



---

# Design Document

## SeekMyTeam

---

Brian QUINN  
Abhijeet CHAKRABARTI  
Shengqi WANG  
Sanjit SAMA

TEAM #16

January 31, 2019

# Contents

<b>1</b>	<b>Purpose</b>	<b>2</b>
1.1	Functional Requirements . . . . .	2
1.2	Non-Functional Requirements . . . . .	3
<b>2</b>	<b>Design Outline</b>	<b>4</b>
2.1	High Level Overview . . . . .	4
2.2	Components . . . . .	4
<b>3</b>	<b>Design Issues</b>	<b>6</b>
3.1	Functional . . . . .	6
3.1.1	How should we implement registration with a valid Purdue email? . . . . .	6
3.1.2	How should we manage users listing their skills? . . . . .	6
3.1.3	How should we implement applying to a project? . . . . .	6
3.2	Non-Functional . . . . .	7
3.2.1	What type of architecture should we use? . . . . .	7
3.2.2	What front end framework should we use? . . . . .	7
3.2.3	What user interface tools should we use? . . . . .	8
3.2.4	What back end language should we use? . . . . .	8
3.2.5	What sort of protocol will we expose for the API? . . . . .	8
3.2.6	What web page design scheme should we use? . . . . .	9
3.2.7	What database should we use? . . . . .	9
<b>4</b>	<b>Design Details</b>	<b>10</b>
4.1	Class Diagram . . . . .	10
4.2	Class Descriptions . . . . .	10
4.3	Class Interactions . . . . .	11
4.4	State Diagram . . . . .	11
4.5	Sequence Diagrams . . . . .	12
4.5.1	Login . . . . .	12
4.5.2	Creating a New Post . . . . .	13
4.5.3	Managing Profile . . . . .	14
4.6	User Interface Mockups . . . . .	15
4.6.1	Registration Page . . . . .	15
4.6.2	Login Page . . . . .	16
4.6.3	Home Page . . . . .	17
4.6.4	Profile Page . . . . .	18

# 1 | Purpose

Attending a prestigious university such as Purdue, many students have a drive to create and innovate. Our goal is to allow for an exclusive and collaborative space for Purdue students to seek projects they can work on or for existing teams to find new qualified students to best further their ideas. Our incubator is quite unique as much of our competitors focus on either pure networking or freelancing opportunities.

---

## Functional Requirements

---

### I. Posts Feed for Projects

- As a user, I want to be able to post a project that I am looking for collaboration with.
- As a user, I want to be able to edit my project posting.
- As a user, I want to be able to delete my project posting.
- As a user, I want to be able to view projects posted by other users.
- As a user, I want to be able to apply to projects posted by other users.
- As a user, I want to be able to find contact information for the user posting the project.
- (If time permits) As a user, I want to be able to apply to a project by clicking a button that will send an automatic email notification to the poster.
- (If time permits) As a user, I want to be able to invite a user to apply to a project by clicking a button that will send an automatic email notification to the user.
- (If time permits) As a user, I would like to have multiple owners associated with a project.

### II. Search Functionality

- As a user, I want to be able to search for projects by posting name.
- As a user, I want to be able to view all of the users available on the platform.
- As a user, I want to be able to filter user results by certain tags.
- As a user, I want to be able to filter user results by certain tags.
- As a user, I want to be able to search for other users by name.
- 

### III. User Accounts

- As a user, I want to be able to register for an account.
- As a user, I want to be able to delete my account.
- As a user, I want to be able to log in to my account.
- As a user, I want to have a profile page.
- As a user, I want to be able to have people view my profile.
- As a user, I want to be able to view another user's profile.

## IV. User Profile

- As a user, I want to be able to upload a profile picture.
- As a user, I want to be able to change my profile picture.
- As a user, I want to be able to write a description of my experience.
- As a user, I want to be able to change my description of my experience.
- As a user, I want to be able to create a list of skills.
- As a user, I want to be able to change my list of skills.
- As a user, I want to be able to display my social media links for other users to view.
- As a user, I want to be able to change my social media links for other users to view.
- As a user, I want to be able to view a list of projects that I am a part of.
- (If time permits) As a user, I want to be able to view a list of projects I have applied to.

