SkillVine

An online platform for managing, evaluating & grading activity points

Team Members and Project Guide

- Alvin Varghese (KTE20CS008)
- Anson Anthrayose Thomas (KTE20CS013)
- Sreerag M (KTE20CS056)
- Vignesh R Pillai (KTE20CS068)

- Project Guide: Prof. Aswathy B

Introduction

Modules

- Front End
- Connectivity Algorithms
- Database Design

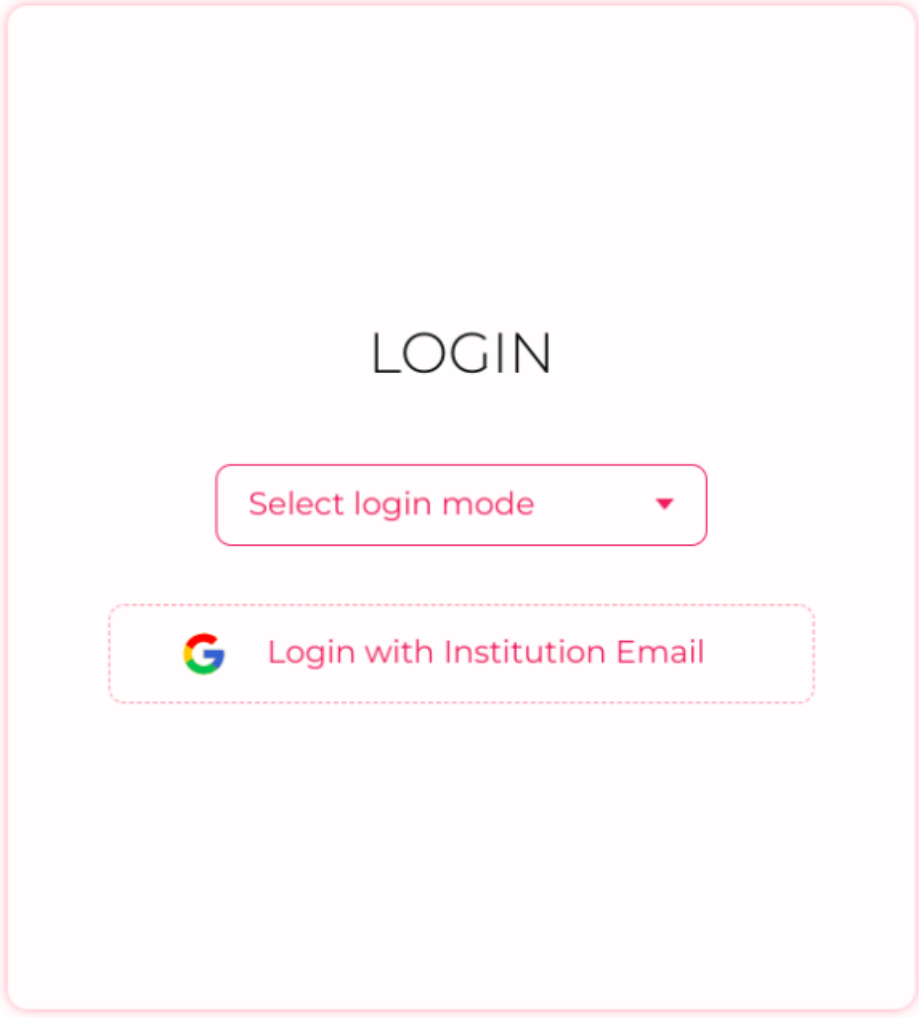
Module-1

User Frontend

- Major components -
  - Login.
  - Register
  - Profile
  - Dashboard.
  - Add Certificate.
  - View certificate(teacher and student).
  - Edit certificate
  - Batch select

