Database Project
Student Activity Tracker

Group Members:
Adam Rizky (24/536885/PA/22777)
Panji Merah Balakosa (24/532826/PA/22541)
Zidni Dziaulhaq (24/536623/PA/22759)

Course: Database - CSA

Submission Date:
December 28, 2025

Contents

Chapter 1

Introduction

1.1 Background Problem

In many schools and universities, student participation in extracurricular activities such as clubs, community service, competitions, or sports is still tracked manually. These are often recorded on paper-based lists or simple digital spreadsheets. This can lead to problems for example incomplete records, lost data, and difficulty in monitoring each student's level of participation. Advisors also face challenges in managing large numbers of participants and verifying who joined or completed certain activities. To solve these problems, the Student Activity Tracker System is designed to provide a structured, database-based solution. The system will allow advisors to register activities, record student participation, while students can view available activities, status of their applications and personal activity history.

1.2 System Objectives

- Provide an easy way to record and manage student extracurricular activities
- Help advisors register activities and track student participation
- Allow students to view available activities and their own participation history
- Keep activity records accurate, complete, and easy to access
- Reduce mistakes and data loss from manual recording
- Recognize active students based on their verified participation

1.3 System Users

The system involves three primary user roles: Admins, Advisors and Students.

- **Admins**: Staff members responsible for managing extracurricular activities. They can create, update, and maintain activity records within the system. Advisors review and approve or reject student participation requests and record activity outcomes once an activity is completed. Acting as supervisors, they oversee multiple activities and ensure accurate reporting of student involvement.
- **Advisors**: Staff members responsible for managing extracurricular activities and students enrolled on it. They can accept or reject students application as their advisor for the particular activities
- **Students**: The primary participants in extracurricular activities. They can view available activities, apply to participate, and track the status of their applications. Students can also access their participation history and view achievements and activity results, enabling them to maintain a complete record of their extracurricular involvement during their studies.

1.4 Use Cases

Admins:

- Register activity
- Update activity information
- View participation requests
- Change participation request status (accepted/rejected)

Advisors:

- Accept or reject student activity applications

Students:

- View list of activities
- Apply for an activity
- Track application status
- View participation history

