Project: Milestone 1

Name: Sean Allgaier

UIN: 01138928

1. Overview

Currently, my website is a simple UI that prompts users to either log in or create an account. The user's credentials are currently being stored in a database via the back-end. The project is written with a combination of Node, React, JavaScript, HTML, and CSS. It is using the Express framework.

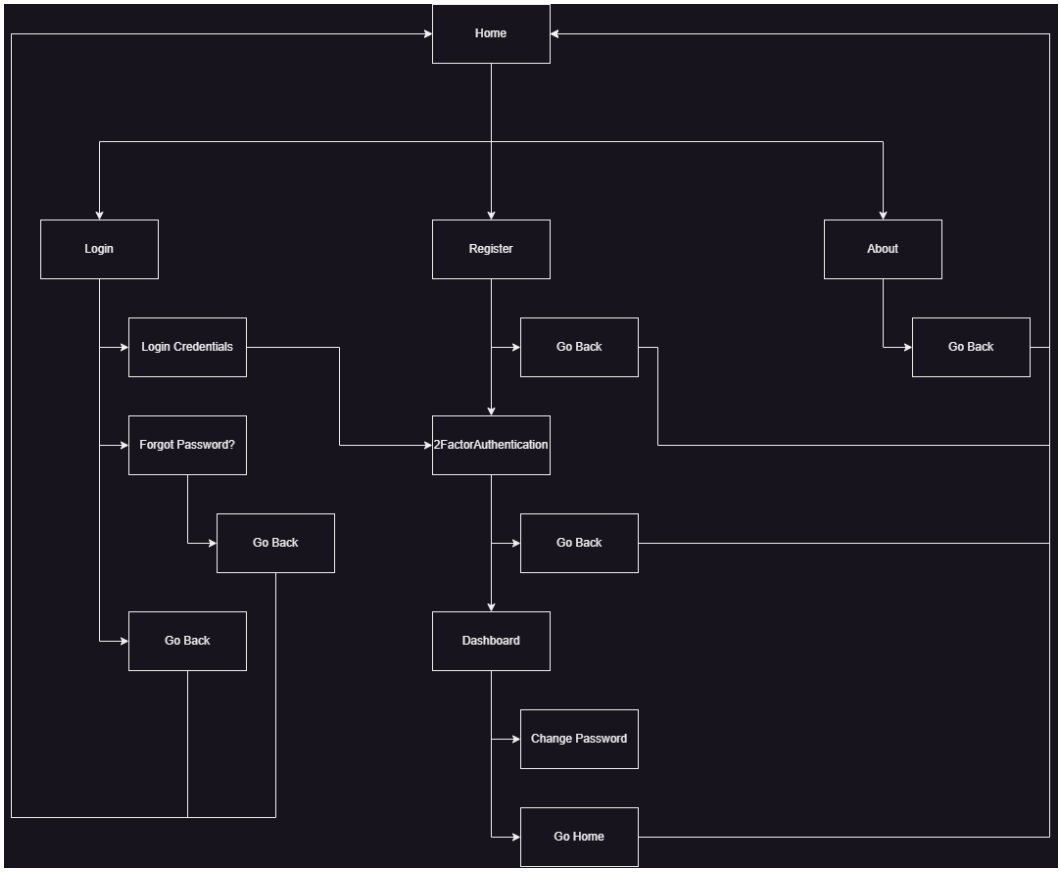
2. Milestone Accomplishments

Fulfilled	Feature #	Specification
Yes	1	Users should be able to register new accounts using email addresses.
Yes	2	Users are identified by email address.
Yes	3	Password must be encrypted before storing in the database.
Yes	4	Users cannot register duplicate accounts using the same email address.
Yes	5	The user should receive a verification email upon successful registration.
Yes	6	Users cannot log in to the system until their email has been verified.
Yes	7	Users should be able to log into your website using the accounts they registered.
Yes	8	Users should be able to reset their passwords if they forget it.
No	9	Users should be able to change their passwords after they login.
Yes	10	A 2-factor-authentication should be used when a user attempt to login. This can be done by email, phone text, or a DUO push. You can just implement **one** of them.
Yes	11	The website should have a homepage for each user, where they can view their **profiles**, **change passwords**, and **update information**. Email cannot not be changed.
Yes	12	An admin user should be created from the back-end
Yes	13	An admin user has a different view from a regular user. (Later admin will approve the submitted advising sheet by student)

3. Architecture

Like I mentioned in the overview, this project uses the Express framework and was written with a combination of Node, React, JavaScript, HTML, and CSS.

