**High-Level Design Document for: GainIt**

## **1. Introduction**

GainIt is a web-based platform designed to bridge the gap between students, mentors, and nonprofit organizations by providing hands-on experience through real-world projects. This document outlines the high-level design, including system architecture, main software modules, and database structure.

## **2. System Architecture**

The system follows a **client-server architecture**, consisting of:

* **Frontend**: A web-based user interface built using React.js for dynamic and responsive interactions.
* **Backend**: implemented in C# (.NET Core), handling business logic and database interactions.
* **Database**: A SQL-based database - PostgreSQL to store user data, project details, and interactions.
* **Authentication**: OAuth 2.0-based authentication with Google Sign-In and manual email/password registration.
* **Hosting & Deployment**: Cloud-based deployment using Azure.

## **3. Main Software Modules**

### **3.1 User Management Module**

* User Authentication & Authorization
* Role-based access control (Students, Mentors, Organizations, Admins)
* Profile management and skill tagging

### **3.2 Project Management Module**

* Project creation and configuration
* Project browsing and matching algorithm
* Team formation and mentor involvement
* Project status tracking (Pending, In Progress, Completed)

### **3.3 Collaboration & Task Management Module**

* Integration with GitHub for version control
* Task assignment and tracking
* Chat and discussion forums

### **3.4 Knowledge Sharing & Community Module**

* Q&A Forum for discussions on technical topics
* Upvoting and accepted answers for knowledge curation
* AI-driven smart recommendations for best practices

### **3.5 Gamification & Achievements Module**

* Trophy system rewarding user engagement
* Milestone tracking (e.g., first project completion, mentor guidance)

### **3.6 Notifications & Alerts Module**

* Real-time notifications for project updates, team invitations, and forum replies
* Email and in-platform alerts

### **3.7 Admin & Moderation Module**

* User and content moderation
* Platform analytics and performance monitoring

**3.8** **Smart Matching & Recommendation Module:**

* Personalized project suggestions based on skills and experience.
* Role-based matching for balanced team formation.
* Mentor recommendations tailored to project needs.
* Collaboration insights based on past teamwork.

### **3.9 Feedback & Performance Evaluation Module:**

* User feedback and performance ratings after projects.
* Cumulative reputation score based on received feedback.
* AI-driven insights for personal and professional growth.
* Skill endorsements and recognition for contributions.

## **4. Database Design (Key Tables)**

| **Table Name** | **Description** |
| --- | --- |
| Users | Stores user details, roles, and authentication info |
| Projects | Contains project details, team members, and status |
| Tasks | Tracks assigned tasks, progress, and deadlines |
| Discussions | Stores forum posts, replies, and voting system |
| Achievements | Tracks user progress and awarded trophies |

