UNIVERSITY OF NORTH TEXAS
DENTON, TEXAS

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

ALMA MINGLE

Submitted By:

Adre Sharp - 11374558
Madison Mccauley - 11509676
Ponnuru Raja Lakshmi Kamalini - 11659081
Shashank Verma - 11703352
Suhaibuddin Ahmed -11699042
Usama Bin Faheem - 11698740
Varun Mahankali - 11708358
Rahman Mehmood - 11707971

INDEX

Team Project
Deliverable 3 – Project Phase 1
CSCE 5430 (Spring 2024)

A. Requirements

List of requirements designated for phase 1

- Coding the UI components such as the
    - Dashboard
    - Landing page
    - Registration page.
- Functional requirements for
    - user profiles,
    - member management,
    - events
- Backend development - creating the necessary database and establishing connections.
- Workon the admin dashboard which will be having additional privileges
- Researching and determining implementation
    - messaging, notifications, and authentication functionalities.
    - visibility of posts within specific groups

Detailed description & explanation of the requirements of Phase 1 :

Since this was the first phase where the team jumped into coding for the application there was delay and human resource constraints due to the spring break.It was a sharp learning curve for all the team members which took some extra time and effort from the team , therefore the scope of our phase 1 has slightly changed.

Requirements :

Landing Page : The landing page has login, register and ContactUs buttons all of these correctly redirect the user to the designated pages.

- Register-
    - redirects to **registration page**
    - This component seamlessly redirects the user to the page where they can register.
- Login Button -
    - redirects to **Login page**
    - This component seamlessly redirects the user to register the dashboard of their alma mingle profile

Missing Components :

1. Info Tab - static information (may have tabs such as about us, history etc all static information)
2. Gallery -
    a. A scrolling slideshow of images (A carousel)

Explanation - The above two components are not a core functionality of our project and needed extra time to develop the static page with the university information and data , while the gallery carousel requires non copyrighted pictures that we can use , these requirements will be implemented in early part of phase 2. While we focussed on implementing the logical components like login and register we did not have enough time to gather information that could be added to the university information page.

Register(sign up):

- We have a simple form which allows the user to create their profile with the portal.
- Once the user data is entered in the form the user data is successfully pushed into the database.
- The user profile for the almamingle is created.

Explanation : All the requirements of the registration page have been met, however we can work on improving the UI and design of the page and make it more appealing.

Create Account page:
It was decided that this particular requirement was duplicated during our requirement analysis.

Explanation - This page was redundant as we already have the signup page , therefore it was decided to add the components of this page to the existing signup page and remove this page from our requirements.

Login Page :

- This page takes the sign in details namely email id and password from the user .
- Then it verifies the login credentials existing in the database.
- If the credentials are verified the user is signed in signs into the alumni portal.

Missing Component :

- Forgot Password
- Admin Login

Explanation - We have to implement the “ forgot password “ functionality which will allow users to update their password and the admin login functionality with special privileges . We have to implement the logic for updating the password by the user in the backend. We encountered a minor bug while updating the password and enabling admin login user story .This functionality will be ready by phase 2.

Alumni User: These are the registered users of Almamingle who can access all our services.

Dashboard :

- This is our homepage where all the services can be accessed.
- The dashboard also has a logout button, this revokes all the access the user has and redirects the user to the login page.

Missing Components :

Burger Menu which redirects to University information page user

Explanation - Burger menu with the University information tabs ; since the requirement of landing page where the options in the burger menu redirects, is not ready it has been decided to implement this in phase 2.

Navigation Menu :

- Currently it has the create post ,view profile and edit profile services.
- When the user presses these buttons the user is sent to the designated services .

Missing Components :

- Member Management
- Events

Explanation - Functional requirements not met in this phase : it has most of the services we decided to implement for this phase except member management and events tab. The code for these components is ready however it was decided to merge with the main repository in phase 2 due to lack of time. The UI of the navigation menu will be enhanced in phase 2.

Post Box :

- The create post button in the navigation menu opens the create post dialog box.
- Here the user can give a title , body and category to the post and post it.
- This content is then persisted into the database.

Explanation - All the functional requirements of the post box functionality are met for this phase , we need to finalise the UI for the post box after review by some users. This feedback will be incorporated in phase 2.

Posts :

The users can view the post after clicking on the posts button in the top menu of the dashboard , this fetches all the posts created by users and displays it to the user.

Missing Components :

- Post Update
- Post Delete
- Comment
- Like

Explanation of functional requirements not met in this phase - Post update , delete ,comment and like of the post are pending . While we encountered some errors while updating and deleting the posts due to mistakes we made in our code logic , the other functionalities required a more in depth and clean way to implement in the code. To prevent any major code discrepancies while implementing these essential functionalities we have decided to implement them in phase 2.

Filter & footer menu : Missing Entire component

Explanation - These small UI components were ignored due to oversight on part of the team , hence we have decided to categorise the requirements more clearly and designate two team members to verify all the implementation in the next phase

Edit Profile - Missing Entire component

Explanation - We have encountered the same code logic error where the updating profile was not being persisted to the database , it showed multiple profiles, therefore we decided to get this functionality in place in phase 2.

Admin User: The admin will login same as the normal user and have the same UI as a general user but will be able to perform some special functions

Admin User : Missing this user story

Explanation - This is something we are still working on , we only have to implement a separate dedicated login user story for admin and integrate the alumni use cases to the admin, while incorporating special privileges in phase 2 of the project.

Overall, the functionalities mentioned above had a common theme of updating the existing data and persisting it to the backend , therefore all the components that required such a function have been delayed due to the error we encountered where the updates led to duplicating of the existing data. We have decided to tackle this issue first in the phase 2 of our development process.Once this is implemented a bulk of our functional components will be ready by phase 2.

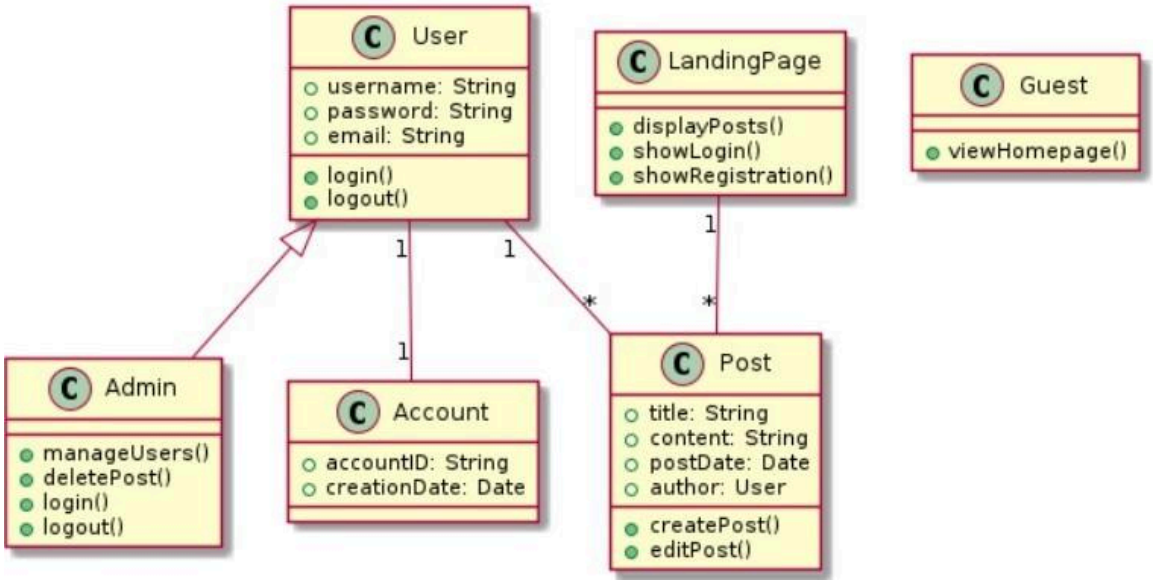
Updated Phase Plan Reasoning : While we still hope that the phase three plan remains intact we have made some major modifications to our phase two .We have decided to break down the functional requirements in more detail and give a chronological phase plan for phase 2 so that the requirements are implemented more systematically.