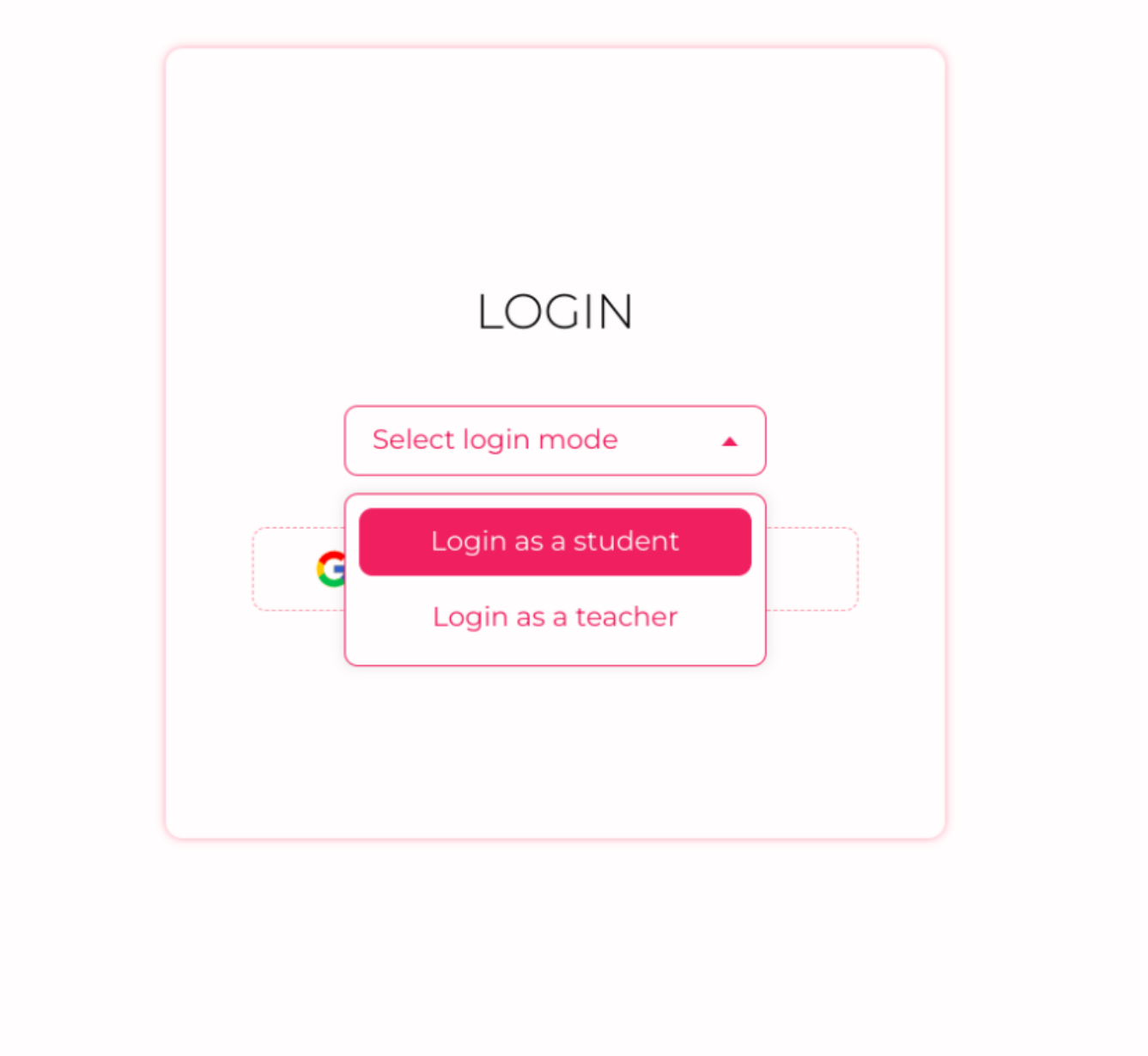
Login

Login overview



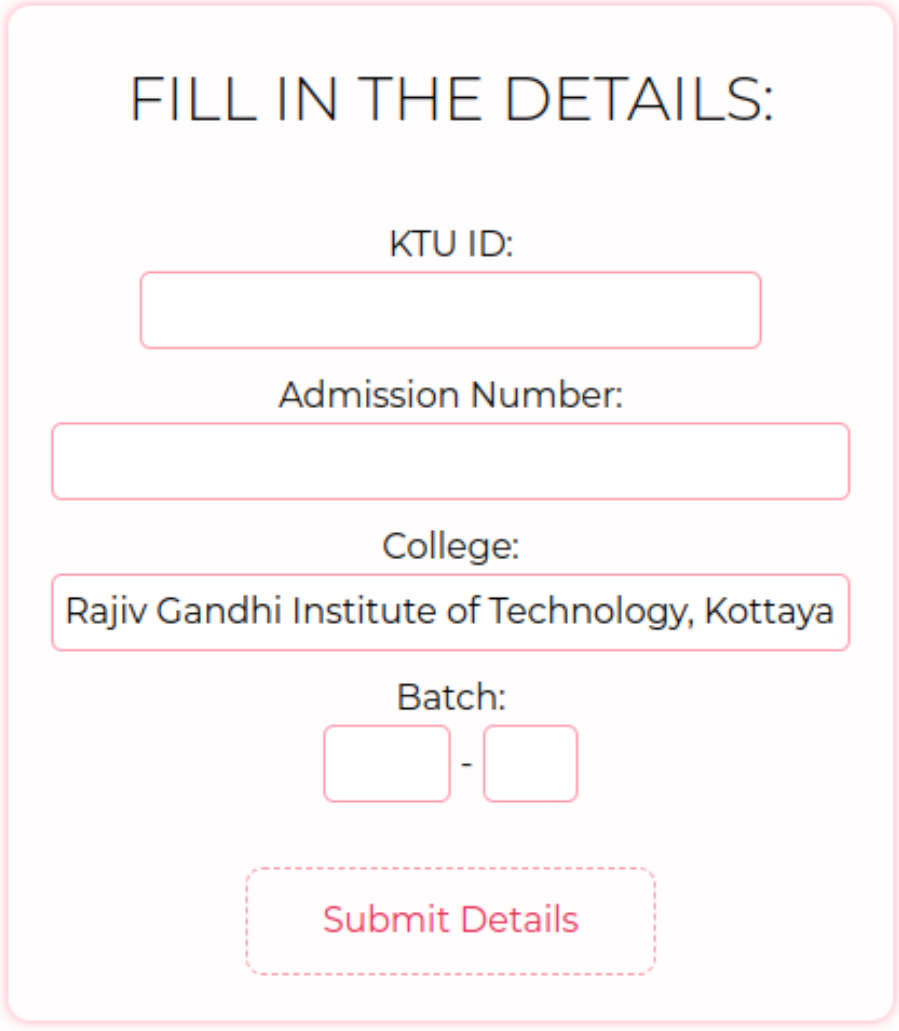
Login

User select

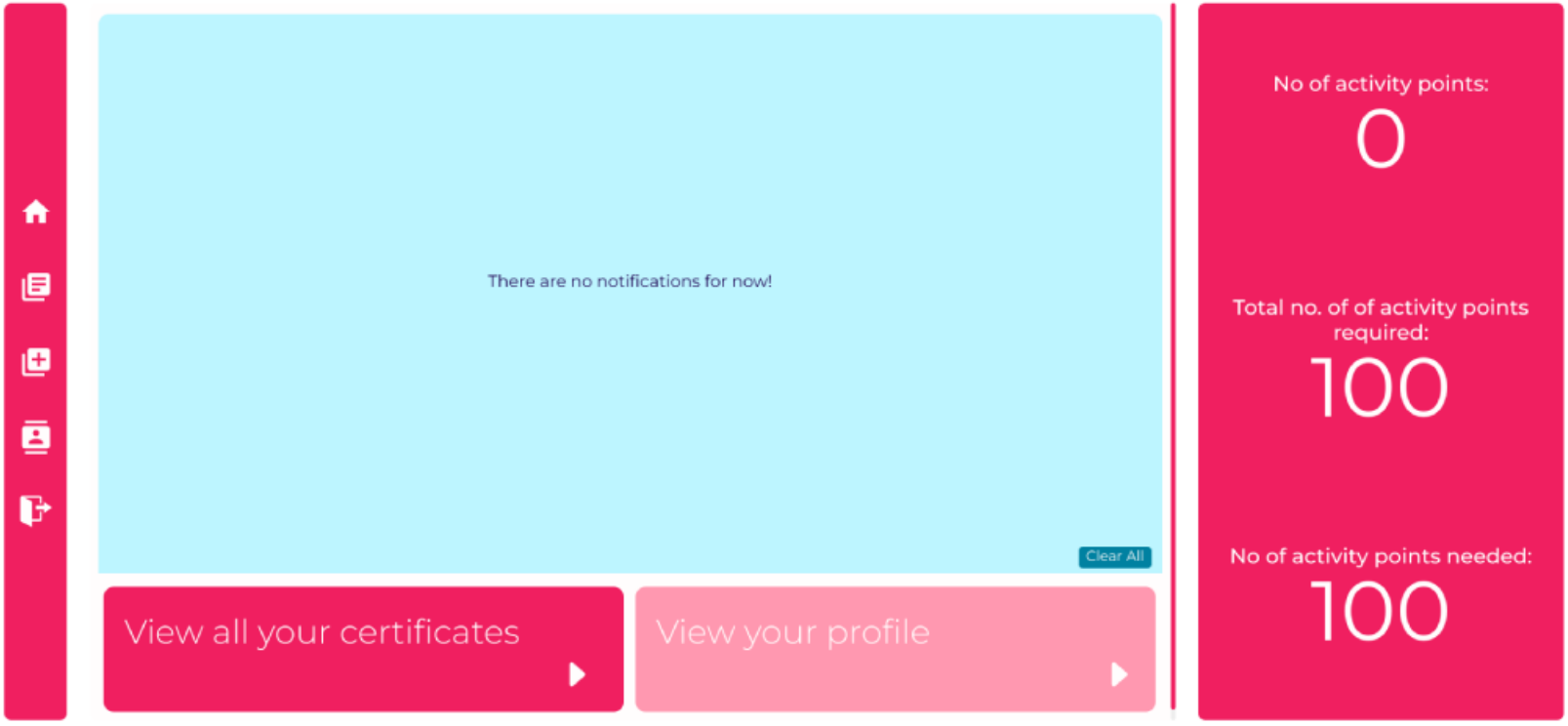


Register

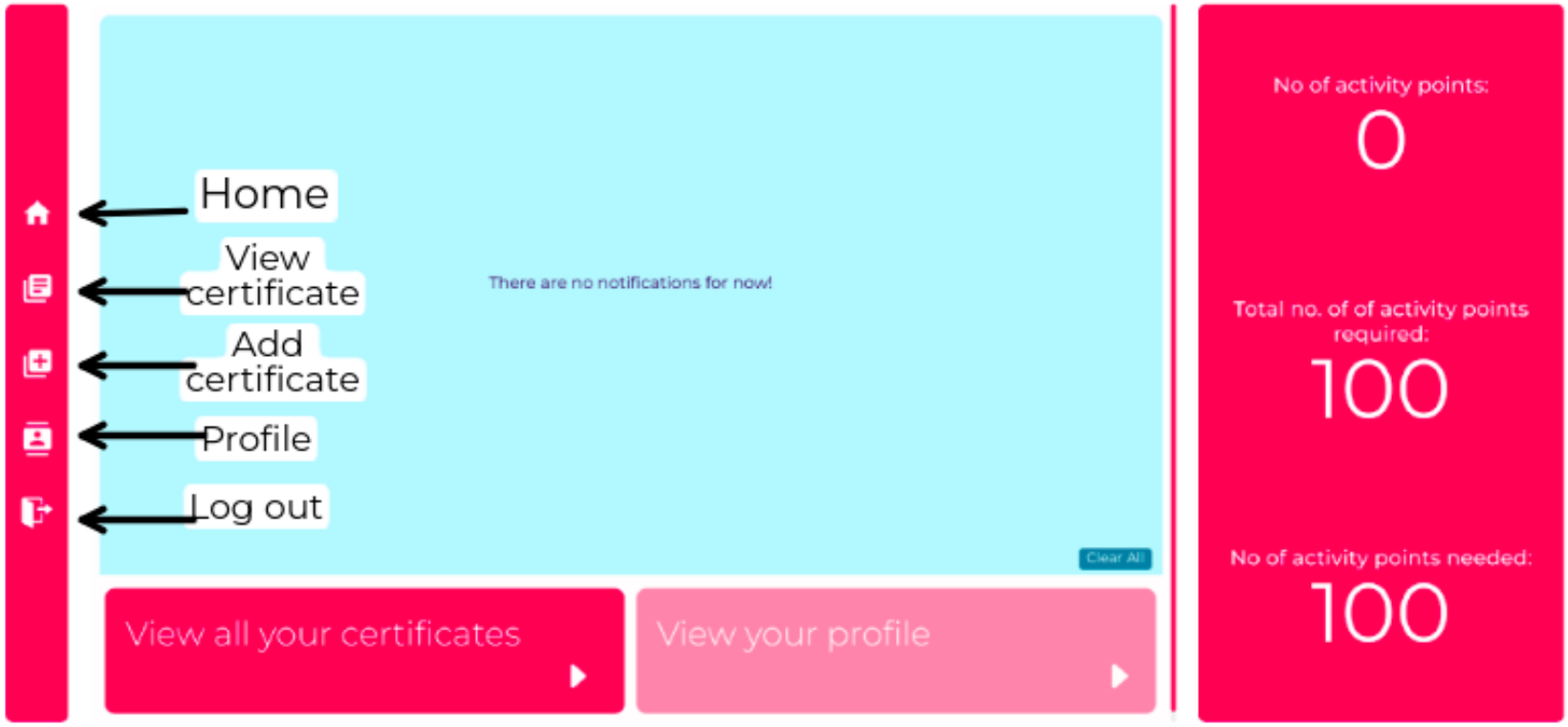
Student side



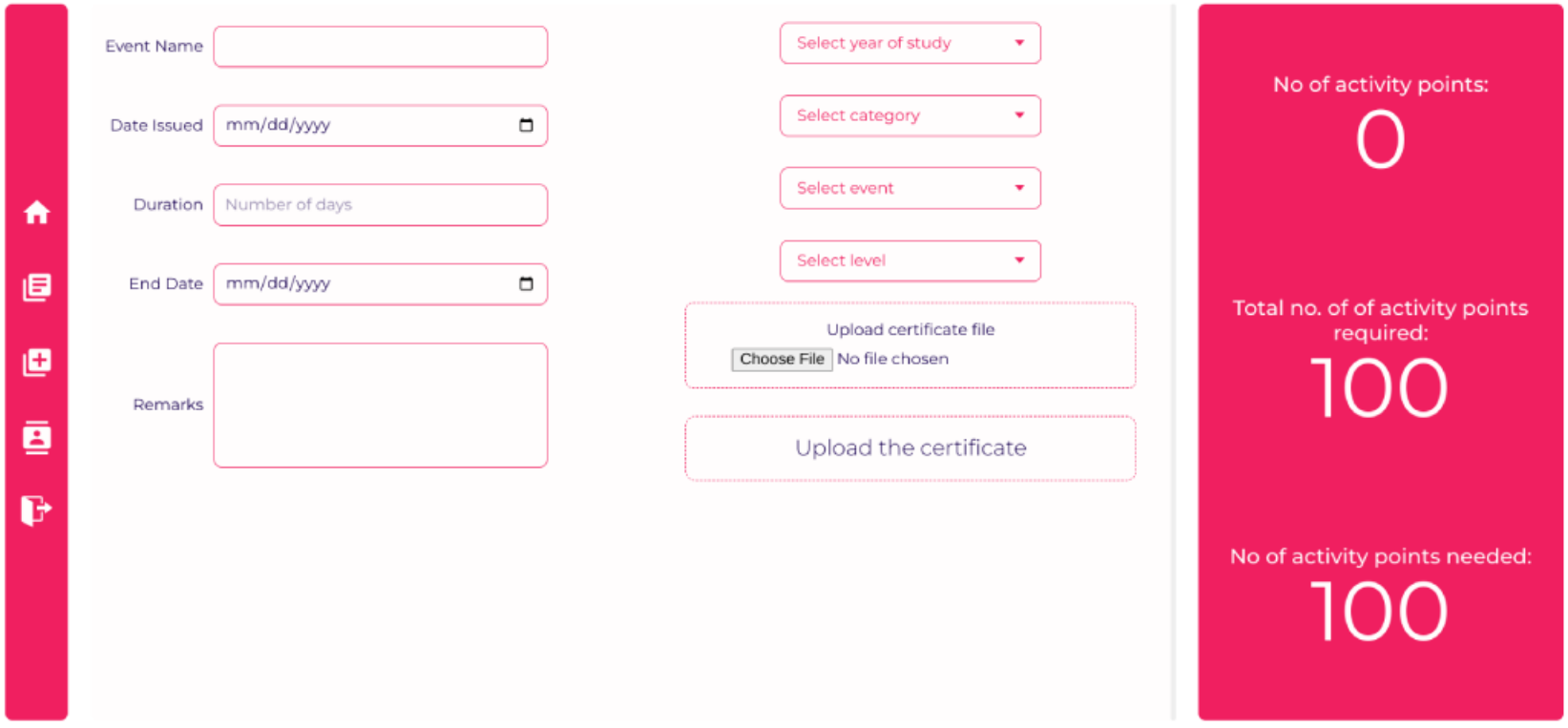
