

**IT 309 SOFTWARE ENGINEERING**

PROJECT DOCUMENTATION

IBU STARTUP

Prepared by:

**Sead Mašetić**

**Fikret Zajmović**

Proposed to:

**Nermina Durmić, Assist. Prof. Dr.**

**Lejla Breščić, Teaching Assistant**

**Ajla Korman, Teaching Assistant**

15.06.2025.

TABLE OF CONTENTS

[**1. Introduction 4**](#)

[1.1. About the Project 4](#_30j0zll)

[1.2. High-level Plan 4](#_okixkz7zwn7)

[Version 1.0 - Minimum Viable Product (MVP) 5](#_jye5899e8g07)

[Release Date: Q1 2025 5](#_lfuqjkg3h865)

[Goal: 5](#_hf2w4lhpo66e)

[Key Features: 5](#_g8h0apfmoci2)

[Success Metrics: 6](#_a8je9mckuma8)

[Version 1.1 - Feature Enhancements & Expansion 6](#_mfp0ckifrz09)

[Release Date: Q2 2025 6](#_npf2yr43hhxi)

[Goal: 6](#_81ba60atucm7)

[Key Enhancements: 6](#_yabl86dhm3u5)

[Success Metrics: 7](#_hlr2oukxmqwh)

[Version 1.2 - Investor & Funding Expansion 7](#_cirabo4uuqqe)

[Release Date: Q3 2025 7](#_5bouztm5tkf7)

[Goal: 7](#_9qdyyx6ivob3)

[Key Features: 7](#_qmg4tck88t2w)

[Success Metrics: 8](#_f7nlrprqjaps)

[Version 1.3 - Monetization & Scaling 8](#_s5iguuxylj2d)

[Release Date: Q4 2025 8](#_zfnypjntvjbw)

[Goal: 8](#_v5x8ua71socl)

[Key Features: 8](#_saf7ovtp4d8z)

[Success Metrics: 9](#_p5aj32h5fqjq)

[1.3. Project Requirements 9](#_6wsm1iak7lhc)

[1.4. UML diagrams 20](#_u1ejpbf1br2u)

[**2. Project Structure 25**](#)

[2.1. Technologies 25](#_1fob9te)

[Frontend: 25](#_ppeorfyccvb2)

[Backend: 25](#_skyseuf4yg76)

[Database: 25](#_dhaxg27vwzwf)

[Other Tools & Technologies: 25](#_v7snnhhqw4cr)

[2.2. Architectural Pattern 26](#_w68xlj6llfxc)

[1. Controller Layer (rest/) 26](#_29ihwwz35tvh)

[2. Service Layer (service/) 26](#_5ricgnjkh887)

[3. Repository Layer (repository/) 26](#_4yk285ws5igk)

[4. Model Layer (model/) 26](#_g4aqrwdruof)

[5. DTO Layer (dto/) 27](#_vwq754dx8f6y)

[1. Component-Based Structure 27](#_tauy18j5qklb)

[2. Routing with React Router 27](#_e05w4kg0v38t)

[3. Service Layer (API abstraction) 27](#_7citomoq0c9a)

[4. Styling with Bootstrap 28](#_30f42xqex7s0)

[Why 28](#_9m71w8ny77op)

[**2.3. Database Entities 28**](#)

[**2.4. Design Patterns 29**](#)

[1. Singleton Pattern 29](#_lx1ag28uylrg)

[2. Data Transfer Object (DTO) Pattern 29](#_ayv6ugqjblv0)

[3. Repository Pattern 29](#_xx326yy5fnhi)

[4. Mapper (Adapter) Pattern 30](#_nu52w8dbvir)

[5. Dependency Injection Pattern 30](#_lk7bscm80jzh)

[2.5. Project Functionalities and Screenshots 30](#_eg3x741w2j86)

[**2.6. Tests 30**](#)

[**3. Conclusion 31**](#)

[**4. Individual Contributions 31**](#_t9zh9jxfus91)

# 1. Introduction

<https://github.com/seomasha/it-309-se>

## 1.1. About the Project

**IBU Startup** is a web-based startup management tool designed specifically for students at International Burch University. The application helps students create, manage, and showcase their startup ideas while connecting with potential collaborators, job applicants, and investors within the university ecosystem. It includes features like startup registration, team and job management, user roles (founders, investors, students), and a clean interface for exploring active projects. The goal is to encourage innovation and entrepreneurship among students by providing a centralized platform for startup growth and networking.

**Deployed at:** <https://it-309-se.onrender.com>

## 1.2. High-level Plan

**1.2.1. Product roadmap**

| DATE | Q1 - 2025 | Q2 - 2025 |
| --- | --- | --- |
| NAME | IBU Startup 1.0 | IBU Startup 1.1 |
| GOAL | Build and launch the Minimum Viable Product (MVP) to gather feedback and validate core features. | Refine the platform based on MVP feedback, improve user experience, and test with a larger audience. |
| FEATURES | - User registration and profiles  - Startup creation and management  - Networking (adding friends)  - Job posts | - Enhanced search/filter for startups, jobs, and users  - In-app messaging system for real-time communication  - Dashboard for startup analytics (views, applications, etc.)  - Allows users to showcase their skills and receive endorsements from peers, helping startups and recruiters identify suitable collaborators. |
| METRICS | - At least 100 registered users by the end of the quarter  - 10 startups created on the platform  - 20 job posts published by startups  - Collect at least 50 responses from user feedback surveys | - At least 300 active users by the end of the quarter  - 50% increase in startup creation (15+ startups total)  - Resolve 90% of reported bugs by the end of the quarter |

| DATE | Q3 - 2025 | Q4 - 2025 |
| --- | --- | --- |
| NAME | IBU Startup 1.2 | IBU Startup 1.3 |
| GOAL | Onboard companies and investors, enabling startups to showcase their ideas and attract funding or mentorship. | Scale platform usage, boost engagement, and begin generating revenue. |
| FEATURES | - Investor profiles and dashboards for managing investment opportunities and advanced startup analytics  - Startup pitch sections with video and project details  - Algorithm to connect startups with relevant investors  - In-app collaboration tools for startup teams to share documents and manage tasks. | - Event integration(hackathons, workshops)  - Subscription plans for premium features (advanced analytics, more startup posts)  - Advertising options for startups and investors  - Mentorship programs connecting experienced professionals with startups. |
| METRICS | - Onboard at least 20 companies/investors to the platform  - At least 5 startups receive funding or mentorship  - At least 50% engagement from investors on the platform | - 1,000+ active users on the platform  - 500+ startup-related posts  - 50+ premium subscribers to the platform  - $5,000+ in revenue generated from subscriptions and ads |

**1.2.2. Product release**

## Version 1.0 - Minimum Viable Product (MVP)

### Release Date: Q1 2025

### Goal:

Launch the core platform to gather user feedback and validate essential features.

### Key Features:

● **User Registration & Onboarding**

○ Secure account creation

○ Profile setup and management

● **Startup Management**

○ Startup creation and dashboard

○ Team member invitations

● **Networking & Communication**

○ Friend requests and messaging

● **Job Marketplace**

○ Posting and applying for startup jobs

● **Search & Discovery**

○ Advanced search for startups, jobs, and users

● **Basic Analytics Dashboard**

○ Profile views, job applications tracking

### Success Metrics:

● 100+ registered users

● 10+ startups created

● 20+ job posts published

● 50+ feedback responses collected

## Version 1.1 - Feature Enhancements & Expansion

### Release Date: Q2 2025

### Goal:

Improve user experience, refine platform features, and expand the user base.

### Key Enhancements:

● **User Experience Improvements**

○ Enhanced onboarding flow

○ Optimized search and filtering options

● **Startup & Investor Engagement**

○ Startup endorsement system

○ Investor profiles and dashboards

● **Real-time Communication Upgrades**

○ Improved in-app messaging and notifications

● **User Growth & Community Features**

○ Trending startup highlights

○ Like and comment functionalities for posts

### Success Metrics:

● 300+ active users

● 50% increase in startup registrations (15+ total)

● 90% of reported bugs resolved

## Version 1.2 - Investor & Funding Expansion

### Release Date: Q3 2025

### Goal:

Onboard investors and companies, enabling startups to showcase their ideas and secure funding.

### Key Features:

● **Investor-Focused Features**

○ Investor profiles and dashboards

○ Startup pitch sections (with video & project details)

○ Smart matchmaking algorithm to connect startups with relevant investors

● **Collaboration & Team Features**

○ In-app document sharing and task management

● **Engagement & Events**

○ Event integration (hackathons, workshops, networking events)

### Success Metrics:

● 20+ investors/companies onboarded

● 5+ startups secure funding or mentorship

● 50% investor engagement on the platform

## Version 1.3 - Monetization & Scaling

### Release Date: Q4 2025

### Goal:

Increase platform revenue and scale usage with premium features and advertising opportunities.

### Key Features:

● **Monetization Strategies**

○ Subscription plans for premium features (advanced analytics, unlimited startup posts)

○ Advertising options for startups and investors

● **Mentorship & Growth Programs**

○ Investor-startup mentorship programs

○ AI-powered startup recommendations

● **Platform Scalability Enhancements**

○ Improved backend performance for scalability

○ Cross-platform compatibility updates

### Success Metrics:

● 1,000+ active users

● 500+ startup-related posts

● 50+ premium subscribers

● $5,000+ in revenue from subscriptions and ads

## 1.3. Project Requirements

**User stories**

**Theme: User**

**Epic 1: User Registration & Onboarding**

**User Story :** As a new user, I want to register for an account so that I can start using the platform.  
  
**Acceptance Criteria:**

● User is prompted to enter their name, email, and password.

● If the email is already registered, an error message **“Email is already in use”** is shown.

● After successful registration, the user is redirected to the onboarding page to complete their profile.

● User receives a confirmation email to verify the account.

● If the email is invalid, an error message **“Invalid email address”** is displayed.

**User Story:** As a new user, I want to go through an onboarding process to complete my profile.  
  
**Acceptance Criteria:**

● User is prompted to fill in profile details such as bio, photo, and contact information.

● User can upload a profile photo (supported file types: jpg, png).

● If any required field is left blank, an error message **“This field is required”** is displayed.

● After completing the onboarding, the user is redirected to the home page.

**Epic 2: User Profile Management**

**User Story:** As a user, I want to update my personal information so that my profile is current and accurate.  
  
**Acceptance Criteria:**

● User can edit fields such as name, bio, and contact details.

● Changes are saved when the "Save Changes" button is clicked.

● If no changes are made and the user clicks "Save," an error message **“No changes were made”** is displayed.

● The updated profile is immediately reflected on the user's page.

**User Story:** As a user, I want to change my profile photo so that my account better represents me.  
  
**Acceptance Criteria:**

● User can upload a new photo from their device.

● Only image files (jpg, png) are accepted; any other file type shows an error message **“Invalid file type”**.

● After uploading, the new photo is displayed immediately on the profile page.

**Epic 3: Account Security**

**User Story:** As a user, I want to reset my password if I forget it so that I can regain access to my account.  
  
**Acceptance Criteria:**

● User is prompted to enter their registered email address.

● If the email is not found, an error message **“Email not registered”** is displayed.

● A 4-digit one-time password (OTP) is sent to the registered email.