---

## Non-Functional Requirements

---

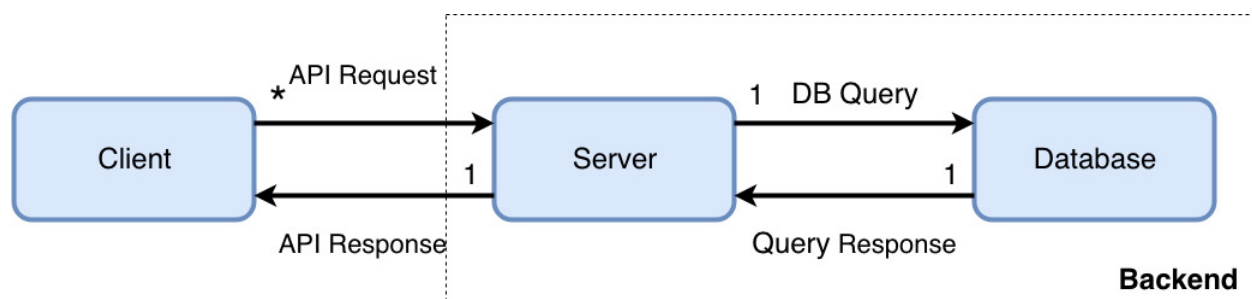
### I. Responsive and Intuitive Design

- As a developer, I want to ensure that the application runs as a web-app.
- As a developer, I want to ensure that our app's user interface is responsive.
- As a developer, I want to ensure that the app is interactive with respect to user's actions.
- As a user, I would like to edit my information and ensure it is accurate and consistent across all platforms.
- As a user, I would like the app to be intuitive and perform seamlessly without any lags.
- As a user, I would like my account and project information to be secured.

## 2 | Design Outline

### High Level Overview

The design model for SeekMyTeam is based on Client-Server architecture. The server will handle the business logic associated with postings, user operations and interactions, and the transmission of data to the database to maintain data consistency. The clients get updates pushed to them from the server when the state of the application changes, for example when a user posts a new project or applies to a posting. Clients then act upon these updates and re-render the page appropriately.



#### I. Client

Upon opening the SeekMyTeam web page, service calls will be made to the appropriate back-end endpoints of our application. The client will then receive responses from the back-end in the form of JSON objects, and render the information onto the user interface to be displayed. As the user interacts with the client through the web browser, service calls will continue to be made to the back-end to maintain data consistency with the front-end and allow business logic to be executed.

#### II. Server

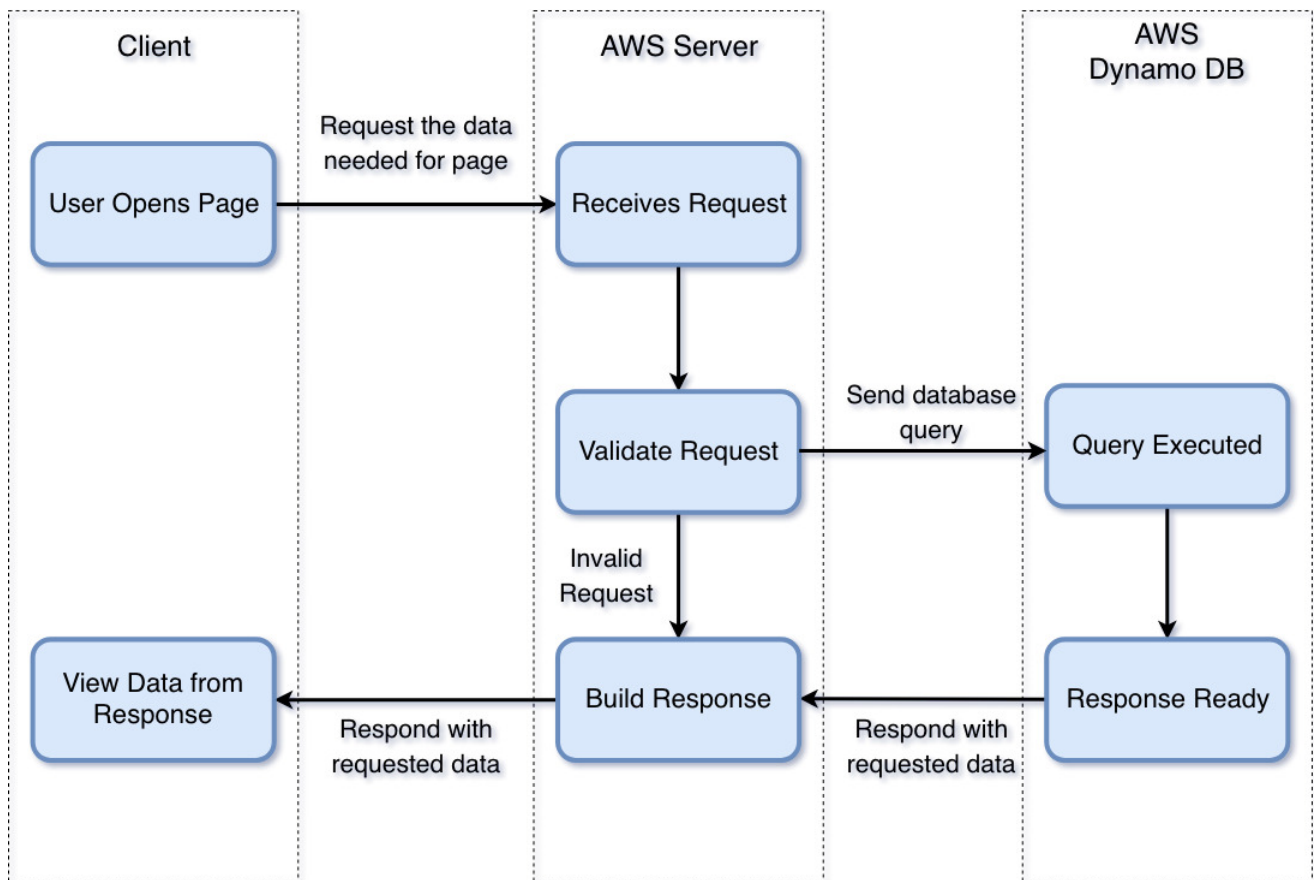
The server serves dynamic HTML/CSS files in addition to listening for service calls from clients. When called, the server will expect an HTTP request and perform some action on the data. It will then respond to the client accordingly and make necessary updates to the data in the database.