1.	Phase 1 pending functional requirements implementation & Code Enhancement and improvement	In Phase 2, Missing requirements to be prioritised in phase 2: 1. Work on the errors encountered in updating the database when a user is updating the existing data. 2. Once 1. Is completed we implement ○ Update password ○ Update Profile ○ Post update , ○ Post delete ○ Post Comment Post ○ Filter posts 3. Admin User Story - with all functional requirements ○ Post Broadcast ○ All Posts Delete 4. UI Enhancement of all pages 5. Additional UI components ○ Burger menu on dashboard ○ Information tab page for the landing page ○ Gallery on Landing page ○ Footer on all pages 6. Member management on navigation bar 7. Events on navigation bar Phase 3 New Enhancements: 8. Messages	April 7 2024 Deliverable 4

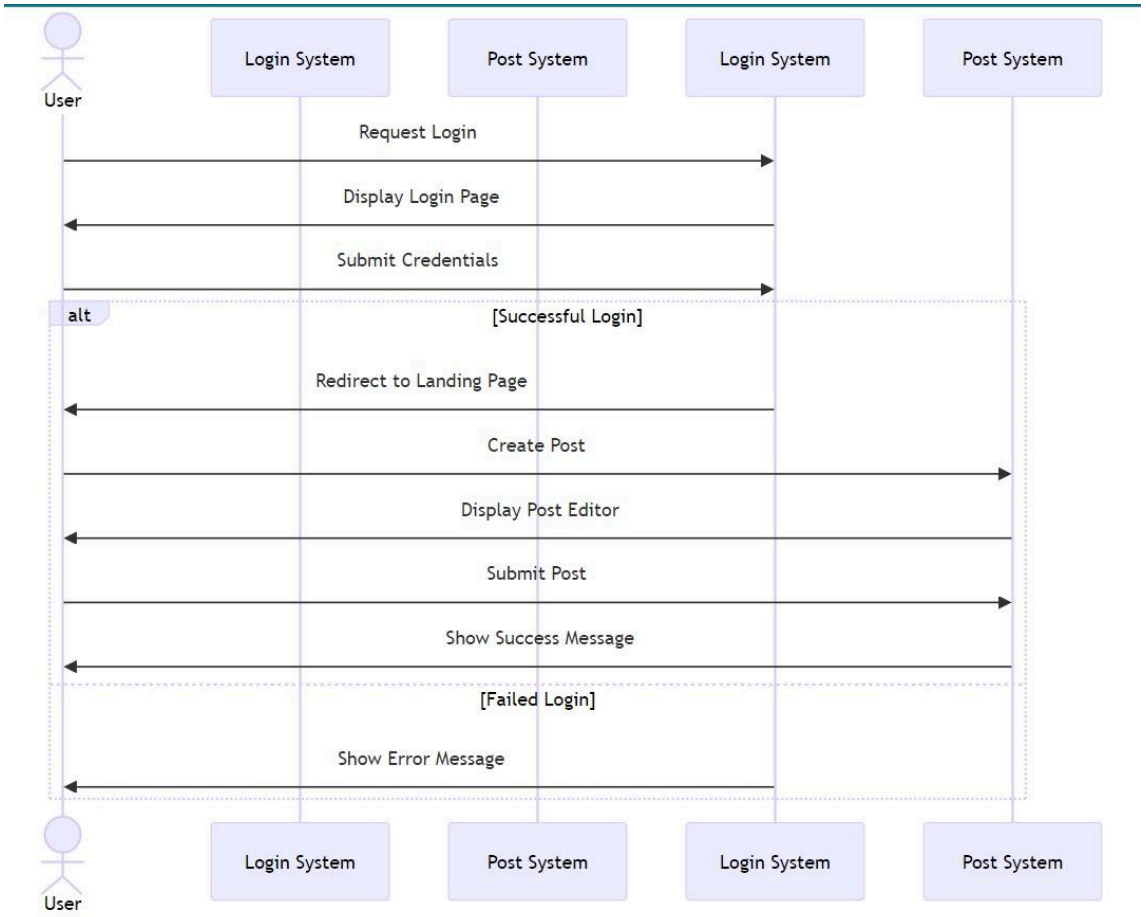
		9. Notifications 10. User Authentication 11. Post Visibility 12. Continue code development for remaining functionalities researched in Phase 1. 13. Develop test case scenarios to evaluate the website and fix identified bugs, we will simultaneously test the functionalities of both the user dashboard and the admin dashboard. 14. Criticality: It includes successful implementation of Messages and notifications, along with identifying and implementing appropriate test case scenarios for comprehensive website testing.	
2.	Phase 1 pending functional requirements implementation & Finalise the code and Website Launch	In Phase 3, 1. Make sure the issues that crept up in phase 1 and phase 2 are thoroughly avoided. 2. Complete the pending requirements from phase 1 and phase 2 3. Review and make final adjustments to the website code before freezing it. 4. Initiate User Acceptance Testing (UAT) and address any issues identified during this phase. 5. Proceed with the deployment process, addressing various deployment-related tasks. 6. Once all tasks are completed, the website will go live and be ready for use. Criticality: Handling different kind Deployment issues and responsiveness of the website.	April 28 2024 Deliverable 5

B. UML Diagrams

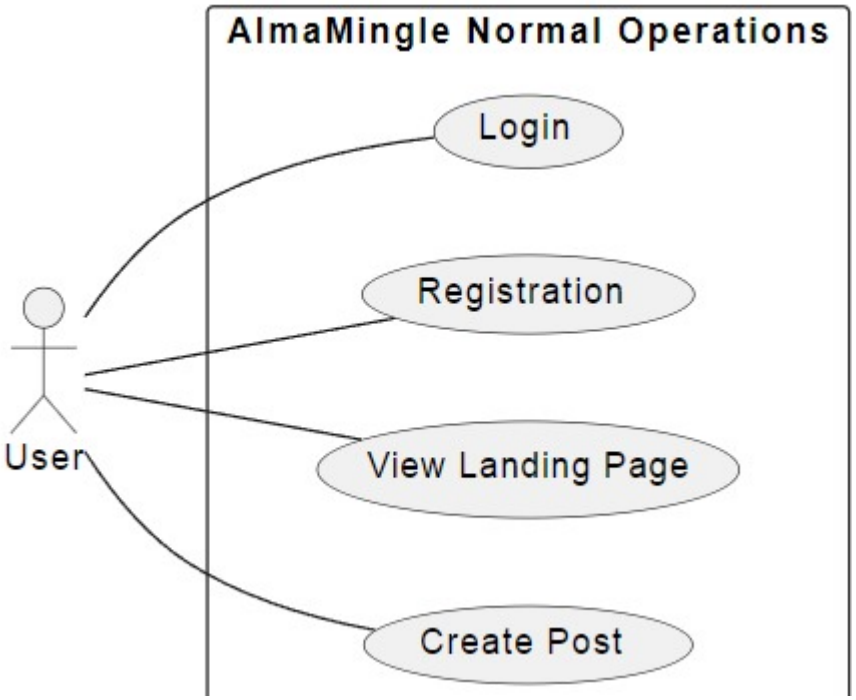
i) Class Diagram



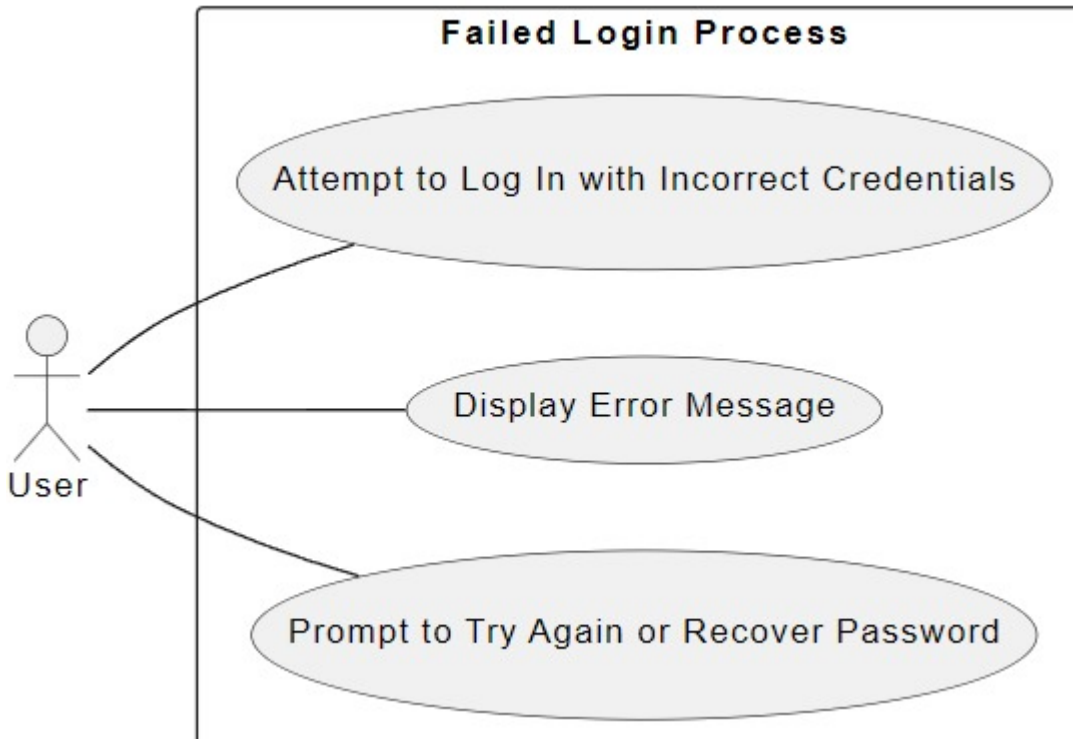
ii) Sequence Diagram



iii) UseCase Diagram



a) Login Success Scenario



b) Login Failure Scenario

C. Test Cases (unit tests) for phase 1

List a set of test cases used for testing the working program including descriptions of tests (e.g., what functionality they test, and inputs/outputs for them).

TEST CASE 1:

Fig 3.C.1

This is a test suite written using the Jest testing framework and the React Testing Library to test the `Home` component of a React application.

- The `describe` function is used to group related test cases together. In this case, it's describing the testing suite for the `Home` component.
- Inside the `describe` block, there's a single test case defined using the `test` function. This test case is checking whether the `Home` component renders with a small description.
- Within the test case:

  - The `render` function from React Testing Library is used to render the `Home` component.

