Software Requirements Specification (SRS)

Classroom Key & Cycle Management System

Prepared By: AN-1

NAME	ROLL NUMBER
N TEJA	B221046CS
N VIVEK TEJA	B221051CS
K NAVANEETH	B220959CS

CONTENTS

1. Introduction

1.1 Purpose

The purpose of this system is to provide an efficient digital solution for managing the borrowing and returning of **classroom keys** and **cycles**. It aims to eliminate manual processes, reduce delays, and ensure transparency.

1.2 Scope

This system enables:

- **Students to borrow cycles** using **QR codes**.
- **Class Representatives (CRs) to borrow classroom keys** for a specified duration.
- **Admins to approve borrow and return requests** for keys.
- **Automated tracking and notifications** for borrowing and returning cycles and keys.

1.3 Definitions, Acronyms, and Abbreviations

- **CR (Class Representative)**: A designated student who can borrow both cycles and classroom keys.
- **Admin**: The person responsible for approving borrow and return requests.
- **QR Code**: A machine-readable code used for cycle borrowing.
- **SRS (Software Requirements Specification)**: This document detailing system requirements.

1.4 Overview

This document describes the functional and non-functional requirements, system design, and operational aspects of the **Classroom Key & Cycle Management System**.

---

2. Functional Requirements

2.1 User Roles & Permissions

Role	Permissions
Student	Borrow and return cycles
CR (Class Representative)	Borrow and return cycles & classroom keys

Role	Permissions
Admin	Approve/reject key borrowing & return requests

2.2 Key Borrowing Process (Only for CRs)

1. CR selects an available key.
2. CR enters the **borrow duration (in minutes)**.
3. A **request is sent to the admin for approval**.
4. Upon admin approval, the key is assigned to the CR.
5. After the duration expires, the key is **automatically marked as "Available"**.
6. The CR can request an **early return**, which must also be approved by the admin.

2.3 Cycle Borrowing Process (For Students & CRs)

1. User scans the **QR code on a cycle**.
2. A request is generated.
3. The system **checks if the user already has a borrowed cycle**.
4. If not, the cycle is assigned to the user.
5. User can return the cycle by scanning the **same QR code**.

2.4 Admin Approval System

- Admin can **view pending key borrow & return requests**.
- Admin can **approve or reject** requests.
- Admin can view **current key holders & cycle status**.

2.5 Notifications & Alerts

- Users receive **real-time notifications** upon approval or rejection.
- Reminders are sent when a **borrowed key's duration is about to expire**.

---

3. Non-Functional Requirements

3.1 Performance Requirements

- The system should handle **100+ concurrent users**.
- QR code scanning should take **less than 2 seconds**.

3.2 Security Requirements

- JWT authentication for **secure user login**.
- Role-based access control for **Students, CRs, and Admins**.
- **Encrypted QR codes** to prevent forgery.

3.3 Usability Requirements

- The system should have a **simple UI for students & CRs**.
- Admin dashboard should display **all pending requests clearly**.
- Mobile responsiveness for **easy access from phones**.

