WhereToEat

Team Noobs

Ibrahim Alkuwaifi

Mohammad Khaja Moinuddin

Sai Kumar Gurram

Teja Peddi

Ashesh Kumar Pinisetti

Vishal Kandunuri

Mohammed Abdul Wahid

Gouthami Gongidi

Mallikarjun Narra

CSCE 5430.008

Requirements:

- Authentication (100%)
  - User Sign-up, Login, and Password Reset
    - Implement a user sign-up, login, and password reset functionality using Firebase Authentication.
    - Utilize Firebase's SDK to handle the backend logic for authentication. Integrate this into the UI to provide a seamless experience for users.
  - Secure Access and Restrictions
    - Ensure that access to the app is secure and that only authenticated users can create and join hangouts.
    - Use Firebase rules to limit database reads/writes to authenticated users. Implement middleware to check authentication status before rendering sensitive components in the app.
- Hangout Creation (100%)
  - Create Hangouts
    - Allow users to create hangouts, specifying a name, location, and an optional description.
    - Implement a form capturing the hangout name, location, and optional description. Use geolocation APIs to suggest locations.
  - Hangout Options
    - Enable the hangout creator to choose whether users can add their own restaurants or if the system should generate random restaurants.
    - Add toggles or checkboxes in the hangout creation form to allow this selection. Implement logic to handle either scenario.
- Adding Restaurants (100%)

- User-Added Restaurants
    - Enable users to add their own restaurants, capturing relevant information like name, address, cuisine, etc.
    - Create a form that captures the necessary details. Validate the information before saving it to the database.

Scope Update

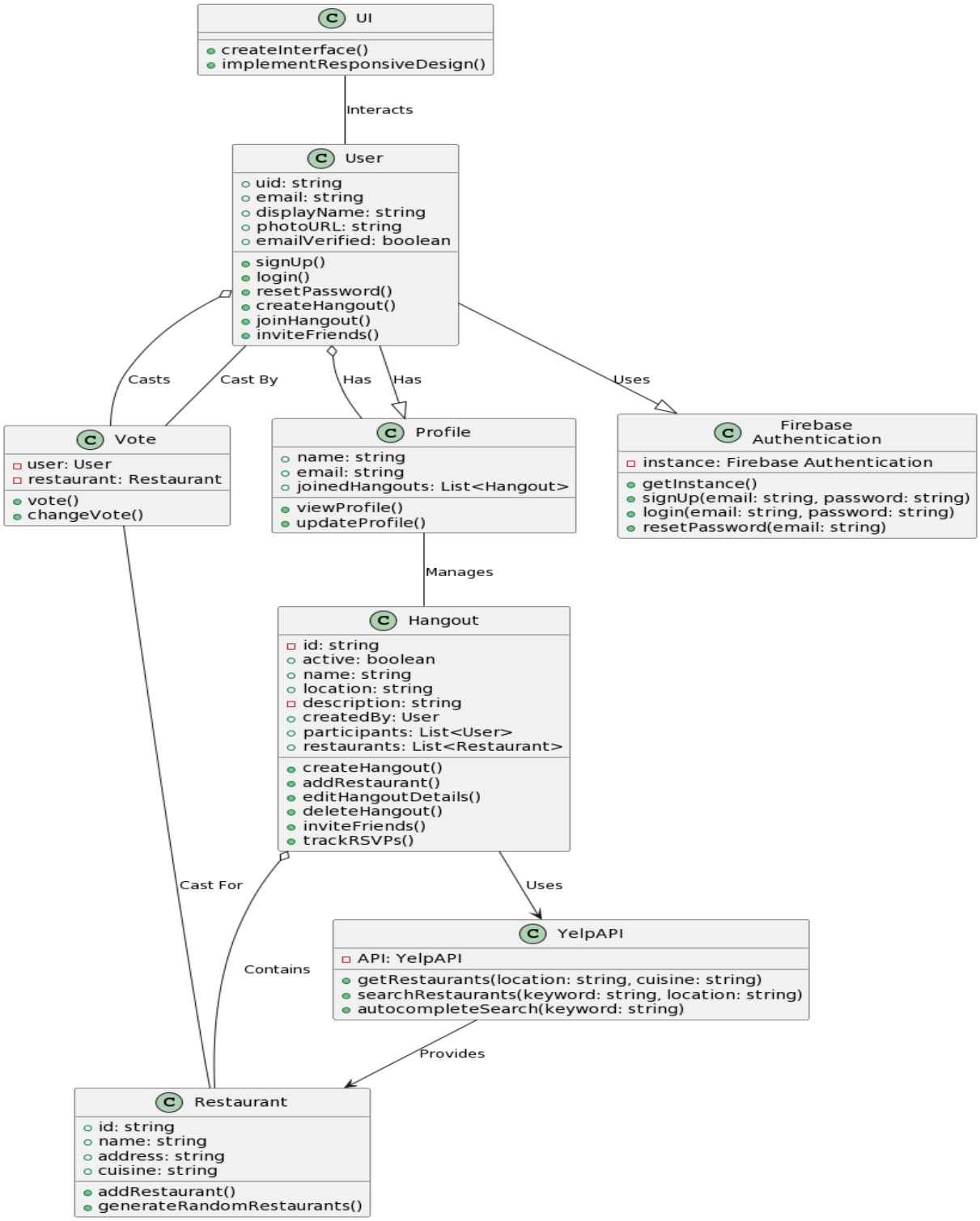
The scope for Phase 2 changed from Deliverable 2. We are using yelp fusion instead of Google Places since it is free. Below are the more detailed descriptions of the requirements specified for Phase 2.

