| Description | SystemAdmin publishes notices for residents on the system. |
| Preconditions | |  | | --- | | SystemAdmin must be logged in. | |
| |  | | --- | | Main Flow |  |  | | --- | |  | | 1. SystemAdmin navigates to the "Publish Notices" section.  2. SystemAdmin creates a new notice by entering a title, description, and relevant dates.  3. The system validates the notice details.  4. The system publishes the notice and notifies residents. |
| Post conditions | The notice is successfully published and viewable by residents. |
| Exceptions | - System error prevents the notice from being saved or published. |

* + 1. **Fully Dressed Use Case: Manage Users**

The following table 3.7 shows the fully dressed use case for Manager Users:

*Table 3.7 Fully Dressed Use Case Manage Users*

|  |  |
| --- | --- |
| Use Case Name | Manage Users |
| Actor | SystemAdmin |
| Description | Allows the system admin to manage user accounts. |
| Preconditions | Admin is logged in. |
| |  | | --- | | Main Flow |  |  | | --- | |  | | Admin navigates to the User Management section. |
| Post conditions | User accounts are managed (added, modified, deleted). |
| Exceptions | User management actions fail. |

* + 1. **Fully Dressed Use Case: View Dashboard**

The following table 3.8 shows the fully dressed use case for View Dashboard:

*Table 3.8 Fully Dressed Use Case View Dashboard*

|  |  |
| --- | --- |
| Use Case Name | Manage Users |
| Actor | SystemAdmin |
| Description | Allows the system admin to view the system dashboard. |
| Preconditions | Admin is logged in. |
| |  | | --- | | Main Flow |  |  | | --- | |  | | Admin navigates to the Dashboard section. |
| Post conditions | Dashboard is displayed with relevant information. |
| Exceptions | Dashboard data is unavailable. |

* + 1. **Fully Dressed Use Case: Approve Events**

The following table 3.9 shows the fully dressed use case for Approve Events:

*Table 3.9 Fully Dressed Use Case Approve Events*

|  |  |
| --- | --- |
| Use Case Name | Approve Events |
| Actor | SystemAdmin |
| Description | SystemAdmin approves or rejects event requests submitted by residents. |
| Preconditions | The SystemAdmin must be logged in. |
| |  | | --- | | Main Flow |  |  | | --- | |  | | 1. SystemAdmin navigates to the "Event Requests" section.  2. SystemAdmin views the list of pending event requests.  3. SystemAdmin selects a request and reviews its details.  4. SystemAdmin approves or rejects the request.  5. The system notifies the resident about the decision. |
| Post conditions | The resident is notified about the event approval/rejection. |
| Exceptions | System error prevents the decision from being saved. |

* + 1. **Fully Dressed Use Case: Manage Issue Reports**

The following table 3.10 shows the fully dressed use case for Issue Reports:

*Table 3.10 Fully Dressed Use Case Issue Reports*

|  |  |
| --- | --- |
| Use Case Name | Update Issue RT Status |
| Actor | SystemAdmin |
| Description | SystemAdmin updates Issue status like Resolved or pending submitted by residents. |
| Preconditions | The SystemAdmin must be logged in. |
| |  | | --- | | **Main** Flow |  |  | | --- | |  | | 1. SystemAdmin navigates to the "Reported Issues" section. 2. SystemAdmin reviews the list of reported issues. 3. SystemAdmin selects an issue to review details. 4. SystemAdmin takes the necessary action and marks the issue as resolved in the system. 5. The system notifies the reporter about the resolution. |
| Post conditions | The issue is updated as resolved |
| Exceptions | System error prevents the status update. |

* + 1. **Fully Dressed Use Case: Verify Property Listings**

The following table 3.11 shows the fully dressed use case for Verify Property Listings:

*Table 3.11 Fully Dressed Use Case Verify Property Listings*

|  |  |
| --- | --- |
| Use Case Name | Update Issue RT Status |
| Actor | SystemAdmin |
| Description | SystemAdmin verifies property listings submitted by residents for approval before they are published on the marketplace. |
| Preconditions | The SystemAdmin must be logged in. Property listing must be submitted by a resident. |
| |  | | --- | | Main Flow |  |  | | --- | |  | | 1. SystemAdmin navigates to the "Property Listings" section. 2. SystemAdmin views the list of pending property submissions. 3. SystemAdmin selects a specific listing to review its details (e.g., property description, images, price). 4. SystemAdmin verifies the information for authenticity and compliance with the rules. 5. SystemAdmin approves or rejects the listing. 6. The system updates the listing status and notifies the resident of the decision. |
| Postconditions | Approved listings are published on the marketplace and visible to residents.  Rejected listings remain unpublished, and the resident is notified with the reason for rejection. |
| Exceptions | System error prevents listing status updates or notifications. |

* + 1. **Fully Dressed Use Case: Register**

The following table 3.12 shows the fully dressed use case for Register:

*Table 3.12 Fully Dressed Use Case Register*

|  |  |
| --- | --- |
| Use Case Name | Register |
| Actor | Resident |
| Description | A new resident registers to access the system. |
| Preconditions | The Resident shall enter the credentials which are required. |
| |  | | --- | | Main Flow |  |  | | --- | |  | | 1. Resident opens the registration page. 2. Fills in personal details (e.g., name, email, phone). 3. Sets a password. 4. Submits the form for approval. 5. The system sends the data to SystemAdmin for verification. |
| Post conditions | Registration request is submitted, and the Resident awaits approval. |
| Exceptions | System fails to send the request due to connectivity issues. |

* + 1. **Fully Dressed Use Case: Login**

The following table 3.13 shows the fully dressed use case for Login:

*Table 3.13 Fully Dressed Use Case Login*

|  |  |
| --- | --- |
| Use Case Name | Login |
| Actor | Resident |
| Description | A resident log into the system |
| Preconditions | Resident must have an approved account. |
| |  | | --- | | Main Flow |  |  | | --- | |  | | 1. Resident enters username and password. 2. System authenticates the credentials. 3. Resident is redirected to the dashboard. |
| Postconditions | Resident is successfully logged in. |
| Exceptions | System error prevents login. |

* + 1. **Fully Dressed Use Case: View Dashboard**

The following table 3.14 shows the fully dressed use case for View Dashboard:

*Table 3.14 Fully Dressed Use Case View Dashboard*

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Use Case Name** | | View Dashboard |
| Actor | Resident |
| Description | Resident views personalized dashboard with Analytics and notifications. |
| Preconditions | Resident must be logged in. |
| |  | | --- | | Main Flow |  |  | | --- | |  | | 1. Resident navigates to the dashboard. 2. System displays personalized data such as issue status, event reminders, and announcements. |
| Post conditions | Dashboard information is displayed. |
| Exceptions | None. |

* + 1. **Fully Dressed Use Case: View Announcements**

The following table 3.15 shows the fully dressed use case for View Announcements:

*Table 3.51 Fully Dressed Use Case View Announcements*

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Use Case Name** | | View Announcements |
| Actor | Resident |
| Description | Resident views announcements posted by the SystemAdmin. |
| Preconditions | Resident must be logged in. |
| |  | | --- | | Main Flow |  |  | | --- | |  | | 1.Resident opens the "Announcements" tab. 2. System displays the latest announcements. |
| Post conditions | Announcements information is displayed. |
| Exceptions | None. |

* + 1. **Fully Dressed Use Case: Report Issues**

The following table 3.16 shows the fully dressed use case for Report Issues:

*Table 3.16 Fully Dressed Use Case Reporting Issues*

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Use Case Name** | | Report Issues |
| Actor | Resident |
| Description | Resident reports maintenance or other issues. |
| Preconditions | Resident must be logged in. |
| |  | | --- | | Main Flow |  |  | | --- | |  | | 1. Resident navigates to the "Report Issues" section. 2. Fills in issue details (description, priority, etc.). 3. Submits the issue. 4. System logs the issue and notifies the SystemAdmin. |
| Post conditions | Issue is successfully recorded. |
| Exceptions | System fails to submit the issue due to connectivity. |

* + 1. **Fully Dressed Use Case: Add Events**

The following table 3.17 shows the fully dressed use case for Add Events:

*Table 3.17 Fully Dressed Use Case Add Events*

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Use Case Name** | | Add Events |
| Actor | Resident |
| Description | Resident creates an event for the society. |
| Preconditions | Resident must be logged in. |
| |  | | --- | | Main Flow |  |  | | --- | |  | | 1. Resident navigates to the "Add Events" section. 2. Fills in event details (title, date, description). 3. Submits the event for approval. |
| Post conditions | Event is pending approval or added. |
| Exceptions | System error prevents submission. |

* + 1. **Fully Dressed Use Case: View Marketplace and Add Listings**

The following table 3.18 shows the fully dressed use case for Add Listings:

*Table 3.18 Fully Dressed Use Case Add Listings*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | **Use Case Name** | | | **View Marketplace and Add Listings** | | --- |  |  | | --- | |  | |
| Actor | Resident |
| Description | Resident views property listings and adds their own. |
| Preconditions | Resident must be logged in. |
| |  | | --- | | Main Flow |  |  | | --- | |  | | 1. Resident navigates to the "Marketplace" tab. 2. Views available property listings. 3. Clicks "Add New Listing". 4. Fills in listing details (property type, price, location). 5. Submits the listing for SystemAdmin approval. |
| Post conditions | Listing is successfully submitted. |
| Exceptions | SystemAdmin rejects the listing due to non-compliance. |

* + 1. **Fully Dressed Use Case: Personalized Settings**

The following table 3.19 shows the fully dressed use case for Personalized Settings:

*Table 3.19 Fully Dressed Use Case Personalized Settings*

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Use Case Name** | | Personalized Settings |
| Actor | Resident |
| Description | |  | | --- | | Resident manages personal settings like changing password or deleting account. |  |  | | --- | |  | |
| Preconditions | Resident must be logged in. |
| |  | | --- | | Main Flow |  |  | | --- | |  | | 1. Resident navigates to "profile settings". 2. Updates relevant fields (e.g., contact details),  also change password. 3. Confirms changes. 4. System saves the updates. |
| Post conditions | Personal settings are updated. |
| Exceptions | System error prevents updates. |

* + 1. **Fully Dressed Use Case: Login**

The following table 3.20 shows the fully dressed use case for Login:

*Table 3.20 Fully Dressed Use Case Login*

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Use Case Name** | | Log in |
| Actor | Staff |
| Description | |  | | --- | | Staff logs into the RSMS system. |  |  | | --- | |  | |
| Preconditions | Staff must be logged into the system. |
| |  | | --- | | **Main Flow** |  |  | | --- | |  | | 1. Staff enters username and password. 2. System authenticates the credentials. 3. Staff is redirected to their dashboard. |
| **Post conditions** | Staff is successfully logged in. |
| **Exceptions** | System error prevents login. |

* + 1. **Fully Dressed Use Case: Task Assigned to Staff by RSMS System**

The following table 3.21 shows the fully dressed use case for Task Assign:

*Table 3.21 Fully Dressed Use Case Task Assign*

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Use Case Name** | | Task Assigned to Staff by RSMS System |
| Actor | Staff |
| Description | |  | | --- | | System assigns tasks to Staff based on predefined criteria like category wise. |  |  | | --- | |  | |
| Preconditions | Staff must be logged into the system. |
| |  | | --- | | **Main Flow** |  |  | | --- | |  | | 1. SystemAdmin creates or updates a task. 2. The RSMS system assigns the task to the relevant Staff member. 3. Staff is notified via the dashboard. |
| **Post conditions** | Task is visible on the Staff's task list. |
| **Exceptions** | If no Staff is available, task assignment is delayed. |

* + 1. **Fully Dressed Use Case Update Task Status by Staff**

The following table 3.22 shows the fully dressed use case for Update Task Status:

*Table 3.22 Fully Dressed Use Case Update Task Status*

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Use Case Name** | | Update Task Status by Staff |
| Actor | Staff |
| Description | |  | | --- | | Staff updates the status of their assigned tasks (e.g., completed, in-progress). |  |  | | --- | |  | |
| Preconditions | Staff must be logged into the system. |
| |  | | --- | | **Main Flow** |  |  | | --- | |  | | 1. Staff navigates to the "My Tasks" section. 2. Selects a specific task. 3. Updates the status (e.g., completed, in-progress, pending). 4. Submits the update. 5. System logs the updated status. |
| **Post conditions** | Task status is updated and visible to SystemAdmin. |
| **Exceptions** | System error prevents status updates. |

The following table 3.23 shows the fully dressed use case for Browse and View Properties:

*Table 3.23 Fully Dressed Use Case Browser and View Properties*

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Use Case Name** | | Browse and View Properties |
| Actor | Visitor |
| Description | |  | | --- | | A visitor can browse and view property listings and contact property owners without authentication. | |
| Preconditions | Visitor has access to the RSMS system. |
| |  | | --- | | **Main Flow** |  |  | | --- | |  | | 1. Visitor opens the RSMS system. 2. Navigates to the Marketplace. 3. Views property details (e.g., images, descriptions, prices). 4. Uses the "Contact Owner" button to send a message (via a phone no. or email). |
| **Post conditions** | Visitor successfully browses properties and contacts owners. |
| **Exceptions** | System error prevents viewing or contacting. |

## 3.3. Functional Requirements

**FR1:** **Create Society and SocietyAdmin:** SuperAdmin will create and manage SystemAdmin as well as can create Societies.

**FR2: Manage Residents and Staff:** SystemAdmin shall manage Residents and Staff, including account approvals and creation and a lot of other things.

