

Indian Institute of Information Technology, Nagpur

Project Report





Threads App

for Software Architecture

By Rohan Udhwani (BT21CSE092)

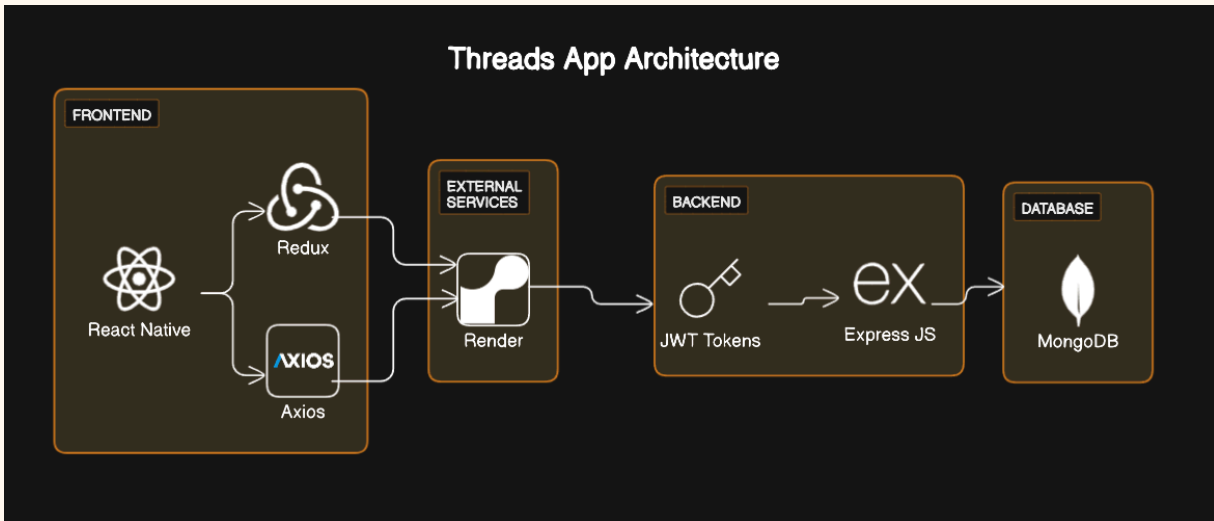
Project Repo Link: [Github](#)

Table of Contents

Project Report.....	1
Table of Contents.....	2
 Threads App Architecture: An Overview.....	3
 Design Decisions.....	5
 Design Principles.....	6
 Tech Stack.....	8
Use Case Diagram.....	9
Sequence Diagram.....	10
State Transition Diagram.....	11
Class Diagram.....	12
BPMN Diagram.....	13
Team Contributions.....	14
Timeline.....	17
Areas for Improvement.....	18
Benefits of Software Architecture in Developing the Threads Clone.....	20
ReadMe.MD.....	22
Threads.....	22
Features.....	22
Installation.....	22
Screenshots.....	23
Getting Started.....	24
Feedback and Contributions.....	25



Threads App Architecture: An Overview



Model-View-Controller (MVC) Design Paradigm

The Threads app adopts the Model-View-Controller (MVC) architectural pattern to ensure a clear separation of concerns and promote maintainability and scalability. Below, we delve into how each component aligns with the MVC pattern:

1. Model:

- The Model component serves as the data layer, encompassing data structures, business logic, and data management functionalities.
- It facilitates operations such as data retrieval, manipulation, and storage, maintaining data integrity and consistency.
- Within the Threads app, the Model encompasses classes or modules responsible for managing user profiles, message threads, notifications, and other data-related tasks.

2. View:

- The View component represents the presentation layer, responsible for rendering the user interface and displaying data to users.
- It includes user interface elements such as screens, widgets, buttons, and input fields.

- In the Threads app, the View consists of activity layouts, fragments, custom views, and UI elements designed using XML layouts or programmatically.

3. Controller:

- Acting as an intermediary between the Model and the View, the Controller handles user input, processes requests, and coordinates interactions.
- It interprets user actions and translates them into operations performed on the Model or updates to the View.
- Components within the Threads app's Controller include activity or fragment classes, event listeners, and other elements responsible for managing user input and orchestrating interactions with the Model and View.

Conclusion:

- By adhering to the MVC design paradigm, the Threads app architecture ensures modularization, maintainability, and scalability. Through the clear separation of concerns and organization of components based on their responsibilities, the application achieves robustness and extensibility while enhancing the user experience.



Design Decisions

In crafting the Threads app, we've taken a tech-savvy approach, leveraging cutting-edge tools and frameworks to deliver a slick, seamless social media experience. Here's a rundown of our design decisions:

1. React Native Awesomeness:

- With React Native at the helm, we're diving into cross-platform development nirvana. It's our ticket to building the Threads app for both iOS and Android with just one codebase. Talk about coding efficiency on steroids!

2. Express.js Expressway:

- On the backend, we're cruising down the Express.js expressway. This lightweight Node.js framework is our turbocharged engine for handling HTTP requests, routing, and middleware magic. Smooth sailing ahead!

3. MongoDB Magic:

- MongoDB is our go-to NoSQL database for storing all that juicy social media data. Its flexible, scalable nature is the perfect fit for housing user profiles, posts, comments, and likes – all the essentials for a thriving Threads community.

4. Redux Rock 'n' Roll:

- Enter Redux, our rockstar state management solution. With Redux orchestrating the show, we're orchestrating state updates like a symphony conductor. It's the secret sauce behind our consistent, responsive user experience.

5. Render's Cloud Paradise:

- Last but not least, we're soaring high on Render's cloud platform. Automatic scaling, SSL certificate management, continuous deployment – it's like having our own personal cloud paradise. Hosting and scaling? Piece of cake!

With this powerhouse lineup of tech goodies, Threads isn't just another social media app – it's a next-level, turbocharged social experience. Get ready to dive in and experience social media like never before! 🚀



Design Principles

1. *Should not suffer from “Tunnel Vision”:* This principle suggests that our design considers the entire system rather than focusing solely on individual components or features. This means ensuring that each file (such as `index.js`, `App.js`, etc.) contributes to the overall functionality and cohesiveness of the application. It also means avoiding overly specific or isolated solutions that may not integrate well with the rest of the system.

2. *Traceable to analysis model:* To adhere to this principle, the design is directly linked to the analysis model, which outlines the requirements and specifications of the system. For instance, `User.js`, `ProfileScreen.js`, `LoginScreen.js`, etc., reflects the user-centric features and functionalities identified during the analysis phase. Any deviation from the analysis model may result in discrepancies between the intended functionality and the implemented system.

