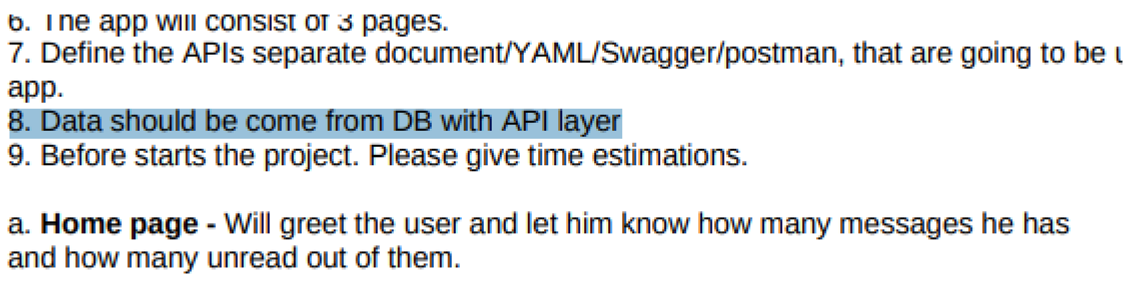
AUTHOR: HILARIO JUNIOR NENGARE

TITLE: MAIL INBOX APP DOCUMENTATION

SERVER:

1. Database Configuration (db.js)

- The file serves as the configuration for the SQLite database using Sequelize in the Node.js application.
- Advantages of using SQLite are it's lightweight, lite and needs no complex configurations and sequelize ORM is a good mediator between NodeJS and SQL queries, making our database queries much more readable.
- I define two particular models User and Message, and then establish a one-to-many association between them as the **predefined user will have many messages** but all the messages will be the user's.
- The syncDatabase function ensures the synchronization of models with the database.
- This module encapsulates number 8 of the requirements specification which states that

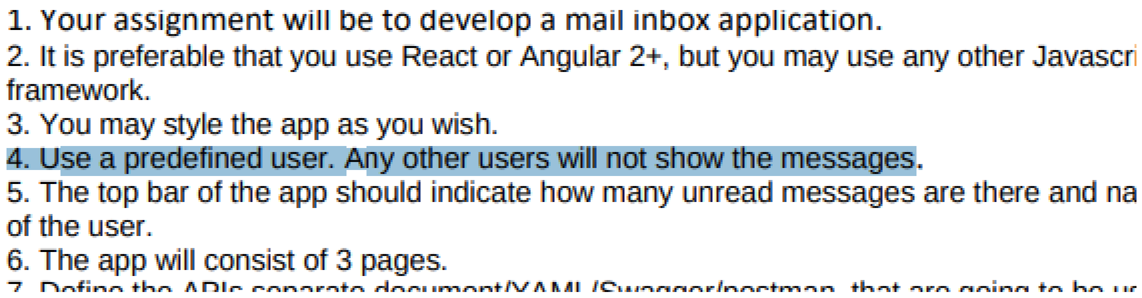


6. The app will consist of 3 pages.
7. Define the APIs separate document/YAML/Swagger/postman, that are going to be u
app.
8. Data should be come from DB with API layer
9. Before starts the project. Please give time estimations.

a. **Home page -** Will greet the user and let him know how many messages he has
and how many unread out of them.

2. Dummy Data Initialization (dummyData.js):

- The dummyData.js file is responsible for populating the SQLite database with predefined data.
- It utilizes the Sequelize models and the syncDatabase function from db.js to ensure data consistency.
- I define a preDefinedUserData function to creates a predefined User with name Jim and insert multiple messages into the database.
- This file provides a realistic datset which facilitates a predefined user JIM as per requirement number 4



