CEBU INSTITUTE OF TECHNOLOGY – UNIVERSITY
COLLEGE OF COMPUTER STUDIES

Software Requirements Specifications

for
STUDENT CLEARANCE SYSTEM (SCS)

Change History

Version	Date	Description	Author
1.0	June 19, 2025	Initial Draft	Francis Garry Nillama
1.1	TBD	Final Review & Update	Team SCS

Table of Contents

---

1. Introduction

1.1. Purpose

The purpose of this document is to specify the requirements of the **Student Clearance System (SCS)**—a centralized web application that facilitates student clearance workflows in a digital, efficient, and trackable format. This document serves as a reference for the development team, project stakeholders, and testers.

1.2. Scope

The SCS is a full-stack web application built using **ReactJS**, **Spring Boot**, and **MySQL**. It supports multiple user roles and allows clearance tasks to be submitted, approved, rejected, or commented on by various departments. It features OAuth authentication, dashboard views, and real-time notification mechanisms.

1.3. Definitions, Acronyms and Abbreviations

Term	Definition
SCS	Student Clearance System
OAuth	Open Authorization
API	Application Programming Interface

Term	Definition
UI	User Interface
ERD	Entity-Relationship Diagram
CRUD	Create, Read, Update, Delete

1.4. References

- Student Clearance System Project Charter (CIT-U, 2025)
- Google OAuth Documentation: https://developers.google.com/identity
- Swagger Documentation: https://swagger.io/
- Spring Boot Reference Guide
- MySQL Documentation: https://dev.mysql.com/doc/

---

2. Overall Description

2.1. Product Perspective

The system replaces the traditional paper-based clearance system with a secure digital solution. It integrates with Google OAuth for authentication and offers role-based dashboards, REST APIs, and visual clearance progress indicators.

2.2. User Characteristics

- **Students**: Submit clearance tasks and track progress.
- **Department Heads**: Approve, reject, and comment on tasks.
- **Registrar**: Monitors overall student clearance.
- **Admin (optional)**: Configure clearance flows per department.

2.4. Constraints

- Development duration: 3 weeks only.
- Deployment limited to web (no mobile app).
- Requires stable internet connection.
- Google OAuth support is required.
- University server limitations may affect hosting.
- Budget does not include paid third-party services.

2.5. Assumptions and Dependencies

- Departments will actively test and provide feedback.
- Students and faculty have Gmail accounts.
- The university allows OAuth access and necessary firewall permissions.
- End-users are familiar with basic web usage.

---

3. Specific Requirements

3.1. External Interface Requirements

3.1.1. Hardware Interfaces

- Web client supported on desktops/laptops with internet access.
- Backend hosted on a university/staging server.
- No special hardware peripherals required.

3.1.2. Software Interfaces

- Google OAuth 2.0
- Spring Boot backend
- ReactJS frontend
- MySQL database
- Swagger or Postman for API documentation

3.1.3. Communications Interfaces

- HTTPS communication for secure API requests
- Email server for email notifications (optional SMTP)

3.2. Functional Requirements

Module 1: Authentication and User Roles

1.1 Login with Google OAuth

- Use Case Diagram
- Use Case Description
- Activity Diagram
- Wireframe

1.2 Role Assignment (Student, Department Head, Registrar)

- Use Case Diagram
- Use Case Description
- Activity Diagram
- Wireframe

Module 2: Clearance Workflow Management

2.1 View Clearance Tasks (Student)

- Use Case Diagram
- Use Case Description
- Activity Diagram
- Wireframe

2.2 Approve/Reject/Comment on Tasks (Department Head)

- Use Case Diagram
- Use Case Description
- Activity Diagram
- Wireframe

2.3 Track Clearance Status (Registrar)

- Use Case Diagram
- Use Case Description
- Activity Diagram
- Wireframe

Module 3: Notifications and Comments

3.1 In-app and Email Notifications

- Use Case Diagram
- Use Case Description
- Activity Diagram
- Wireframe

3.2 Commenting on Clearance Tasks

- Use Case Diagram
- Use Case Description
- Activity Diagram
- Wireframe

Module 4: Dashboards and Tracking

4.1 Student Dashboard with Progress View

- Use Case Diagram
- Use Case Description
- Activity Diagram
- Wireframe

4.2 Registrar Overview Dashboard

- Use Case Diagram
- Use Case Description
- Activity Diagram
- Wireframe

3.4 Non-functional Requirements

Performance

- The system must support at least 200 concurrent users.
- All API requests should respond within 2 seconds under normal load.