Chapter 2

Database Design

2.1 Entity Relationship Diagram (ERD)

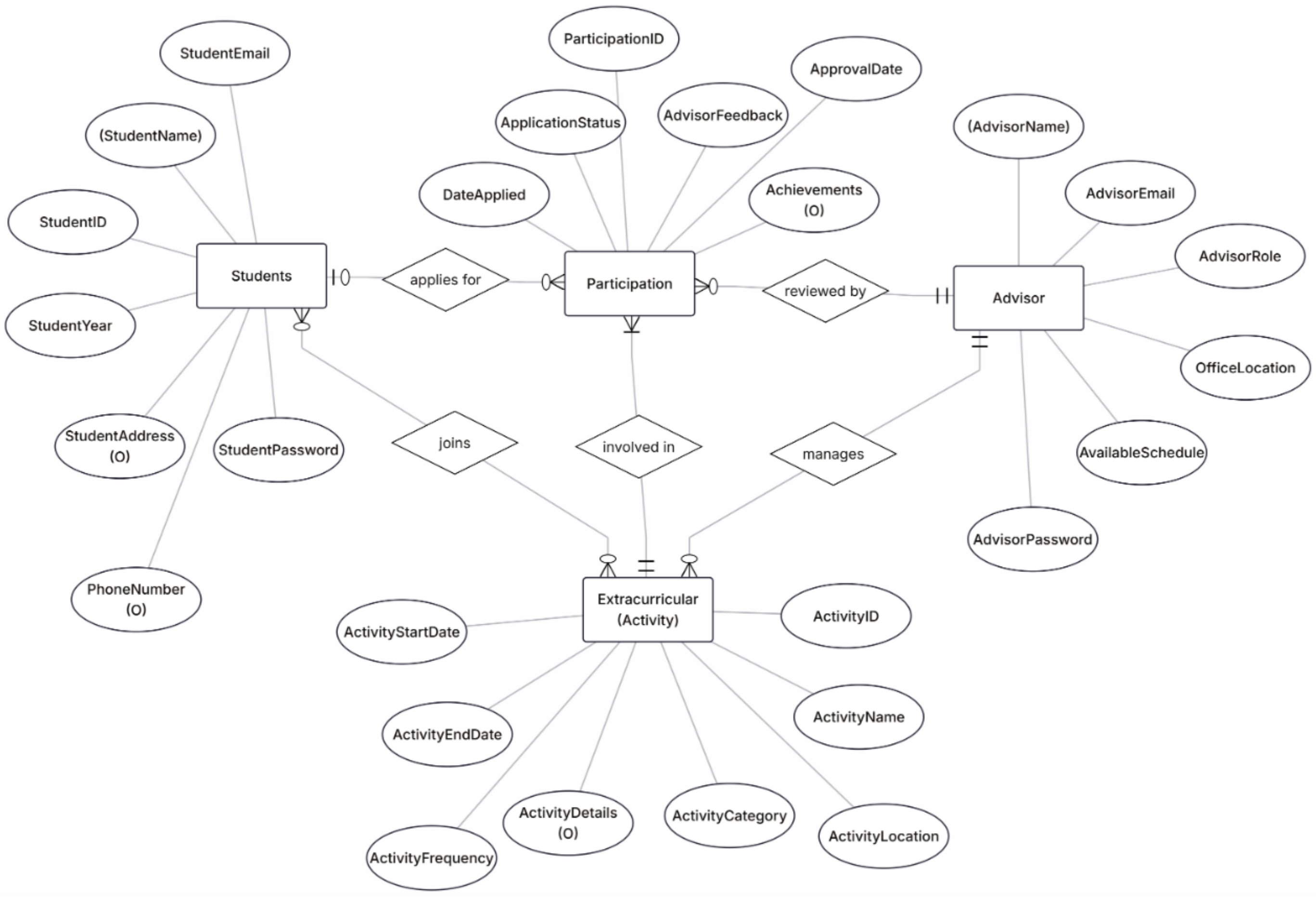


Figure 2.1: Entity Relationship Diagram

2.2 Entity Descriptions

2.2.1 Students Entity

The students entity represents individuals who can apply for and participate in extracurricular activities.

- **StudentID**: A unique identifier for each student
- **StudentName**: The full name of the student
- **StudentEmail**: The email address of the student
- **StudentYear**: The academic year of the student
- **StudentAddress**: The home address of the student (optional)
- **PhoneNumber**: The student's phone number (optional)
- **StudentPassword**: The password used by the student

2.2.2 Participation Entity

The participation entity represents the relationship between students and extracurricular activities, including application and approval details.

- **ParticipationID**: A unique identifier for each participation record
- **DateApplied**: The date when the student applied for the activity
- **ApplicationStatus**: The current status of the application
- **AdvisorFeedback**: Feedback provided by the advisor
- **ApprovalDate**: The date on which the application was approved
- **Achievements**: Achievements earned during participation (optional)

2.2.3 Advisor Entity

The advisor entity represents staff members who manage and review extracurricular activities.

- **AdvisorName**: The full name of the advisor
- **AdvisorEmail**: The email address of the advisor

- **AdvisorRole**: The role of the advisor
- **OfficeLocation**: The office location of the advisor
- **AvailableSchedule**: The advisor's working schedule
- **AdvisorPassword**: The password used by the advisor

2.2.4 Extracurricular Entity

The extracurricular entity represents activities that students can join.

- **ActivityID**: A unique identifier for each activity
- **ActivityName**: The name of the activity
- **ActivityCategory**: The category of the activity
- **ActivityLocation**: The location of the activity
- **ActivityFrequency**: When the activity takes place
- **ActivityStartDate**: The start date of the activity
- **ActivityEndDate**: The end date of the activity
- **ActivityDetails**: Additional details about the activity (optional)

2.3 Normalization Steps

2.3.1 Unnormalized Form (UNF)

At the beginning, all the information about students, advisors, extracurricular activities, and participation is stored in one large table. This causes a lot of repetition because the same student, advisor, or activity details may appear many times.

2.3.2 First Normal Form (1NF)

To convert the data into 1NF, each field is made so that it contains only one value. Repeating groups are removed, and the data is separated into different tables based on their entities, such as Students, Advisors, Extracurricular, and Participation.

2.3.3 Second Normal Form (2NF)

A table becomes 2NF if all non-key attributes depend on the full primary key. In this database, each table uses a single-column primary key, so there are no partial dependencies. Thus, all tables meet the conditions of 2NF.

2.3.4 Third Normal Form (3NF)

A table becomes 3NF if it doesnt't contain transitive dependencies. Meaning that non-key attributes should not depend on other non-key attributes. In this design, all attributes depend directly on their unique identifier, such as StudentID, AdvisorID, ActivityID, or ParticipationID. Because there are no transitive dependencies, the database meets the conditions of 3NF.

