AceGrade

AceGrade is a lightweight, full-stack web app to help students organize study materials, track CGPA, and access previous year question papers. It provides a simple static frontend and a Spring Boot backend with PostgreSQL for persistence and file storage.

Tech Stack

- Backend: Spring Boot 3 (Java 17), Spring Web, Spring Data JPA, Bean Validation
- Database: PostgreSQL
- Build: Maven
- Frontend: HTML, CSS, Vanilla JavaScript
- File Storage: Local filesystem (`/uploads`)

Core Features and Use Cases

- Study Resources
  - Filter by semester, department, and regulation
  - View subjects and download uploaded PDF notes
  - Upload new notes as PDF with contributor name and description
- Question Papers
  - Upload and download previous year question papers (PDF)
  - Filter by semester/department/regulation
- CGPA Tracker
  - Persist per-semester courses, grades, and credits per user
  - Save and retrieve data tied to a user
- Authentication (Basic)
  - Simple login via college email and password
  - Health check endpoint

Repository Structure

```
AceGrade/
  backend/
    pom.xml                     # Spring Boot + Maven configuration (Java 17)
    database-setup.sql          # PostgreSQL schema/seed helper
    src/main/java/com/acegrade/
      AceGradeApplication.java
      controller/
        CgpaController.java     # /api/cgpa
        LoginController.java    # /api/auth
        QpaperController.java   # /api/qpaper
        StudyResourceController.java  # /api/study-resources
      dto/                      # Request/response DTOs
      entity/                   # JPA entities (User, Subject, StudyResource, CGPA*)
      repository/               # Spring Data repositories
    src/main/resources/
      application.properties    # DB, CORS, and multipart config
    uploads/                    # Runtime: stored files (notes/qpapers)

  frontend/
    index.html                  # Landing page
    login.html                  # Login UI
    dashboard.html              # User dashboard
    cgpa-calculator.html        # CGPA tracker UI
    study-resources.html        # Notes UI (upload/list/download)
    study-resources.js          # Notes page scripting
    qpapers.html                # Question papers UI
    qpapers.js                  # QP page scripting
    styles.css / study-resources.css  # Styling
    FONTS/, IMAGES/             # Assets

  start-backend.sh              # macOS/Linux helper to run backend
  start-backend.bat             # Windows helper to run backend
  README.md                     # This file
  SETUP_GUIDE.md                # Extra setup help
  TROUBLESHOOTING.md            # Common issues
```

API Overview (high level)

- Auth: `POST /api/auth/login`, `GET /api/auth/health`
- Study Resources:
  - `GET /api/study-resources/subjects` (filters: semester, department, regulation)
  - `GET /api/study-resources` (filters + optional searchTerm)
  - `POST /api/study-resources` (multipart PDF upload)
  - `GET /api/study-resources/download/{id}`
- Question Papers: similar endpoints under `/api/qpaper`
- CGPA: `GET /api/cgpa/{userId}`, `POST /api/cgpa/{userId}`

Run Locally

Prerequisites

- Java 17 (verify: `java -version`)
- Maven (verify: `mvn -version`)
- PostgreSQL 13+ (verify: `psql --version`)

1) Database Setup

1. Ensure PostgreSQL is running.
2. Create a database and user that match `backend/src/main/resources/application.properties`:
   - Default expected values:
     - URL: `jdbc:postgresql://localhost:5432/acegrade`
     - Username: `postgres`
     - Password: `password`
3. Optionally run the helper SQL (`backend/database-setup.sql`) in psql:
   - `psql -U postgres -d acegrade -f backend/database-setup.sql`
4. Adjust credentials if needed in `application.properties`.

2) Start the Backend

- Windows (PowerShell or cmd):
  - Double-click `start-backend.bat` or run:
    - `. start-backend.bat`
- macOS/Linux (Terminal):
  - `bash start-backend.sh`

Backend will serve on `http://localhost:8080`.

Health check: `http://localhost:8080/api/auth/health`

Notes

- File uploads are saved to `backend/uploads/` (created on demand).
- Max upload size is 10MB (configurable in `application.properties`).

3) Open the Frontend

The frontend is static HTML/CSS/JS and can be opened directly in a browser.

Option A: Open files directly

- Open `frontend/study-resources.html` for notes
- Open `frontend/qpapers.html` for question papers
- Open `frontend/cgpa-calculator.html` for CGPA tracker

Option B: Serve with a simple static server (recommended for CORS)

- Python 3: from `frontend/` run `python -m http.server 5500`
- Node: from repo root run `npx serve frontend -l 5500`

Then visit `http://localhost:5500/study-resources.html`, etc.

4) Configure CORS (if needed)

`application.properties` permits `http://localhost:3000`, `http://127.0.0.1:3000`, and `file://` by default; controllers also allow `http://127.0.0.1:5500` and `http://localhost:5500`. If your frontend runs on a different origin, add it to:

- `spring.web.cors.allowed-origins` in `application.properties`
- Or the `@CrossOrigin` annotations in controllers

Development

Useful Commands

- Build backend: `cd backend && mvn clean package`
- Run backend: `cd backend && mvn spring-boot:run`

Where to change things

- DB credentials: `backend/src/main/resources/application.properties`
- Upload size limits: same file (multipart settings)
- Save location for files: `UPLOAD_DIR` in controllers
- Frontend API base URLs: see JS files (e.g., `frontend/study-resources.js`)

Common Troubleshooting

- Cannot connect to PostgreSQL: verify credentials and that DB exists
- CORS errors: align frontend origin with allowed origins in backend
- 413 on upload: increase `spring.servlet.multipart.max-file-size`
- Downloads 404: confirm file exists under `backend/uploads/` and DB path

License

This project is provided as-is for educational purposes. Add your preferred license.