#### III. Database

The database is in charge of storing all the non-ephemeral data such as login information, profile information, and data about project postings. The server will query the database for this information when certain actions are taken such as when a user logs in.

### Components

Our web application will use a Client-Server model. The server will access our database to post and retrieve user account information, user skills information and individual posts information. The purpose of the server is to facilitate the communication of data and the client. The purpose of the client is to display UI to the user. The UI will consist of all user related fields and project information.



## 3 | Design Issues

---

### Functional

---

#### 3.1.1 How should we implement registration with a valid Purdue email?

- Use Purdue login portal
- Simply check for purdue.edu in email address
- **Verification code with sign-up**

One of the main attractions of our application is that users are exclusively Purdue students. Thus, in our registration system, we needed a way to verify that the user signing up is a Purdue student. We considered the possibility of using the Purdue career account portal to do this, however there is no documentation on how to implement it in a third party application. A simple way we could have verified a user was to check if their e-mail address ended with "@purdue.edu" however this could potentially allow for users to sign up with fake e-mail addresses in order to access our application.

Our final decision was that users will attempt to register with an e-mail address, which must end in "@purdue.edu". Our application will then automatically send them a verification code to that e-mail address which they will verify for us by typing it back into our application in their web browser.

#### 3.1.2 How should we manage users listing their skills?

- Allow for a plain-text list of any skill the user inputs
- **Drop-down list of pre-defined generalized skills**
- Drop-down list of pre-defined specific skills

In order for our users to connect more fluidly, we thought it would be good idea for a user to be able to list the skills they have experience with on their profile, and to allow project posters to request users with certain skill sets. One way we thought of doing this was just allowing the user to enter their own skills manually in a text box, but we realized that this data would be hard to leverage for useful operations such as sorting and filtering, as some users might not refer to the same skill by the same name i.e. one user lists "node" as a skill while another lists "Node.js".

Our solution to this is to pre-populate a drop down list of generalized skills such as "Front-End Development" or "Project Management", which should allow users to match skill sets easily. We chose to avoid skills such as specific languages or tools to avoid users failing to connect due to granularity problems, i.e. a project poster looks for someone with React experience while a user has AngularJS experience when in reality the project could have been developed using either technology.

#### 3.1.3 How should we implement applying to a project?

- **List contact information for project poster**
- Apply button with automated email notification system

One of the most important features of our application is the way in which we will actually allow the users to connect with each other. When a user finds a project they like, the question arose of how he/she will notify the poster that they are interested. The way we decided to implement this was just to list the email address of the project poster and allow a potential collaborator to get in contact with them in the manner that they please.

However, as noted in the project backlog, if time permits we would like to have an apply button for each project. This button, when clicked, would send an automated email to the project poster saying that the specific user has requested to collaborate on the project. The email would also contain a direct link to the applicant's profile on our platform to allow the poster to see what their abilities and contact information are.

---

## Non-Functional

---

### 3.2.1 What type of architecture should we use?

- Monolithic Architecture
- Micro-Services Architecture
- **Client-Server Architecture**

As SeekMyTeam is a web-application, we chose a Client-Server architecture. To elaborate on this, the clients will be the users on their browsers while the server, running a node.js web-app, will keep track of the state of the application and push updates to clients when changes occur. In addition to this ephemeral data, it will also keep permanent data in a database.