Chapter 3

Database Implementation

3.1 Relational Schema

The relational schema defines the structure of the database tables, attributes, and how they are connected with primary and foreign keys

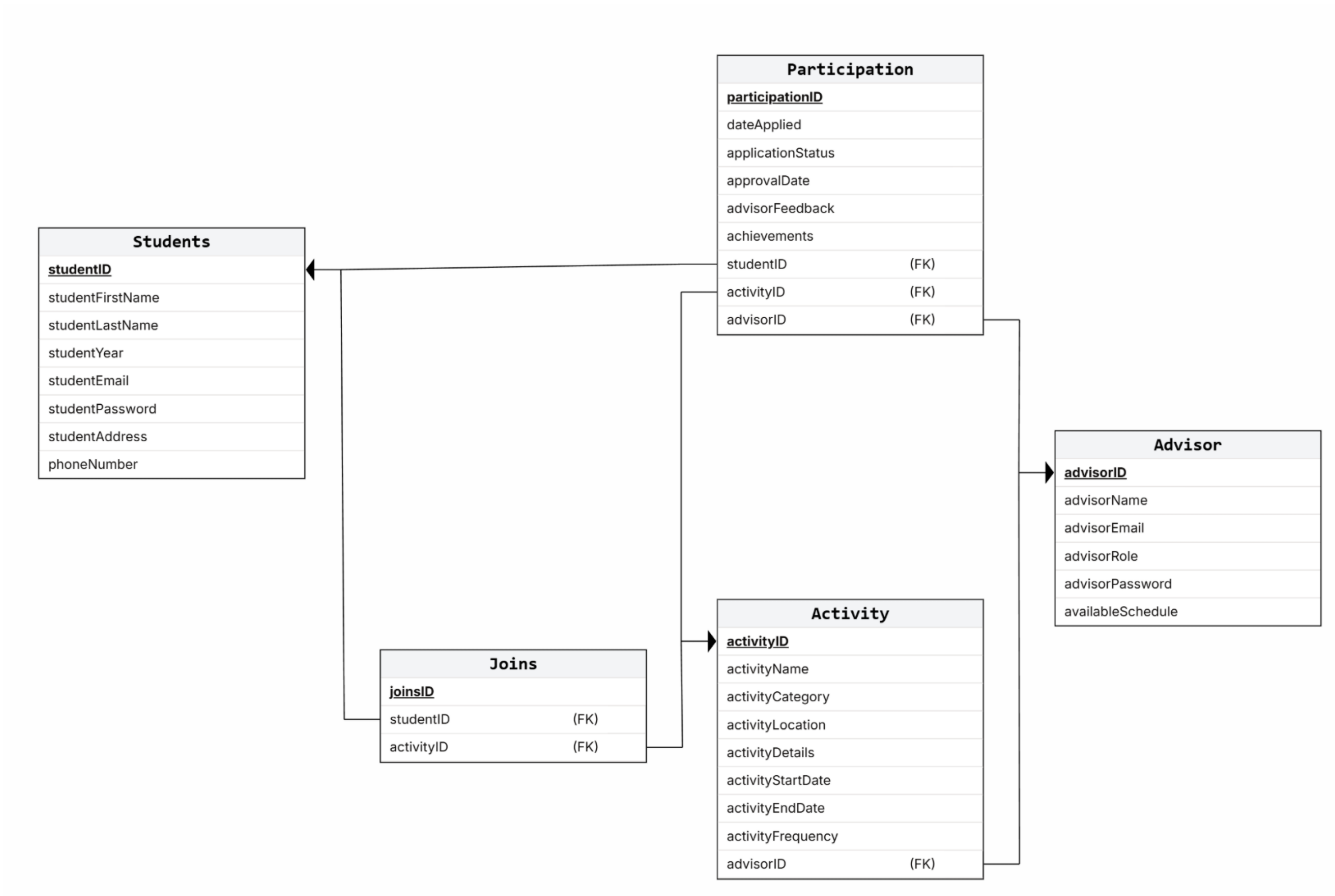


Figure 3.1: Entity Relationship Diagram

3.2 Key Constraints

3.2.1 Primary Key Constraints

Primary keys are used to uniquely identify each record in a table.

- studentID is the primary key of the STUDENTS table
- advisorID is the primary key of the ADVISOR table
- activityID is the primary key of the ACTIVITY table
- participationID is the primary key of the PARTICIPATION table

3.2.2 Foreign Key Constraints

Foreign keys are used to connect tables and maintain relationships between them.

- PARTICIPATION.studentID references STUDENTS(studentID)
- PARTICIPATION.activityID references ACTIVITY(activityID)
- PARTICIPATION.advisorID references ADVISOR(advisorID)
- ACTIVITY.advisorID references ADVISOR(advisorID)

3.3 Example SQL Statements

Advisor Table

```
 CREATE TABLE advisor (
advisorID INT(11) NOT NULL AUTO_INCREMENT,
advisorName VARCHAR(20),
advisorEmail VARCHAR(25),
advisorRole VARCHAR(15),
advisorPassword VARCHAR(15),
availableSchedule VARCHAR(100),
PRIMARY KEY (advisorID)
);
```