3. *Do not “Reinvent The Wheel”:* This principle advocates for reusing existing solutions and components rather than creating new ones from scratch. In this project, we may leverage libraries or frameworks specified in `package.json` or `yarn.lock` to handle common tasks such as routing (`StackNavigator.js`), state management (`UserContext.js`, `store.js`), and UI components. By utilizing existing solutions, we can save time and effort while benefiting from established best practices.

4. *Exhibit uniformity and integration:* Consistency across files is crucial for maintaining readability, scalability, and collaboration within the project. This involves adhering to consistent naming conventions (`index.js`, `User.js`, etc.), directory structures, coding styles, and design patterns. Additionally, ensuring seamless integration between different components (`App.js`, `StackNavigator.js`, etc.) promotes cohesion and facilitates easier maintenance and updates.

5. *Accommodate change:* Designing for change involves creating flexible and adaptable solutions that can evolve with evolving requirements. Our project is structured in a modular fashion, allowing individual components to be modified or replaced without impacting the entire system. Additionally, separating concerns (e.g., separating UI components from business logic) and minimizing dependencies (`package.json`, `yarn.lock`) can enhance the project's agility and resilience to change.

6. Design is not coding and coding is not design: This principle emphasizes the distinction between design (conceptualizing the structure and behavior of the system) and coding (implementing the design in code). While files like App.js, User.js, etc., represent the implementation aspect, the design should be documented separately, possibly in design documents, architectural diagrams, or requirements specifications. This separation clarifies the intended functionality and promotes better understanding.

7. Review to discover errors: Regular reviews of the design are essential for identifying and rectifying errors, inconsistencies, or deviations from the requirements. This involves peer reviews, code inspections, or design walkthroughs to ensure that each file accurately reflects the intended functionality, adheres to best practices, and aligns with the analysis model. Addressing issues early in the design phase helps prevent costly errors and rework later in the development process.



Tech Stack

React Native: React Native allows for cross-platform mobile app development, enabling the Instagram threads application to be efficiently developed for both iOS and Android platforms with a single codebase. It offers a smooth user experience, native-like performance, and a rich ecosystem of libraries and components.

Express.js: Express.js is a popular Node.js web application framework known for its simplicity, flexibility, and scalability. It's well-suited for building the backend API of the Instagram threads application, handling HTTP requests, routing, and middleware functionalities efficiently.

Node.js: Node.js is chosen for its event-driven, non-blocking I/O model, which makes it ideal for building real-time applications like Instagram threads. It enables handling concurrent connections effectively, facilitating the development of a fast and responsive backend for the application.

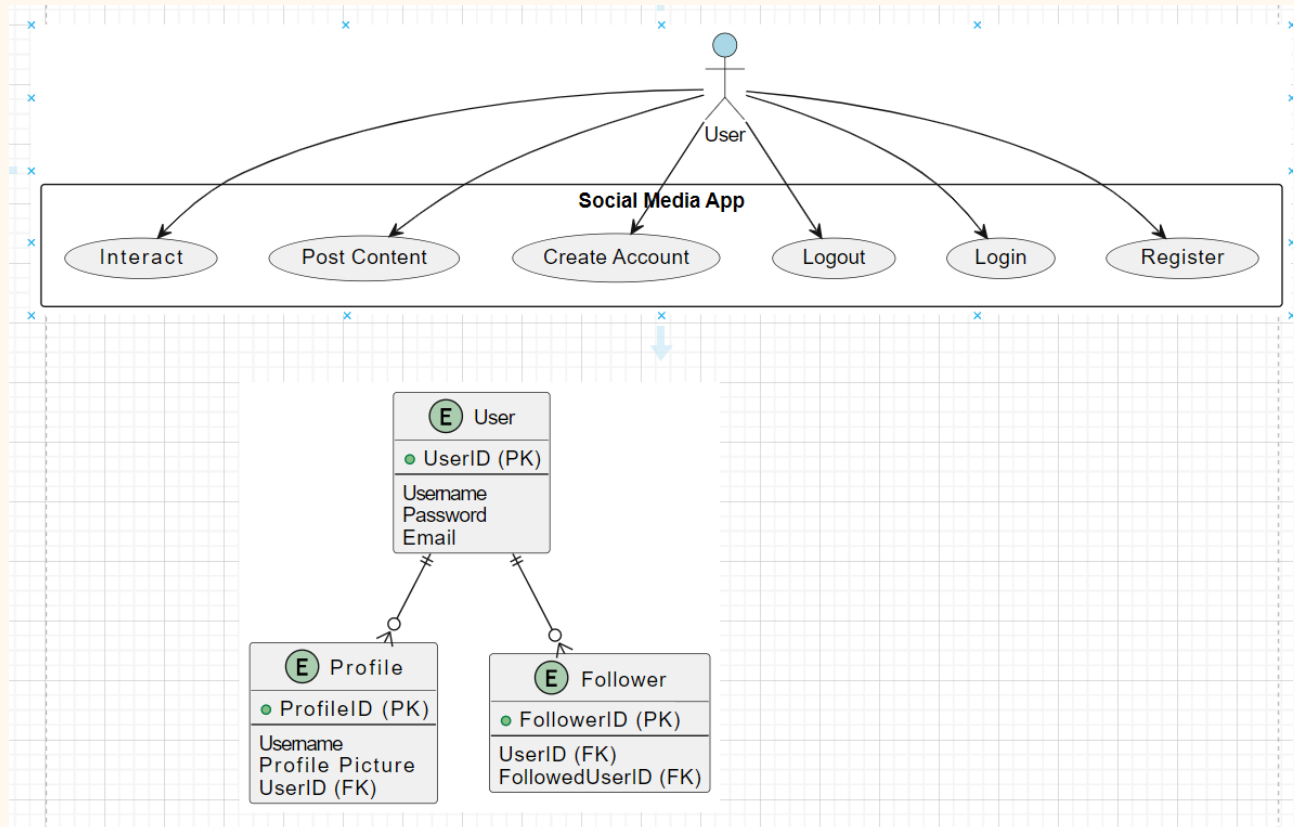
MongoDB: MongoDB is a NoSQL database known for its scalability, flexibility, and ease of integration with Node.js. It's a suitable choice for storing the unstructured or semi-structured data typical in social media applications like Instagram threads, such as user profiles, posts, comments, and likes.

Redux: Redux is a predictable state container for JavaScript applications, commonly used with React.js and React Native. It provides a centralized store for managing the application's state, making it easier to maintain and reason about the application's data flow, especially in complex applications like Instagram threads. Redux facilitates efficient state management, data persistence, and seamless communication between components.