On the back-end side of things, we could potentially implement the backend as a Monolith, or with Micro-Services. We've chosen to go with an extensible monolith since we believe this application will be simple enough to justify not having the added complexity of inter-service communication.

### 3.2.2 What front end framework should we use?

- React
- **AngularJS with TypeScript**
- Vanilla JavaScript

We decided to use AngularJS as our front-end framework as we believe the organizational structure of it will allow us to maintain our web application easily on the front-end. React was another viable option for a front-end framework, however it tends to have more flexible design options, which we believe will create more overhead than necessary to build a webapp of this scope. JavaScript without a framework could very well have been used for this project as well, but we believe AngularJS will offer us built in development tools for the front-end display that will allow us to develop more rapidly.



### 3.2.3 What user interface tools should we use?

- HTML/CSS from scratch
- Foundation
- **Bootstrap**

To create a decent looking front-end from scratch would take a significant amount of time, which is why frameworks like Bootstrap/Foundation/Pure exist. They allow for rapid prototyping and creation of beautiful looking interfaces with significantly reduced time. Due to the authors being experienced with Bootstrap, we went with this option.

### 3.2.4 What back end language should we use?

- PHP
- Python
- Java
- **node.js**

When it comes to deciding the language for the back-end server, there are a couple of very solid options that are generally used for web development. Our focus was finding a language that allowed for very simple real-time communication between the browser and the server. PHP is mostly designed to render static pages, or pages with slight dynamic content and not handle active communications so it was instantly off the table. Python has some decent frameworks for real time web-apps but they are very immature and buggy.

Java has some very mature frameworks, however the verbosity of Java and its frameworks was an instant turn off. In contrast, node.js allows for quick prototyping, and has been tested and proven to be used for exactly these kind of applications. It also has a great deal of supporting libraries specifically tied for web development.

### 3.2.5 What sort of protocol will we expose for the API?

- Proprietary
- Messaging
- **RESTful JSON**

A method of communication between the browser and server has been established, but now we need to decide what sort of protocol they'll use. We could go with a proprietary text/binary protocol that we create by ourselves, however this is some very heavy NIHS (Not Invented Here Syndrome). We could go with messaging protocols such as those used by RabbitMQ/General message queuing systems, however we decided to use a RESTful JSON API simply because that is the current standard for web applications and allows for a great deal of interoperability such as being able to be consumed by an Android/iOS/Desktop app in addition to the web-app itself.

### 3.2.6 What web page design scheme should we use?

- Fixed
- Adaptive
- **Responsive**

A fixed design schema would mean that our application would be limited to looking good on only one size of screen, which is not really acceptable considering we want it to be usable by anyone with a web browser.

For ensuring a consistent user experience, Responsive design is a better choice as it reacts to the size of a user's screen, thus, optimizing the browsing experience.

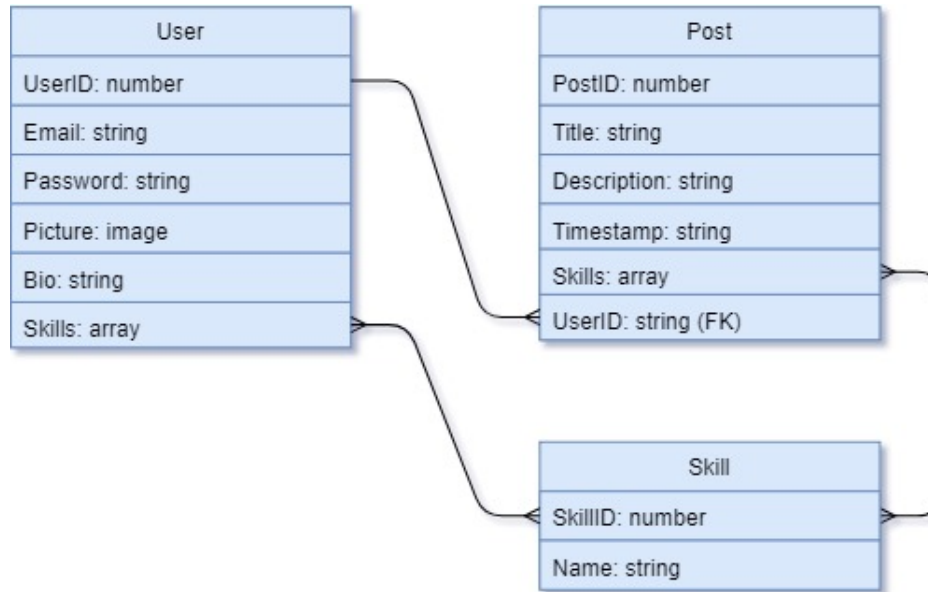
### 3.2.7 What database should we use?

