Real Estate Management System (REMS)

Presented By

Sienna Markham, 220274536
Mahjabin Mollah, 220416038
Eamon Ryan, 219447192

Team 3

November 23rd, 2025

Table of Contents

1. Introduction

The field of software engineering increasingly relies on structured methodologies, disciplined design practices, and iterative development cycles to deliver reliable and scalable systems. Modern organizations depend on comprehensive software solutions to automate processes, improve operational efficiency, and support decision-making. Real estate management, in particular, requires sophisticated digital systems capable of tracking property data, supporting client interactions, handling transactions securely, and ensuring lifecycle visibility across large portfolios.

The Real Estate Management System (REMS) project provides an opportunity to apply the full Software Development Lifecycle (SDLC) to a real-world problem domain. By combining Agile iterative development with formal specification and design artifacts, this project replicates modern industry practices and reinforces the practical skills of requirement analysis, architectural modelling, implementation, and testing.

1.1 Motivation

Current Situation

Large real estate development companies manage numerous commercial properties, often containing hundreds of retail units across multiple mall locations. Without an integrated software system, tasks such as tracking rental units, scheduling viewings, collecting rent, reviewing lease agreements, and responding to maintenance requests become error-prone, fragmented, and inefficient. Manual or semi-digital processes frequently result in double-booked appointments, delayed reporting, missing invoices, and inconsistent record-keeping.

Existing Solutions

While generic property management platforms exist, they rarely provide the level of customization required for commercial mall management, including complex rental cycles, multi-unit discounts, utility billing, digital lease renewals, and automated emergency-level maintenance escalation. Existing systems also may not integrate with the company's workflow, lack tailored modules, or fail to support simultaneous Agile and documentation-heavy development processes required in regulated environments.

Why this Project is Needed

This project aims to bridge these shortcomings by developing a robust, end-to-end software solution specifically tailored for large-scale commercial real estate operations. Creating a custom REMS allows complete alignment with business rules while providing students hands-on experience applying SDLC principles.

1.2 Project Description

The Real Estate Management System (REMS) is a Java-based software solution designed to automate and streamline commercial property management. The system will support:

Main Requirements

- **Store Unit Management:**
  - Track store size, location, rental tier, classification, and tenant purpose
- **Search & Availability:**
  - Allow tenants to filter and search for rentable units
- **Appointment Scheduling:**
  - Enable tenants to book viewing appointments while preventing double-bookings
- **Rental Applications & Digital Leases:**
  - Electronic submissions, approval workflows, and lease renewal logic
- **Automated Rent Collection:**
  - Support multiple billing cycles (monthly/quarterly/annual), discounts, invoices, and overdue notifications
- **Utility Billing:**
  - Combine electricity, water, and waste charges into unified monthly statements
- **Maintenance Portal:**
  - Log, prioritize, and escalate maintenance requests; apply charges for misuse
- **Administrative Tools:**
  - Reporting, data entry, management dashboards (future enhancement)

Benefits of the System

- Reduces administrative workload and manual errors
- Improves transparency for tenants and leasing agents
- Offers predictable and automated billing workflows
- Enables better planning through structured data and reporting
- Provides a modern digital experience for a traditionally paperwork-heavy industry
- Scales across multiple mall properties

1.3 Organization of the Report

This report is organized into seven main sections.

- **Section 1** introduces the project, including its motivation, description, and overall structure
- **Section 2** outlines the problem statement, the proposed solution, the design objectives, and the project's limitations
- **Section 3** presents the system specification, detailing all functional, nonfunctional, and optional pseudo requirements
- **Section 4** describes the system's use cases, including the use case diagram and corresponding descriptions
- **Section 5** provides the user stories associated with each major system feature

- **Section 6** explains the project management plan, including the Gantt chart, backlogs, and team meeting logs
- **Section 7** presents the testing and performance analysis plan for validating the system

A references section is included at the end of the report.

2. Problem Statement and Design Objectives

2.1 Problem Statement and Proposed Solution

Problem Statement

The leasing division of a commercial real estate development company currently lacks an integrated system to manage retail units, schedule appointments, handle rental applications, collect rent, track utility consumption, and coordinate maintenance operations across multiple mall properties. Fragmented tools and manual processes cause inefficiencies, double-bookings, delayed payments, inconsistent maintenance handling, and a poor tenant experience.

What the Project Aims to Do

This project aims to design and implement a full-featured Real Estate Management System that centralizes all property management workflows into a unified platform. The system will support both staff-facing and tenant-facing interfaces, ensuring accuracy, automation, and scalability.

Main Solution

The proposed solution is a Java-based software system developed using Agile methods and formal SDLC documentation. It will integrate modules for unit management, appointment scheduling, rental processing, automated rent and utility billing, and maintenance request handling. The solution will be supported by a relational or non-relational database, UML-based architectural modeling, and a full suite of tests.

2.2 Design Objectives

Objective 1 - Requirements Documentation

Due Date: November 23, 2025 (Deliverable 1)

Deliverables:

- Complete Software Requirements Specification (SRS)
- Full list of Functional Requirements (e.g., unit management, rent billing, scheduling)

- Full list of Non-Functional Requirements (e.g., reliability, maintainability, usability)
- Detailed User Stories and Acceptance Criteria
- Use Case List, Use Case Descriptions, and Use Case Diagram

Measurement Criteria:

- All requirements approved by team and instructor
- Requirements traceable to system features

Purpose:
Establishes a precise, shared understanding of what the system must do.

Objective 2 - System Architecture & Design Models

Due Date: February 1, 2026 (Deliverable 2)

Deliverables:

- Complete Software Design Document (SDD)
- Defined System Architecture (Layered or MVC)
- UML Class Diagram showing major classes and relationships
- UML Sequence Diagrams for core workflows (e.g., renewal, booking, payment)
- UML Activity Diagrams showing process flows
- Full Database Schema / ERD

Measurement Criteria:

- Diagrams accurately reflect requirements
- Architecture supports scalability, maintainability, and required performance

Purpose:
Provides a blueprint for implementation and ensures the design supports all REMS features.