Students Table

```
 CREATE TABLE students (
studentID INT(11) NOT NULL AUTO_INCREMENT,
studentFirstName VARCHAR(15),
studentLastName VARCHAR(15),
studentYear INT(11),
studentEmail VARCHAR(25),
studentPassword VARCHAR(15),
studentAddress VARCHAR(255),
phoneNumber INT(11),
PRIMARY KEY (studentID)
);
```

Activity Table

```
 CREATE TABLE activity (
activityID INT(11) NOT NULL AUTO_INCREMENT,
activityName VARCHAR(30),
activityCategory VARCHAR(20),
activityLocation VARCHAR(50),
activityDetails TEXT,
activityStartDate DATE,
activityEndDate DATE,
activityFrequency VARCHAR(20),
advisorID INT(11),
PRIMARY KEY (activityID),
FOREIGN KEY (advisorID) REFERENCES advisor(advisorID)
);
```

Participation Table

```
 CREATE TABLE participation (
participationID INT(11) NOT NULL AUTO_INCREMENT,
dateApplied DATE,
applicationStatus VARCHAR(20),
approvalDate DATE,
advisorFeedback TEXT,
achievements TEXT,
studentID INT(11),
activityID INT(11),
advisorID INT(11),
PRIMARY KEY (participationID),
FOREIGN KEY (studentID) REFERENCES students(studentID),
FOREIGN KEY (activityID) REFERENCES activity(activityID),
FOREIGN KEY (advisorID) REFERENCES advisor(advisorID)
);
```

Joins Table

```
 CREATE TABLE joins (
joinsID INT(11) NOT NULL AUTO_INCREMENT,
studentID INT(11),
activityID INT(11),
PRIMARY KEY (joinsID),
FOREIGN KEY (studentID) REFERENCES students(studentID),
FOREIGN KEY (activityID) REFERENCES activity(activityID)
);
```

Initiations of the SQLAlchemy

```
def create_app():
    app = Flask(__name__)
    app.config["SECRET_KEY"] = "AkuCintaARIs"
    app.config["SQLALCHEMY_DATABASE_URI"] = "sqlite:///student_activities.db"
    app.config["SQLALCHEMY_TRACK_MODIFICATIONS"] = False

    db.init_app(app)
```

Abstracting student table using SQLAlchemy

```
class Students(db.Model, UserMixin):
    __tablename__ = "students"

    studentID = db.Column(db.Integer, primary_key=True)
    studentFirstName = db.Column(db.String(15))
    studentLastName = db.Column(db.String(15))
    studentYear = db.Column(db.Integer)
    studentEmail = db.Column(db.String(25))
    studentPassword = db.Column(db.String(15))
    studentAddress = db.Column(db.String(255))
    phoneNumber = db.Column(db.Integer)

    @property
    def id(self):
        return f"s{self.studentID}"

    @property
    def role_type(self):
        return "student"
```

Abstracting participation table using SQLAlchemy

```
class Participation(db.Model):
    __tablename__ = "participation"

    participationID = db.Column(db.Integer, primary_key=True)
    dateApplied = db.Column(db.Date)
    applicationStatus = db.Column(db.String(20))
    approvalDate = db.Column(db.Date)
    advisorFeedback = db.Column(db.Text)
    achievements = db.Column(db.Text)

    studentID = db.Column(db.Integer, db.ForeignKey("students.studentID"))
    activityID = db.Column(db.Integer, db.ForeignKey("activity.activityID"))
    advisorID = db.Column(db.Integer, db.ForeignKey("advisor.advisorID"))

    student = db.relationship("Students")
    activity = db.relationship("Activity")
    advisor = db.relationship("Advisor")
```

Chapter 4

Application Implementation

4.1 System Overview

The Student Activity Tracker is implemented as a web-based application using the Flask framework in Python. The system follows a modular architecture where each functionality is separated into different components such as authentication, database models, views, and application programming interfaces (API).

The application uses SQLAlchemy as an Object Relational Mapping (ORM) tool to interact with a SQLite database. User authentication and session management are handled using Flask-Login, while a RESTful API secured with JSON Web Token (JWT) is provided for external access.

You may create a new account as a student to test the functionality of the website. No token or one-time code needed since it's just an oversimplified version of a real world website.

4.2 Application Architecture

The system follows an MVC-style architecture:

- **Models**: Define database tables using SQLAlchemy ORM.
- **Views**: Handle routing, user interaction, and rendering templates.
- **Controllers**: Business logic implemented through Flask Blueprints.

4.3 Main Modules

4.3.1 Authentication Module

The authentication module supports student and advisor login. Advisors require approval from an administrator before registering as an advisor. Passwords are stored securely using hashing. The module also supports advisor registration and logout functionality.