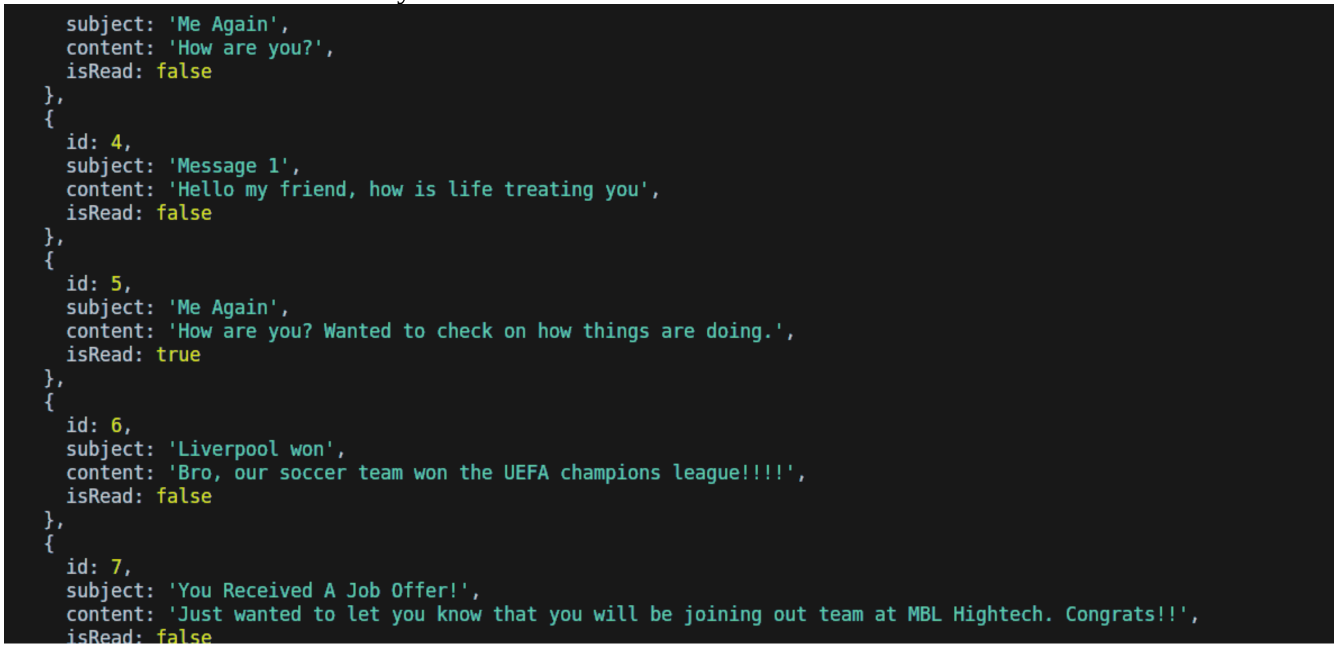
1. Your assignment will be to develop a mail inbox application.
2. It is preferable that you use React or Angular 2+, but you may use any other Javascri
framework.
3. You may style the app as you wish.
4. Use a predefined user. Any other users will not show the messages.
5. The top bar of the app should indicate how many unread messages are there and na
of the user.
6. The app will consist of 3 pages.
7. Define the APIs separate document/YAML/Swagger/postman, that are going to be u

3. **NodeJS Express API Implementation (server.js):**

- This file defines the Express application, serving as the main entry point for the Node.js server.
- I configure routes to handle requests related to user messages, including retrieving messages for a user and fetching a specific message by ID.
- I incorporate a logging statement to track the jsonified response:

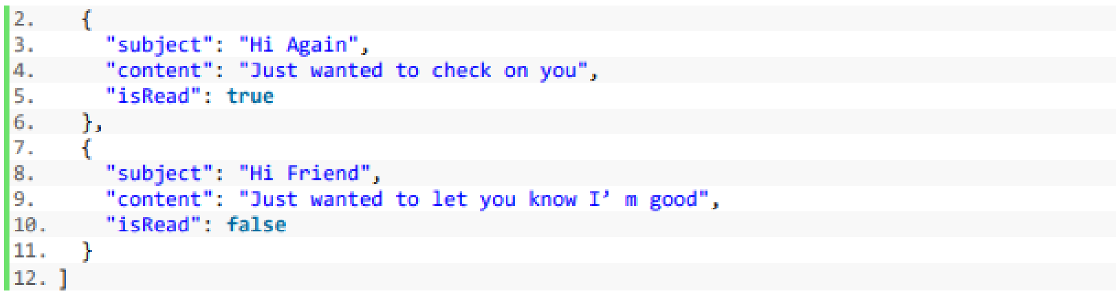
```
console.log('these are user messages\n', userMessages.toJSON());
```

which in return yields this to the console



```
    subject: 'Me Again',
    content: 'How are you?',
    isRead: false
  },
  {
    id: 4,
    subject: 'Message 1',
    content: 'Hello my friend, how is life treating you',
    isRead: false
  },
  {
    id: 5,
    subject: 'Me Again',
    content: 'How are you? Wanted to check on how things are doing.',
    isRead: true
  },
  {
    id: 6,
    subject: 'Liverpool won',
    content: 'Bro, our soccer team won the UEFA champions league!!!!',
    isRead: false
  },
  {
    id: 7,
    subject: 'You Received A Job Offer!',
    content: 'Just wanted to let you know that you will be joining out team at MBL Hightech. Congrats!!',
    isRead: false
```

