CLASSROOM KEY AND CYCLE KEY MANAGEMENT SYSTEM

Software Requirements Specification

GROUP – 4

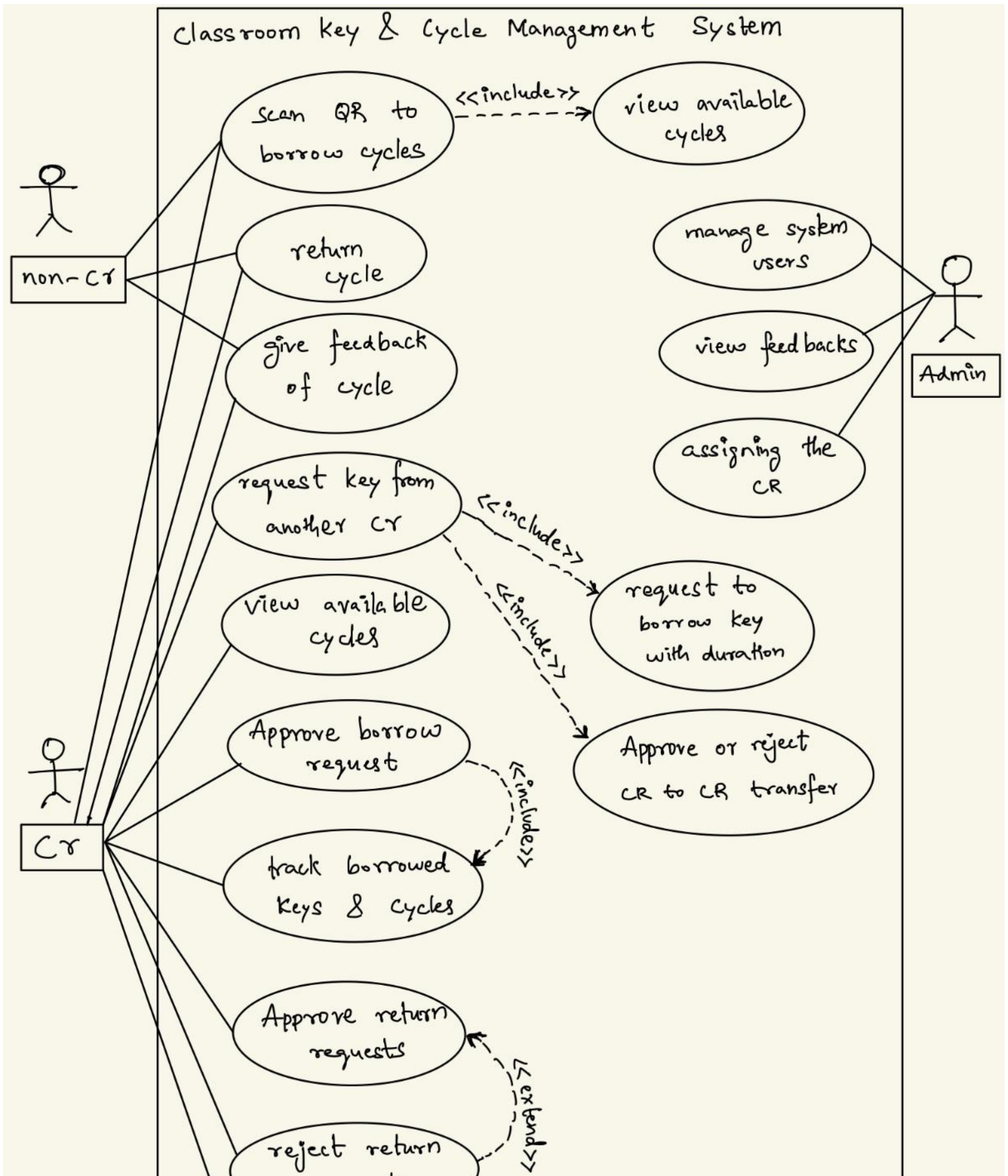
Group Members	Roll Number
Gedala Vamsi Vihari	B220286CS
Chilekampalli Reddy Sekhar Reddy	B220806CS
Meragala Sujanesh Nandan	B220389CS

Revision History

Version	Date	Description Of Changes
1.0	19-02-2025	SRS DOCUMENT

CONTENT

USE CASE DIAGRAM



Functional Requirements

User Requirements

1. Class Representatives (CRs)

Role-Based Login

- CRs can log in using Google authentication.