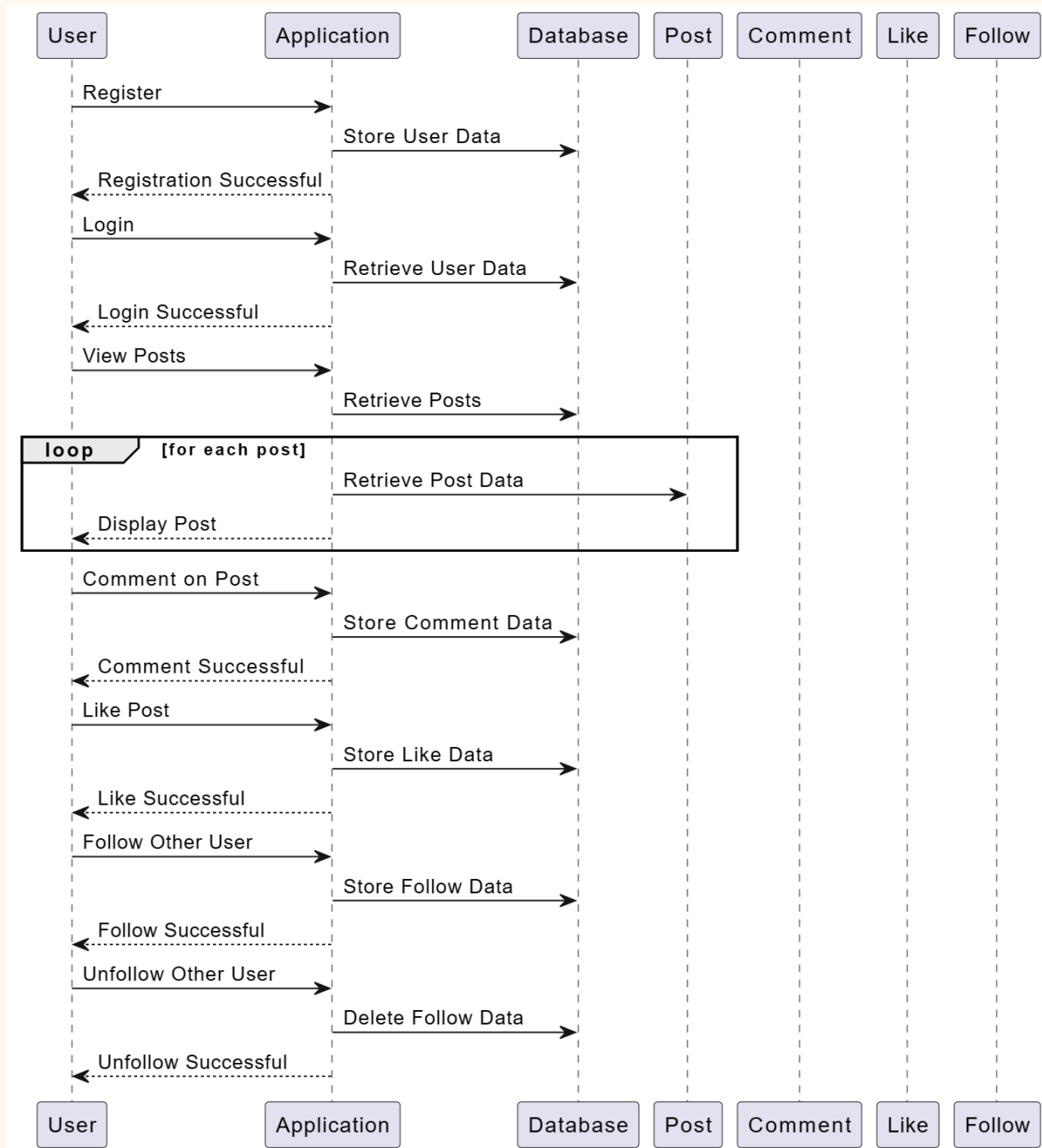
Render: Render is a modern cloud platform designed for hosting and scaling web applications and APIs. It offers features such as automatic scaling, SSL certificate management, and continuous deployment, simplifying the deployment and management of the Instagram threads application's backend and frontend components.

By leveraging these technologies, the Instagram threads application can be developed with efficiency, scalability, and maintainability in mind, providing users with a seamless and engaging social media experience across multiple platforms.

