University Management System

Sprint 1 Documentation

Facilities Module - Phase 1

Omar Mohamed Mostafa (22p0197) - Product Owner
Ezzeldin Ismail Kaoud (22p0141) - Scrum Master
Roaa Sherif Gadara (22p0188) - Developer
Youssef Amr Farouk (2200844) - Developer
Ahmed Wael Rafaat (22p0221) - Developer

November 16-21, 2025

Abstract

This document provides complete Sprint 1 documentation for the University Management System, covering the Facilities Module. Sprint duration: 5 days (Nov 16-21, 2025). Delivered features: User Authentication, Room Availability Viewing, Room Management (Admin), and Maintenance Reporting. Total: 23 story points completed (100% velocity).

Contents

1 Introduction

1.1 Project Overview

The University Management System is a comprehensive software solution for managing university operations including facilities, student records, and administrative processes.

Key Objectives:

- Facilities Management (classrooms, labs, equipment)
- Role-based access control for different user types
- Maintenance tracking and reporting
- Resource optimization

1.2 Sprint 1 Scope

Sprint 1 focuses on core facilities management:

- User authentication with role-based access
- Room availability viewing for all users
- Room CRUD operations for administrators
- Maintenance issue reporting system

1.3 Technology Stack

Component	Technology
Frontend	JavaFX 21.0.2
Backend	Java 21
Database	Microsoft SQL Server
Build Tool	Maven
Version Control	Git/GitHub
Project Management	Jira

2 Why We Chose Scrum

2.1 Key Reasons

1. **Iterative Development:** Deliver working software every sprint
2. **Flexibility:** Adapt to changing requirements quickly
3. **Transparency:** Daily scrums keep everyone informed
4. **Team Size:** 5 members is ideal for Scrum
5. **Continuous Improvement:** Retrospectives drive process enhancement

2.2 Scrum vs Other Methodologies

Aspect	Scrum (Chosen)	Waterfall
Feedback	Every sprint	End of project
Flexibility	High	Low
Risk	Early detection	Late detection
Delivery	Incremental	Big bang

3 Scrum Team Roles

3.1 Product Owner: Omar Mohamed Mostafa

Responsibilities:

- Manages and prioritizes Product Backlog
- Defines user stories and acceptance criteria
- Makes decisions on feature priorities
- Validates completed work meets requirements

Sprint 1 Contributions: Database design, backlog prioritization, acceptance testing.

3.2 Scrum Master:Ezzeldin Ismail

Responsibilities:

- Facilitates all Scrum ceremonies
- Removes impediments blocking the team
- Coaches team on Scrum practices
- Ensures process is followed correctly

Sprint 1 Contributions: Auth service, Maintenance backend Facilitated 6 daily scrums.

3.3 Development Team

Member	Role	Sprint 1 Work
Roaa Sherif	Frontend Dev	Rooms UI, Admin forms, CSS styling
Youssef Amr	Backend Dev	Room CRUD, Controllers, DB integration
Ahmed Wael Rafaat	Frontend Dev	login UI, maintenance UI, testing.

4 Product Backlog

4.1 Backlog Overview

The Product Backlog contains all features for the Facilities Module, prioritized using MoSCoW method.

ID	User Story	Points	Priority	Sprint
US 4.1	User Authentication	5	Must	1
US 1.1	View Room Availability	5	Must	1
US 1.3	Manage Room Records	5	Must	1
US 1.6	Report Maintenance Issue	5	Must	1
US 1.2	Book a Room	8	Should	2
US 1.4	Modify/Cancel Booking	5	Should	2
US 1.5	View My Bookings	3	Should	2
US 1.7	View Maintenance Status	3	Could	2

Table 1: Product Backlog with MoSCoW Prioritization

4.2 MoSCoW Classification

- **Must Have:** Essential features for MVP (Sprint 1)
- **Should Have:** Important but not critical (Sprint 2)
- **Could Have:** Nice to have if time permits
- **Won't Have:** Out of scope for this phase

5 Sprint Planning

5.1 Planning Meeting Details

Attribute	Value
Date	November 16, 2025
Duration	1 hours
Attendees	All 5 team members
Sprint Duration	5 days
Sprint Goal	Core facilities with authentication
Committed Points	23 points

5.2 Sprint Goal

"Deliver a working increment that allows users to authenticate based on their roles, view room availability, enables administrators to manage room records, and provides all users the ability to report maintenance issues."

