# 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._Introduction)
   * **Purpose**
   * **Scope**
   * **Definitions, Acronyms, and Abbreviations**
   * **Overview**
2. [**Functional Requirements**](#_2._Functional_Requirements)
   * **User Roles & Permissions**
   * **F1 - User Authentication**
   * **F2 – Cycle Borrowing and Returning**
   * **F3 – Classroom key Borrowing and Returning**
   * **F4 – Request Approval and Rejection**
   * **F5 – Transfer Classroom Keys among CRs**
   * **F6 - Notifications**
   * **F7 – User and Admin Dashboard**
   * **F8 – System User Management**
3. [**Non-Functional Requirements**](#_3._Non-Functional_Requirements)
   * **Performance Requirements**
   * **Security Requirements**
   * **Usability Requirements**
4. [**System Design**](#_4._System_Design)
   * **Technologies to be used**
   * **Database Design**
5. [**Assumptions & Constraints**](#_5._Assumptions_&)
6. [**Future Enhancements**](#_6._Future_Enhancements)
7. [**Use Case Diagram**](#_USE_CASE_DIAGRAM)

**8.** [**Class Diagram**](#_CLASS_DIAGRAM)

**9.** [**Version History**](#_Version_History)

# 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** in an educational institution. 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 |
| Admin | Approve/reject key borrowing & return requests |

**F1: Functional Requirements for Students**

**Description: Students can borrow and return cycles and receive notifications related to their activities.**

**F1.1 User Authentication**

* F1.1.1 Register using a unique email and password.
* F1.1.2 Login using credentials or NITC mail ID.
* F1.1.3 Reset password if required.

**F1.2 Cycle Borrowing and Returning**

* F1.2.1 Scan a QR code to borrow a cycle.
* F1.2.2 Verify cycle availability before borrowing.
* F1.2.3 Record borrow time and expected return time.
* F1.2.4 Return a cycle by scanning a QR code.
* F1.2.5 Update cycle status upon return.
* F1.2.6 Receive notifications for borrowing time limits.

**F1.3 Notifications and Tracking**

* F1.3.1 Receive notifications when a borrow request is approved or rejected.
* F1.3.2 Receive reminders before the return due time.
* F1.3.3 View borrowed cycle history.

**F2: Functional Requirements for Class Representatives (CRs)**

**Description: CRs can manage classroom key borrowing, approvals, and transfers.**

**F2.1 Key Borrowing and Returning**

* F2.1.1 Request a key for a specified duration.
* F2.1.2 Verify key availability before borrowing.
* F2.1.3 Submit a return request for keys.
* F2.1.4 View key borrowing history.

**F2.2 Request Approval and Rejection**

* F2.2.1 Approve or reject classroom key borrow requests.
* F2.2.2 Notify students of approval or rejection decisions.

**F2.3 Key Transfers Between CRs**

* + F2.3.1 Request a key transfer from another CR.
  + F2.3.2 Approve or reject key transfer requests.
  + F2.3.3 Update key ownership status upon approval.

**F2.4 Notifications and Tracking**

* + F2.4.1 Receive notifications for key borrow requests.
  + F2.4.2 Receive reminders for due returns.
  + F2.4.3 Track borrowed classroom keys.

**F3: Functional Requirements for Admins**

**Description: Admins oversee system users, roles, and tracking borrowed assets.**

**F3.1 User Authentication & Management**

* F3.1.1 Enforce role-based access (Student, CR, Admin).
* F3.1.2 Support multi-factor authentication if required.
* F3.1.3 Manage users (add, update, remove).
* F3.1.4 Assign and update roles.

**F3.2 System Monitoring and Reports**

* F3.2.1 Maintain a list of currently borrowed cycles and keys**.**
* F3.2.2 Display borrowing history for each user.
* F3.2.3 Allow filtering and searching of borrowing records.
* F3.2.4 Generate reports on borrowed items.

**F3.3 Notifications and Oversight**

* F3.3.1 Receive notifications when borrow/return requests are submitted.
* F3.3.2 Configure notification preferences for users.