Phase 2: Enhanced User Experience and Integration

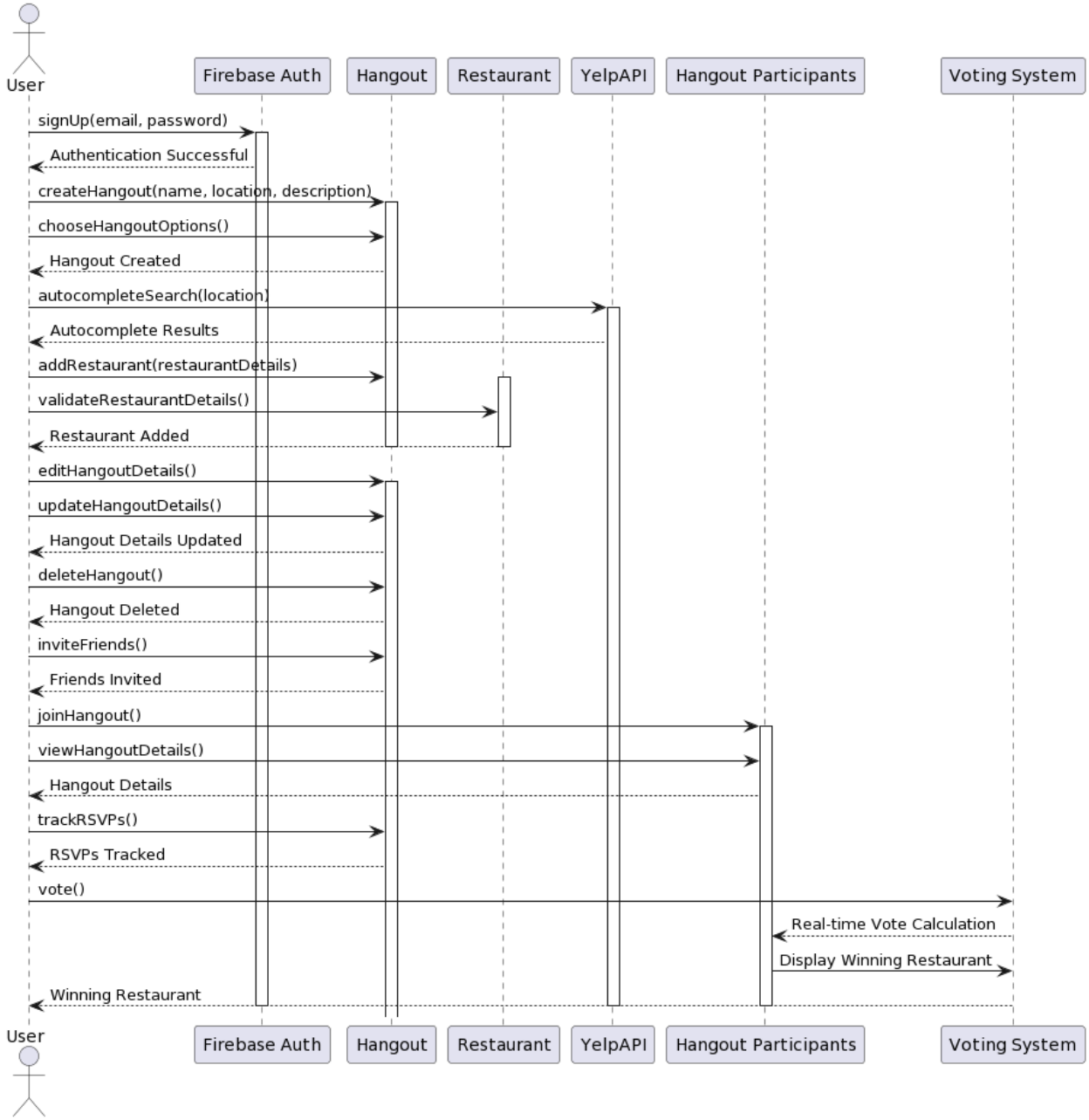
1. Yelp Fusion Integration (100%):
    a. Integrate Yelp Fusion API to access restaurant data and enhance the restaurant suggestion feature.
    b. Implement autocomplete functionality for restaurant search and location input.
2. User Interface Development (100%):
    a. Utilize Bootstrap and Angular Material to create a unified and user-friendly interface.
    b. Develop responsive design for optimal viewing on various devices (desktop, tablet, mobile).
3. Hangout Management(50%):
    a. Implement features to edit, delete, and manage hangouts.
    b. Allow users to invite friends to join hangouts and track RSVPs.

UML design:

Class Diagram:

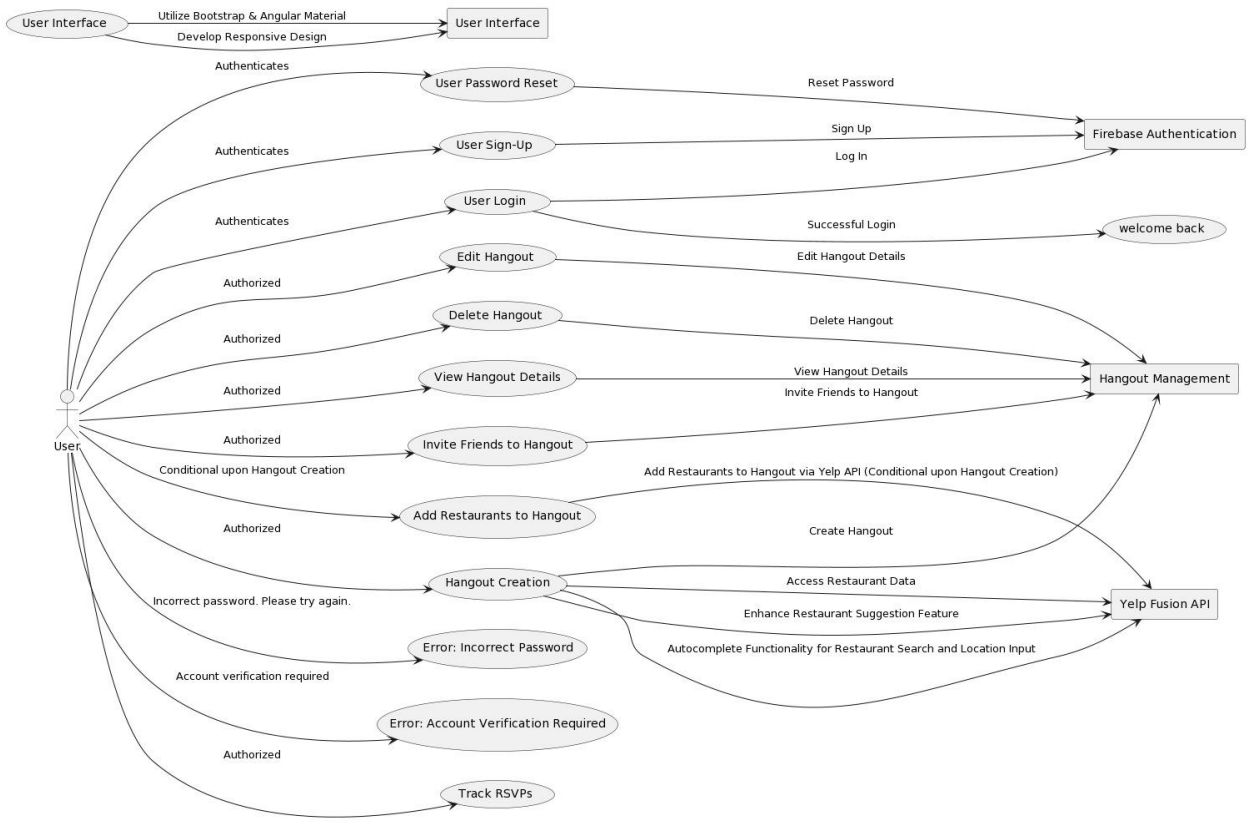


Sequence Diagram:



Use Case Diagram:

Added normal and error case



Test cases:

Unit tests for Forgot Password page:

Code:

```
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { FormGroup, ReactiveFormsModule } from '@angular/forms';
import { of } from 'rxjs';
import { AuthService } from 'src/app/auth/auth.service';
import { ForgotPasswordPageComponent } from './forgot-password-page.component';
fdescribe('ForgotPasswordPageComponent', () => {
let component: ForgotPasswordPageComponent;
let fixture: ComponentFixture<ForgotPasswordPageComponent>;
let authService: AuthService;
beforeEach(() => {
```

```
TestBed.configureTestingModule({
declarations: [ForgotPasswordPageComponent],
imports: [ReactiveFormsModule],
providers: [
{
provide: AuthService,
useValue: {
ForgotPassword: () => of(true), // Mock the AuthService's ForgotPassword
method
},
},],
});
fixture = TestBed.createComponent(ForgotPasswordPageComponent);
component = fixture.componentInstance;
authService = TestBed.inject(AuthService);
});
it('should create the component', () => {
expect(component).toBeTruthy();
});
it('should initialize the form with username control', () => {
fixture.detectChanges(); // Trigger ngOnInit
const usernameControl = component.passwordForm.get('username');
expect(usernameControl).toBeTruthy();
});
it('should reset password and set loading to false', () => {
fixture.detectChanges();
const username = 'test@example.com';
// Spy on AuthService's ForgotPassword method to check if it was called
const authServiceSpy = spyOn(authService,
'ForgotPassword').and.returnValue(Promise.resolve());
// Set the username value
component.passwordForm.get('username')?.setValue(username);
// Call the resetPassword method
component.resetPassword();
// Expect that AuthService's ForgotPassword was called with the username
expect(authServiceSpy).toHaveBeenCalledWith(username);
// Expect loading to be set to false after completing
expect(component.loading).toBeTruthy();
});
```

Description:

Three test cases have been written in this example:

The first test case determines whether the component was correctly constructed.

When ngOnInit is triggered, the second test case determines whether the form has an empty username field upon initialization.
Third test scenario determines whether the resetWhen a password is reset, the password function changes the loading flag to true.After the process is finished, return to false.

Unit test for Home Page:

Code:

```
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { ReactiveFormsModule } from '@angular/forms';
import { of } from 'rxjs';
import { AuthService } from 'src/app/auth/auth.service';
import { HomePageComponent } from './home-page.component';
import { HangoutService } from 'src/app/services/hangout.service';
import { AngularFireAuth } from '@angular/fire/compat/auth';
import { Hangout } from 'src/app/models/hangout.model';
fdescribe('HomePageComponent', () => {
let component: HomePageComponent;
let fixture: ComponentFixture<HomePageComponent>;
let authService: AuthService;
let hangoutService: HangoutService;
beforeEach(() => {
TestBed.configureTestingModule({
declarations: [HomePageComponent],
imports: [ReactiveFormsModule],
providers: [
{
provide: AngularFireAuth,
useValue: {
authState: of({ uid: 'user-uid' }), // Mock AngularFireAuth's authState
},
},
{
provide: AuthService,
useValue: {
user$: of({ uid: 'user-uid' }), // Mock AuthService's user$
userData: { uid: 'user-uid' }, // Mock AuthService's userData
},
},
{
provide: HangoutService,
useValue: {
getActiveHangoutsForUser: (uid: string) => of([]), // Mock HangoutService's
getActiveHangoutsForUser
```

```
getPastHangoutsForUser: (uid: string) => of([]), // Mock HangoutService's
getPastHangoutsForUser
createHangout: (hangout: Hangout) => of(hangout), // Mock HangoutService's createHangout
},
},
],
});
fixture = TestBed.createComponent(HomePageComponent);
component = fixture.componentInstance;
authService = TestBed.inject(AuthService);
hangoutService = TestBed.inject(HangoutService);
});
it('should create the component', () => {
expect(component).toBeTruthy();
});
it('should initialize the hangoutForm with the necessary controls', () => {
fixture.detectChanges(); // Trigger ngOnInit
const nameControl = component.hangoutForm.get('name');
const locationControl = component.hangoutForm.get('location');
const descriptionControl = component.hangoutForm.get('description');
expect(nameControl).toBeTruthy();
expect(locationControl).toBeTruthy();
expect(descriptionControl).toBeTruthy();
});
it('should create a new hangout', () => {
fixture.detectChanges(); // Trigger ngOnInit
component.hangoutForm.setValue({
name: 'Test Hangout',
location: 'Test Location',
description: 'Test Description',
});
const createHangoutSpy = spyOn(hangoutService,
'createHangout').and.returnValue(Promise.resolve());
component.createNewHangout();
expect(createHangoutSpy).toHaveBeenCalled();
expect(component.loading).toBeFalse();
});
});
```

Description:
The first test case determines whether the component was correctly constructed.
The accuracy of the form's initialization with empty values is confirmed by the second test case.
In the third test scenario, form values are set and it is verified whether the anticipated data is passed to the createHangout method call.