Dashboard


There are no notifications for now!
Clear All
View all your certificates
View your profile
No of activity points:
0
Total no. of of activity points required:
100
No of activity points needed:
100

Dashboard

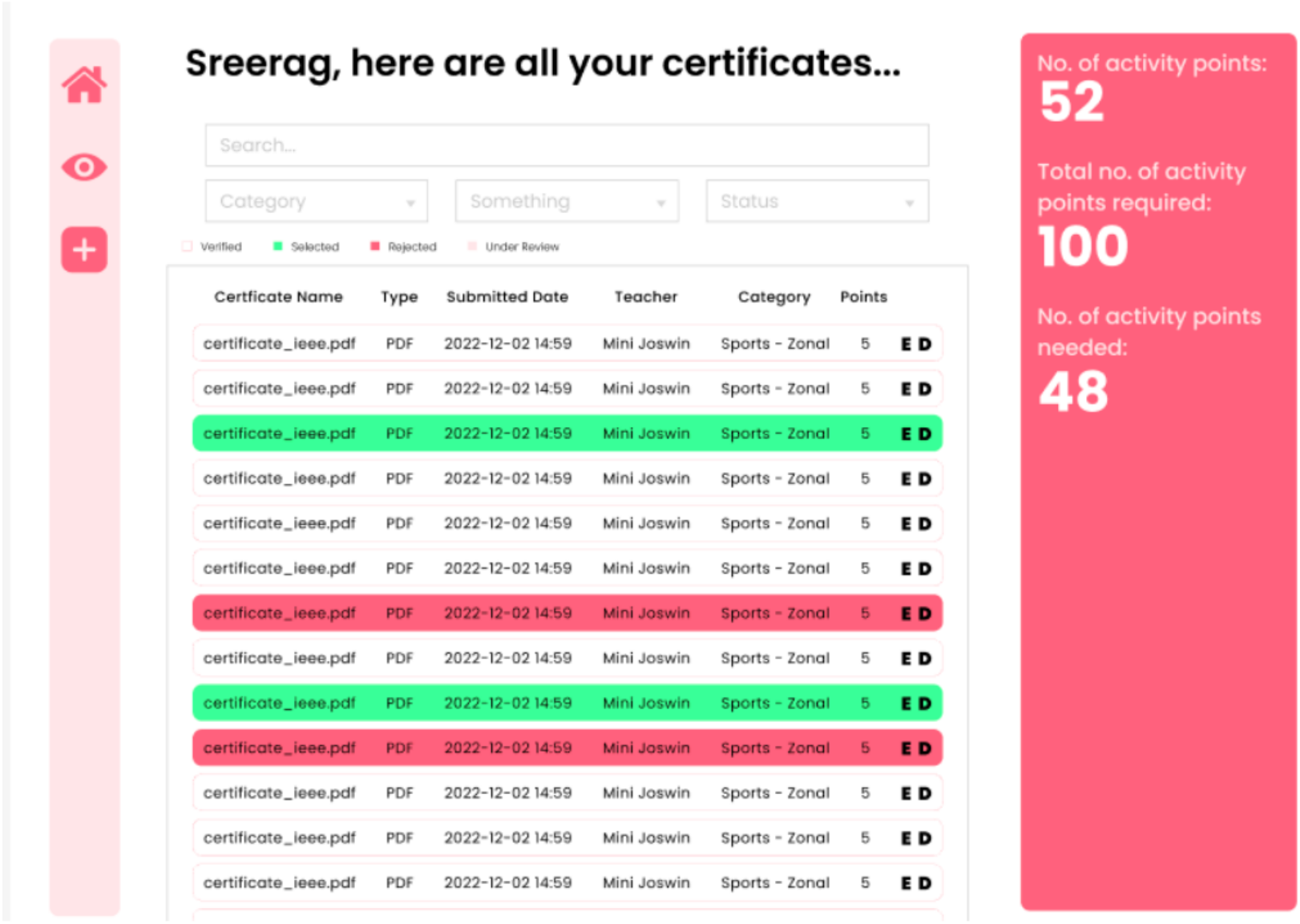

Home
View certificate
Add certificate
Profile
Log out
There are no notifications for now!
Clear All
View all your certificates
View your profile
No of activity points:
0
Total no. of of activity points required:
100
No of activity points needed:
100

