# Detailed Phase 1 Implementation Guide - Keeping Your UI

I'll walk you through implementing Phase 1 step-by-step while preserving your existing UI. This guide is beginner-friendly and focuses on adding AWS Amplify functionality to your existing website.

## Step 1: Set Up AWS Amplify Project

### 1.1 Install Node.js and npm (if not already done)

Follow the instructions from the tools setup guide to install Node.js and npm.

### 1.2 Install AWS Amplify CLI

1. Open Command Prompt
2. Type: `npm install -g @aws-amplify/cli`
3. Wait for installation to complete

### 1.3 Configure AWS Amplify with your AWS Account

1. In Command Prompt, type: `amplify configure`
2. Follow the prompts to sign in to AWS and create an IAM user
3. Save your access key and secret key when prompted

### 1.4 Initialize Amplify in your project

1. Navigate to your project folder in Command Prompt:

```javascript
cd c:/Users/shrey/OneDrive/Documents/GitHub/ShreyasFitnessWeb
```

2. Initialize Amplify:

```javascript
amplify init
```

3. Answer the prompts:

- __Enter a name for the project:__ "ShreyasFitnessWeb" (or any name you prefer)
- __Enter a name for the environment:__ "dev"
- __Choose your default editor:__ "Visual Studio Code"
- __Choose the type of app:__ "javascript"
- __Choose JavaScript framework:__ "none"
- __Source directory path:__ "/" (just press Enter)
- __Distribution directory path:__ "/" (just press Enter)
- __Build command:__ (just press Enter)
- __Start command:__ (just press Enter)
- __Select "Yes"__ when asked if you want to use an AWS profile

## Step 2: Add Authentication

### 2.1 Add Authentication to your project

1. In Command Prompt, type:

```javascript
amplify add auth
```

2. Answer the prompts:

- __Choose the default configuration:__ "Default configuration"

- __How do you want users to sign in:__ "Username" (they'll actually use email as username)

- __Do you want to configure advanced settings:__ "Yes"

- __What attributes are required:__ Press spacebar to select "Email" and "Name", then press Enter

- __Do you want to enable any of the following capabilities:__ Don't select any, just press Enter

- __Select "Yes"__ for "Do you want to edit your user pool groups"

- __Select "Add a group"__ and create two groups:

  - First group name: "clients"

  - Second group name: "coaches"

- Select "No" when asked if you want to add another group

### 2.2 Add API for Messages

1. In Command Prompt, type:

  ```javascript
  amplify add api
  ```

2. Answer the prompts:

- Select __GraphQL__ as the API service
- Enter API name: __shreymethodapi__
- Select authorization type: __Amazon Cognito User Pool__
- Do you want to configure additional auth types: __No__
- Select __Single object with fields__ as the schema template
- Do you want to edit the schema now: __Yes__

3. It will open a file in VS Code. Replace EVERYTHING in this file with:

```graphql
type Message @model
```

```
@auth(rules: [
 { allow: groups, groups: ["coaches"], operations: [create, read, update, delete] },
 { allow: owner, operations: [create, read] }
]) {
 id: ID!
 senderName: String!
 senderEmail: String!
 subject: String!
 content: String!
 read: Boolean!
 archived: Boolean!
 createdAt: AWSDateTime!
}

type User @model
@auth(rules: [
 { allow: groups, groups: ["coaches"], operations: [read, update] },
 { allow: owner, operations: [read, update] }
]) {
 id: ID!
 name: String!
 email: String!
 phone: String
 userGroup: String!
 createdAt: AWSDateTime!
}
```

4. Save the file (Ctrl+S)

### 2.3 Deploy your backend resources

1. In Command Prompt, type:

```javascript
amplify push
```

2. Review the changes and confirm by typing "y"

3. Select "Yes" when asked to generate code for the GraphQL API

4. Choose the default options for the code generation

## Step 3: Create AWS Configuration File

### 3.1 Get your AWS configuration values

1. In Command Prompt, type:

```javascript
amplify status
```

  This shows the resources you've added.

2. Then type:

  ```javascript
  amplify console auth
  ```

  This will open the AWS Cognito console in your browser.