- This response’s schema matches the one specified in the requirements,

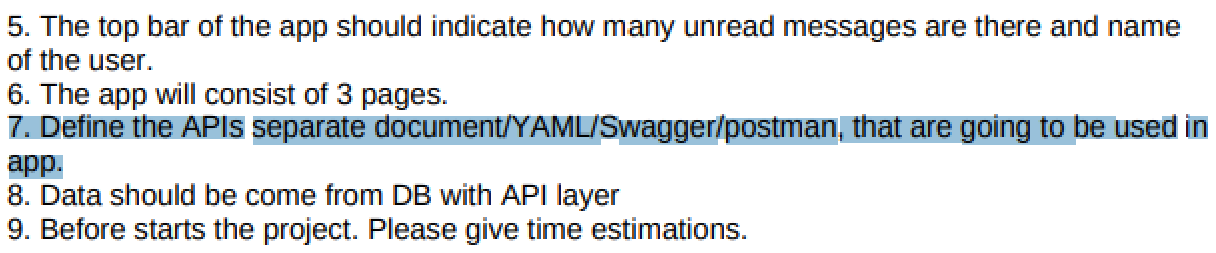


```
  {
    "subject": "Hi Again",
    "content": "Just wanted to check on you",
    "isRead": true
  },
  {
    "subject": "Hi Friend",
    "content": "Just wanted to let you know I’ m good",
    "isRead": false
  }
]
```

- Also, the file serves static React files and implements a fallback mechanism to redirect unspecified routes to the React application.

4. **API Specification (swagger.yml):**

- The swagger.yml file is an OpenAPI Specification 3.0 document, providing a detailed description of the Mail Inbox API.
- I define two main API paths: one for retrieving messages for a user and another for retrieving a specific message by ID.
- I also include data models; User and Message under the components section to maintain consistency throughout the API definition.
- This corresponds to the requirement Number 7 of the Requirements Specification stating to

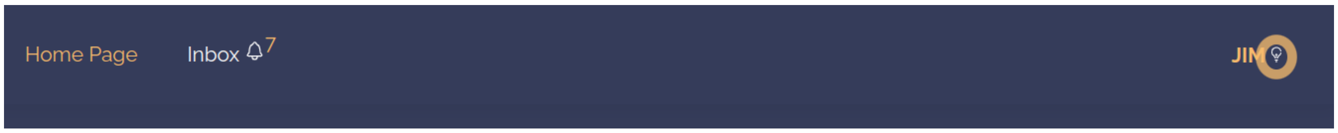

5. The top bar of the app should indicate how many unread messages are there and name of the user.
6. The app will consist of 3 pages.
7. Define the APIs separate document/YAML/Swagger/postman, that are going to be used in app.
8. Data should be come from DB with API layer
9. Before starts the project. Please give time estimations.

CLIENT

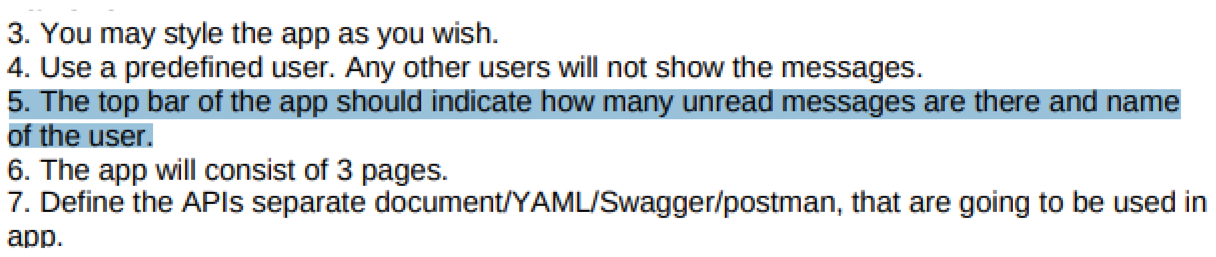
1. **Navigation Component:**

- I use a **Scroll Effect on the header**
  - by tracking the window scrollY position.
  - then updating the isHeaderActive state based on the scroll position.

- The **User Data Fetching**
  - Fetches user data, including unread messages, from the server using the api endpoint I specified.
  - Updates userName and unreadMessages states based on the fetched data.
  - Provides real-time user information in the navigation bar.
  - **Improvements** I could enforce would be using websocket.io to make the navbar notifications badge real-timely!!