Objective 3 - Core Feature Implementation

Due Date: March 22, 2026 (Deliverable 3 – Implementation)

Functional Modules to Deliver:

- Store Unit Management (CRUD, classification, rental tiering)
- Appointment Scheduling with double-booking prevention
- Rental Application & Digital Lease Module
- Automated Rent Billing (cycles, invoices, discounts)
- Utility Billing (water, electricity, waste consolidation)
- Maintenance Portal with prioritization & escalation logic

Measurement Criteria:

- All modules implemented in Java
- All modules integrated with the database
- All modules verified against the SRS

Purpose:
Implements the tangible, functional components that define the system.

Objective 4 - Testing & Quality Assurance

Due Date: March 22, 2026 (Deliverable 3 – Testing)

Deliverables:

- Complete Test Plan (unit, integration, acceptance tests)
- JUnit Test Suites with measurable coverage
- Test Report documenting results, defects, and verification of each requirement
- Evidence of continuous integration and code quality checks

Measurement Criteria:

- Meets code coverage expectations (set by instructor/team)
- All tests map to functional + non-functional requirements

Purpose:
Ensures the system meets reliability, usability, and performance expectations.

Objective 5 - Project Management & Documentation

Due Date: Throughout the Project

Deliverables:

- Fully maintained GitHub/GitLab Repository
  - Branching strategy, commits, pull requests, code reviews
- Sprint Backlogs, Burndown Charts, Retrospectives
- Final documentation package delivered by **March 22, 2026**:
  - Project Management Plan
  - SRS
  - SDD
  - Test Plan
  - Test Report
  - Implementation Notes

Measurement Criteria:

- Repository is up-to-date and reflects real collaborative work
- All documentation meets SDLC standards

Purpose:
Guarantees professional-level workflow and traceability across all project phases.

Objective 6 - Enhancement Sprint & Portfolio Update

Due Date: End of Summer 2026

Deliverables:

- Enhanced Version of REMS (bug fixes, optimizations, minor feature additions)
- Completed E-Portfolio documenting architecture, implementation, testing, and learning outcomes

Measurement Criteria:

- Enhancements successfully integrated and tested
- Portfolio meets program requirements

Purpose:
Demonstrates continuous improvement and professional reflection.

2.3 Limitations

Due to time, scope, and resource constraints, the project will not include:

- Full production-level security hardening (only basic authentication/authorization)
- Integration with real payment processors (simulated or mocked payments only)
- Real hardware-based utility metering (only manual or simulated data inputs)
- Multi-language localization or internationalization

These limitations ensure the project remains achievable within the course schedule while still meeting core learning objectives.

3. System Specification

3.1 Functional Requirements

List the core functional requirements that the system must fulfill, linking them to user needs and stories.

Property & Store Unit Management

- **FR1.** The system shall allow authorized staff to add, update, view, and delete store unit records.
- **FR2.** The system shall store and manage details for each store unit, including location, size, rental rate, classification tier, intended business purpose, and availability.
- **FR3.** The system shall support management of multiple mall properties, each with its own inventory of store units.

Tenant Store Search

- **FR4.** The system shall allow potential tenants to search for available store units using filters such as location, size, rental rate, classification tier, business purpose, and availability.
- **FR5.** The system shall display search results with standardized store information and contact or scheduling options.

Appointment Scheduling

- **FR6.** The system shall allow tenants to schedule viewing appointments with a leasing agent based on agent availability.
- **FR7.** The system shall prevent double-bookings by ensuring no overlapping appointments for the same leasing agent or store unit.
- **FR8.** The system shall send automated notifications confirming appointments and any subsequent updates.

Rental Applications & Lease Management

- **FR9.** The system shall allow tenants to submit digital rental applications, including file uploads for required documents.
- **FR10.** The system shall automatically generate digital lease agreements based on property and rental information.
- **FR11.** The system shall support electronic signing of lease agreements.
- **FR12.** The system shall track lease start and end dates for each store unit.
- **FR13.** The system shall automatically process lease renewals according to configurable company policies.

Rent Collection & Billing

- **FR14.** The system shall allow tenants to choose a payment cycle, including monthly, quarterly, bi-annual, or annual options and calculate rental charges dynamically based on the selected payment cycle.

- **FR15.** The system shall apply discount logic for tenants leasing multiple store units.
- **FR16.** The system shall generate invoices for rent, utilities, and applicable fees.
- **FR17.** The system shall track tenant payments and send notifications for overdue payments to tenants and alerts to administrative staff.

Utility Billing

- **FR18.** The system shall allow authorized staff to input manually or import utility consumption data for electricity, water, and waste management.
- **FR19.** The system shall automatically consolidate rent and utility charges into a unified monthly bill.

Maintenance Requests

- **FR20.** The system shall allow tenants to submit maintenance requests through an online portal.
- **FR21.** The system shall categorize maintenance requests by priority, automatically escalating emergency requests.
- **FR22.** The system shall apply charges to tenant accounts for misuse-related maintenance requests.

User Access & Roles

- **FR23.** The system shall support user authentication for tenants, leasing agents, and administrative staff.
- **FR24.** The system shall enforce role-based access control, restricting sensitive operations to authorized users.

3.2 Nonfunctional Requirements

1. **Performance:**
   a. **NFR1.** The system shall process search queries and return available unit results within 2 seconds under normal load.
   b. **NFR2.** The system shall maintain efficient data retrieval for large property portfolios (25+ malls, hundreds of units).
2. **Quality:**
   a. **NFR3.** The system shall maintain 99% uptime, excluding scheduled maintenance.
   b. **NFR4.** The system shall ensure that no lease, appointment, or billing transaction is partially completed (atomicity).
3. **Safety and Security:**
   a. **NFR5.** The system shall validate user input to prevent malformed or invalid data from corrupting stored information.
   b. **NFR6.** The system shall provide confirmation dialogs for critical actions (e.g., deleting property records or terminating leases).
4. **Interface:**
   a. **NFR7.** The system shall present a user-friendly, responsive, and mobile-friendly interface for tenants and staff.
   b. **NFR8.** The system shall be compatible with standard web browsers (Chrome, Firefox, Edge, Safari).

c. **NFR9.** The system shall support integration with external data sources for utility consumption (simulated or API-based).