3. In the AWS Cognito console:

  - Click on "User Pools" in the left sidebar if it's not already selected
  - Click on your user pool from the list
  - Click on "App integration" in the left sidebar
  - Under "App clients and analytics", find your app client
  - Note the "App client ID" - this is your `userPoolWebClientId`
  - At the top of the page, note the "User Pool ID" - this is your `userPoolId`
  - Note the "Region" in the URL of your browser (e.g., "us-east-1" or "us-west-2") - this is your region

### 3.2 Create AWS configuration file

1. Create a new file called `js/aws-config.js` in your project

2. Add the following code, replacing the placeholder values with your actual values:

```javascript
// AWS Amplify Configuration
const awsConfig = {
  Auth: {
    region: 'us-west-2', // Replace with your actual region
    userPoolId: 'us-west-2_xxxxxxxx', // Replace with your actual User Pool ID
    userPoolWebClientId: 'xxxxxxxxxxxxxxxxxxxxxxxxxx', // Replace with your actual App Client ID
  },
  API: {
    endpoints: [{
      name: "shreymethodapi",
      endpoint: "https://xxxxxxxxxx.execute-api.us-west-2.amazonaws.com/dev" // This will be updated later
    }]
  }
};
```

3. Save the file

## Step 4: Connect Your Signup Form

### 4.1 Add AWS Amplify to signup.html

1. Open `signup.html` in VS Code

2. Add the AWS Amplify library by adding this line before the closing `</head>` tag:

  ```html
  <script src="https://cdn.jsdelivr.net/npm/aws-amplify@5.0.4/dist/aws-amplify.min.js"></script>
  ```

3. Add your AWS configuration file by adding this line after the Amplify library:

  ```html
  <script src="js/aws-config.js"></script>
  ```

### 4.2 Create signup JavaScript file

1. Create a new file called `js/signup-handler.js`
2. Add the following code:

```javascript
// Initialize Amplify with our configuration
Amplify.configure(awsConfig);

// Wait for the DOM to be fully loaded
document.addEventListener('DOMContentLoaded', function() {
```

```
// Get the signup form
const signupForm = document.querySelector('.signup-form');

// Add submit event listener
signupForm.addEventListener('submit', async function(event) {
  // Prevent the default form submission
  event.preventDefault();

  // Get form values
  const fullName = document.getElementById('fullname').value;
  const email = document.getElementById('email').value;
  const phone = document.getElementById('phone').value || '';
  const password = document.getElementById('password').value;
  const confirmPassword = document.getElementById('confirm-password').value;

  // Basic validation
  if (password !== confirmPassword) {
    alert('Passwords do not match!');
    return;
  }

  try {
    // Show loading state
    const submitButton = signupForm.querySelector('button[type="submit"]');
    const originalButtonText = submitButton.innerHTML;
```

```
        submitButton.innerHTML = '<i class="fas fa-spinner fa-spin"></i> Creating 
Account...';
        submitButton.disabled = true;

        // Sign up the user with Cognito
        const { user } = await Amplify.Auth.signUp({
          username: email,
          password: password,
          attributes: {
            email: email,
            name: fullName,
            phone_number: phone ? `+1${phone.replace(/\D/g, '')}` : undefined
          }
        });

        console.log('Sign up successful!', user);

        // Create a success message div
        const successMessage = document.createElement('div');
        successMessage.className = 'success-message';
        successMessage.innerHTML = `
          <div style="text-align: center; padding: 40px 20px;">
            <div style="font-size: 48px; color: #4CAF50; margin-bottom: 20px;">
              <i class="fas fa-check-circle"></i>
            </div>
            <h3 style="color: #4CAF50; margin-bottom: 15px;">Account Created 
Successfully!</h3>
```

```
            <p style="margin-bottom: 25px;">We've sent a verification code to your email.
Please check your inbox to verify your account.</p>
            <a href="account.html" style="display: inline-block; background-color: #4CAF50;
color: white; padding: 10px 20px; border-radius: 5px; text-decoration: none;">
              <i class="fas fa-sign-in-alt"></i> Go to Login
            </a>
          </div>
        `;

        // Replace the form with the success message
        const signupContainer = document.querySelector('.signup-form-container');
        signupContainer.innerHTML = '';
        signupContainer.appendChild(successMessage);

      } catch (error) {
        console.error('Error signing up:', error);
        alert(`Sign up failed: ${error.message}`);

        // Reset button
        const submitButton = signupForm.querySelector('button[type="submit"]');
        submitButton.innerHTML = originalButtonText;
        submitButton.disabled = false;
      }
    });

    // Remove the "Coming Soon" overlay
    const comingSoonOverlay = document.querySelector('div[style*="position: fixed"]');
```

    if (comingSoonOverlay) {
        comingSoonOverlay.remove();
    }
});