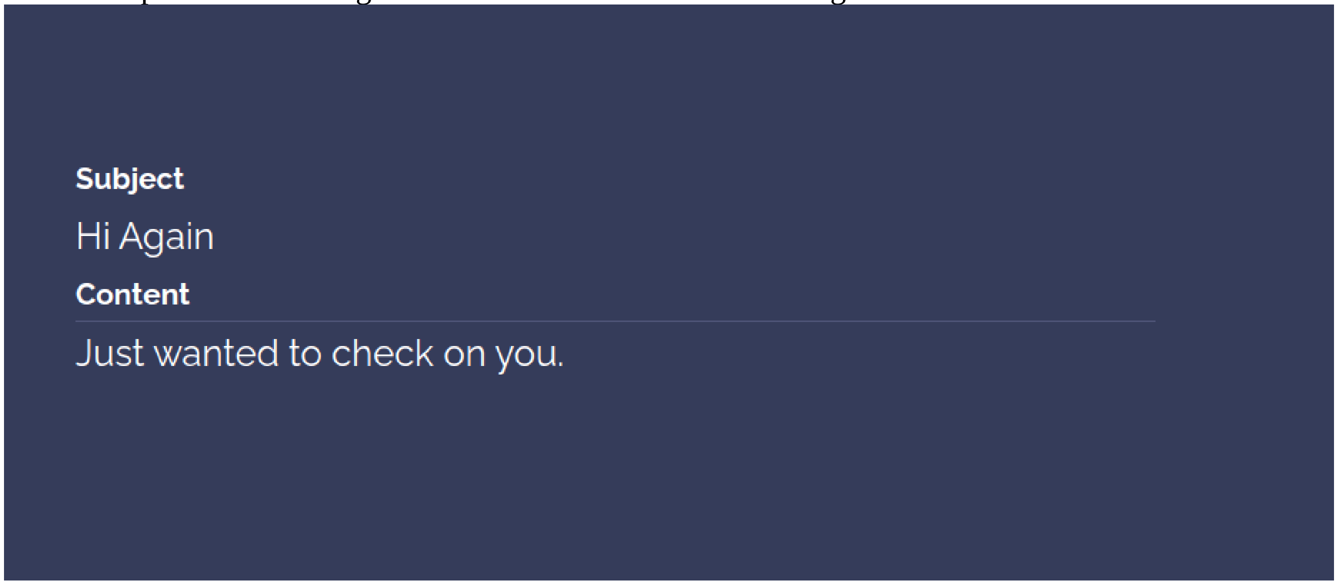


- This facilitates number require number 5


3. You may style the app as you wish.
4. Use a predefined user. Any other users will not show the messages.
5. The top bar of the app should indicate how many unread messages are there and name of the user.
6. The app will consist of 3 pages.
7. Define the APIs separate document/YAML/Swagger/postman, that are going to be used in app.

2. **Message Page Component:**

- **Message Fetching**
  - Utilizes useParams hook to extract messageId from the URL.
  - Fetches the specific message data corresponding to the messageId.
  - Updates the message state to render the retrieved messages



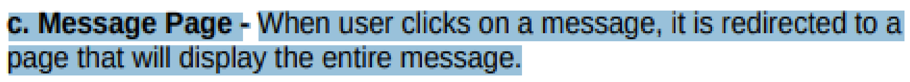
Subject

Hi Again

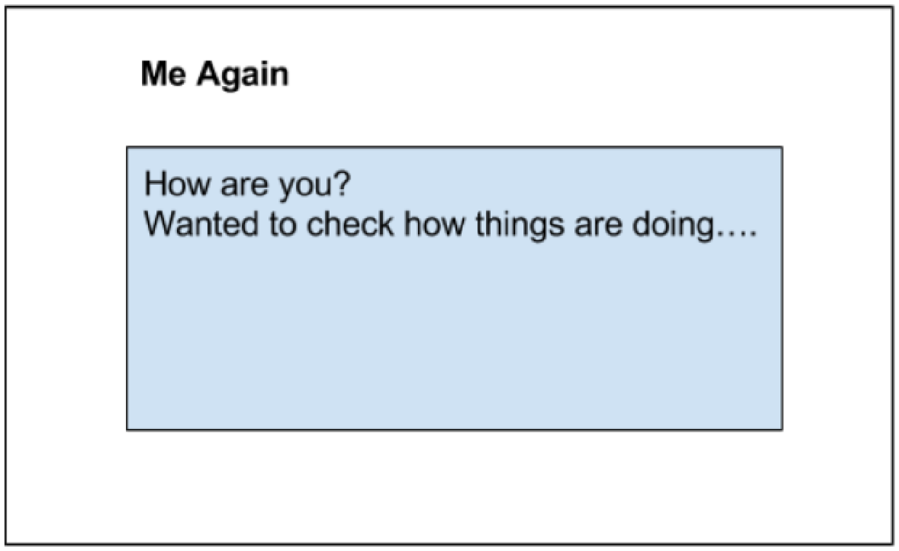
Content

Just wanted to check on you.