---

4. System Design

4.1 System Architecture

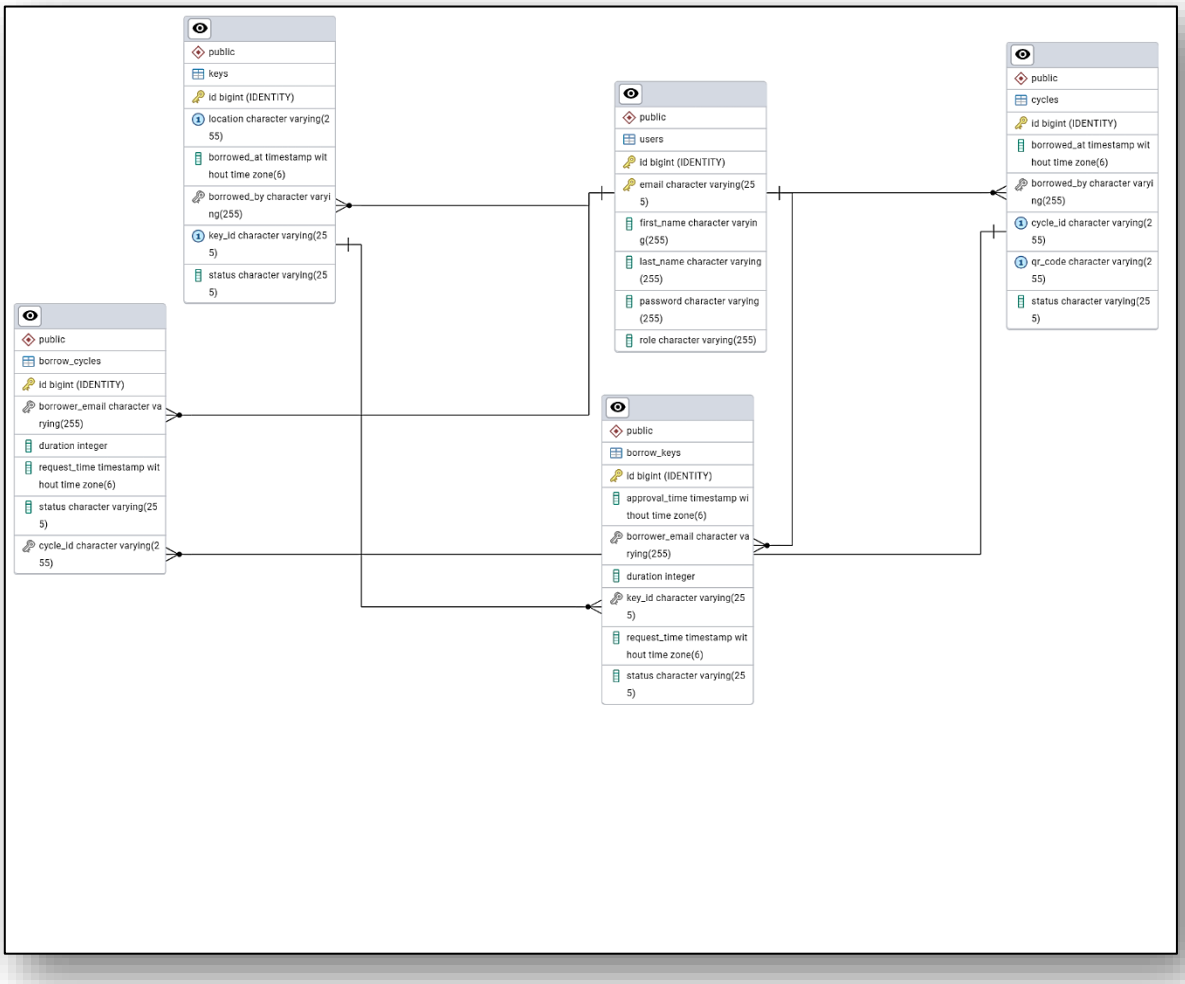
- **Frontend:** Angular (TypeScript, Bootstrap, QR Code Scanner)
- **Backend:** Spring Boot (REST APIs, JWT Authentication)
- **Database:** PostgreSQL (Stores users, keys, cycles, transactions)
- **Caching:** Redis (For real-time updates & quick retrieval)

4.2 Database Design

Tables

1. **Users** (id, name, email, role)

2. **Cycles** (id, cycleId, qrCode, status, borrowedBy)
3. **ClassroomKeys** (id, keyId, status, borrowedBy)
4. **KeyRequests** (id, keyId, borrowerEmail, duration, status)
5. **CycleRequests** (id, cycleId, borrowerEmail, status)



