

Project: Student Course Advising Website

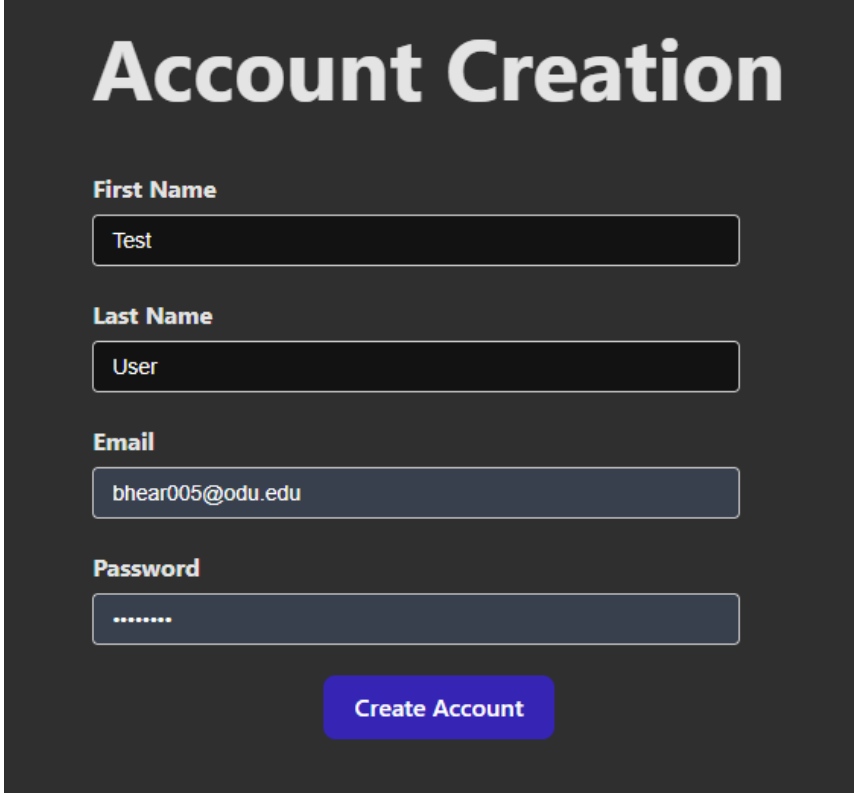
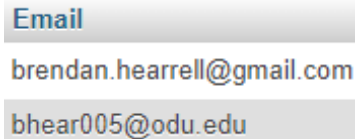
Student name: Brendan Hearrell