● User enters the OTP and is allowed to create a new password.

● If the OTP is incorrect, an error message **“Invalid OTP”** is displayed.

**User Story:** As a user, I want to log in to my account securely with my email and password.  
  
**Acceptance Criteria:**

● User can log in by entering their registered email and password.

● If the email or password is incorrect, an error message **“Incorrect email or password”** is displayed.

● User is redirected to the homepage after a successful login.

● After login, user session is maintained until they log out.

**User Story :** As a user, I want to log out of my account so that I can secure my account when I am done using the platform.  
  
**Acceptance Criteria:**

● User can log out by clicking the "Logout" button.

● After logging out, the user is redirected to the login page.

● If the user is not logged in and clicks logout, an error message **“You are not logged in”** is displayed.

**Epic 4: User Connections**

**User Story:** As a user, I want to send a friend request to another user so that I can connect with them.  
  
**Acceptance Criteria:**

● User can send a friend request by clicking the "Add Friend" button on another user's profile page.

● If the friend request is sent successfully, the button changes to **“Request Sent”**.

● If the user is already friends, the button changes to **“Friend”**.

**User Story:** As a user, I want to accept or decline friend requests from other users so that I can manage my connections.  
  
**Acceptance Criteria:**

● User can see pending friend requests in the notifications section.

● User can accept or decline friend requests.

● If accepted, the button changes to **“Friend”**, and the requestor is notified.

● If declined, the request is removed from the pending list, and no further notifications are sent.

**User Story:** As a user, I want to send direct messages to my friends so that I can communicate with them privately.  
  
**Acceptance Criteria:**

● User can select a friend from their friend list and send a message.

● Messages are delivered instantly, and the conversation is visible to both users.

● If there is no internet connection, an error message **“Message failed to send. Please try again”** is displayed.

**Epic 5: Notification System**

**User Story :** As a user, I want to receive notifications and activity updates so that I can stay informed about my connections and platform activity.  
  
**Acceptance Criteria:**

● User receives a notification when they have a new friend request, message, or startup activity update.

● Notifications are displayed in the notification center and as a bell icon in the header.

● If no new notifications exist, the bell icon shows **“No New Notifications”**.

**Epic 6: Search and Discover Startups**

**User Story:** As a user, I want to search for startups by name, industry, or location so that I can find startups matching my interests.  
  
**Acceptance Criteria:**

● User can input search keywords and filter by industry, location, and startup stage.

● Matching startups are displayed in real-time.

● Error message appears if no results are found (e.g., "No startups match your search criteria.").

**User Story:** As a user, I want to browse trending startups so that I can explore popular and successful ventures.  
  
**Acceptance Criteria:**

● Trending startups are displayed in a featured section.

● Startups are sorted by metrics like views, followers, or recent funding rounds.

● Error message appears if trending startups fail to load (e.g., "Unable to load trending startups. Please try again later.").

**Epic 7: Explore Job Opportunities**

**User Story:** As a user, I want to browse job openings from startups so that I can find relevant positions to apply for*.***Acceptance Criteria:**

● User can filter jobs by industry, position type, or location.

● Job listings include title, description, and application deadline.

● Error message appears if no jobs are available (e.g., "No job opportunities found for the selected criteria.").

**User Story:** As a user, I want to save jobs I’m interested in so that I can apply later.  
  
**Acceptance Criteria:**

● User can save a job by clicking a "Bookmark" button.

● Saved jobs appear in the user’s dashboard under "Saved Jobs."

● Error message appears if saving fails (e.g., "Unable to save this job. Please try again.").

**Epic 8: Social Engagement**

**User Story:** As a user, I want to follow startups so that I can stay updated on their latest activities.  
  
**Acceptance Criteria:**

● User can follow a startup by clicking a "Follow" button.

● Followed startups appear in the user’s dashboard under "Following."

● Error message appears if the follow action fails (e.g., "Unable to follow this startup. Please try again.").

**User Story:** As a user, I want to like and comment on startup job posts so that I can engage with their updates.  
  
**Acceptance Criteria:**

● User can like posts by clicking a "Like" button.

● User can add comments to posts via a comment box.

● Error message appears if comments fail to post (e.g., "Unable to post your comment. Please try again.").

**Epic 9: Personalized Recommendations**

**User Story:** As a user, I want to receive personalized startup recommendations so that I can discover relevant opportunities.  
  
**Acceptance Criteria:**

● User sees recommended startups based on their interests, activities, and saved preferences.

● Clicking a recommended startup navigates to its profile page.

● Error message appears if recommendations fail to load (e.g., "Unable to load recommendations. Please refresh the page.").

**User Story:** As a user, I want to get job recommendations based on my profile so that I can easily find relevant roles.  
  
**Acceptance Criteria:**

● User sees a curated list of job openings tailored to their skills, location, and preferences.

● Job details include title, company, and application link.

● Error message appears if recommendations fail to load (e.g., "Unable to load job recommendations. Please refresh the page.").

**Epic 10: Content Sharing and Collaboration**

**User Story:** As a user, I want to share resources or ideas with startups so that I can contribute to their growth.  
  
**Acceptance Criteria:**

● User can upload or share links to resources directly on a startup’s profile.

● Shared content is visible to the startup and other users.

● Error message appears if sharing fails (e.g., "Unable to share your content. Please try again.").

**User Story:** As a user, I want to join discussions about startups so that I can engage with like-minded individuals.  
  
**Acceptance Criteria:**

● User can post, reply to, and like discussion threads.

● Threads are categorized by topics (e.g., industries, funding stages).

● Error message appears if a post fails to submit (e.g., "Unable to post your message. Please try again.").

**Theme: Startup owner**

**Epic 1: Startup Registration**

**User Story:** As a startup owner, I want to register a new startup so that I can showcase it on the platform.  
  
**Acceptance Criteria:**

● User can input startup name, description, logo, and key details (e.g., industry, location).

● User receives a confirmation message upon successful registration.

● Error message appears if any required field is missing (e.g., "Startup name is required").

● Uploaded logo must meet file type/size requirements (e.g., "File must be JPG/PNG and under 2MB").

**User Story:** As a startup owner, I want to verify my startup profile so that it gains credibility on the platform.  
  
**Acceptance Criteria:**

● User can submit proof of legitimacy (e.g., business registration documents).

● Verified badge appears on the startup profile after approval.

● Error message appears if uploaded documents are missing or invalid (e.g., "Upload failed. Please provide valid documents").

**Epic 2: Startup Profile Management**

**User Story:** As a startup owner, I want to update my startup profile so that it reflects the latest information.  
  
**Acceptance Criteria:**

● User can edit startup name, description, logo, and other details.

● Updates are saved and displayed immediately.

● Error message appears if an invalid input is entered (e.g., "Description exceeds character limit").

**User Story:** As a startup owner, I want to add team members to my startup profile so that I can showcase my team.  
  
**Acceptance Criteria:**

● User can invite other platform users to join their startup.

● Team members appear on the profile after accepting the invitation.

● Error message appears if an invitation fails (e.g., "User not found").

**Epic 3: Job Posting and Management**

**User Story:** As a startup owner, I want to post job openings so that I can recruit team members.  
  
**Acceptance Criteria:**

● User can create a job post with title, description, requirements, and deadlines.

● Job post is displayed under the startup’s profile and in the job listing section.

● Error message appears if required fields are left empty (e.g., "Deadline is required").

**User Story:** As a startup owner, I want to update job postings so that they remain accurate and up-to-date.  
  
**Acceptance Criteria:**

● User can edit or delete existing job posts.

● Changes are reflected immediately.

● Error message appears if an invalid input is entered (e.g., "Invalid date format for deadline").

**Epic 4: Analytics and Performance**

**User Story:** As a startup owner, I want to view analytics for my startup profile so that I can track its performance.  
  
**Acceptance Criteria:**

● User can view metrics like profile views, job applications, and followers.

● Data is presented in charts/graphs.

● Error message appears if metrics fail to load (e.g., "Unable to retrieve analytics. Please try again later").

**User Story:** As a startup owner, I want to monitor my job posting analytics so that I can evaluate candidate engagement.  
  
**Acceptance Criteria:**

● User can view metrics like the number of applications per job post.

● Metrics are updated in real-time.

● Error message appears if metrics fail to load (e.g., "Unable to retrieve job posting data").

**Epic 5: Startup Promotion**

**User Story:** As a startup owner, I want to promote my startup so that it reaches a larger audience.  
  
**Acceptance Criteria:**

● User can select promotion packages (e.g., featured placement, ads).

● Confirmation message appears after purchase.

● Error message appears if payment fails (e.g., "Payment could not be processed. Please try again").

**User Story:** As a startup owner, I want to highlight my startup achievements so that it attracts more attention.  
  
**Acceptance Criteria:**

● User can add awards, milestones, or press mentions to their profile.

● Highlighted achievements appear in a dedicated section on the profile.

● Error message appears if an invalid file is uploaded (e.g., "File exceeds size limit").

**Theme: Investor**

**Epic 1: Investor Account Setup**

**User Story:** As an investor, I want to create an investor profile so that startups can see my interests and expertise.  
  
**Acceptance Criteria:**

● User can input details like investment focus, industries, funding range, and location.

● Profile is successfully created after submission.

● Error message appears if mandatory fields are empty (e.g., "Funding range is required").

**User Story:** As an investor, I want to verify my profile so that startups trust my credibility.  
  
**Acceptance Criteria:**

● User can upload verification documents (e.g., business registration, ID).

● Verified badge is added after approval.

● Error message appears if uploaded documents are invalid or missing (e.g., "Verification failed. Please provide valid documentation").

**Epic 2: Discover Startups**

**User Story:** As an investor, I want to search for startups based on industries and funding needs so that I can find relevant investment opportunities.  
  
**Acceptance Criteria:**

● User can filter startups by industry, location, funding stage, and team size.

● Relevant startups are displayed in real-time.

● Error message appears if no startups match the criteria (e.g., "No results found for the selected filters").

**User Story:** As an investor, I want to view detailed startup profiles so that I can evaluate their potential.  
  
**Acceptance Criteria:**

● User can see startup details such as description, team members, funding history, and pitch decks.

● User can download available files (e.g., pitch decks or financial reports).

● Error message appears if files fail to load (e.g., "File could not be downloaded. Please try again later").

**Epic 3: Investment Tracking and Analytics**

**User Story:** As an investor, I want to track startups I’ve invested in so that I can monitor their growth.  
  
**Acceptance Criteria:**

● User can view a list of startups they’ve invested in.

● User can see metrics like funding milestones and recent updates from the startup.

● Error message appears if metrics fail to load (e.g., "Unable to retrieve data. Please refresh the page").

**User Story:** As an investor, I want to receive performance analytics on startups I follow so that I can evaluate their progress.  
  
**Acceptance Criteria:**

● User can view data visualizations of key metrics (e.g., growth rate, revenue).

● User receives notifications about significant changes (e.g., funding secured, team updates).

● Error message appears if analytics fail to load (e.g., "Performance analytics are currently unavailable").

**Epic 4: Communication with Startups**

**User Story:** As an investor, I want to message startup owners so that I can discuss potential opportunities.  
  
**Acceptance Criteria:**

● User can initiate and receive messages with startup owners.