3.3 Pseudo Requirements (Optional)

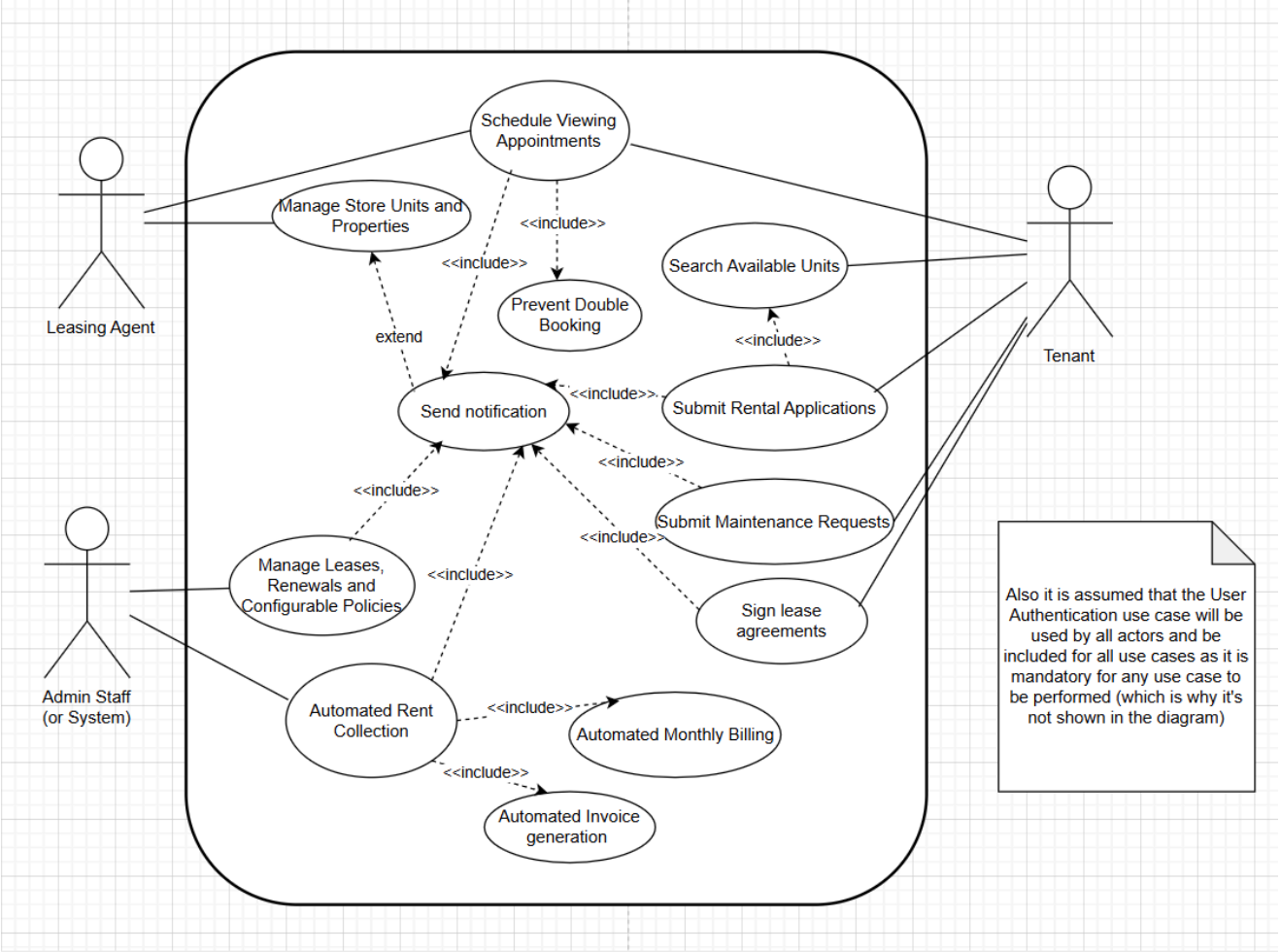
1. **C1.** The system must use a relational database (e.g., MySQL or PostgreSQL)
2. **C2.** The system must be compatible with legacy property data exported from an old record-keeping system.
3. **C3.** The system shall be implemented using Java
4. **C4.** The GUI must be built using a framework that is compatible with Java

4. Use Cases

Use cases provide a structured approach to understanding the interactions between users and the system.

4.1 Use Case Diagram

Include a UML use case diagram that shows the actors (e.g., Admin, Regular User) and the main use cases (e.g., Manage Inventory, Generate Reports, User Authentication).



4.2 Use Case Descriptions

Detail 4 key use cases with the following structure:

1. **Use Case Name**: Manage Store Units and Properties
   Actors: Leasing Agent
   Description: This use case describes how leasing agents add, update, or remove store unit information (availability, pricing, size, amenities) and manage property details within the system.
   Preconditions: Leasing Agent must be authenticated.
   Main Flow:

- The Leasing Agent selects “Manage Store Units and Properties” from the menu.
- The system displays all current units and properties.
- The agent chooses to add, edit, or remove a unit/property.
- The agent enters or modifies required information.
- The system validates the input.
- The system saves the changes and updates the property database.
- The system triggers the Send Notification use case (included) to inform tenants or staff when relevant changes occur.

Postconditions:

- Unit or property data is updated and visible to relevant users.
- Notifications are sent if applicable.

2. **Use Case Name**: Schedule Viewing Appointments
   Actors: Leasing Agent, Tenant
   Description: This use case allows tenants or leasing agents to request, schedule, or update property viewing appointments while preventing double-bookings.
   Preconditions:
   - Leasing Agent and Tenant must be authenticated.
   - Units being scheduled to view are available

   Main Flow:
   - Tenant selects “Schedule Viewing Appointments.”
   - The system shows available dates/times based on unit and agent availability.
   - Tenant selects a date and time.
   - The system performs Prevent Double Booking (included).
   - The system confirms the appointment and saves the booking.
   - The system triggers Send Notification (included) to confirm the appointment to all parties.

   Postconditions:
   - A viewing appointment is saved in the system.
   - Notifications are sent to the tenant and leasing agent.
   - Time slot becomes unavailable to others.

