Project Documentation

# Project Planning & Management

## **1. Project Proposal**

### Overview

The React Frontend Chat App is designed to provide a seamless, real-time messaging experience with a modern and aesthetic user interface. It will be a responsive web application, ensuring accessibility across devices.

### Objectives

- Develop an intuitive and visually appealing chat interface.  
- Enable real-time messaging using WebSocket technology.  
- Implement authentication for secure user access.  
- Provide a scalable frontend architecture that integrates smoothly with backend APIs.

### Scope

***In-Scope***: UI/UX design, user authentication, message synchronization, notifications, and user settings.

***Out-of-Scope:*** Backend development, database management, AI chatbots.

## **2. Project Plan**

### Timeline (Gantt Chart)

A detailed Gantt chart should be created using tools like Microsoft Project, Excel, or online tools such as Trello or ClickUp.

### Milestones & Deliverables

- *Week 2*: Approved project proposal  
- *Week 4*: Finalized UI design  
- *Week 10*: Functional chat UI with authentication  
- *Week 12*: Fully tested chat app  
- *Week 13*: Deployment and user guide

### Resource Allocation

- *Frontend Developer* : UI/UX, component development, API integration  
- *Project Manager*: Planning & tracking progress  
- *QA Tester*: Bug identification and testing

## **3. Task Assignment & Roles**

|  |  |  |
| --- | --- | --- |
| ***Team Member*** | ***Role*** | ***Responsibilities*** |
| Tasneem Gomaa Anter Abdel Maged | React FrontEnd Developer | Use case diagram  Data flow diagram  Component Diagram |
| Shahd Gaber Fouad Ramadan | React FrontEnd Developer | State diagram  Sequence diagram  Deployment digram |
| Anasimon Raafat Ibrahim | React FrontEnd Developer | Project planning and management |
| Ali Mohamed Gomaa | React FrontEnd Developer | Requirements Gathering  Activity Diagram |
| Salma abdelsamei ahmed | React FrontEnd Developer | UI / UX Designs and Prototypes |
| Ibrahim Mohamed | React FrontEnd Developer | Use case description |

## **4. Risk Assessment & Mitigation Plan**

|  |  |  |
| --- | --- | --- |
| ***Risk*** | ***Impact*** | ***Mitigation*** ***Strategy*** |
| Delays in development | High | Set realistic deadlines, prioritize core features |
| API integration issues | Medium | Test API endpoints early, maintain clear documentation |
| Performance bottlenecks | High | Optimize React components, lazy loading |
| Security vulnerabilities | High | Implement authentication, sanitize inputs |

## **5. Key Performance Indicators (KPIs)**

|  |  |
| --- | --- |
| ***Metric*** | ***Target*** |
| Response time | Less than 200ms per interaction |
| System uptime | 99% availability |
| User adoption rate | 80% of users return after first use |

# Literature Review

* + **Feedback & Evaluation** – Lecturer’s assessment of the project.
  + **Suggested Improvements** – Areas where the project can be enhanced.
  + **Final Grading Criteria** – Breakdown of marks based on documentation, implementation, testing, and presentation.

**Functional Requirements for Chat Application**

**1. User Authentication and Account Management**

**1.1. User Registration**

* The system should allow new users to register using their phone number only.
* Users should verify their accounts via a one-time password (OTP) sent to their phone number.

**1.2. User Login**

* The system should allow users to log in using their phone number and OTP.
* Users should be able to log in on multiple devices by scanning a QR code generated by the primary device (e.g., mobile phone).

**1.3. Profile Management**

* Users should be able to update their profile information, including:
  + Profile picture
  + Display name
  + Status (e.g., "Available", "Busy", "Custom status")
* Users should be able to delete their account permanently if they choose to.

**2. Messaging Features**

**2.1. Text Messaging**

* Users should be able to send and receive text messages in real-time.
* Support for emoji, text formatting (e.g., bold, italics), and hyperlinks should be provided.