Unit test for Login page:

Code:

```
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { ReactiveFormsModule } from '@angular/forms';
import { of } from 'rxjs';
import { AuthService } from 'src/app/auth/auth.service';
import { LoginPageComponent } from './login.component';
fdescribe('LoginPageComponent', () => {
let component: LoginPageComponent;
let fixture: ComponentFixture<LoginPageComponent>;
let authService: AuthService;
beforeEach(() => {
TestBed.configureTestingModule({
declarations: [LoginPageComponent],
imports: [ReactiveFormsModule],
providers: [
{
provide: AuthService,
useValue: {
SignIn: () => of(true), // Mock the AuthService's SignIn method
},
},
],
});
fixture = TestBed.createComponent(LoginPageComponent);
component = fixture.componentInstance;
authService = TestBed.inject(AuthService);
});
it('should create the component', () => {
expect(component).toBeTruthy();
});
it('should initialize the loginForm with the necessary controls', () => {
fixture.detectChanges(); // Trigger ngOnInit
const usernameControl = component.loginForm.get('username');
const passwordControl = component.loginForm.get('password');
expect(usernameControl).toBeTruthy();
expect(passwordControl).toBeTruthy();
});
it('should call the AuthService SignIn method and set loading to false', () => {
fixture.detectChanges();
const username = 'test@example.com';
const password = 'password123';
```

```
// Spy on AuthService's SignIn method to check if it was called
const signInSpy = spyOn(authService, 'SignIn').and.returnValue(Promise.resolve());
// Set the username and password values
component.loginForm.get('username')?.setValue(username);
component.loginForm.get('password')?.setValue(password);
// Call the login method
component.login();
// Expect that AuthService's SignIn was called with the username and password
expect(signInSpy).toHaveBeenCalledWith(username, password);
// Expect loading to be set to false after completing the request
expect(component.loading).toBeTruthy();
});
});
```

Description:
The first test case determines whether the component was correctly constructed.
The accuracy of the form's initialization with empty values is confirmed by the second test case.
In order to replicate a login activity, the third test case mocks the AuthService and confirms that the SignIn method is called.

Unit test for Sign up page:

Code:

```
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { ReactiveFormsModule } from '@angular/forms';
import { of } from 'rxjs';
import { AuthService } from 'src/app/auth/auth.service';
import { SignupPageComponent } from './signup-page.component';
fdescribe('SignupPageComponent', () => {
let component: SignupPageComponent;
let fixture: ComponentFixture<SignupPageComponent>;
let authService: AuthService;
beforeEach(() => {
TestBed.configureTestingModule({
declarations: [SignupPageComponent],
imports: [ReactiveFormsModule],
providers: [
{
provide: AuthService,
useValue: {
SignUp: () => of(true), // Mock the AuthService's SignUp method
},
},
],
```

```
});
fixture = TestBed.createComponent(SignupPageComponent);
component = fixture.componentInstance;
authService = TestBed.inject(AuthService);
});
it('should create the component', () => {
expect(component).toBeTruthy();
});
it('should initialize the registerForm with the necessary controls', () => {
fixture.detectChanges(); // Trigger ngOnInit
const usernameControl = component.registerForm.get('username');
const passwordControl = component.registerForm.get('password');
expect(usernameControl).toBeTruthy();
expect(passwordControl).toBeTruthy();
});
it('should call the AuthService SignUp method and set loading to false', () => {
fixture.detectChanges();
const username = 'test@example.com';
const password = 'password123';
// Spy on AuthService's SignUp method to check if it was called
const signUpSpy = spyOn(authService, 'SignUp').and.returnValue(Promise.resolve());
// Set the username and password values
component.registerForm.get('username')?.setValue(username);
component.registerForm.get('password')?.setValue(password);
// Call the register method
component.register();
// Expect that AuthService's SignUp was called with the username and password
expect(signUpSpy).toHaveBeenCalledWith(username, password);
// Expect loading to be set to false after completing the request
expect(component.loading).toBeTruthy();
});
});
```

Description:

We have created test cases on the Sign Up page to see if the user can log in with valid credentials and if the information they have entered is correct.

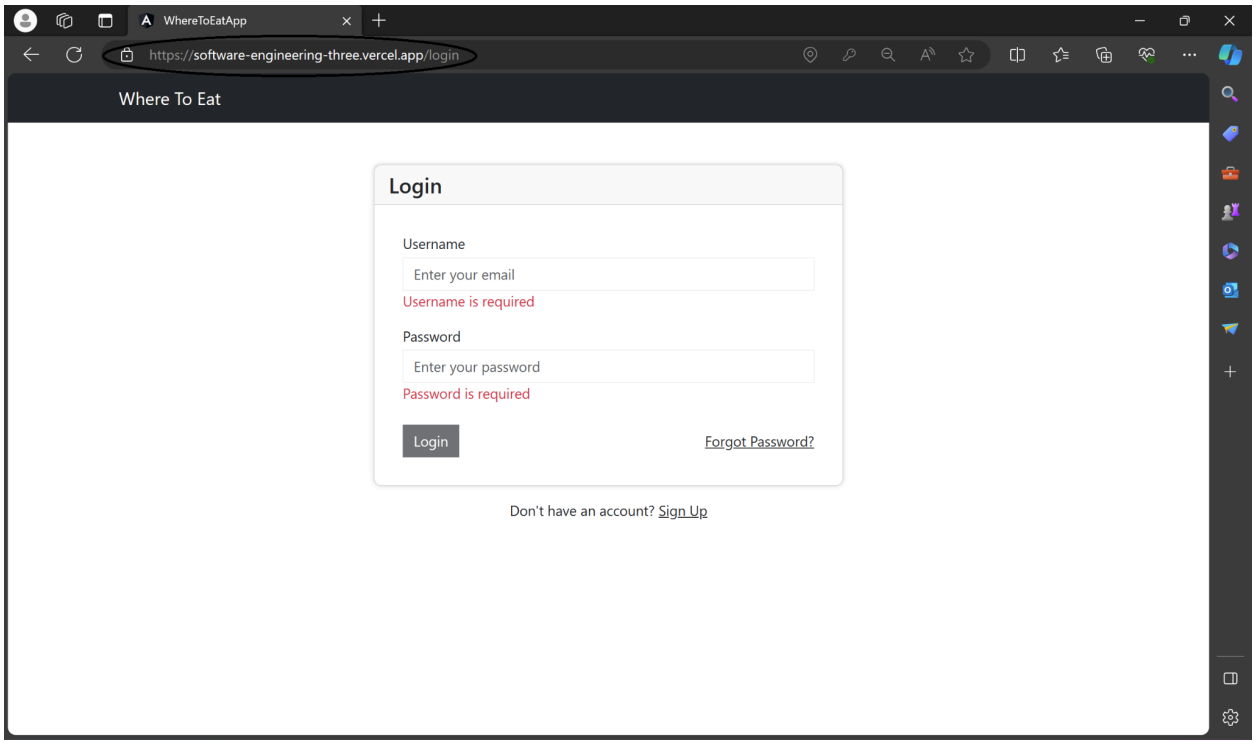
Additionally, it verifies that all fields have data entered and nothing is left blank.