3. **Use Case Name**: Automated Rent Collection
   Actors: Admin Staff (System as Internal Actor)
   Description:This use case automatically processes recurring rent payments according to each tenant’s lease terms.
   Preconditions:
   - System must have valid tenant lease and billing information.
   - Admin Staff must be authenticated to view or configure settings.

   Main Flow:
   - The system identifies all tenants with rent due.
   - The system calculates the amount owed based on lease terms.
   - The system triggers Automated Monthly Billing (included).
   - The system attempts to collect payment automatically.
   - If successful, the system updates the rent balance.
   - The system triggers Automated Invoice Generation (included) to create a receipt.
   - The system triggers Send Notification (included) to update the tenant and staff on payment status.

Postconditions:
- Rent is collected or marked as failed.
- Notifications and invoices are generated.

4. **Use Case Name**: Submit Rental Applications
   Actors: Tenant
   Description: This use case allows tenants to fill out and submit rental applications for available units.
   Preconditions:
   - Tenant must be authenticated.
   - Unit information that tenant searched for must be available in the system

   Main Flow:
   - The tenant selects "Submit Rental Applications."
   - The tenant searches for a unit (included)
   - The system displays a list of available units.
   - The tenant selects a unit and fills in application details.
   - The system validates the submitted application.
   - The system saves the application.
   - The system triggers Send Notification (included) to inform staff that a new application is received.

   Postconditions:
   - Application is saved and marked as "Submitted."
   - Notifications are sent to leasing/administrative staff.

5. **Use Case Name**: Manage Leases, Renewals and Configurable Policies
   Actors: Admin Staff (System as internal actor)
   Description: This use case allows administrative staff to create new leases, update existing lease terms, process renewals, and manage configurable policy settings (late fees, deposit rules, renewal options).
   Preconditions:
   - Admin Staff must be authenticated.
   - Property/unit records must exist in the system.

   Main Flow:
   - The Admin Staff selects "Manage Leases, Renewals and Configurable Policies."
   - The system displays all existing leases and policy configurations.
   - The Admin chooses to create a new lease, edit an existing lease, process a renewal, or update policies.
   - The Admin enters or edits the required information.
   - The system validates lease parameters and policy rules.
   - The system saves the updated or new lease/policy information.
   - The system triggers Send Notification (included) to inform tenants or staff of changes.

   Postconditions:
   - Lease or policy information is updated in the system.
   - Notifications are sent to relevant parties.

6. **Use Case Name**: Sign Lease Agreements
   Actors: Tenant
   Description: This use case describes how a tenant reviews and electronically signs a

lease agreement after receiving approval for their rental application.
Preconditions:
- Tenant must be authenticated.
- An approved lease must exist for the tenant.
- System must have the digital lease document generated and available.

Main Flow:
- The tenant selects "Sign Lease Agreements."
- The system displays the lease terms and agreement document.
- The tenant reviews the document.
- The tenant electronically signs the lease.
- The system verifies the signature and stores the signed document.
- The system updates the tenant's status to "Lease Signed."
- The system triggers Send Notification (included) to inform staff of the completed signing.

Postconditions:
- Lease is marked as signed and stored in the database.
- Staff are notified for next steps (e.g., move-in scheduling).

7. **Use Case Name**: Submit Maintenance Requests
Actors: Tenant
Description: This use case allows tenants to submit maintenance or repair requests for issues in their rented units.
Preconditions:
   - Tenant must be authenticated.
   - Tenant must have an active lease.

Main Flow:
   - The tenant selects "Submit Maintenance Requests."
   - The system displays categories of maintenance issues (e.g., plumbing, electrical, appliance).
   - The tenant enters the problem description and optionally uploads photos.
   - The system validates the request information.
   - The system saves the maintenance request and assigns a ticket number.
   - The system triggers Send Notification (included) to notify maintenance staff or administrators.
   - The system updates the request status to "Submitted."

Postconditions:
   - Maintenance request is logged and visible to maintenance/admin staff.
   - Notification is sent to the staff for action.

5. User Stories

- Format: As a [user role], I want to [goal] so that [reason]
- Map each user story to functional requirements where applicable
- Indicate story points (effort estimate) and acceptance criteria

5.1 Feature 1: Store Unit Management & Search

User Story 1 — Create Software Requirements Specification (SRS)

Functional Requirement(s): *All FRs*
Story:
As a project manager, I want a complete SRS so that all stakeholders have a clear understanding of functional and non-functional requirements.

Acceptance Criteria:

- SRS includes FRs, NFRs, Use Cases & User Stories
- All requirements mapped to user needs
- Structure follows SDLC format
- SRS is reviewed and approved by team/instructor

Priority: High
Story Points: 5

User Story 2 — Develop System Architecture & UML Design

Functional Requirement(s): *All FRs*

Story:
As a system architect, I want UML diagrams, system architecture, and ERD so the development team has a clear blueprint for implementation.

Acceptance Criteria:

- High-level architecture (MVC / layered) created
- UML Class, Sequence & Activity Diagrams completed
- ERD reflects all FRs
- Review meeting held with developers

Priority: High
Story Points: 8

User Story 3 — Implement Role-Based Access Control (RBAC)

Functional Requirement(s): FR23–FR24

Story:
As an Admin, I want to assign roles (Tenant, Agent, Admin) so only authorized users can access sensitive features.

Acceptance Criteria:

- Users must login/have an account to be assigned a role
- Roles are stored in DB
- Permission logic enforced on UI/API
- Unauthorized access is blocked
- Authentication tested with sample users

Priority: High
Story Points: 5

User Story 4 — Digital Lease Agreement Generation & Electronic Signing

Functional Requirement(s): FR9–FR11

Story:
As a Tenant, I want the system to generate digital lease agreements automatically that I can electronically sign so I don't need physical paperwork.

Acceptance Criteria:
- Lease template generated automatically
- Lease stored in database
- Electronic signing enabled when lease is approved
- Status changes to "Active" after signing

Priority: High
Story Points: 8

User Story 5 — Automated Late Payment & Overdue Alerts
Functional Requirement(s): FR17

Story:
As an Admin, I want automated overdue alerts so reminders don't need to be manually sent.