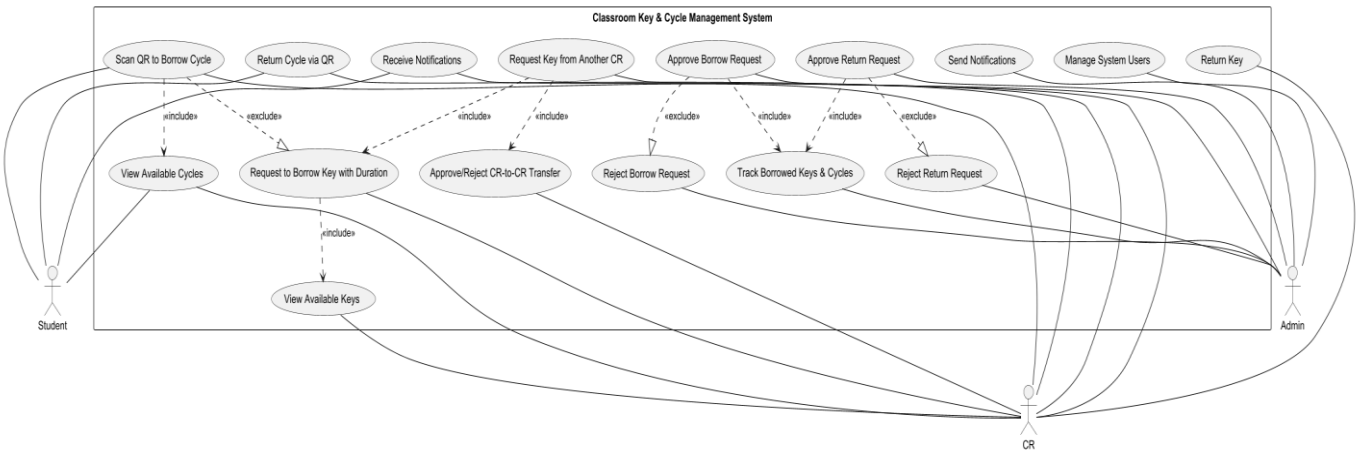
5. Assumptions & Constraints

- A user **can only borrow one cycle at a time**.
- CRs must **specify a time duration** when borrowing keys.
- The system must **automatically mark keys as "Available"** after the borrow duration expires.

6. Future Enhancements

- **Auto-reminders for cycle returns**.
- **Integration with college ID cards** for authentication.
- **Mobile App version** for quick QR scanning.

USE CASE DIAGRAM



Actors (Users):

1. **Student** – A user who borrows and returns cycles.
2. **CR (Class Representative)** – A user who manages borrow/return requests and transfers keys.
3. **Admin** – A superuser who oversees the entire system, including user management.

Use Cases and Relationships:

Each oval represents a **use case** (a system functionality), and the lines indicate which actor interacts with each use case.

1. Student Actions:

- **Scan QR to Borrow Cycle** – Student borrows a cycle by scanning a QR code.
- **Return Cycle via QR** – Student returns the borrowed cycle using a QR scan.
- **View Available Cycles** – Student can see which cycles are available for borrowing.
    - **Includes**: View Available Keys

- **Request to Borrow Key with Duration** – Student requests a key for a specific duration.
    - **Includes**: View Available Keys
    - **Excludes**: Reject Borrow Request (if the request is denied).
- **Receive Notifications** – Student gets updates on requests.

---

2. CR (Class Representative) Actions:

- **Approve/Reject Borrow Request** – CR can approve or reject a student's request to borrow a key.
    - **Excludes**: Reject Borrow Request
- **Track Borrowed Keys & Cycles** – CR keeps track of borrowed items.
    - **Includes**: Approve Borrow Request
    - **Includes**: Approve Return Request
    - **Excludes**: Reject Return Request
- **Approve/Reject CR-to-CR Transfer** – CR can approve or reject a request to transfer a key to another CR.
    - **Includes**: Request Key from Another CR
- **Return Key** – CR returns a key after use.

---

3. Admin Actions:

- **Manage System Users** – Admin oversees and manages all system users.
- **Send Notifications** – Admin sends notifications related to key and cycle management.
- **Approve Return Request** – Admin approves requests to return a borrowed cycle or key.
    - **Excludes**: Reject Return Request
- **Reject Return Request** – Admin denies a return request if conditions aren't met.

Relationships:

- **Include (<<include>>)** – One use case depends on another for completion.
- **Exclude (<<exclude>>)** – One use case prevents another from occurring under certain conditions.

Version History

Version	Date	Prepared By	Reviewed By	Description
1.0	18-Feb-2025	N. Vivek Teja	K. NAVANEETH	Initial Draft
1.0.1	19-Feb-2025	N. Teja	N. Vivek Teja	Added Use case Model and Entity Relationship Diagram