We verified the user's registration by providing inaccurate information. and made an attempt to register with previously used data, which shouldn't permit the user to register once more.

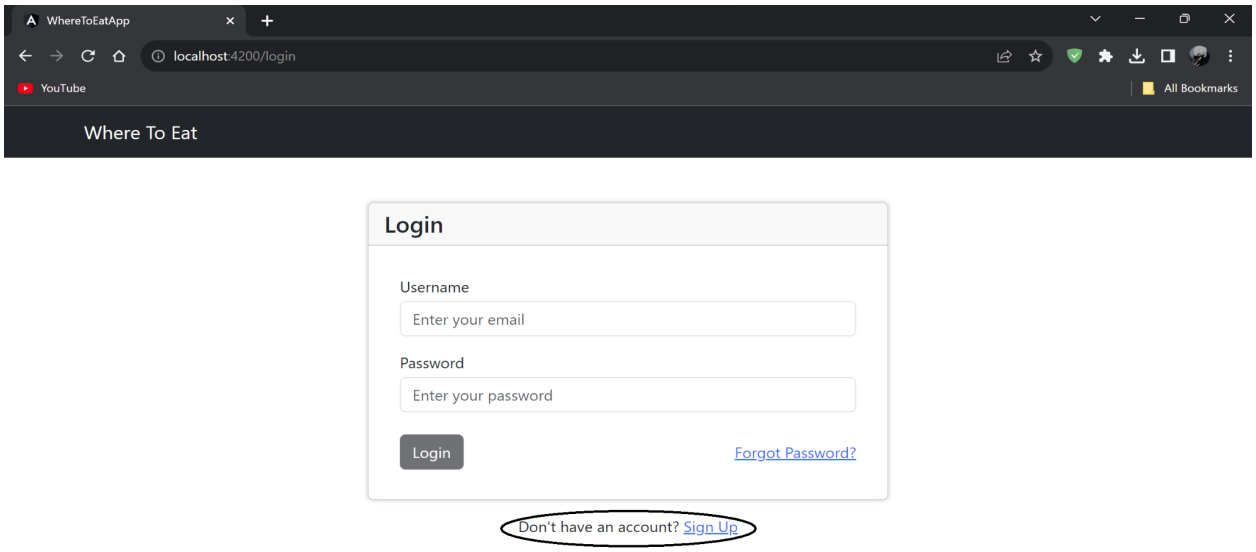
User Manual:

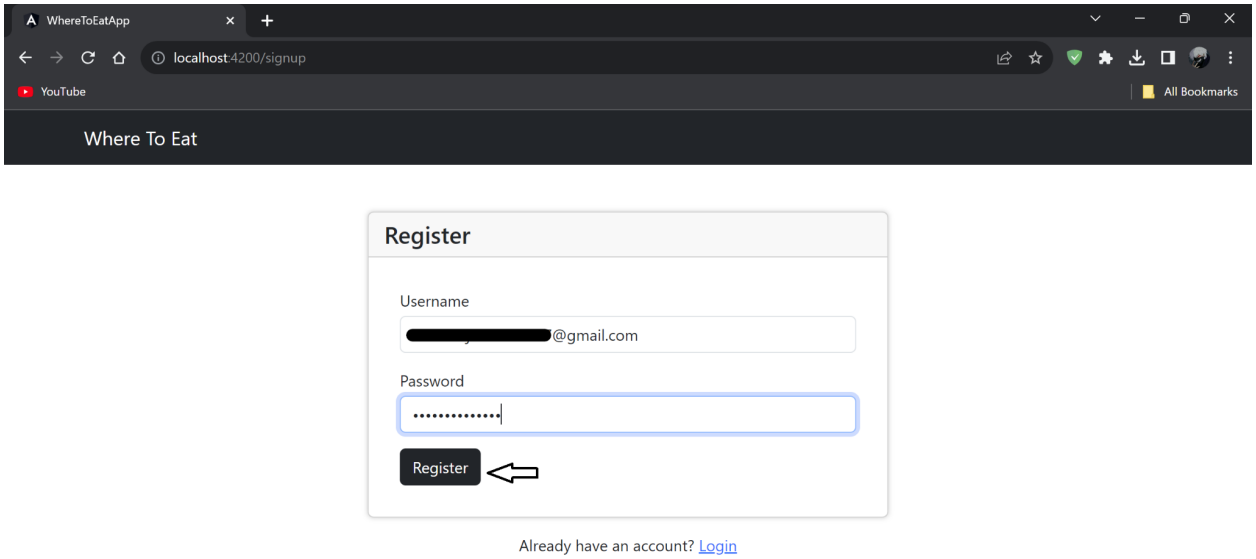
Here is a step-by-step follow-up on accessing our application WHERE TO EAT.

Step 1: Once the user accesses the page using the URL, the User gets redirected to the landing page.

