Phase 1: User Login System

The Shrey Method Fitness Platform Guide

← Back to Tools Setup | Next: Phase 2 - Dashboards →

In this phase, we'll create the login system for your fitness platform. This will allow:

- Clients to sign up for accounts
- Users to log in securely
- Different access levels for clients and coaches

We'll break this down into small, manageable steps.

Step 1: Initialize Your Project with Amplify

First, we need to set up AWS Amplify in your project folder.

1. Open VS Code
2. Open your project folder (File > Open Folder > select your ShreyMethodFitness folder)
3. Open the terminal (View > Terminal or press Ctrl+`)
4. Initialize Amplify in your project:

```
amplify init
```

5. Answer the prompts:
    - Enter a name for the project: "ShreyMethodFitness"
    - 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: "npm run build" (just press Enter)
    - Start command: "npm start" (just press Enter)
    - Select "Yes" when asked if you want to use an AWS profile

What's happening here? You're telling AWS Amplify about your project so it can help you build it. Amplify will create configuration files in your project folder.

Step 2: Add Authentication

Now we'll add user authentication to your project.

1. In the terminal, type:

```
amplify add auth
```

2. Answer the prompts:
    - Choose the default configuration: "Default configuration"
    - How do you want users to sign in: "Username"
    - Do you want to configure advanced settings: "Yes"
    - What attributes are required: Select "Email" and "Name" using spacebar, 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

What's happening here? You're setting up the system that will handle user accounts, logins, and passwords. You're also creating two user groups: one for clients and one for coaches.

Step 3: Set Up Database for Messages

Let's create a database to store messages between you and your clients.

1. In the terminal, type:

```
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:

```
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 or File > Save)

What's happening here? You're creating a database to store messages between you and your clients. The code defines two types of data: Messages and Users, along with rules about who can access them.

Step 4: Deploy Your Backend Resources

Now let's upload your configuration to AWS so they can create all the necessary resources.

1. In the terminal, type:

```
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

What's happening here? AWS is creating all the necessary resources for your application based on your configuration. This includes the authentication system and database.

Step 5: Create Login Page

Now let's create the login page that users will see when they want to access their dashboard.

1. In VS Code, create a new file called `client-login.html`
2. Copy and paste this code:

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login - The Shrey Method Fitness</title>
    <link rel="stylesheet" href="css/styles.css">
    <!-- Add AWS Amplify libraries -->
    <script src="https://cdn.jsdelivr.net/npm/aws-amplify@5.0.4/dist/aws-
amplify.min.js"></script>
</head>
<body>
    <!-- Navigation bar (reuse existing) -->

    <div class="auth-container">
        <div class="auth-form">
            <h2>Log In to Your Dashboard</h2>
            <div id="error-message" class="error-message"></div>

            <form id="login-form">
                <div class="form-group">
                    <label for="email">Email</label>
                    <input type="email" id="email" required>
                </div>

                <div class="form-group">
                    <label for="password">Password</label>
                    <input type="password" id="password" required>
                </div>

                <div class="form-actions">
                    <a href="forgot-password.html" class="forgot-password">Forgot
Password?</a>
                    <button type="submit" class="btn-primary">Log In</button>
                </div>
            </form>

            <p class="auth-redirect">New client? <a href="signup.html">Sign Up</a>
</p>
        </div>
    </div>

    <script src="js/auth.js"></script>
</body>
</html>
```

3. Save the file (Ctrl+S or File > Save)

What's happening here? You're creating the HTML structure for the login page, which includes a form for users to enter their email and password.

Step 6: Create Authentication JavaScript

Now let's create the JavaScript code that will handle the login process.