  - The `getByAltText` function is used to find the image rendered in the `Home` component based on its alt text.

  - The `getByText` function is used to find elements containing the text "Welcome to AlmaMingle: Connect, Learn, Thrive!" case-insensitively.

- Assertions are then made using Jest's `expect` function:
- It ensures that the image is present in the component and its source contains 'main_pic.png'. It checks that the small description text is rendered in the component.

Overall, this test suite verifies that the `Home` component renders as expected with the specified content and image.

Input: This will take the “Home.test.js” file as input and give the output that is present in the screenshot. In order to run the file we need to give the following command.

%npm test

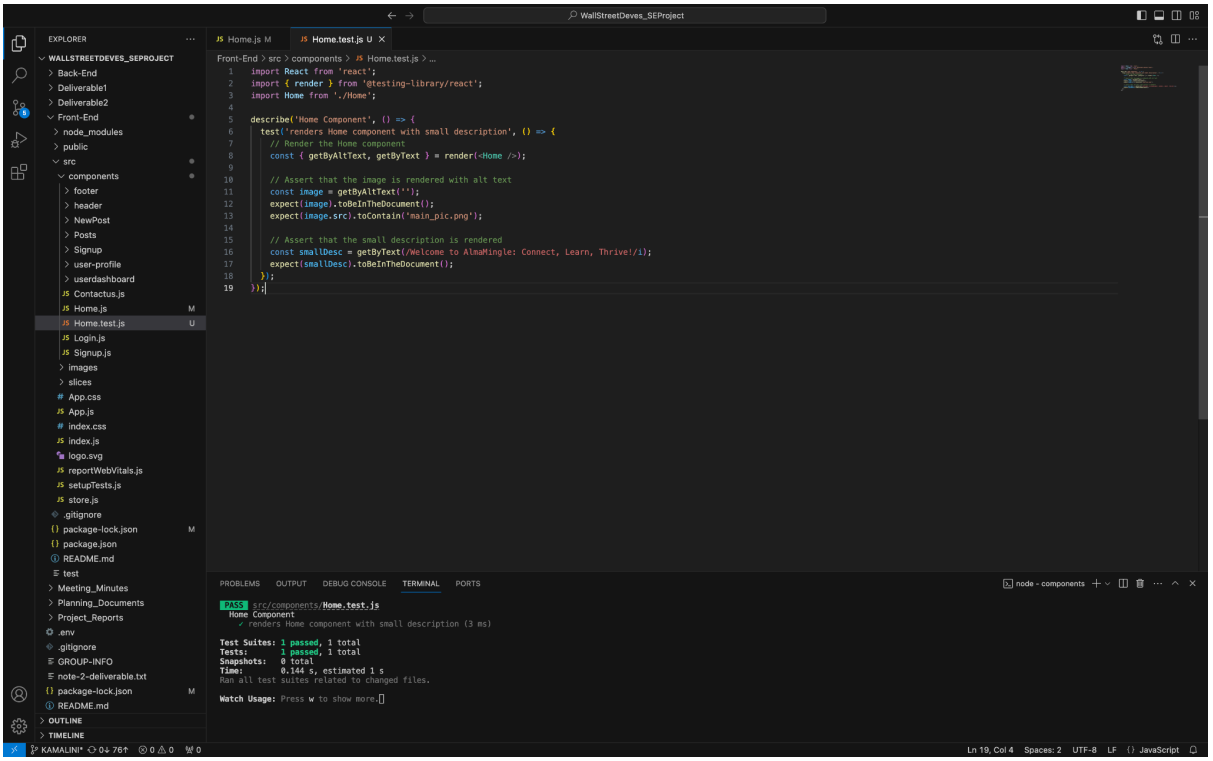


Fig 3.C.2

Output:

The test suite for the `Home` component has been executed successfully, with all test cases passing. It confirms that the `Home` component renders as expected, displaying both the image and the small description as specified in the test. The output indicates that there is one test suite, containing one test case, which has passed without any failures. Additionally, no snapshots were generated during the test execution. The total execution time for the test suite is approximately 0.144 seconds. This output is typical of a successful test run, indicating that the component behaves as intended and meets the specified requirements.

TEST CASE 2:

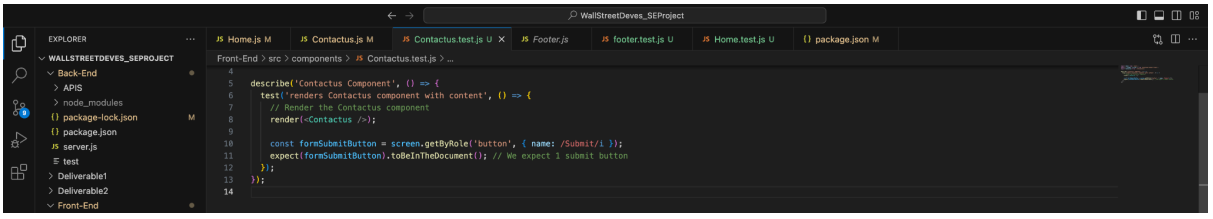


Fig 3.C.3

This test code ensures that the `Contactus` component renders correctly and contains a submit button for the contact form. It utilizes the `render` and `screen` utilities from `@testing-library/react` to render the component and perform assertions.

The `getByRole` query is used to find the submit button element based on its role as a button and its accessible name (label) matching the text "Submit". Finally, it asserts that the submit button is present in the rendered component.

Input: This will take the “Contactus.test.js” file as input and give the output that is present in the screenshot. In order to run the file we need to give the following command.
%npm test

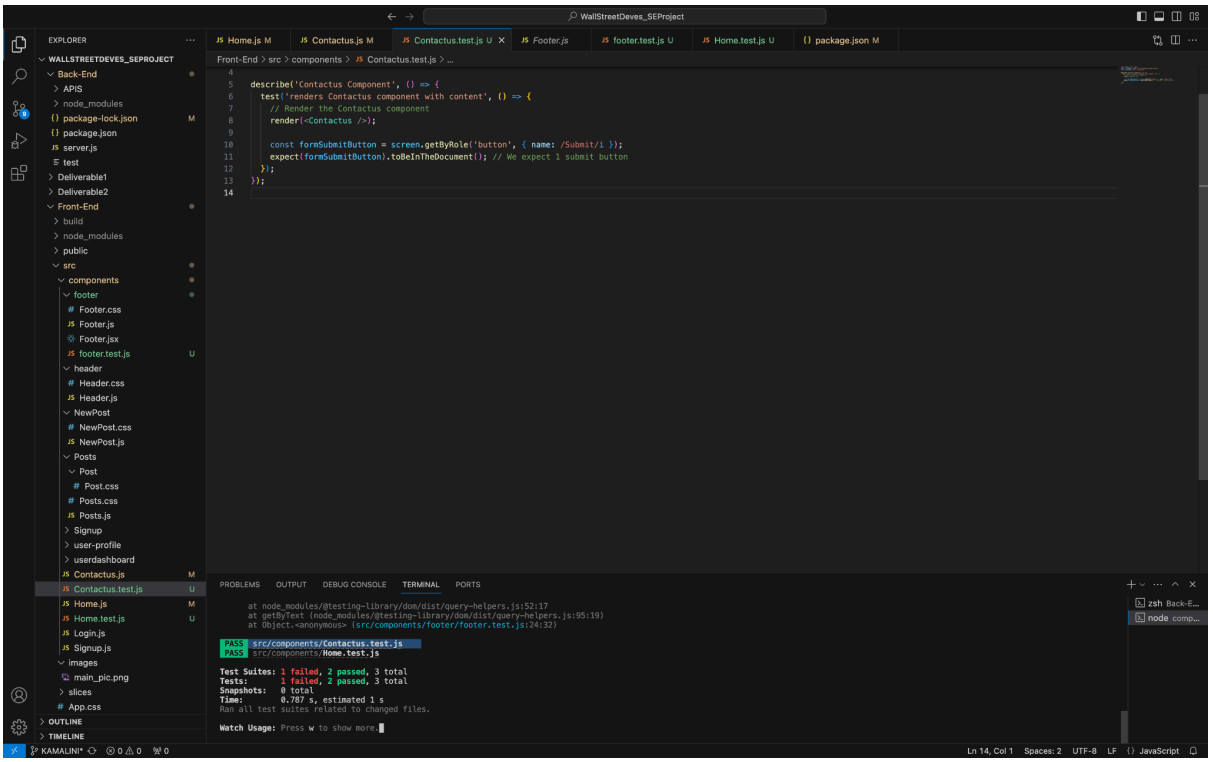


Fig 3.C.4

Output: This output indicates that the test suite for the `Contactus` component has passed successfully. The test case named "renders contact information correctly" within the suite passed without any failures. The test suite executed one test, which verified that the `Contactus` component renders the contact information correctly, including the heading "Contact Us," and there exists a submit button. Since the test passed, it confirms that the `Contactus` component behaves as expected and renders the contact information accurately.

<u>TEST CASE 3:(FAIL EXAMPLE)</u>
This test code ensures that the `Footer` component renders correctly by verifying the presence of specific text content and links within the footer. It uses the `render` function from `@testing-library/react` to render the `Footer` component. Then, it checks for the presence of the following elements:

1. Year text: Verifies that the footer contains the text "@2024".
2. University text: Ensures that the footer contains the text "University of North Texas".
3. Rights text: Checks that the footer contains the text "All Rights Reserved".

