📘 PRODUCT REQUIREMENTS DOCUMENT (PRD)

1. 📌 Problem Statement

Students frequently face difficulties in finding suitable project partners who share similar skills, interests, availability, and commitment levels. Existing solutions rely on informal methods such as WhatsApp groups, classroom announcements, or random selection. These approaches often result in mismatched teams, poor collaboration, spam communication, and incomplete projects.

There is no centralized, structured, and privacy-aware platform that helps students discover compatible project partners efficiently and reliably.

---

2. 🎯 Objective

To build a focused, web-based platform that enables students to:

- Create structured collaboration profiles
- Showcase skills, availability, and project experience
- Discover potential project partners
- Send and manage collaboration requests
- Connect securely only after mutual acceptance

The goal is to **reduce friction, noise, and misuse in team formation**, while improving collaboration outcomes.

---

3. 👤 Target Users & Roles

Primary Users

- College students
- Hackathon participants
- Final-year / mini-project students

User Roles

Student (Primary Role)

- Sign up and log in
- Create and update profile
- Discover partners
- Send and respond to collaboration requests
- View collaboration status
- Access contact details only after mutual acceptance

Admin (Moderation Role – MVP Scope)

- Review reported users
- Handle misuse and spam
- Warn, suspend, or dismiss reported accounts

> The admin role is intentionally limited to moderation and safety to keep the system lightweight and hackathon-friendly.

---

4. 💡 Proposed Solution

CollabX is a **privacy-first, intent-based collaboration platform** where students can:

1. Sign up and log in
2. Create a structured partner profile
3. Browse other student profiles
4. Send explicit collaboration requests
5. Accept or reject requests
6. Track collaboration status on a dashboard
7. Contact partners only after mutual acceptance

This removes random messaging and enforces clear intent.

---

5. 🔄 User Flow

1. User lands on the website
2. User signs up / logs in
3. User completes profile creation
4. User navigates to “Discover Partners”
5. User views partner cards
6. User sends a collaboration request
7. Recipient receives request
8. Recipient accepts or rejects request
9. Match status updates on both dashboards
10. After acceptance, contact details (Mobile & LinkedIn) are revealed

---

6. 🖥️ Screens & Pages

1️⃣ Landing Page

- Product overview
- Key benefits
- Differentiation from WhatsApp
- Call-to-action: **Find Project Partners**

2️⃣ Authentication Pages

- Sign Up (Name, Email, Password, Confirm Password)
- Log In (Email, Password)

3️⃣ Profile Creation Page (Single Page)

- Name
- Short bio
- Skills (tags)
- Availability
- Project links (GitHub / Live)
- Contact details (hidden until match)

4️⃣ Discover Partners Page

- Partner cards showing:

    - Name
    - Skills
    - Availability
    - Projects
- “Connect” button
- Disabled state for sent requests

5️⃣ Dashboard Page

- **Matches** (accepted users with contact details)
- **Received Requests** (accept / reject)
- **Sent Requests** (pending / accepted / rejected status)

6️⃣ Admin Portal (Moderation Only)

- Admin login
- List of reported users
- Report detail view
- Actions: Warn / Suspend / Dismiss

Admin access is available via a separate route (`/admin`).

---

7. ⚙️ Functional Requirements (MVP)

Authentication

- User sign up with email & password
- User log in
- JWT-based authentication

Profile Management

- User can create or update profile
- Profile stored in database
- User can add skills, availability, and projects
- User can add contact details
- Contact details visible only after successful match

Partner Discovery

- User can view other student profiles
- User cannot view their own profile in discovery

Collaboration Requests

- User can send collaboration requests
- User can accept or reject requests
- Request status updates accordingly

Dashboard

- User can view sent requests
- User can view received requests

- Accepted requests appear as matches

Match & Contact

- After acceptance, both users can view:
  - Mobile number
  - LinkedIn profile

Admin & Moderation

- Admin can log in securely
- Admin can view reported users
- Admin can review misuse reports
- Admin can warn, suspend, or dismiss users
- Suspended users cannot send or receive requests

---

8. 🔐 Safety, Misuse & Consequences

8.1 Potential Consequences

- Users may exaggerate skills
- Some users may be inactive or unresponsive
- High-profile users may receive many requests

These are mitigated by structured profiles, availability clarity, and explicit accept/reject actions.

8.2 Possible Misuse

- Spam collaboration requests
- Fake or misleading profiles
- Harassment after contact exchange

8.3 Misuse Prevention & Mitigation

- No direct contact before mutual acceptance
- Contact details hidden by default
- Users can report misuse
- Admin moderation for flagged users

8.4 Admin Governance

Admins act as moderators and:

- Review reports
- Take corrective action
- Maintain platform trust

This ensures safety without adding unnecessary complexity.

---

9. 🧩 Non-Functional Requirements

- Responsive UI (desktop priority)
- Clean, intuitive UX
- Fast page loads

- Secure authentication
- Minimal backend complexity

---

10. 🚫 Out of Scope (Explicitly Not Building)

- In-app chat or messaging
- AI-based matching
- Group/team creation
- Notifications (email / push)
- Resume or file uploads
- Advanced admin analytics
- Automated moderation
- Multi-level admin roles

---

11. 🛠️ Tech Stack

Frontend

- HTML
- CSS
- JavaScript
- Figma Make → Code

Backend

- Node.js
- Express.js
- MongoDB
- JWT Authentication

Tools

- GitHub
- Postman
- VS Code
- Figma Make

---

12. 🗄️ Database Design

User Collection

```
{name:String,email:String,password:String,skills: [String],availability:String,bio:String,projects: [String],contact: {mobil
e:String,linkedin:String
  },isSuspended:Boolean,warningCount:Number
}
```

Request Collection

```
{fromUserId:ObjectId,toUserId:ObjectId,status:"pending" |"accepted" |"rejected",createdAt:Date
}
```

Reports Collection

```
{reportedUserId:ObjectId,reportedByUserId:ObjectId,reason:String,status:"pending" |"resolved",actionTaken:"warned"
|"suspended" |"dismissed",createdAt:Date
}
```

---

13. 🔌 API Endpoints (MVP)

Auth

- POST `/signup`
- POST `/login`

Profile

- GET `/me`
- PUT `/me`

Partner Discovery

- GET `/partners`

Requests

- POST `/request`
- GET `/requests`
- POST `/request/:id/respond`

Admin

- POST `/admin/login`
- GET `/admin/reports`
- POST `/admin/reports/:id/action`

---

14. 🧪 Success Criteria (Hackathon)

The project is successful if:

- Users can sign up and log in
- Profiles can be created and viewed
- Partner discovery works
- Collaboration requests work end-to-end
- Dashboard reflects correct request status
- Admin can review and act on misuse reports
- Demo flow is smooth and understandable

---

15. 🔮 Future Scope (For Judges)

- AI-based partner matching
- Team formation (more than 2 users)
- In-app chat
- Skill verification
- College-wide collaboration
- Hackathon-specific matching

---

16. 🏁 Final Notes

CollabX prioritizes:

- **Intent over noise**
- **Privacy over exposure**
- **Clarity over complexity**
- **Safety over unchecked access**

The scope is intentionally controlled to ensure successful completion within hackathon timelines while demonstrating real-world product thinking.