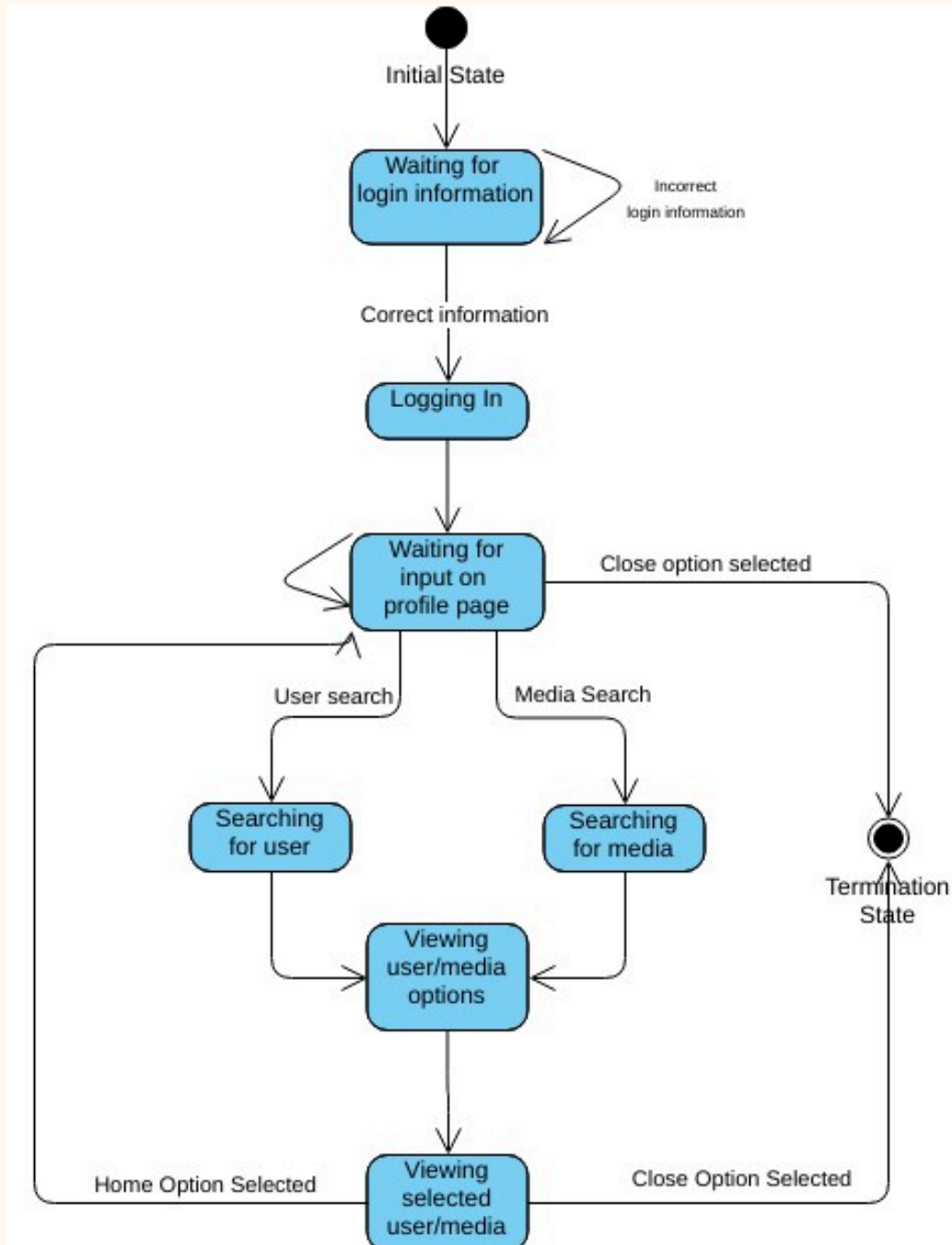
Use Case Diagram



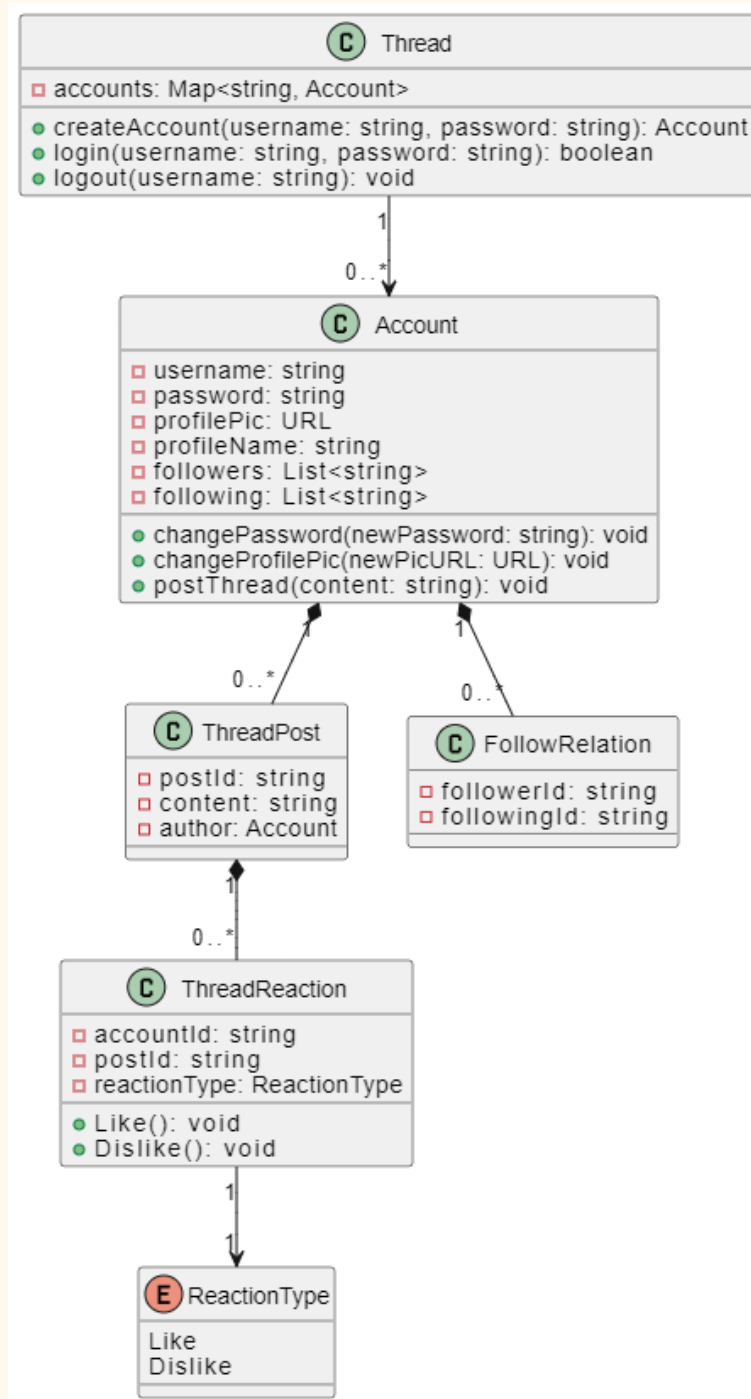
Sequence Diagram



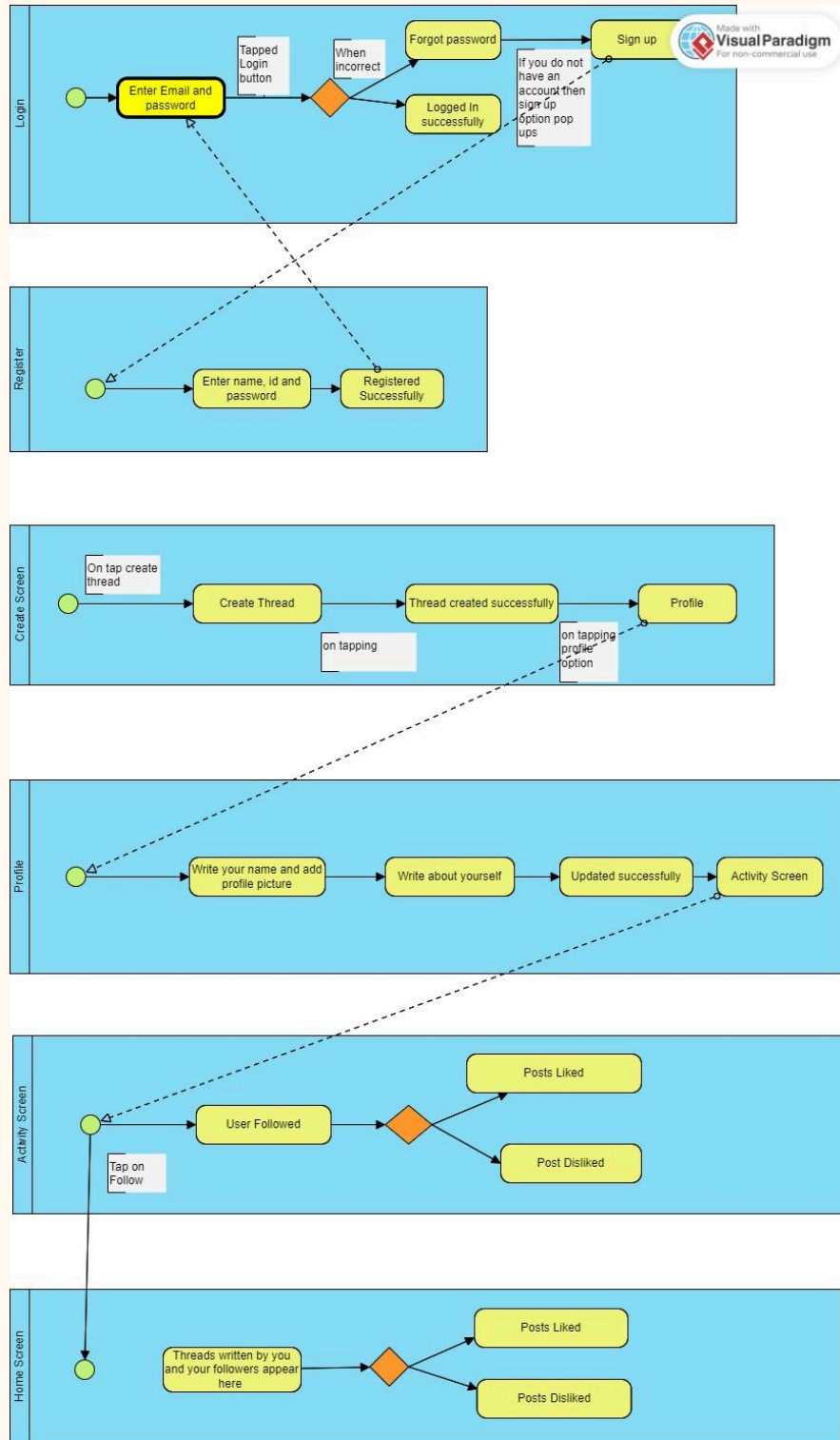
State Transition Diagram



Class Diagram



BPMN Diagram



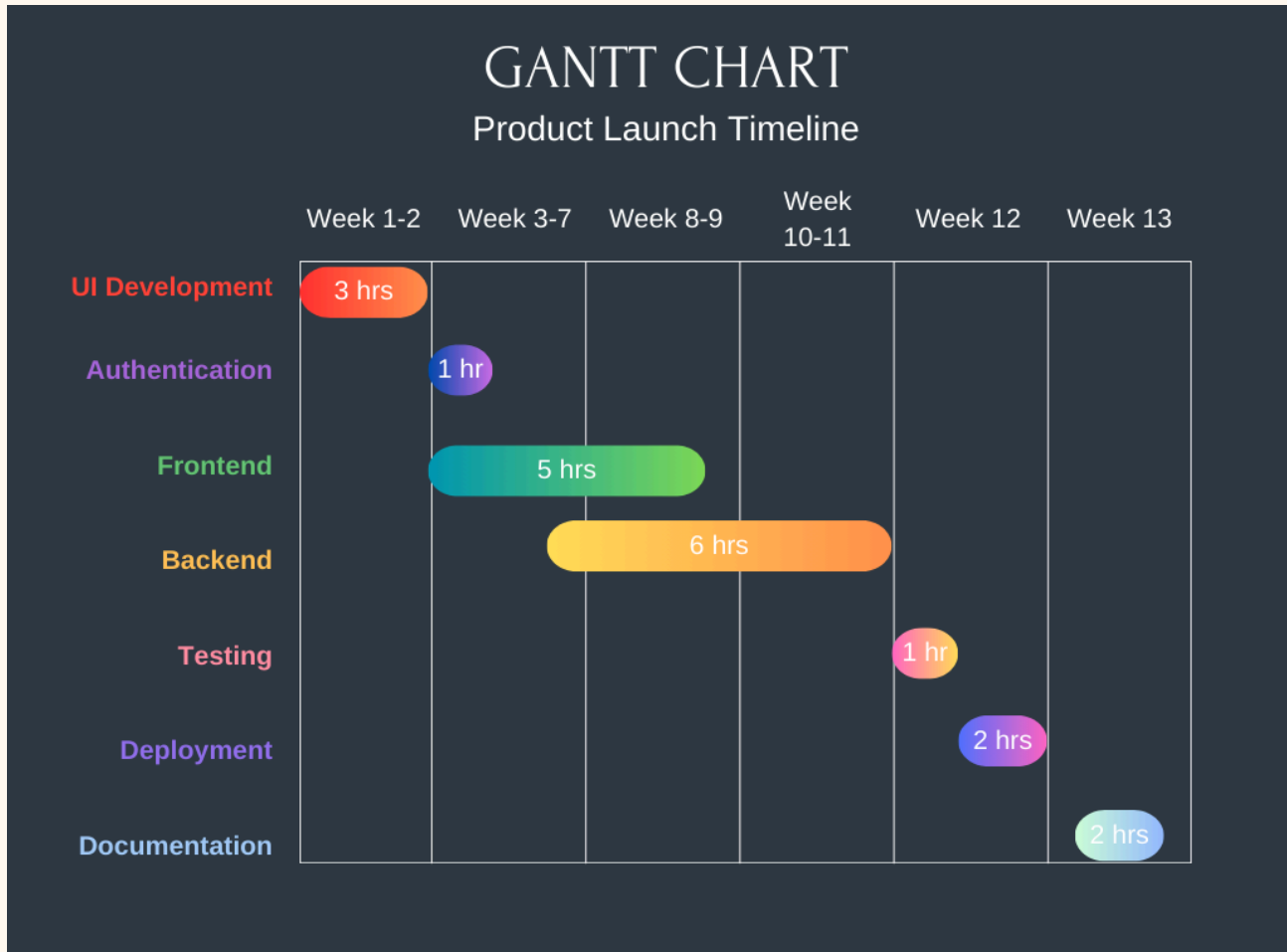
Team Contributions

Domains	Tasks
UI Development	<ul style="list-style-type: none"><input checked="" type="checkbox"/> Design the login screen.<input checked="" type="checkbox"/> Design the sign-up screen.<input checked="" type="checkbox"/> Create an onboarding screen with introductory information.<input checked="" type="checkbox"/> Design the profile screen.<input checked="" type="checkbox"/> Create the activity screen to display other users and enable following/unfollowing.<input checked="" type="checkbox"/> Design the create thread screen for users to write and post content.<input checked="" type="checkbox"/> Develop the feed screen to display already posted threads by the user and others.<input checked="" type="checkbox"/> Ensure consistency in UI elements across screens.
Authentication	<ul style="list-style-type: none"><input checked="" type="checkbox"/> Implement user authentication using email and password.<input checked="" type="checkbox"/> Send verification link to the user's email for email verification.<input checked="" type="checkbox"/> Create a function to send user verification email.<input checked="" type="checkbox"/> Handle email verification process in the app.<input checked="" type="checkbox"/> Enable users to sign in after email verification.
Frontend Development	<ul style="list-style-type: none"><input checked="" type="checkbox"/> Develop frontend components using React Native and Expo.<input checked="" type="checkbox"/> Implement navigation between screens using React Navigation.<input checked="" type="checkbox"/> Integrate Redux or Context API for state management if needed.<input checked="" type="checkbox"/> Handle user interactions and input validations.<input checked="" type="checkbox"/> Fetch data from backend APIs using Axios or Fetch API.<input checked="" type="checkbox"/> Display fetched data dynamically in UI components.
Backend Development	<ul style="list-style-type: none"><input checked="" type="checkbox"/> Set up a backend server using Node.js and Express.<input checked="" type="checkbox"/> Connect the backend to a MongoDB database for data storage.<input checked="" type="checkbox"/> Create MongoDB schemas/models for users, threads, and other entities.<input checked="" type="checkbox"/> Implement CRUD operations for managing user and thread

	<p>data.</p> <ul style="list-style-type: none"> <input checked="" type="checkbox"/> Develop API endpoints for user authentication, user data management, and thread management. <input checked="" type="checkbox"/> Implement JSON Web Tokens (JWT) for user authentication and authorization. <input checked="" type="checkbox"/> Ensure secure data handling and implement necessary security measures. <input checked="" type="checkbox"/> Create functions to handle sending emails for user verification and password reset.
