Software Requirements Specification (SRS)

Project Name: Apartment & Tenant Management System (ATMS)

Team Members:

- Syed Huzaifa Mustansar
- Syed Asfandyar Ali
- M. Moin Ud Din
- M. Abdur Rafay

1. Introduction

1.1 Purpose

The **Apartment & Tenant Management System (ATMS)** is a web-based platform designed to assist landlords, tenants, and maintenance staff in managing rental properties. It facilitates rent payments, tenant record management, maintenance requests, and expense management.

1.2 Scope

The system will enable:

- **Tenants** pay rent online and request maintenance.
- **Landlords** to track rent payments, manage tenants, handle invoices, and assign maintenance tasks.
- **Maintenance Staff** to receive work orders and update task status.

1.3 Technology Stack

- **Frontend:** React.js with Tailwind CSS/Bootstrap/Material-UI
- **Backend:** Node.js with Express.js
- **Database:** MongoDB (Mongoose ORM)
- **Authentication:** JSON Web Tokens (JWT), bcrypt.js for password hashing
- **Payments:** JazzCash Payment Gateway API
- **Notifications:** Push notifications
- **Deployment:** Frontend (Vercel/Netlify), Backend (Heroku/AWS EC2), Database (MongoDB Atlas) **(optional)**

1.4 Technical & Non-technical Tools to be Used

- **Trello:** Project Management
- **Figma:** Prototyping, System and Frontend Designing
- **Git & GitHub:** Version Controlling, Collaboration
- **VS Code:** Code Editor IDE
- **MongoDB Atlas:** Managing MongoDB Cloud
- **MongoDB Compass:** Managing MongoDB
- **Vercel/Netlify:** Deployment
- **Dropbox:** File sharing & Backups
- **Whatsapp:** Communication, Texting
- **Google Meet:** Team Meetings, Communication

---

2. System Requirements

2.1 Functional Requirements

User Management

- Secure authentication (JWT-based login & registration)
- Dedicated dashboards (Tenant, Landlord, Maintenance Staff, Admin)
- Profile management (update details, change password)

Tenant Features

- View rent payment status & transaction history
- Online rent payment via JazzCash
- Submit & track maintenance requests
- Receive rent reminders & maintenance updates

Landlord Features

- Add, edit, and remove tenants
- Track rent payments & send overdue reminders
- Manage property expenses & invoices
- Assign maintenance requests to staff
- Generate financial reports (rent income, expenses)

Maintenance Staff Features

- View assigned work orders
- Update work status (Pending, In Progress, Completed)
- Upload images/documents as proof of work completion

- Receive notifications for new maintenance requests

General System Features

- Dashboard & analytics (rent collection stats, pending maintenance)
- Mobile-friendly responsive UI
- Secure online payments via JazzCash
- Search & filter functionality (tenants, payments, maintenance requests)
- Push notifications for reminders & updates

2.2 Non-Functional Requirements

- **Scalability:** System should support multiple apartment units and tenants
- **Security:** JWT based authentication, encrypted user data, and secure payments
- **Performance:** API response time should be optimized for efficiency
- **Availability:** 99.9% uptime with cloud deployment
- **Usability:** Intuitive UI for non-technical users

---

3. System Design

3.1 System Architecture

The system will follow the **MERN stack architecture**:

- **Frontend (React.js)**: Client-side UI, API requests, state management with Redux/Context API
- **Backend (Node.js, Express.js)**: Handles business logic, authentication, and database operations
- **Database (MongoDB)**: Stores tenant records, payment transactions, maintenance requests
- **Third-party Services**:
    - JazzCash for rent payments
    - Vercel, Netlify for deployment

3.2 User Roles & Dashboard Features

Role	Dashboard Features
Tenant	Pay rent, submit maintenance requests
Landlord	Manage tenants, payments, and expenses

Maintenance Staff	View and update assigned maintenance requests

---

4. Implementation Plan

4.1 Process Model

- Agile based implementation
- Using Scrum Methodology
- Incremental development will be followed
- Each sprint will last one week

4.2 Development Timeline

Sprint 1: Planning & Design (Week 1)

- Requirement gathering
- Wireframe and UI design
- Database schema design

Sprint 2: Backend Development (Week 2)

- Set up Node.js, Express.js, and MongoDB
- Implement authentication and basic dashboards
- Develop APIs for tenants, landlords, and staff

Sprint 3: Frontend Development (Week 3)

- Build UI components in React.js
- Integrate API calls and state management
- Implement authentication and authorization

Sprint 4: Core Business Logic Development (Week 4)

- Implement backend logic, database operations etc
- Implement notification system
- Implement reports, charts, analysis on relevant dashboards

Sprint 5: Payment System (Week 5)

- Integrate JazzCash for rent payments

Sprint 6: Testing & Deployment (Week 6)

- Perform unit, integration, and user acceptance testing
- Deploy frontend and backend to production
