# Project Feature List & Database Schema

### Module A: Authentication & User Setup (The Foundation)

1. **Secure Registration & Login:** Users sign up using email/password. The system issues a secure token (JWT) to keep them logged in.
2. **Role-Based Access Control (RBAC):** Upon login, the system detects the user's role:

**Students** are routed to the Learning Dashboard.

**Instructors** are routed to the Management Dashboard.

1. **Profile Management:** Users can update their personal details, change passwords, and upload profile pictures.

### Module B: Instructor Portal (Content & Management)

1. **Template-Based Problem Generation (Hybrid):**

**Step 1 - Tag Selection:** Instructor selects parameters via dropdowns: **Topic** (e.g., "Arrays"), **Difficulty** (e.g., "Medium").

**Step 2 - Library Check:** The system automatically searches the database for existing matches.

**Step 3 - Template Injection:** If creating new, the system injects these tags into a **Predefined Prompt Template** (e.g., *"Generate a [Difficulty] Python problem focused on [Topic]..."*).

**Step 4 - Customization:** The instructor sees this pre-filled prompt and can add specific context (e.g., "...using a real-world banking scenario").

**Step 5 - Approval & Save:** Approved problems are saved to the library tagged with the original parameters for future search.

1. **Manual Problem Editor:** Instructors can manually tweak the AI-generated content or create problems from scratch.
2. **Test Case Management:**

**Public Cases:** Visible to students (for testing basic logic).

**Hidden Cases:** Invisible to students (used for grading).

1. **Classroom Creation:** Instructors create a virtual "Class" (e.g., "Data Structures A") and generate a unique join code for students.
2. **Assignment Logic:** Instructors select a problem from the library, assign it to a specific Class, and set a due date.
3. **Instructor Analytics:** A dashboard showing which students are struggling, the average score per problem, and a "Plagiarism Alert" list.

### Module C: Student Portal (The Learning Loop)

1. **Assignment Dashboard:** Students see a list of "To-Do" assignments with deadlines and difficulty levels.
2. **Coding Workspace (IDE):** An in-browser code editor that supports syntax highlighting and auto-completion.
3. **AI Hint System:** If stuck, a student clicks "Get Hint." The system analyzes their *current* code and gives a nudge (without revealing the answer).
4. **Code Execution:**

**Run:** Execute code against Public test cases to check basic correctness.

**Submit:** Execute code against All (Hidden) test cases for a final grade.

1. **Dual-Mode Feedback System:**

**If Code Fails (Debugging):** The AI explains *why* it failed (logic error) and suggests a fix.

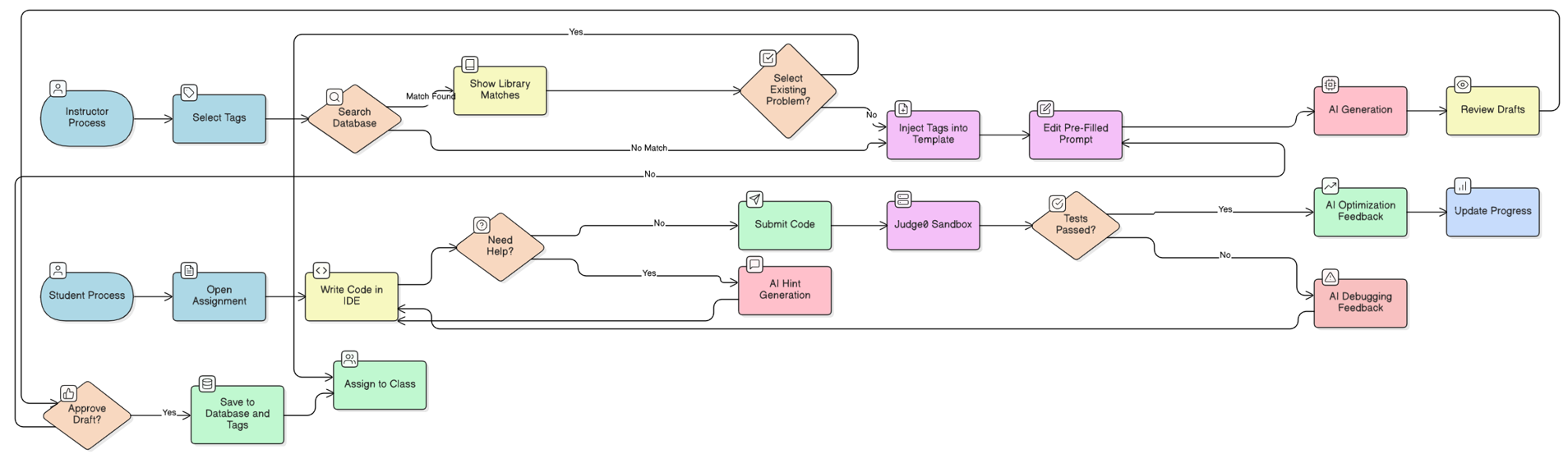
**If Code Passes (Optimization):** The AI provides advice on Time Complexity (Big O) and Code Style improvements.