**FR3:** **Verify Listings:** SocietyAdmin shall verify property listings before publishing them.

**FR4: Report and Track Issues:** Residents shall report issues and track their resolution status.

**FR5**: **Create Property Listings:** Residentsshall create property listings and view others’ listings.

**FR6**: **Create Events:** Residentswill create events that require SystemAdmin approval.

**FR7**: **Account Customization:** Personalized settings for Residents to change credentials and others info.

**FR8**: **View Announcements:** Announcements and event details viewable by Residents in a separate interface.

**FR9**: **Browser Properties:** Visitors shall browse properties without authentication.

**FR10**: **Contact Sellers:** Visitors shall Contact properties Sellers without authentication.

## 3.4. Non-Functional Requirements

**Simplicity**  
The RSMS system shall have a clean and intuitive interface to ensure ease of use for all roles (SuperAdmin, SystemAdmin, Residents, Staff, Visitors), minimizing the need for training or external instructions.

**Usability**  
The RSMS system shall deliver a consistent and user-friendly experience across both web and mobile platforms, ensuring accessibility and providing immediate feedback to user actions.

**Security**  
The RSMS system shall prioritize user data protection through secure authentication (e.g., role-based access), encryption of sensitive data, and regular security reviews to identify and address vulnerabilities.

**Flexibility**  
The RSMS system shall be built with a modular and scalable architecture, allowing for seamless integration of third-party services, easy updates, and adaptability to new features, devices, and user preferences.

# **4. Design and Architecture**

The Residential Society Management System (RSMS) is designed as a scalable, multi-platform solution (web and mobile) to automate and streamline operations in residential communities. The architecture adopts a modular, role-based approach, ensuring seamless interaction between stakeholders (SuperAdmin, SystemAdmin, Residents, and Staff) while prioritizing real-time communication, security, and ease of use.

## 4.1. Data Representation

The diagram below shows clearly all data with types. It helps developers for better understand the system and follow up the consistent and accurate data for the features.

**ER Diagram:**

The following diagram 4.1 shows the ER Diagram of RSMS.

**A diagram of a computer

AI-generated content may be incorrect.** *Figure 4.1 shows ER Diagram*

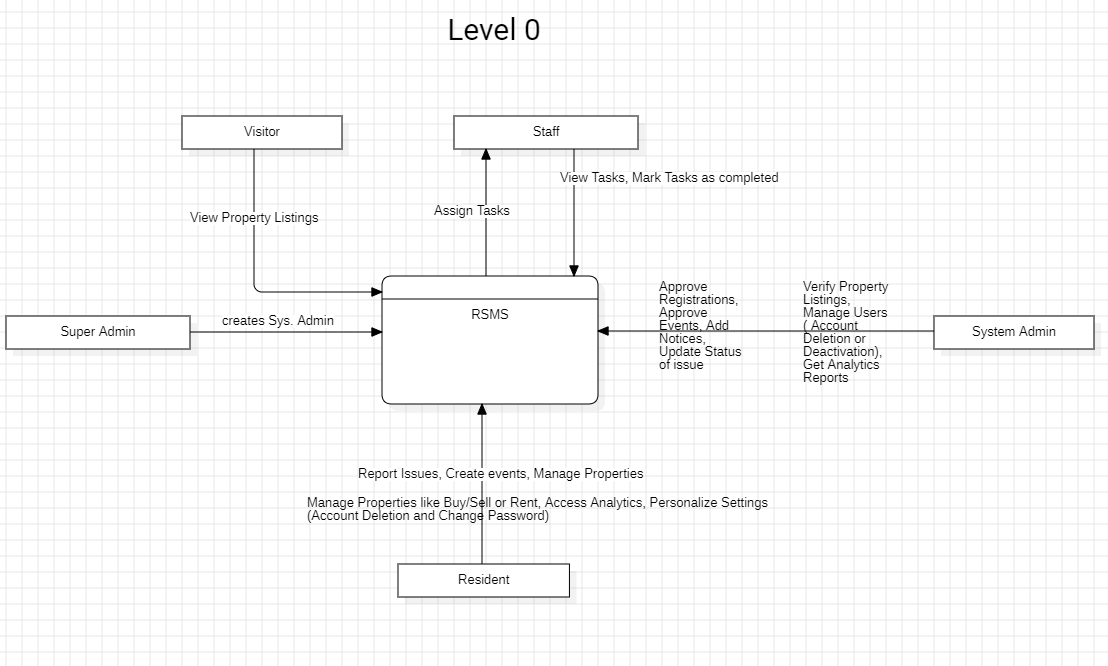


## 4.2. Process Flow/Representation

Process Flow diagrams like DFDs, or Class Diagrams help more in this case for clearance about the flow of your project features and modules, like how the process goes and how the system works under the hood.

**DFD Level 0:**

Following Figure 4.2 is the DFD level 1 Diagram:

*Figure 4.2 Level 0 DFD*

**DFD Level 1:**

Following Figure 4.3 is the DFD level 1 Diagram:

A diagram of a company

Description automatically generated

*Figure 4.3 DFD Level 1*

**DFD Level 2:**

The following Figure 4.4 is the DFD level2 User management Diagram:

A diagram of a user management system

Description automatically generated

*Figure 4.4 DFD level 2 User management Diagram*

Following Figure 4.5 is the DFD level2 Announcement management Diagram:

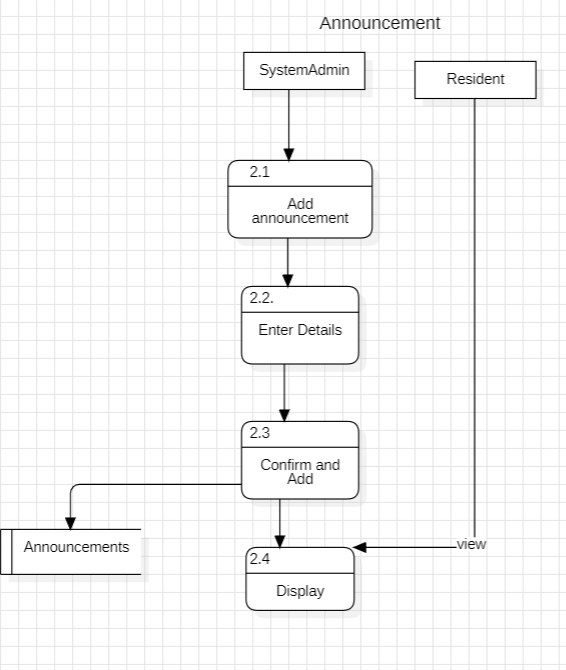
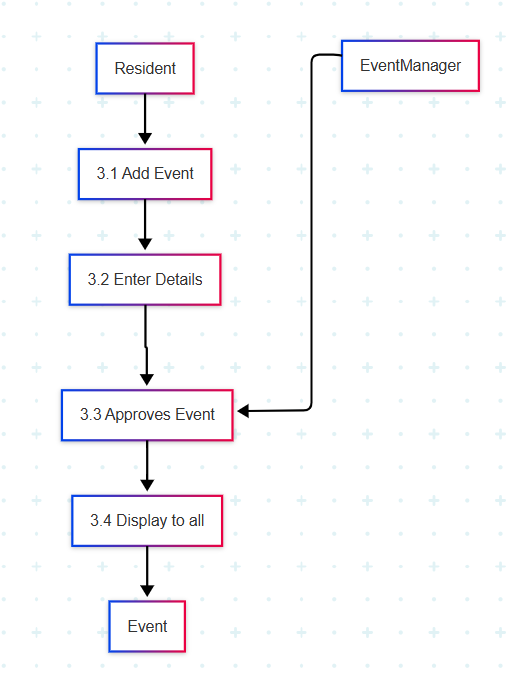


