Passenger Documentation

Project Overview

Project Name

Passenger

Description

Passenger is your go-to messaging app for real-time communication with friends, family, and colleagues.

Goals and Objectives

- **Enhance Communication**: Provide users with a seamless platform for instant communication, enabling them to connect with others in real time.
- **Deliver a User-Friendly Experience**: Design an intuitive interface that makes it easy for users of all ages and tech-savvies to navigate and utilize the app.

Target Audience

- **Young Adults and Students**: Individuals aged 18-40 who seek easy, instant communication for socializing, study groups, and collaborations.
- **Professionals and Remote Workers**: Users who require efficient communication tools for team collaboration and project management, especially in remote work environments.
- **Families**: Parents and relatives looking for a secure way to communicate and share updates with family members.

Key Features

- **Instant Messaging**: Fast text, and multimedia messaging for real-time conversations.
- **User-Friendly Interface**: An intuitive design that makes navigation easy for users of all ages.

Technical Documentation

Github

https://github.com/polanaeem1/Chat_App

Architecture Overview

1. **Client-Server Model**:
    - **Client-Side**: Built with React.js for user interaction.
    - **Server-Side**: Firebase handles authentication and real-time database services.

Key Components

Client-Side (React.js):

- **User Interface**: React components for managing messages, profiles, and chat rooms.
- **State Management**: Uses hooks or context API.
- **Real-Time Updates**: Listens for Firestore changes to update the UI dynamically.

Server-Side (Firebase):

- **Firebase Authentication**: Manages user sessions and identity verification.
- **Cloud Firestore**: A NoSQL database storing chat data in real-time.
- **Firebase Security Rules**: Protects user data.

Communication Flow

- **User Authentication**: Validates users with Firebase Authentication.
- **Real-Time Messaging**: Sends messages to Firestore, triggering updates for connected clients.
- **Data Storage**: Organizes messages, user profiles, and chat data in Firestore.

Database Schema

1. **Users Collection**

    Collection Name: `users`

    Document Structure:

    - `id`: (string, auto-generated by Firebase)
    - `finalUserName`: (string)

- `finalUserEmail`: (string)
- `profilePicture`: (string, URL to image)

2. **Chats Collection**

   Collection Name: `chats`

   Document Structure:

   - `createdAt`: (timestamp)
   - `senderId`: (string)
   - `text`: (string)

3. **User Chats Sub collection**

   Collection Name: `users/{userId}/chats`

   Document Structure:

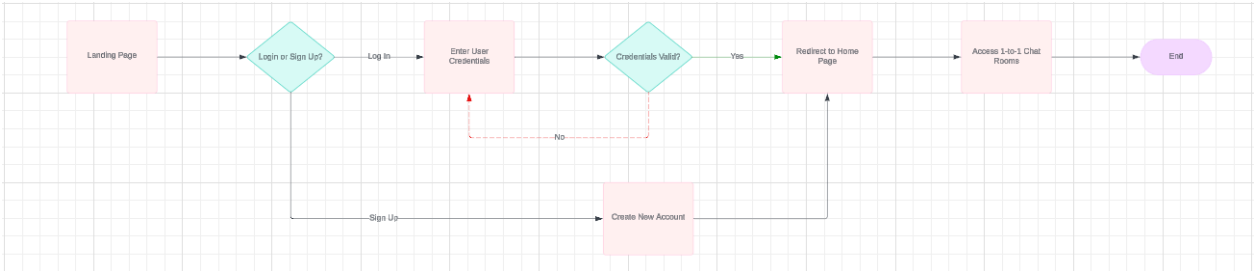
   - `chatId`: (string, reference to the chat)
   - `lastMessage`: (string)
   - `updatedAt`: (timestamp)
   - `receiverId`: (string)

Third-Party Integrations

- **Authentication Services**: Firebase Authentication for user login and security.
- **Real-Time Database**: Firebase Firestore for real-time data synchronization.

User Documentation

User Flow



Getting Started

1. Navigate to the Passenger website.
2. Sign Up / Log In

Registration:

- Click on the "Sign Up" button.
- Enter required information (email, username, password)

Login:

- If already registered, click on "Log In" and enter credentials.

Common Issues and Resolutions

1. **Login Problems**
    - **Issue**: Users may forget their password or enter incorrect credentials.
    - **Resolution**: Implement a "Forgot Password?" feature. Provide clear error messages.

Design Documentation

Brand Guidelines

Refer to the [Figma Design](https://www.figma.com/design/HhN0YC52z0DBiM1UVWPU75/DEPI-Project-(Testing)?node-id=0-1&node-type=canvas&m=dev) for brand colors, logos, and typography.

Design Principles

1. **Accessibility**
    - Color Contrast
    - Keyboard Navigation
2. **Responsiveness**
    - Fluid Layouts
    - Touch Targets

Testing Documentation

Testing Strategy

1. **Unit Testing**
    - Purpose: Test individual components/functions in isolation.
    - Focus Areas: Component rendering, form validation.

2. **Integration Testing**
    - Purpose: Test interactions between components.
    - Focus Areas: User flows, component interactions.

3. **End-to-End Testing**
    - Purpose: Test the application as a whole.
    - Focus Areas: User registration, message sending.

Test Cases

1. **Unit Testing**
    - Test Case: Component Rendering
    - Scenario: Check if the Message component renders correctly.

2. **Integration Testing**
    - Test Case: Login Flow
    - Scenario: User logs in successfully.

3. **End-to-End Testing**
    - Test Case: User Registration
    - Scenario: New user registers for an account.

Team Members:

Pola Naeem Roshdy:

- Form Validation
- Real time chatting functionality (Firebase)
- Website Responsiveness
- Main Chat Component (chat hero ,messages and send functionality)
- Test and optimize the code

Ahmed Mohamed Mahmoud:

- UI & UX
- Documentation
- Testing

Mina Mahrous Samoel:

- Sign Up Page
- Sidebar Component in the chat room page

Marvy Nady:

- Home Page
- Contact Details Component

Mario Mina Gaballah:

- User Authentication & Forget password Method

Samira Abdallah Mohamed:

- Sign In Page
- Chat Contacts Component