3. Save the file

### 4.3 Add the signup handler to signup.html

1. Add this line before the closing `</body>` tag in `signup.html`:

   ```html
   <script src="js/signup-handler.js"></script>
   ```

2. Save the file

## Step 5: Connect Your Login Form

### 5.1 Add AWS Amplify to account.html

1. Open `account.html` in VS Code

2. Add the AWS Amplify library by adding this line before the closing `</head>` tag:

```html
   <script src="https://cdn.jsdelivr.net/npm/aws-amplify@5.0.4/dist/aws-amplify.min.js"></script>
   ```

3. Add your AWS configuration file by adding this line after the Amplify library:

   ```html
   <script src="js/aws-config.js"></script>
   ```

### 5.2 Create login JavaScript file

1. Create a new file called `js/login-handler.js`
2. Add the following code:

```javascript
// Initialize Amplify with our configuration
Amplify.configure(awsConfig);

// Wait for the DOM to be fully loaded
document.addEventListener('DOMContentLoaded', function() {
  // Get the login form
  const loginForm = document.getElementById('login-form');

  // Add submit event listener
```

```
loginForm.addEventListener('submit', async function(event) {
  // Prevent the default form submission
  event.preventDefault();

  // Get form values
  const email = document.getElementById('login-email').value;
  const password = document.getElementById('login-password').value;

  try {
    // Show loading state
    const submitButton = loginForm.querySelector('button[type="submit"]');
    const originalButtonText = submitButton.innerHTML;
    submitButton.innerHTML = '<i class="fas fa-spinner fa-spin"></i> Logging In...';
    submitButton.disabled = true;

    // Sign in the user with Cognito
    const user = await Amplify.Auth.signIn(email, password);
    console.log('Sign in successful!', user);

    // Check user group and redirect accordingly
    const session = await Amplify.Auth.currentSession();
    const idToken = session.getIdToken().payload;

    // Update the user name in the dashboard
    document.getElementById('user-name').textContent = user.attributes.name || email;
```

```
        // Hide the login form and show the dashboard
        document.querySelector('.hero-content-account').style.display = 'none';
        document.getElementById('dashboard').classList.remove('hidden');

        // If user is in coaches group, redirect to coach dashboard
        if (idToken['cognito:groups'] && idToken['cognito:groups'].includes('coaches')) {
            // For now, we'll just alert - we'll create the coach dashboard later
            alert('Coach login detected! Redirecting to coach dashboard...');
            // In the future: window.location.href = 'coach-dashboard.html';
        }

    } catch (error) {
        console.error('Error signing in:', error);
        alert(`Login failed: ${error.message}`);

        // Reset button
        const submitButton = loginForm.querySelector('button[type="submit"]');
        submitButton.innerHTML = originalButtonText;
        submitButton.disabled = false;
    }
});

// Add logout functionality
const logoutBtn = document.getElementById('logout-btn');
if (logoutBtn) {
    logoutBtn.addEventListener('click', async function() {
        try {
          await Amplify.Auth.signOut();
          console.log('Sign out successful!');

          // Show the login form and hide the dashboard
          document.querySelector('.hero-content-account').style.display = 'block';
          document.getElementById('dashboard').classList.add('hidden');

        } catch (error) {
          console.error('Error signing out:', error);
          alert(`Logout failed: ${error.message}`);
        }
      });
    }

    // Check if user is already signed in
    async function checkAuthState() {
      try {
        const user = await Amplify.Auth.currentAuthenticatedUser();
        console.log('User is signed in:', user);

        // Update the user name in the dashboard
        document.getElementById('user-name').textContent = user.attributes.name ||
user.username;

        // Hide the login form and show the dashboard
        document.querySelector('.hero-content-account').style.display = 'none';
        document.getElementById('dashboard').classList.remove('hidden');

      } catch (error) {
        console.log('User is not signed in');
        // User is not signed in, show the login form (default state)
      }
    }

    // Check auth state when page loads
    checkAuthState();

    // Remove the "Coming Soon" overlay
    const comingSoonOverlay = document.querySelector('div[style*="position: fixed"]');
    if (comingSoonOverlay) {
      comingSoonOverlay.remove();
    }
});
```