● Messages are displayed in a chat interface.

● Notification is sent for new messages.

● Error message appears if sending fails (e.g., "Message could not be sent. Please try again").

**User Story:** As an investor, I want to request additional information from startups so that I can make informed decisions.

**Acceptance Criteria:**

● User can send a request for specific details (e.g., financial reports, projections).

● Startup owners receive the request in their dashboard.

● Error message appears if the request cannot be sent (e.g., "Request failed. Please check your connection").

**Epic 5: Investment Opportunities**

**User Story:** As an investor, I want to see highlighted investment opportunities so that I can easily discover promising startups.  
  
**Acceptance Criteria:**

● User sees a curated list of featured startups based on their interests.

● Startups are tagged with labels like "Rising Star" or "Trending."

● Error message appears if the page fails to load (e.g., "Unable to retrieve opportunities. Please try again").

**User Story:** As an investor, I want to bookmark startups so that I can revisit them later.  
  
**Acceptance Criteria:**

● User can bookmark a startup by clicking a "Save" button.

● Bookmarked startups appear in the user’s dashboard under "Saved Startups."

● Error message appears if saving fails (e.g., "Unable to save this startup. Please try again").

**Epic 6: Promotion Management**

**User Story:** As an investor, I want to promote my profile so that startups can reach out to me directly.

**Acceptance Criteria:**

● User can select from promotion packages (e.g., "Top Investor" badge, featured profile placement).

● User receives confirmation after purchase.

● Error message appears if payment fails (e.g., "Payment could not be processed. Please try again later").

**Non functional requirements**

● The system must support at least 1,000 concurrent users without significant performance degradation

● The platform should be able to scale horizontally to accommodate an increase in users, startups, and job postings.

● User data, including personal information and startup details, must be encrypted using AES-256 for storage and TLS 1.2 or higher for transmission.

● The platform should maintain a 99.9% uptime to ensure accessibility for users at all times.

● The system should provide a simple and intuitive user interface, ensuring that users can complete major tasks (e.g., registering a startup, applying for jobs) within 3 clicks.

● The platform should be built using a modular architecture to allow easy updates and maintenance with minimal downtime.

● The system must comply with GDPR and ISO 27001 standards for data protection and privacy.

● The platform should perform automated daily backups, and in case of failure, data recovery should be possible within 30 minutes.

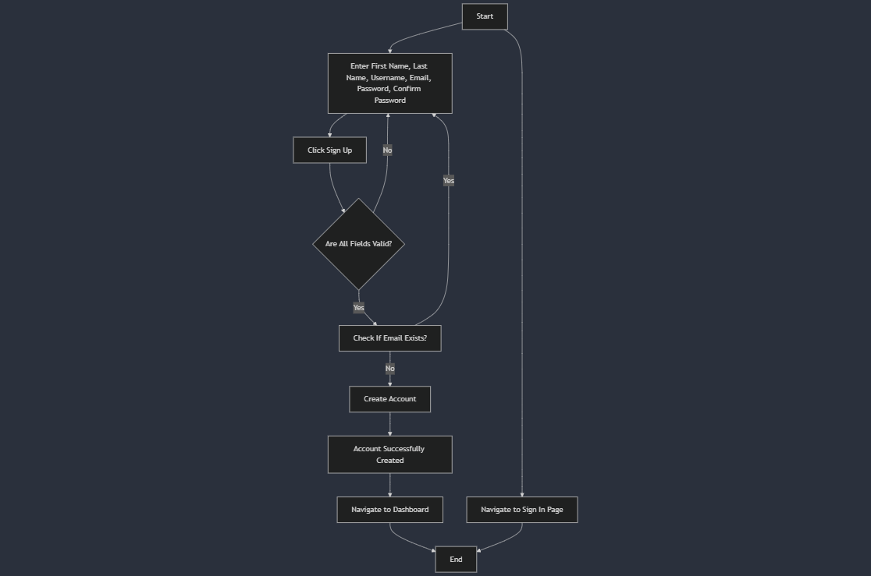
● The system must be cross-platform compatible, functioning smoothly on Windows, macOS, Android, and iOS with responsive design.

● The system should implement real-time logging and monitoring using tools like ELK Stack or AWS CloudWatch to track errors, user activity, and security incidents.