Acceptance Criteria:
- Email/SMS alert logic implemented
- Alerts stored in DB / logs
- Triggered automatically after due date
- Can be disabled/configured by admin

Priority: Medium
Story Points: 3

User Story 6 — Bulk Discount & Multiple Unit Billing Logic
Functional Requirement(s): FR15

Story:
As a Tenant leasing multiple units, I want bulk discount logic so I get accurate billing.

Acceptance Criteria:
- Billing formula applies discount correctly
- Discount shown on invoice
- Can handle multiple scenarios
- Logs recorded for audit purposes

Priority: Medium
Story Points: 5

User Story 7 — Multi-Mall Property Management
Functional Requirement(s): FR1–FR3

Story:
As an Admin, I want to manage multiple malls with separate store inventories so that each mall can operate independently and be tracked accurately.

Acceptance Criteria:
- Mall entity created in database
- Stores linked to specific mall
- UI supports mall switching/filtering
- View per-mall inventory page

- Add/Update/Delete mall inventory(units)

Priority: Medium
Story Points: 4

User Story 8 — Mobile-Friendly & Responsive UI
Functional Requirement(s): NFR7–NFR8

Story:
As a Tenant, I want a responsive UI so I can manage leases and requests on any device.

Acceptance Criteria:
- UI tested on mobile, tablet, desktop
- Works on Chrome, Firefox, Edge, Safari
- Menu/navigation adapts per device
- Layout meets accessibility guidelines

Priority: Medium
Story Points: 5

User Story 9 — System Recovery from Partial Failures
Functional Requirement(s): NFR4

Story:
As a system user, I want transactions to fully complete or roll back so that no data is corrupted.

Acceptance Criteria:
- Failure detection logic implemented
- Rollback tested on sample transactions
- DB state remains consistent after failure
- Failure log recorded per event

Priority: High
Story Points: 5

User Story 10 — Lease Tracking & Automatic Renewal
Functional Requirement(s): FR12–FR13
Story:
As a Tenant I want to be able to track lease expiries and have automatic renewals

Acceptance Criteria:
- Lease start & end date stored in DB
- Email alert **X days before expiry and renewal**
- Summary dashboard shows soon-to-expire leases
- Leases can be automatically renewed
- User notification triggered automatically

Priority: Medium
Story Points: 4

User Story 11 — Testing & Code Quality (JUnit, CI/CD)
Functional Requirement(s): NFR3–NFR4

Story:
As a QA Engineer, I want automated testing & CI/CD so that system reliability is verified.

Acceptance Criteria:
- JUnit test suite created
- CI/CD pipeline runs on push
- Code coverage reviewed & documented
- Minimum threshold set for pass/fail

Priority: High
Story Points: 5

User Story 12 — 2-Second Search Property Performance
Functional Requirement(s): NFR1 & FR4–FR5

Story:
As a Tenant, I want to be able to search properties for viewing and search results to load within 2 seconds so the system feels fast.

Acceptance Criteria:
- User can search for properties/units
- Properties and Units are displayed to the user
- Search optimized for speed
- Indexing added to DB
- Time logged for performance testing
- Performance report generated

Priority: High
Story Points: 3

User Story 13 — Transaction Atomicity (Full Rollback)
Functional Requirement(s): NFR4

Story:
As a user, I want full rollbacks so database safety is guaranteed.

Acceptance Criteria:
- Rollback rules defined
- Failure scenarios tested
- DB remains consistent after crashes
- Rollback logs recorded

Priority: High
Story Points: 3

User Story 14 — Import & Manage Utility Usage Data for Monthly Bill
Functional Requirement(s): FR14–FR19

Story:
As Admin Staff, I want to input/import utility data so monthly bills reflect real consumption.

Acceptance Criteria:
- CSV/manual input supported
- Data stored in utility table
- Calculate monthly bill based on rent, payment cycle and utility charges

- Generate invoice

Priority: High
Story Points: 5

User Story 15 — Misuse Billing for Maintenance
Functional Requirement(s): FR22

Story:
As a Tenant, I want misuse-related maintenance charges applied fairly so that I'm only billed when I'm actually responsible for the damage.

Acceptance Criteria:
- Misuse billing logic added
- Admin override available
- Cost shown on invoice
- Logged in DB for traceability

Priority: Medium
Story Points: 4

User Story 16 — Enhancement & E-Portfolio Update
Functional Requirement(s): *General – Final Sprint*

Story:
As a student developer, I want to fix bugs and polish UI so I can present the project professionally.

Acceptance Criteria:
- Bugs from Sprint 4 fixed
- UI reviewed & improved
- Portfolio prepared for submission
- Final documentation updated

Priority: Low
Story Points: 3

User Story 17 — Schedule Viewing Appointments
Functional Requirement(s): FR6–FR8

Story:
As a tenant, I would like to schedule viewing appointments with a leasing agent to view a property.