**2.2. Media Sharing**

* Users should be able to share images, videos, documents, and audio files.
* The system should compress large files to ensure fast delivery without compromising quality.

**2.3. Group Chats**

* Users should be able to create group chats with multiple participants.
* Group admins should have the ability to:
  + Add or remove participants
  + Change the group name and icon
  + Assign admin roles to other members

**2.4. Message Status**

* The system should display message status indicators such as:
  + Sent (message delivered to the server)
  + Delivered (message received by the recipient’s device)
  + Read (message viewed by the recipient)

**2.5. Message Deletion**

* Users should be able to delete messages for themselves or for everyone in the chat.

**3. Real-Time Communication**

**3.1. Push Notifications**

* The system should send push notifications to users when they receive new messages, even when the app is not open.

**3.2. Online Status**

* The system should display the online/offline status of users.
* Users should have the option to hide their status if desired.

**3.3. Typing Indicators**

* The system should show a typing indicator when a user is composing a message.

**4. Voice and Video Calls**

**4.1. Voice Calls**

* Users should be able to make voice calls to other users.
* The system should support call waiting, call holding, and call muting.

**4.2. Video Calls**

* Users should be able to make video calls with high-quality audio and video.
* The system should support group video calls with up to 8 participants.

**4.3. Call Logs**

* The system should maintain a log of missed, received, and dialed calls.

**5. Privacy and Security**

**5.1. End-to-End Encryption**

* All messages, calls, and shared media should be end-to-end encrypted to ensure privacy.

**5.2. Blocking Users**

* Users should be able to block other users to prevent them from sending messages or making calls.

**5.3. Privacy Settings**

* Users should be able to configure privacy settings, such as:
  + Who can see their last seen status
  + Who can see their profile picture
  + Who can add them to groups

**6. Additional Features**

**6.1. Search Functionality**

* Users should be able to search for messages, contacts, and media within the app.

**6.2. Starred Messages**

* Users should be able to star important messages for quick access later.

**6.3. Backup and Restore**

* Users should be able to back up their chat history to the cloud and restore it when needed.

**6.4. Dark Mode**

* The app should support a dark mode theme for reduced eye strain in low-light conditions.

**7. Admin and Moderation Features (For Group Admins)**

**7.1. Group Moderation**

* Admins should be able to mute participants, remove messages, or ban users from the group.
* Admins should be able to send announcements to all group members.

**Non-Functional Requirements for Chat Application**

**1. Performance**

* The chat application should deliver messages within 1 second under normal load conditions.
* The system should handle peak loads (e.g., during holidays) without performance degradation.

**2. Scalability**

* The application should support a growing number of concurrent users and active chats.
* It should support horizontal scaling to handle increased traffic.

**3. Reliability**

* The system should achieve 99.9% uptime.
* It should include automatic failover and recovery mechanisms.

**4. Security**

* All user data should be encrypted both in transit and at rest.
* The system should enforce strong authentication and authorization mechanisms.

**5. Usability**

* The user interface should be intuitive and easy to navigate.
* Accessibility features should be included for users with disabilities.

**6. Compatibility**

* The application should be compatible with smartphones, tablets, and desktops.
* It should support multiple languages.

**7. Maintainability**

* The system should use modular, clean code to facilitate easy updates.
* Comprehensive documentation should be provided.

**8. Performance Monitoring and Logging**

* The system should include real-time monitoring tools.
* Comprehensive logging should be implemented for troubleshooting.

**9. Backup and Disaster Recovery**

* Regular backups should be performed.
* A disaster recovery plan should ensure system restoration within a defined timeframe.

**Stakeholder Analysis for Chat Application**

**1. Key Stakeholders**

* **End Users:** Expect a seamless, reliable, and secure messaging experience.
* **Application Developers:** Need clear requirements, scalable architecture, and monitoring tools.
* **Business Owners/Product Managers:** Focus on market demand, user retention, and monetization.
* **System Administrators:** Require reliable infrastructure, monitoring tools, and security measures.
* **Regulatory Bodies:** Expect compliance with data protection laws and security standards.
* **Third-Party Integrations:** Require secure APIs and reliable uptime.

