1 User Model

Purpose: Stores user information, authentication data, and user progress stats.

Fields:

- `user_id`: Unique identifier (Primary Key)
- `name`: User's full name
- `email`: User's email address
- `password_hash`: Hashed password (if not using OAuth2)
- `google_id`: Google OAuth ID (if using Google login)
- `profile_picture`: URL to profile image
- `role`: (Enum: Admin, User, etc.)
- `date_joined`: Date of account creation
- `last_login`: Last login timestamp
- `stats`: Foreign Key to `UserStats` (user performance stats)

Relationships:

- **1-to-many with UserStats**: A user has many performance stats entries.
- **1-to-many with ForumPosts**: A user can create many posts in the forum.

---

2 UserStats Model

Purpose: Stores user performance stats such as coding progress, ranking, etc.

Fields:

- `stats_id`: Unique identifier (Primary Key)
- `user_id`: Foreign Key to User
- `problems_solved`: Number of problems solved
- `time_spent`: Total time spent solving problems
- `average_score`: Average score across problems
- `rank`: Ranking in leaderboard
- `study_plan`: JSON or text storing AI suggestions for improvement
- `last_updated`: Timestamp for the last stats update

Relationships:

- **Many-to-1 with User**: Each stat entry is tied to one user.

---

3 Problem Model

Purpose: Stores coding problems (including LeetCode integration).

Fields:

- `problem_id`: Unique identifier (Primary Key)
- `title`: Problem title
- `description`: Detailed problem description
- `difficulty`: Difficulty level (Easy, Medium, Hard)
- `tags`: List of tags (e.g., DP, Greedy)
- `input`: Example input
- `output`: Example output
- `constraints`: Problem constraints
- `leetcode_id`: (Optional) Foreign Key to LeetCode API or ID
- `creator_id`: Foreign Key to User (Admin who created the problem)
- `created_at`: Timestamp when problem was created

Relationships:

- **Many-to-1 with User**: The problem is created by an admin.
- **1-to-many with Submissions**: A problem can have multiple submissions.
- **1-to-many with ProblemTags**: A problem can have many tags.

---

4 ProblemTag Model

Purpose: Stores tags associated with coding problems.

Fields:

- `tag_id`: Unique identifier (Primary Key)
- `problem_id`: Foreign Key to Problem
- `tag`: Tag name (e.g., "Dynamic Programming")

Relationships:

- **Many-to-1 with Problem**: A tag belongs to one problem.

---

5 Submission Model

Purpose: Stores code submissions by users for evaluation.

Fields:

- `submission_id`: Unique identifier (Primary Key)
- `problem_id`: Foreign Key to Problem
- `user_id`: Foreign Key to User
- `code`: The user-submitted code
- `language`: Language of the submitted code (e.g., Python, C++)
- `status`: (Enum: Pending, Compiling, Running, Completed, Failed)
- `result`: Output of the code execution
- `execution_time`: Time taken to execute the code
- `submitted_at`: Timestamp when the code was submitted

Relationships:

- **Many-to-1 with Problem**: A submission is tied to one problem.
- **Many-to-1 with User**: A submission is made by one user.

---

6 InterviewSession Model

Purpose: Stores details of mock interview sessions.

Fields:

- `session_id`: Unique identifier (Primary Key)
- `user_id`: Foreign Key to User
- `interview_type`: (Enum: Technical, Behavioral, Coding)
- `start_time`: Timestamp when interview starts
- `end_time`: Timestamp when interview ends
- `status`: (Enum: Pending, In Progress, Completed)
- `feedback`: AI-generated feedback after completion

Relationships:

- **Many-to-1 with User**: The interview is for a particular user.
- **1-to-many with InterviewQuestion**: An interview session contains multiple questions.

---

7 InterviewQuestion Model

Purpose: Stores questions asked during a mock interview session.

Fields:

- `question_id`: Unique identifier (Primary Key)
- `session_id`: Foreign Key to InterviewSession
- `question`: The interview question text
- `answer`: The expected answer or hint
- `submitted_answer`: The user's answer to the question
- `feedback`: AI-generated feedback for the answer

Relationships:

- **Many-to-1 with InterviewSession**: A question belongs to one interview session.

---

8 ForumPost Model

Purpose: Stores forum posts in the community discussion section.

Fields:

- `post_id`: Unique identifier (Primary Key)

- `user_id`: Foreign Key to User
- `title`: Title of the post
- `content`: Content of the post
- `created_at`: Timestamp when post was created
- `updated_at`: Timestamp when post was last updated

Relationships:

- **Many-to-1 with User**: A post is written by one user.
- **1-to-many with PostComment**: A post can have multiple comments.

---

9 PostComment Model

Purpose: Stores comments on forum posts.

Fields:

- `comment_id`: Unique identifier (Primary Key)
- `post_id`: Foreign Key to ForumPost
- `user_id`: Foreign Key to User
- `content`: Comment content
- `created_at`: Timestamp when comment was created

Relationships:

- **Many-to-1 with ForumPost**: A comment is tied to one post.
- **Many-to-1 with User**: A comment is made by one user.

---

10 CollaborationSession Model

Purpose: Stores live pair programming collaboration sessions.

Fields:

- `session_id`: Unique identifier (Primary Key)
- `host_user_id`: Foreign Key to User (Host)
- `code`: Shared code being worked on
- `status`: (Enum: Pending, Active, Completed)
- `start_time`: Timestamp when session starts
- `end_time`: Timestamp when session ends

Relationships:

- **Many-to-1 with User**: A session has one host.
- **Many-to-many with User**: A session can have multiple participants.

---

11 TechNews & JobPost Model

Purpose: Stores tech news and job posts fetched from external sources.

Fields:

- `news_id`: Unique identifier (Primary Key)
- `title`: News title
- `content`: News content or job description
- `source`: (Enum: LinkedIn, Indeed, CodeChef, etc.)
- `url`: Link to the original source
- `type`: (Enum: Job, News, Promotion)
- `created_at`: Timestamp when the news/job was fetched

Relationships:

- **N/A**: This is a standalone model, not related to other models directly.

High-Level Relationships Overview

1. **User** ↔ **UserStats**: One-to-many (One user can have many stats entries)
2. **User** ↔ **ForumPost**: One-to-many (One user can create many posts)
3. **Problem** ↔ **Submission**: One-to-many (A problem can have many submissions)
4. **Problem** ↔ **ProblemTag**: One-to-many (A problem can have many tags)
5. **User** ↔ **Submission**: One-to-many (A user can make many submissions)
6. **InterviewSession** ↔ **InterviewQuestion**: One-to-many (One session can have many questions)
7. **ForumPost** ↔ **PostComment**: One-to-many (A post can have many comments)
8. **User** ↔ **CollaborationSession**: Many-to-many (A user can join many sessions)