5.3 Planning Poker Estimation

We used Planning Poker with Fibonacci sequence (1, 2, 3, 5, 8, 13, 21):

Story	Omar	Ahmed	Roaa	Youssef	Final
US 4.1 Auth	5	8	5	5	5
US 1.1 View Rooms	5	5	5	5	5
US 1.3 Manage Rooms	5	5	8	5	5
US 1.6 Maintenance	5	5	5	5	5

Table 2: Planning Poker Results

6 User Stories

6.1 US 4.1 - User Authentication

Story: As a user, I want to log in based on my role, so that I can access services available to me.

Acceptance Criteria:

- Login with valid credentials grants access
- Invalid credentials show error message
- Users redirected to role-appropriate dashboard
- Input validation on all fields

Tasks:

1. DB design for Users & Roles (Omar)
2. Backend login logic (Ezzeldin)
3. Login UI (Ahmed)
4. Role-based access (Ezzeldin/Omar)
5. Testing (Ahmed)

6.2 US 1.1 - View Room Availability

Story: As a student/professor, I want to view available classrooms/labs so that I can find a free room.

Acceptance Criteria:

- Display all rooms with real-time availability
- Show: ID, type, capacity, location, status
- Filter by type and status
- Search functionality

Tasks:

1. DB design for Rooms (Omar)
2. Backend service to list rooms (Youssef)
3. Rooms page UI (Roaa)
4. Connect UI to backend (Roaa/Youssef)

6.3 US 1.3 - Manage Room Records

Story: As an admin, I want to create, edit, and delete room records so that room data is accurate.

Acceptance Criteria:

- Admin can create rooms with: name, type, capacity, location
- Admin can edit type and capacity
- Delete verifies no future bookings exist
- Non-admin users cannot access management features

Tasks:

1. Admin-only access check (Youssef) -
2. Backend CRUD operations (Youssef) -
3. Admin UI forms (Roaa) -
4. Integration testing (Youssef/Roaa) -

6.4 US 1.6 - Report Maintenance Issue

Story: As a user, I want to report maintenance issues so they can be resolved.

Acceptance Criteria:

- Submit ticket with Room ID and description
- Ticket created with NEW status
- Success message shown after submission
- Ticket ID displayed for tracking

Tasks:

1. DB design for MaintenanceTickets (Omar)
2. Backend ticket creation (Ezzeldin)
3. Maintenance form UI (Ahmed)
4. Testing (Ezzeldin/Omar)

7 Sprint Backlog

7.1 Task Breakdown

Task ID	Task Description	Assignee	Hours	Status
T4.1.1	DB structure for Users/Roles	Omar	4	Done
T4.1.2	Backend login logic	Ezzeldin	6	Done
T4.1.3	Login UI	Ahmed	5	Done
T4.1.4	Role-based access check	Ezzeldin	4	Done
T4.1.5	Login testing	Ahmed	3	Done
T1.1.1	DB design for Rooms	Omar	3	Done
T1.1.2	Backend room service	Youssef	4	Done
T1.1.3	Rooms page UI	Roaa	5	Done
T1.1.4	UI-Backend connection	Youssef	4	Done
T1.3.1	Admin access control	Youssef	2	Done
T1.3.2	Backend CRUD	Youssef	6	Done
T1.3.3	Admin UI forms	Roaa	6	Done
T1.3.4	Integration testing	Roaa/Youssef	3	Done
T1.6.1	DB for MaintenanceTickets	Omar	3	Done
T1.6.2	Ticket creation backend	Ezzeldin	4	Done
T1.6.3	Maintenance form UI	Ahmed	4	Done
T1.6.4	Maintenance testing	Ezzeldin	2	Done