Figure 4.5 DFD Level 2 Announcement Management

Following Figure 4.6 is the DFD level2 Event management Diagram:



*Figure 4.6 DFD Level 2 Event Management Diagram*

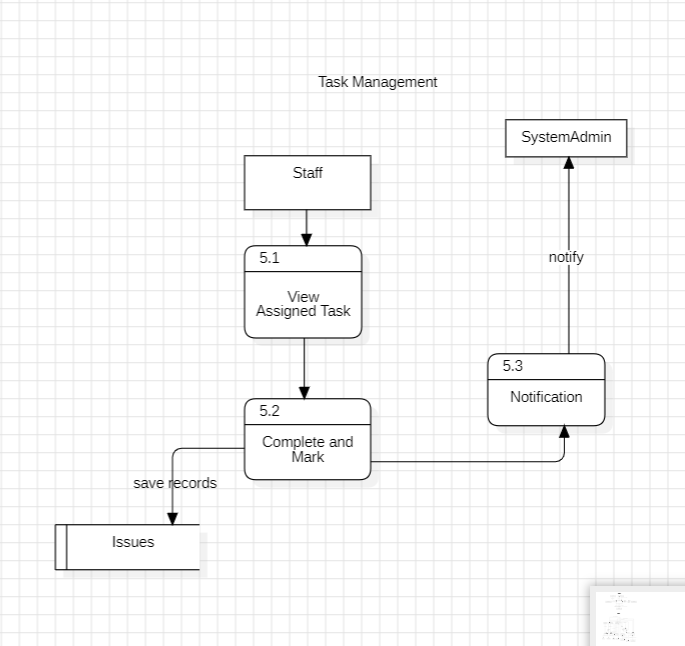
Following Figure 4.7 is the DFD level2 Issue management Diagram:

A diagram of a system

Description automatically generated

*Figure 4.7 DFD Level 2 Issue Management Diagram*

Following Figure 4.8 is the DFD level2 Task management Diagram:



*Figure 4.8 DFD Level 2 Task Management Diagram*

Following Figure 4.9 is the DFD level2 Marketplace Diagram*:*

A diagram of a market

Description automatically generated

Figure 4.9 DFD Level 2 Marketplace Diagram

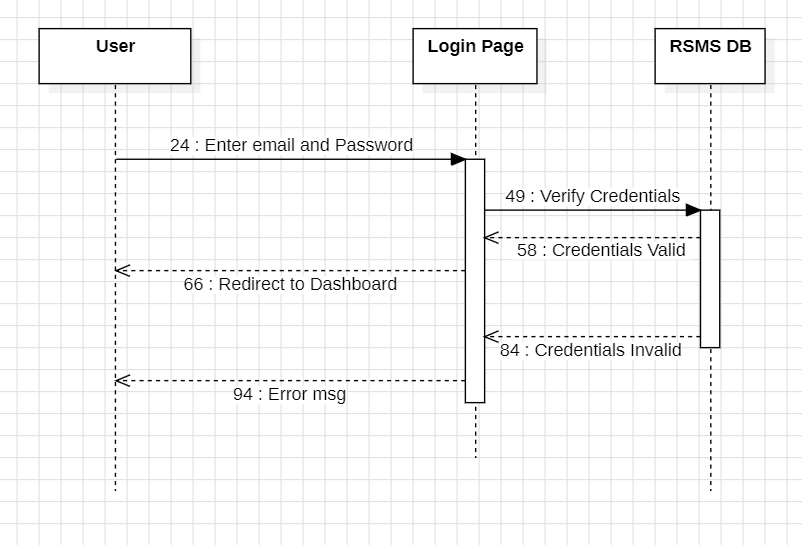
## 4.3. Design Models:

Design Models or we can say the system’s representation, that how actually start interaction with the stuff that takes part and to retrieve the data from there.

Here we define the process step by step in sequence to help developers/testers.

* + 1. **System Sequence Diagram for generic Login:**

The following figure 4.10 shows system diagram for generic login



*Figure 4.10 generic login*

* + 1. **System Sequence Diagram for SuperAdmin Login/Signup:**

The following figure 4.11 shows system diagram for SuperAdmin Login/Signup

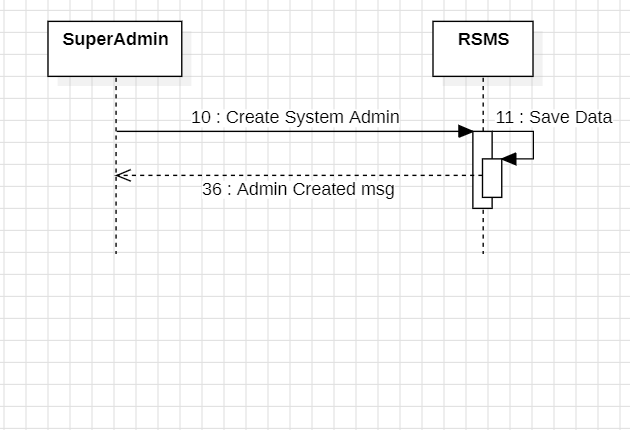
A diagram of a computer program

Description automatically generated

*Figure 4.10 System Sequence Diagram Login/Signup*

* + 1. **System Sequence Diagram create system Admin:**

The following figure 4.12 shows system diagram for creating SocietyAdmin



*Figure 4.11 System Sequence Diagram Create System Admin*

* + 1. **System Sequence Diagram Admin Add Notice:**

The following figure 4.11 shows system diagram for Admin Add Notice

A diagram of a document

AI-generated content may be incorrect.

*Figure 4.11 System Sequence Diagram Add Notice*

* + 1. **System Sequence Diagram Manage Users:**

The following figure 4.12 shows system diagram for SystemAdmin to Manage Users

A diagram of a diagram

AI-generated content may be incorrect.

*Figure 4.12 System Sequence Diagram Manager Users*

* + 1. **System Sequence Diagram Admin Update Issue Status:**

The following figure 4.13 shows system diagram for System Admin Update Issue Status

A graph with text and numbers

AI-generated content may be incorrect.

*Figure 4.13 System Sequence Diagram Update Issue Status*

* + 1. **System Sequence Diagram Admin Property Verification:**

The following figure 4.14 shows system diagram for Admin Property Verification

A diagram of a marketplace page

AI-generated content may be incorrect.

*Figure 4.14 System Sequence Diagram Property Verification*

* + 1. **System Sequence Diagram Generic Logout:**

The following figure 4.15 shows a system diagram for Generic Logout

A diagram of a company

AI-generated content may be incorrect.

*Figure 4.15 System Sequence Diagram Generic Logout*

* + 1. **System Sequence Diagram Resident Registration Approval/Rejection:**

The following figure 4.16 shows system diagram for Resident Registration Approval/Rejection:

A diagram of a project

AI-generated content may be incorrect.