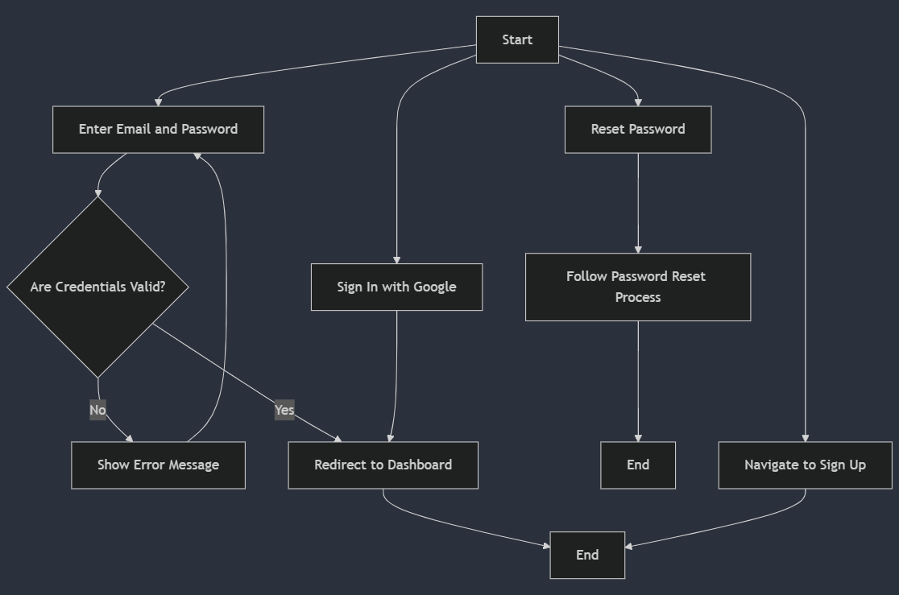
## 1.4. UML diagrams

**Activity Diagrams**

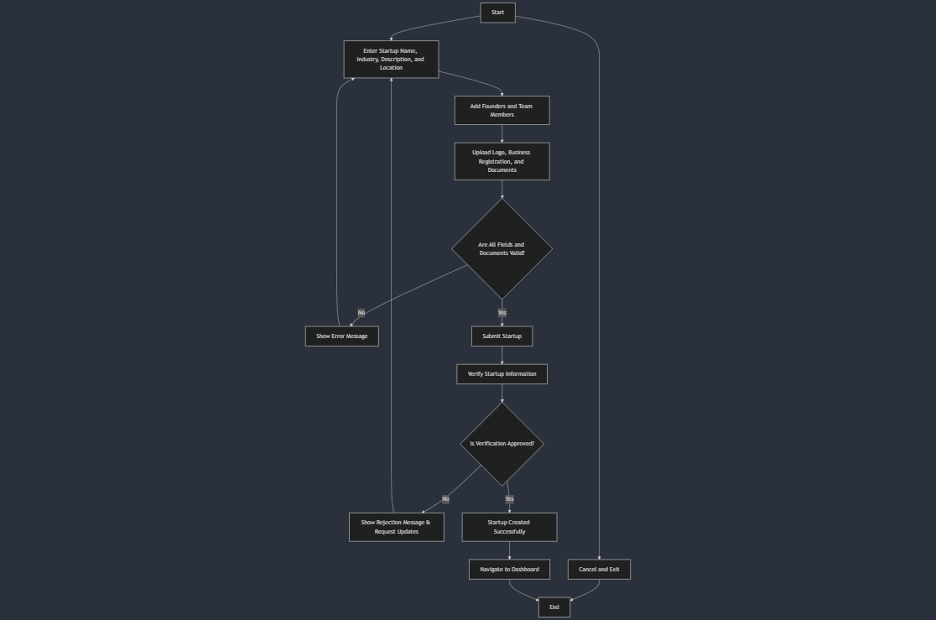
**User registration**

****

**User login**

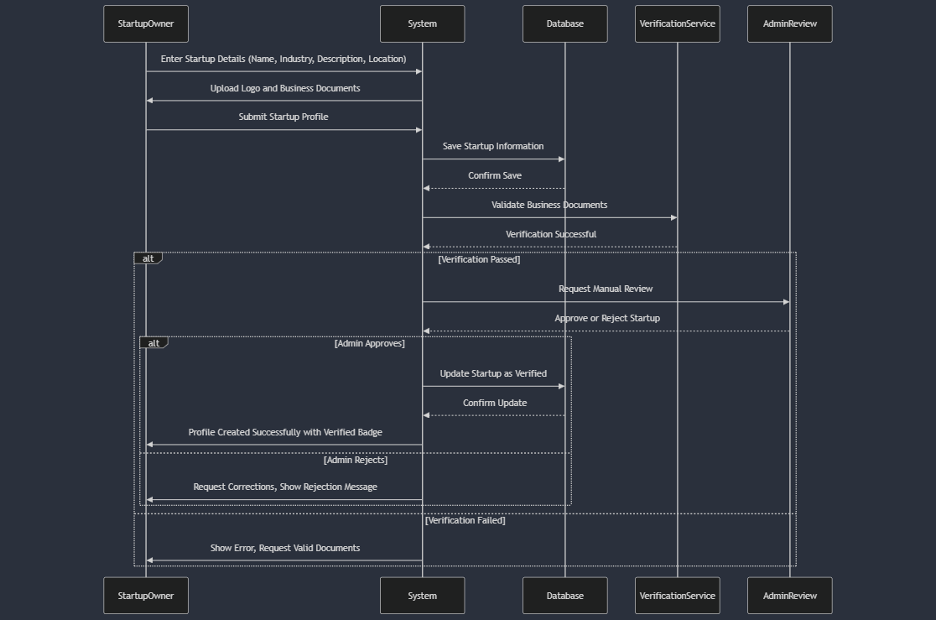
****

**Create startup profile**

****

**Sequence diagrams**

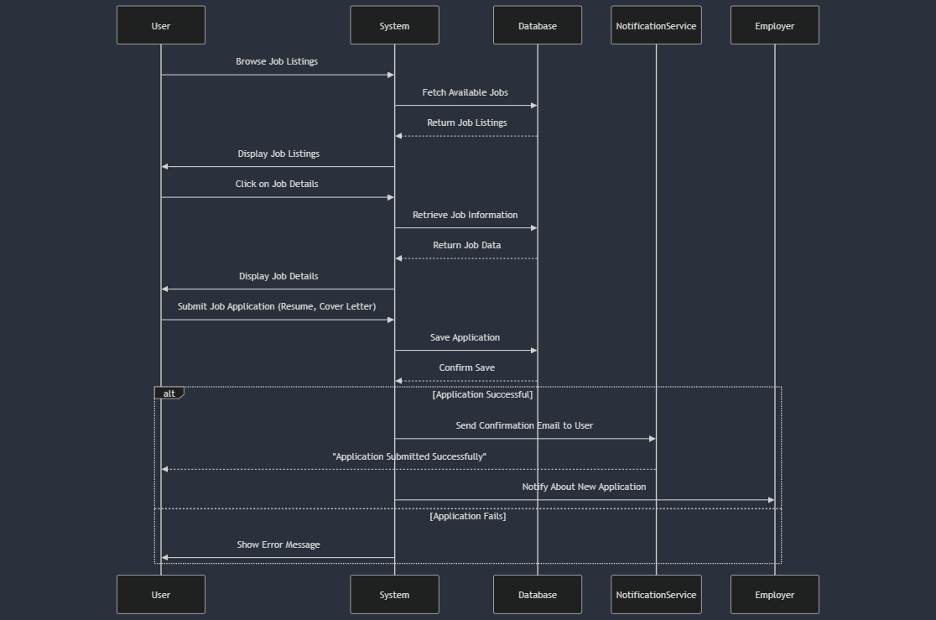
**Startup creation**

****

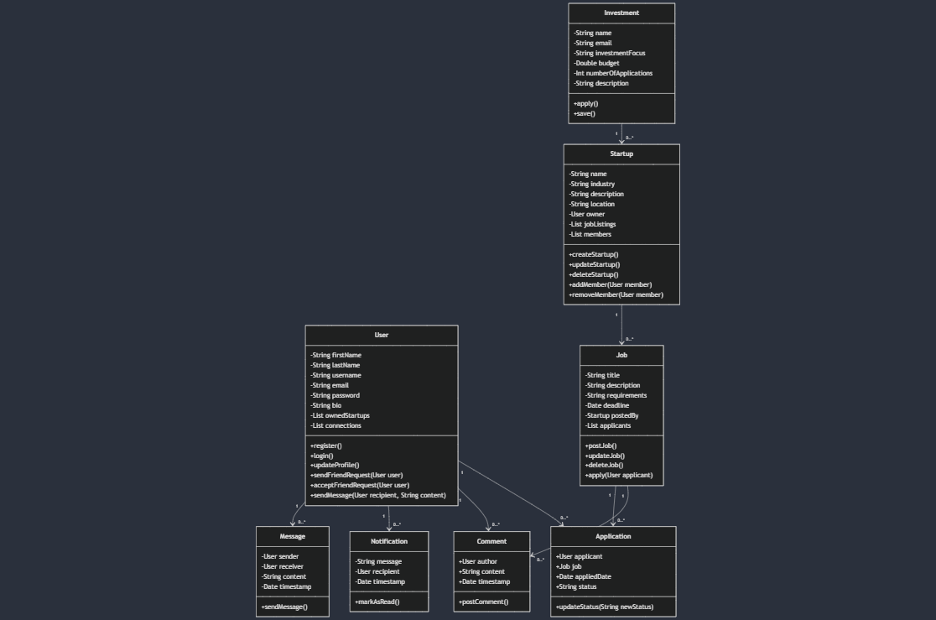
**Login**

****

**Job application by user**

****

**Class diagrams**

****

# 2. Project Structure

## 2.1. Technologies

#### Frontend:

* **React** – for building a dynamic, responsive single-page application (SPA).
* **Bootstrap** – for fast and responsive UI styling using pre-built components and grid layout system.

#### Backend:

* **Spring Boot** – for developing a robust and scalable RESTful API in Java. It simplifies configuration and integrates well with database and security layers.

#### Database:

* **PostgreSQL** – as the relational database system to store application data reliably, including user information, startup profiles, job applications, etc.

#### Other Tools & Technologies:

* **Maven** – for dependency management and project build.
* **Docker** – used for containerizing the application to simplify deployment.
* **Neon** – a serverless PostgreSQL provider used to host our production database in the cloud.
* **Git & GitHub** – for version control and team collaboration.
* **Render** – for deploying the backend and frontend services to production.

## 2.2. Architectural Pattern

**Backend Architectural Pattern: Layered Architecture**

#### 1. Controller Layer (rest/)

* Handles HTTP requests (e.g., /api/v1/users)
* Calls the service layer to process business logic
* Returns HTTP responses to the client
* Example: UserController.java

#### 2. Service Layer (service/)

* Contains core **business logic** and orchestrates the application's behavior
* Acts as an intermediary between controllers and repositories
* Example: UserServiceImpl.java

#### 3. Repository Layer (repository/)

* Handles direct interaction with the **database** using Spring Data JPA
* Abstracts data access logic away from the rest of the app
* Example: UserRepository.java

#### 4. Model Layer (model/)

* Contains entity classes that map to database tables using JPA annotations
* Example: User.java, Startup.java

#### 5. DTO Layer (dto/)

* Contains Data Transfer Objects used for updating or returning user-defined subsets of data (e.g., UserDTO)

Why:

* **Separation of concerns** – Each layer has a single responsibility, making the codebase easier to maintain and understand.
* **Scalability** – You can easily expand or refactor individual layers without impacting others.
* **Testability** – Each layer (especially the service layer) can be unit tested independently using mocks.
* **Industry standard** – This is a best practice for Spring Boot applications and is well-supported by documentation and community resources.

**Frontend Architectural Pattern**: **Component-Based + Service Layer**

#### 1. Component-Based Structure

**Each part of the UI is built as a reusable component, making the app easy to maintain and extend. For example:**

* **Navbar.js can be reused across all pages.**
* **LoginPage.js handles login view and logic in isolation.**

#### 2. Routing with React Router

**Your App.js configures routing between different pages like:**

* **/ → Home page**
* **/login → Login page**
* **/dashboard → Startup dashboard**

**This enables fast navigation without full page reloads.**

#### 3. Service Layer (API abstraction)

**You created a services/ folder (e.g., userService.js) to separate API calls from the UI logic. This helps with:**

* **Clean code**
* **Easier testing**
* **Reusability across components**

#### 4. Styling with Bootstrap

**Bootstrap is used for layout and UI styling, ensuring responsiveness and a consistent look.**

### Why

* **Separation of concerns: Components handle view logic; services handle data logic.**
* **Modularity: Easy to update/replace individual parts.**
* **Scalability: New pages and features can be added with minimal changes to existing code.**
* **Maintainability: Clean folder structure makes it easy for new developers to understand the codebase.**

## 2.3. Database Entities

* **user** Stores information about all users of the platform, including students, startup owners, and investors (e.g., name, email, role, password, status).
* **startups** Represents each registered startup on the platform, including its name, description, status, and associated owner.
* **startup\_members** A join table that tracks which users are members of which startups. Useful for team management and collaboration features.
* **post** Represents content/posts shared by startups (like updates, announcements, etc.).
* **post\_likes** Tracks which users liked which posts — used for engagement metrics.
* **comment** Stores comments made by users on posts. Supports interaction and discussion on startup activity.

## 2.4. Design Patterns

### 1. Singleton Pattern

**Where:** All Spring-managed beans are singletons by default. You’ll see this in every @Service, @Repository, and @RestController class.

* **StartupServiceImpl.java** (line 14: @Service)
* **UserServiceImpl.java** (line 14: @Service)
* **StartupRepository.java** (line 9: @Repository)
* **PostController.java** (line 10: @RestController)  
   <br>

**Why:** Ensures one shared instance per bean, simplifying resource management and guaranteeing consistent configuration across requests.

### 2. Data Transfer Object (DTO) Pattern

**Where:** DTOs decouple our API payloads from JPA entities.

* **com.ibustartup.ibustartup.dto.UserDTO.java** (lines 1–15)
* **com.ibustartup.ibustartup.dto.StartupDTO.java** (lines 1–20)
* **com.ibustartup.ibustartup.dto.PostDTO.java** (lines 1–18)
* **com.ibustartup.ibustartup.dto.CommentDTO.java** (lines 1–16)  
   <br>

**Why:** Prevents over-exposure of internal fields (e.g. password hashes, lazy-loaded relations) and provides a stable API contract to the frontend.

### 3. Repository Pattern

**Where:** Spring Data JPA interfaces abstract all persistence logic.

* **com.ibustartup.ibustartup.repository.UserRepository.java** (extends JpaRepository, lines 1–20)
* **com.ibustartup.ibustartup.repository.StartupRepository.java** (lines 1–20)
* **com.ibustartup.ibustartup.repository.PostRepository.java** & **CommentRepository.java** <br>

**Why:** Declares finders like findByIndustry(...) or findByLocationContainingIgnoreCase(...) without boilerplate, keeping data access simple and type-safe.

### 4. Mapper (Adapter) Pattern

**Where:** Converts between entity and DTO.

* **com.ibustartup.ibustartup.mapper.StartupMapper.java** (lines 1–30)
* **com.ibustartup.ibustartup.mapper.PostMapper.java** (lines 1–25)
* **com.ibustartup.ibustartup.mapper.CommentMapper.java** (lines 1–20)  
   <br>

**Why:** Centralizes all mapping logic in one place, keeps controllers slim, and makes it easy to evolve the API shape without touching business logic.

### 5. Dependency Injection Pattern

**Where:** Constructor injection in controllers/services via @Autowired.

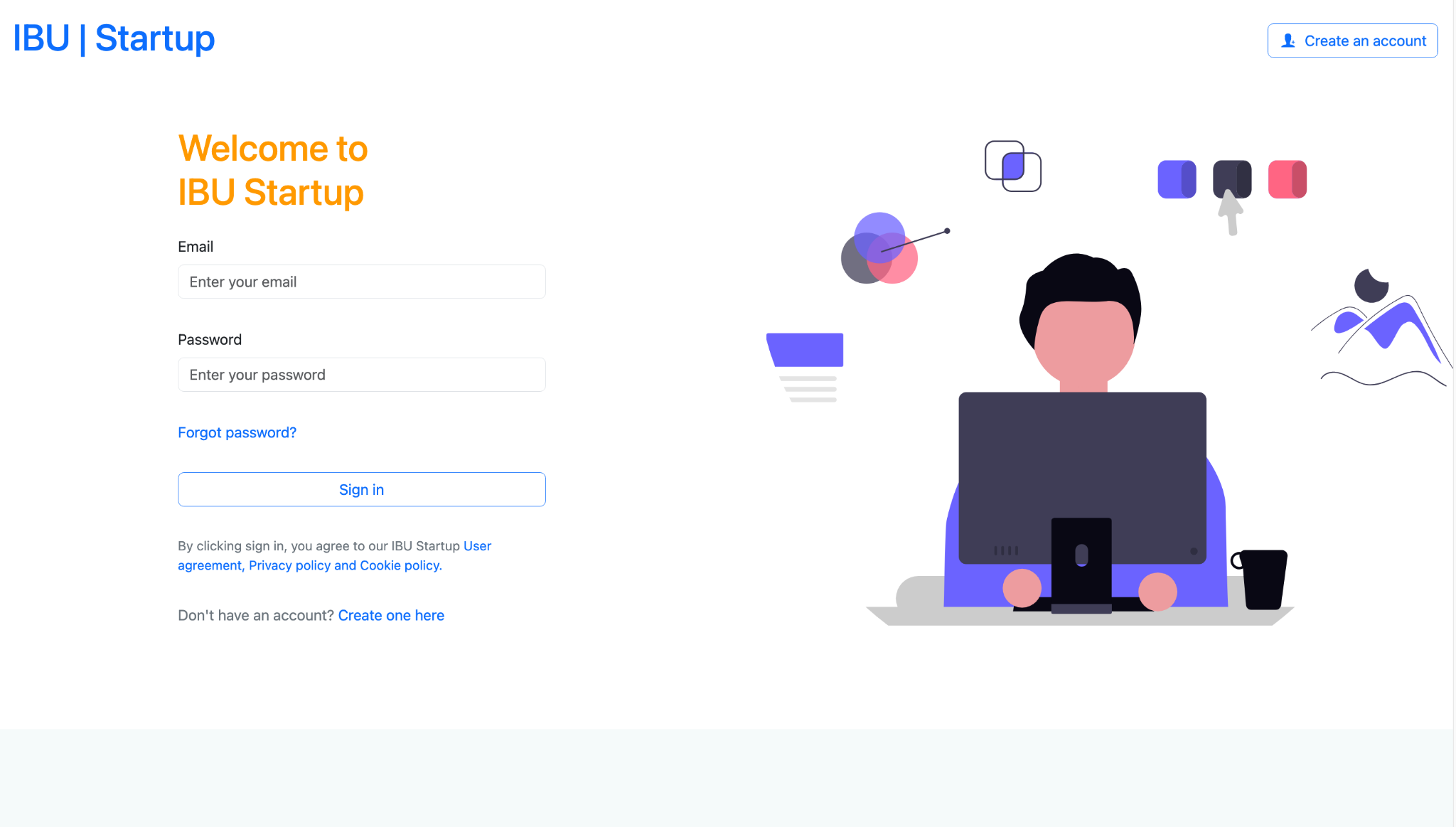
* **StartupController.java** (lines 12–20)
* **StartupServiceImpl.java** (lines 16–25)
* **PostServiceImpl.java** (lines 18–32)  
   <br>