Add Certificate



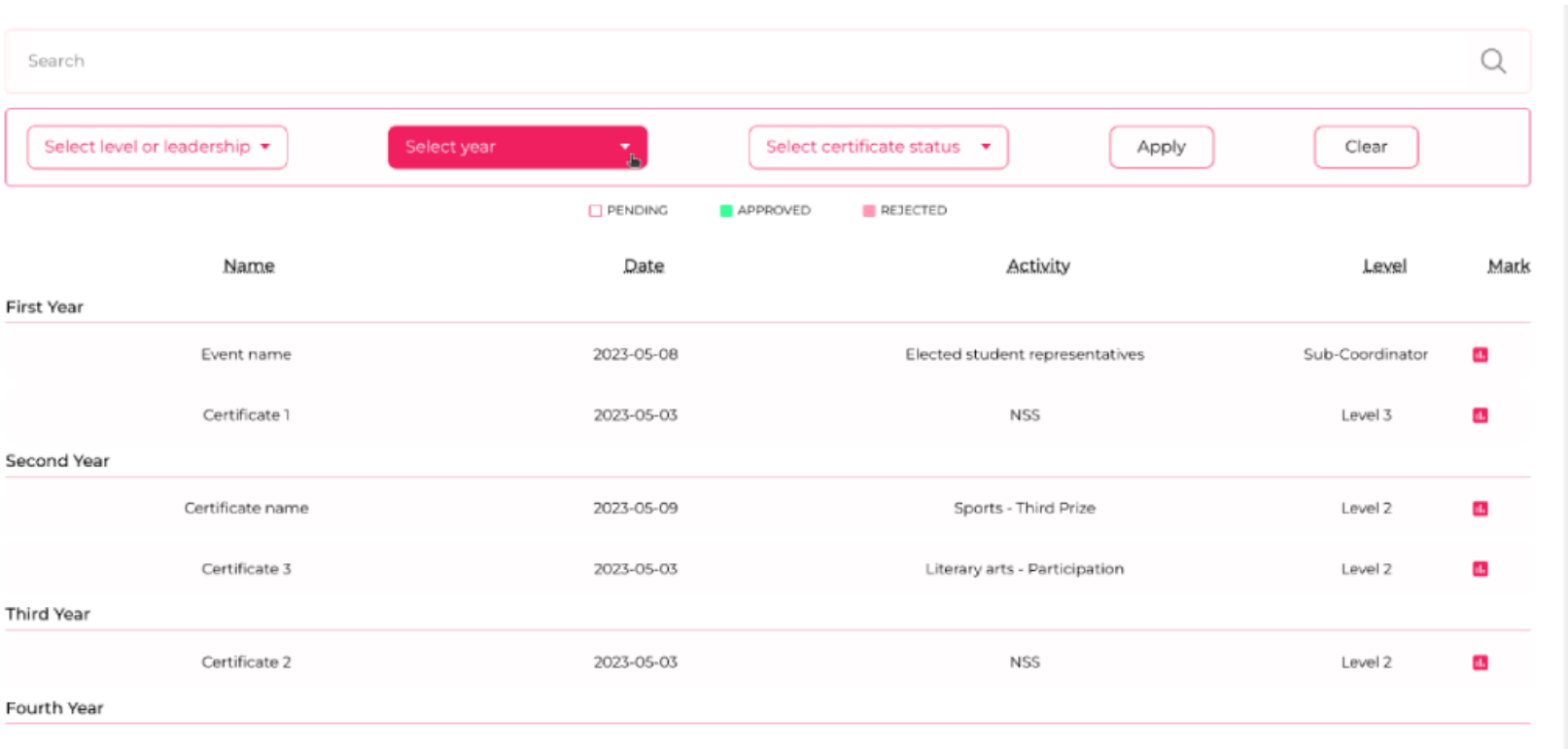
View Certificate

Student Side



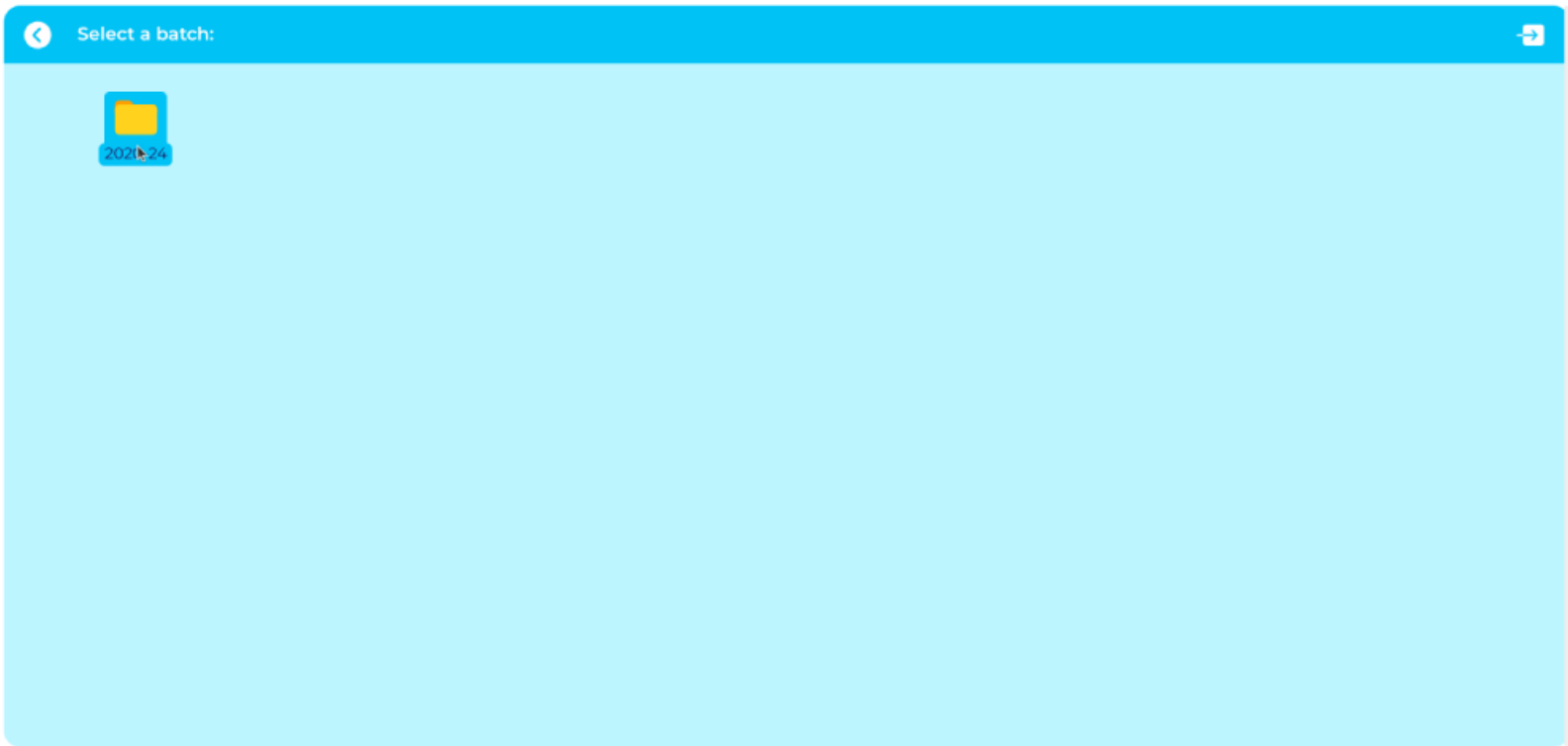
View Certificate

Teacher Side



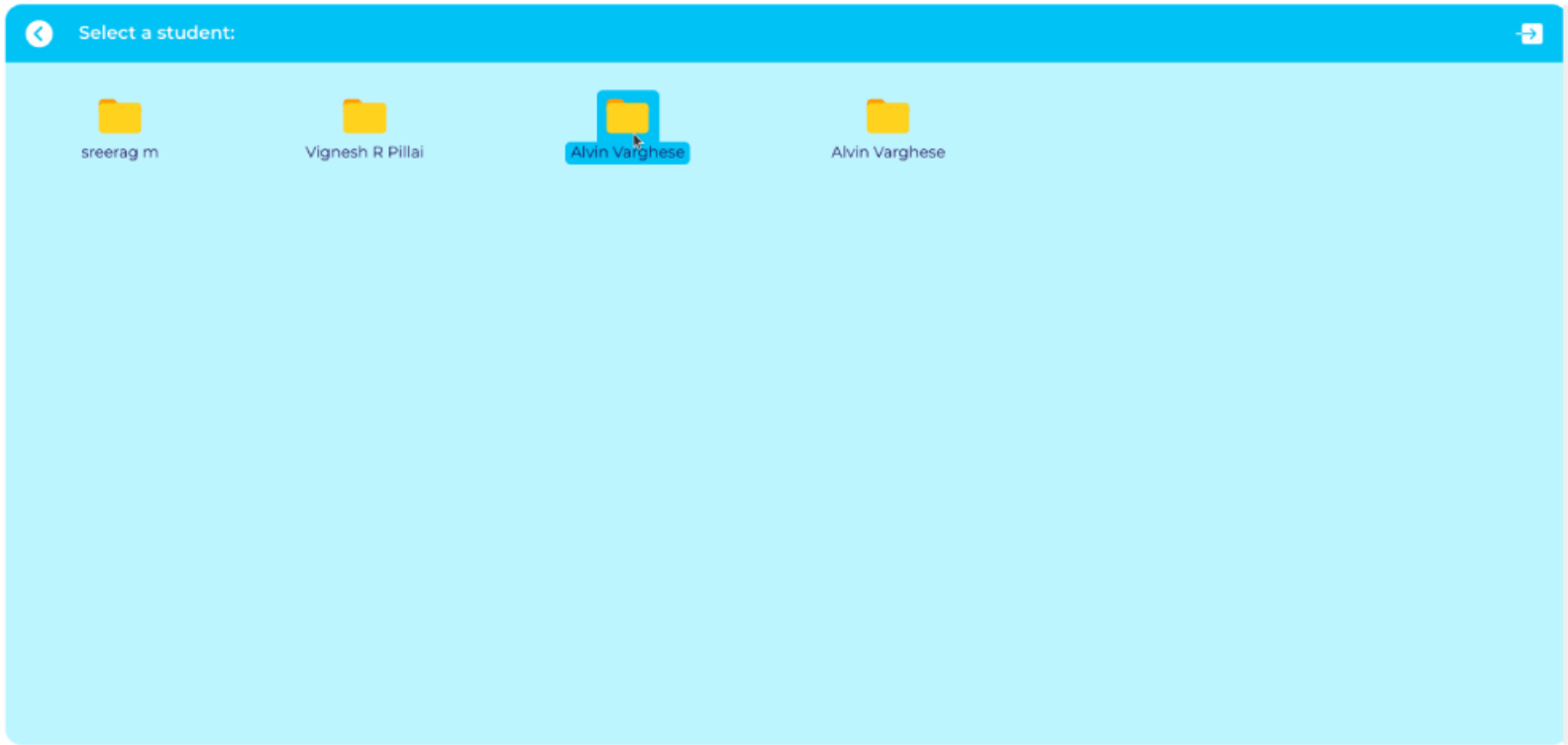
Batch Select

Teacher Side



Batch Select

Teacher Side



Module-2

Connectivity Algorithms

- Student Login algorithm
- Teacher Login algorithm
- Student Dashboard algorithm
- Certificate Add algorithm
- Certificate view(Teacher and Student) algorithm
- Report generating algorithm

Student Login

Connectivity Algorithm

1. User selects ”Login as student”.
2. Student logs in with their institution email.
3. Client sends request to server at api/auth/students/.
4. Server checks if student exists in database. If not, redirect student to a data collection form to enter admission number, KTU ID, and name. Server updates student collection in database.
5. Server generates access token for student.
6. Server redirects student to dashboard using access token for subsequent requests.

Teacher Login

Connectivity Algorithm

1. User selects "Login as teacher".
2. Teacher logs in with their institution email.
3. Client sends request to server at api/auth/teacher.
4. Server checks if teacher exists in database. If not, create teacher in collection.
5. Server generates access token for teacher.
6. Server redirects teacher to student select page using access token for subsequent requests.

Student Dashboard

Connectivity algorithm

1. Student logs in and is redirected to dashboard.
2. Client sends GET request to server for notifications at api/students/notifications.
3. Client displays notifications on the notification panel.
4. Sidebar displays button to fetch student's current activity points.
5. Client sends GET request to server at api/students/score and displays score in sidebar.
6. Student can navigate to all pages from navbar on the left.

Teacher Dashboard

Connectivity Algorithm

1. Teacher logs in and is redirected to dashboard.
2. Client sends GET request to server for all batches in database at api/batches.
3. Client displays all batches on UI.
4. Teacher selects a batch.
5. Client sends GET request to server for all students in that batch at api/batches/batchID.
6. Teacher selects a student.
7. Client sends GET request to server for all certificates of that student at api/certificates/studentID.
8. Client displays a page where all certificates are listed year-wise.