*Figure 4.15 System Sequence Diagram Registration Approval/Rejection*

* + 1. **System Sequence Diagram Resident Dashboard:**

The following figure 4.16 shows a system diagram for Resident Dashboard:

A diagram of a data flow

AI-generated content may be incorrect.

*Figure 4.16 System Sequence Diagram Resident Dashboard*

* + 1. **System Sequence Diagram Resident View Announcement Interface:**

The following figure 4.17 shows a system diagram for Resident **View Announcement Interface:**

A diagram of a diagram

AI-generated content may be incorrect.

*Figure 4.17 System Sequence Diagram Login/Signup*

* + 1. **System Sequence Diagram Resident and Events:**

The following figure 4.18 shows a system diagram for Resident and Events:

A diagram of a project

AI-generated content may be incorrect.

*Figure 4.18 System Sequence Diagram Login/Signup*

* + 1. **System Sequence Diagram Resident Marketplace:**

The following figure 4.19 shows a system diagram for Resident Marketplace:

A diagram of a company

AI-generated content may be incorrect.

*Figure 4.19 System Sequence Diagram Login/Signup*

* + 1. **System Sequence Diagram Resident Personalized Setting:**

The following figure 4.20 shows system diagram for Resident Personalized Setting:

A diagram of a computer program

Description automatically generated

*Figure 4.20 System Sequence Diagram Login/Signup*

* + 1. **System Sequence Diagram Staff Task Management:**

The following figure 4.21 shows a system diagram for Staff Task Management:

A diagram of a project

Description automatically generated *Figure 4.21 System Sequence Diagram Login/Signup*

* + 1. **System Sequence Diagram Visitor Marketplace:**

The following figure 4.22 shows a system diagram for Visitor Marketplace:

A diagram of a project

Description automatically generated *Figure 4.22 System Sequence Diagram Login/Signup*

# **5. Implementation**

The implementation phase involves building the system based on the design model. The code is developed, tested, and integrated using tools and programming languages suitable for the project. This stage ensures the system functions as planned and meets the required objectives.

## 5.1. Algorithm

* + 1. **Algorithm for User Authentication:**

Please enter your username and password.

Output: Success or failure of authentication

Algorithm:

Obtain user information from the input.

Check the user credentials against the user database that has been saved.

Verify the user's identity and authorize access if the credentials match.

Access should be denied, and new credentials should be requested if the credentials do not match.

* + 1. **Algorithm for User Registration:**

Prompt the user to enter name, email, and password.

Validate the input:

Ensure the email follows a valid format.

Ensure the password meets security criteria (e.g., at least 8 characters, one special character).

Check if the email already exists in the user database.

If the email exists:

Output: "Email already registered. Please log in."

If the email does not exist:

Save the user information in the database.

Send a verification email to the user.

Output: "Registration successful. Please verify your email."

* + 1. **Algorithm for Issue Reporting:**

Input: Resident submits issue details (description, category, images).

Process:

Assigns the unique issue ID

Notifies relevant staff real time

Updates Resident Dashboard with status (“Pending”)

Output: Issue generated with tracking ID

* + 1. **Algorithm for Event Creation and Voting:**

Resident Creates an Event and do voting.

Process:

Validates Resident’s authenticity

Updates vote count on each event

One vote per Resident

Output: Updated vote displayed to admins/residents.

* + 1. **Algorithm for Property Listings:**

Input: Resident submits property details (photos, price, contact)

Process:

System flags listing as "Unverified"

Notifies SystemAdmin for review

Admin Verifies the listing i.e. Approve or Reject

Updates listing status and notifies owner via email

Output: Verified badge applied to listing’s owner

## 5.2. External APIs

Describe the APIs in table 5.1

Table 5.1 shows the external APIs that we have used in our RSMS System.

*Table 5.1: Details of APIs used in the project*

|  |  |  |  |
| --- | --- | --- | --- |
| **Name of API** | **Description of API** | **Purpose of usage** | **List down the function/class name in which it is used** |
| Google Authentication API | Provides user authentication and authorization services real time from Google Server | Used for managing user login, registration, and authentication within the Residential Society Management System | -handleGoogleLogin()  -verifyIdToken()  -GoogleOAuthProvider  -passport.authenticate()  -useGoogleLogin()  -getUserDataFromGoogle() |

## 5.3. User Interface:

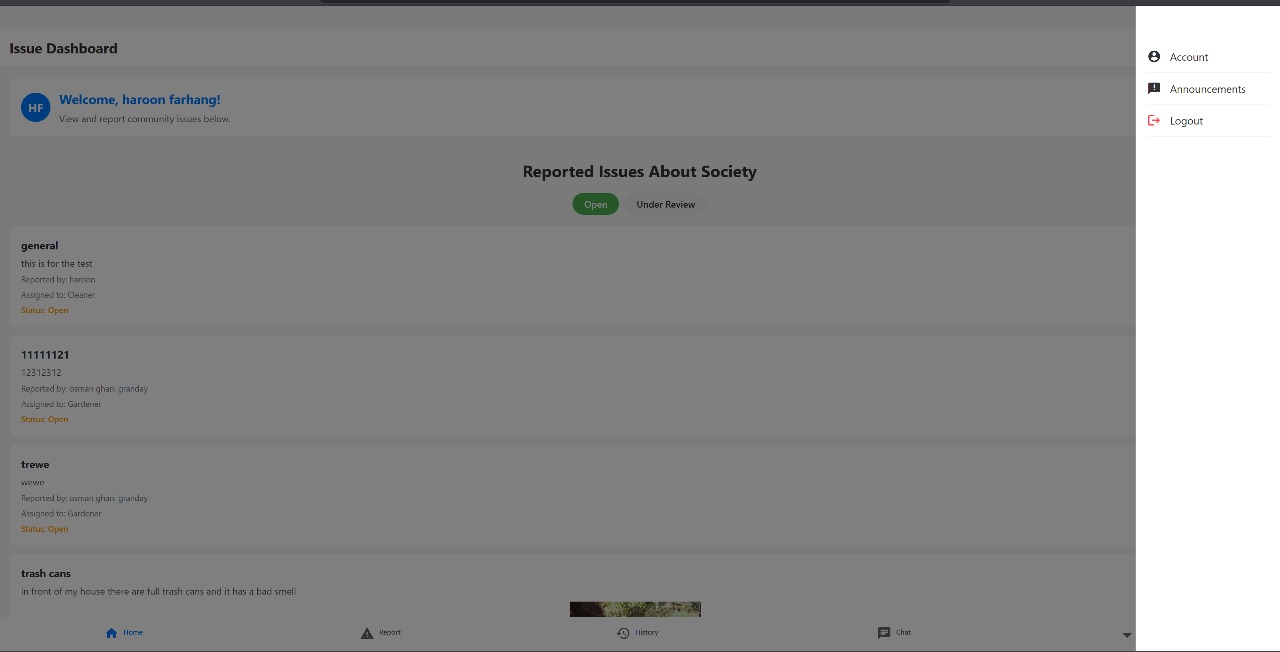
Below diagram 5.1 is the User interface of RSMS Mobile Version

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 5.1 shows Resident Login*

Below diagram 5.2 is the Resident Dashboard of RSMS Mobile Version



*Figure 5.2 Shows Resident Dashboard*

Below diagram 5.3 is the Resident Profile InterfaceA screenshot of a computer

AI-generated content may be incorrect.