1. Create a folder called `js` in your project folder
2. Create a new file called `js/auth.js`
3. Copy and paste this code:

```
// Configure Amplify
const awsConfig = {
    Auth: {
        region: 'us-west-2', // Your AWS region
        userPoolId: 'us-west-2_xxxxxxxx', // Your Cognito User Pool ID
        userPoolWebClientId: 'xxxxxxxxxxxxxxxxxxxxxxxxxx', // Your App Client ID
    }
};

// Replace the placeholder values with your actual values
// You can find these in the AWS Cognito console or in the aws-exports.js file

Amplify.configure(awsConfig);

// Login form handler
document.getElementById('login-form').addEventListener('submit', async
function(event) {
    event.preventDefault();

    const email = document.getElementById('email').value;
    const password = document.getElementById('password').value;
    const errorMessage = document.getElementById('error-message');

    try {
        const user = await Amplify.Auth.signIn(email, password);

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

        if (idToken['cognito:groups'] &&
idToken['cognito:groups'].includes('coaches')) {
            window.location.href = 'coach-dashboard.html';
        } else {
            window.location.href = 'client-dashboard.html';
        }
    } catch (error) {
        errorMessage.textContent = error.message;
    }
});
```

4. Save the file (Ctrl+S or File > Save)

What's happening here? You're creating the JavaScript code that will handle the login process when someone submits the login form. It will check if the user is a coach or a client and redirect them to the appropriate dashboard.

Step 7: Update Configuration Values

Now we need to update the JavaScript code with your actual AWS configuration values.

1. In the terminal, type:

```
amplify status
```

2. Note the name of your Auth resource (something like "authshreymethod123456")
3. Type:

```
amplify console auth
```

4. This will open the AWS Cognito console in your browser
5. Click on "User Pools" in the left sidebar
6. Click on your user pool (it should have the name you noted earlier)
7. Click on "App integration" in the left sidebar
8. Under "App clients and analytics", find your app client
9. Note the "App client ID" value
10. Go back to VS Code and open `js/auth.js`
11. Replace the placeholder values with your actual values:
    - Replace `us-west-2` with your region (if different)
    - Replace `us-west-2_xxxxxxxx` with your User Pool ID
    - Replace `xxxxxxxxxxxxxxxxxxxxxxxxxx` with your App Client ID
12. Save the file (Ctrl+S or File > Save)

What's happening here? You're connecting your login page to your AWS authentication system by updating the configuration values.

Step 8: Create CSS for Login Page

Let's create the styling for your login page.

1. Create a folder called `css` in your project folder
2. Create a new file called `css/styles.css`
3. Copy and paste this code:

```
/* General Styles */
:root {
    --primary: #4CAF50;
    --primary-dark: #388E3C;
    --secondary: #2E7D32;
    --light-gray: #f5f5f5;
    --dark-gray: #333;
}
```

```
body {
    font-family: 'Inter', sans-serif;
    margin: 0;
    padding: 0;
    color: var(--dark-gray);
    line-height: 1.6;
}

.container {
    max-width: 1200px;
    margin: 0 auto;
    padding: 0 20px;
}

/* Authentication Styles */
.auth-container {
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
    background-color: var(--light-gray);
}

.auth-form {
    background-color: white;
    padding: 40px;
    border-radius: 10px;
    box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1);
    width: 100%;
    max-width: 400px;
}

.auth-form h2 {
    margin-top: 0;
    margin-bottom: 20px;
    color: var(--secondary);
    text-align: center;
}

.form-group {
    margin-bottom: 20px;
}

.form-group label {
    display: block;
    margin-bottom: 5px;
    font-weight: 500;
}

.form-group input {
    width: 100%;
    padding: 10px;
    border: 1px solid #ddd;
    border-radius: 5px;
```

```
    font-size: 16px;
}

.form-actions {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-top: 30px;
}

.forgot-password {
    color: var(--primary);
    text-decoration: none;
}

.forgot-password:hover {
    text-decoration: underline;
}

.btn-primary {
    background-color: var(--primary);
    color: white;
    border: none;
    padding: 10px 20px;
    border-radius: 5px;
    cursor: pointer;
    font-size: 16px;
    font-weight: 500;
    transition: background-color 0.3s ease;
}

.btn-primary:hover {
    background-color: var(--primary-dark);
}

.auth-redirect {
    text-align: center;
    margin-top: 20px;
}

.auth-redirect a {
    color: var(--primary);
    text-decoration: none;
}

.auth-redirect a:hover {
    text-decoration: underline;
}

.error-message {
    color: red;
    margin-bottom: 15px;
    text-align: center;
}
```

4. Save the file (Ctrl+S or File > Save)

What's happening here? You're creating the CSS styling for your login page so it looks nice and professional.