-This meets the requirement 9 c stating that



c. Message Page - When user clicks on a message, it is redirected to a page that will display the entire message.

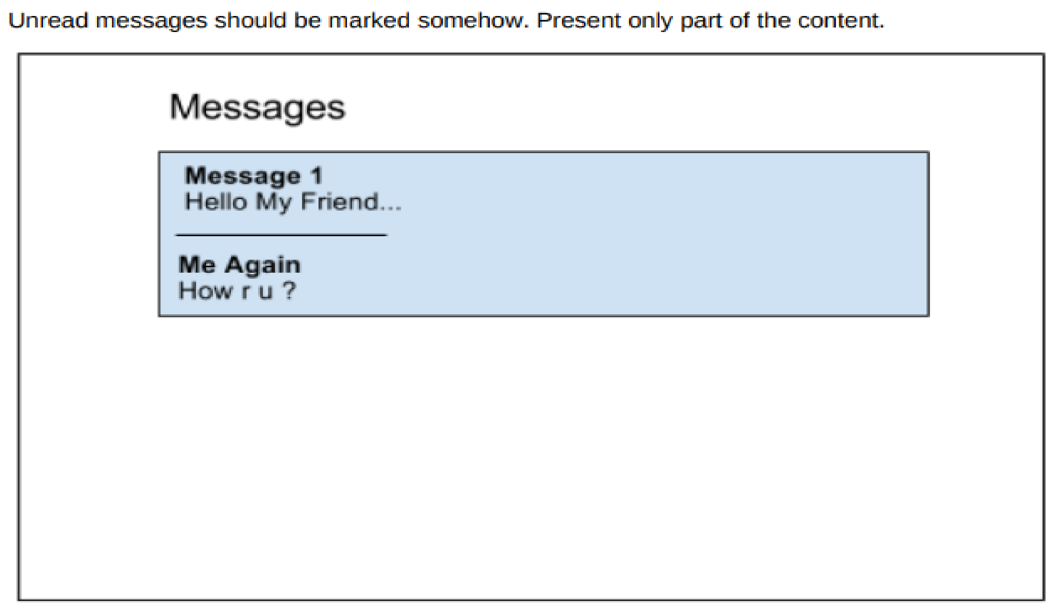


Me Again

How are you?
Wanted to check how things are doing….

3. **Inbox Page Component:**

- **User Inbox Data Fetching :**
  - Fetches the user's inbox messages from the server.
  - Updates the messages state with the fetched data.
  - Facilitates dynamic rendering of messages in the inbox by mapping the retrieved data.messages.
  - Requirement **9 b** states that