*Figure 5.3 Shows Resident Profile Interface*

Below is the Figure 5.4 Resident Event Interface

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 5.4 Shows Resident Event Interface*

Below is Figure 5.5 Shows Resident Chat Box

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 5.5 Shows Resident Chat Box*

Below is Figure 5.6 Shows *Resident Resolved Issues History*

A screenshot of a car

AI-generated content may be incorrect.

*Figure 5.6 Shows Resident Resolved Issues History*

Below is Figure 5.7 Shows *Resident Report an Issue*

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 5.7 Shows Resident Report an Issue*

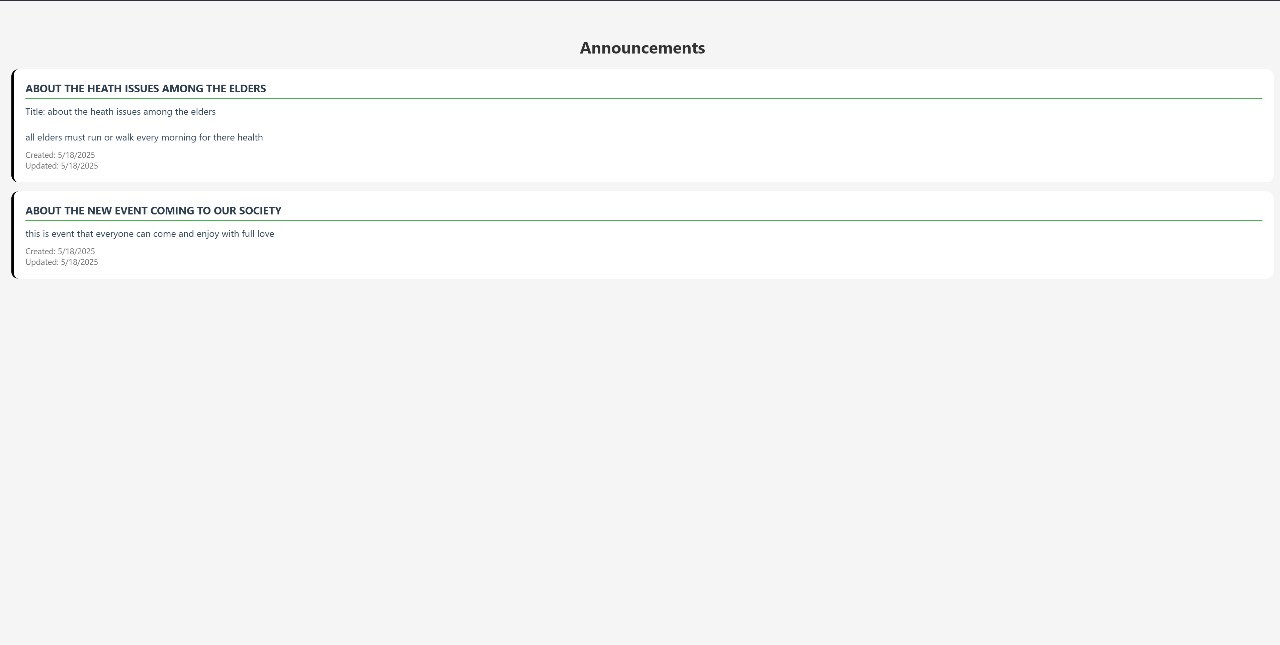
Below is Figure 5.8 Shows *Login for Mobile Screen*

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 5.8 shows Login for Mobile Screen*

Below is Figure 5.9 Shows *Announcements page*



*Figure 5.9 Shows Announcements page*

Below is Figure 5.10 Shows *Personal Issue on Mobile Screen*

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 5.10 Shows Resident Personal Issue on Mobile Screen*

Below is Figure 5.11 Shows *Staff Login on Mobile Screen*

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 5.11 Shows Staff Login on Mobile Screen*

Below is Figure 5.12 *Event Manager Interface*

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 5.12 Shows Event Manager Interface*

Below is Figure 5.13 Shows *Shows Staff Dashboard*

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 5.13 Shows Staff Dashboard*

Below is Figure 5.14 Shows *Admin (SuperAdmin) Panel/Dashboard*

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 5.14 Shows Admin (SuperAdmin) Panel/Dashboard*

Below is Figure 5.15 Shows *Admin Panel View Societies*

A screenshot of a website

AI-generated content may be incorrect.*Figure 5.15 Shows Admin Panel View Societies*

Below is Figure 5.16 Shows *SocietyAdmin Panel*

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 5.16 Shows SocietyAdmin Panel*

Below is Figure 5.17 Shows *Marketplace Interface*

A screenshot of a computer

AI-generated content may be incorrect.*Figure 5.17 Shows Marketplace Interface*

Below is Figure 5.18 Shows *Staff Allocation Interface by SocietyAdmin*

A screenshot of a computer

AI-generated content may be incorrect.*Figure 5.18 Shows Staff Allocation Interface by SocietyAdmin*

Below is Figure 5.19 Shows *Under Review Issues by SocietyAdmin*

A screenshot of a computer

AI-generated content may be incorrect.*Figure 5.19 Shows Under Review Issues by SocietyAdmin*

Below is Figure 5.20 Shows *Marketplace Listing Modal View*

A screenshot of a computer

AI-generated content may be incorrect.*Figure 5.20 Shows Marketplace Listing Modal View*

Below is Figure 5.21 Shows *Marketplace Add Property Listing by SocietyAdmin*

A screenshot of a computer

AI-generated content may be incorrect.*Figure 5.21 Shows Marketplace Add Property Listing by SocietyAdmin*

Below is Figure 5.22 Shows *Open Issues by SocietyAdmin*

A screenshot of a computer

AI-generated content may be incorrect.*Figure 5.22 shows Open Issues by SocietyAdmin*

Below is Figure 5.23 Shows *Staff Management Interface by SocietyAdmin*

A screenshot of a computer

AI-generated content may be incorrect.*Figure 5.23 Shows Staff Management Interface by SocietyAdmin*

Below is Figure 5.24 Shows *Resolved Issues by SocietyAdmin* A screenshot of a computer

AI-generated content may be incorrect.*Figure 5.24 shows Resolved Issues by SocietyAdmin*

Below is Figure 5.25 Shows *Announcements by SocietyAdmin*

A screenshot of a computer

AI-generated content may be incorrect.*Figure 5.25 shows Announcements by SocietyAdmin*

Below is Figure 5.26 Shows *Resident’s Registration Requests on SocietyAdmin Dashboard*

A screenshot of a computer

AI-generated content may be incorrect.*Figure 5.26 shows Resident’s Registration Requests on SocietyAdmin Dashboard*

Below is Figure 5.27 shows *Housing Entry by SocietyAdmin*

A screenshot of a computer

AI-generated content may be incorrect.*Figure 5.27 shows Housing Entry by SocietyAdmin*

# **6. Testing and Evaluation:**

This phase ensures the system works correctly and meets user requirements. Various testing methods, such as unit, integration, and system testing, are performed to identify and fix issues. Evaluation measures system performance, reliability, and usability to confirm it aligns with project goals

## 6.1. Manual Testing

Manual testing is done to check for defects in the system. The end user will play the role of tester where every feature of the system will be checked and ensure correct behavior.