Table 3: Complete Sprint Backlog

8 Daily Scrum Meetings

8.1 Meeting Format

- **Time:** 10 :00 PM daily
- **Duration:** 15 minutes (timeboxed)
- **Facilitator:** Ezzeldin Ismail(Scrum Master)
- **Format:** Three questions per member

8.2 Daily Scrum Screenshot

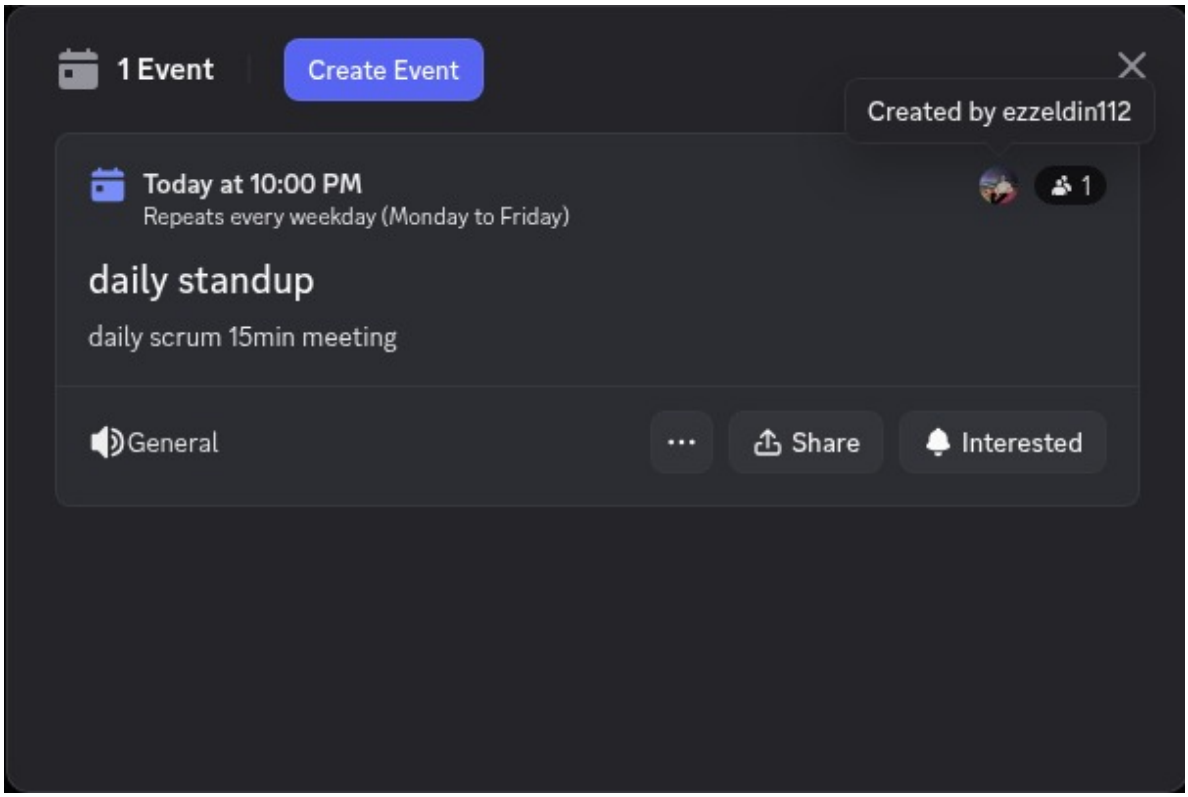


Figure 1: Daily Scrum Meeting Screenshot

8.3 Daily Scrum Records

8.3.1 Day 1 - November 16 (Saturday)

Member	Yesterday	Today	Blockers
Omar	Sprint planning	DB schema design	None
Ahmed	Sprint planning	Login UI mockup	None
Roaa	Sprint planning	Rooms UI design	None
Youssef	Sprint planning	Backend structure	None
Ezzeldin	Sprint planning	Auth research	None

8.3.2 Day 2 - November 17 (Sunday)

Member	Yesterday	Today	Blockers
Omar	DB schema done	Fix auto-increment	ID config issue
Ahmed	Login UI 75%	Complete login UI	None
Roaa	Rooms UI done	Admin UI forms	None
Youssef	Backend CRUD	UI integration	None
Ezzeldin	Auth service	Login logic	None

Blocker Resolved: Omar fixed SQL Server auto-increment configuration.