3. Save the file

### 5.3 Add the login handler to account.html

1. Add this line before the closing `</body>` tag in `account.html`:

```html
  <script src="js/login-handler.js"></script>
  ```

2. Save the file

## Step 6: Connect Your Contact Form

### 6.1 Add AWS Amplify to connect.html

1. Open `connect.html` in VS Code

2. Add the AWS Amplify library by adding this line before the closing `</head>` tag:

  ```html
  <script src="https://cdn.jsdelivr.net/npm/aws-amplify@5.0.4/dist/aws-amplify.min.js"></script>
  ```

3. Add your AWS configuration file by adding this line after the Amplify library:

  ```html
  <script src="js/aws-config.js"></script>
  ```

### 6.2 Create contact form JavaScript file

1. Create a new file called `js/contact-handler.js`
2. Add the following code:

```javascript
// Initialize Amplify with our configuration
Amplify.configure(awsConfig);

// Wait for the DOM to be fully loaded
document.addEventListener('DOMContentLoaded', function() {
  // Get the contact form
  const contactForm = document.getElementById('contact-form');

  // Add submit event listener
  contactForm.addEventListener('submit', async function(event) {
    // Prevent the default form submission
    event.preventDefault();

    // Get form values
    const name = document.getElementById('name').value;
    const email = document.getElementById('email').value;
    const phone = document.getElementById('phone').value || '';
    const service = document.getElementById('service').value;
    const messageText = document.getElementById('message-text').value;

    try {
```

```
// Show loading state
const submitButton = contactForm.querySelector('button[type="submit"]');
const originalButtonText = submitButton.innerHTML;
submitButton.innerHTML = '<i class="fas fa-spinner fa-spin"></i> Sending...';
submitButton.disabled = true;

// Create message in database using GraphQL API
const createMessageMutation = `
  mutation CreateMessage($input: CreateMessageInput!) {
    createMessage(input: $input) {
      id
      senderName
      senderEmail
      subject
      content
      read
      archived
      createdAt
    }
  }
`;

// Prepare the message input
const messageInput = {
  senderName: name,
  senderEmail: email,
    subject: `New ${service} Inquiry`,

    content: `Service Interest: ${service}\nPhone: ${phone}\n\n${messageText}`,

    read: false,

    archived: false

};


// Send the message to the API

const response = await Amplify.API.graphql({

    query: createMessageMutation,

    variables: {

        input: messageInput

    },

    authMode: 'API_KEY' // Use API key for unauthenticated users

});


console.log('Message sent successfully:', response);


// Hide the form

contactForm.style.display = 'none';


// Show success message (using your existing success message code)

const successMessage = document.createElement('div');

successMessage.className = 'success-message';

successMessage.innerHTML = `

    <div class="success-icon">

        <i class="fas fa-check-circle"></i>
```

```
            </div>

            <h3>Message Sent Successfully!</h3>

            <p>Thank you for reaching out, ${name}. I'll get back to you regarding your interest
in ${service} within 2-4 hours.</p>

            <button class="btn-secondary" onclick="window.location.reload()">Send Another
Message</button>

        `;


        contactForm.parentNode.appendChild(successMessage);


        // Add some basic styles for the success message

        const style = document.createElement('style');

        style.textContent = `

          .success-message {

            text-align: center;

            padding: 40px 20px;

            background-color: #f8fff8;

            border-radius: 10px;

            border: 1px solid #4CAF50;

          }

          .success-icon {

            font-size: 48px;

            color: #4CAF50;

            margin-bottom: 20px;

          }

          .success-message h3 {

            color: #4CAF50;
            margin-bottom: 15px;
          }
          .success-message p {
            margin-bottom: 25px;
          }
        `;
        document.head.appendChild(style);

      } catch (error) {
        console.error('Error sending message:', error);
        alert(`Failed to send message: ${error.message}`);

        // Reset button
        const submitButton = contactForm.querySelector('button[type="submit"]');
        submitButton.innerHTML = originalButtonText;
        submitButton.disabled = false;
      }
    });
});
```