### 6.1.1. System Testing

After developing the system successfully, the system must be tested to ensure that the system is giving the intended results. The testing will help in finding errors that can be hidden from the user. The sequence in which testing will be done is unit testing, functional testing and integration testing. Testing is must before deployment for the user consumption.

### 6.1.2. Unit Testing

Unit testing involves testing individual components or functions of the system to ensure they work as expected. Each unit of code is tested in isolation to identify and fix bugs early in the development process.

**Unit Testing 1:** Login as SuperAdmin

**Testing Objective:** To ensure the login form works correctly.

The following table 6.1 shows the UTC for Login as SuperAdmin

Table 6.1: Login Unit Test case

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| 1. | Verify login after clicking on the ‘Login’ button on login form with correct input data | Email:  dkdanyal123@gmail.com  Password:  danyal123@ | Successfully logged into the main page of the RSMS by the name Muhammad Danyal | Pass |
| 2. | Verify login after clicking on the ‘Login’ button on login form with incorrect input data | Email:  danyal@gmail  Password:  danyal123@ | Incorrect Credential | Pass |

**Unit Testing 2:** Login as System Manager

**Testing Objective:** To ensure the login form is working correctly

The following table 6.2 shows the UTC for Login as System Manager

Table 6.2: Login Unit Test case

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| 1. | Verify login after clicking on the ‘Login’ button on login form with correct input data | Email:  osmanghani143@gmail.com  Password:  Osman12@ | Successfully login into the admin interface by the name of Osman Ghani. | Pass |
| 2. | Verify login after clicking on the ‘Login’ button on login form with incorrect input data | Email:  osmanghani143  Password:  Osman12@ | Incorrect Credential | Pass |

**Unit Testing 3:** Login as Resident

**Testing Objective:** To ensure the login form works correctly.

The following table 6.3 shows the UTC for Login as Resident

Table 6.3: Login Unit Test case

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| 1. | Verify login after clicking on the ‘Login’ button on login form with correct input data | Email:  Abdullah00@gmail.com  Password:  abddd123@ | Successfully logged into the main page of the RSMS by the name Abdullah Ludin | Pass |
| 2. | Verify login after clicking on the ‘Login’ button on login form with incorrect input data | Email:  Abdullah00  Password:  abddd123@ | Incorrect Credential | Pass |

**Unit Testing 4:** Login as Staff

**Testing Objective:** To ensure the login form works correctly.

The following table 6.4 shows the UTC for Login as Staff

*Table 6.4: Login Unit Test case*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| 1. | Verify login after clicking on the ‘Login’ button on login form with correct input data | Phone No:  +92-3159826555  Password:  Sample123@ | Successfully logged into the Staff Interface of the RSMS by the name SampleStaff | Pass |
| 2. | Verify login after clicking on the ‘Login’ button on login form with incorrect input data | Phone No:  +92-31598265556  Password:  Sample123@ | Incorrect Credential | Pass |

* **Unit Testing 5:** Register as Admin and Resident

**Testing Objective:** To ensure the Register form is working correctly

The following table 6.5 shows the UTC for Register as Admin and Resident

Table 6.5: Register Unit Test case

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| 1. | Verify Register after click on the ‘Register’ button on Register form with correct input data. | Username:  Muhammad Danyal  Email:  dkdanyal123@gmail.com  Password:  danyal123@ | A verification email has been sent to the user to confirm their identity. | Pass |
| 2. | Verify Register after clicking on the ‘Register’ button on Register form with incorrect input data | Username:  Muhammad Danyal  Email:  dkdanyal123  Password:  danyal12@ | Incorrect Credential | Pass |

**Unit Testing 6:** Report an Issue by Resident

**Testing Objective:** To ensure the Issue Report form works correctly.

The following table 6.6 shows the UTC for Reporting an Issue by Resident

Table 6.6: Issue Report Unit Test case

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| 1. | To verify that the issue reported successfully by resident | Category: Maintenance Description: "Elevator not working" Images: [elevator.jpg] | Issue created with "Pending" status Confirmation message displayed | Pass |
| 2. | Resident submits issue with empty description | Category: Security Description: (empty) Images: [gate.jpg] | "Description required" error message | Pass |

**Unit Testing 7:** Listing a Property by Resident on Marketplace

**Testing Objective:** To ensure the Property listed on the marketplace works correctly.

The following table 6.7 shows the UTC for Listing Property by Resident on Marketplace

Table 6.7: Property Listing Unit Test case

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| 1. | Resident lists property with proper details | Type: Rent Price: $500/month Photos: [3 images] Contact: +123456789 | Listing saved as "Pending Verification" Owner receives confirmation SMS | Pass |
| 2. | Visitor browses the marketplace without login | Search: "ABC under $600" | Listings displayed with "Contact Owner" button No edit/delete options visible | Pass |

**Unit Testing 8:** Creating Event on RSMS

**Testing Objective:** To ensure the Event Created on RSMS Event Interface works correctly.

The following table 6. shows the UTC for Event Management

Table 6.8: Event Management Unit Test case

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| 1. | Residents create valid events with proper details | Title: "Annual BBQ" Date: 2024-12-25 Description: "Community gathering | Event appears in calendar with Voting enabled | Pass |
| 2. | Resident creates event with past date | Title: "Meeting" Date: 2023-01-01 | "Future date required" error message | Pass |
| 3. | Resident upvotes event | Event ID: #EVT-101 Vote: upvote | Vote count increments by 1 Reminder SMS scheduled |  |

### 6.1.3. Functional Testing

The functional testing will take place after the unit testing. In this functional testing, the functionality of each of the modules is tested. This is to ensure that the system produced meets the specifications and requirements

* **Functional Testing 1:** Login with Different Roles
* **Objective**:To ensure that the correct page with the correct navigation bar is loaded.

The following table 6.9 shows the FTC for Login with Different Roles

*Table 6.9: Login Functional Test case*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and value** | **Expected Result** | **Result** |
| 1 | Login as a Resident | Username: [dkdanyal123@gmail.com](mailto:dkdanyal123@gmail.com)  Password: danyal123@ | Main page for the Resident is loaded with the Resident navigation bar. | Pass |
| 2 | Login as an Admin | Username: [Osmanghani33@gmail.com](mailto:Osmanghani33@gmail.com)  Password: OsmanGhani12@ | Main page for the admin is loaded with the admin navigation bar. | Pass |
| 3 | Login with incorrect credentials | Username: dkdanyal123  Password: danyal123@ | Incorrect credentials error message is displayed. | Pass |
| 4 | Login with another incorrect set | Username: osmanghani  Password: OsmanGhani12@ | Login is unsuccessful; appropriate error messages are displayed. | Pass |

* **Functional Testing 2:** Registration for Residents and System Managers
* **Objective:** To ensure that users can successfully register and receive verification emails.