4.3.2 Student Module

Students can view available activities, search activities by name, category, or location, apply for participation, view application status, and manage their personal profile. Students can also view their activity history through a filtered search.

4.3.3 Advisor Module

Advisors can access a dashboard showing participation requests assigned to them. They can approve, reject, update feedback, or delete participation requests. Advisors can also update their personal profile.

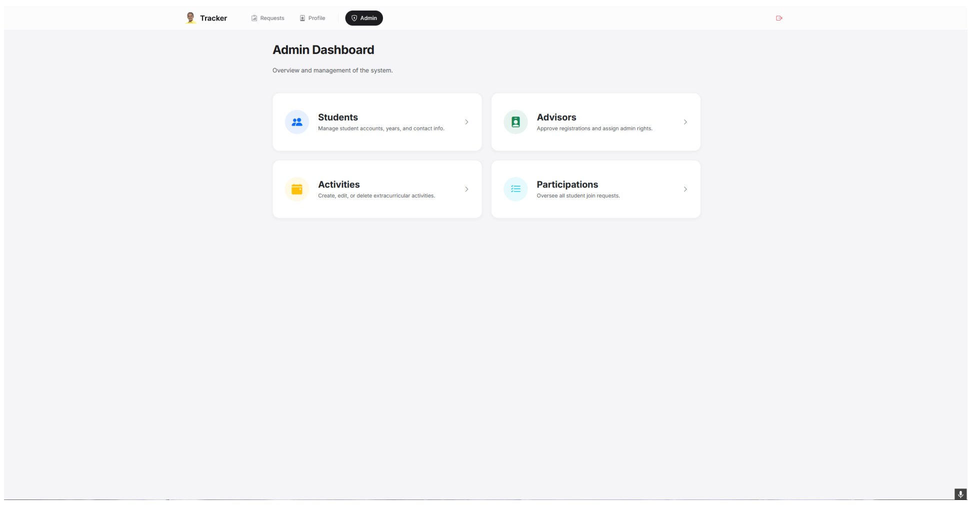
4.3.4 Admin Module

The admin module provides full management of students, advisors, activities, and participation records. Admins can approve advisor account registrations, assign admin privileges, and perform complete CRUD operations.

4.3.5 REST API Module

The system includes a RESTful API secured using JWT. The API supports login, student management, activity management, participation management, and activity search, allowing integration with external applications.

4.4 Application Screenshots



Admin Dashboard

Overview and management of the system.

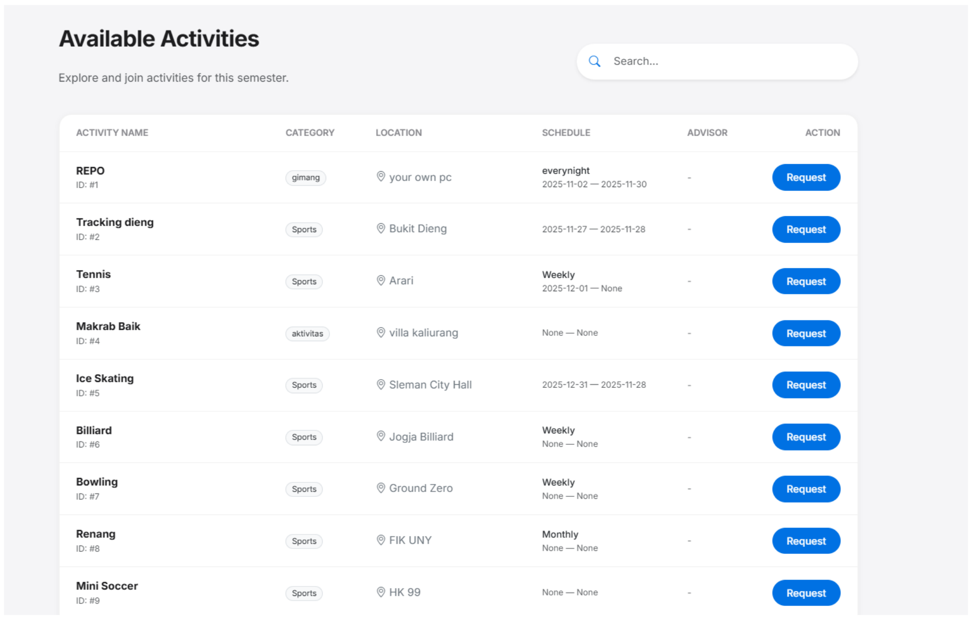
Students
Manage student accounts, years, and contact info.

Advisors
Approve registrations and assign admin rights.

Activities
Create, edit, or delete extracurricular activities.

Participations
Oversee all student join requests.