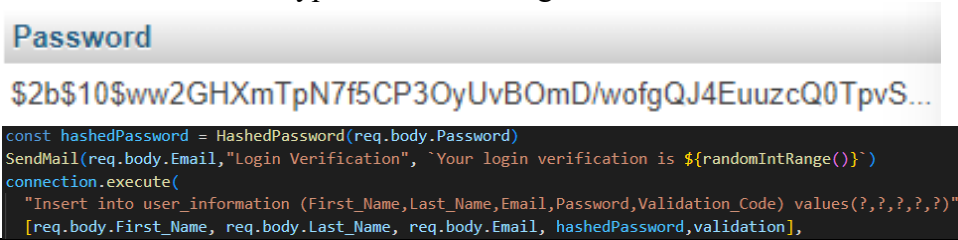
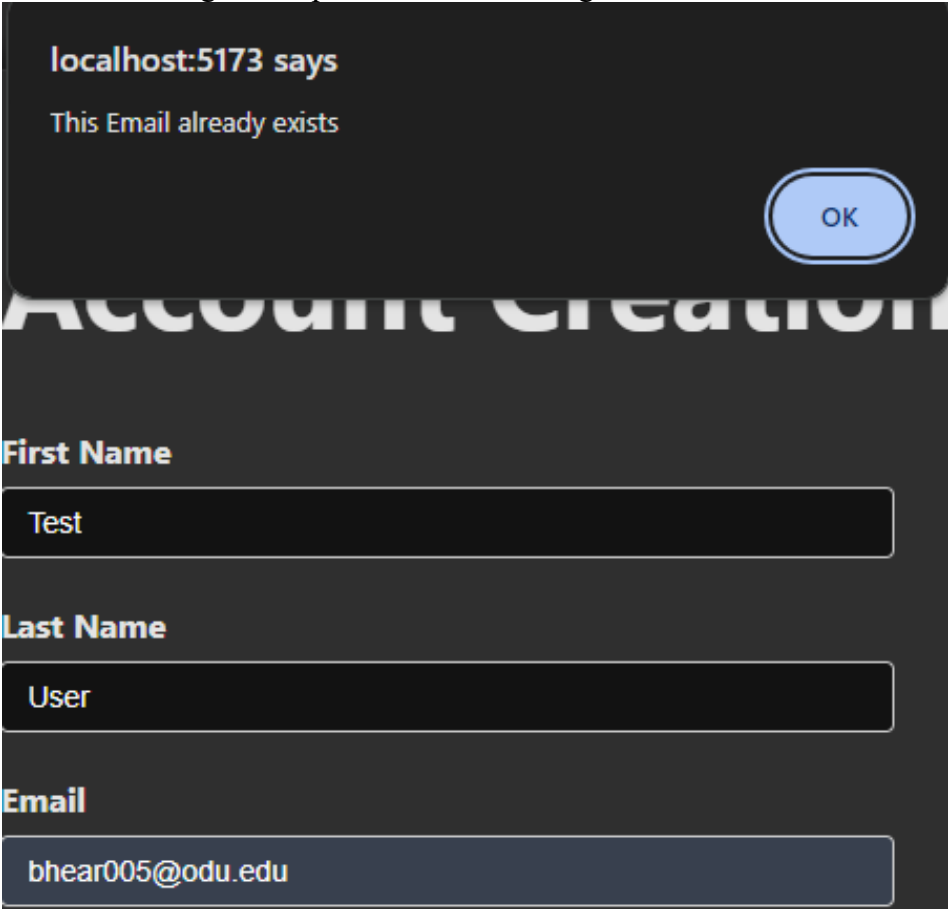
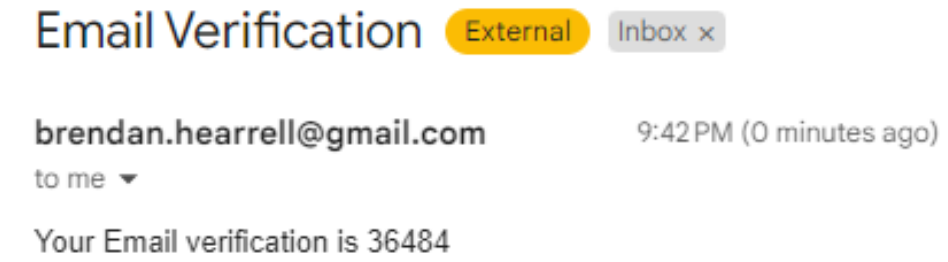
Student UIN: 01219737