Security

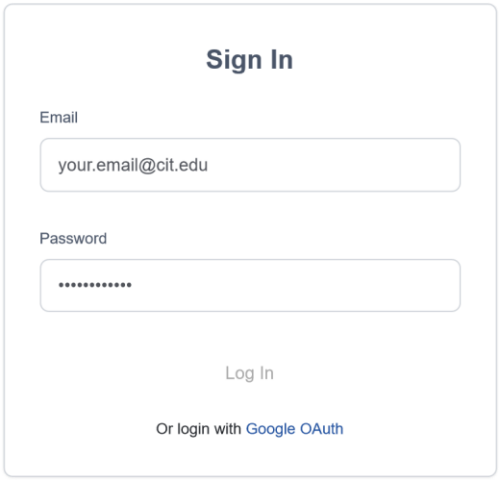
- Google OAuth will handle authentication.
- HTTPS will secure all data in transit.
- Access to dashboards and actions is role-restricted.

Reliability

- The system should be operational 99% during academic periods.
- All data changes should be logged and traceable.
- Backups will be taken weekly.

Student Clearance System

1. Login Page



2. Student Dashboard

Student Info: [Name], [ID]
Graduation Year: [Year]

Overall Status
In Progress
Updated: [Date/Time]

Notifications
- New notification 1
- New notification 2

Clearance Progress Table

Department	Status	Comments	Action
Dept A	Approved	Comment text here.	View
Dept B	Pending	Comment text here.	View

Submit Clearance Request

Select type:

Option 1

Upload docs:

Choose File No file chosen

Submit Request

3. Department Head Dashboard

Your Dept: [Name]

Overview of departmental clearances.

Dept Notifications

- New request for review.
- Reminder for pending tasks.

Pending Requests for Review

Student Name	Student ID	Date	Status	Action
Student A	ID123	2025-06-19	Pending	Approve Reject Comment
Student B	ID456	2025-06-18	Pending	Approve Reject Comment

Processed Requests

List of approved/rejected requests.

4. Registrar Dashboard

Registrar Overview

Monitor all clearance processes.

System Alerts

- No new alerts.

Overall Student Clearance Status

Student Name	Student ID	Overall Status	Progress	Action
Student C	ID789	Completed	100%	Generate
Student D	ID012	In Progress	50%	View

Audit Logs

Searchable list of system activities.

Generate Reports

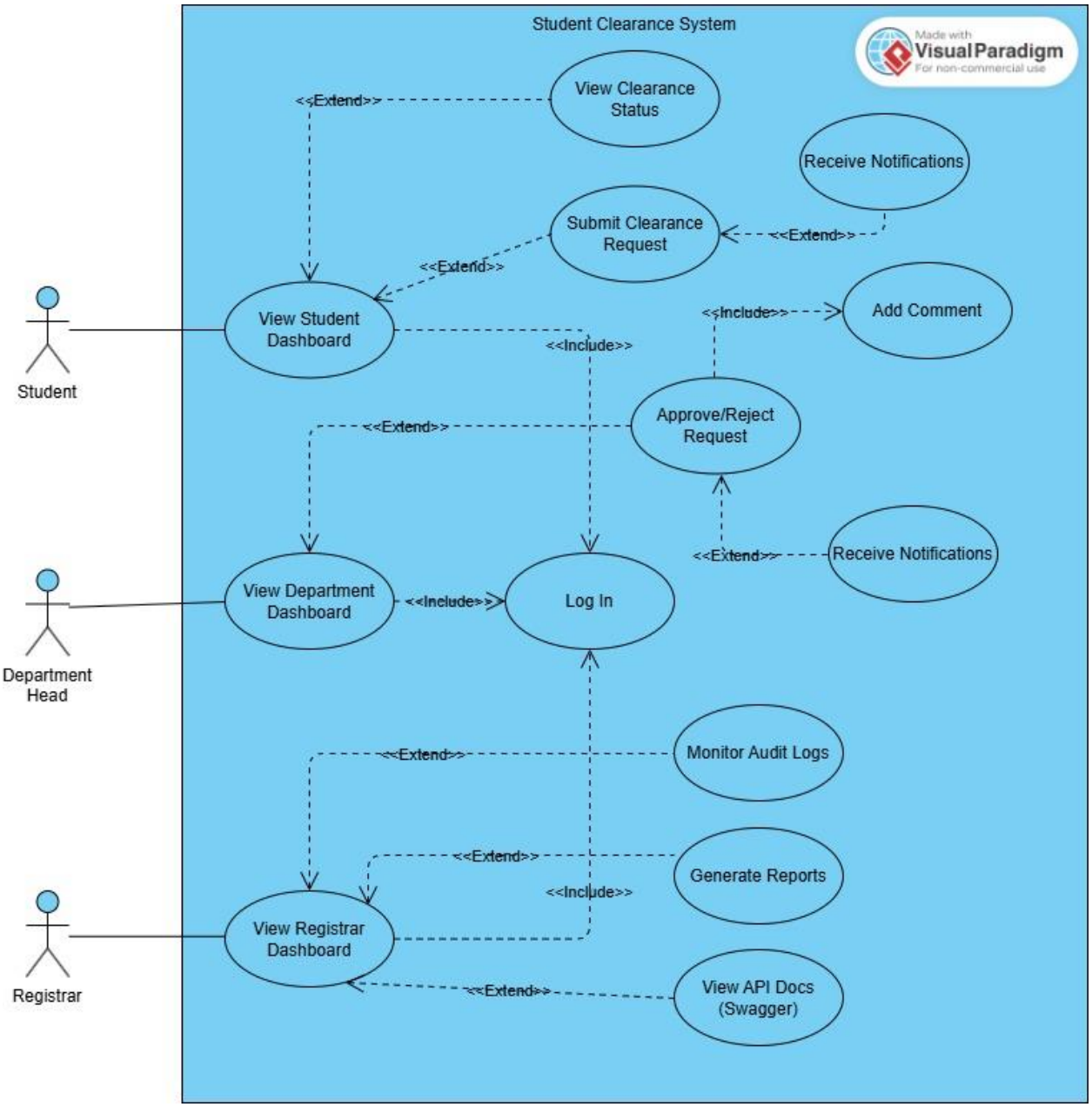
Report generation options.