# 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.
* **Usage encryption to store sensitive user data.**
* **Enforce role-based access control (RBAC).**
* **Only Admins can modify roles.**

**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

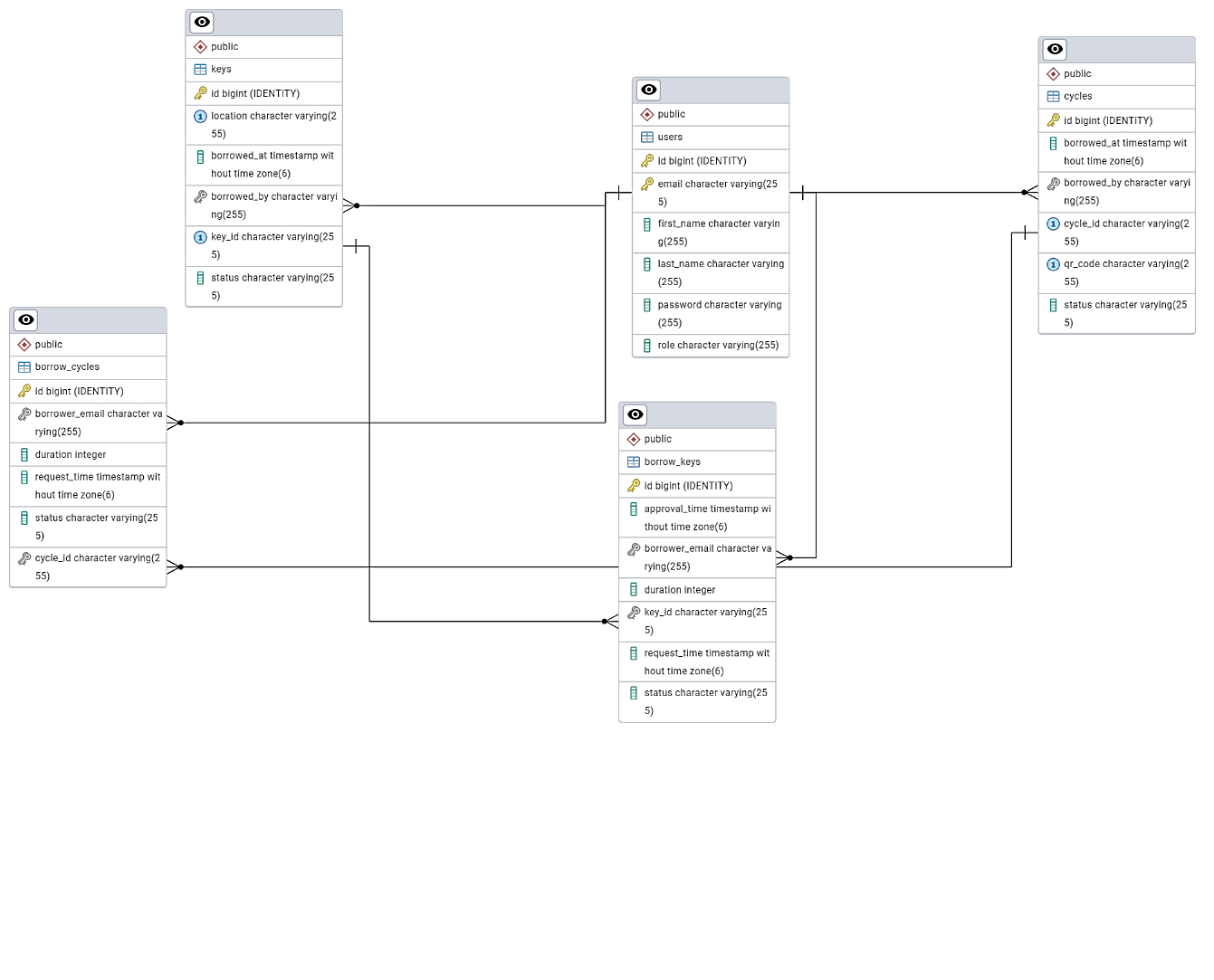
**4.1 Technologies to be used**

* **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,email,firstName,lastName, role,EncryptedPassword)
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)