Below is a diagram that illustrates the layout of my website.



4. Database Design

Field	Type	Key	Example
user_id	bigint(20)	Primary	1
First_Name	varchar(255)		Sean
Last_Name	varchar(255)		Allgaier
Email	varchar(255)		sean.g.allgaier.99@gmail.com
Password	varchar(255)		$2b$10$V9/6uf9EYippVYHGa4J7OuBukkAq9ON4iF38oFDBT6GI72HIhVzFm
Is_Admin	tinyint(1)		1

5. Implementation

1. Register new accounts using email addresses:

- Prompts user for their first name, last name, email, desired password, and a confirmation of their desired password.
- Performs a series of checks to determine whether the user should be allowed to proceed.
    - Did the user input two matching passwords?
    - Is this a new email that does not yet exist in the database?
    - Is this a valid email?
- If the answer is yes to **ALL** questions, a variable labeled as "userStateVal" is assigned a value of false (will be further explained in the 2FactorAuthentication section), the user is sent to the 2FactorAuthentication screen, and a code is sent to the email address they provided.
- If the answer is no to any of the questions, or if the user decides to input improper information, error messages will appear on the screen.
- The front-end code is located at:
    - Client->src->components->Register.jsx
- The back-end code is located at:
    - Server->routes->user.js

2. Users are identified by email address:

- Whenever the user performs an action that does anything requiring the database, (logging in, registering an account, changing their password, etc.), the codebase will always first perform some sort of check to verify that the user's email exists in the database, then it performs the chosen action at the row in the database where the email is located.
- The front-end code is located at:
    - Client->src->components->Register.jsx
    - Client->src->components->Login.jsx
    - Client->src->components->ForgotPassword.jsx
    - Client->src->components->TwoFactorAuthentication.jsx
    - Client->src->components->Dashboard.jsx
- The back-end code is located at:
    - Server->routes->user.js

3. Password must be encrypted before storing in the database.

- When a user registers an account, their password is encrypted before being stored in the database.
- The back-end code is located at:
    - Server->routes->user.js

4. Users cannot register duplicate accounts using the same email address.

- When a user registers an account, the database checks if the email address already exists in the database.
    - If the email already exists, the user will receive an error message.
    - If the email does not yet exist, the user will be permitted to register their account.
- The front-end code is located at:
    - Client->src->components->Register.jsx
- The back-end code is located at:
    - Server->routes->user.js

5. The user should receive a verification email upon successful registration.

- Every time the user registers an account, they will receive a verification email and will not be able to proceed to their dashboard until they enter the verification code.
- The front-end code is located at:
    - Client->src->components->Register.jsx
    - Client->src->components-> TwoFactorAuthentication.jsx
- The back-end code is located at:
    - Server->routes->user.js

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

- Every time the user logs in, they will receive a verification email and will not be able to proceed to their dashboard until they enter the verification code.
- The front-end code is located at:
    - Client->src->components->Login.jsx
    - Client->src->components-> TwoFactorAuthentication.jsx
- The back-end code is located at:
    - Server->routes->user.js

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

- Prompts user for their email and password.
- Performs a series of checks to determine whether or not the user properly inputted their credentials.
    - Did the user input a valid email address?
    - Does the inputted email exist in the database?
    - Does the hashed version of the inputted password match the hashed password in the database?
- If the answer is yes to **ALL** of the questions, a variable labeled as “userStateVal” is assigned a value of true, the user is sent to the 2FactorAuthentication screen, and a verification code is sent to their email.