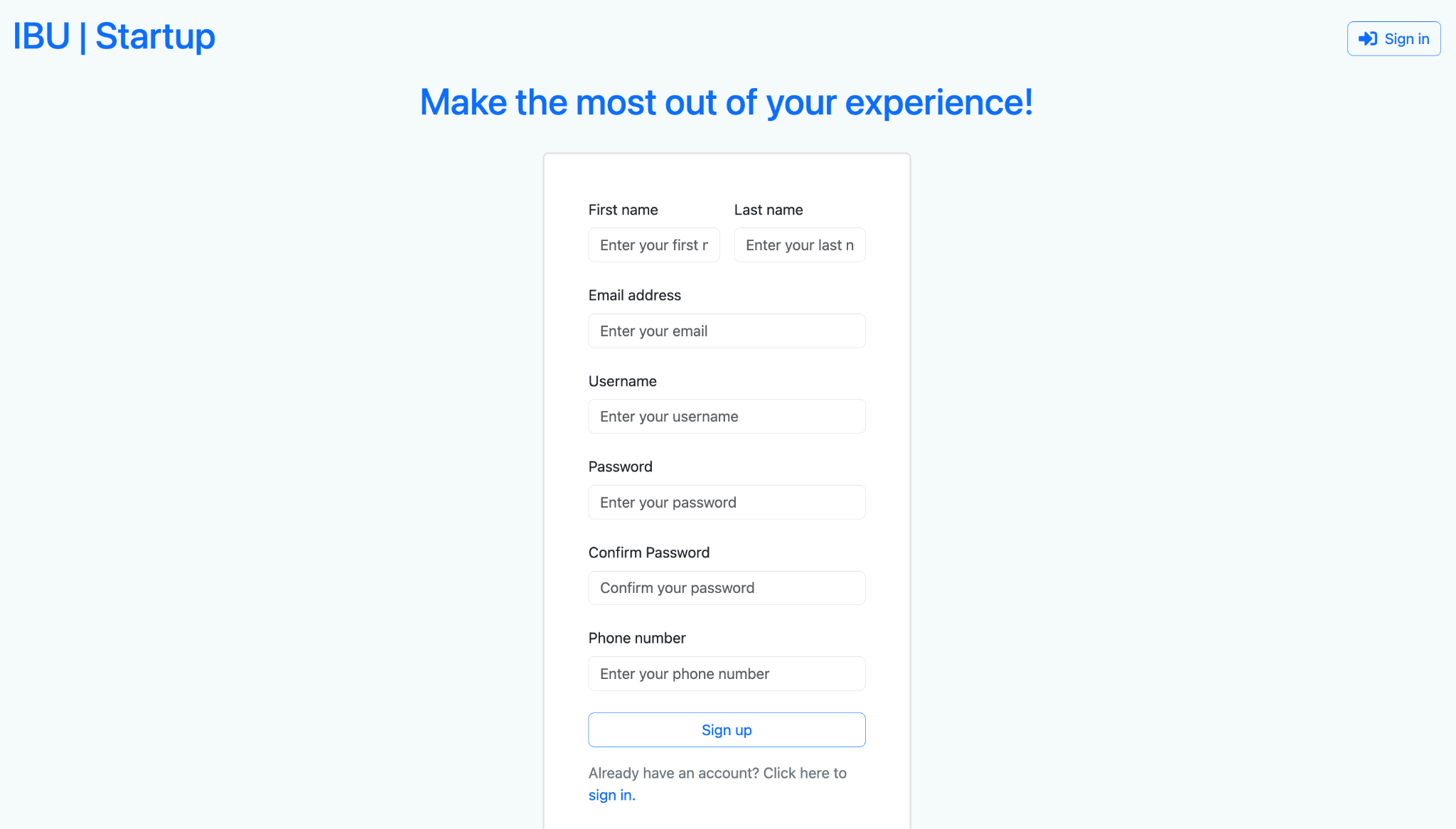
**Why:** Promotes loose coupling between components, simplifies unit testing (mocks can be injected), and offloads object‐lifecycle management to Spring.

## 2.5. Project Functionalities and Screenshots

### 2.5.1 User Authentication & Profile

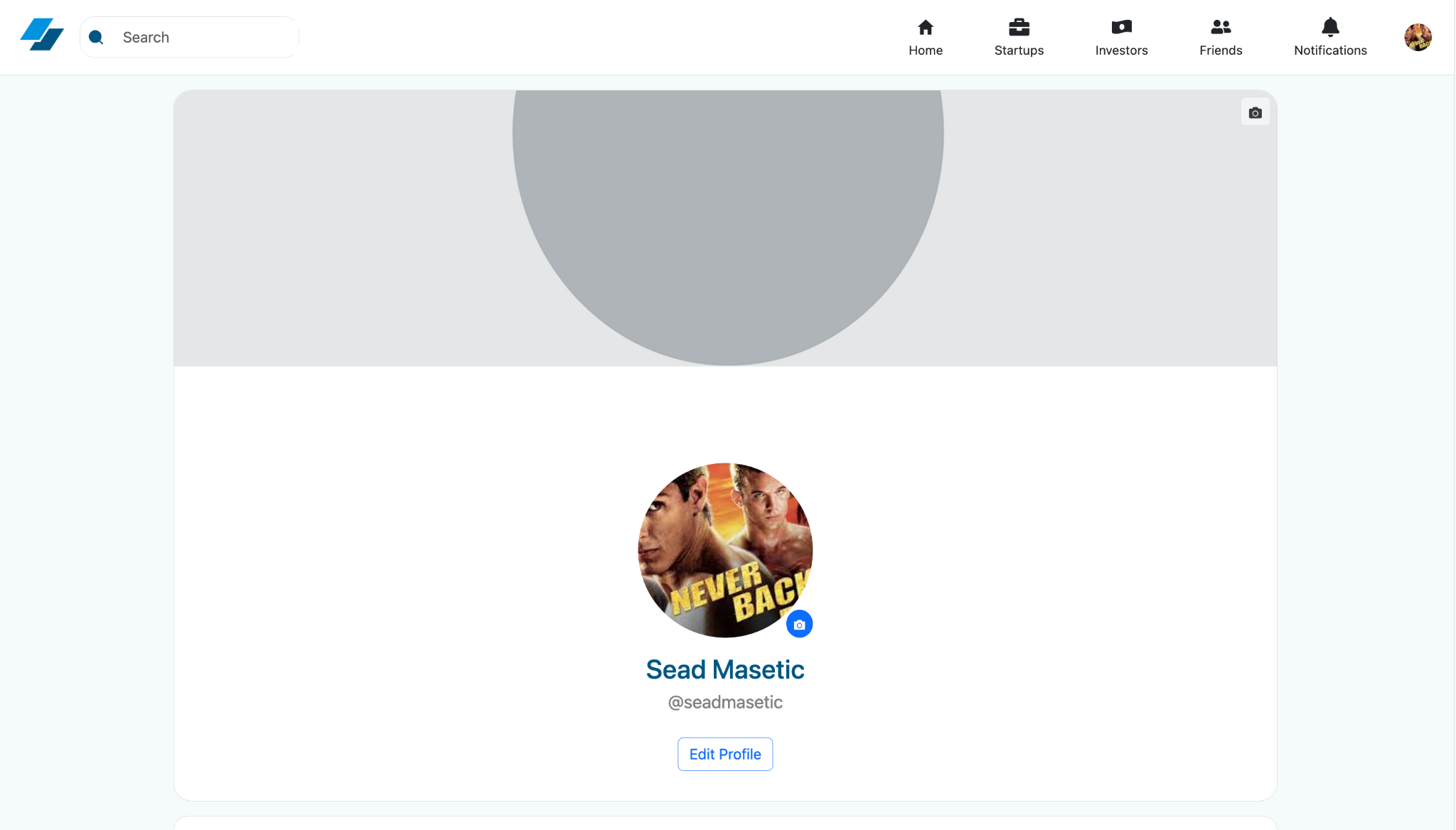
* **Sign Up & Login** Users can register with email/password, then log in to obtain a JWT.