Figure 4.1: Student Dashboard Interface

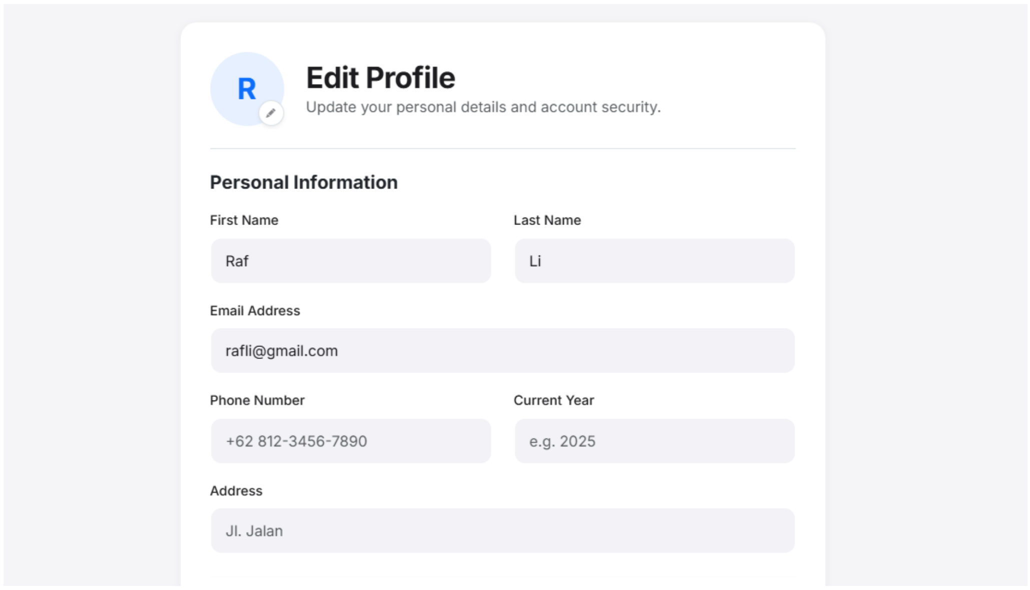


Available Activities

Explore and join activities for this semester.

ACTIVITY NAME	CATEGORY	LOCATION	SCHEDULE	ADVISOR	ACTION
REPO ID: #1	gimang	your own pc	everynight 2025-11-02 — 2025-11-30	-	Request
Tracking dieng ID: #2	Sports	Bukit Dieng	2025-11-27 — 2025-11-28	-	Request
Tennis ID: #3	Sports	Arari	Weekly 2025-12-01 — None	-	Request
Makrab Baik ID: #4	aktivitas	villa kaliurang	None — None	-	Request
Ice Skating ID: #5	Sports	Sleman City Hall	2025-12-31 — 2025-11-28	-	Request
Billiard ID: #6	Sports	Jogja Billiard	Weekly None — None	-	Request
Bowling ID: #7	Sports	Ground Zero	Weekly None — None	-	Request
Renang ID: #8	Sports	FIK UNY	Monthly None — None	-	Request
Mini Soccer ID: #9	Sports	HK 99	None — None	-	Request

Figure 4.2: Available Activities Page



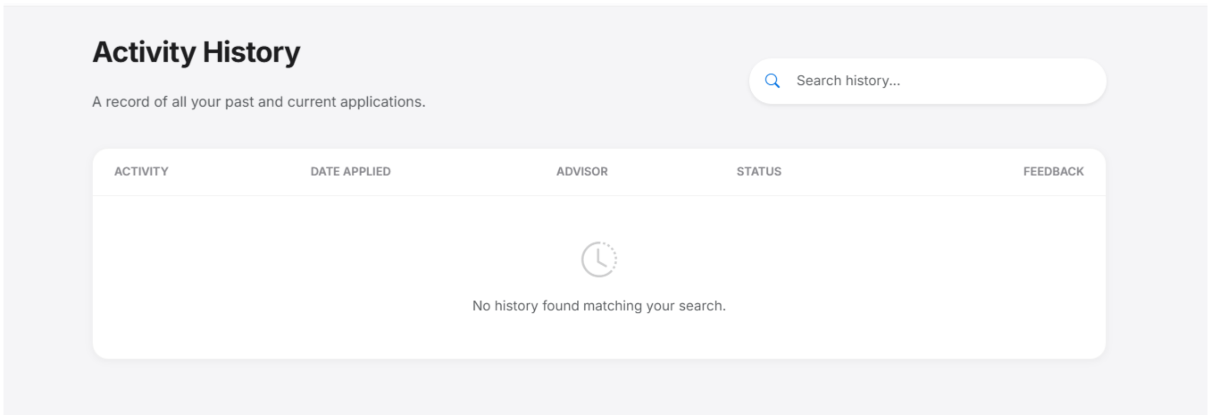
Edit Profile

Update your personal details and account security.