Step 9: Create Signup Page

Now let's create the signup page for new users.

1. Create a new file called `signup.html`
2. Copy and paste this code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sign Up - The Shrey Method Fitness</title>
    <link rel="stylesheet" href="css/styles.css">
    <!-- Add AWS Amplify libraries -->
    <script src="https://cdn.jsdelivr.net/npm/aws-amplify@5.0.4/dist/aws-
amplify.min.js"></script>
</head>
<body>
    <!-- Navigation bar (reuse existing) -->

    <div class="auth-container">
        <div class="auth-form">
            <h2>Create Your Account</h2>
            <div id="error-message" class="error-message"></div>

            <form id="signup-form">
                <div class="form-group">
                    <label for="name">Full Name</label>
                    <input type="text" id="name" required>
                </div>

                <div class="form-group">
                    <label for="email">Email</label>
                    <input type="email" id="email" required>
                </div>

                <div class="form-group">
                    <label for="password">Password</label>
                    <input type="password" id="password" required>
                </div>

                <div class="form-group">
                    <label for="confirm-password">Confirm Password</label>
                    <input type="password" id="confirm-password" required>
                </div>
```

```
                <div class="form-actions">
                    <button type="submit" class="btn-primary">Sign Up</button>
                </div>
            </form>

            <p class="auth-redirect">Already have an account? <a href="client-
login.html">Log In</a></p>
        </div>
    </div>

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

3. Save the file (Ctrl+S or File > Save)

What's happening here? You're creating the HTML structure for the signup page, which includes a form for users to enter their name, email, and password.

Step 10: Create Signup JavaScript

Now let's create the JavaScript code that will handle the signup process.

1. Create a new file called `js/signup.js`
2. Copy and paste this code:

```
// Configure Amplify (same as auth.js)
const awsConfig = {
    Auth: {
        region: 'us-west-2', // Your AWS region
        userPoolId: 'us-west-2_xxxxxxxx', // Your Cognito User Pool ID
        userPoolWebClientId: 'xxxxxxxxxxxxxxxxxxxxxxxxxx', // Your App Client ID
    }
};

// Replace with your actual values (same as in auth.js)

Amplify.configure(awsConfig);

// Signup form handler
document.getElementById('signup-form').addEventListener('submit', async
function(event) {
    event.preventDefault();

    const name = document.getElementById('name').value;
    const email = document.getElementById('email').value;
    const password = document.getElementById('password').value;
    const confirmPassword = document.getElementById('confirm-password').value;
    const errorMessage = document.getElementById('error-message');

    // Clear previous error messages
```

```
    errorMessage.textContent = '';

    // Validate passwords match
    if (password !== confirmPassword) {
        errorMessage.textContent = 'Passwords do not match';
        return;
    }

    try {
        // Sign up the user
        const { user } = await Amplify.Auth.signUp({
            username: email,
            password: password,
            attributes: {
                email: email,
                name: name
            }
        });

        // Redirect to verification page
        window.location.href = 'verify.html?email=' + encodeURIComponent(email);

    } catch (error) {
        errorMessage.textContent = error.message;
    }
});
```

3. Save the file (Ctrl+S or File > Save)
4. Update the configuration values with your actual values (same as in Step 7)

What's happening here? You're creating the JavaScript code that will handle the signup process when someone submits the signup form. It will create a new user account and redirect them to the verification page.

Step 11: Create Verification Page

Finally, let's create the verification page that users will see after signing up.