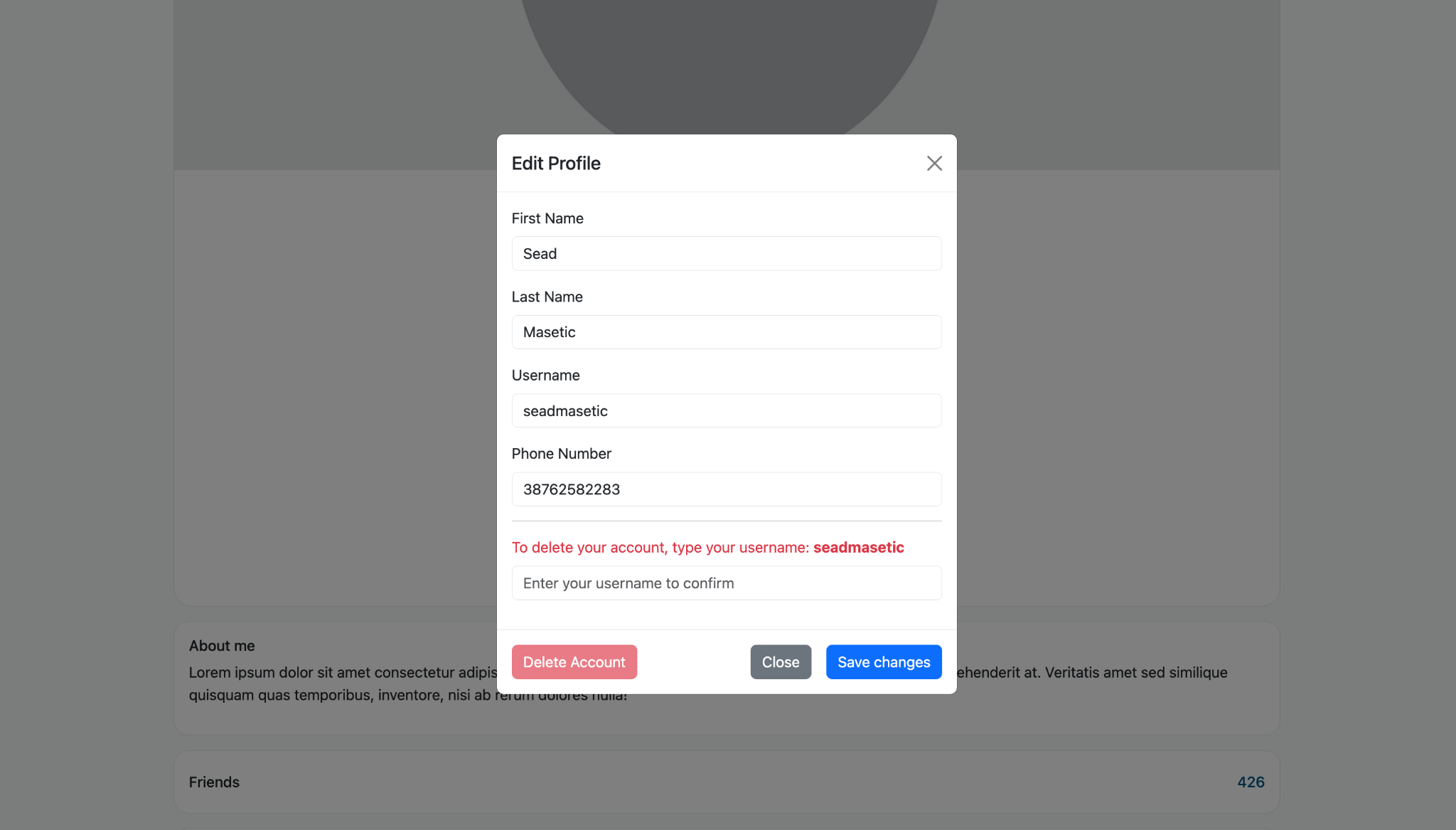




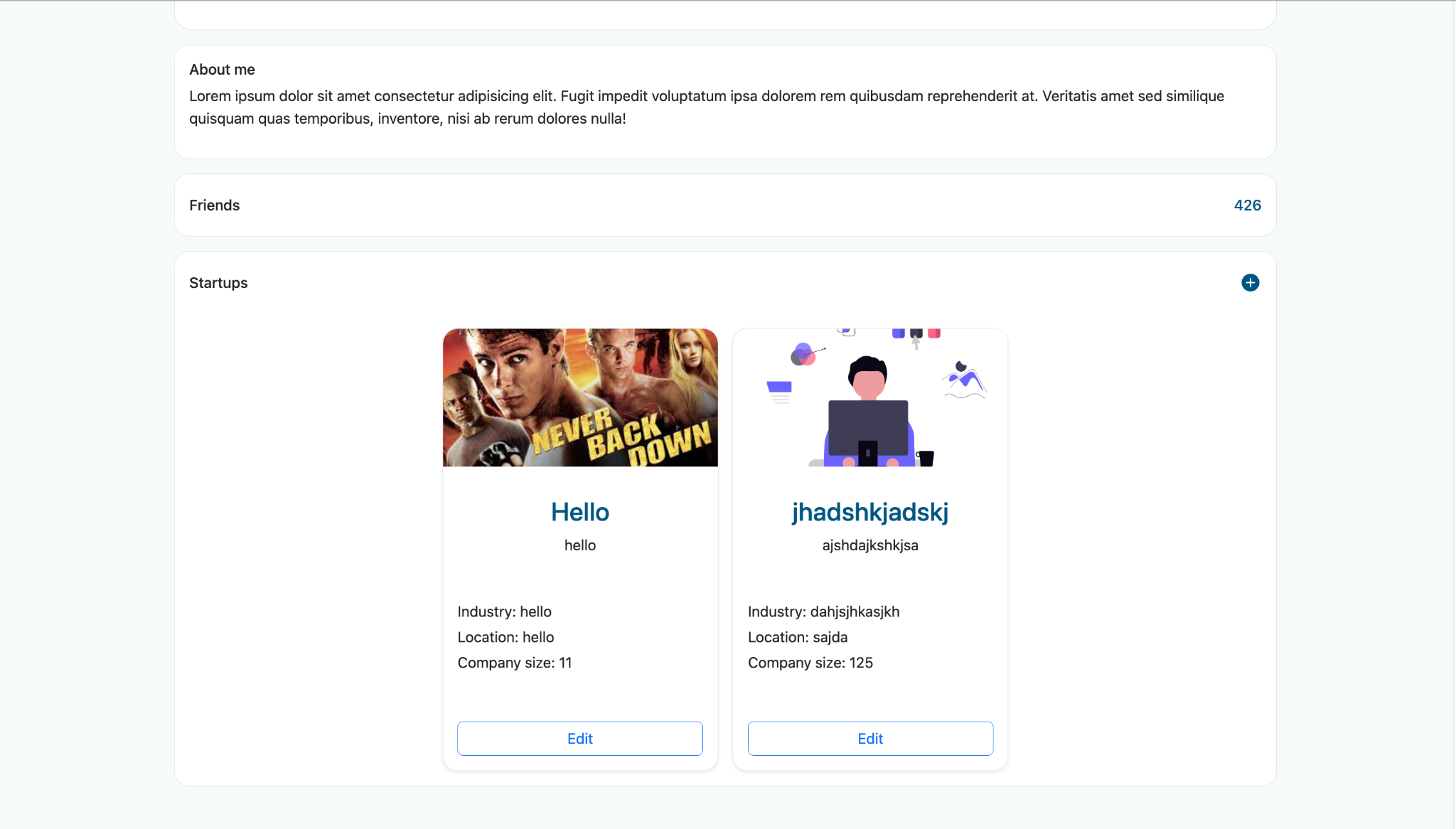
### 2.5.2 User Dashboard & Profile Management

* **View & Edit Profile** Users can view and update their personal information (name, phone, username).



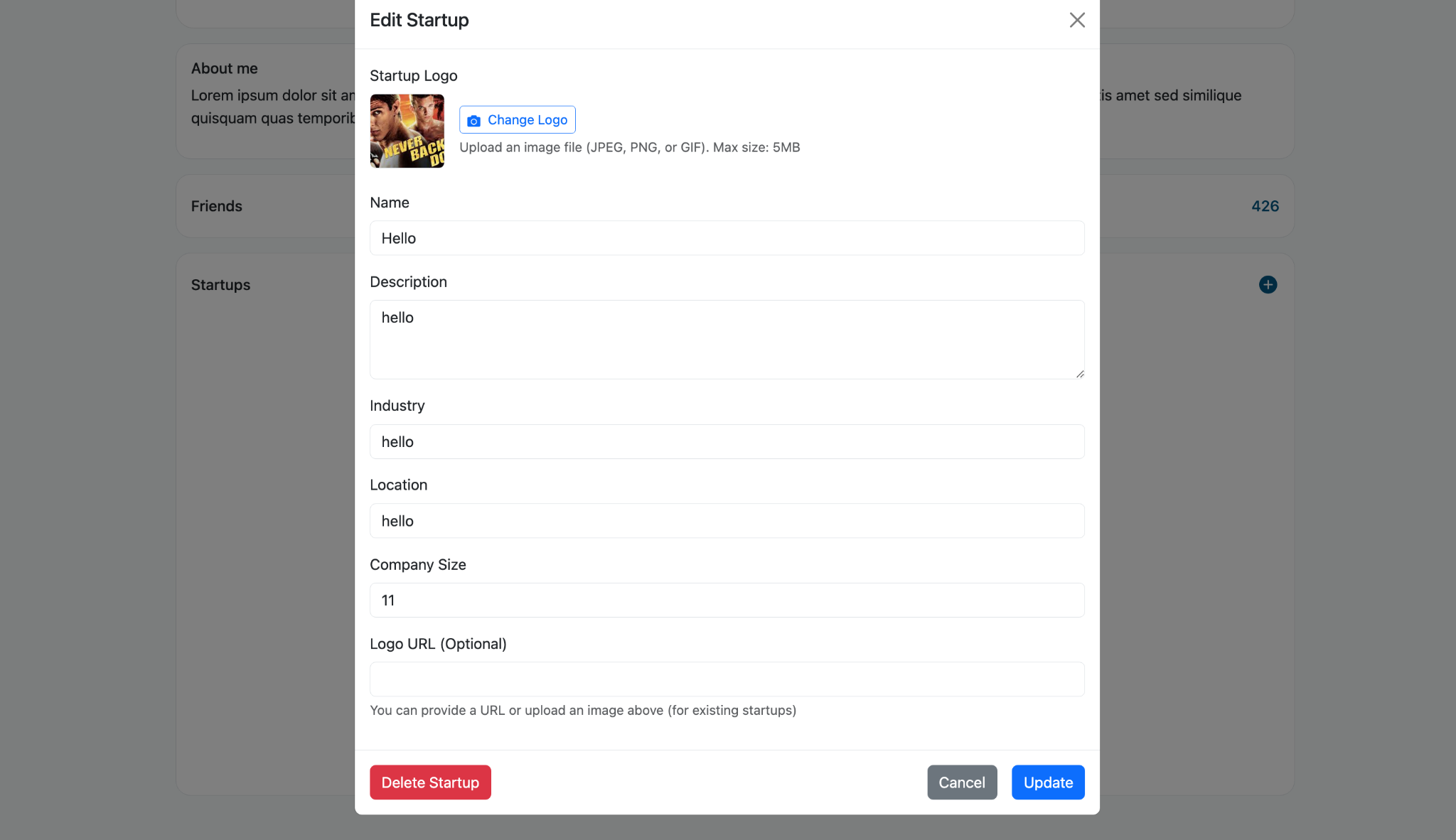
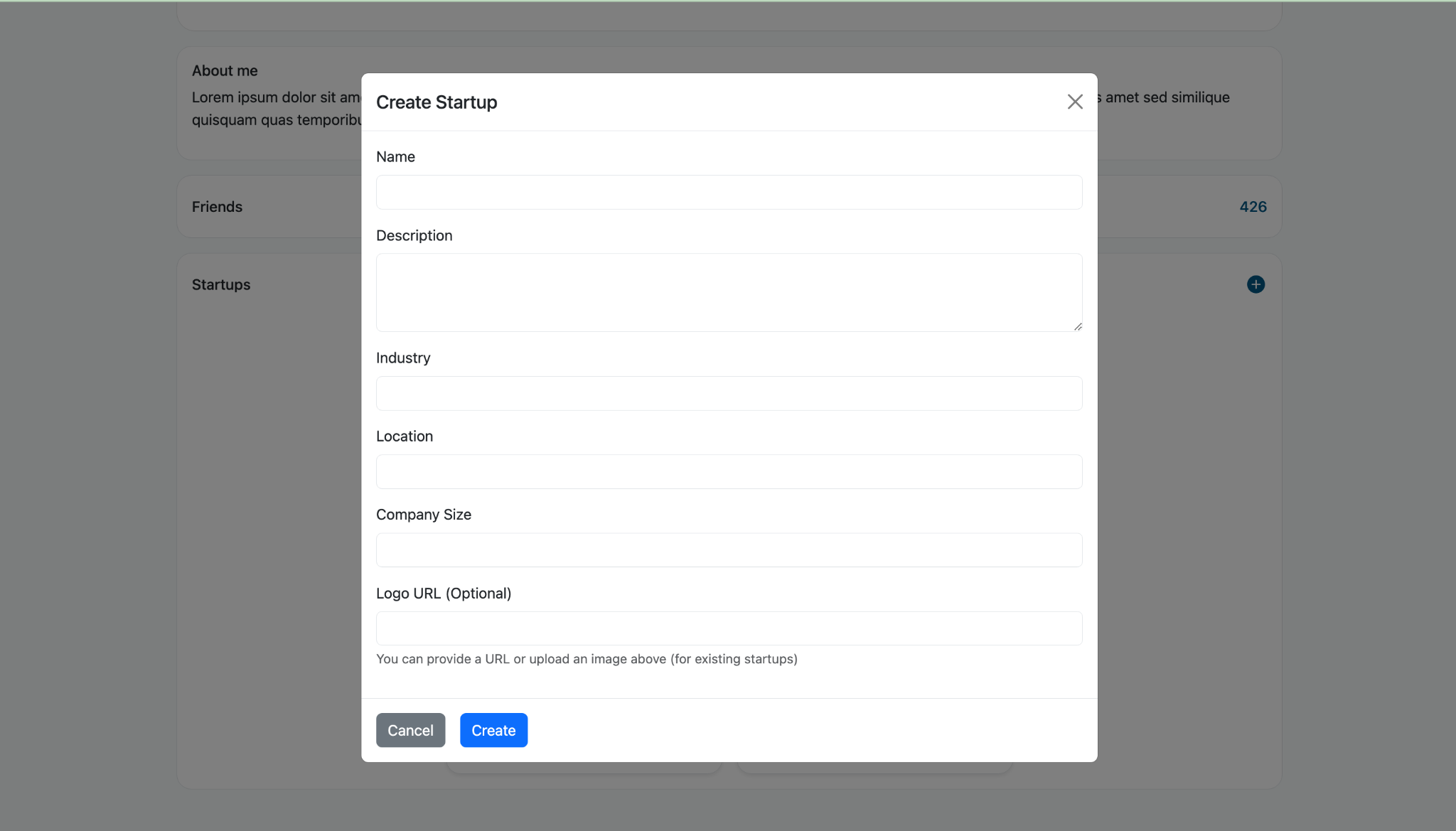