Personal Information

First Name	Raf
Last Name	Li
Email Address	rafli@gmail.com
Phone Number	+62 812-3456-7890
Current Year	e.g. 2025
Address	Jl. Jalan

Figure 4.3: Edit Profile Page



Activity History

A record of all your past and current applications.

Search history...

ACTIVITY	DATE APPLIED	ADVISOR	STATUS	FEEDBACK

No history found matching your search.

Figure 4.4: Activity History Page

Chapter 5

Testing and Results

5.1 Testing Methodology

The system was tested using functional testing by simulating real user actions for each role. Testing was performed for students, advisors, and administrators to ensure that access control, data processing, and database operations worked correctly.

Both web-based interactions and REST API endpoints were tested using sample data.

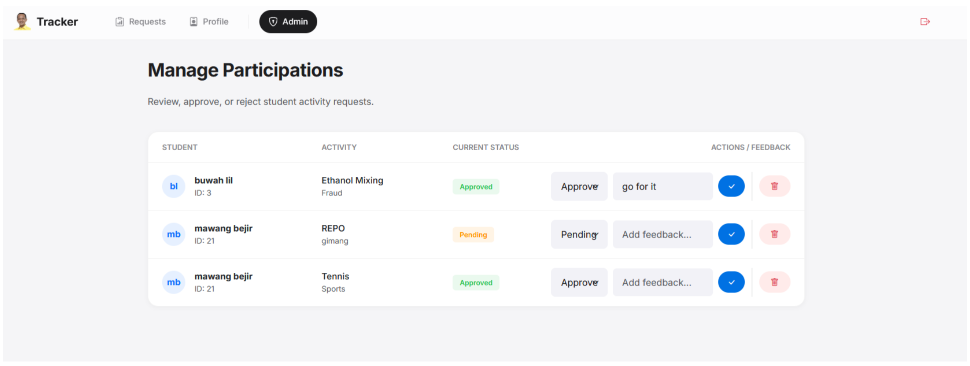
5.2 Sample Queries and Operations

The following database operations were tested through the application:

- Student registration and login
- Advisor registration and admin approval
- Creating, updating, and deleting activities
- Submitting participation requests
- Approving and rejecting participation requests
- Searching activities by keywords
- Viewing participation history

5.3 Results and Screenshots

All tested features functioned correctly. The system successfully enforced role-based access control and maintained data consistency through foreign key constraints.

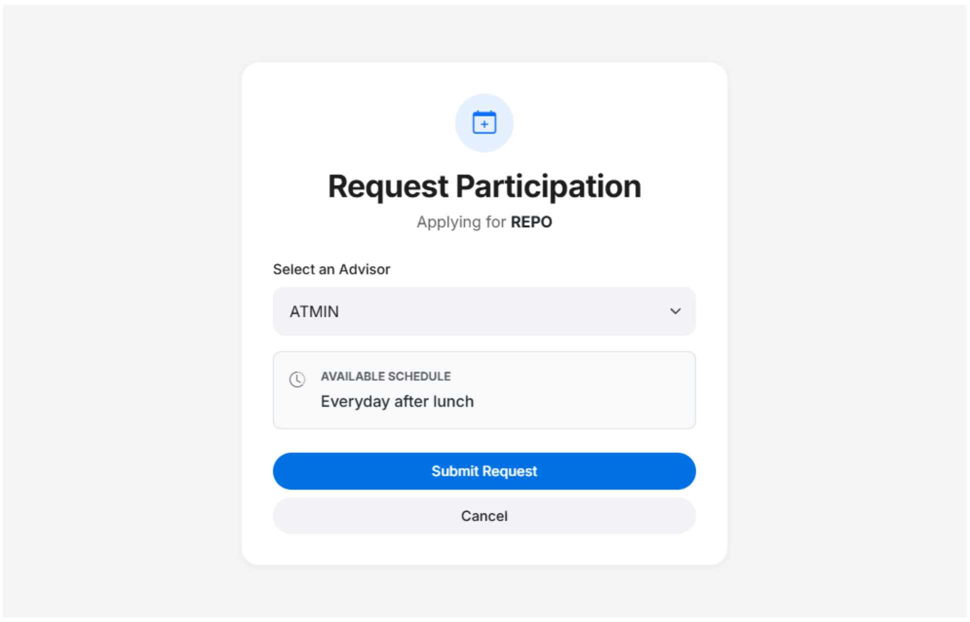


Manage Participations

Review, approve, or reject student activity requests.

STUDENT	ACTIVITY	CURRENT STATUS	ACTIONS / FEEDBACK	
bl buwah lil ID: 3	Ethanol Mixing Fraud	Approved	Approve	go for it
mb mawang bejir ID: 21	REPO gimang	Pending	Pending	Add feedback...
mb mawang bejir ID: 21	Tennis Sports	Approved	Approve	Add feedback...