Add Certificate

Connectivity Algorithm

1. Student selects "Add Certificate" from dashboard.
2. Client sends GET request to server to fetch all available activities and categories of events.
3. Client displays available activities, categories of certificates in dropdown menus for student to select from.
4. Student selects certificate to add and enters required fields: student ID, certificate name, description, date, and any additional information.
5. Client sends new certificate data as POST request to server at api/certificates.
6. Server updates student's certificate collection with new certificate.
7. Server sends response to client indicating success or failure of certificate addition.

View Certificate

Connectivity Algorithm

1. Student or teacher navigates to the certificates page.
2. User clicks on a certificate.
3. If specific certificate is selected, client sends GET request to server for certificate details at api/certificates/certificateID.
4. Server returns details of selected certificate.
5. Client displays selected certificate's details on separate page.

Report generation

Connectivity Algorithm

1. Teacher navigates to report generation page.
2. Client sends GET request to server to fetch all available batches at api/batches.
3. Server returns all available batches.
4. Client displays all batches in dropdown menu for teacher to select from.
5. Teacher selects a batch.
6. If teacher clicks on "Generate Report":
   - 6.1 Client sends GET request to server to fetch details for selected batch at api/reports/batches/batchID.
   - 6.2 Client generates report based on data sent from server and creates PDF report.
   - 6.3 Client displays generated report on separate PDF page and allows download.

Report generation(continued...)

7. Else the client sends a GET request to server to fetch students at api/batches/batchID.
   - 7.1 The client displays all students in a dropdown menu for the teacher to select from.
   - 7.2 If teacher clicks on generate report here,
   - 7.3 The client sends a GET request to server to fetch details for the selected student at api/reports/students/studentID.
   - 7.4 The client generates the report based on the data sent from server and makes a PDF report.
   - 7.5 The client displays the generated report on a separate PDF page and can download if they want.

Module-3

Back End

- Background collections
- End point table

Category collection

The Category collection contains data on the points for each of the activities that students can participate in, following KTU guidelines and point allotment chart.

1. activityHead: type: String; required
2. activity: type: String; required
3. isLeadership: type: Boolean; default false
4. activityPoints: type: Object;
   - level1: type: Number
   - level2: type: Number
   - level3: type: Number
   - level4: type: Number
   - level5: type: Number

Category collection (Continued)

5. leadershipPoints: type: Object;
   - coreCoordinator: type: Number; default null
   - subCoordinator: type: Number; default null
   - volunteer: type: Number; default null
6. maxPoints: type: Number; required
7. minDuration: type: Number; required

Student Collection

The Student collection stores information about the students.

1. name: type: String; student's name; required
2. email: type: String; student's email address; required; unique
3. ktuId: type: String; KTU Id of the student; unique
4. college: type: String; student's college; required; default 'RIT, Kottayam'
5. admissionNumber: type: String; student's admission number; unique
6. batch: type: String; student's batch
7. profileImage: type: String; URL of the student's profile image; default " (empty string)

Teacher Collection

1. name: type: String; name of the teacher; required
2. email: type: String; email of the teacher; required, unique
3. password: type: String; password of the teacher; required

Certificate collection

The Certificate collection stores information about certificates submitted by students for completing different activities.

1. categoryId: type: mongoose.Schema.Types.ObjectId; reference to Category Collection
2. certificateName: type: String; name or title of certificate; required
3. studentId: type: mongoose.Schema.Types.ObjectId; reference to Student Collection; required
4. level: type: Number; default 0
5. points: type: Number
6. duration: type: Number; required
7. year: type: Number; required
8. status: type: String; 'pending', 'approved', 'rejected', or 'unapplied'; default 'pending'

Certificate collection (Continued)

9. certificateUrl: type: String; required
10. certificateDescription: type: String; required
11. lastVerifiedBy: type: mongoose.Schema.Types.ObjectId; reference to Teacher Collection; default null
12. participationDate: type: Date; required
13. leadershipLevel: type: Number; default 0
14. isLeadership: type: Boolean; default false
15. remarks: type: String; default ” (empty string)

Notification Collection

The Notification collection stores the notifications sent to students.

1. studentId: type: mongoose.Types.ObjectId; reference to Student Collection
2. message: type: String; notification message; required

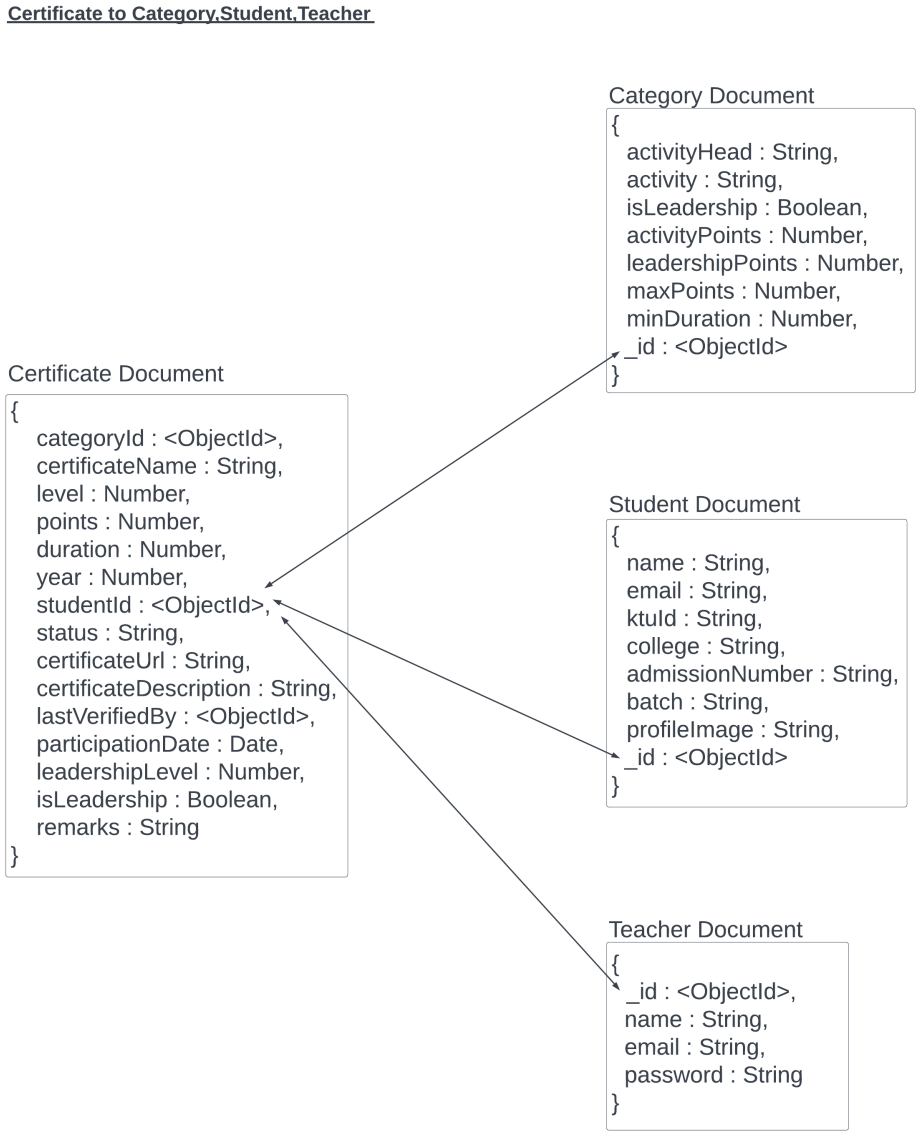
Score collection

The Score collection stores information about the scores of students.