Generate Report

API Documentation

Link to API docs.

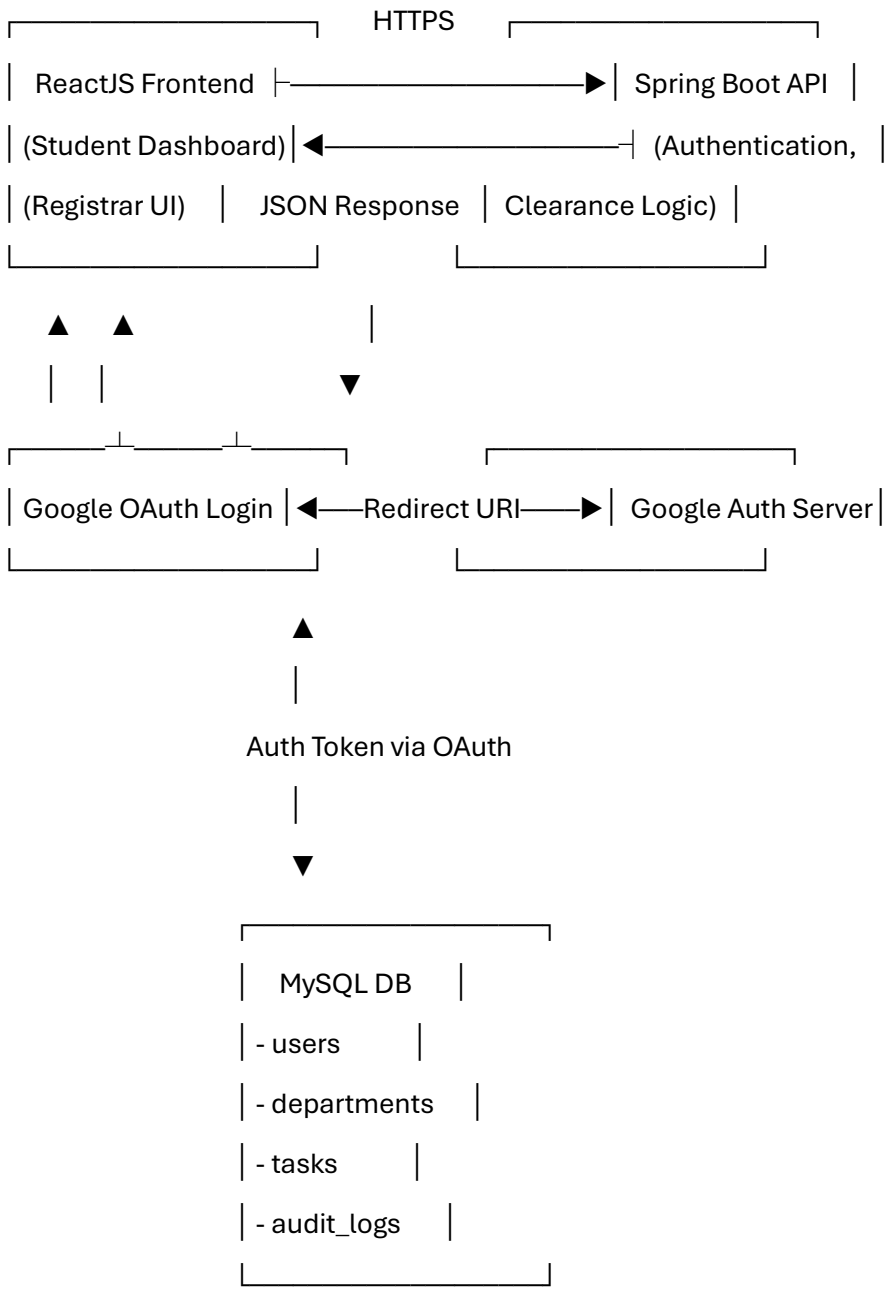
Go to API Docs


Student Clearance System
Made with VisualParadigm For non-commercial use
View Clearance Status
Receive Notifications
Submit Clearance Request
<<Extend>>
<<Extend>>
<<Extend>>
View Student Dashboard
<<Include>>
Add Comment
<<Include>>
Approve/Reject Request
<<Extend>>
<<Extend>>
Receive Notifications
Student
View Department Dashboard
<<Include>>
Log In
Department Head
Monitor Audit Logs
<<Extend>>
<<Extend>>
Generate Reports
<<Include>>
View Registrar Dashboard
<<Extend>>
View API Docs (Swagger)
Registrar

System Architecture Diagram (Client-Server, API Flows)

plaintext

CopyEdit



ERD (Entity Relationship Diagram)

plaintext

CopyEdit

[User]

- user_id (PK)
- google_id
- email
- full_name
- role (Student, DeptHead, Registrar)
- profile_url

[Department]

- department_id (PK)
- name

[ClearanceTask]

- task_id (PK)
- user_id (FK -> User)
- department_id (FK -> Department)
- status (Pending, Approved, Rejected)
- comment
- updated_by (FK -> User)
- updated_at

[AuditLog]

- log_id (PK)
- user_id (FK -> User)
- action (string)
- target (string)
- timestamp

[Notification]

- notification_id (PK)

- user_id (FK -> User)

- message

- seen (bool)

- created_at

[Comment]

- comment_id (PK)

- task_id (FK -> ClearanceTask)

- sender_id (FK -> User)

- message

- created_at

API Design Document

Endpoint: /api/login

Name	**User Login**
Request	POST
URL	/api/login