Testing	<ul style="list-style-type: none"> <input checked="" type="checkbox"/> Perform unit testing for frontend components and backend API endpoints. <input checked="" type="checkbox"/> Conduct integration testing to ensure proper communication between frontend and backend. <input checked="" type="checkbox"/> Test different scenarios such as user login, registration, posting threads, following/unfollowing users, etc.
Deployment	<ul style="list-style-type: none"> <input checked="" type="checkbox"/> Deploy the backend server on a platform like Heroku, AWS, or Azure. <input checked="" type="checkbox"/> Build the React Native app for iOS and Android platforms. <input type="checkbox"/> Distribute the app for testing using services like TestFlight for iOS and Google Play Store Beta for Android. <input type="checkbox"/> Prepare the app for production release and publish it on the respective app stores.
Documentation	<ul style="list-style-type: none"> <input type="checkbox"/> Write comprehensive documentation for both frontend and backend code. <input checked="" type="checkbox"/> Document the API endpoints and their usage. <input checked="" type="checkbox"/> Provide instructions for setting up the development environment and running the app locally.
Feedback and Iteration	<ul style="list-style-type: none"> <input type="checkbox"/> Gather feedback from users and stakeholders for improvements. <input type="checkbox"/> Address any bugs or issues reported by users. <input type="checkbox"/> Iterate on the app based on feedback to enhance user experience and functionality.
Monitoring and Maintenance	<ul style="list-style-type: none"> <input type="checkbox"/> Set up monitoring tools to track app performance, errors, and user analytics. <input type="checkbox"/> Monitor server uptime and resource usage to ensure smooth operation.

	<input type="checkbox"/> Regularly update dependencies and libraries to maintain security and compatibility.
Community Building	<input type="checkbox"/> Engage with the app community through social media, forums, and developer communities. <input type="checkbox"/> Encourage user participation and feedback to foster a thriving user community. <input type="checkbox"/> Provide support and assistance to users facing issues or seeking help.

Timeline



Areas for Improvement

1. *Enhanced User Profiles:* Allow users to customize their profiles with profile pictures, bios, and other personalization options. This enhances the user experience and makes the platform more engaging.
2. *Explore Feature:* Implement a feature that allows users to discover new posts and users based on their interests, trending topics, or recommendations. This could improve user retention by keeping the content fresh and relevant.
3. *Direct Messaging:* Introduce a direct messaging feature that enables users to communicate privately with each other. This could include text messages, photos, videos, and other multimedia content.
4. *Notifications:* Implement a notification system to alert users about new likes, comments, follows, and other relevant activities. This keeps users engaged and informed about interactions on their posts and profiles.
5. *Search Functionality:* Add a robust search functionality that allows users to search for posts, users, hashtags, and locations. This makes it easier for users to find specific content and discover new connections.