* **My Startups**Sidebar lists startups owned by the logged-in user, with options to manage them.

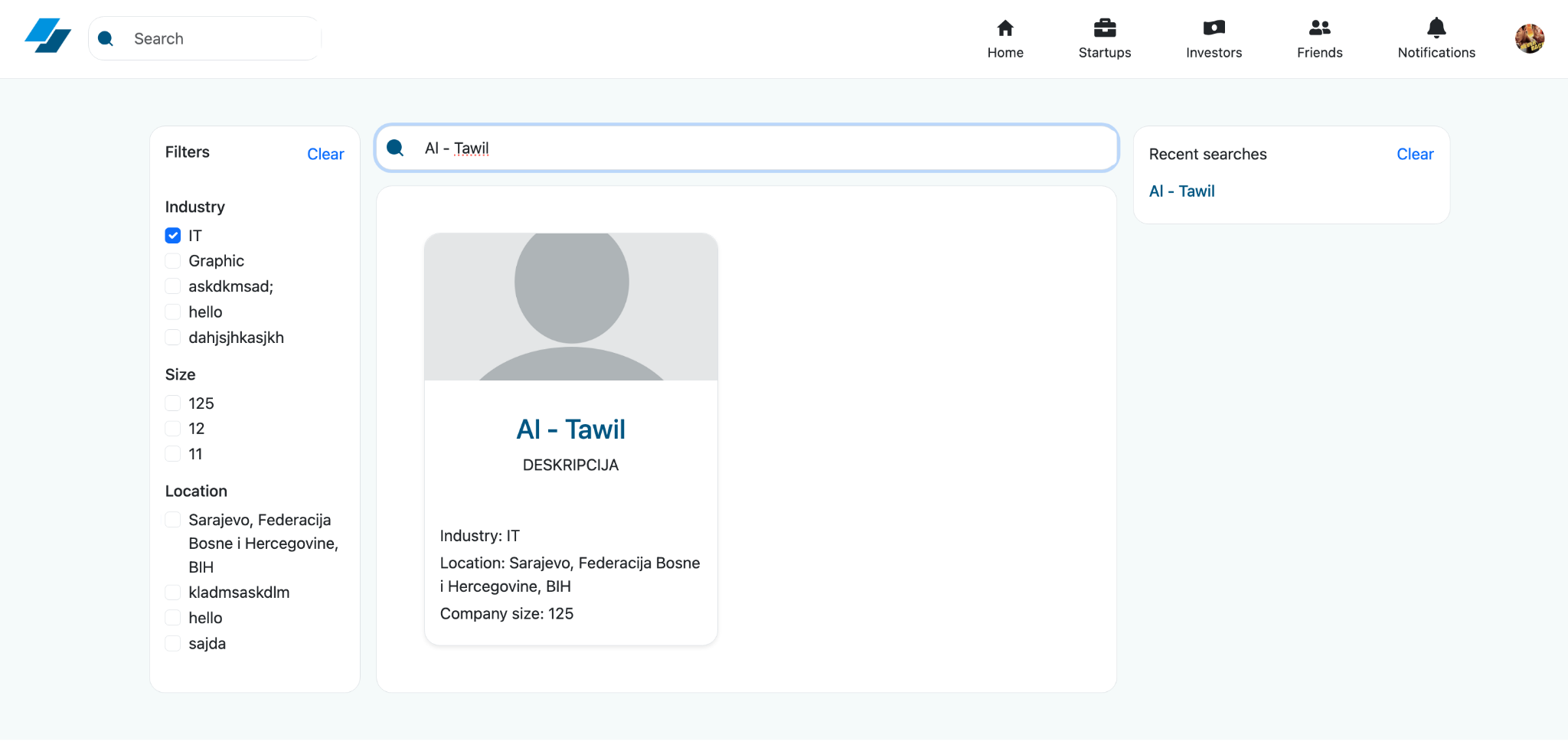
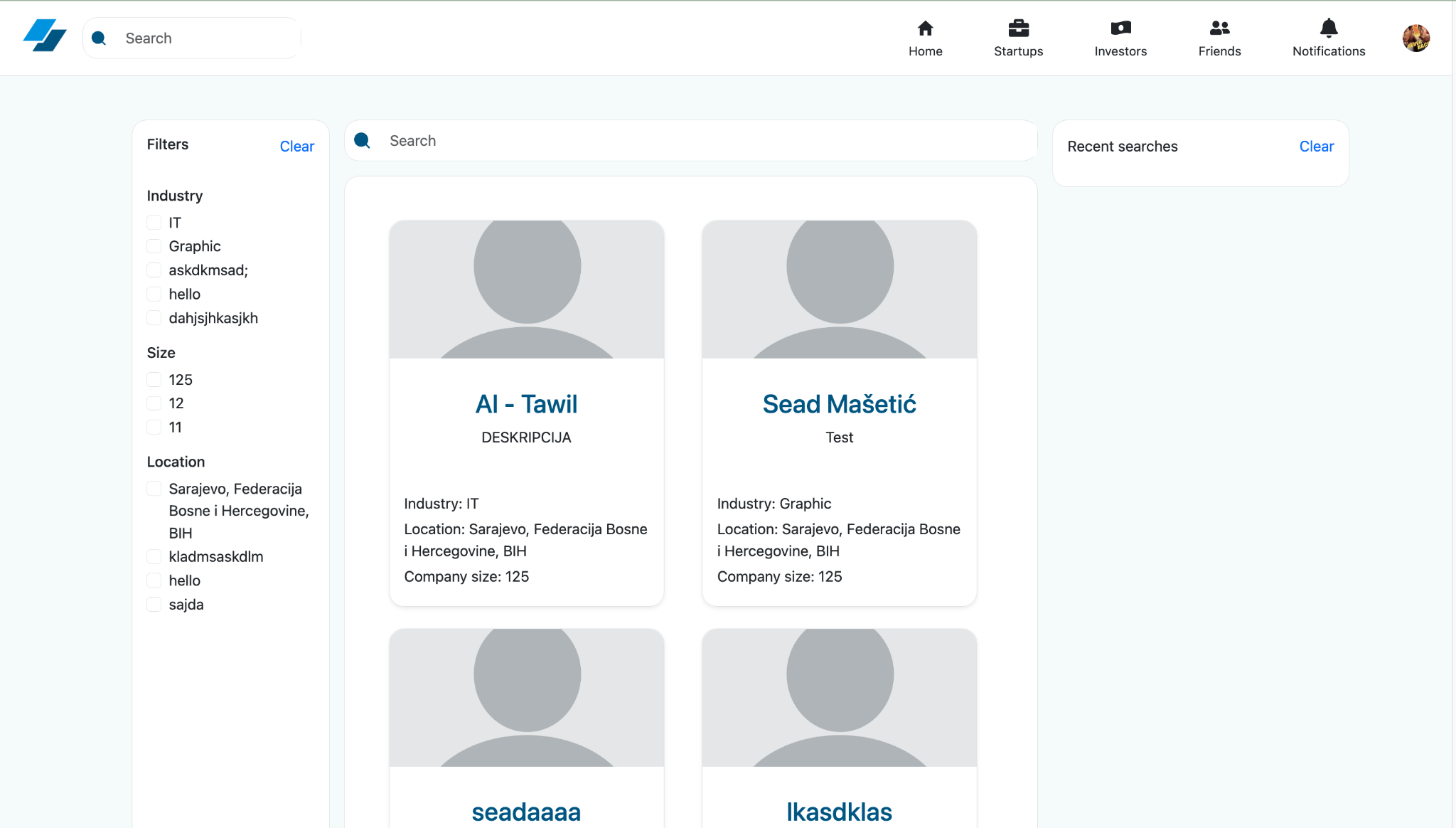


### 2.5.3 Startup Management

* **Create / Edit / Delete Startup** Owners can add new startups, fill in details (name, description, industry, location, logo), and invite team members by user ID.

