FACILE

Introduction:

- Overview of the messaging app.
- Purpose and goals of the application.
- Brief explanation of the MERN stack and its components.

Architecture:

- High-level architecture diagram showcasing the components and their interactions.
- Description of the frontend and backend components.
- Explanation of how WebSockets, WebRTC, Face API, Redis, and other technologies are integrated.

Features and Functionality:

- Detailed explanation of each feature, such as channels, event channels, direct messages, voice calls, video calls, file sharing, and threaded conversations.
- Description of the admin role and its abilities to add or remove members and manage conversations.
- How WebSockets and real-time updates enhance the user experience.
- How WebRTC is used for voice and video calls.
- Integration of Face API and its functionality.
- How file sharing is implemented.
- Frontend automation using Selenium.

Technology Stack:

- Explanation of the MERN stack and its advantages for building the messaging app.
- Detailed information about each component and its purpose (MongoDB, Express.js, React, Node.js).
- How Redis is used for caching and improving performance.
- How Postman is utilized for API testing.
- Logging using both MongoDB and PostgreSQL for different aspects of the application.

Installation and Setup:

- Step-by-step instructions for setting up the development environment.
- Guide for installing and configuring the required dependencies and tools.
- Instructions for configuring the database (MongoDB, PostgreSQL) and other external services (Redis).

API Documentation:

- Detailed documentation for the API endpoints, including the request/response format, parameters, and examples.
- Explanation of authentication mechanisms and security considerations.
- API routes and their purposes.

User Guide:

- Instructions for user registration and login.
- Walkthrough of the app's interface and features.
- How to create workspaces, channels, and event channels.
- How to join and leave channels.
- How to send direct messages, initiate voice and video calls, and share files.
- Explanation of threaded conversations and their usage.

Testing and Quality Assurance:

- Overview of the testing strategy, including unit tests, integration tests, and end-to-end tests.
- Guide for running tests and ensuring code quality.
- Description of the frontend automation tests using Selenium.

Deployment and Scalability:

- Guide for deploying the application to production environments.
- Considerations for scaling the application, such as load balancing and horizontal scaling.
- Recommendations for optimizing performance and handling high traffic.

User Stories:

1. User Registration:

- Required information: Username, email address, password.

- User story: As a new user, I want to register an account to access the messaging app.

2. User Login:

- Required information: Username/email and password.

- User story: As a registered user, I want to log in to the messaging app to start using its features.

3. Create a Channel:

- Required information: Channel name, optional channel description.
- User story: As a user, I want to create a new channel to facilitate communication with a specific group of people.

4. Join a Channel:

- Required information: Channel ID or invitation link.
- User story: As a user, I want to join an existing channel to participate in the discussions within that channel.

5. Leave a Channel:

- Required information: Channel ID.
- User story: As a user, I want to leave a channel to discontinue my participation in its discussions.

6. Add Members to a Channel:

- Required information: Channel ID, usernames/emails of members to add.
- User story: As an admin or channel owner, I want to add new members to a channel.

7. Remove Members from a Channel:

- Required information: Channel ID, usernames/emails of members to remove.
- User story: As an admin or channel owner, I want to remove specific members from a channel.

8. Send Direct Message:

- Required information: Recipient's username or email, message content.
- User story: As a user, I want to send a direct message to another user for one-on-one communication.

9. Start a Threaded Conversation:

- Required information: Channel ID or direct message ID, message content, optional parent message ID.
- User story: As a user, I want to start a threaded conversation within a channel or direct message to discuss a specific topic.

10. Reply to a Thread:

- Required information: Channel ID or direct message ID, parent message ID, reply message content.

- User story: As a user, I want to reply to an existing threaded conversation to provide my input or ask questions.

11. Send Text Messages:

- Required information: Channel ID or direct message ID, message content.

- User story: As a user, I want to send text messages within a channel or direct message to communicate with other members.

12. Initiate Voice Call:

- Required information: Channel ID or direct message ID, selected members to include in the call.

- User story: As a user, I want to initiate a voice call within a channel or direct message to have real-time audio conversations.