Headers	Content-Type: application/json
Format	JSON

JSON Request Parameters:

Field Name	**Type**	**Required**	**Constraints**	**Description**
email	String	Yes	Encrypted	User's email
password	String	Yes	Encrypted	Login password

Sample Request:

json

CopyEdit

{

```
 "email": "user@example.com",
 "password": "encryptedpassword"
}
```

JSON Response Parameters:

Field Name	Type	Required	Description
status	Boolean	Yes	1: Success, 0: Failed
message	String	Yes	Status description
serverToken	String	Yes	JWT or Session Token
results	JSON Object	Yes	User Profile Object

Sample Success Response:

json

CopyEdit

```
{
 "status": 1,
 "message": "Login successful",
 "serverToken": "ADFADSFA123123",
 "results": {
  "userId": 1,
  "email": "user@example.com",
  "role": "Student",
  "profileUrl": "/images/user1.png"
 }
}
```

Endpoint: /api/clearance-tasks

Name	Get All Clearance Tasks (per student)
Request	GET

Name **Get All Clearance Tasks (per student)**

URL /api/clearance-tasks

Headers Authorization: Bearer <token>

Format JSON

Response Parameters:

Field Name	Type	Required	Description
status	Boolean	Yes	1 = success, 0 = fail
message	String	Yes	Text description
results	Array of Objects	Yes	List of clearance task items

Sample Response:

json

CopyEdit


```
{
 "status": 1,
 "message": "Tasks fetched successfully",
 "results": [
  {
   "taskId": 101,
   "department": "Library",
   "status": "Pending",
   "comment": null,
   "updatedAt": "2025-06-24T09:15:00"
  }
 ]
}
```