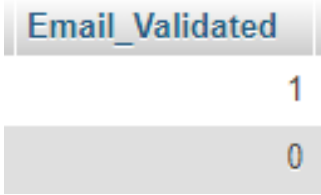
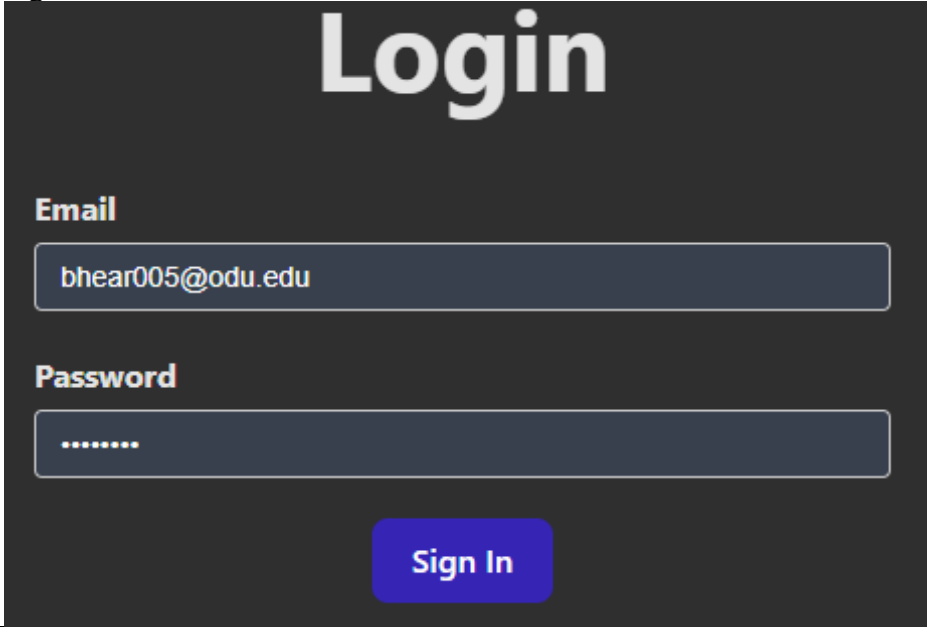
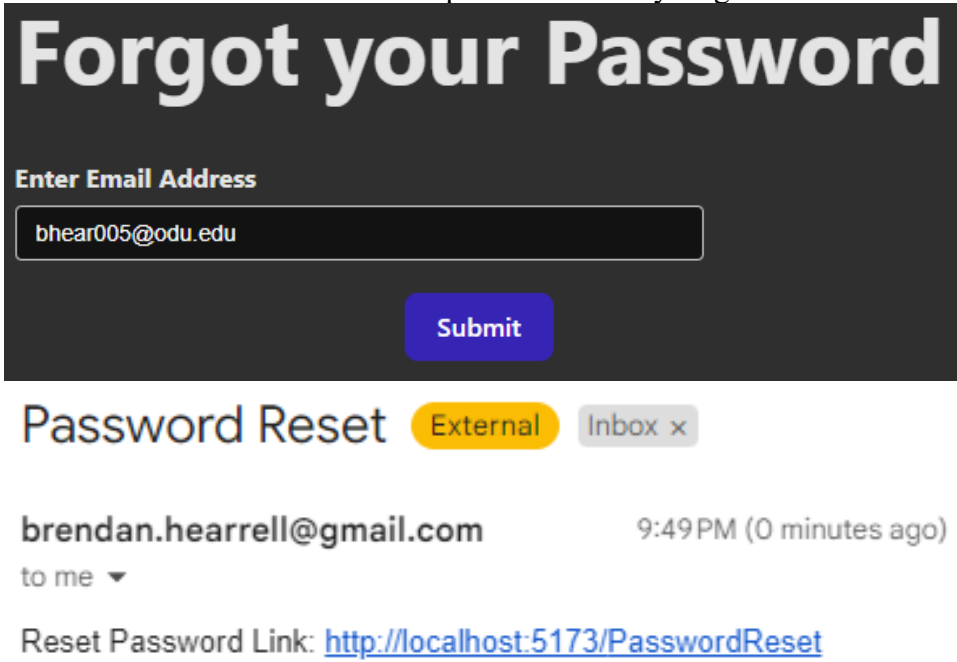
1. Overview

The website is an advising tool for students attending college. This first portion of the project covers account creation, email validation, user login, two factor authentication, account editing, and differentiating between normal user accounts and administrator accounts. The language used for the project will be JavaScript which will utilize the React library and Vite for local development. Node.js will be used as the runtime environment for JavaScript and Express.js will be used for the back-end framework.

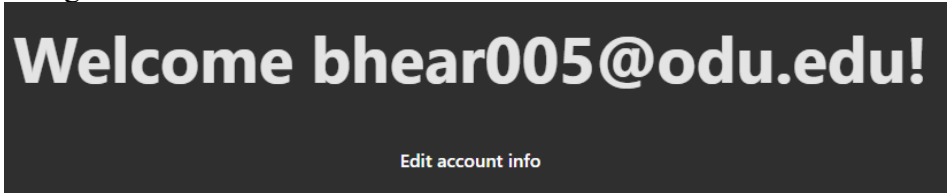
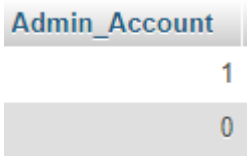
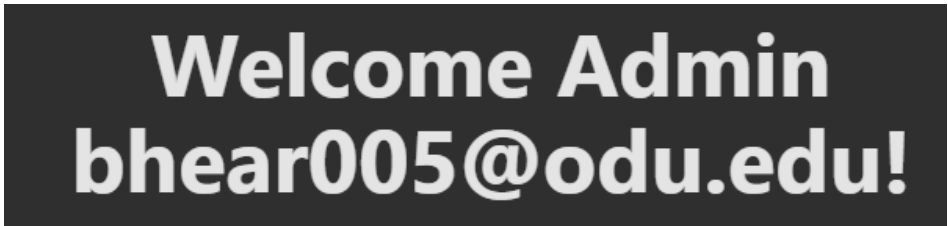
2. Milestone Accomplishments