13. Initiate Video Call:

- Required information: Channel ID or direct message ID, selected members to include in the call.

- User story: As a user, I want to initiate a video call within a channel or direct message to have real-time video conversations.

14. Send Files:

- Required information: Channel ID or direct message ID, selected file(s) to upload and share.

- User story: As a user, I want to send files within a channel or direct message to share documents, images, or other media.

15. Create an Event Channel:

- Required information: Event channel name, event details, event date/time.

- User story: As a user, I want to create an event channel to facilitate communication and coordination for a specific event.

16. Join an Event Channel:

- Required information: Event channel ID or invitation link.

- User story: As a user, I want to join an existing event channel to participate in the event discussions and updates.

17. Leave an Event Channel:

- Required information: Event channel ID.

- User story: As a user, I want to leave an event channel to discontinue my participation in the event discussions.

18. Assign Admin Role:

- Required information: User's username or email, channel or event channel ID.

- User story: As a super admin or channel/event owner, I want to assign an admin role to a user to grant them administrative privileges.

19. Revoke Admin Role:

- Required information: User's username or email, channel or event channel ID.

- User story: As a super admin or channel/event owner, I want to revoke the admin role from a user to remove their administrative privileges.

Requirements:

User Registration and Authentication:

- User registration form (React, Node.js, MongoDB)
- User login form (React, Node.js, MongoDB)
- User authentication and session management (Node.js, Express.js, MongoDB)

User Dashboard:

- Dashboard layout (React, HTML, CSS)
- Displaying channels, event channels, direct messages, and notifications (React, Node.js)
- Navigation menus and links (React, HTML)

Create a Channel:

- Channel creation form (React, Node.js, MongoDB)
- Channel model and schema (MongoDB)
- API endpoint for creating and managing channels (Node.js, Express.js)

Join a Channel:

- Join channel functionality (React, Node.js, MongoDB)
- API endpoint for joining channels and retrieving channel data (Node.js, Express.js)

Leave a Channel:

- Leave channel functionality (React, Node.js, MongoDB)
- API endpoint for leaving channels (Node.js, Express.js)

Add Members to a Channel:

- Member addition form (React, Node.js, MongoDB)
- API endpoint for adding and removing members from channels (Node.js, Express.js)

Remove Members from a Channel:

- Member removal functionality (React, Node.js, MongoDB)
- API endpoint for removing members from channels (Node.js, Express.js)

Send Direct Message:

- Direct message functionality (React, Node.js, MongoDB)
- Direct message model and schema (MongoDB)
- API endpoint for sending and retrieving direct messages (Node.js, Express.js)

Start a Threaded Conversation:

- Threaded conversation creation form (React, Node.js, MongoDB)
- Threaded conversation model and schema (MongoDB)
- API endpoint for starting and retrieving threaded conversations (Node.js, Express.js)

Reply to a Thread:

- Reply form for threaded conversations (React, Node.js, MongoDB)
- API endpoint for replying to threaded conversations (Node.js, Express.js)

Send Text Messages:

- Text messaging functionality (React, Node.js, MongoDB)
- API endpoint for sending and retrieving text messages (Node.js, Express.js)

Initiate Voice Call:

- WebRTC implementation for voice calls (React, Node.js)
- API endpoint for handling voice calls (Node.js, Express.js)

Initiate Video Call:

- WebRTC implementation for video calls (React, Node.js)
- API endpoint for handling video calls (Node.js, Express.js)

Send Files:

- File upload functionality (React, Node.js, MongoDB)
- API endpoint for uploading and sharing files (Node.js, Express.js)

Frontend Automation with Selenium:

- Selenium integration for frontend automation testing (JavaScript)

WebSocket Integration:

- WebSocket implementation for real-time updates (Node.js, Express.js)
- API endpoints for handling WebSocket connections and messages (Node.js, Express.js)

Face API Integration:

- Integration with Face API for facial recognition or other facial-related functionalities (React, Node.js)

Backend with Redis:

- Redis integration for caching frequently accessed data (Node.js, Express.js)