3. Save the file

### 6.3 Add the contact handler to connect.html

1. Add this line before the closing `</body>` tag in `connect.html` (but after the existing script tags):

  ```html
  <script src="js/contact-handler.js"></script>
  ```

2. Save the file

## Step 7: Create a Simple Coach Dashboard

### 7.1 Create coach dashboard HTML file

1. Create a new file called `coach-dashboard.html` in your project root
2. Add the following code:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Coach Dashboard - SHREY.FIT</title>
    <link rel="stylesheet" href="css/styles.css">
    <link
href="https://fonts.googleapis.com/css2?family=Inter:wght@300;400;500;600;700&displa
y=swap" rel="stylesheet">
```

```
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.0.0/css/all.min.css">
  <script src="https://cdn.jsdelivr.net/npm/aws-amplify@5.0.4/dist/aws-
amplify.min.js"></script>
  <script src="js/aws-config.js"></script>
  <style>
    /* Coach Dashboard Styles */
    .coach-dashboard {
      padding: 40px 0;
      background-color: #f9f9f9;
      min-height: 100vh;
    }
    .dashboard-header {
      display: flex;
      justify-content: space-between;
      align-items: center;
      margin-bottom: 30px;
    }
    .dashboard-title {
      margin: 0;
      color: #2E7D32;
    }
    .messages-container {
      background-color: white;
      border-radius: 10px;
      box-shadow: 0 2px 10px rgba(0,0,0,0.1);
      padding: 20px;
  margin-bottom: 30px;
}
.message-item {
  border-bottom: 1px solid #eee;
  padding: 15px 0;
  display: flex;
  align-items: flex-start;
}
.message-item:last-child {
  border-bottom: none;
}
.message-status {
  width: 10px;
  height: 10px;
  border-radius: 50%;
  background-color: #4CAF50;
  margin-right: 15px;
  margin-top: 8px;
}
.message-status.read {
  background-color: #ccc;
}
.message-content {
  flex: 1;
}
.message-header {
  display: flex;
  justify-content: space-between;
  margin-bottom: 5px;
}
.message-sender {
  font-weight: 600;
}
.message-date {
  color: #777;
  font-size: 0.9em;
}
.message-subject {
  font-weight: 500;
  margin-bottom: 5px;
}
.message-preview {
  color: #555;
  font-size: 0.9em;
  white-space: nowrap;
  overflow: hidden;
  text-overflow: ellipsis;
  max-width: 600px;
}
.message-actions {
  display: flex;
  gap: 10px;
  margin-left: 15px;
}
.message-btn {
  background: none;
  border: none;
  color: #4CAF50;
  cursor: pointer;
  font-size: 1.1em;
}
.message-btn:hover {
  color: #2E7D32;
}
.logout-btn {
  background-color: #f44336;
  color: white;
  border: none;
  padding: 10px 15px;
  border-radius: 4px;
  cursor: pointer;
}
.message-modal {
  display: none;
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background-color: rgba(0,0,0,0.5);
  z-index: 1000;
  justify-content: center;
  align-items: center;
}
.modal-content {
  background-color: white;
  border-radius: 10px;
  width: 90%;
  max-width: 600px;
  max-height: 80vh;
  overflow-y: auto;
  padding: 30px;
  position: relative;
}
.close-modal {
  position: absolute;
  top: 15px;
  right: 15px;
  font-size: 1.5em;
  cursor: pointer;
  color: #777;
}
.close-modal:hover {
  color: #333;
```

```
        }
        .modal-header {
          margin-bottom: 20px;
        }
        .modal-subject {
          margin: 0 0 5px 0;
          color: #2E7D32;
        }
        .modal-sender {
          margin: 0 0 15px 0;
          color: #555;
        }
        .modal-body {
          white-space: pre-wrap;
          line-height: 1.6;
        }
        .no-messages {
          text-align: center;
          padding: 40px 0;
          color: #777;
        }
    </style>
</head>
<body>
    <!-- Navigation -->
    <nav class="navbar">
```