Borrow Classroom Keys

- Can check classroom availability before borrowing.
- If available, they can book the key.
- If a key is in use, they can request it from another CR.

Manage Classroom Key Requests

- Can view, send, and track requests for classroom keys.
- Can check pending and fulfilled requests.

Borrow a Bicycle

- Can scan a QR code at designated locations to borrow a bicycle.
- Can check bicycle availability before borrowing.

QR Code-Based Borrowing System

- The system shall support QR code scanning for borrowing bicycles.
- Users can scan the QR code at designated locations to borrow bicycles.

Submit Classroom Key

- Must return the borrowed classroom key after use.
- The system updates the key's status and removes it from their holding list.

Submit Bicycle with Feedback

- When returning a bicycle, CRs must submit feedback.
- The system marks the bicycle as available for others.

View Borrowing History

- Can track past borrowings, including:
  - Classroom keys.
  - Bicycles (with feedback details).
  - Dates and times of borrowing and return.

---

2. Non-CR Students

Role-Based Login

- Non-CR students can log in using Google authentication.

Borrow a Bicycle

- Can scan a QR code at designated locations to borrow a bicycle.
- Can check bicycle availability before borrowing.

QR Code-Based Borrowing System

- The system shall support QR code scanning for borrowing bicycles.
- Users can scan the QR code at designated locations to borrow bicycles.

Submit Bicycle with Feedback

- When returning a bicycle, students must submit feedback.
- The system marks the bicycle as available for others.

View Borrowing History

- Can track past borrowings, including:
  - Bicycles (with feedback details).
  - Dates and times of borrowing and return.

---

3. Administrator

Role-Based Login

- Admins log in using credentials (not Google authentication).

Manage Users

- Can assign roles (CR or Non-CR) to students.
- Can update user details if needed.
- Can delete users from the system if necessary.

Manage Classroom Keys and Bicycles

- Can maintain an updated list of available classroom keys and bicycles.
- Can update status when a key or bicycle is borrowed or returned.

View Borrowings and Feedback

- Can track all borrowed keys and bicycles.
- Can access student feedback on bicycles.

NON -FUNCTIONAL REQUIREMENTS

1. Performance Requirements

- The system should efficiently manage at least 50 concurrent users, with no significant drop in performance.
- The CR and non-CR dashboards should load within 3-5 seconds under normal conditions.
- The booking process (cycle or classroom) should complete within 2-3 seconds after submission.

2. Scalability Requirements

- Initially designed for up to few users, but should support easy upgrades for a larger user base in the future.
- API calls should be optimized to minimize unnecessary database queries and data fetching.

3. Security Requirements

- Use Spring Security for login authentication with role-based access (CR and non-CR).
- Session management should be done using JWT (JSON Web Token).
- Passwords must be securely hashed (using BCrypt or an equivalent) before storage.
- Sensitive data, such as booking history and user credentials, should be encrypted or anonymized where possible.

4. Usability Requirements

- The UI (including dashboards, booking systems, etc.) should be fully responsive and optimized for laptops, tablets, and mobile devices.
- Basic help documentation or tooltips should guide users through key functions like booking cycles or classrooms.

5. Maintainability Requirements

- The backend should follow a modular design, with clear separation of controllers, services, and repositories.
- Proper documentation and comments should be maintained for all modules, especially those dealing with bookings, authentication, and cycle/classroom management.

6. Compliance Requirements

- The system must comply with basic data privacy rules, ensuring no unauthorized access to sensitive information such as user login details and booking history.