Figure 5.1: Participation and Activity Data Displayed


Request Participation

Applying for **REPO**

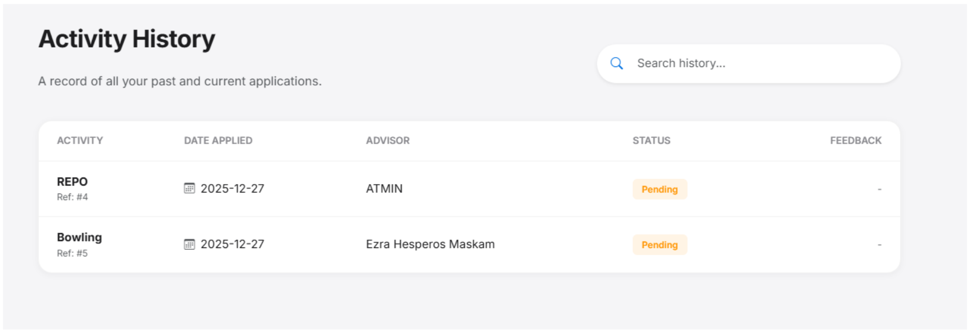
Select an Advisor

ATMIN

AVAILABLE SCHEDULE

Everyday after lunch

Figure 5.2: Request to join an activity

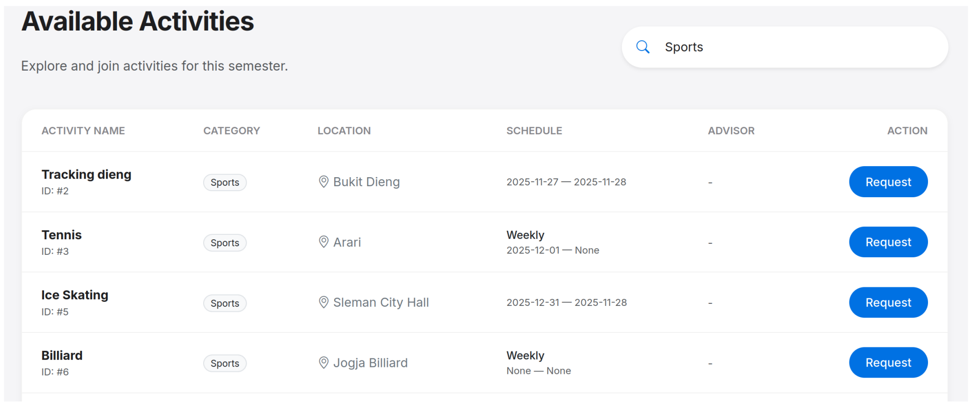

Activity History

A record of all your past and current applications.

ACTIVITY	DATE APPLIED	ADVISOR	STATUS	FEEDBACK
REPO Ref: #4	2025-12-27	ATMIN	Pending	-
Bowling Ref: #5	2025-12-27	Ezra Hesperos Maskam	Pending	-

Figure 5.3: History record data from student POV

As demonstrated in Figure 5.4, the search function is designed for versatility. In this specific example, the search query successfully matches the activity category rather than the activity name itself, highlighting its multipurpose nature. While users can certainly search using specific keywords for activity names, the system is flexible enough to parse broader attributes to return relevant results.



Available Activities

Explore and join activities for this semester.

ACTIVITY NAME	CATEGORY	LOCATION	SCHEDULE	ADVISOR	ACTION
Tracking dieng ID: #2	Sports	Bukit Dieng	2025-11-27 — 2025-11-28	-	Request
Tennis ID: #3	Sports	Arari	Weekly 2025-12-01 — None	-	Request
Ice Skating ID: #5	Sports	Sleman City Hall	2025-12-31 — 2025-11-28	-	Request
Billiard ID: #6	Sports	Jogja Billiard	Weekly None — None	-	Request

Figure 5.4: Search Functionality Example

5.4 Discussion of Results

The testing results show that the application correctly implements the designed database structure. Data integrity is preserved, and user actions result in appropriate database updates. The use of ORM reduces query errors and improves maintainability.

Chapter 6

Conclusion and Reflection

6.1 Conclusion

The Student Activity Tracker system successfully implements a complete database-driven web application. The system supports multiple user roles, structured activity management, and a clear participation workflow. Core database concepts such as normalization, primary keys, foreign keys, and relational integrity are applied effectively.

6.2 Challenges Faced

Several challenges were encountered during development:

- Implementing role-based access control
- Managing approval workflows for advisors and students
- Ensuring secure authentication and authorization
- Designing scalable database relationships

6.3 Future Improvements

Possible future enhancements include:

- Adding activity attendance tracking
- Enhancing API security and permissions
- Implementing login timeout for security