# Hackathon Platform — Approach & Design Document

## 1. Objective

Build a platform that connects **recruiters** and **candidates** for hackathons. Recruiters can create hackathons via a dedicated UI; created hackathons are surfaced on candidates' dashboards according to their profiles. Each hackathon has its own dashboard (registered participants, latest submissions, leaderboard). Candidates can view details, register, and submit via an in-app form (name, email, documentation, repo link, etc.). GitHub repos are monitored via GitHub API integration and mentors review submissions.

## 2. Scope

- **Recruiter portal**: create/manage hackathons, view participants/submissions, leaderboard and analytics.
- **Candidate portal**: discover hackathons matching profile, view details, register, submit, view personal hackathon dashboard and leaderboards.
- **Admin functions**: user management, content moderation, system settings.
- **Integrations**: GitHub API (verify/retrieve repo metadata)

## 3. High-level User Flows

1. Recruiter signs in → opens "My Hackathons" → clicks "Create Hackathon" → fills form → publishes hackathon.
2. Platform evaluates candidate profiles and surfaces matching hackathons on candidate dashboard.
3. Candidate views hackathon detail → registers → dashboard shows registered hackathons.
4. Candidate clicks **Submit** → fills the submission form + repo link → submission stored and mentor-notified.
5. Mentor/judge reviews submission; scoring recorded → leaderboard updated.

# 4. Key Features

- Hackathon creation form (rich description, rules, eligibility, technologies, prizes).
- Candidate discovery & profile-based matching.
- Registration for desired hackathon
- Submission handling with GitHub repo validation.
- Judging & scoring engine + leaderboard.

# 5. Backend APIs

## Recruiter: Hackathon Management

- `GET /api/recruiter/hackathons` → List recruiter's hackathons.
- `POST /api/recruiter/hackathons` → Create new hackathon.
- `GET /api/hackathons/{hackathonId}` → Get hackathon details.
- `PUT /api/recruiter/hackathons/{hackathonId}` → Update hackathon.
- `DELETE /api/recruiter/hackathons/{hackathonId}` → Delete hackathon.

## Registration & Teams

- `POST /api/hackathons/{hackathonId}/register` → Candidate registers.
- `GET /api/hackathons/{hackathonId}/registrations` → View registered participants.
- `POST /api/hackathons/{hackathonId}/teams` → Create team.

## Submissions

- `POST /api/hackathons/{hackathonId}/submissions` → Submit project.
- `GET /api/hackathons/{hackathonId}/submissions` → View all submissions.
- `GET /api/submissions/{submissionId}` → Get submission detail.
- `POST /api/submissions/{submissionId}/comment` → Judge/mentor comment.

## Judging & Scoring

- `POST /api/submissions/{submissionId}/score` → Submit judge's score.
- `GET /api/hackathons/{hackathonId}/leaderboard` → View leaderboard.

### GitHub Integration

- Verify repos linked by candidates.
- Fetch repo metadata (commits, contributors, languages).

# 6. Frontend UIs / Pages

## Recruiter-facing

- Recruiter Dashboard (list hackathons + create new).
- Hackathon Creation/Edit Form (basic details, rules, prizes, eligibility, technologies).
- Hackathon Admin Dashboard (participants, submissions, leaderboard, settings).

## Candidate-facing

- Discover Page (hackathons matching profile + filters).
- Hackathon Detail Page (full info + register option).
- Candidate Dashboard (registered hackathons, status, submissions).
- Submission Form (repo link, documentation, demo link).
- Leaderboard (rankings, scores).

## Shared

- Login/Signup (with social login).
- Notifications Center.
- Profile Page.
- Admin Panel.

# 7. Database Schema (Core Tables)

## `hackathons`

- id, creator_id, title, banner_url, description, start_at, end_at, instructions, eligibility, allowed_technologies, prizes, settings, status.

### `registrations`

- id, hackathon_id, user_id, eligibility

### `submissions`

- id, hackathon_id, team_id, title, description, repo_url, doc_url, demo_url, repo_metadata

### `notifications`

- id, user_id, type, payload, read.

# 8. GitHub API Integration

- Validate submitted repo ownership.
- Fetch languages, commits, last update, contributors.
- Store metadata alongside submission.
- Optionally set up webhook for live push events.