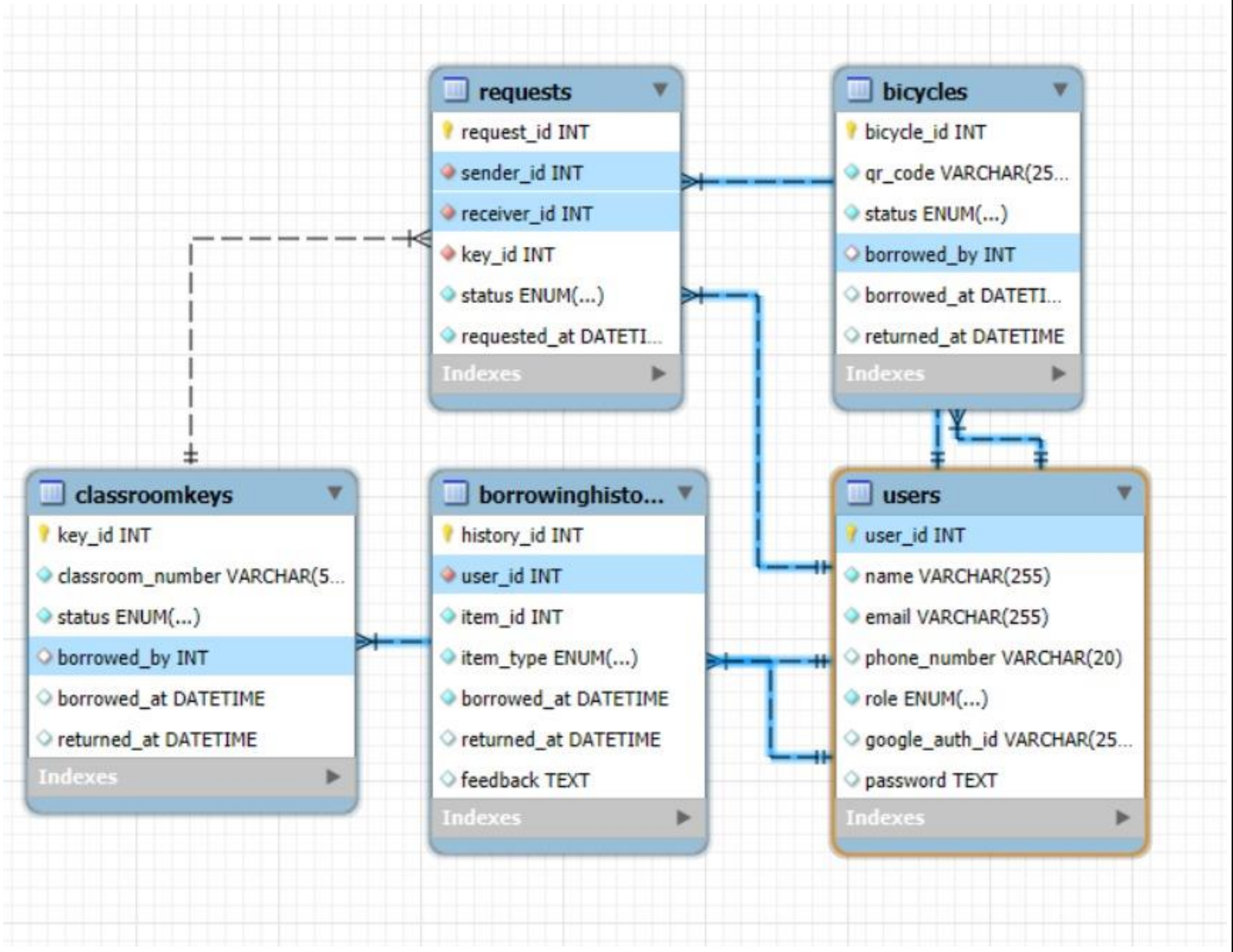
7. **Integration Requirements**

- Future integration with college databases (student and classroom details) should be possible.
- For now, the system can use a dummy student dataset to simulate user interactions.