Fulfilled	Feature	Specification
Yes	1	<div>Users should be able to register new accounts using email address.</div> <div></div>
Yes	2	<div>Users are identified by email address.</div> <div></div>

Yes	3	<p>Password must be encrypted before storing in the database.</p>  <p>The screenshot shows a password field with the value "\$2b\$10\$www2GHXmTpN7f5CP3OyUvBOMD/wofgQJ4EuuzcQ0TpvS...". Below it, a code editor shows the following code:</p> <pre>const hashedPassword = HashedPassword(req.body.Password) SendMail(req.body.Email, "Login Verification", `Your login verification is \${randomIntRange()}`) connection.execute("Insert into user_information (First_Name, Last_Name, Email, Password, Validation_Code) values(?, ?, ?, ?, ?)" [req.body.First_Name, req.body.Last_Name, req.body.Email, hashedPassword, validation],</pre>
Yes	4	<p>Users cannot register duplicate accounts using the same email address.</p>  <p>The screenshot shows a registration form titled "Account Creation". It has fields for "First Name" (containing "Test"), "Last Name" (containing "User"), and "Email" (containing "bhear005@odu.edu"). An error message "localhost:5173 says This Email already exists" is displayed at the top with an "OK" button.</p>
Yes	5	<p>The user should receive a verification email upon successful registration.</p>  <p>The screenshot shows an email titled "Email Verification" with a yellow "External" label and a grey "Inbox x" label. The email is from "brendan.hearrell@gmail.com" and is dated "9:42 PM (0 minutes ago)". The body of the email says "Your Email verification is 36484".</p>

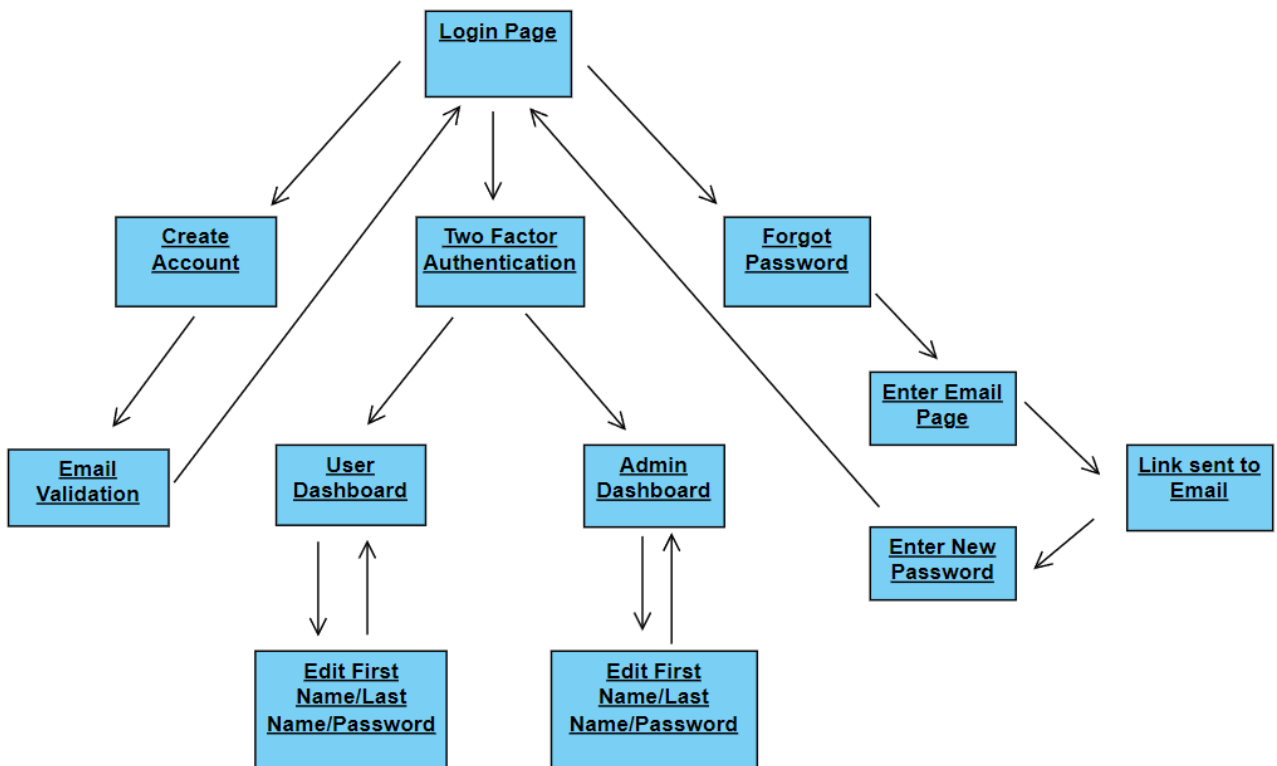
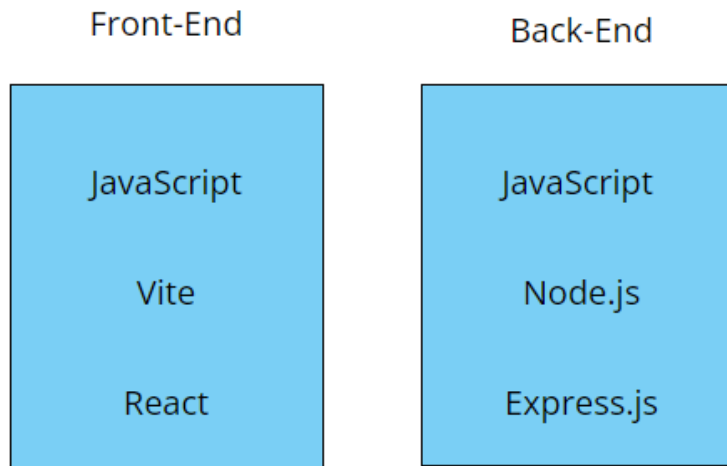
Yes	6	<p>Users cannot log in to the system until their email has been verified.</p>  <p>The user will not be allowed to login until validation code is entered.</p>
Yes	7	<p>Users should be able to log into your website using the accounts they registered.</p> 
Yes	8	<p>Users should be able to reset their passwords if they forget it.</p> 

