# High Level Design (HLD) - Online Judge Platform

## Overview

A web-based **Online Judge** platform where users can register, solve coding problems, participate in contests, submit solutions, and receive automatic evaluation based on correctness, time, and memory. Includes a social system, leaderboard, and problem management for admins.

---

## Database Design

### 1. users

```
{
  _id: ObjectId,
  username: String,
  email: String,
  passwordHash: String,
  role: "user" || "admin",
  emailVerified: Boolean,
  verificationToken: String,
  resetPasswordToken: String,
  resetTokenExpiry: Date,
  joinedAt: Date
}
```

### 2. problems

```
{
  _id: ObjectId,
  title: String,
  description: String,
  difficulty: "Easy" || "Medium" || "Hard",
  tags: [String],
  type: "rated" || "practice" || "contest",
  points: Number,
  contestId: ObjectId (optional),
  testCases: [
    { input: String, expectedOutput: String }
  ],
  createdBy: ObjectId,
  createdAt: Date
}
```

### 3. submissions

```
{
  _id: ObjectId,
  userId: ObjectId,
  problemId: ObjectId,
  contestId: ObjectId (optional),
  code: String,
  language: String,
  verdict: "AC" | "WA" | "TLE" | "MLE" | "RTE" | "CE",
  score: Number,
  time: Number,
  memory: Number,
  status: "Pending" | "Running" | "Evaluated",
  submittedAt: Date
}
```

### 4. contests

```
{
  _id: ObjectId,
  name: String,
  startTime: Date,
  endTime: Date,
  problems: [ObjectId],
  participants: [ObjectId]
}
```

### 5. friends

```
{
  _id: ObjectId,
  userId: ObjectId,
  friendId: ObjectId,
  status: "pending" | "accepted",
  createdAt: Date
}
```

---

## User Interface Design

### Screen 1: Home

- Platform info
- Sign up/Login buttons

### Screen 2: Authentication

- Login (email + password)
- Sign up (username + email + password)
- Forgot Password (email input)
- Email verification message

### Screen 3: Dashboard

- View/Edit profile
- Practice & rated problems
- Join contests
- View statistics & leaderboard

### Screen 4: Problem Page

- Problem info
- Language selector & code editor
- Run & Submit buttons
- Output and verdict section

### Screen 5: Leaderboards

- Global leaderboard
- Contest leaderboard
- Friends leaderboard

### Screen 6: Submissions History

- List of all past submissions with verdicts, time, memory

### Screen 7: Friends System

- Send/accept friend requests
- Search friends
- View friend list & their stats

### Screen 8: Admin Panel

- Create/Edit problems
- Manage contests

---

## API Route Design

### Auth

```
POST /api/auth/signup
POST /api/auth/login
POST /api/auth/verify-email
POST /api/auth/resend-verification
```

```
POST /api/auth/forgot-password
POST /api/auth/reset-password
```

### User

```
GET /api/user/me
GET /api/user/stats
```

### Problems

```
GET /api/problems/practice
GET /api/problems/rated
GET /api/problems/search?title=...
GET /api/problems/filter?tag=...
GET /api/problems/:id
POST /api/problems/create (admin)
```

### Submissions

```
POST /api/submissions/run
POST /api/submissions/submit
GET /api/submissions/history
```

### Contests

```
GET /api/contests/:id/problems
```

### Leaderboards

```
GET /api/leaderboard/global
GET /api/leaderboard/contest/:id
GET /api/leaderboard/friends
```

### Friends

```
POST /api/friends/send-request/:friendId
POST /api/friends/accept-request/:requestId
GET /api/friends/list
GET /api/friends/search?name=...
```

---

## Controllers Design

### 1. authController

- signup
- login
- verifyEmail
- resendVerification
- forgotPassword
- resetPassword

### 2. userController

- getProfile
- getStats

### 3. problemController

- getPracticeProblems
- getRatedProblems
- getProblemById
- getContestProblems
- searchProblems
- filterProblems
- createProblem (admin)

### 4. submissionController

- runCode
- submitSolution
- getSubmissionHistory

### 5. leaderboardController

- getGlobalLeaderboard
- getContestLeaderboard
- getFriendsLeaderboard

### 6. friendsController

- sendRequest
- acceptRequest
- getFriendsList
- searchFriends

---

## Code Execution & Evaluation Workflow

### Run Code (Sample Input)

1. User clicks 'Run'
2. Frontend sends code, language to `/api/submissions/run`

3. Backend adds job to queue
4. Worker picks job → runs in Docker → returns output

### Submit Code (Evaluation)

1. User clicks 'Submit'
2. Frontend sends code, language, problemId to `/api/submissions/submit`
3. Backend creates `submission` (status: Pending)
4. Worker evaluates code against all test cases
5. Worker updates submission with: verdict, time, memory, score, status: Evaluated

---

## Additional Features

### Verdicts

- AC (Accepted)
- WA (Wrong Answer)
- TLE (Time Limit Exceeded)
- MLE (Memory Limit Exceeded)
- RTE (Runtime Error)
- CE (Compilation Error)

### Dashboard Stats

- Total submissions
- Accepted problems
- Avg time & memory
- Graph of verdict distribution

---

## Architecture Notes

- Backend: Node.js + Express
- Database: MongoDB
- Frontend: React.js
- Worker: Docker containers to isolate code execution
- Job Queue: Bull / Redis
- Email Service: Nodemailer (SMTP) or external (SendGrid, Mailgun)
- Authentication: JWT

---

## Future Enhancements

- Discussion forums
- Editorials for problems
- Badges & Achievements
- Mobile app support
- WebSocket notifications for real-time verdicts

---

**End of Document**