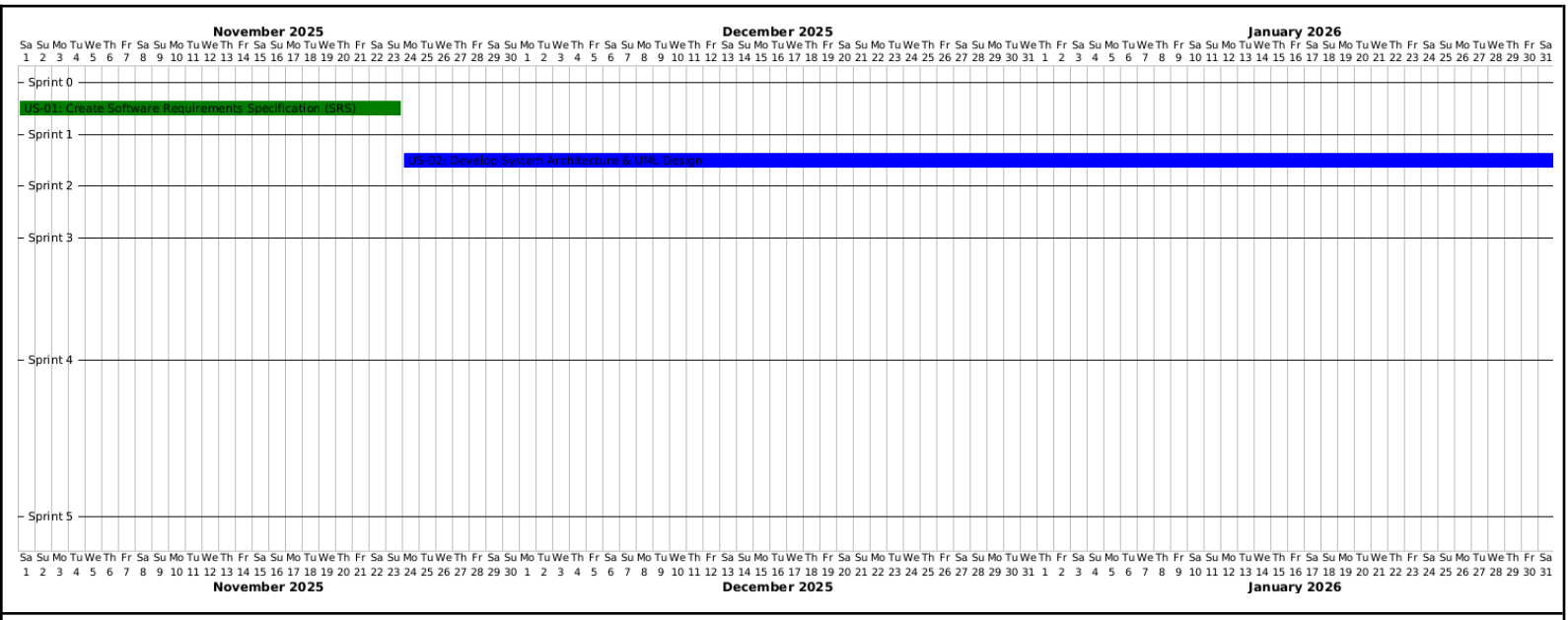
Acceptance Criteria:
- Choose a property and schedule an appointment to view it in an available time slot
- Prevent any double booking
- Get a confirmation notification