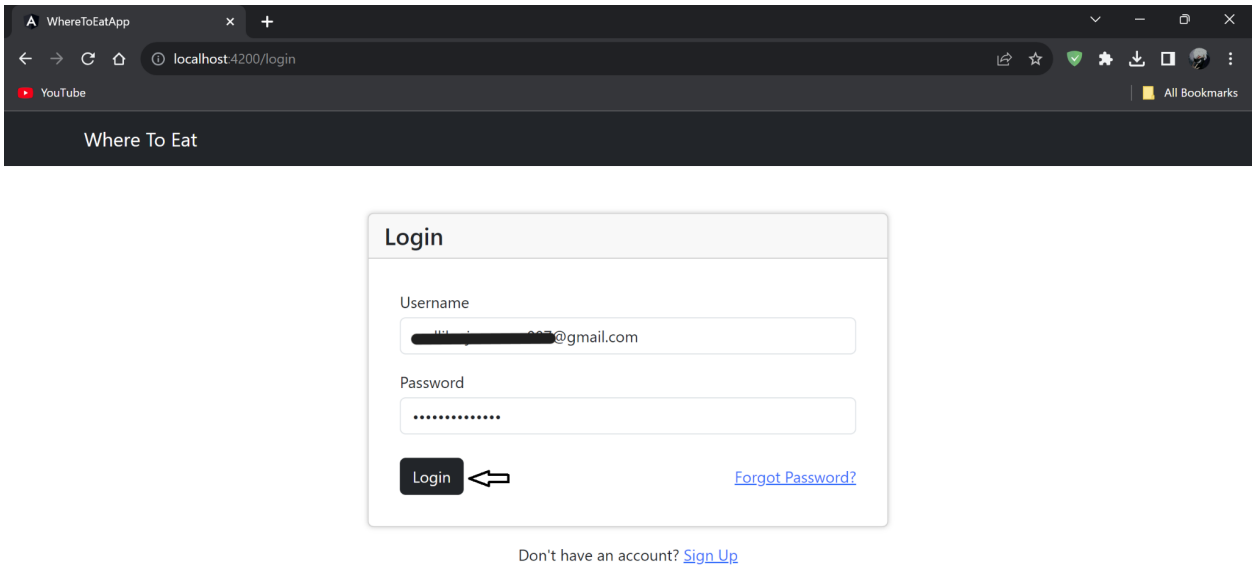


Step 2: If the client is a new user, he can use the app's Sign Up feature for making a new UserAcc.





Step 3: Once the user gets to register he is redirected to the Login page where he gets to enter the same credentials to Login.



Step 4: After **step 3**, the User is done with the registration and login, and the User gets a verification email on the registered email.

Verify your email for project-460596574420 Inbox ×

noreply@where-to-eat-60c46.firebaseapp.com 4:33 PM (2 minutes ago)

to me

Hello,

Follow this link to verify your email address.

[redacted]

If you didn't ask to verify this address, you can ignore this email.

Thanks,

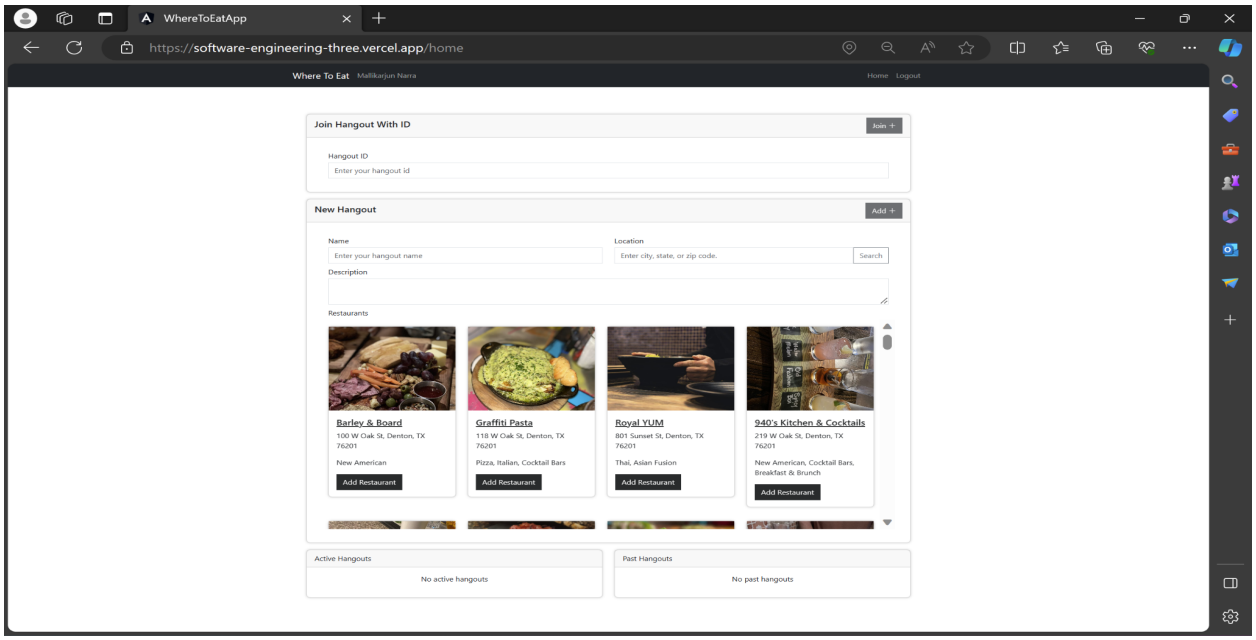
Your project-460596574420 team

Once verified, the User gets a confirmation that the email is verified.