Yes	9	<p>Users should be able to change their passwords after they login.</p> <div><h1>Edit Profile</h1><div><p>First Name</p><input type="text" value="test"/></div><div>Submit</div><div><p>Last Name</p><input type="text" value="user"/></div><div>Submit</div><div><p>Password</p><input type="password" value="....."/></div><div>Submit</div><div>Return</div></div>
Yes	10	<p>A 2-factor authentication should be used when a user attempts to login.</p> <div><h2>Login Validation</h2><div>ExternalInbox x</div><div><p>brendan.hearrell@gmail.com9:50 PM (0 minutes ago)</p><p>to me ▼</p><p>Your login validation code is 50134</p></div><div><h1>Email Verification</h1><div><p>Enter code</p><input type="text" value="50134"/></div><div>Submit</div></div></div>

Yes	11	<p>The website should have a homepage for each user, where they can view their profile, change password, and update information. Email cannot be changed</p> 
Yes	12	<p>An admin user should be created from the backend.</p> 
No	13	<p>An admin user has a different view from a regular user.</p> <p>I created an admin dashboard page, but I couldn't get the logic figured out for the actual admin account to login.</p> 

[This portion intentionally left blank.]

3. Architecture



4. Database Design

My database is set up to hold user login credentials, personal information, and tokens to be used for validation. The user_id key is used for table position identification. First and last name fall under personal information and can be edited once logged in. Email and password are used for login and user verification and password is encrypted before being stored in the database. Admin_Account is a boolean token that identifies the administrator. Email_Validated is a Boolean token used to prove user email validation. Validation_Code is the randomly generated code that is sent to the user email for two factor identification and initial email validation.

Field	Type	Key	Example	Details
user_id	int(11)	Primary	1	AUTO INCREMENT
First_Name	varchar(50)		Brendan	
Last_Name	varchar(50)		Hearrell	
Email	varchar(50)		bhear005@odu.edu	
Password	varchar(100)		Password	
Admin_Account	boolean/tinyint(1)		1	Default = 0
Email_Validated	boolean/tinyint(1)		1	Default 0 unless validated
Validation_Code	int(11)		35823	

5. Implementation

Users should be able to register new accounts using email addresses.