8.3.3 Day 3 - November 18 (Monday)

Member	Yesterday	Today	Blockers
Omar	DB issues fixed	Coordinate backend	None
Ahmed	Login UI done	Maintenance UI	None
Roaa	Admin forms 80%	Complete forms	None
Youssef	Integration 50%	Complete integration	None
Ezzeldin	Login logic done	Maintenance backend	None

8.3.4 Day 4 - November 19 (Tuesday)

Member	Yesterday	Today	Blockers
Omar	Backend coord	Role-based testing	None
Ahmed	Maint UI done	Login testing	None
Roaa	Forms complete	Room testing	None
Youssef	Integration done	CRUD testing	None
Ezzeldin	Maint backend	Auth edge cases	Minor auth bugs

8.3.5 Day 5 - November 20-21 (Wed-Thu)

Member	Yesterday	Today	Blockers
All	Testing complete	Sprint Review prep	None

Sprint Status: ALL 4 USER STORIES COMPLETED

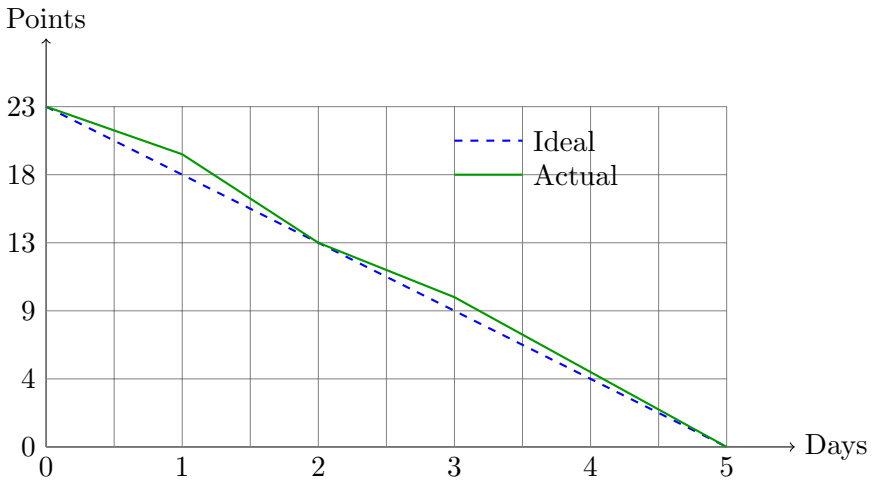
9 Sprint Burndown Chart

9.1 Burndown Data

Day	Ideal Remaining	Actual Remaining	Completed
Day 0	23	23	0
Day 1	18.4	20	3
Day 2	13.8	14	9
Day 3	9.2	10	13
Day 4	4.6	5	18
Day 5	0	0	23

Table 4: Burndown Data (Story Points)

9.2 Burndown Analysis



Key Observations:

- Sprint started slightly behind ideal (Day 1-2)
- Team caught up by Day 3
- Finished exactly on target
- Velocity: 23 points / 5 days = 4.6 points/day

10 Implementation Highlights

10.1 Architecture Overview

The system follows MVC (Model-View-Controller) architecture:

- **Model:** User, Room, MaintenanceTicket, enums (RoomType, RoomStatus, TicketStatus)
- **View:** FXML files with CSS styling
- **Controller:** LoginController, RoomsController, MaintenanceController, etc.
- **Service:** AuthService, RoomService, MaintenanceService

10.2 Key Components Implemented

10.2.1 Authentication System

- Singleton AuthService for session management
- Role-based user types: ADMIN, STUDENT, PROFESSOR, STAFF
- SQL Server integration with prepared statements
- Password validation and error handling

10.2.2 Room Management

- Complete CRUD operations via RoomService
- Admin-only access control in RoomsController
- Search and filter functionality
- Real-time statistics (available, booked, maintenance)