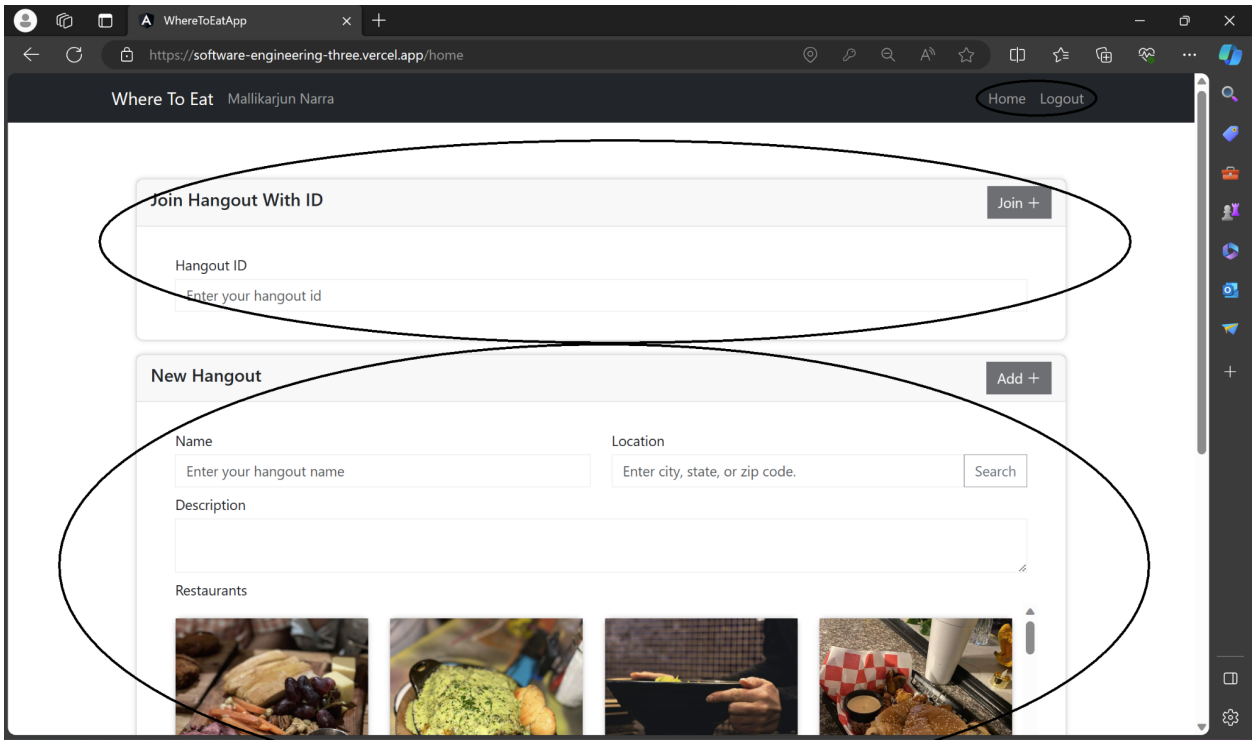
Your email has been verified

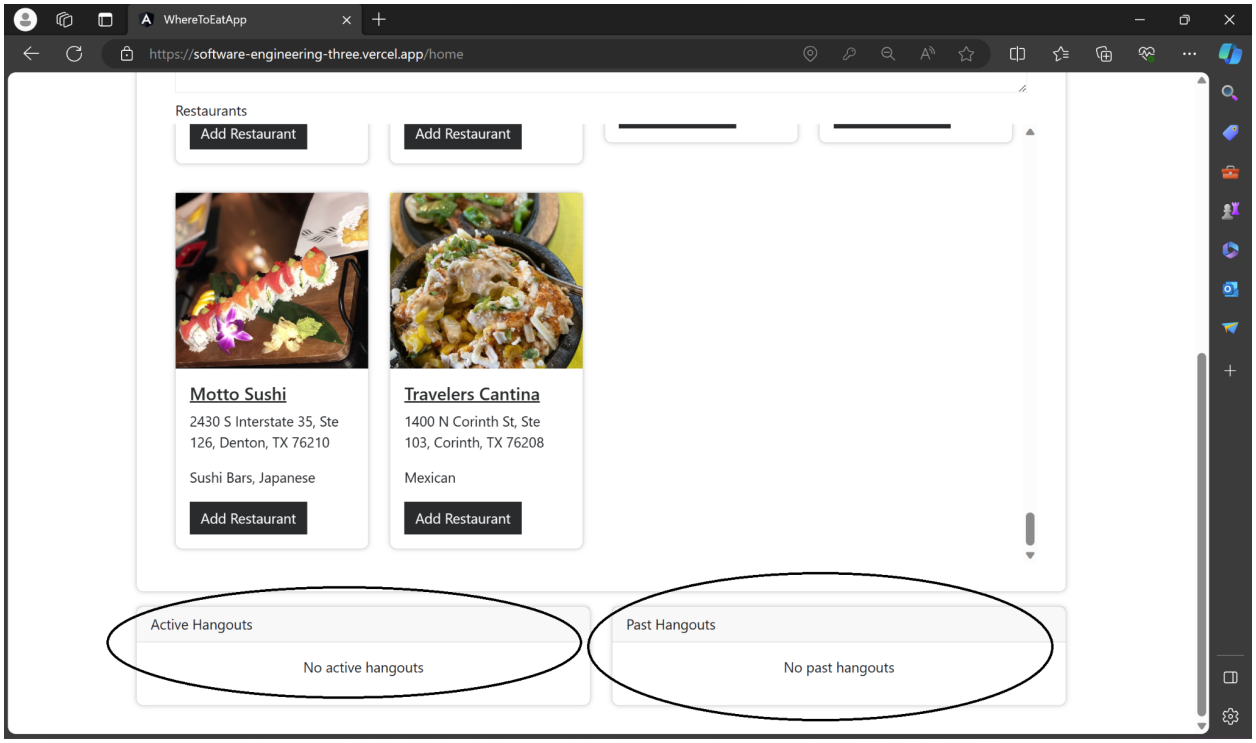
You can now sign in with your new account

Step 5: Once the user successfully logs in, the User gets redirected to the home page of our application.