1. Create a new file called `verify.html`
2. Copy and paste this code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Verify Account - The Shrey Method Fitness</title>
    <link rel="stylesheet" href="css/styles.css">
    <!-- Add AWS Amplify libraries -->
    <script src="https://cdn.jsdelivr.net/npm/aws-amplify@5.0.4/dist/aws-
amplify.min.js"></script>
</head>
```

```
<body>
    <!-- Navigation bar (reuse existing) -->

    <div class="auth-container">
        <div class="auth-form">
            <h2>Verify Your Account</h2>
            <p>We've sent a verification code to your email. Please enter it below
to verify your account.</p>
            <div id="error-message" class="error-message"></div>

            <form id="verify-form">
                <div class="form-group">
                    <label for="email">Email</label>
                    <input type="email" id="email" readonly>
                </div>

                <div class="form-group">
                    <label for="code">Verification Code</label>
                    <input type="text" id="code" required>
                </div>

                <div class="form-actions">
                    <button type="submit" class="btn-primary">Verify
Account</button>
                </div>
            </form>

            <p class="auth-redirect">Didn't receive a code? <a href="#"
id="resend-code">Resend Code</a></p>
        </div>
    </div>

    <script src="js/verify.js"></script>
</body>
</html>
```

3. Save the file (Ctrl+S or File > Save)

Step 12: Create Verification JavaScript

Now let's create the JavaScript code that will handle the verification process.

1. Create a new file called `js/verify.js`
2. Copy and paste this code:

```
// Configure Amplify (same as auth.js)
const awsConfig = {
    Auth: {
        region: 'us-west-2', // Your AWS region
        userPoolId: 'us-west-2_xxxxxxxx', // Your Cognito User Pool ID
        userPoolWebClientId: 'xxxxxxxxxxxxxxxxxxxxxxxxxx', // Your App Client ID
```

```
    }
};

// Replace with your actual values (same as in auth.js)

Amplify.configure(awsConfig);

// Get email from URL parameter
function getEmailFromUrl() {
    const urlParams = new URLSearchParams(window.location.search);
    return urlParams.get('email');
}

// Set email field value
document.addEventListener('DOMContentLoaded', function() {
    const emailField = document.getElementById('email');
    const email = getEmailFromUrl();

    if (email) {
        emailField.value = email;
    } else {
        window.location.href = 'signup.html'; // Redirect if no email provided
    }
});

// Verification form handler
document.getElementById('verify-form').addEventListener('submit', async
function(event) {
    event.preventDefault();

    const email = document.getElementById('email').value;
    const code = document.getElementById('code').value;
    const errorMessage = document.getElementById('error-message');

    try {
        // Confirm signup with verification code
        await Amplify.Auth.confirmSignUp(email, code);

        // Show success message
        alert('Account verified successfully! You can now log in.');

        // Redirect to login page
        window.location.href = 'client-login.html';

    } catch (error) {
        errorMessage.textContent = error.message;
    }
});

// Resend code handler
document.getElementById('resend-code').addEventListener('click', async
function(event) {
    event.preventDefault();
```

```
    const email = document.getElementById('email').value;
    const errorMessage = document.getElementById('error-message');

    try {
        // Resend verification code
        await Amplify.Auth.resendSignUp(email);

        // Show success message
        alert('Verification code resent. Please check your email.');

    } catch (error) {
        errorMessage.textContent = error.message;
    }
});
```

3. Save the file (Ctrl+S or File > Save)
4. Update the configuration values with your actual values (same as in Step 7)

What's happening here? You're creating the JavaScript code that will handle the verification process when someone submits the verification form. It will verify their account and redirect them to the login page.

Step 13: Test Your Authentication System

Now let's test your authentication system to make sure everything works correctly.

1. Open your website in a browser:
    - Right-click on `client-login.html` in VS Code
    - Select "Open with Live Server" (if you have the Live Server extension)
    - Or open the file directly in your browser
2. Try signing up a new user:
    - Click "Sign Up" on the login page
    - Fill out the form with your name, email, and password
    - Click "Sign Up"
3. Check your email for the verification code:
    - You should receive an email with a verification code
    - Enter the code on the verification page
    - Click "Verify Account"
4. Try logging in with your new account:
    - Enter your email and password on the login page
    - Click "Log In"

What's happening here? You're testing the entire authentication flow to make sure users can sign up, verify their accounts, and log in successfully.

Congratulations!

You've completed Phase 1 of your fitness platform. You now have a working authentication system that allows users to sign up, verify their accounts, and log in. In the next phase, we'll create the dashboards for both clients and coaches.

← Back to Tools Setup | Next: Phase 2 - Dashboards →