10.2.3 Maintenance System

- Ticket creation with auto-generated ID
- Room validation before ticket submission
- Status tracking (NEW, IN_PROGRESS, RESOLVED)

10.3 Database Schema

Table	Key Fields
Users	UserID (PK), Username, Password, UserType
Rooms	RoomID (PK), Code, Name, Type, Capacity, Location, Status
MaintenanceTickets	TicketID (PK), RoomID (FK), ReporterUserID (FK), Description, Status

11 Testing Summary

11.1 Test Cases Executed

Feature	Test Case	Result	Tester
Login	Valid credentials login	Pass	Ahmed
Login	Invalid credentials rejection	Pass	Ahmed
Login	Empty field validation	Pass	Ahmed
Login	Role-based redirect	Pass	Ahmed
Rooms	View all rooms	Pass	Roaa
Rooms	Search functionality	Pass	Roaa
Rooms	Filter by type/status	Pass	Roaa
Rooms	Admin access only	Pass	Youssef
Room CRUD	Create new room	Pass	Youssef
Room CRUD	Edit room details	Pass	Youssef
Room CRUD	Delete room	Pass	Youssef
Room CRUD	Validation checks	Pass	Youssef
Maintenance	Submit ticket	Pass	Ezzeldin
Maintenance	Invalid room error	Pass	Ezzeldin
Maintenance	Success confirmation	Pass	Ezzeldin

Table 5: Test Results Summary

Test Coverage: 16/16 test cases passed (100%)

12 Sprint Review

12.1 Meeting Details

Attribute	Value
Date	November 22, 2025
Duration	20 mins
Attendees	All team members and Stakehokder(DR)

12.2 Demo Summary

1. **Login Demo:** Showed authentication for different roles (admin, student, professor)
2. **Room Viewing:** Demonstrated room list with search and filters
3. **Admin Functions:** Created, edited, and deleted a room
4. **Maintenance:** Submitted a ticket and showed confirmation

13 Sprint Retrospective

13.1 Meeting Details

Attribute	Value
Date	November 21, 2025
Duration	1 hour
Format	Start-Stop-Continue

13.2 What Went Well (Continue)

- Daily scrums kept everyone aligned
- Task breakdown was detailed and accurate
- Team collaboration was excellent
- All stories completed on time
- Code quality was maintained

13.3 What Didn't Go Well (Stop)

- Initial DB configuration caused delay (Day 2)
- Some code was written without prior discussion
- Documentation was left until the end

13.4 What To Improve (Start)

- Set up development environment before sprint starts
- Document code as we write it
- More pair programming for complex tasks
- Create coding standards document

14 Definition of Done

14.1 Story-Level DoD

A user story is considered "Done" when:

1. All acceptance criteria are met
2. Code is written and committed to Git
3. Code has been reviewed by at least one team member
4. Unit/integration tests pass
5. Feature works in the test environment
6. No critical bugs remaining
7. Product Owner has accepted the feature

14.2 Sprint-Level DoD

The sprint is "Done" when:

1. All committed user stories meet story-level DoD
2. Sprint demo has been conducted
3. Sprint retrospective has been completed
4. Documentation is updated
5. Code is merged to main branch

14.3 Sprint 1 DoD Checklist

Criteria	Status
All acceptance criteria met	Yes
Code committed to Git	Yes
Code reviewed	Yes
Tests passing	Yes
Demo conducted	Yes
Retrospective completed	Yes
Documentation updated	Yes