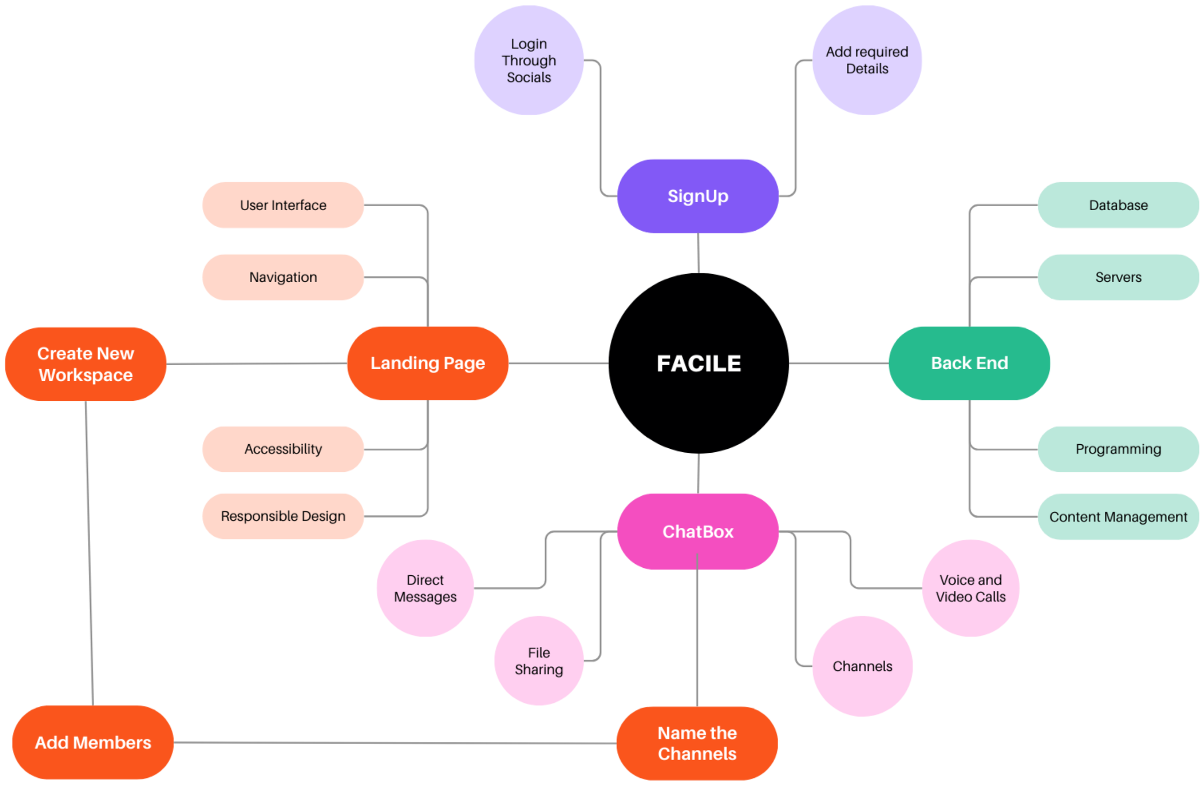
Postman for API Testing:

- Use Postman for testing and validating API endpoints (Node.js, Express.js)

Logging with MongoDB and PostgreSQL:

- Store application logs in both MongoDB and PostgreSQL for auditing and analysis purposes

Flow Chart:



Test Cases:

User Registration and Login:

- Verify that a user can register with valid credentials.
- Verify that a user cannot register with invalid or duplicate credentials.
- Verify that a registered user can log in successfully.
- Verify that login fails with incorrect credentials.

Workspace and Channel Management:

- Verify that a user can create a new workspace.
- Verify that a user can join an existing workspace.
- Verify that a user can create a new channel within a workspace.
- Verify that a user can join an existing channel.
- Verify that an admin can add members to a channel.
- Verify that an admin can remove members from a channel.

Direct Messages:

- Verify that a user can send a direct message to another user.
- Verify that the recipient receives the direct message.
- Verify that the recipient can reply to the direct message.