1. **Progress Tracker:** A personal view for students to see their submission history and improvement over time.

### Module D: Backend Intelligence (The Engine)

1. **Secure Sandbox (Judge0):** Ensures student code runs in an isolated container so it cannot crash the server or steal data.
2. **AST Analysis:** Parses code into a structural tree to detect complexity (O(n^2) vs O(n)) and bad coding practices.
3. **Plagiarism Detection:** Runs the Winnowing algorithm on every new submission to compare it against previous submissions and flag high similarity.

**Implementation Workflow:**

****

## Database Schema

### 1. User & Class Management

| **Table Name** | **Description** | **Columns (Attributes)** |
| --- | --- | --- |
| **users** | Stores user details for everyone. Linked to the login system. | • **id** (Unique ID)  • **email** (Text)  • **full\_name** (Text)  • **role** ("student" or "instructor")  • **avatar\_url** (Link to image)  • **created\_at** (Time) |
| **classrooms** | Represents a specific batch or class group. | • **id** (Unique Class ID)  • **instructor\_id** (Link to *profiles*)  • **name** (e.g., "CS101 Fall")  • **join\_code** (Unique text)  • **created\_at** (Time) |
| **enrollments** | Tracks which student is in which class. | • **id** (Unique ID)  • **classroom\_id** (Link to *classrooms*)  • **student\_id** (Link to *profiles*)  • **joined\_at** (Time) |

### 2. Content & Assignments

| **Table Name** | **Description** | **Columns (Attributes)** |
| --- | --- | --- |
| **problems** | The library of coding questions. | • **id** (Unique Problem ID)  • **title** (Text)  • **description** (Text/Markdown)  • **difficulty** ("Easy", "Medium", "Hard")  • **tags** (Array of Text: ["Arrays", "Python"])  • **usage\_count** (Integer, for popularity sorting)  • **starter\_code** (Boilerplate)  • **solution\_code** (Hidden solution)  • **created\_by** (Link to *profiles*) |
| **test\_cases** | Inputs/Outputs to grade the code. | • **id** (Unique Case ID)  • **problem\_id** (Link to *problems*)  • **input\_data** (Text)  • **expected\_output** (Text)  • **is\_hidden** (True/False - True means hidden from student) |
| **assignments** | Links a problem to a class with a deadline. | • **id** (Unique Assignment ID)  • **classroom\_id** (Link to *classrooms*)  • **problem\_id** (Link to *problems*)  • **due\_date** (Time)  • **status** ("Active" or "Archived") |

### 3. Execution & Intelligence

| **Table Name** | **Description** | **Columns (Attributes)** |
| --- | --- | --- |
| **submissions** | Records every attempt a student makes. | • **id** (Unique Submission ID)  • **student\_id** (Link to *profiles*)  • **assignment\_id** (Link to *assignments*)  • **code\_content** (The code they wrote)  • **status** ("Accepted", "Wrong Answer")  • **execution\_time** (Number, e.g., 0.05s)  • **submitted\_at** (Time) |
| **ai\_feedback\_logs** | Stores the AI analysis so we don't re-generate it. | • **id** (Unique Log ID)  • **submission\_id** (Link to *submissions*)  • **feedback\_type** ("Debugging" or "Optimization")  • **message** (The AI text explanation)  • **complexity\_score** (e.g., "O(n^2)") |
| **plagiarism\_flags** | Alerts if code is copied. | • **id** (Unique Flag ID)  • **submission\_id\_a** (The new code)  • **submission\_id\_b** (The original code it matches)  • **similarity\_score** (Percentage, e.g., 0.95)  • **detected\_at** (Time) |