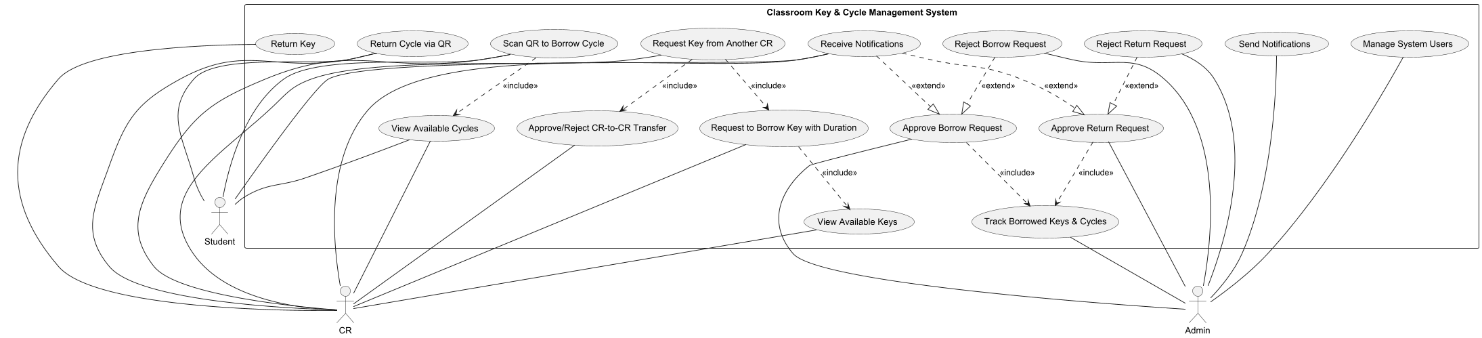
# 5. Assumptions & Constraints

* 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.

# 7.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 (User) Actions**

**A student interacts with the system to borrow and return cycles and receive notifications.**

**Cycle Borrowing and Returning**

* **Scan QR to Borrow Cycle → The student can scan a QR code on the cycle to borrow it.**
* **View Available Cycles → The student can check available cycles before borrowing.**
* **Return Cycle via QR → The student can scan a QR code to return the borrowed cycle.**
* **Receive Notifications → The system sends alerts for borrowing approvals, due return reminders, and overdue cycles.**

**2. Class Representative (CR) Actions**

**A CR (Class Representative) has all the actions of a student and additional privileges to manage classroom keys.**

**Cycle Borrowing and Returning *(Same as Students)***

* **View Available Cycles**
* **Scan QR to Borrow Cycle**
* **Return Cycle via QR**

**Key Borrowing and Returning**

* **View Available Keys → The CR can check which classroom keys are available.**
* **Request to Borrow Key with Duration → The CR can request a key for a specified time.**
* **Return Key → The CR can return a key after use.**
* **Receive Notifications → The CR is notified of key borrow approvals, return reminders, and overdue keys.**

**Request Transfers Between CRs**

* **Request Key from Another CR → A CR can request a key transfer from another CR.**
* **Approve/Reject CR-to-CR Transfer → The recipient CR can approve or reject a key transfer request.**

**3. Admin Actions**

**An Admin has full control over the system, including managing users and tracking borrowed items.**

**Request Approval and Rejection**

* **Approve Borrow Request → Admin can approve requests for keys and cycles.**
* **Reject Borrow Request → Admin can reject borrow requests if needed.**
* **Approve Return Request → Admin can approve cycle or key returns.**
* **Reject Return Request → Admin can reject return requests if conditions are not met.**