- **DynamoDB**
- MySQL
- MongoDB

We chose to use DynamoDB for our application's database needs. Since we will be hosting our application on AWS, DynamoDB will integrate easily with our application, allowing for more rapid development. MongoDB and MySQL were other options that members of our team have experience with, however we believe DynamoDB will be easier to configure to our specific needs.

## 4 | Design Details

### Class Diagram



### Class Descriptions

- User**
- A user will be uniquely identified by a UserID number that will be generated for them
  - A user will have an email address that they register with
  - A user will have a password that authenticates and authorizes them to use their account
  - A user may have a profile picture if they choose to upload one
  - A user will have a bio that briefly describes themselves and their achievements
  - A user will have an array of **Skill** objects which describe skills that they have experience with
- Post**
- A post will be uniquely identified by a PostID number that will be generated upon post creation
  - A post will have a title chosen by its creator
  - A post will have a description that provides collaborators with an idea of what the project owner is trying to build
  - A post will have a timestamp which will be useful for keeping track of how old the post is
  - A post will have a list of **Skill** objects which describe the skills being sought out by the project owner
  - A post will have a UserID associated with it which is a foreign key into the users table. This ID will keep track of who created each post.

- Skill**
- A skill will be uniquely identified by a SkillID number that will be auto generated
  - A skill will have a name that describes itself

---

## Class Interactions

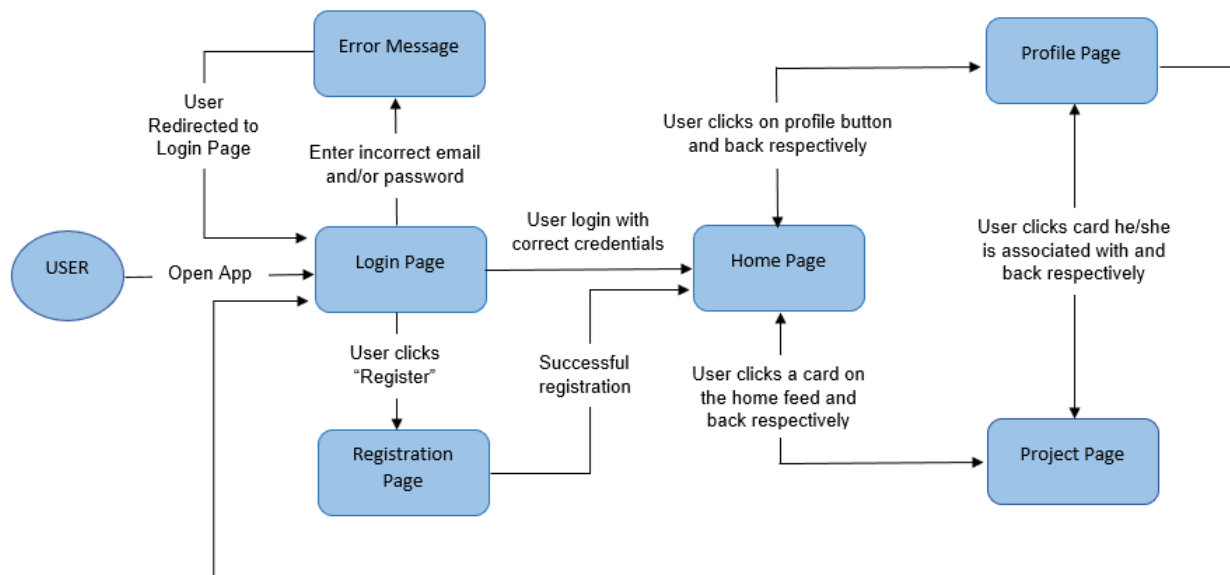
---

The classes above comprise the conceptual idea of our application. A User interacts with a Post when it is creating or changing a Post that it owns. Users and Posts both interact with the Skill class by maintaining a list of skills that are important to them. We chose to make a skill its own class so that we can maintain lists of Skill objects in both the User and Post classes. This allows for easy skill comparisons between a User and a Post, rather than slow and complicated string matching functions, which will enhance sorting, filtering, etc.