The login page serves as the landing page for the site currently. From the login page there is a “Forgot Password?” link that leads to the account creation page. The account creation page allows the user to enter their first name, last name, email address, and password. From there an email with a validation code will be sent and the code within can be used for validation to access the site.

Code contained within: Project/client/src/components/CreateAccount.jsx

Users are identified by email address.

In order to login to the site properly the user email will have to be validated. Once this is complete the user will be able to enter their email and password from the login page. Once their credentials are entered, they will be sent to the two-factor authentication page and an email will be sent containing the code needed.

Code contained within: Project/client/src/components/Login.jsx

Password must be encrypted before storing in the database.

Once the user enters their information into the account creation page their email is sent via `user.post('/')` API call. From there the password is hashed using the `HashPassword()` function and stored within the database.

Code contained within: `Project/client/src/components/CreateAccount.jsx`

`Project/server/routes/user.js` (API)

`Project/server/utills/helper.js` (HashPassword function)

Users cannot register duplicate accounts using the same email address.

When the user attempts to create an account with an email that already exists, the database is checked for any duplicate email addresses and returns an error code from the API. If an error code is returned, then the user is prompted with an alert stating that the email entered already exists.

Code contained within: `Project/client/src/components/CreateAccount.jsx`

`Project/server/routes/user.js` (`user.post('/check/email')`)

The user should receive a verification email upon successful registration.

After the user enters their personal information into the create account page, an email will be sent containing a code to be entered on the email verification page. Until that code is entered the user will not be able to successfully enter the site.

Code contained within: `Project/client/src/components/CreateAccount.jsx`

`Project/server/routes/user.js` (`user.post('/')`)

`Project/server/utills/SendMail.js` (Contains SendMail function)

Users cannot log in to the system until their email has been verified.

When the user is directed to the email verification page from the account creation page they will be prompted to enter a code. The code is contained within an email sent after entering their personal information. Once the code has been successfully entered the `Email_Validated` token within the database will be set to 1 indicating that the email has been validated.

Code contained within: `Project/client/src/EmailValidation.jsx`

`Project/server/utills/SendMail.js` (Contains SendMail function)

Users should be able to log into your website using the accounts they registered.

The login page is the landing page for the website. From there the user can enter their email and, if it has been validated in the database, they will be directed to the two-factor identification page.

Code contained within: Project/client/src/components/Login.jsx

Users should be able to reset their passwords if they forget it.

From the login page there is a link “Forgot Password?” which will direct the user to the forgot password page. The user will be prompted to enter their email address, and upon completion they will be sent an email containing the link <http://localhost:5173/PasswordReset>. From the password reset page the user can enter a new password and will be directed to the login page to enter the site.

Code contained within: Project/client/src/components/EmailRecovery.jsx (Forgot password)

Project/client/src/components/PasswordReset.jsx (Reset Password)

Project/server/utils/SendMail.js (Contains SendMail function)

Users should be able to change their passwords after they login.

From the user dashboard page under the greeting there is an ‘Edit Account’ link. From there the user can change their password. Just like the initial setup the password is hashed before being stored in the database.

Code contained within: Project/client/src/components/Dashboard.jsx

Project/client/src/components/EditProfile.jsx

A 2-factor-authentication should be used when a user attempts to login.

From the login page, after the user enters their credentials, they are directed to the two-factor authentication page. The code needed is contained within an email, and when successfully entered, the user is directed to the dashboard. Otherwise, they will be prompted to re-enter the code.

Code contained within: Project/client/src/components/Login.jsx

Project/server/utils/SendMail.js (Contains SendMail function)

The website should have a homepage for each user, where they can view their profiles, change passwords, and update information. Email cannot be changed.

Once the user enters the dashboard page, under the welcome there is an 'Edit Account' link. From there the user can change their first name, last name, and password and they can return to the dashboard.

Code contained within: Project/client/src/components/Dashboard.jsx

Project/client/src/components/EditAccount.jsx

An admin user should be created from the backend.

Within mysql I created a token Admin_Account which contains a Boolean identifier. From there I created one admin user, and all subsequent new accounts are defaulted to 0 which is identified as a normal user account.

An admin user has a different view from a regular user.

I was not able to implement the logic to identify an admin account and direct them to the admin dashboard. I did however create the admin dashboard page and plan to implement the logic in the coming days.

Code contained within: Project/client/src/components/AdminDashboard.jsx