15 Jira Project Management

15.1 Product Backlog View

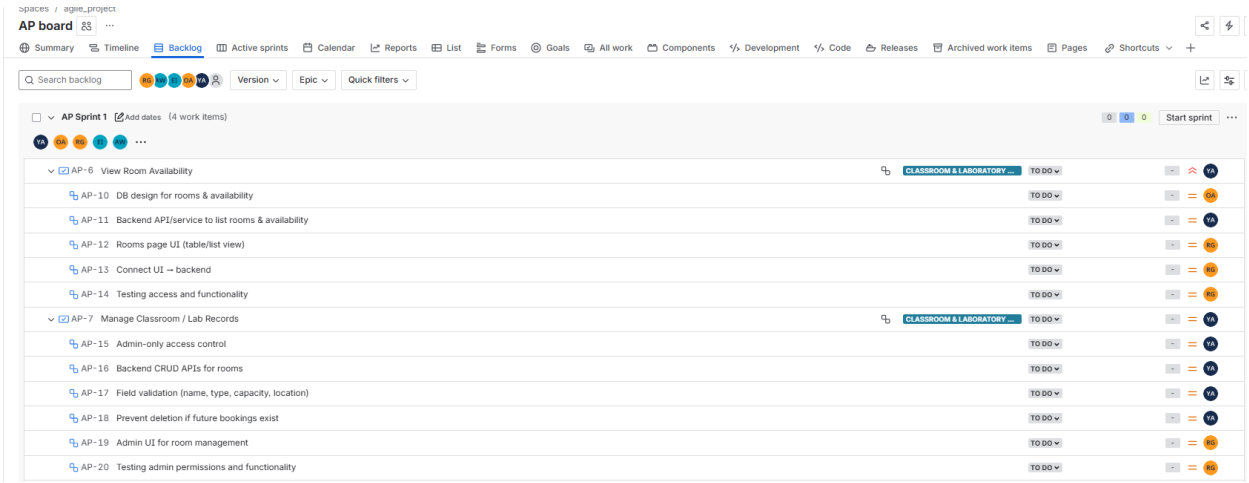


Figure 2: Jira Product Backlog

15.2 User Story Detail

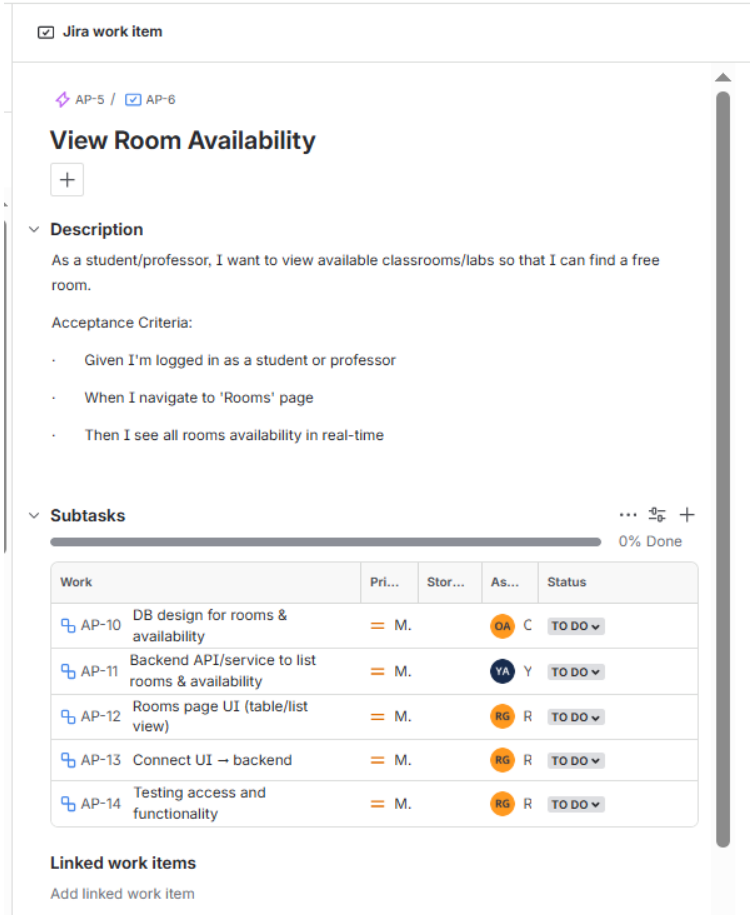


Figure 3: Jira User Story Detail View example

16 Lessons Learned

16.1 Technical Lessons

1. **Database Setup:** Always configure and test DB connection before sprint starts
2. **JavaFX + FXML:** Controller injection requires proper fx:id naming
3. **SQL Server:** Auto-increment requires IDENTITY specification
4. **Service Pattern:** Singleton pattern useful for shared services like AuthService