**2. Stakeholder Communication Plan**

* **End Users:** In-app notifications, emails (regular updates).
* **Developers:** Meetings, documentation (weekly).
* **Business Owners:** Reports, presentations (monthly).
* **System Administrators:** Alerts, dashboards (real-time).
* **Regulatory Bodies:** Compliance reports (annually).
* **Third-Party Integrations:** API documentation, support (as needed).

**3. Stakeholder Engagement Strategies**

* Conduct user surveys, provide customer support, and offer beta testing.
* Use Agile methodologies for developer collaboration.
* Share reports and involve business owners in product planning.
* Implement real-time monitoring for system administrators.
* Maintain compliance documentation for regulatory bodies.
* Provide API documentation for third-party integrations.

**User Stories**

**Epic 1: User Authentication**

**User Story 1: User Registration**

* As a new user, I want to register using my phone number so that I can create an account.

**User Story 2: User Login**

* As a user, I want to log in using my phone number and OTP so that I can access my account.

**User Story 3: Profile Management**

* As a user, I want to update my profile picture, display name, and status.

**User Story 4: Delete Account**

* As a user, I want to permanently delete my account.

**Epic 2: Messaging Features**

**User Story 5: Sending Messages**

* As a user, I want to send and receive text messages in real-time.

**User Story 6: Text Formatting and Emojis**

* As a user, I want to format my messages using bold, italics, and emojis.

**User Story 7: Media Sharing**

* As a user, I want to send images, videos, documents, and audio files.

**User Story 8: Group Chats**

* As a user, I want to create group chats.

**User Story 9: Message Deletion**

* As a user, I want to delete messages for myself or everyone.

# System Analysis & Design

**Use Case 1: Send Message**

|  |  |
| --- | --- |
| Actors | User |
| Preconditions | - The user must be logged into the system.  - The recipient must be an active user in the system |
| Main flow | |  |  | | --- | --- | | User Action | System Response | | 1. The user selects a contact to send a message to. |  | |  | |  | | --- | | The system displays the chat interface for the selected contact. |  |  | | --- | |  | | | 2. The user types a message in the text input field. |  | |  | |  | | --- | | The system validates the message and processes the request. |  |  | | --- | |  | | | 4. The system sends the message successfully. |  | |  | The recipient receives the message in their chat window. | |
| Aternative flows | - If the recipient is offline, the message is stored and delivered when they are online.  - If the message fails to send due to network issues, the system notifies the user and allows them to retry |
| Postconditions | - The message is stored in the system and displayed in both the sender’s and recipient’s chat history. |

**Use Case 2: Make Calls**

|  |  |
| --- | --- |
| Actors | User |
| Preconditions | - The user must be logged in.  - The recipient must be online and available for a call |
| Main flow | |  |  | | --- | --- | | User Action | System Response | | 1. The user selects a contact from their list. |  | |  | |  |  |  | | --- | --- | --- | | |  | | --- | | The system displays the contact’s profile with call options. |  |  | | --- | |  | |  |  | | --- | |  | | | |  | | --- | | 2. The user chooses between a voice or video call. |  |  | | --- | |  | |  | |  | |  |  |  | | --- | --- | --- | | |  | | --- | | The system prepares the selected call type and initiates the connection. |  |  | | --- | |  | |  |  | | --- | |  | | | |  | | --- | | 3. The user starts the call. |  |  | | --- | |  | |  | |  | |  | | --- | | The system attempts to connect the user with the recipient. |  |  | | --- | |  | | | |  | | --- | | 4. The recipient answers the call. |  |  | | --- | |  | |  | |  | |  | | --- | | The system establishes the call, enabling voice/video communication. |  |  | | --- | |  | | | |  | | --- | | 5. The user or recipient speaks and interacts during the call. |  |  | | --- | |  | |  | |  | |  | | --- | | The system transmits the audio and video data between both users. |  |  | | --- | |  | | | |  | | --- | | 6. The user or recipient ends the call. |  |  | | --- | |  | |  | |  | |  | | --- | | The system disconnects the call and updates the call log. |  |  | | --- | |  | | |
| Aternative flows | - If the recipient declines the call, the system notifies the caller.  - If the recipient is unavailable, the system provides an option to leave a message.  - If the network is unstable, the system may attempt to reconnect the call or notify the user of poor quality |
| Postconditions | - The call log is updated with the call details.  - The user can review past call logs |