4. Terms of Use link: Asserts that the footer contains a link with the text "Terms of Use".
5. Privacy link: Verifies that the footer contains a link with the text "Privacy".

By testing these elements, the code ensures that the footer component is rendering the expected content and links, providing confidence in its correctness and completeness.

Input: This will take the “Footer.test.js” file as input and give the output that is present in the screenshot. In order to run the file we need to give the following command.
%npm test

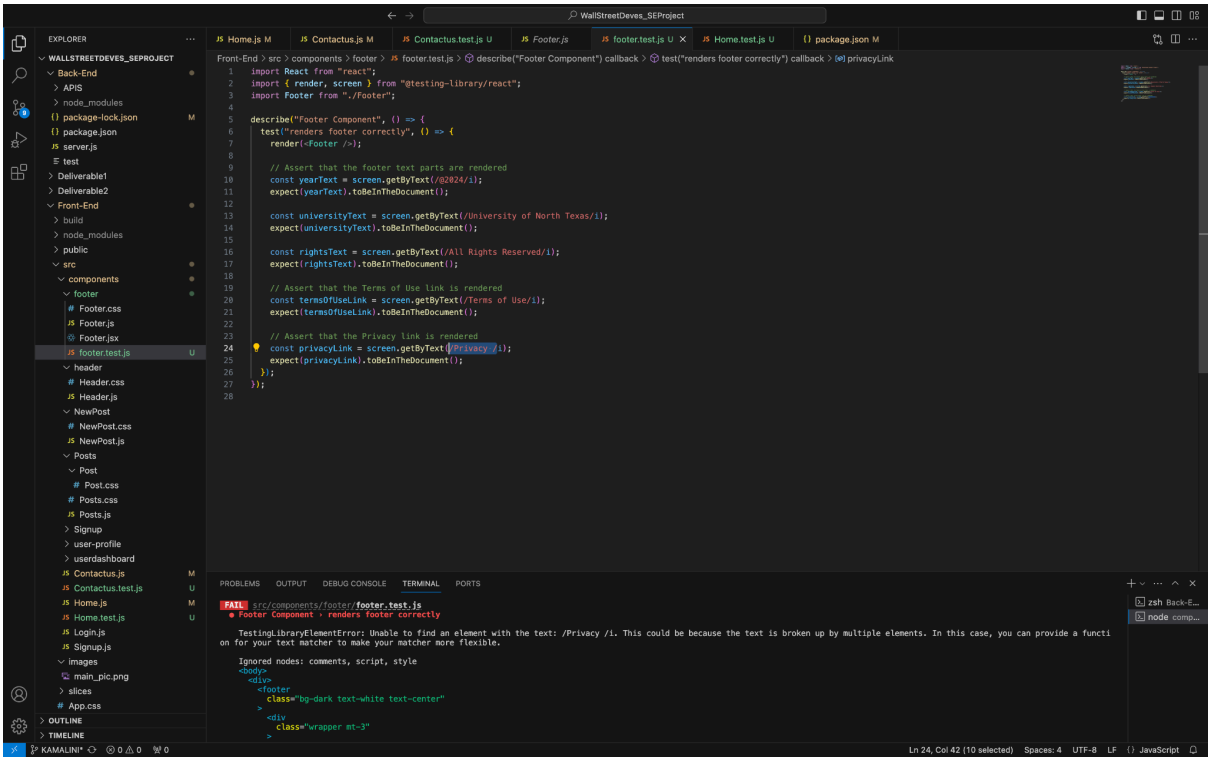


Fig 3.C.5

Output:
We got the right output as failed because we need to have “Privacy “ instead of “Privacy” since there is a difference in space. This test case is failing.

TEST CASE 4/5:

Input:
Both of these cases are testing the initialised app to make sure that there is a blank slate for the code to run. These tests will run on the response and body of the initial HTML to make sure they exist so that when a request is made and JS code is run it will be able to get started.

```
var request = require("request");
var base_url = "http://localhost:3000/"

Run | Debug | Show in Test Explorer
describe("SWE Project", function() {
  Run | Debug | Show in Test Explorer
  describe("GET /", function() {
    Run | Debug | Show in Test Explorer
    it("blank page loads", function(done) {
      request.get(base_url, function(error, response, body) {
        expect(body).toBeUndefined();
        done();
      });
    });

    Run | Debug | Show in Test Explorer
    it("check response", function(done) {
      request.get(base_url, function(error, response, body) {
        expect(response).toBeUndefined();
        done();
      });
```

Fig 3.C.6

Output:

```
Randomized with seed 95985
Started
..


2 specs, 0 failures
Finished in 0.181 seconds
Randomized with seed 95985 (jasmine --random=true --seed=95985)
```

Fig 3.C.7

Both test cases passed and should always pass moving forward as this is an initial ping to the app.

TEST CASE 6 (Fail example):

Input:

This is a test case that can always run to test deployment to a host. This is checking the response is HTTP code 200 meaning that the request has succeeded.

```
var request = require("request");
var base_url = "http://localhost:3000/"

Run | Debug | Show in Test Explorer
describe("SWE Project", function() {
  Run | Debug | Show in Test Explorer
  describe("GET /", function() {
    Run | Debug | Show in Test Explorer
    it("blank page loads", function(done) {
    });

    Run | Debug | Show in Test Explorer
    it("check response", function(done) {
    });

    it("returns status code 200", function(done) {
      request.get(base_url, function(error, response, body) {
        expect(response.statusCode).toBe(200);
        done();
      });
    });
```

Fig 3.C.8

Output:

```
  Message:
    Error: Timeout - Async function did not complete within 5000ms (set by jasmine.DEFAULT_TIMEOUT_INTERVAL)
  Stack:
        at <Jasmine>
        at listOnTimeout (node:internal/timers:573:17)
        at process.processTimers (node:internal/timers:514:7)

1 spec, 1 failure
Finished in 5.028 seconds
Randomized with seed 33773 (jasmine --random=true --seed=33773)
PS C:\Users\Andre\Documents\GitHub\WallStreetDeves_SEProject>
```

Fig 3.C.9

The website is not deployed to a host so the response will fail for now. This test case will be very useful in the final phases of the project as a utility function as it can immediately check the request and help diagnose hosting issues if they arise.

D. Manual, Step wise explanation of our application.

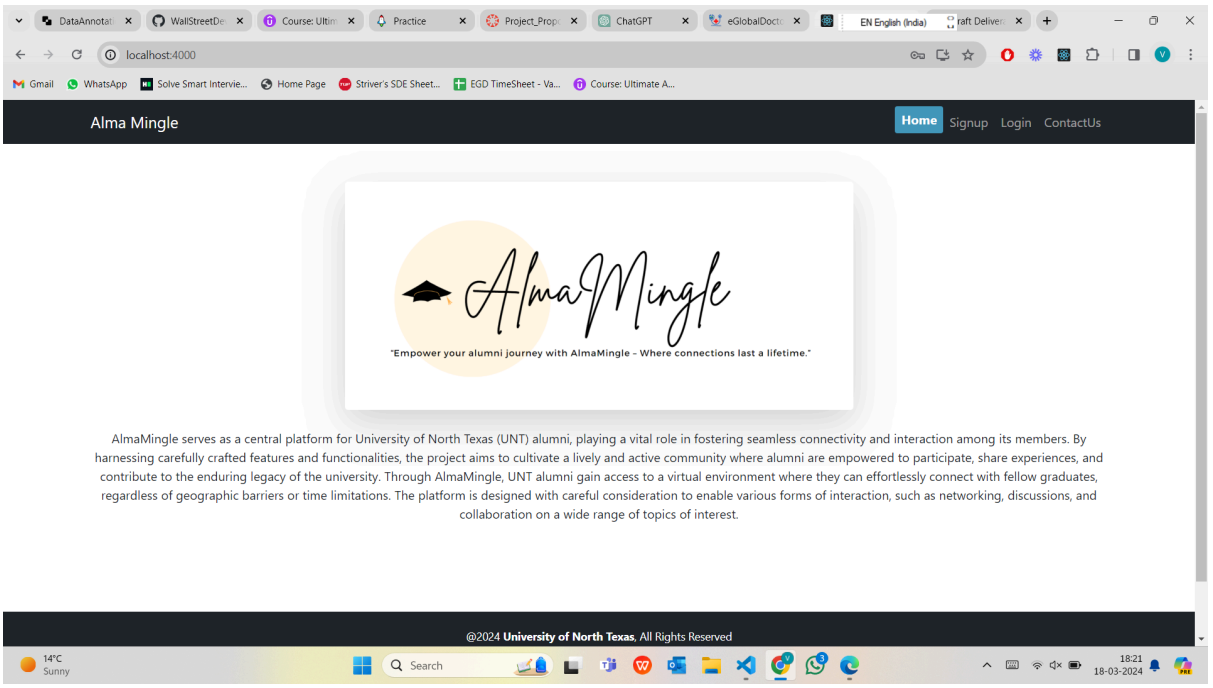


Fig 3.D.1

Fig 3.D.1 is the home screen of our application. It consists of our application logo and content describing our application. It consists of a Home button, Signup button, Login button and ContactUs button on the top of the navbar. The above screenshot is the Home screen. When clicked on the SignUp button, it will navigate to the SignUp page, when clicked on the Login button, it will navigate to Login Page and when clicked on Contact Us button, it will navigate to ContactUs page.