1. targetScore: type: Number; default value is 100
2. currentScore: type: Number; default value is 0
3. studentId: type: mongoose.Types.ObjectId; reference to Student Collection

End Point table

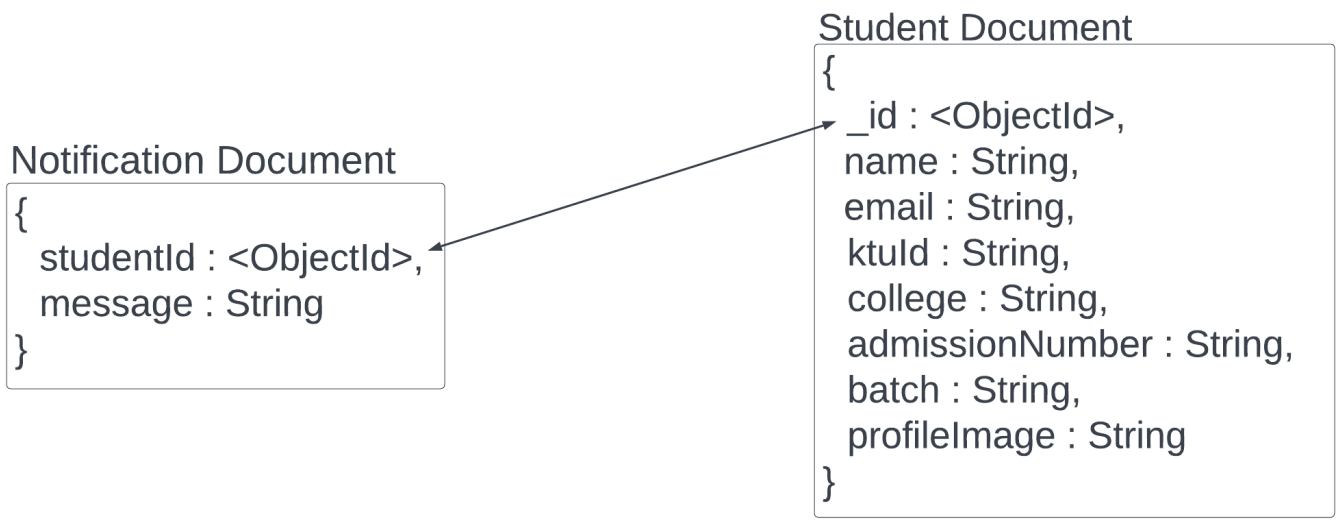
Certificate to category



End Point table

Notification to student

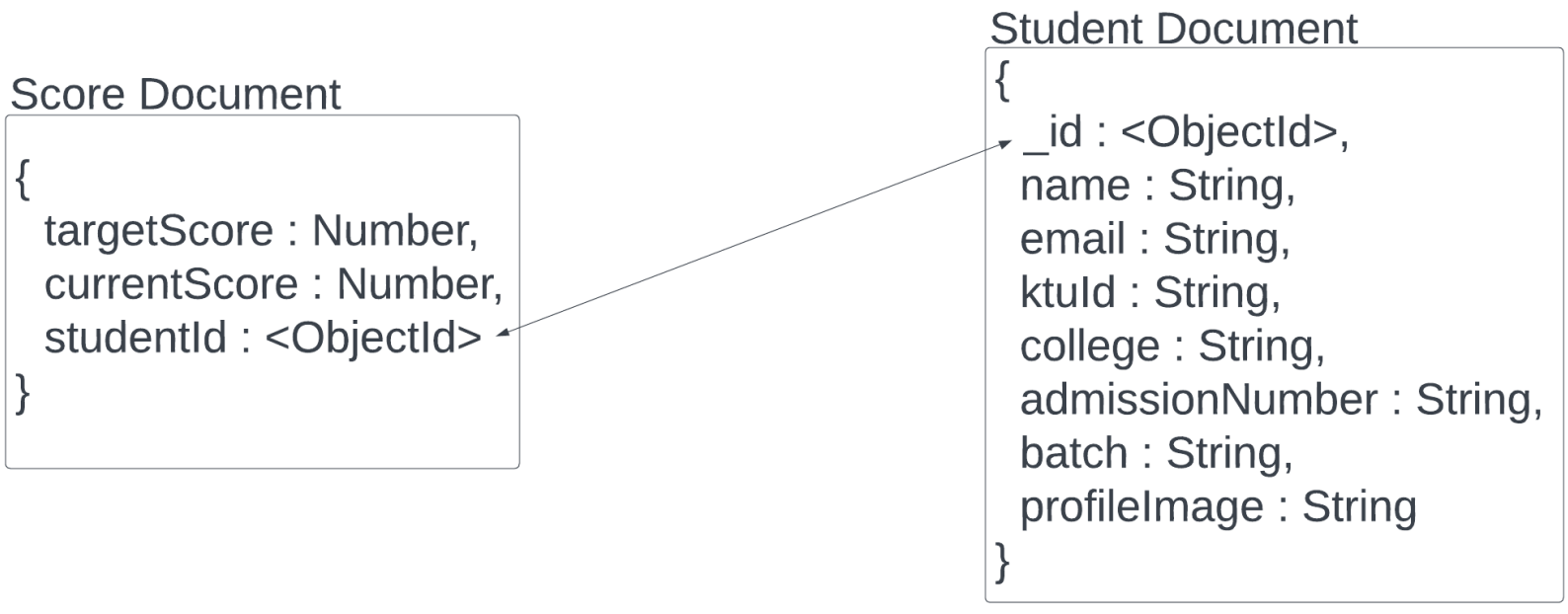
Notification to Student



End Point table

Notification to student

Score to Student



Thank You!