16.2 Process Lessons

1. **Daily Scrums:** 15-minute timebox is essential - longer meetings waste time
2. **Task Breakdown:** Detailed tasks (2-6 hours) are easier to track
3. **Estimation:** Planning Poker creates consensus and shared understanding
4. **Communication:** Daily standups prevent surprises and blockers

16.3 Team Lessons

1. Cross-functional skills help when someone is blocked
2. Pair programming accelerates knowledge sharing
3. Clear role definitions prevent confusion
4. Celebrating small wins boosts morale

17 Sprint 2 Preview

17.1 Planned User Stories

ID	User Story	Points	Priority
US 1.2	Book a Room	8	Should Have
US 1.4	Modify/Cancel Booking	5	Should Have
US 1.5	View My Bookings	3	Should Have
US 1.7	View Maintenance Status	3	Could Have
US 4.2	User Registration	5	Should Have
	Total	**24**	

Table 6: Sprint 2 Candidate Stories

18 Conclusion

18.1 Sprint 1 Summary

Sprint 1 was successfully completed with all planned user stories delivered:

- **23/23 story points** completed (100% velocity)
- **4/4 user stories** met Definition of Done
- **16/16 test cases** passed
- **0 critical bugs** in delivered features

18.2 Key Achievements

1. Established working authentication system with role-based access
2. Created complete room management functionality for administrators
3. Implemented maintenance reporting system
4. Built solid foundation for future sprints
5. Team gained valuable Scrum experience

18.3 Team Velocity

Based on Sprint 1, our established velocity is **23 story points per 5-day sprint**, or approximately **4.6 points per day**. This will be used for Sprint 2 planning.

18.4 Final Remarks

The team successfully applied Scrum principles including:

- Iterative development with working increments
- Daily communication through scrums
- Continuous improvement through retrospectives
- Collaboration and self-organization
- Transparency through Jira and documentation

We are confident in our ability to deliver Sprint 2 with the same level of quality and commitment.

A Appendix A: Git Commit History

Date	Author	Commit Message
Nov 16	Omar	Initial DB schema for Users and Rooms
Nov 16	Ahmed	Project structure setup
Nov 17	Roaa	Add rooms.fxml and RoomsController
Nov 17	Youssef	Implement RoomService with CRUD
Nov 17	Ezzeldin	Add AuthService with login logic
Nov 18	Ahmed	Complete login.fxml UI
Nov 18	Roaa	Add admin room forms (add/edit)
Nov 18	Omar	Add MaintenanceTickets table
Nov 19	Ezzeldin	Implement MaintenanceService
Nov 19	Ahmed	Add maintenanceTicket.fxml
Nov 19	Youssef	Connect RoomsController to service
Nov 20	All	Bug fixes and testing
Nov 21	All	Final testing and documentation

B Appendix B: Code Structure

```
src/main/java/edu/facilities/
 Main.java
 data/
    Database.java
 model/
    User.java (abstract)
    Admin.java
    Student.java
    Professor.java
    Staff.java
    Room.java
    RoomType.java (enum)
    RoomStatus.java (enum)
    MaintenanceTicket.java
    TicketStatus.java (enum)
 service/
    AuthService.java
    RoomService.java
    MaintenanceService.java
    BookingService.java
    DatabaseConnection.java
 ui/
     LoginController.java
     RegisterController.java
     RoomsController.java
     AddRoomController.java
     EditRoomController.java
     MaintenanceController.java
     dashboardcontroller.java

src/main/resources/
 css/
    style.css
```

```
fxml/
    login.fxml
    register.fxml
    dashboard.fxml
    rooms.fxml
    add_room.fxml
    edit_room.fxml
    maintenanceTicket.fxml
```

C Appendix : Project Repository

The project's source code and full commit history are available on GitHub:
https://github.com/OmarAshry1/Agile*project*/*tree*/*main*