The footer at the bottom of each screen displays the application's privileges.

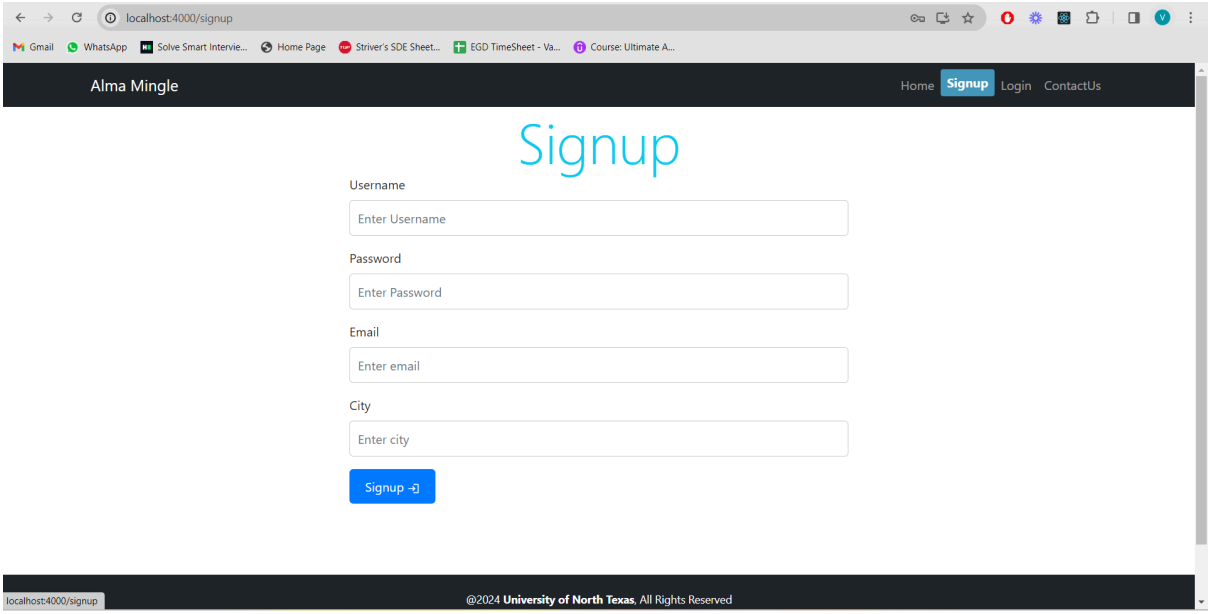


Fig 3.D.2

Figure 3.D.2 shows the Signup page. It consists of form with the Username, Password, Email and City fields. The user can create their account by entering the information in the fields and then clicking on the SignUp button that will create the user account.

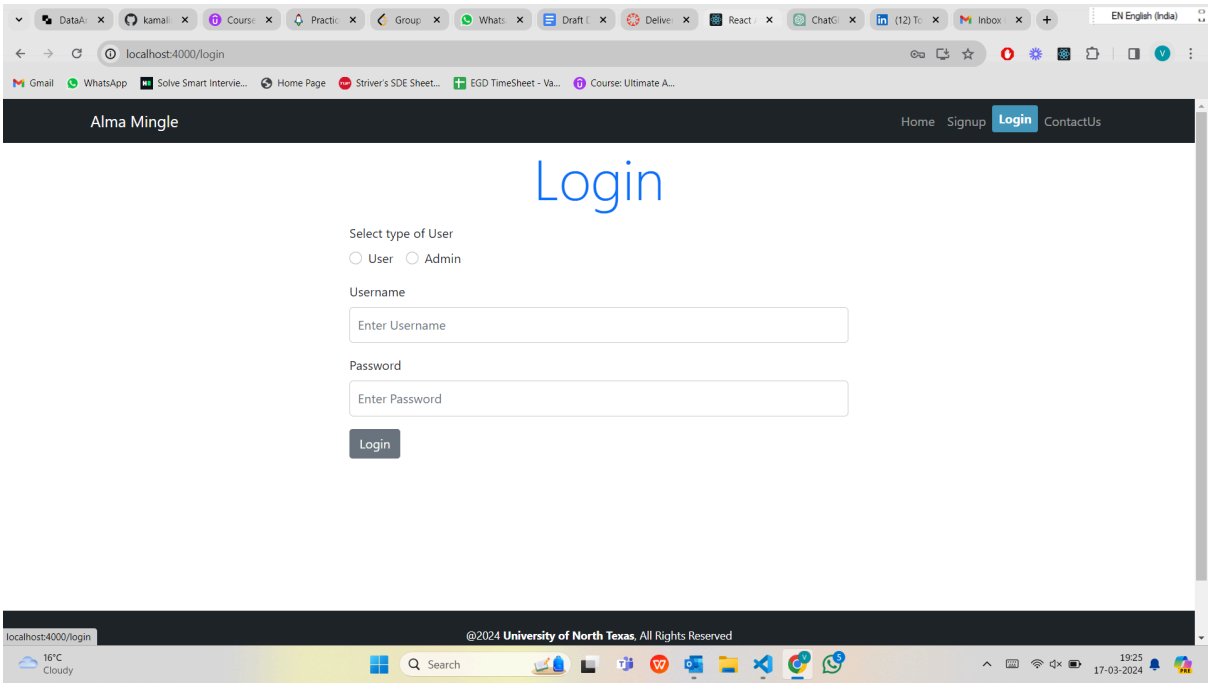


Fig 3.D.3

Fig 3.D.3 shows the login page, where the user and admin can login by entering the credentials. When the Login button is clicked, the screen will navigate to the User Dashboard screen.

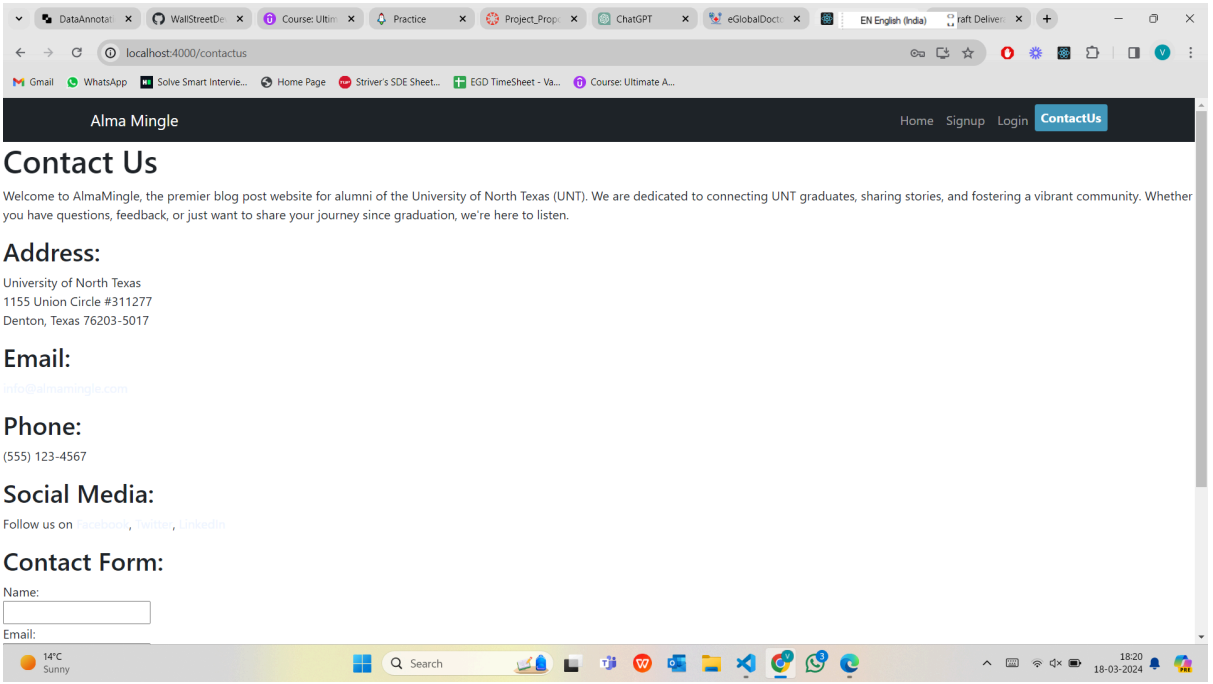


Fig 3.D.4

Fig 3.D.4 shows the Contact Us page, showing the details to contact the application admin. This can be more updated in Phase 2.