Threaded Conversations:

- Verify that a user can start a threaded conversation within a channel or direct message.
- Verify that users can reply to a threaded conversation.
- Verify that replies are correctly nested and displayed in the thread.

Voice and Video Calls:

- Verify that a user can initiate a voice call within a channel or direct message.
- Verify that other participants receive the call request.
- Verify that participants can accept or decline the call.
- Verify that voice calls can be established successfully.
- Verify that a user can initiate a video call within a channel or direct message.
- Verify that video calls can be established successfully.

File Sharing:

- Verify that a user can upload and share a file within a channel or direct message.
- Verify that other participants can view and download the shared file.

Face API Integration:

- Verify that the Face API integration functions correctly for facial recognition or analysis features.
- Verify that user authentication using the Face API works as expected.

Frontend Automation with Selenium:

- Verify that Selenium performs automated tests on the frontend user interface.
- Verify that simulated user actions are executed and verified correctly.

Real-time Updates with WebSocket:

- Verify that real-time updates are received correctly by clients.
- Verify that messages, calls, and other activities are updated in real-time.

Backend with Redis:

- Verify that Redis caching improves performance and data retrieval times.

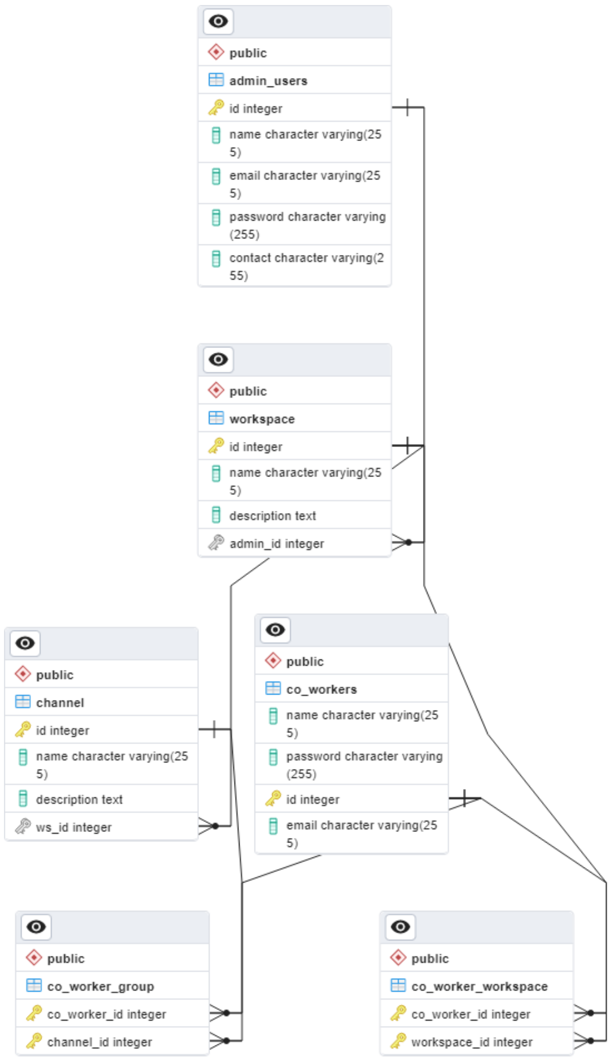
Postman for API Testing:

- Verify that API endpoints function correctly by testing with Postman.

Logging with MongoDB and PostgreSQL:

- Verify that logs are correctly stored in MongoDB and PostgreSQL for auditing and analysis purposes.

ERD Diagram:


public
admin_users
id integer
name character varying(255)
email character varying(255)
password character varying (255)
contact character varying(255)
public
workspace
id integer
name character varying(255)
description text
admin_id integer
public
channel
id integer
name character varying(255)
description text
ws_id integer
public
co_workers
name character varying(255)
password character varying (255)
id integer
email character varying(255)
public
co_worker_group
co_worker_id integer
channel_id integer
public
co_worker_workspace
co_worker_id integer
workspace_id integer