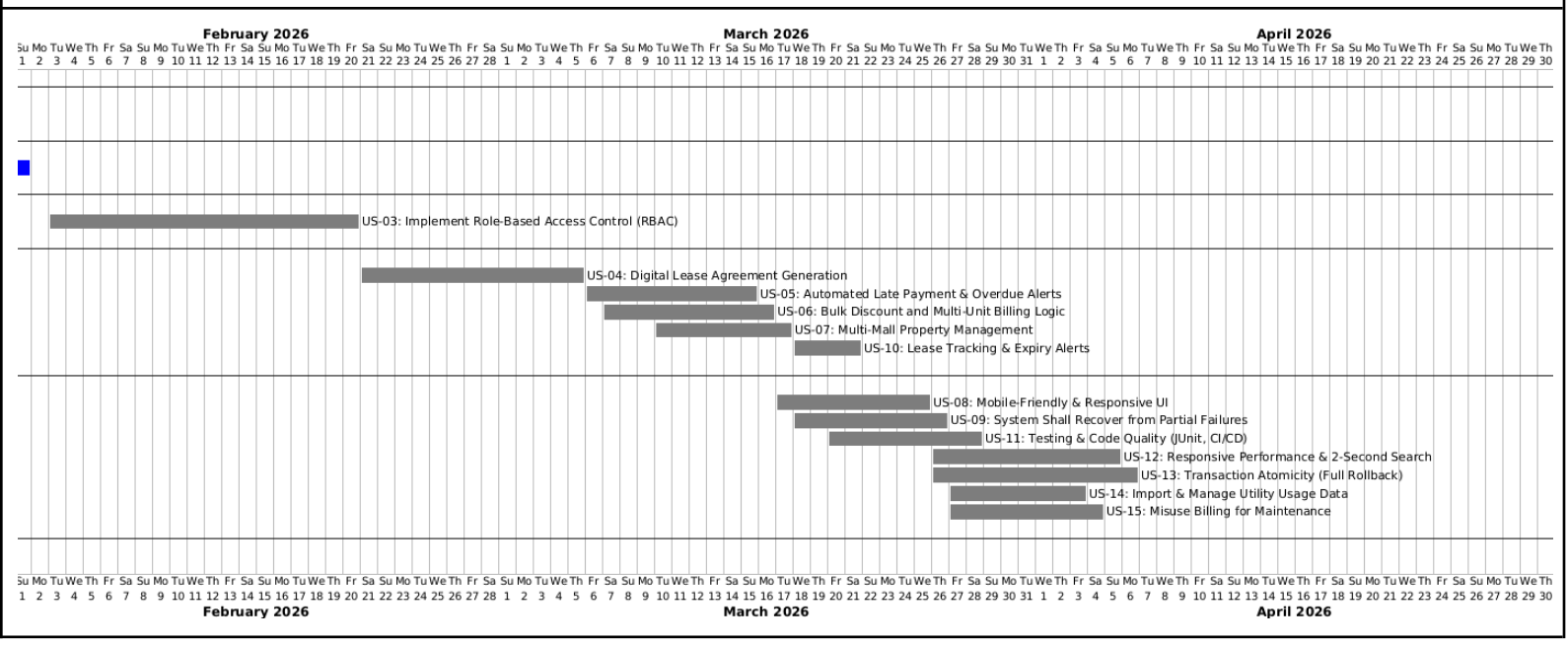
Priority: High
Story Points: 6

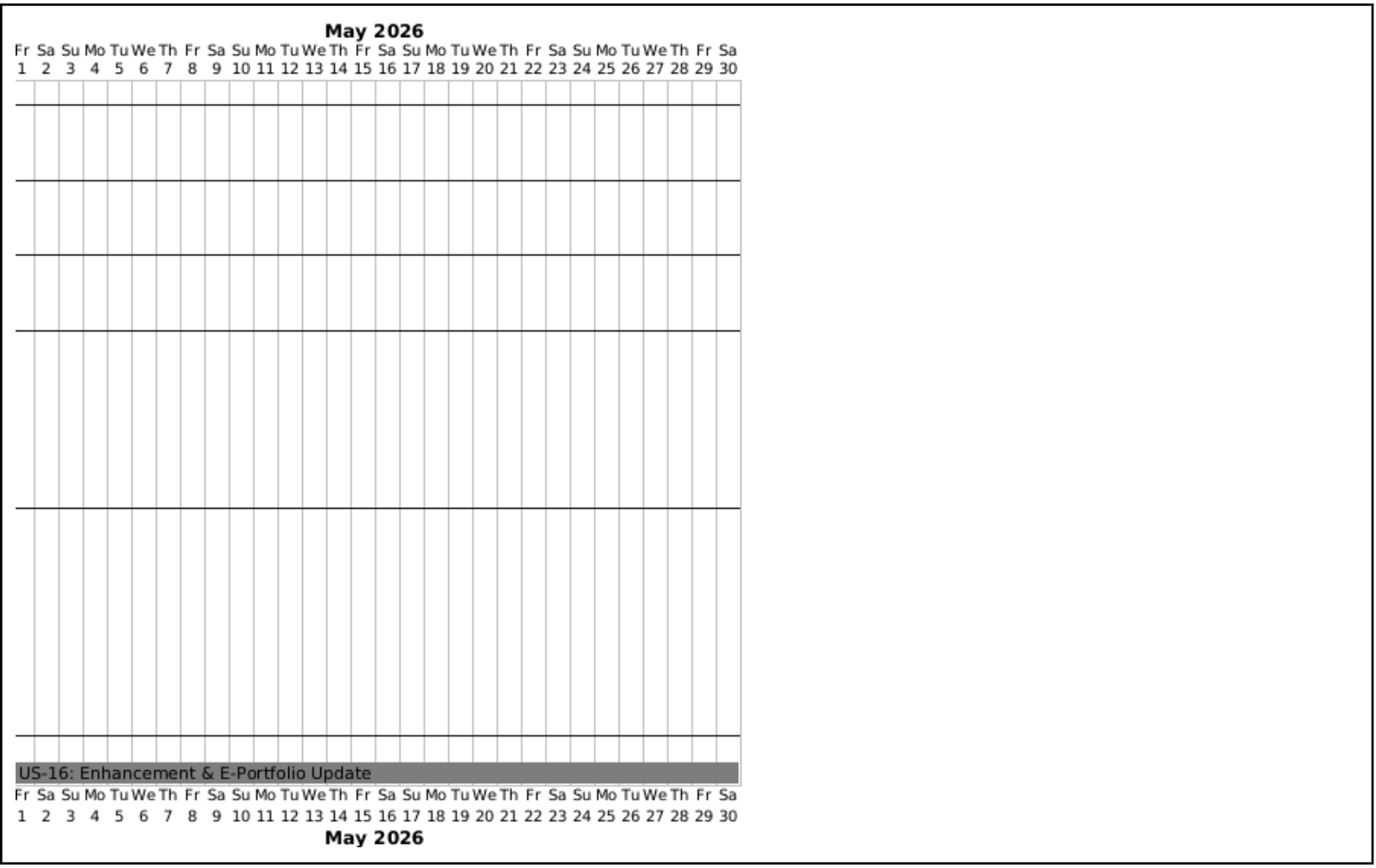
6. Project Management Plan

6.1 Gantt Chart

The following timeline outlines the major phases and deliverables derived from the project design objectives.







6.2 Project Backlog and Sprint Backlog

Assuming you follow a Scrum process model, provide a list of product backlog items so that you can select items for your Sprint backlog. Make sure the product backlog list and the tasks in each product backlog item are consistent with the Gantt Chart in Section 5.1. above. Follow the templates provided for you on eClass.

- agile-product-backlog-Team3.xlsx
- agile-sprint-backlog-Team3.xlsx

6.3 Group Meeting Logs

6.3.1 Deliverable 1 Meeting Logs

Add the minutes of each meeting, listing the attendance, what the topics of discussion in the meeting were, any decisions that were made, and which team members were assigned which tasks. These minutes must be submitted with the project report in each deliverable and will provide input to be used for the overall assessment of the project.

Present	Meeting Date	Issues Discussed / Resolved
Sienna, Mahjabin, Eamon	Oct 18, 2025	Defined project scope and assigned sections. Sienna to start on Introduction and Problem Statement. Mahjabin to draft System Specs. Eamon to begin User Stories and Project Management Plan.
Sienna, Mahjabin, Eamon	Nov 01, 2025	Discussion: Alignment of Requirements and Stories. Resolved: Mahjabin presented the draft Functional Requirements (Section 3). Eamon reviewed them to ensure they mapped 1-to-1 with the User Stories (Section 5) and created the initial backlog items based on them. Sienna finalized the "Existing Solutions" and "Motivation" text to ensure the scope wasn't too broad.
Sienna, Mahjabin, Eamon	Nov 15, 2025	Discussion: Diagrams and Timeline. Resolved: Reviewed Mahjabin's Use Case Diagram (Section 4) to ensure no actors were missing. Eamon presented the Gantt Chart (Section 6) to confirm the sprint dates with the team. Sienna completed the Design Objectives and Limitations sections.
Sienna, Mahjabin, Eamon	Nov 22, 2025	Discussion: Final Report Assembly. Resolved: Merged Eamon's Testing Plan and Project Management sections with Sienna's Intro and Mahjabin's Specs. Sienna generated the Table of Contents and completed both product and sprint backlogs. Eamon verified that the "Testing Plan" (Section 7) covered the high-priority requirements listed in Section 3. Submitted final PDF.