```
        <div class="nav-container">
          <div class="nav-logo">
            <a href="index.html" class="logo-link">
              <div class="logo-container">
                <div class="logo-text">
                  <span class="brand-name">SHREY<span class="brand-
    dot">.</span>FIT</span>
                </div>
              </div>
            </a>
          </div>
          <ul class="nav-menu">
            <li class="nav-item">
              <a href="#" class="nav-link active">Dashboard</a>
            </li>
            <li class="nav-item">
              <a href="#" class="nav-link">Clients</a>
            </li>
            <li class="nav-item">
              <a href="#" class="nav-link">Programs</a>
            </li>
            <li class="nav-item">
              <a href="#" class="nav-link">Schedule</a>
            </li>
            <li class="nav-item">
              <button id="logout-btn" class="logout-btn">Logout</button>
            </li>
        </ul>
        <div class="hamburger">
            <span class="bar"></span>
            <span class="bar"></span>
            <span class="bar"></span>
        </div>
    </div>
</nav>

<!-- Coach Dashboard -->
<section class="coach-dashboard">
    <div class="container">
        <div class="dashboard-header">
            <h1 class="dashboard-title">Coach Dashboard</h1>
            <div class="user-info">
                Welcome, <span id="coach-name">Coach</span>
            </div>
        </div>

        <div class="messages-container">
            <h2>Messages</h2>
            <div id="messages-list">
                <!-- Messages will be loaded here -->
                <div class="loading">Loading messages...</div>
            </div>
```

```html
            </div>
        </div>
    </section>

    <!-- Message Modal -->
    <div id="message-modal" class="message-modal">
        <div class="modal-content">
            <span class="close-modal">&times;</span>
            <div class="modal-header">
                <h2 class="modal-subject" id="modal-subject"></h2>
                <p class="modal-sender" id="modal-sender"></p>
            </div>
            <div class="modal-body" id="modal-body"></div>
        </div>
    </div>

    <script src="js/coach-dashboard.js"></script>
</body>
</html>
```

3. Save the file

### 7.2 Create coach dashboard JavaScript file

1. Create a new file called `js/coach-dashboard.js`

2. Add the following code:

```javascript
// Initialize Amplify with our configuration
Amplify.configure(awsConfig);

// Wait for the DOM to be fully loaded
document.addEventListener('DOMContentLoaded', function() {
  // Check if user is authenticated and in the coaches group
  async function checkAuth() {
    try {
      const user = await Amplify.Auth.currentAuthenticatedUser();
      console.log('User is signed in:', user);

      // Update coach name
      document.getElementById('coach-name').textContent = user.attributes.name || user.username;

      // Check if user is in coaches group
      const session = await Amplify.Auth.currentSession();
      const idToken = session.getIdToken().payload;

      if (!idToken['cognito:groups'] || !idToken['cognito:groups'].includes('coaches')) {
        // Not a coach, redirect to login
        alert('You do not have permission to access the coach dashboard.');
        window.location.href = 'account.html';
        }

        // Load messages
        fetchMessages();

    } catch (error) {
        console.log('User is not signed in');
        // Redirect to login
        window.location.href = 'account.html';
    }
}

// Fetch messages from the API
async function fetchMessages() {
    const messagesList = document.getElementById('messages-list');

    try {
        // Query for messages
        const listMessagesQuery = `
            query ListMessages {
                listMessages(limit: 100, sort: {field: "createdAt", direction: "desc"}) {
                    items {
                        id
                        senderName
                        senderEmail
                        subject
            content
            read
            archived
            createdAt
          }
        }
      }
    `;

    const response = await Amplify.API.graphql({
      query: listMessagesQuery,
      authMode: 'AMAZON_COGNITO_USER_POOLS'
    });

    const messages = response.data.listMessages.items;
    console.log('Messages:', messages);

    // Clear loading message
    messagesList.innerHTML = '';

    if (messages.length === 0) {
      messagesList.innerHTML = '<div class="no-messages">No messages yet</div>';
      return;
    }

    // Display messages
```

```
      messages.forEach(message => {
        const messageDate = new Date(message.createdAt);
        const formattedDate = messageDate.toLocaleDateString() + ' ' + messageDate.toLocaleTimeString();

        const messageItem = document.createElement('div');
        messageItem.className = 'message-item';
        messageItem.dataset.id = message.id;

        messageItem.innerHTML = `
          <div class="message-status ${message.read ? 'read' : ''}"></div>
          <div class="message-content">
            <div class="message-header">
              <div class="message-sender">${message.senderName}
` ` `
```