* .

A diagram of a network

AI-generated content may be incorrect.

**Architecture Style: Single Page Application (SPA)**

* The system follows the **Single Page Application (SPA) architecture**, where the entire web application loads once and dynamically updates the UI without refreshing the page.
* **Frontend-Driven**: The application logic and UI updates are managed on the client-side.
* **Backend-less Approach**: Firebase services handle authentication, real-time database, and storage.

**System Components & Interactions**

Your SPA architecture consists of the following key components:

**1. Client-Side (Frontend - React.js)**

* **React Components:** Responsible for rendering UI dynamically based on user actions.
* **State Management:** Can use React Context API or Redux for managing application state.
* **Routing:** React Router is used for navigation between views without page reloads.
* **API Calls to Firebase:** The frontend interacts directly with Firebase for authentication, messaging, and storage.

**2. Backend (Firebase)**

* **Firebase Authentication:** Handles user registration, login, and OTP verification.
* **Cloud Firestore Database:** Stores messages, user profiles, and chat metadata.
* **Firebase Storage:** Manages media files like images, videos, and documents.
* **Cloud Messaging (FCM):** Sends push notifications when new messages arrive.

**3. External Services (Optional)**

* Third-party APIs for emoji support, language translation, or other features.

**Diagram Representation**

A screenshot of a computer

AI-generated content may be incorrect.

**Data Flow & System Behavior**

**DFD (Data Flow Diagram)**

A diagram of a program

AI-generated content may be incorrect.

* + **Sequence Diagrams**

A diagram of a phone

AI-generated content may be incorrect.

A diagram of a computer

AI-generated content may be incorrect.

**Activity Diagram**

* + A screenshot of a computer

    AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a diagram

AI-generated content may be incorrect.A diagram of a system

AI-generated content may be incorrect.

**State Diagram**

A diagram of a software process

AI-generated content may be incorrect.

**UI/UX Design & Prototyping**

<https://www.figma.com/design/NbT6BtkQEnQwGfdTCyp1oy/WhatsApp-Desktop-UI%3A-Dark-Mode-%26-Pixel-Perfect-Recreation-(Community)?node-id=0-1&t=BgxwPUUHyG1dYIlD-1>

**System Deployment & Integration**

* + Technology Stack

**Technology Stack for Backend-less SPA**

**Frontend (React SPA)**

* **React.js** – Core framework for building the SPA.
* **React Router** – For client-side routing (no full-page reload).
* **Context API / Redux** – For managing application state.
* **LocalStorage / IndexedDB** – To persist data locally.
* **TailwindCSS / Material UI** – For styling and UI components.
* **PWA (Progressive Web App)** – If offline support is needed.

**Data Handling (No Backend)**

* **Static JSON Files** – Store data in /public/data.json instead of a database.
* **LocalStorage / IndexedDB** – Save user preferences or temporary data.
* **Third-Party APIs** – Use public APIs like https://jsonplaceholder.typicode.com/ for dynamic data.

**Hosting & Deployment**

* **Vercel / Netlify / GitHub Pages** – Free hosting for frontend applications.

**Deployment Diagram**

A diagram of a software application

AI-generated content may be incorrect.

**Component Diagram**

A diagram of a system

AI-generated content may be incorrect.

.