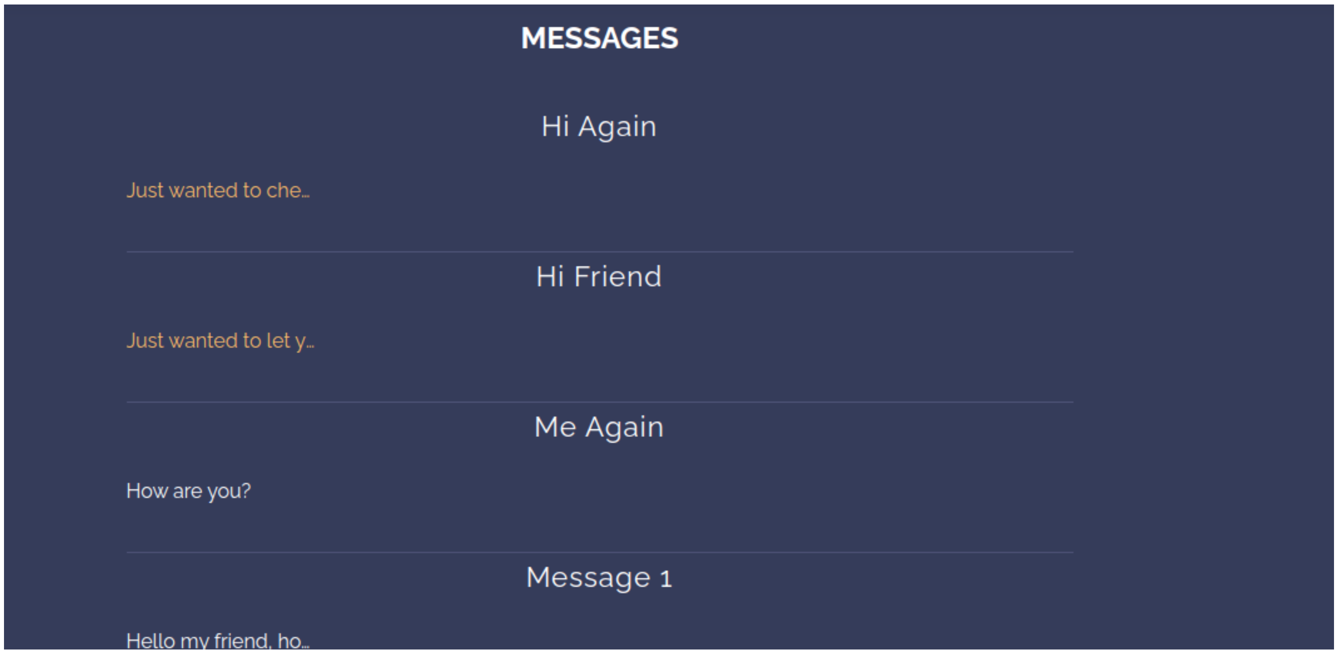


Unread messages should be marked somehow. Present only part of the content.

- I use css text-overflow property to present part of the content like so

```
.a .p {
color: var(--white);
overflow: hidden;
white-space: nowrap;
text-overflow: ellipsis;
max-width: 155px;
}
```

- Unread messages are marked as white in color and read messages are golden in color, like so


MESSAGES

Hi Again

Just wanted to che...

Hi Friend

Just wanted to let y...

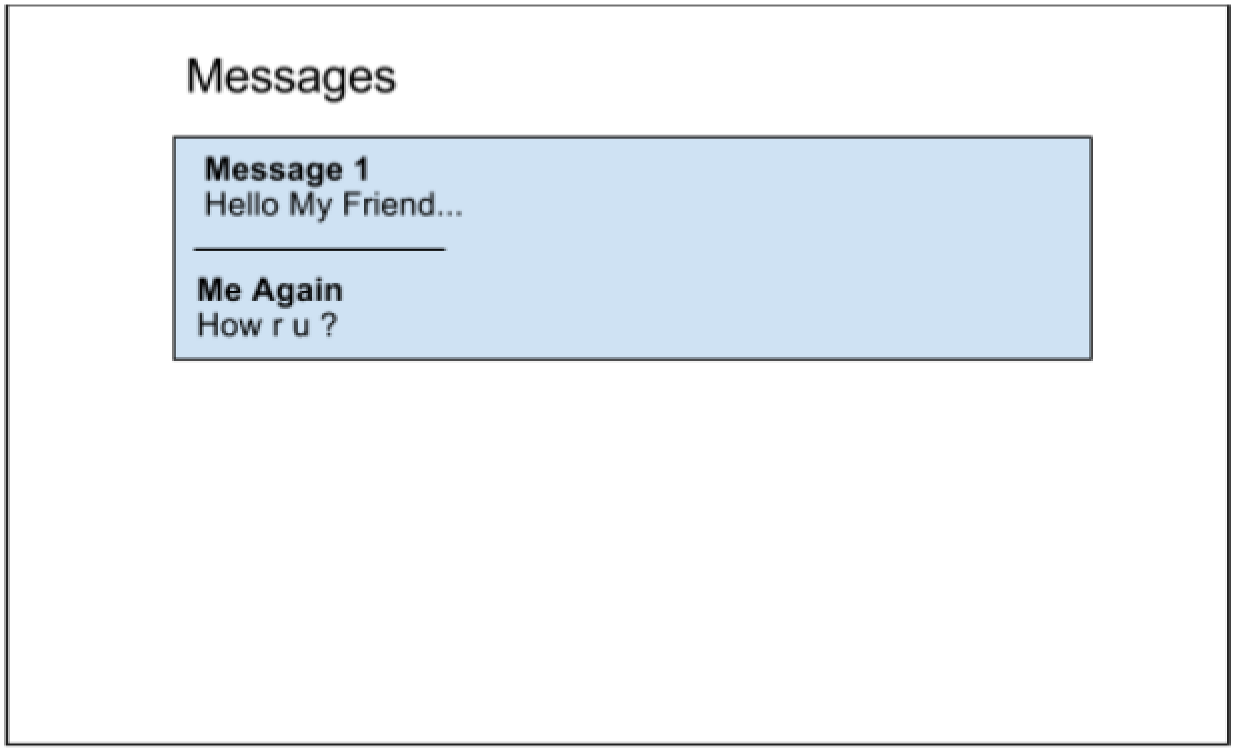
Me Again

How are you?

Message 1

Hello my friend, ho...