6. *Analytics Dashboard:* Provide users with insights into their account activity, such as post performance metrics (likes, comments, shares), follower growth, and engagement trends. This empowers users to optimize their content strategy and track their progress over time.
7. *Accessibility Features:* Ensure the application is accessible to users with disabilities by implementing features such as screen reader support, keyboard navigation, and alternative text for images. This promotes inclusivity and expands the user base.
8. *Security Enhancements:* Strengthen the security measures to protect user data and prevent unauthorized access. This includes implementing encryption for sensitive information, enforcing strong password policies, and regularly auditing the system for vulnerabilities.

9. Content Moderation: Implement content moderation tools to detect and remove inappropriate or harmful content, such as spam, hate speech, and graphic imagery. This creates a safer and more welcoming environment for users.

10. Community Guidelines and Reporting: Provide clear community guidelines outlining acceptable behavior on the platform, and enable users to report violations of these guidelines. Implement a review process to handle reported content and enforce consequences for misconduct.

Benefits of Software Architecture in Developing the Threads Clone

Learning about software architecture can greatly benefit the implementation of a project, especially when developing a clone of an app like Thread.

Here's how:

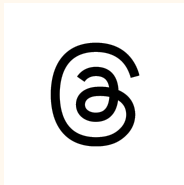
- 1. Scalability:* Understanding software architecture allows you to design the system in a way that it can handle growth and increased usage. For a Thread clone, this could mean architecting the system to accommodate a large user base and heavy traffic.
- 2. Modularity:* Software architecture encourages breaking down the system into smaller, modular components. This makes the codebase easier to manage, understand, and maintain. In the context of a Thread clone, modularity could involve separating features like messaging, user profiles, and groups into distinct modules.
- 3. Flexibility:* A good software architecture provides flexibility, allowing you to adapt and evolve the system over time. This is crucial for a project like a Thread clone, where new features may need to be added or existing ones modified based on user feedback and market trends.
- 4. Reliability:* A well-architected system is more likely to be reliable and robust. It can handle errors gracefully and ensure data integrity. This is essential for an app like Thread, where users rely on the platform for communication and sharing information.
- 5. Performance:* Software architecture influences the performance of the system. By designing for efficiency and optimization, you can ensure that the Thread clone performs well, providing users with a seamless experience.
- 6. Security:* Understanding software architecture enables you to build security measures into the system from the ground up. This includes implementing proper authentication, authorization, and data encryption to protect user privacy and sensitive information.