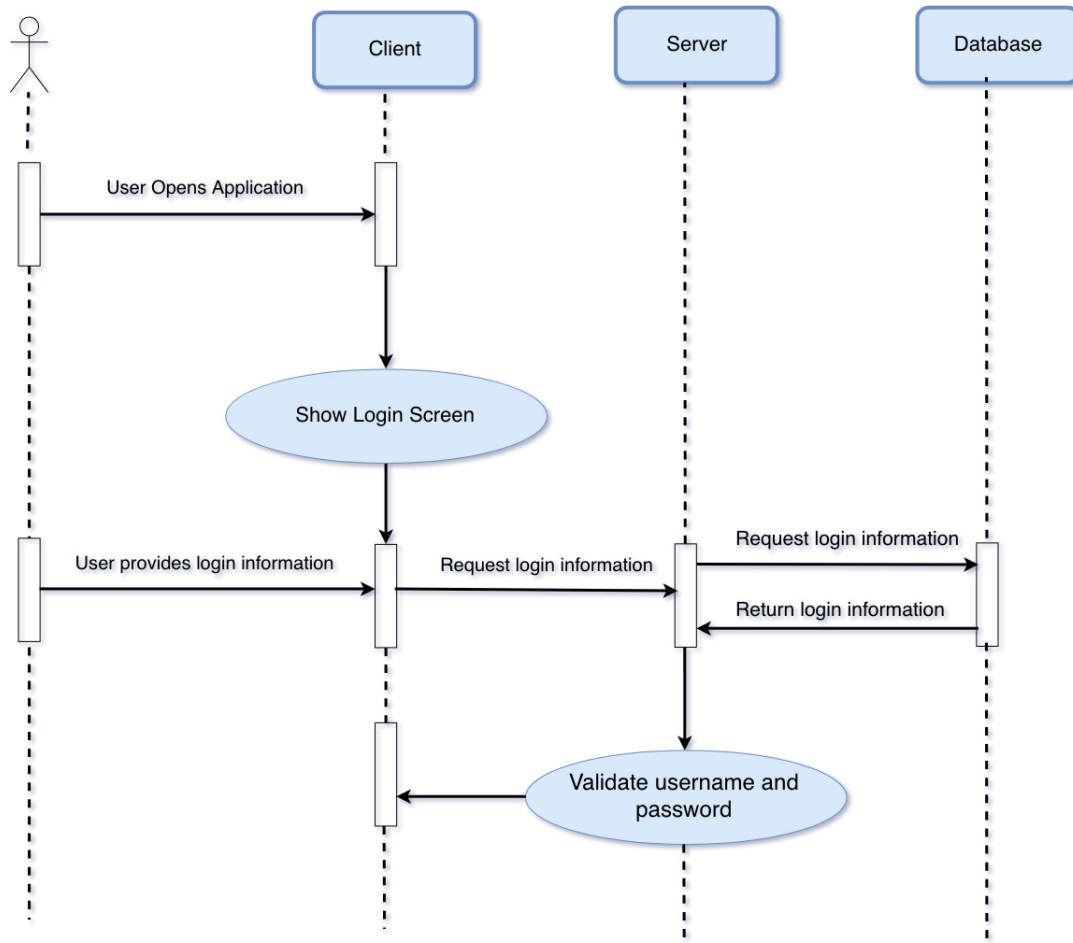
---

## State Diagram

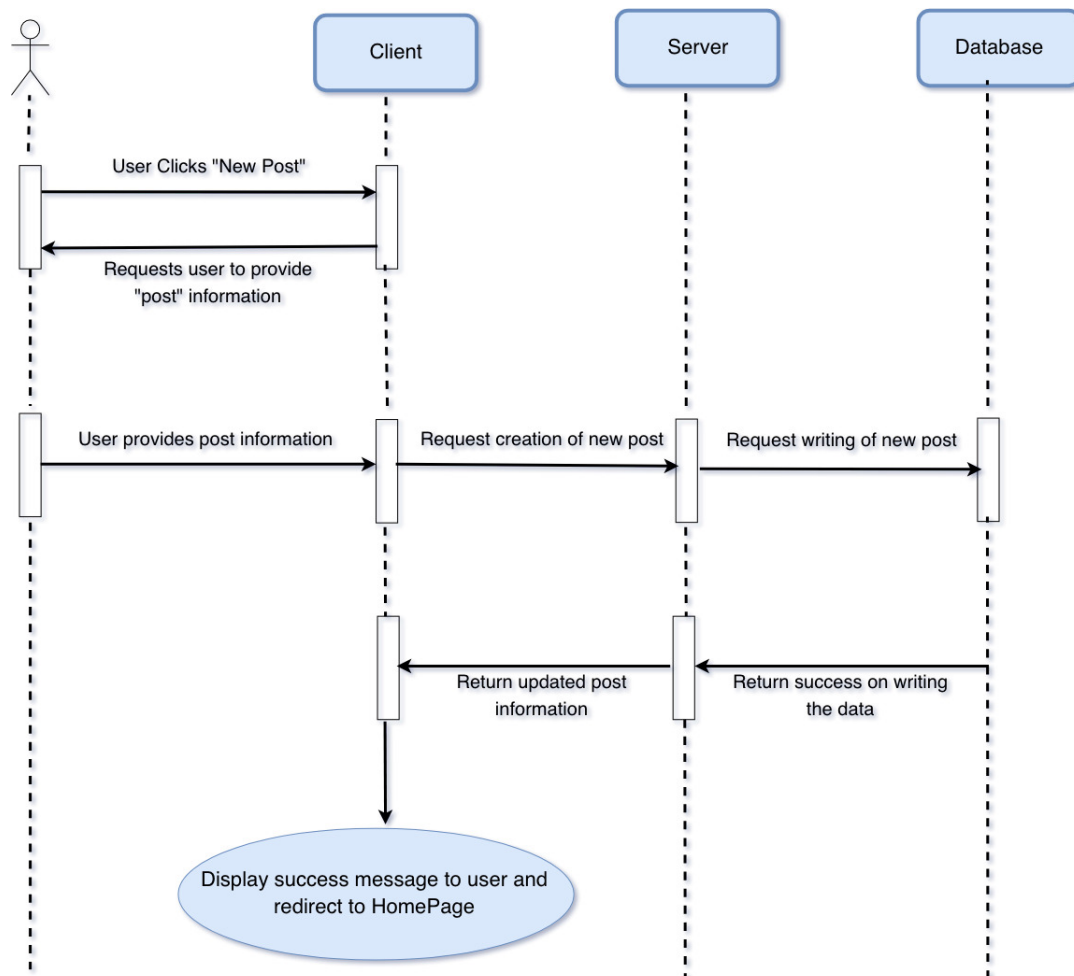
---



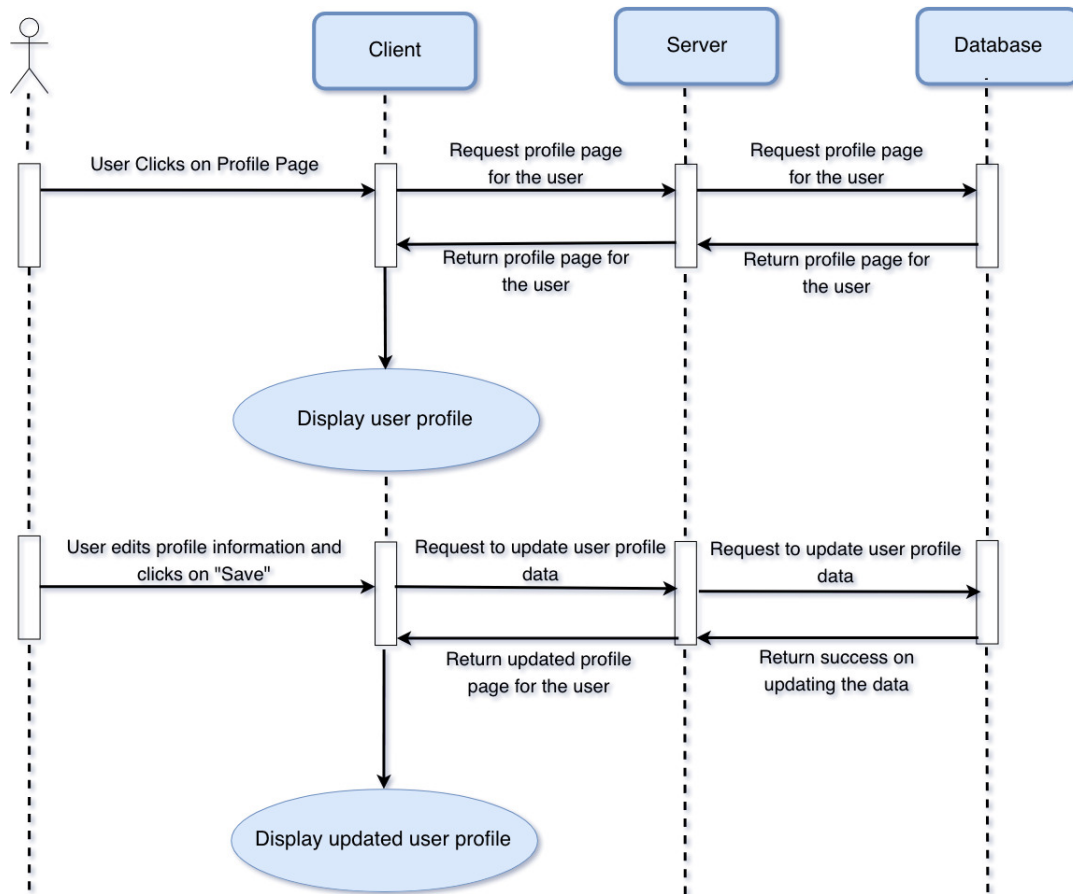
## 4.5.1 Login




## 4.5.2 Creating a New Post



### 4.5.3 Managing Profile



### 