- This corresponds to the inbox page layout specified here

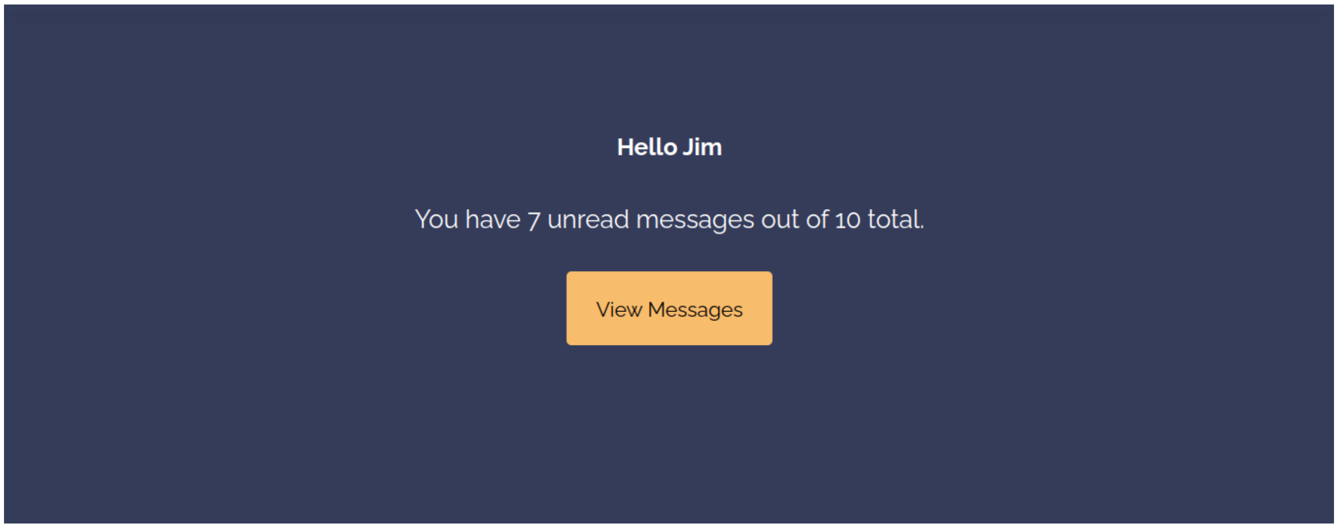

Messages

Message 1
Hello My Friend...

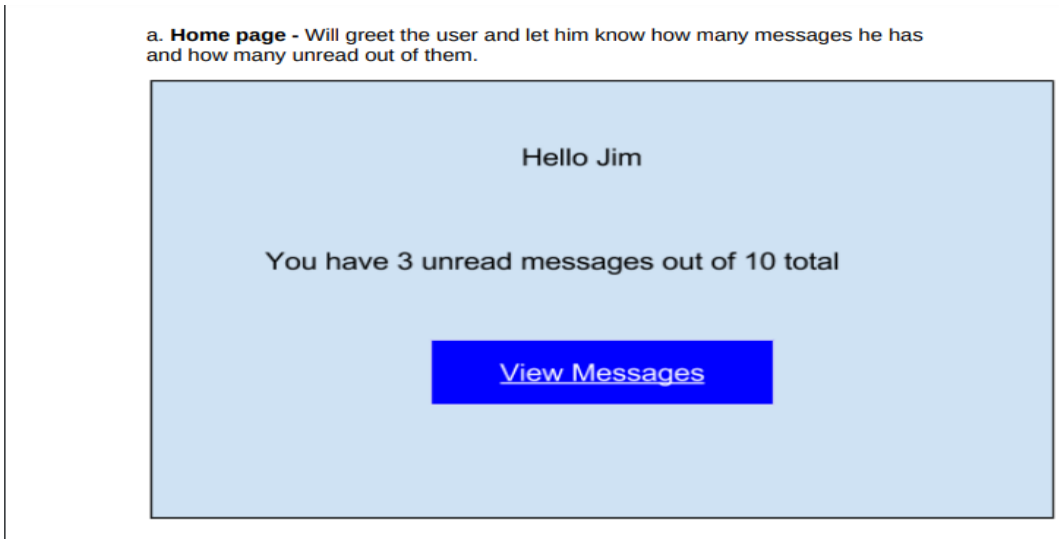
Me Again
How r u ?

4. Home Page Component:

- **User Data Fetching:**
  - Fetches user data, including total and unread message counts.
  - Updates userName, unreadMessages, and messages states.
  - Provides personalized information and encourages interaction with the messaging system.



