EdTech Assignment Tracker (Problem Solving + API Design)

# Objective

Design and implement a simplified assignment tracking system for an EdTech platform that allows teachers to post assignments and students to submit them.

# Part A – System Design

## 1. System Architecture

A Client-Server architecture with RESTful API backend.  
  
Frontend (React/HTML-CSS-JS)  
 ↓  
Backend API (FastAPI/Django/Flask)  
 ↓  
Database (SQLite/PostgreSQL)

## 2. Core Entities and Relationships

|  |  |
| --- | --- |
| Entity | Attributes |
| User | id (PK), username, email, password (hashed), role (student/teacher) |
| Assignment | id (PK), title, description, due\_date, created\_by (FK → User) |
| Submission | id (PK), assignment\_id (FK → Assignment), student\_id (FK → User), content, timestamp, file\_url (optional) |

Relationships:  
- A teacher can create many assignments  
- A student can submit many submissions  
- Each submission is linked to one assignment and one student

## 3. API Endpoints

Endpoints for: Teacher creates assignment, Student submits assignment, Teacher views submissions

|  |  |  |  |
| --- | --- | --- | --- |
| Endpoint | Method | Description | Access |
| /signup | POST | Register user with role | Public |
| /login | POST | Login and receive token | Public |
| /assignments | POST | Teacher creates assignment | Teacher |
| /assignments/{id}/submit | POST | Student submits assignment | Student |
| /assignments/{id}/submissions | GET | Teacher views submissions | Teacher |

## 4. Authentication Strategy

- JWT-based authentication  
- Roles embedded in token payload  
- Role-based access control using decorators or dependencies

## 5. Future Scalability

- Use PostgreSQL in production for better scalability  
- Add Redis for caching  
- Split services into microservices  
- Store files on cloud (e.g., AWS S3)  
- Use background tasks/message queues for notifications and async tasks

# Part B – Prototype Implementation

Implemented using FastAPI and SQLite (for demo purposes). All APIs return JSON responses.

## Implemented APIs

1. User signup/login (role-based)  
2. Create assignment (teacher)  
3. Submit assignment (student)  
4. View submissions (teacher)

## Backend Code Snippets (FastAPI)

@app.post("/signup")  
def signup(user: UserCreate, db: Session = Depends(get\_db)):  
 # register user  
  
@app.post("/login")  
def login(form\_data: OAuth2PasswordRequestForm = Depends(), db: Session = Depends(get\_db)):  
 # authenticate and return token  
  
@app.post("/assignments")  
def create\_assignment(data: AssignmentCreate, user: User = Depends(get\_current\_user)):  
 # create assignment (teacher only)  
  
@app.post("/assignments/{assignment\_id}/submit")  
def submit\_assignment(...):  
 # student submits assignment  
  
@app.get("/assignments/{assignment\_id}/submissions")  
def view\_submissions(...):  
 # teacher views submissions

# Frontend (Basic HTML/CSS/JS)

## 1. Assignment Creation (Teacher)

<form id="createForm">  
 <input type="text" placeholder="Title" id="title" />  
 <textarea id="description"></textarea>  
 <input type="date" id="due\_date" />  
 <button type="submit">Create Assignment</button>  
</form>

## 2. Assignment Submission (Student)

<form id="submitForm">  
 <input type="text" placeholder="Assignment ID" id="assignmentId" />  
 <textarea id="content"></textarea>  
 <input type="file" id="fileUpload" />  
 <button type="submit">Submit Assignment</button>  
</form>

## 3. View Submissions (Teacher)

<div id="submissionsList">  
 <!-- Fetched list of submissions will be rendered here -->  
</div>