4.6.1 Registration Page



Seek My Team

## Sign Up

Full Name

Email

Password

Continue Back



## 4.6.2 Login Page


# Seek My Team

Ever wanted to join a startup?

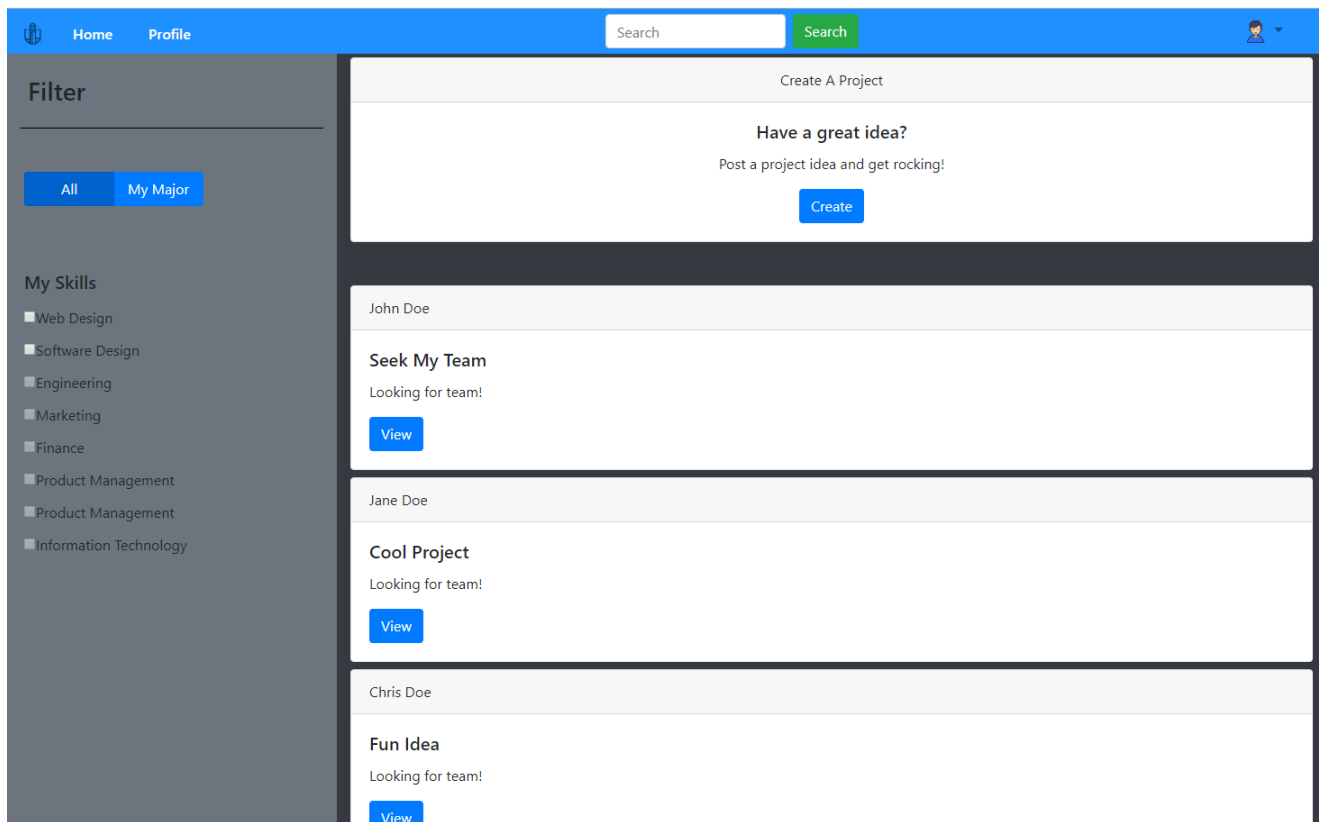
NOW YOU CAN.

Simply scroll through the feed and start working ASAP!

## Login

The logo for 'Seek My Team' is a blue square containing a white icon of a person with arms raised, and the text 'Seek My Team' in white.

## 4.6.3 Home Page



## 4.6.4 Profile Page