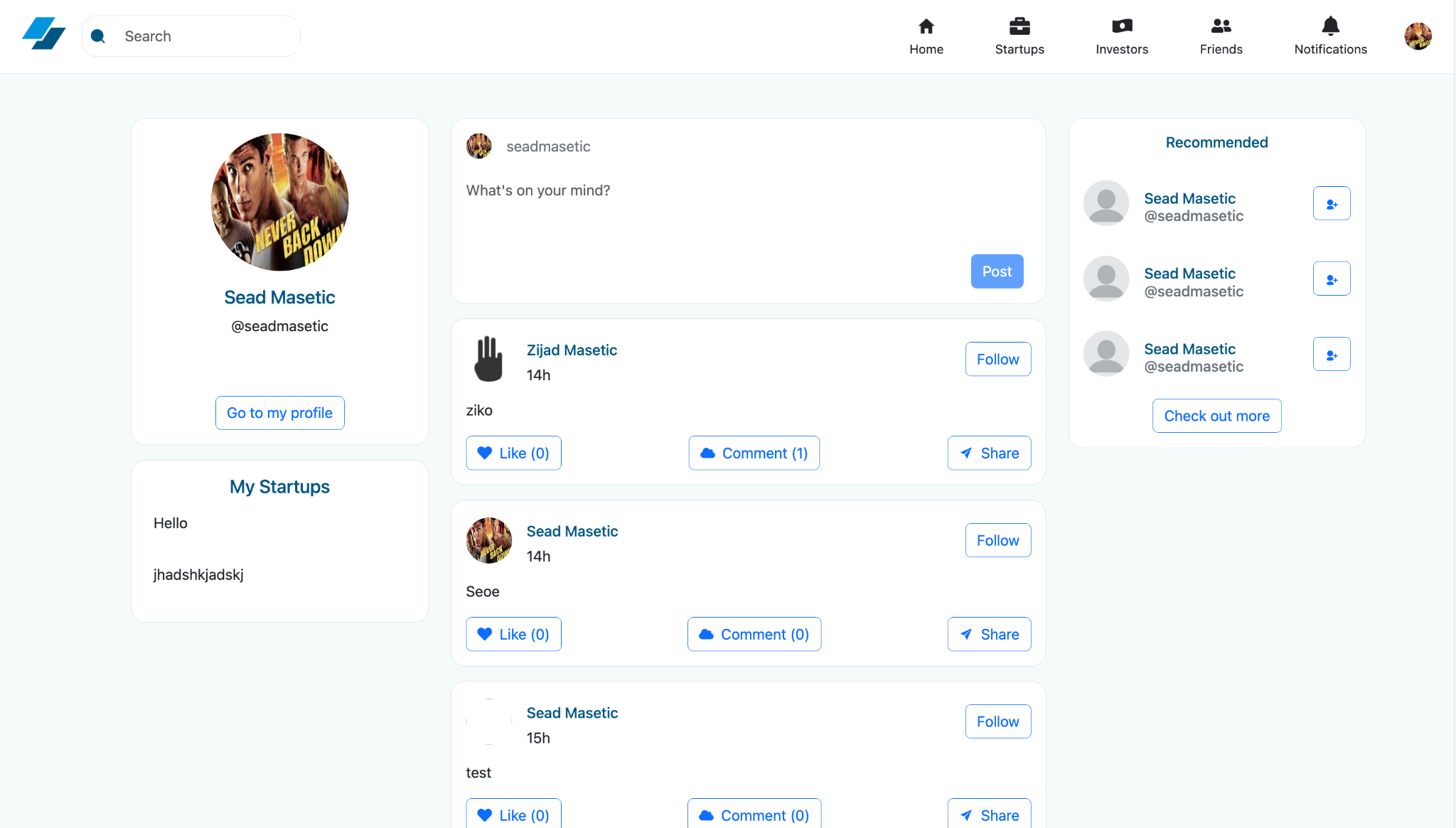


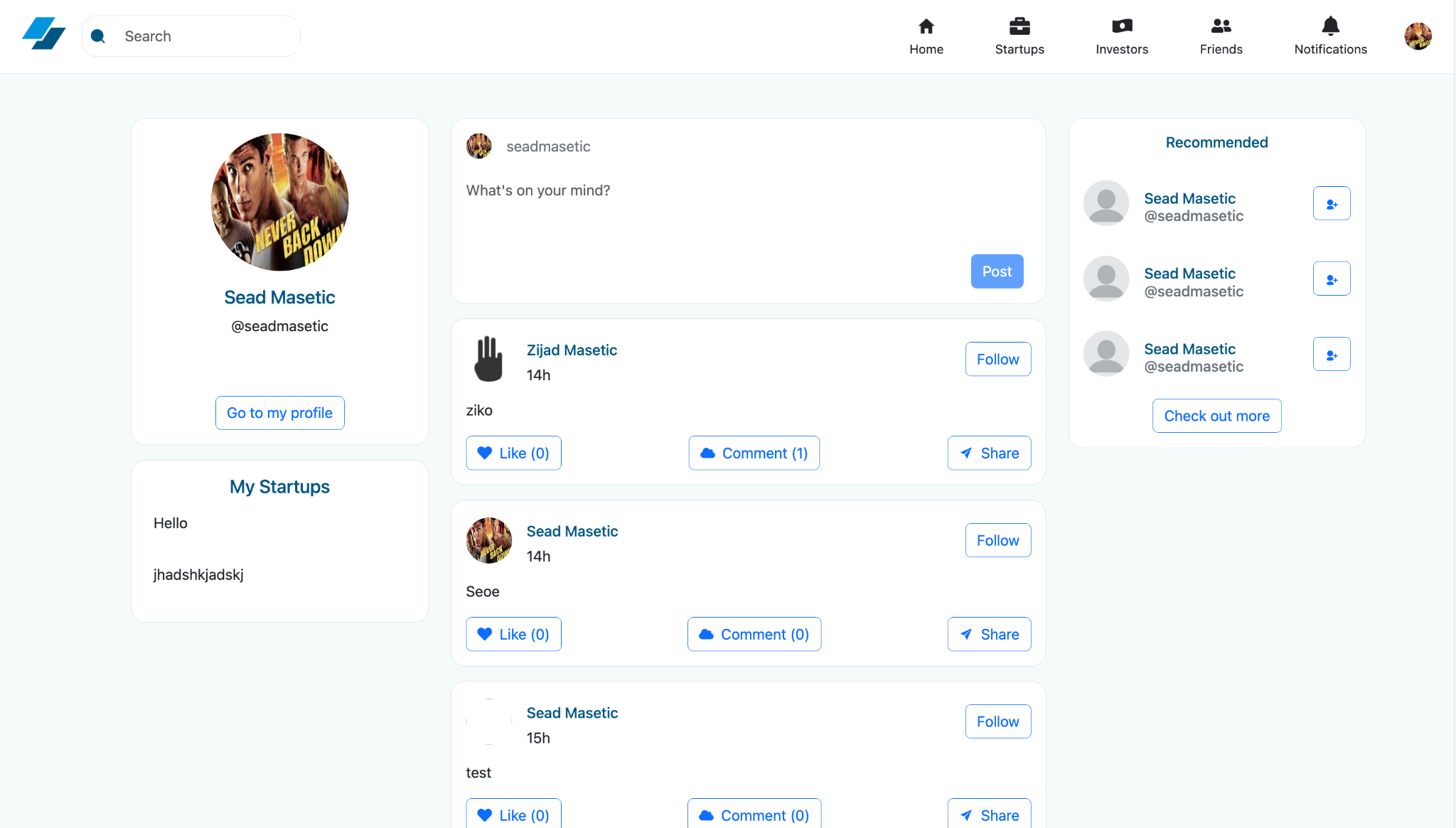
* **Search & Filter** Filter startups by name, industry, and location, or browse all.



### 2.5.4 Posts Feed (Homepage)

* **Create Post** Enter text to broadcast to your network.



* **Like / Unlike** Toggle likes on any post.
* **Comment** Write inline comments on posts and see comment counts.
* **Infinite Feed** Newest posts appear first; feed refreshes after any action.  
  

### 2.5.5 Security

* **JWT-Secured Endpoints** All protected routes verify the token via a Spring Security filter.
* **Input Validation & Error Handling** Both backend (Bean Validation) and frontend (toasts) alert users to invalid inputs.

## 2.6. Tests

For the IBU Startup project, we wrote **unit tests** using **JUnit 5** and **Mockito** to validate the business logic in our service layer.

Specifically, we focused on testing the behavior of the UserServiceImpl class, including:

* Saving a new user with proper password encoding and default values.
* Preventing user creation if the email already exists (conflict handling).
* Deactivating user accounts by updating the user’s status.

These tests are located in the **servicetests** package inside the **src/test/java** directory of the project under the name **UserServiceTest**. By mocking dependencies like UserRepository and PasswordEncoder, we ensured that our tests remained isolated and fast, without relying on the database or external services.

# 3. Conclusion

Overall, we are very satisfied with the implementation we have achieved for the IBU Startup application. We successfully built a full-stack platform where users can register, create and manage startup profiles, connect with others, and share updates via posts, likes, and comments. Integrating Spring Boot on the backend with our React frontend proved straightforward, and although we encountered the usual minor hiccups in wiring API routes and handling CORS, nothing was insurmountable.

Looking ahead, there are several enhancements we’d like to pursue:

* **Social features**: Add a dedicated “social page” for startups—allowing richer media uploads (video, slides), threaded discussions, and event listings.
* **AI-driven recommendations**: Incorporate machine-learning models to suggest relevant startups, jobs, or collaborators based on user behavior and profile data.
* **Personalized feeds**: Build a dynamic newsfeed algorithm that learns each user’s interests over time and surfaces the most engaging content first.

The most challenging area was ensuring smooth frontend-to-backend communication—especially around authentication, file-upload endpoints, and handling nested JSON payloads—but iterating on error messages and logging quickly got us to a stable integration.

In conclusion, this project has been an excellent proof of concept, demonstrating how multiple features—user management, networking, content sharing, and search—can come together in a single, coherent application. We believe the IBU Startup platform has real potential to evolve into a production-ready system that helps entrepreneurs, investors, and job-seekers connect in an innovative, one-stop environment.

# 4. Individual Contributions

**Fikret Zajmović (50%)**

He was primarily responsible for the backend development and overall system architecture. His contributions included:

* **System Analysis and Design (SAD) Diagrams**: For the System Analysis and Design part, we created diagrams like Activity, Sequence, and Class diagrams to better understand how the system should work and what it needs to do. These helped us plan things out clearly before we started coding.
* **Backend & API Development:** Designed and implemented the database using PostgreSQL and built the server-side logic with Spring Boot, developing RESTful APIs to enable efficient data retrieval and seamless communication between the frontend and backend.
* **Roadmap & Release Planning:** Defined project milestones and structured development into sprints, creating a clear and organized roadmap to guide the development process.
* **Design Patterns:** Applied established design patterns to create scalable, maintainable, and efficient code architecture throughout the project.
* **Unit Testing:** Wrote comprehensive unit tests using JUnit and Mockito to validate business logic, ensure code reliability, and facilitate early detection of bugs.

**Sead Mašetić (50%)**

He focused on the frontend development and user experience aspects of the project. His contributions included:

* **Frontend Development:** Built a responsive and user-friendly interface using React.js, focusing on performance, usability, and seamless user experience.
* **Backend Integration:** Integrated frontend components with John’s API endpoints to ensure smooth and reliable data flow between the client and server.
* **Deployment:** Dockerized the entire application and deployed both the React frontend and Spring Boot backend on Render, ensuring a smooth and scalable production setup, also deploying the database on Neon.
* **Architectural Pattern:** Implemented a layered architecture to separate concerns across the application, enhancing scalability, maintainability, and ease of testing.
* **Requirements Analysis:** Gathered and defined both functional and non-functional requirements to ensure the system meet user needs, performance expectations, and quality standards.