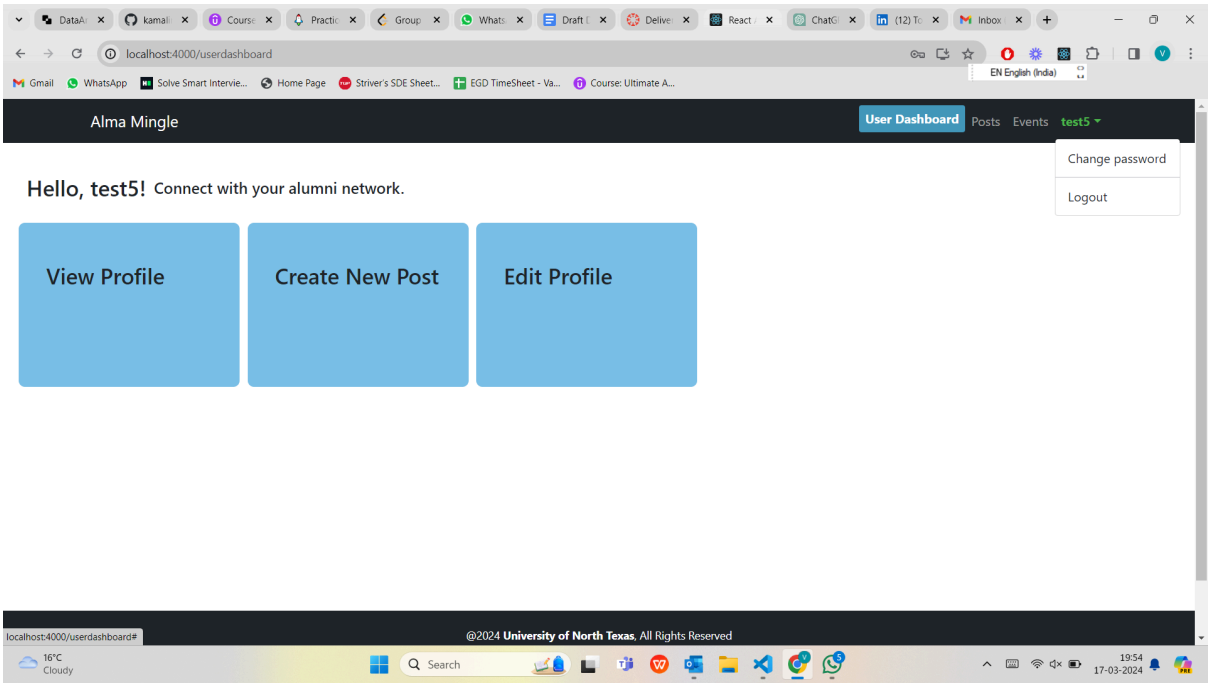


Fig 3.D.5

Figure 3.D.5 shows the User Dashboard screen. In the navbar the buttons are changed. The Posts button, Events button, and the dropdown of the user's name consist of Change Password and Logout options. When the Logout button is clicked, it will navigate to the Login page as shown in figure 3.D.3 will be shown to the user.

On the screen the user can see the welcome message to the user and then three cards below of the message. View Profile, Create New Post and Edit Profile are three cards.

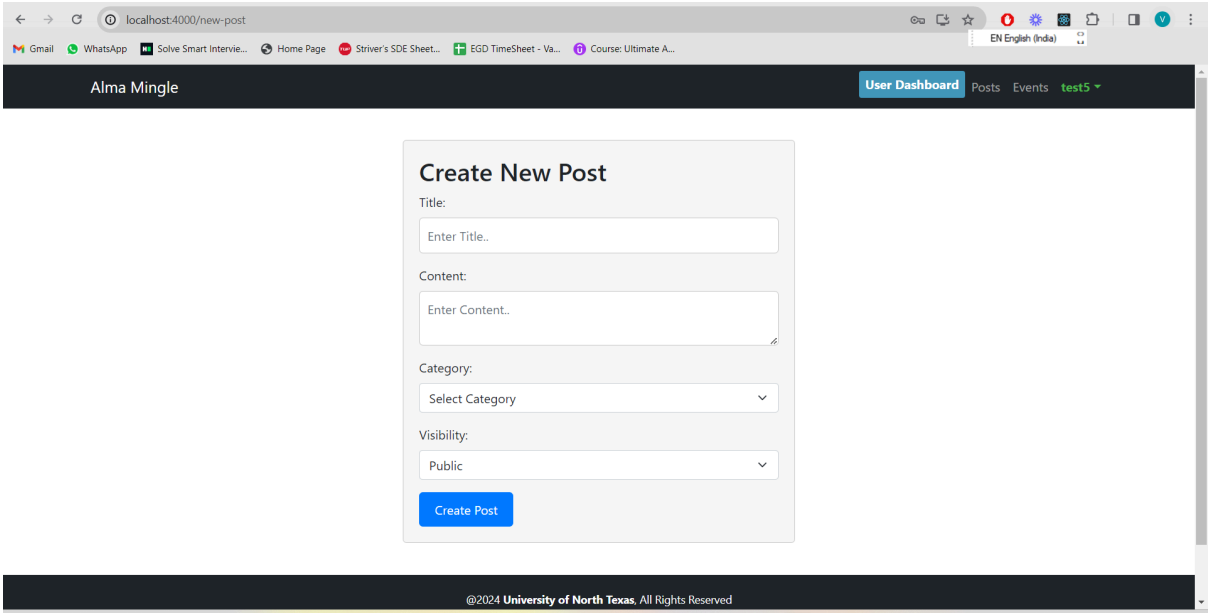


Fig 3.D.6

When the Create New Post card is clicked, the screen in Figure 3.D.6 is shown. The user can create a post by entering the Title, Content, Company, and Visibility fields. When the Create Post button is clicked, the post is created.

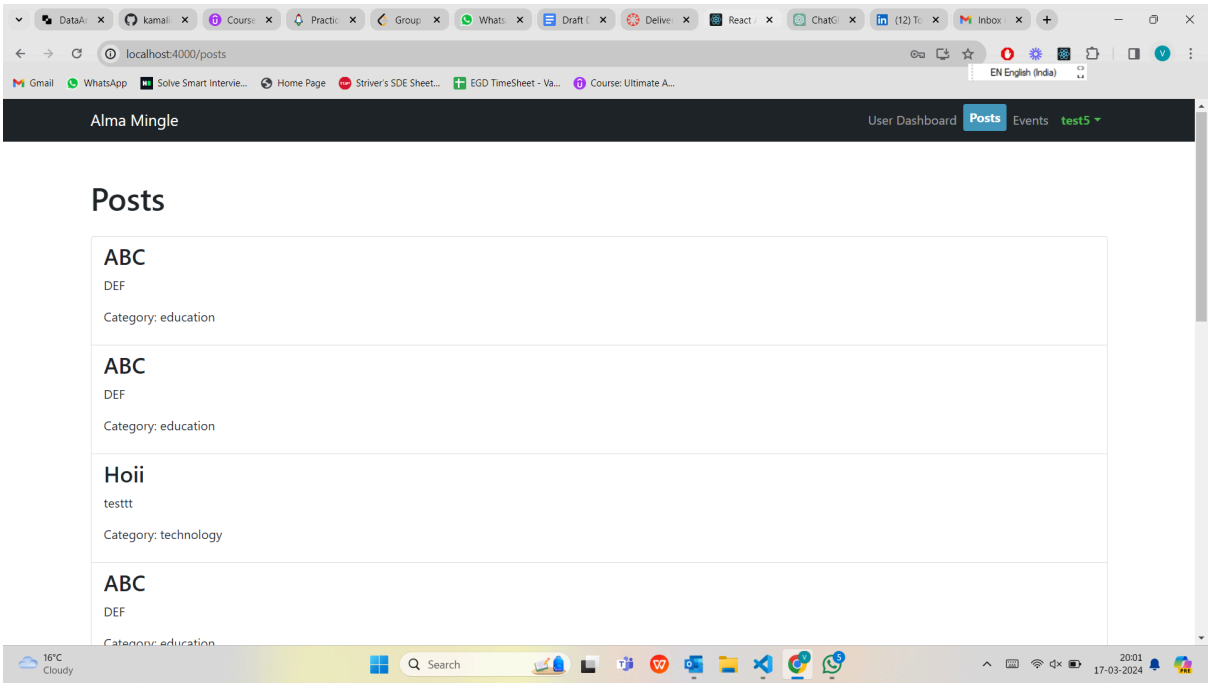


Fig 3.D.7

In figure 3.D.7, the Posts Screen is shown. It shows all the posts posted by all the users. In above I have entered sample data for better understanding of the functionality.

The Events screen, View Profile Card, Edit Profile Card and some more other features can be updated in the Phase 2 of the coding. More changes can be done if there are any to the UI screens.

E. Instructions on how to Compile/Run the program.

Initially after cloning the repository, the main code lies in the Back-End folder as it takes the build folder from the Front-End side. The build folder consists of all the files that are required to run or present in the Front-End folder. Inorder to run the code, follow below steps.