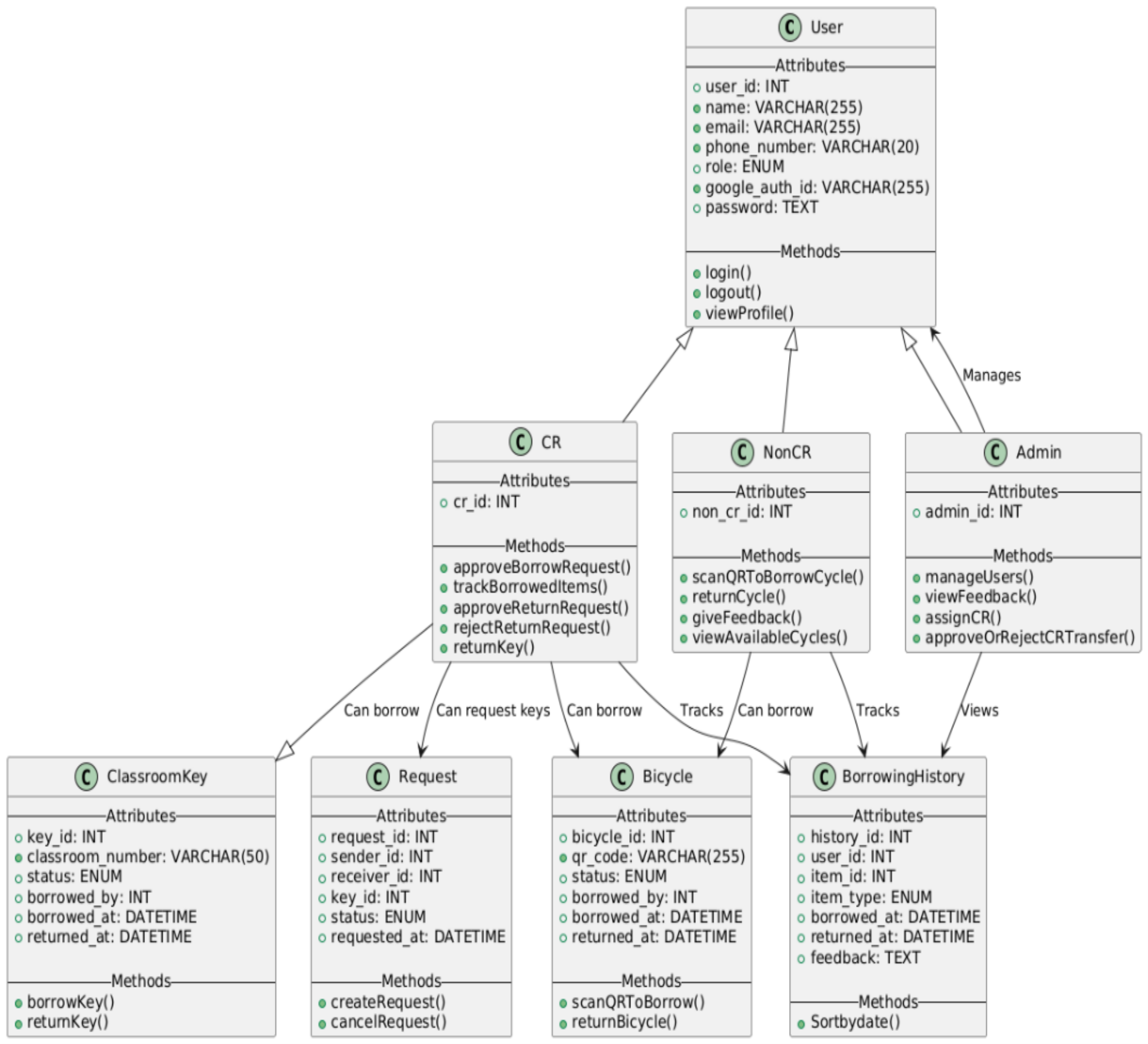
8. **Portability Requirements**

- The backend should be able to run on both Windows and Linux servers.
- The frontend should be compatible with common browsers like Chrome, Firefox, and Edge.

DATABASE DESIGN



Class Diagram


User
Attributes
user_id: INT
name: VARCHAR(255)
email: VARCHAR(255)
phone_number: VARCHAR(20)
role: ENUM
google_auth_id: VARCHAR(255)
password: TEXT
Methods
login()
logout()
viewProfile()
Manages
CR
Attributes
cr_id: INT
Methods
approveBorrowRequest()
trackBorrowedItems()
approveReturnRequest()
rejectReturnRequest()
returnKey()
NonCR
Attributes
non_cr_id: INT
Methods
scanQRToBorrowCycle()
returnCycle()
giveFeedback()
viewAvailableCycles()
Admin
Attributes
admin_id: INT
Methods
manageUsers()
viewFeedback()
assignCR()
approveOrRejectCRTransfer()
Can borrow
Can request keys
Can borrow
Tracks
Can borrow
Tracks
Views
ClassroomKey
Attributes
key_id: INT
classroom_number: VARCHAR(50)
status: ENUM
borrowed_by: INT
borrowed_at: DATETIME
returned_at: DATETIME
Methods
borrowKey()
returnKey()
Request
Attributes
request_id: INT
sender_id: INT
receiver_id: INT
key_id: INT
status: ENUM
requested_at: DATETIME
Methods
createRequest()
cancelRequest()
Bicycle
Attributes
bicycle_id: INT
qr_code: VARCHAR(255)
status: ENUM
borrowed_by: INT
borrowed_at: DATETIME
returned_at: DATETIME
Methods
scanQRToBorrow()
returnBicycle()
BorrowingHistory
Attributes
history_id: INT
user_id: INT
item_id: INT
item_type: ENUM
borrowed_at: DATETIME
returned_at: DATETIME
feedback: TEXT
Methods
Sortbydate()