## **5. External Integrations**

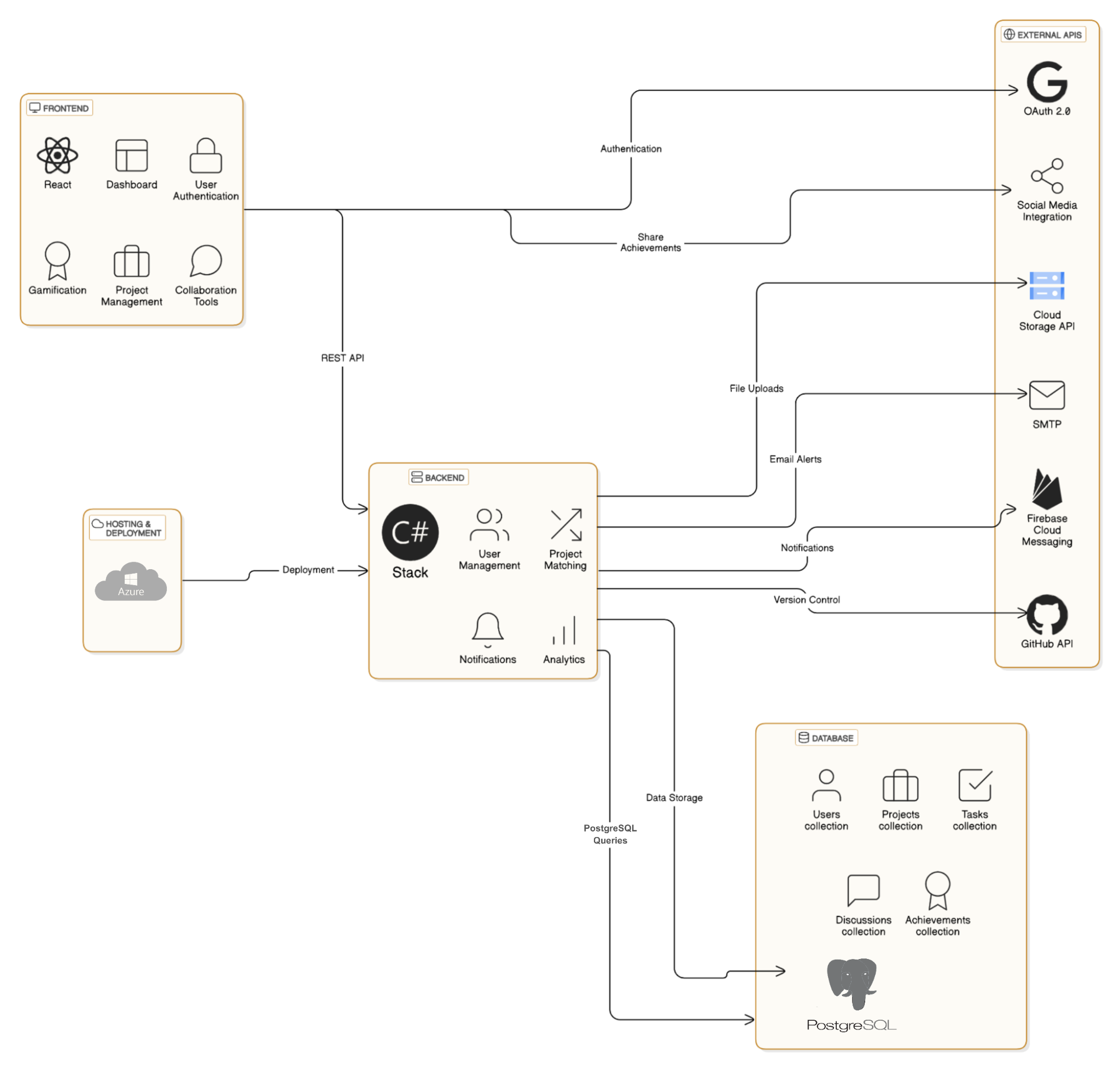
* **OAuth Authentication** (Google Sign-In)
* **GitHub API** for project repositories and version control
* **Cloud Storage** (Azure Blob Storage) for document and image uploads
* **Email & Notifications** (SMTP, Firebase Cloud Messaging)

## **6. Future Enhancements**

* AI-driven project matching
* Mobile app development (React Native or Flutter)
* Smart Summary: AI-generated professional overview of each user's activity on the platform

## **7. Conclusion**

This high-level design ensures a scalable, user-friendly, and efficient platform for collaborative learning and professional growth. Future iterations will enhance features based on user feedback and technological advancements.

Architecture Diagram:

**MVP Development Scope (To Be Completed by June):**

The initial development phase will focus on building the core functionality of the platform, ensuring it is operational and allows basic user interactions.

1. System Infrastructure:
   1. Database setup – Creating tables for users, projects, and registrations.
   2. Development environment – Setting up GitHub, a shared development environment and defining the system architecture.
   3. Basic homepage – Displays general information and navigates to different sections of the platform.
   4. User registration and authentication -

* Login via Gmail and/or manual registration with basic validation.
* Session/Auth management (keeping users logged in).

1. Basic Project Management:
   1. Project creation – A form for adding project details: name, description, technologies, status.
   2. Viewing open projects – A screen displaying available projects that users can join.
   3. Joining a project – A "Join" button that either sends a request to the project owner or directly adds the user (based on project settings).
   4. Basic project page – Displays project details such as name, description, technologies, and team members.
2. Profile Pages and User Management:
   1. Basic user profile – Displays profile picture, name, and user type: general user, mentor.
   2. Profile editing – Allows users to add skills, areas of interest, and professional status.
   3. Viewing other users' profiles – Displays basic information without revealing sensitive details.
3. Design:
   1. Interface & Navigation – Designing a clean, intuitive, and modern user experience.
   2. Design System – Establishing consistent styles, colors, and UI components.
   3. Wireframes & Prototypes – Creating initial layouts and interactive mockups.