7. Testing and Performance Analysis Plan

Field	Details
Test Case ID	TC-001
Title	Search Store Unit - Valid Price Filter
Requirement	FR4: The system shall allow tenants to search using filters.
Preconditions	Database contains units with rents: $1000, $2000, $3000. User is on Search Page.
Steps	1. Enter "$2500" in the "Max Price" filter. 2. Click "Search".
Expected Result	The system displays the units costing $1000 and $2000 only.
Actual Result	*(To be filled during testing)*
Status	*(To be filled during testing)*
Priority	High

Field	Details
Test Case ID	TC-002
Title	Appointment Scheduling - Conflict Check
Requirement	FR7: The system shall prevent double-booking.
Preconditions	Agent A is already booked for Nov 20th at 2:00 PM.
Steps	1. User B attempts to book Agent A for Nov 20th at 2:00 PM. 2. User submits booking request.
Expected Result	System rejects the request and displays an error: "Time slot unavailable."
Actual Result	*(To be filled during testing)*
Status	*(To be filled during testing)*
Priority	High

Field	Details
Test Case ID	TC-003
Title	Submit Rental Application – Valid Information
Requirement	FR9: Allow tenants to submit digital rental applications
Preconditions	Tenant is logged in
Steps	1. Navigate to "Submit Rental Application" 2. Complete all mandatory fields 3. Upload required documents 4. Click "Submit"
Expected Result	● Application is stored ● Leasing agent receives notification ● Tenant sees confirmation message
Actual Result	*(To be filled during testing)*
Status	*(To be filled during testing)*
Priority	Medium

Field	Details
Test Case ID	TC-004
Title	Manage Store Units – Add New Unit
Requirement	FR1: Add, update, view, and delete store unit records.
Preconditions	Leasing Agent is logged in
Steps	1. Select "Manage Store Units" 2. Click "Add Unit" 3. Enter valid unit info (size, rate, classification, purpose) 4. Save
Expected Result	New unit stored and unit appears in the unit list
Actual Result	*(To be filled during testing)*
Status	*(To be filled during testing)*
Priority	High

Field	Details
Test Case ID	TC-005
Title	Calculate Bill
Requirement	FR:19: Consolidate rent and utility charges into monthly bill
Preconditions	● Lease exists with monthly rent ● Billing date reached
Steps	1. System calculates rent 2. System applies discounts if applicable 3. System generates invoice
Expected Result	● A monthly bill gets created with rent and utility charges included and the user is notified. ● Invoice is created
Actual Result	*(To be filled during testing)*
Status	*(To be filled during testing)*
Priority	High

Field	Details
Test Case ID	TC-006
Title	User can successfully login
Requirement	FR23: Support user authentication
Preconditions	User is logged out
Steps	1. User clicks “Login” 2. User enters username and password 3. If credentials are correct they are logged in and brought to home page 4. If credentials are incorrect they are prompted to try again
Expected Result	Upon successful login user is redirected to home page
Actual Result	*(To be filled during testing)*
Status	*(To be filled during testing)*
Priority	High

Field	Details
Test Case ID	TC-007
Title	Attempt to Sign Unapproved Lease Agreement
Requirement	FR11: Allows user to electronically sign lease agreement
Preconditions	● Tenant is logged in ● Tenant has no approved lease to sign
Steps	1. Click on “Leases” 2. Click on an unapproved lease to sign 3. Attempt to sign it
Expected Result	● System prevents access to lease signing. ● Message displayed: “Your application must be approved before signing.”
Actual Result	*(To be filled during testing)*
Status	*(To be filled during testing)*
Priority	Medium

Field	Details
Test Case ID	TC-008
Title	Role-Based Access Control - Units/Property
Requirement	FR24: Enforce role-based access control
Preconditions	● Tenant is logged in and has the role of tenant
Steps	1. Tenant clicks on “Properties”
Expected Result	● Tenant does not have any options to add, update or delete any properties ● They can only search for properties
Actual Result	*(To be filled during testing)*
Status	*(To be filled during testing)*
Priority	High

NFR: Performance Analysis, Security, Compatibility & Responsiveness

Field	Details

Test Case ID	TC-009
Title	Search Response Time
Requirement	NFR1: System shall process search queries within 2 second.
Preconditions	Database populated with 100+ mock unit records.
Steps	1. Initiate a search query with multiple filters. 2. Measure time from request to display.
Expected Result	Results load in < 2.0 seconds.
Actual Result	*(To be filled during testing)*
Status	*(To be filled during testing)*
Priority	Medium

Field	**Details**
Test Case ID	**TC-010**
Title	Input Validation - Special Characters
Requirement	**NFR5**: The system shall validate user input to prevent malformed or invalid data.
Preconditions	• Admin is logged in. • User is on the "Add Store Unit" or "Edit Profile" page.
Steps	1. Navigate to a text input field (e.g., "Unit Name"). 2. Enter a script tag string: <script>alert('test')</script>. 3. Click "Save".
Expected Result	• The system rejects the input or sanitizes it (removing the script tags). • An error message "Invalid characters detected" is displayed. • The script does **not** execute.
Actual Result	*(To be filled during testing)*
Status	*(To be filled during testing)*
Priority	High

Field	Details
Test Case ID	**TC-011**
Title	Delete Confirmation Dialog
Requirement	**NFR6**: The system shall provide confirmation dialogs for critical actions.
Preconditions	• Admin is logged in. • "Manage Store Units" list is visible.
Steps	1. Locate an existing store unit. 2. Click the "Delete" button. 3. Wait for the popup. 4. Click "Cancel".
Expected Result	• A confirmation dialog appears asking "Are you sure?". • Upon clicking "Cancel", the dialog closes and the unit is **not** deleted.
Actual Result	*(To be filled during testing)*
Status	*(To be filled during testing)*
Priority	Medium

Field	Details
Test Case ID	**TC-012**
Title	Browser Layout Consistency
Requirement	**NFR8**: The system shall be compatible with standard web browsers (Chrome, Firefox, Edge).
Preconditions	• The system is deployed and accessible via URL.
Steps	1. Open the "Tenant Dashboard" in Google Chrome and verify layout.

	2. Open the same page in Mozilla Firefox and verify layout. 3. Compare the navigation bar and button alignment.
Expected Result	• The layout, fonts, and button positions appear identical and functional across both browsers.
Actual Result	*(To be filled during testing)*
Status	*(To be filled during testing)*
Priority	Low

Field	**Details**
Test Case ID	**TC-013**
Title	Mobile Interface Adaptation
Requirement	**NFR7**: The system shall present a user-friendly, responsive interface for mobile devices.
Preconditions	• User is accessing the system via a desktop browser.
Steps	1. Open the application in a browser. 2. Resize the browser window width to 375px (mobile view). 3. Observe the navigation menu and data tables.
Expected Result	• The navigation menu collapses into a "hamburger" icon. • Data tables stack or become scrollable without breaking the layout. • No horizontal scrolling is required for main content.
Actual Result	*(To be filled during testing)*
Status	*(To be filled during testing)*
Priority	Medium