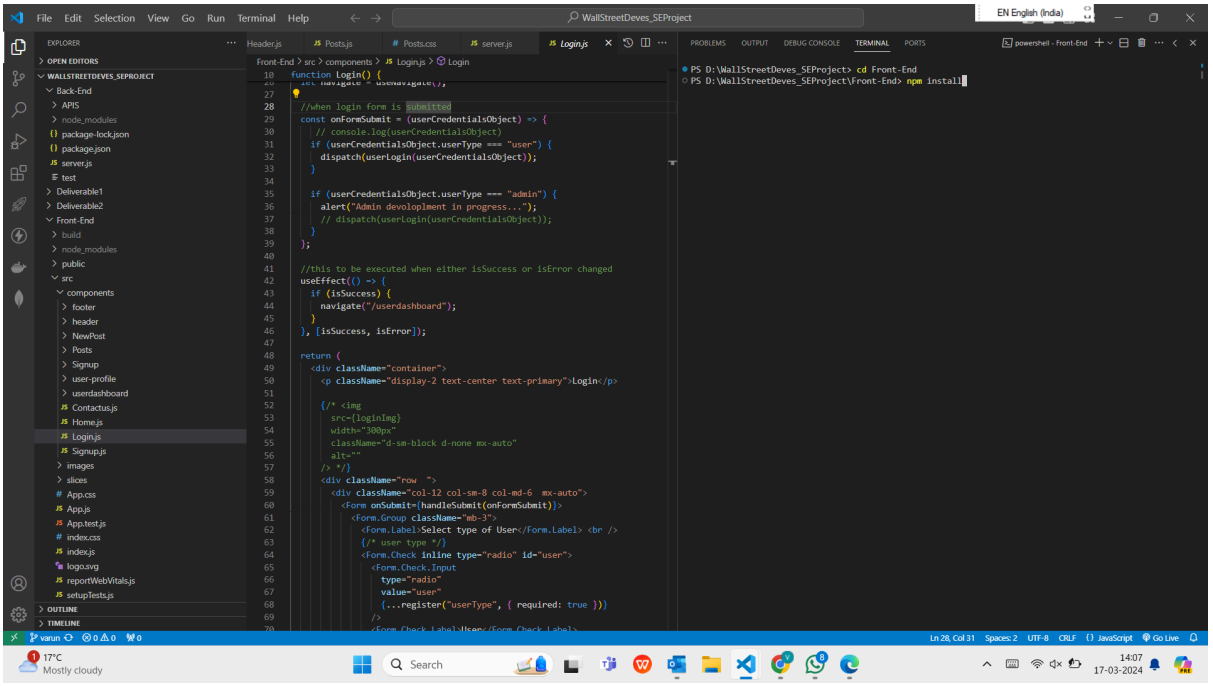


Fig 3.E.1

In above Figure 3.E.1, navigate to the Front-End folder and run the command “npm install” as shown above. This will install all the required packages.

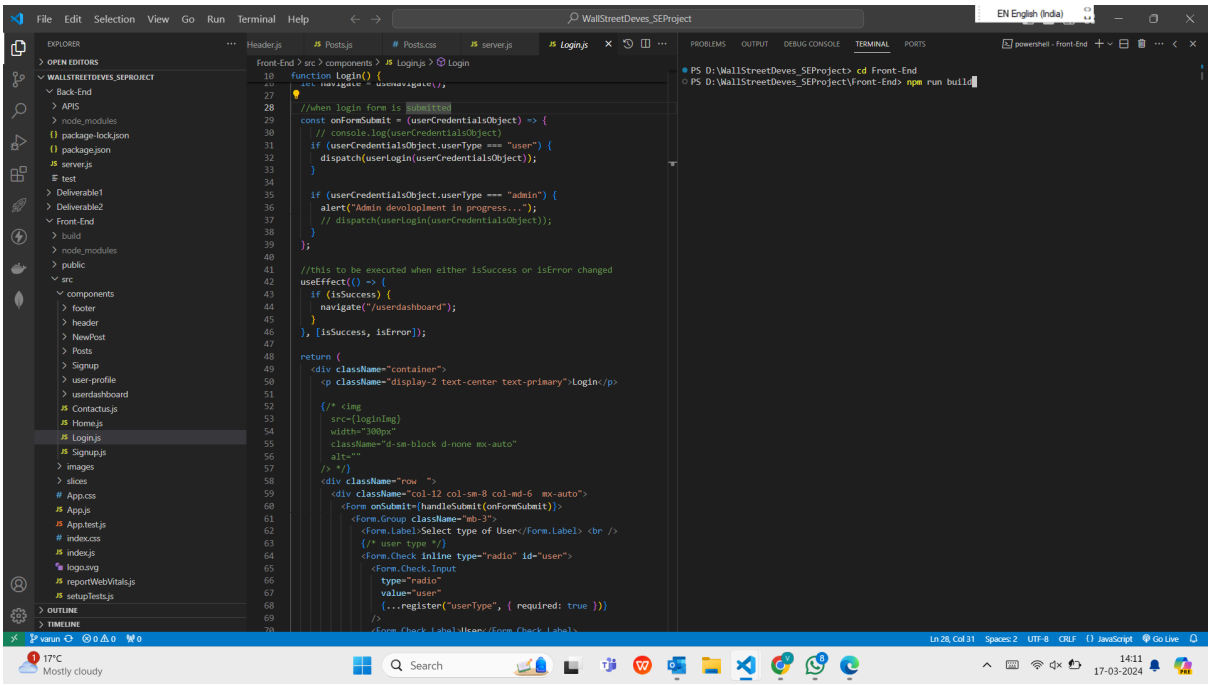


Fig 3.E.2

Now as shown in Figure 3.E.2, run the command “npm run build” to build all the files.

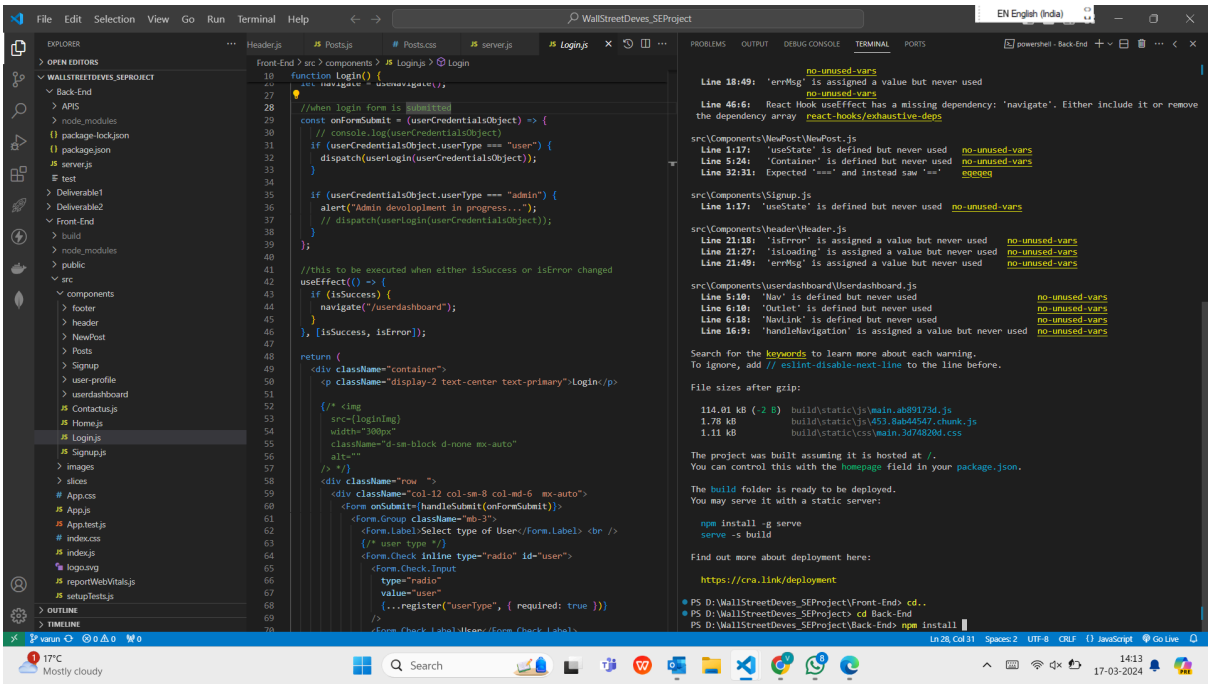


Fig 3.E.3

Now Navigate to the Back-End folder and run the command “npm install” to run the packages in the backend side as shown in the above figure 3.E.3.