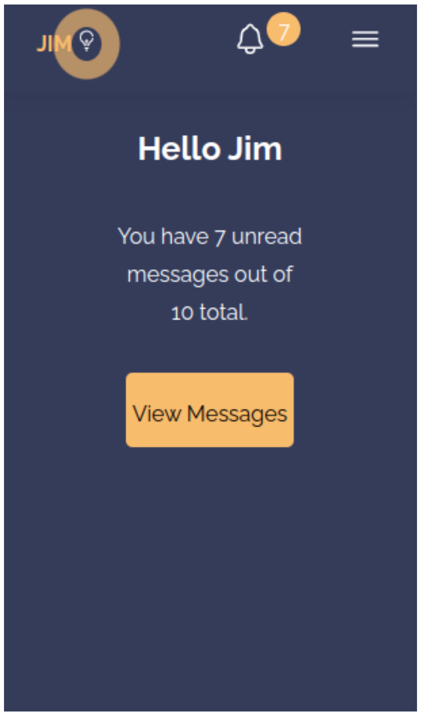
- The homepage component inherits the layout specified in requirement Number **9 a** stating that



a. **Home page -** Will greet the user and let him know how many messages he has and how many unread out of them.

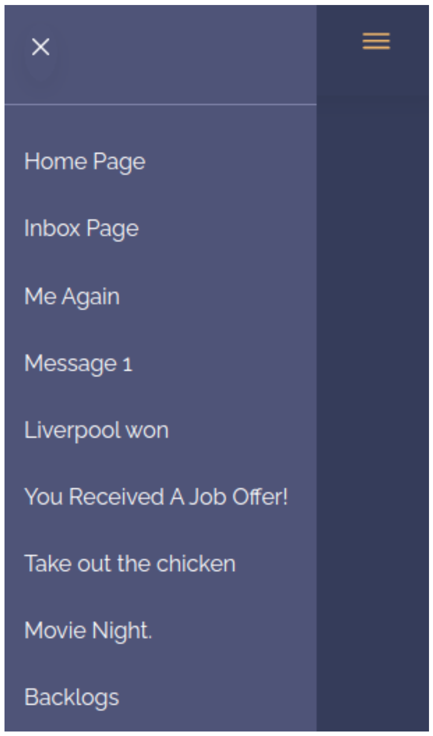
RESPONSIVE LAYOUTs

For Homepage

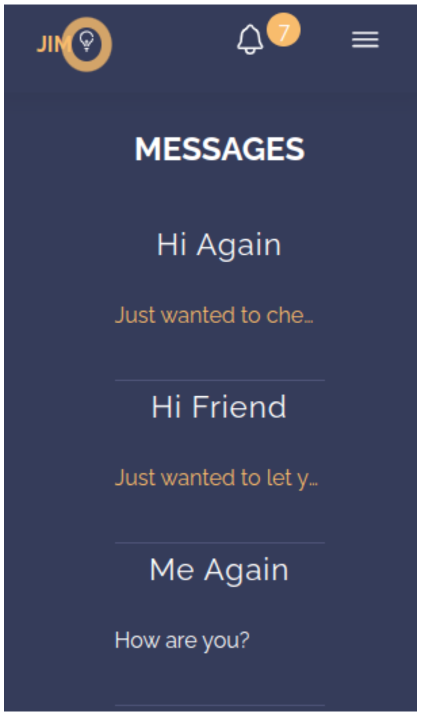


You have 7 unread messages out of 10 total.

For Navigation Bar



For Inbox Page



MESSAGES

Hi Again

Just wanted to che...

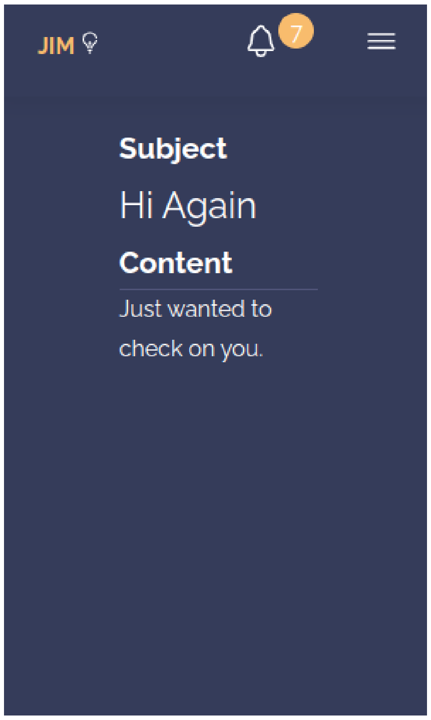
Hi Friend

Just wanted to let y...

Me Again

How are you?

For Message Page



Subject

Hi Again

Content

Just wanted to check on you.

HOW TO RUN

cd server //move into server directory

npm install //install server dependencies

cd ../client //move into client directory

npm install //install client dependencies

npm run build //build to serve with NodeJs Express

cd ../server //move back into server directory

nodemon server //run server

VIEW RUNNING PROJECT

Thank you MBL Hightech team for this wonderful opportunity to be tested for this Job position.