7. *Maintainability*: A clear architecture facilitates easier maintenance and updates to the system. This is particularly important for a long-term project like a Thread clone, where the codebase will evolve over time.

Overall, learning about software architecture equips you with the knowledge and skills needed to design and implement complex systems like a Thread clone effectively. It enables you to make informed decisions, anticipate challenges, and build a robust and scalable solution that meets the needs of users.

ReadMe.MD

Threads



Features

- Secure sign-up using email with email verification.
- Login with improved security using Json Web Tokens (JWT).
- Profile screen for user information and settings.
- Activity screen to discover and follow/unfollow other users.
- Create thread screen to post new content.
- Feed screen displaying threads by the user and others.

Installation

To run the Threads app on your device, follow these steps:

1. Clone this repository to your local machine:

```
2. git clone https://github.com/rohanudhwani/Threads.git
```

3. Navigate to the API directory:

```
4. cd Threads/api
```

5. Start the API server:

```
6. yarn start
```

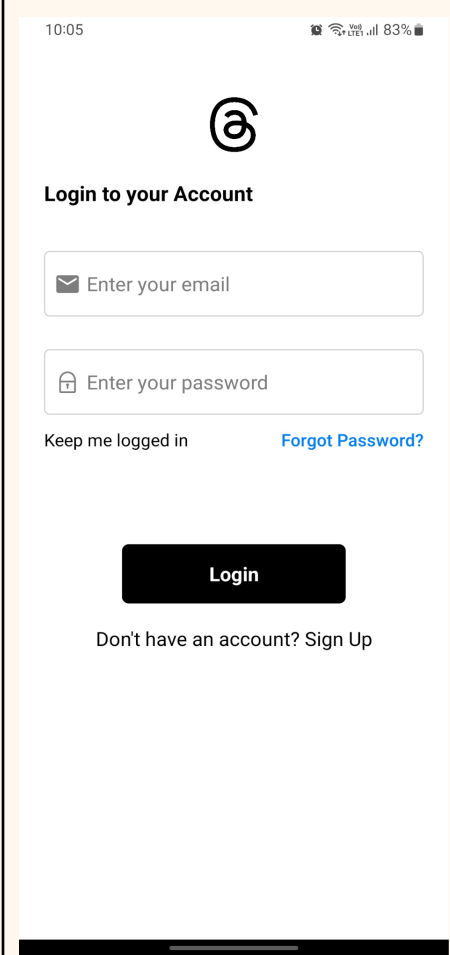
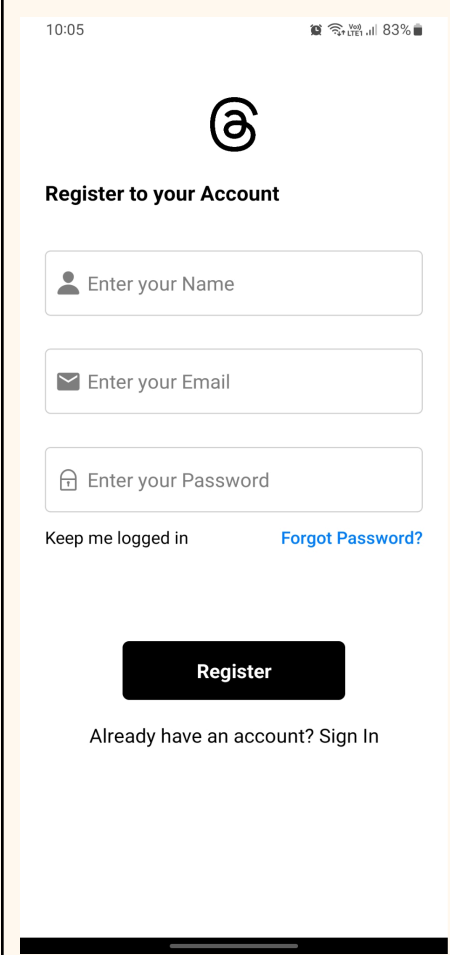
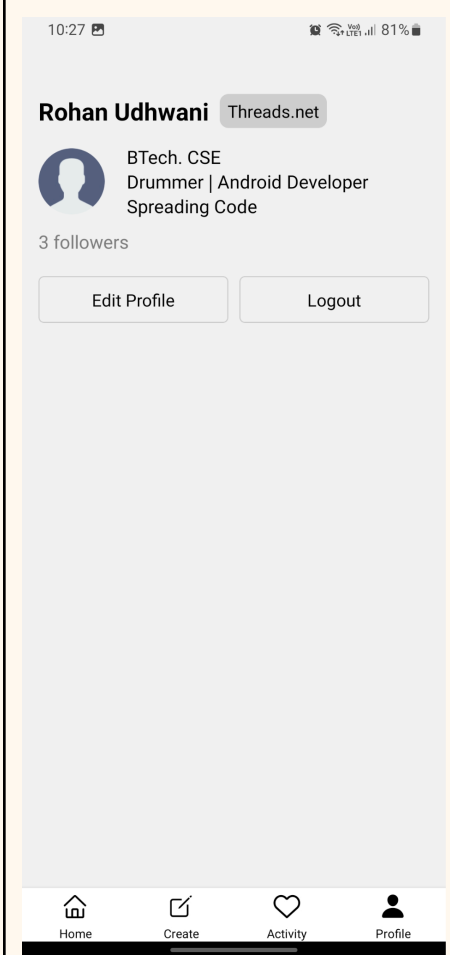
7. Obtain your PC's IP address using the command prompt:

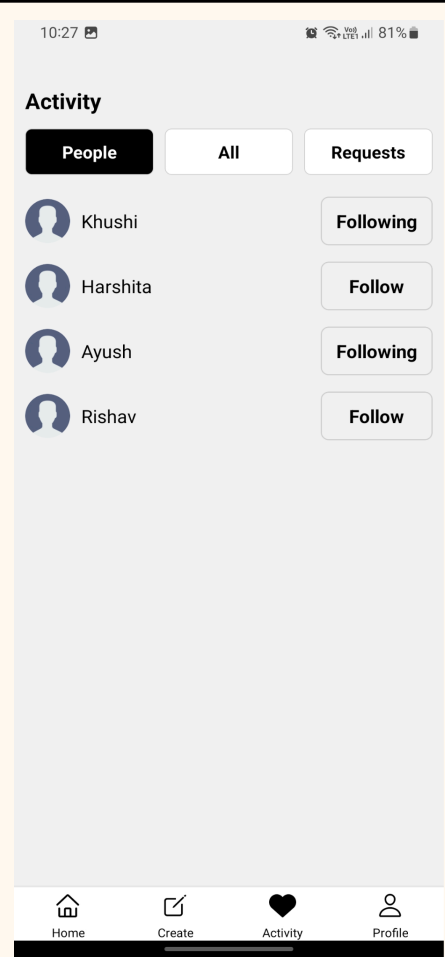
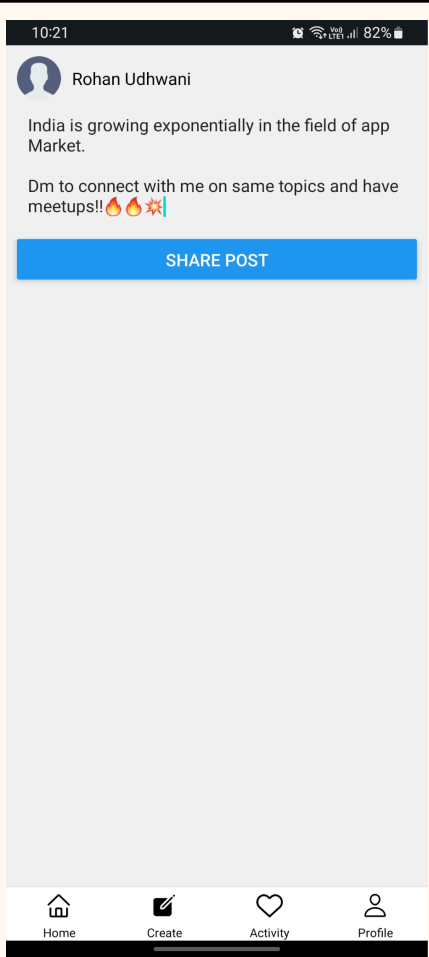
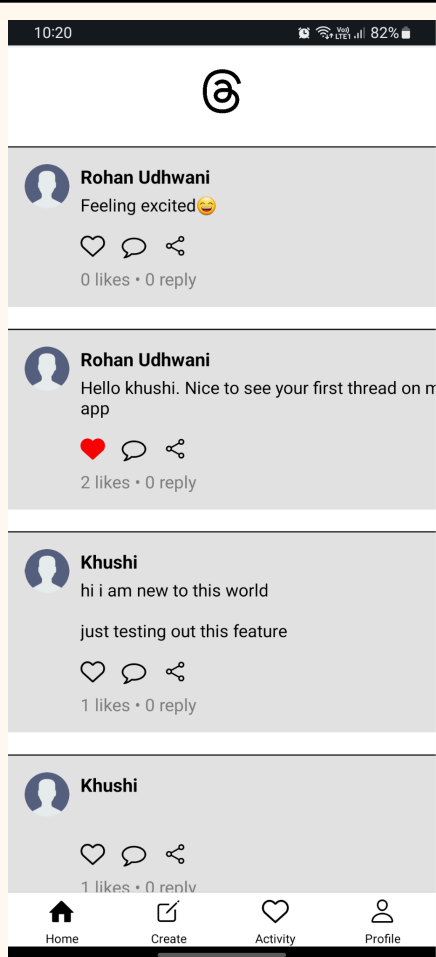
8. `ipconfig`

9. Set the necessary links in the app to your PC's IP address.

10. Run the Threads app on your device:
Download and install the app from [Threads.apk](#).

Screenshots

Login	SignUp	Profile
 <p>The screenshot shows the login screen of the Threads app. At the top, there is a status bar with the time 10:05 and battery level at 83%. Below the status bar is the Threads logo. The main heading is "Login to your Account". There are two input fields: "Enter your email" and "Enter your password". Below the input fields, there is a "Keep me logged in" checkbox and a "Forgot Password?" link. At the bottom, there is a black "Login" button and a link "Don't have an account? Sign Up".</p>	 <p>The screenshot shows the sign-up screen of the Threads app. At the top, there is a status bar with the time 10:05 and battery level at 83%. Below the status bar is the Threads logo. The main heading is "Register to your Account". There are three input fields: "Enter your Name", "Enter your Email", and "Enter your Password". Below the input fields, there is a "Keep me logged in" checkbox and a "Forgot Password?" link. At the bottom, there is a black "Register" button and a link "Already have an account? Sign In".</p>	 <p>The screenshot shows the user profile and settings screen of the Threads app. At the top, there is a status bar with the time 10:27 and battery level at 81%. Below the status bar is the user's profile information: "Rohan Udhwani" with a "Threads.net" badge, a profile picture, and bio "BTech. CSE", "Drummer Android Developer", and "Spreading Code". Below the bio, it says "3 followers". There are two buttons: "Edit Profile" and "Logout". At the bottom, there is a navigation bar with icons for "Home", "Create", "Activity", and "Profile".</p>
Login screen for authentication	Sign-up screen with email verification	User profile and settings screen

Activity	Create Thread	Feed
 <p>The Activity screen shows a list of users to follow/unfollow. At the top, there are tabs for 'People', 'All', and 'Requests'. Below, a list of users (Khushi, Harshita, Ayush, Rishav) is shown with their profile pictures and a 'Following' or 'Follow' button. The bottom navigation bar includes Home, Create, Activity (selected), and Profile.</p>	 <p>The Create Thread screen shows a form to create a new post. At the top, there's a header with the user's name (Rohan Udhwani) and a timestamp (10:21). Below, there's a text input area with a placeholder 'India is growing exponentially in the field of app Market.' and a 'SHARE POST' button. The bottom navigation bar includes Home, Create (selected), Activity, and Profile.</p>	 <p>The Feed screen displays a list of threads. At the top, there's a header with a search icon. Below, a list of threads is shown, each with a user's profile picture, name, and text. The threads are by Rohan Udhwani and Khushi. Each thread has a 'likes' and 'reply' count. The bottom navigation bar includes Home, Create, Activity (selected), and Profile.</p>
Activity screen to follow/unfollow users	Create Thread screen for posting content	Feed screen displaying user and other threads

Getting Started

1. Ensure you have Expo installed:
2. `npm install -g expo-cli`
3. Navigate to the Threads project directory:
4. `cd Threads`
5. Install dependencies:

6. `npm install`
7. Run the app:
8. `npx expo start`
9. Scan the QR code with the Expo Go app to launch Threads on your device.

Feedback and Contributions

Feel free to provide feedback or contribute to the Threads app by creating issues or pull requests in this repository. Your input is valuable, and we appreciate any contributions to make Threads even better!