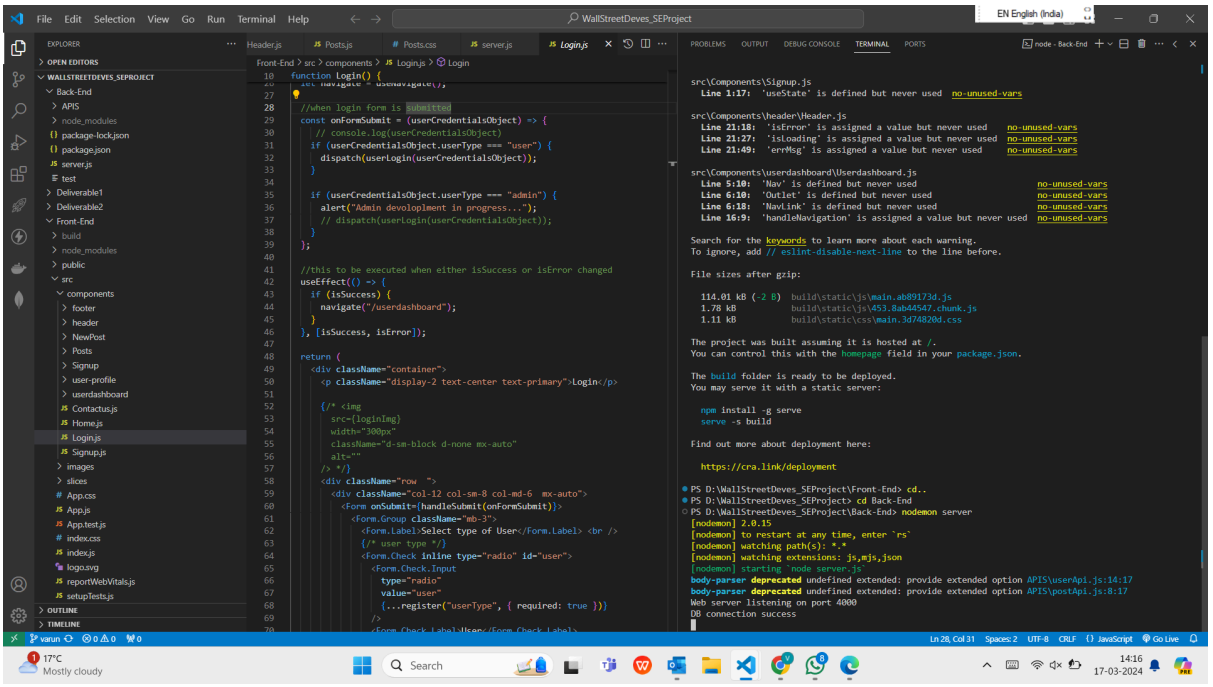


Fig 3.E.4

Now as shown in Figure 3.E.4 run the command “nodemon server” and the connection success message will be shown. Now the user can open the link “http://localhost:4000/” in the browser and can start creating the account and login and adding the posts the user requires.

Instructions on how to Compile/Run the test cases.

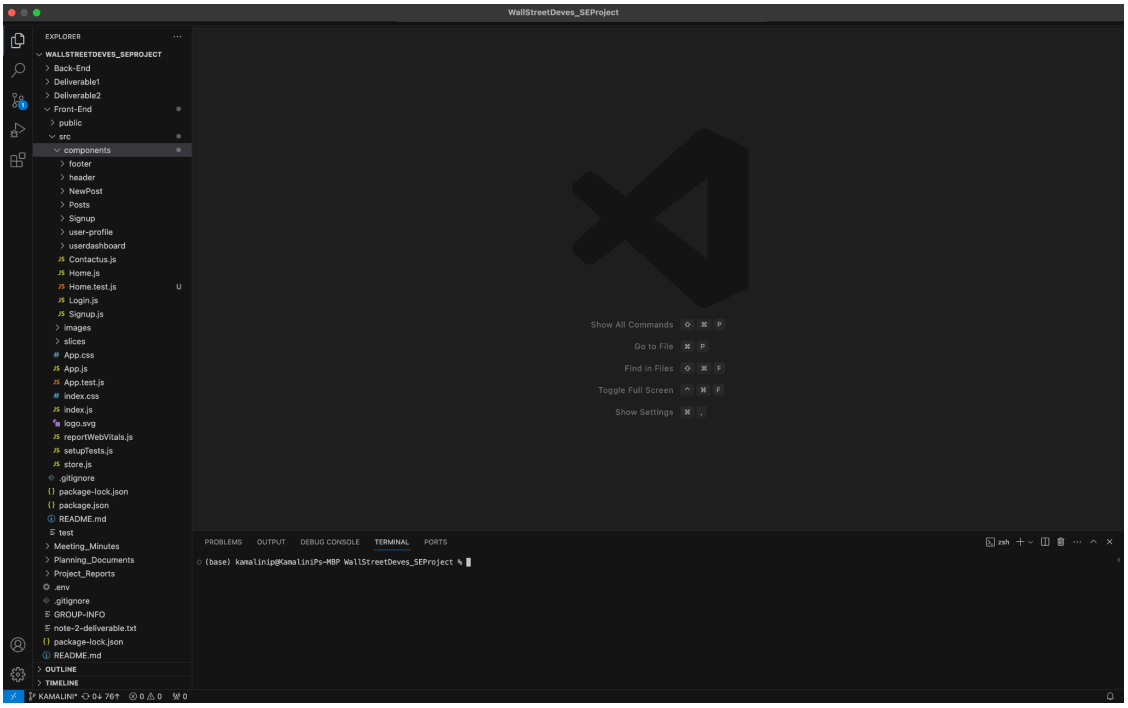


Fig 3.E.5

In order to run the test cases we need to clone the repository from GitHub using Visual Studio Code. After following the content from image 3.E.1, we can open the terminal and then we can give command
% npm test
Before this command, we need to be in the right for that
% cd Front-end/src/components/
And we need to give %npm install

%npm test – This command will automatically run the files named “*.test.js” and provides the respected output in the terminal. Here, we have used jasmine and jest to test the java script code that is present in the src code. The test code files are present in the Front-End/src/components folder.

F. Peer review:

The peer review had during class was with TECH EXPLORERS and Tiny Coders
During the peer review session, valuable feedback was provided regarding several key aspects of our project implementation. Suggestions from other teams included queries about message storage, student verification processes, and registration for old students without access to their UNT student IDs.

To address these concerns, we have initiated a research phase to determine the best approach for implementing message functionality and selecting an appropriate database for storage. It was decided that messages will not disappear after 24 hours. Additionally, for student verification, we are exploring options such as allowing only @my.unt.edu email addresses for login and considering third-party authentication methods.

Regarding registration for old students without access to their UNT student IDs, we are still in the process of finding the most effective solution. Proposed ideas include linking Gmail IDs with UNT email addresses or implementing manual verification by the college administration.

Overall, these feedback points have prompted us to delve deeper into certain aspects of our project implementation, ensuring that we address potential challenges and enhance the functionality of our platform.

G. Reflection

We have successfully implemented crucial features such as the home page, login setup, user signup page and user dashboard, which are the main components of Phase 1. These features were implemented by utilising both front-end and back-end coding and database as well. The front end was executed using 'npm install', while the back end utilised 'npm install' for setup. Once connected, users can access, create the account if they don’t have one and create a login successfully. We have also implemented user authentication successfully.

Moving forward to the next deliverable, improvements can be made to enhance the quality and performance of our platform. While the 'Contact Us' functionality has been created, it requires further refinement to be fully accessible and established on the platform. Similarly, the ability to create posts has been implemented, but users will benefit from more control over creation and better visual representation on the site.

Additionally, the dashboard requires improvement to provide users with more detailed information. Currently, it lacks sufficient detail and appears relatively empty, but we aim to enhance its functionality and visual appeal in future iterations.

While working on the testing section, we identified different test case scenarios that can be implemented for the next testing phase. We are also working on different scenarios where we will be testing if any field in the signup page or login page is empty or not. Whether the given password contains the regex expression or not added with it has a length of 8 - 15 characters or not. In the signup page ‘@universitydomain is present or not.. And many more incidents that are related to the code. Since everyone in the team is new to the testing process, we are facing few difficulties in this section. Our goal is to implement a few more valid test cases as explained in the previous section by the end of the next deployment phase.

H. Member Contribution Table

MEMBER CONTRIBUTION TABLE

S.No	Member Name	Contribution	Overall Contribution (%)
1	Kamalini Ponnuru	Worked collaboratively with Andre on the test cases part and developed a few test cases. Learned the basics of jest and jasmine test framework. Divided the tasks among the team members and attended the meetings and provided the suggestions whenever and wherever it is required. Reviewed the final draft. Added the note-3-deliverable.txt file in GitHub repository.Checked if everything is present in the GitHub repository & report or not.	12.5%
2	Andre Sharp	Developed an initial test case suite using Jasmine and learned how to configure the framework for the length of the project. Worked on additional utility tests for the backend. Added additional dual configuration option to accept Typescript in case it is utilised further in development.	12.5%
3	Madi McCauley	helped with dividing tasks among the team, discussion about the implementation of this design phase, technical background, helped coordinate meetings and kept track of participants and our contributions, peer review, accomplishments and improvements needed according to this development phase, report formatting and final draft.	12.5%
4	Shashank Verma	Worked on the frontend implementation. Created user dashboard, worked on posts functionality and created new post functionality. Implemented Navigation bar for easy access.	12.5%
5	Usama Bin Faheem	Reviewed all the requirements implemented by the team, gave inputs in the UI enhancements . Developed the plan for efficient implementation of phase 2 and phase 3.	12.5%
6	Suhaibuddin Ahmed	Contributed to the home page, verification of the core code functionality document, Helped with the UML diagrams, Involved in the delegation of responsibilities for deliverable 3. Was involved in the creation of the Contact Page. Tweaking the pages	12.5%
7	Varun	Coordinated with the team in the final discussion of	12.5%

	Mahankali	the UI Screens, Created the Login Page, Signup Page, Home Page, and wrote the API's for the flow of data from Frontend to Backend and to Database. Helped in connection with the database and handled the Backend Part. Briefly described along with screenshots the manual for compiling/running of our program and also updated the output screens.	
8	Rahman Mehmood	Created UML Diagrams, Collaborated and Delegated responsibilities for deliverable 3 (Frontend and Backend), UI Design Slides, Helped in the creation of Contact Us page, Reviewed the deliverable 3 document specifically for the peer review and brief reflection on what can be improved.	12.5%