- If the answer is no to any of the questions, or if the user decides to input improper information, error messages will appear on the screen.
- The front-end code is located at:
    - Client->src->components->Login.jsx
- The back-end code is located at:
    - Server->routes->user.js

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

- Prompts user for their email address
- Performs a series of checks to determine whether or not the user properly inputted their credentials.
    - Did the user input a valid email address?
    - Does the inputted email exist in the database?
- If the answer is yes to **ALL** of the questions, the user's password current password is replaced by a randomly generated password that is sent to their email. The user must then use that new email the next time they log in to the website.
- If the answer is no to any of the questions, or if the user decides to input improper information, error messages will appear on the screen.
- The front-end code is located at:
    - Client->src->components-> ForgotPassword.jsx
- The back-end code is located at:
    - Server->routes->user.js

9. Users should be able to change their passwords after they login. (In progress)

- Prompts user for their email, desired new password, and a confirmation of their new password.
- Performs a series of checks to determine whether or not the user properly inputted their credentials.
    - Did the user input a valid email address?
    - Does the inputted email exist in the database?
    - Does the inputted email address match the email the user used to log in?
    - Did the user input two matching passwords?
    - Are the two new passwords different from the user's current password?
- If the answer is yes to **ALL** of the questions, the user's password current password is changed to the new password that they entered.
- If the answer is no to any of the questions, or if the user decides to input improper information, error messages will appear on the screen.
- The front-end code is located at:
    - Client->src->components-> Dashboard.jsx
- The back-end code is located at:
    - Server->routes->user.js

10. 2FactorAuthentication should be used when a user attempt to login:

- When the user reaches the 2FactorAuthentication screen, the user is prompted to enter a code that was sent to the email that they used on the registration screen or login screen.
    - If the user inputs the incorrect authentication code, output an error message.
    - If the user inputs the correct authentication code, check the value of the variable "userStateVal".
        - If the variable "userStateVal" is true, proceed to the Dashboard screen without doing anything else. (true = user is logging in)
            - When logging in, if the user inputs an email that matches the admin's email, send the user to the Admin Dashboard screen.
            - If the user inputs any other email, send the user to the regular Dashboard screen.
        - If the variable "userStateVal" is false, store the user's registration information to the database before sending the user to the Dashboard screen. (false = user is registering an account)
            - I decided to put the calls to the registration back-end here after the 2FactorAuthentication code is confirmed rather than within the registration front-end because it prevents the situation where the registration information would get stored to the database before the user confirms the verification code that was sent to their email.
            - This situation is the entire purpose of why I am using "userStateVal" to keep track of whether the user is coming from the login screen or the registration screen.
- The front-end code is located at:
    - Client->src->components->TwoFactorAuthentication.jsx
- The back-end code is located at:
    - Server->routes->user.js

11. The website should have a homepage for each user

- When a user successfully confirms their 2FactorAuthentication code when they log in or register an account, they will be sent to the Dashboard screen. The Dashboard screen displays a welcome message that changes depending on which user is logged in. It does this by outputting the user's email address.
    - Format: "*Welcome, [email]!*"
    - Example: "*Welcome, sean.g.allgaier.99@gmail.com!*"
- This is where the user will be able to change their password while they are logged in.
- The front-end code is located at:
    - Client->src->components->Dashboard.jsx
- The back-end code is located at:
    - Server->routes->user.js

12. An admin user should be created from the back-end

- The first user in the database is the admin user. It is registered under the email sean.g.allgaier.99@gmail.com.

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

- When an admin user successfully confirms their 2FactorAuthentication code when they log in or register an account, they will be sent to the Admin Dashboard screen. The Admin Dashboard screen displays a welcome message that changes depending on which admin user is logged in. It does this by outputting the admin user's email address.
    - Format: "*Welcome, [email]!*"
    - Example: "*Welcome, sean.g.allgaier.99@gmail.com!*"
- This is where the admin user will be able to change their password while they are logged in.
- The front-end code is located at:
    - Client->src->components->AdminDashboard.jsx
- The back-end code is located at:
    - Server->routes->user.js