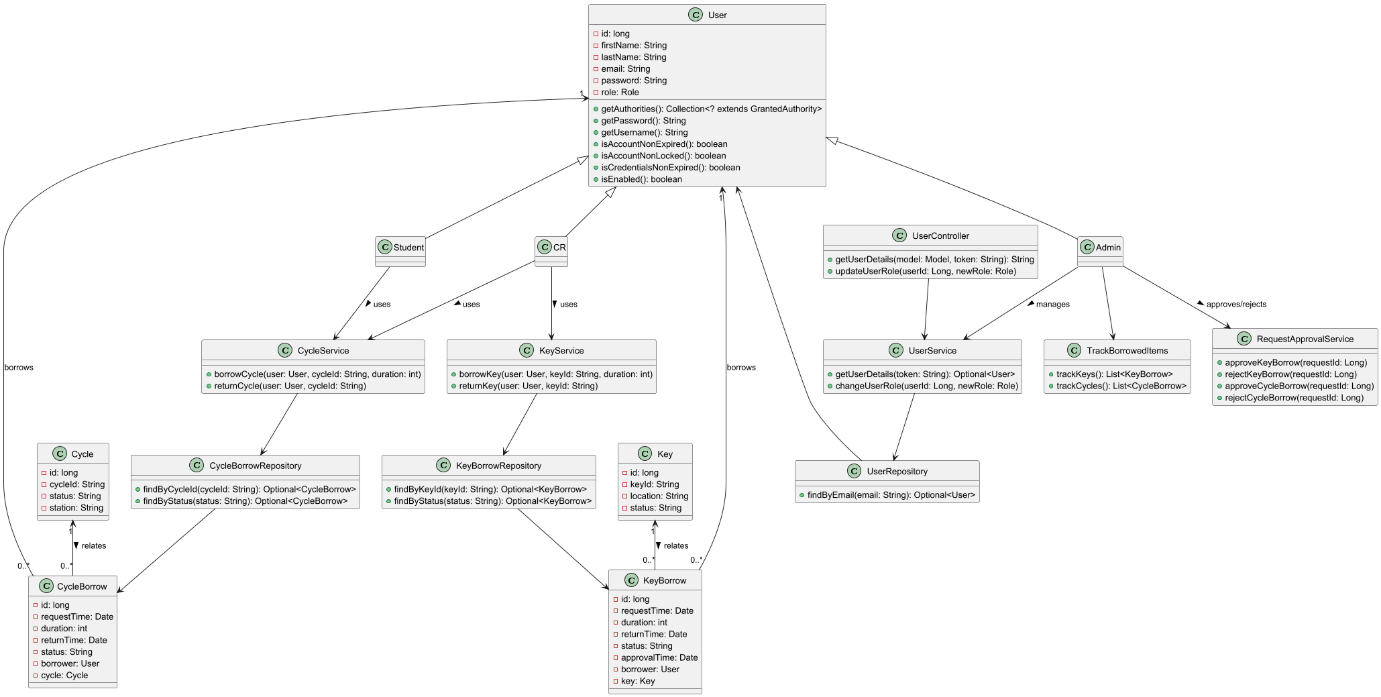
**Tracking and Notifications**

* **Track Borrowed Keys & Cycles → Admin can view all active borrow records.**
* **Send Notifications → The system sends alerts for approvals, due returns, and user updates.**

**System Management**

* **Manage System Users → Admin can add, update, and remove users, and assign roles (Student, CR, Admin).**

# CLASS DIAGRAM



**Class Overview:**

**1. User Hierarchy**

* User: The base class representing all system users, including students, class representatives (CRs), and admins.
* Student: A subclass of User that can borrow and return cycles using CycleService.
* CR (Class Representative): A subclass of User that can borrow and return both keys (KeyService) and cycles (CycleService).
* Admin: A subclass of User that manages users, approves/rejects borrow requests, and tracks borrowed items.

**2. Borrowing Entities**

* KeyBorrow: Represents a key borrowing request with attributes such as keyId, requestTime, duration, and status.
* CycleBorrow: Represents a cycle borrowing request with attributes such as cycleId, requestTime, duration, and status**.**

**3. Services and Repositories**

* KeyService: Manages key borrowing and returning.
* CycleService: Manages cycle borrowing and returning.
* KeyBorrowRepository: Handles database operations for key borrowing.
* CycleBorrowRepository: Handles database operations for cycle borrowing.
* UserRepository: Manages user authentication and retrieval.

**4. Request Approval & Tracking**

* RequestApprovalService: Allows admins to approve or reject borrow requests for keys and cycles.
* TrackBorrowedItems: Enables admins to track currently borrowed cycles and keys.

**5. Controllers & User Management**

* UserController: Handles user-related requests, including authentication and role management.
* UserService: Provides business logic for user authentication and role changes.

# 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 |
| 1.1 | 26-Feb-2025 | N. Teja | N. Vivek Teja | Added Class Diagram and Made changes in Use case Model and Functional Requirements according to the comments made by evaluator |