The following table 6.10 shows the FTC for Registration

*Table 6.10: Registration Functional Testcase*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and value** | **Expected Result** | **Result** |
| 1 | Register as a Resident | Username: Muhammad Danyal  Email: [dkdanyal123@gmail.com](mailto:dkdanyal123@gmail.com)  Password: danyal123@ | Verification email sent to the user; account creation successful. | Pass |
| 2 | Register with invalid email | Username: Muhammad Danyal  Email: dkdanyal111@gmail.com  Password: danyal123@ | Registration unsuccessful; invalid email error message displayed. | Pass |

* **Functional Testing 3:** Issue Reporting to confirm the system’s behavior
* **Objective:** To ensure that users can successfully report the issues to RSMS

The following table 6.11 shows the FTC for Issue Reporting

*Table 6.11: Issue Reporting Functional Testcase*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and value** | **Expected Result** | **Result** |
| 1 | Resident reports maintenance issue | Category: Maintenance Description: "Parking gate broken" Images: [gate.jpg] | Issue appears in staff dashboard Resident sees "Reported" status | Pass |
| 2 | Resident checks issue status | Issue ID: #RSMS-305 | Status ("In Progress") and staff mark as done | Pass |

* **Functional Testing 4:** Listing Properties by Residents and Visitors
* **Objective:** To ensure that users can successfully list properties on RSMS

The following table 6.12 shows the FTC for Property Listing

*Table 6.12: Property Listing Functional Testcase*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and value** | **Expected Result** | **Result** |
| 1 | Resident lists property for rent | Type: Rent Price: 70,000/month Photos: [5 images] Contact: +923001234567 | Listing appears in marketplace with "Pending Verification" tag | Pass |
| 2 | Visitor contacts property owner | Clicks "WhatsApp Owner" on verified listing | Redirected to WhatsApp for further talk | Pass |

* **Functional Testing 5:** Event Creation by Resident
* **Objective:** To ensure that users can successfully create community events on RSMS

The following table 6.13 shows the FTC for Event Creation by Resident

*Table 6.13: Community Events Functional Testcase*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Attribute and value** | **Expected Result** | **Result** |
| 1 | Resident creates community event | Title: "Eid Dinner" Date: 2024-06-17 Description: "Potluck gathering" | Event appears in society Event Interface and Voting buttons (👍/👎) enabled | Pass |
| 2 | Resident receives event reminder | Voted "👍" on #EVT-205 | Email received 24h before event: "Reminder: Eid Dinner tomorrow at 8 PM" | Pass |

## 6.2. Integration Testing

After Unit and Functional testing, it’s time to combine components of our application and test together to identify defects in their interaction.

The following table 6.14 shows the ITC for all Use Cases

*Table 6.14: Integration Test case*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case/ Test Script** | **Attribute and value** | **Expected Result** | **Result** |
| 1 | Login as Manager and manage Residents | Email: [dkdanyal123@gmail.com](mailto:dkdanyal123@gmail.com)  Password: danyal123@ | Successful login, access to Manager dashboard, and successful navigation to the Manager Panel. | Pass |
| 2 | Login as Resident and report issues and create events | Email: [abdullahLudin122@gmail.com](mailto:abdullahLudin122@gmail.com)  Password: abddd12@ | Successful login, access to Resident Interface, issue is successfully posted and event successfully created | Pass |
| 3 | Register as a new Resident and verify login | Username: Osman Ghani  Email: [Osmanghani@gmail.com](mailto:Osmanghani@gmail.com)  Password: Osman123@ | Registration successfully, verification email sent, and login is successful. | Pass |
| 4 | Resident reports issue and staff resolves it | Issue: "Broken elevator" Photos: 2 images | Issue appears in staff dashboard, status updates to "Resolved" when completed | Pass |
| 5 | Resident creates event and others vote | Event: "Annual BBQ" Votes: 15 upvotes, 2 downvotes | Event appears in community calendar, vote count updates in real-time | Pass |
| 6 | Resident lists property and admin verifies | Property: ABC Apartment Price: 20,000 pkr | Listing shows "Pending" status initially, changes to "Verified" after admin approval | Pass |
| 7 | Staff updates task status | Task: "Fix gate" Status: In Progress → Completed | Resident who reported receives notification, status updates in real-time | Pass |
| 8 | The Manager review the issue and update its status | System Admin (Manager) checks and updates the status based on issue state that either resolved or pending!! | Issue status gets updates as “resolved”, “under review”. | Pass |
| 9 | SuperAdmin creates new society and register the admin within that society | Society: "Green Valley" Admin: [newadmin@gmail.com](mailto:newadmin@gmail.com) | New society appears in system, assigned admin/manager receives credentials | Pass |
| 10 | Marketplace transaction | Buyer contacts seller via WhatsApp | Chat initiates successfully | Pass |

# **7. Conclusion and Future Work**

Below is the conclusion and future work of RSMS after all development and technical research.

## 7.1. Conclusion

The Residential Society Management System (RSMS) successfully meets all the requirements we set at the beginning of the project. The system provides an easy-to-use platform for residents, staff, and administrators to manage society operations efficiently.

Key features like issue reporting, event management, property listings, and announcements were carefully tested to ensure they work smoothly. The system keeps data secure and only allows authorized users to access different functions.

RSMS helps residents report problems quickly, organize community events, and provides a safe platform for property deals. It also improves communication between residents and management through announcements and messaging.

Overall, RSMS achieves its goal of making residential society management simpler and more organized for everyone involved. The system is the great contribution to the future of Residential Societies at the country level.

## 7.2. Future Work

To make the RSMS even better, we plan to add these features in the future:

1. **Mobile Payment System**

Let residents pay maintenance fees and bills through the app

Send automatic payment reminders

2. **Smart Home Connections**

Link with smart doorbells and security cameras

Show parking space availability in real-time

3**. Help Chatbot**

Answer common questions about society rules

Guide users how to report issues

4. **Guest Management**

Pre-approve visitors through the app

Generate temporary access codes

5. **Community Surveys**

Take votes on important decisions

Share results with all residents

These improvements will make society life more convenient and secure for everyone.

1. **References**

* How to design using UML (OOP): For guidance please follow the instructions mentioned in the link: http://agilemodeling.com/artifacts/
* How and when to design ER diagrams: For guidance please follow the instructions mentioned in the link:

<http://people.inf.elte.hu/nikovits/DB2/Ullman_The_Complete_Book.pdf>

* Data flow diagrams: For guidance please follow the instructions mentioned in the link and book:
  + http://www.agilemodeling.com/artifacts/dataFlowDiagram.htm
  + Software Engineering –A Practitioner’s approach by Roger Pressman
* Architecture diagram: For guidance, please follow the instructions mentioned in the link and book:
  + Ian Sommerville – Software Engineering 9th Edition– Chapter 6