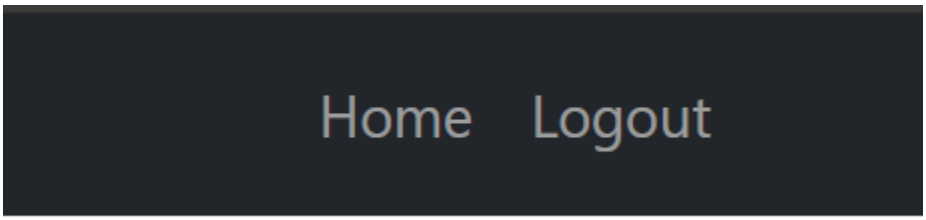


A few key components of the application are

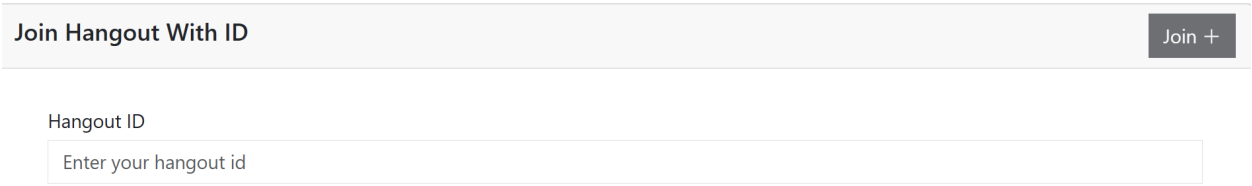




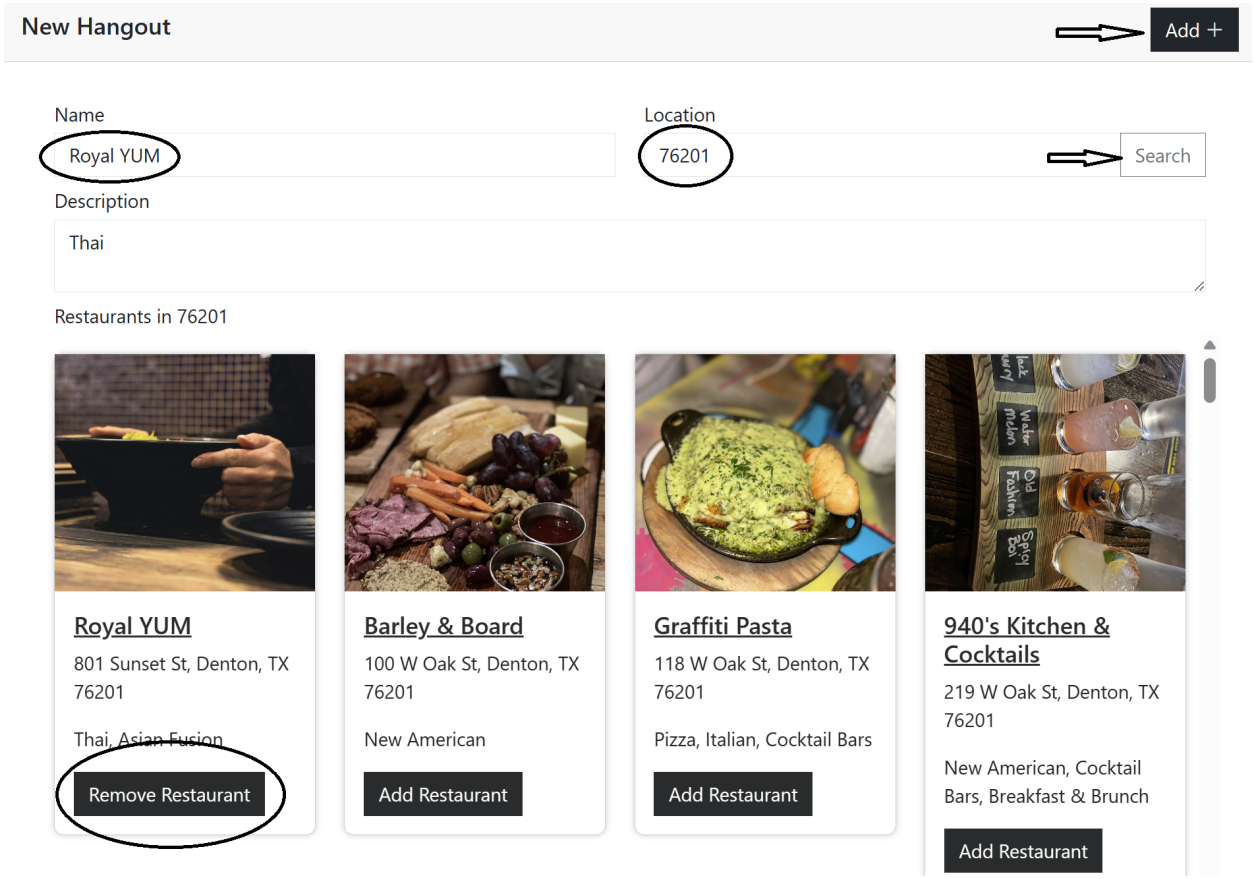
- In the top right corner, we have **Home | Logout** buttons which are responsible for redirecting to the home page when clicked on Home and Logging out of the application when clicked on LogOut.



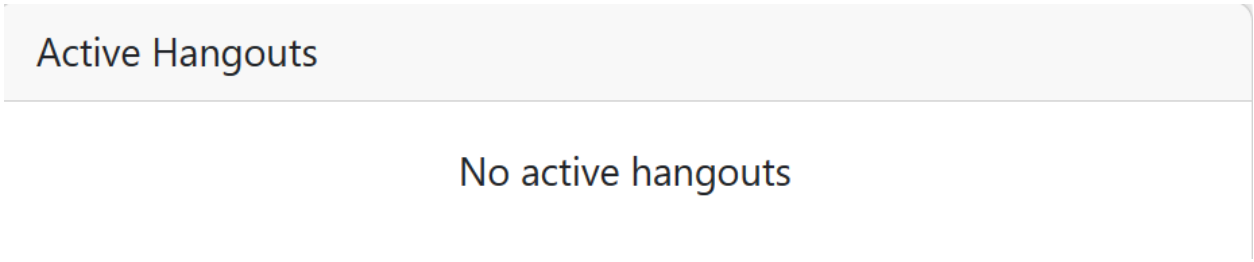
- If the user already has a hangout created and wants to join the **Join Hangout With ID** Section, the User can join the active hangout ID and then Click on **Join +**



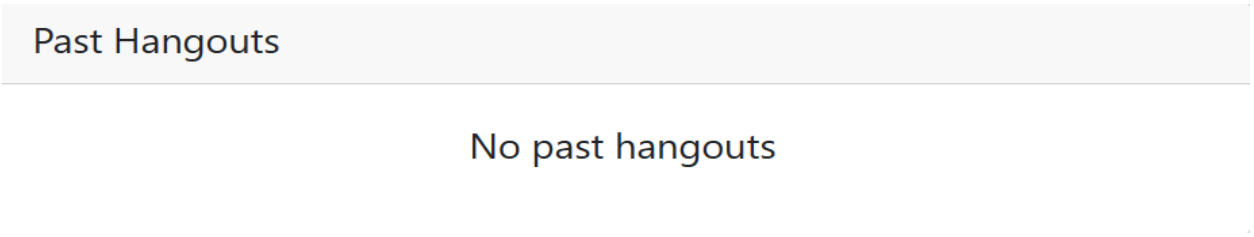
- We also have a feature to create a new Hangout in the **New Hangout** section using the **Add +** button which typically creates a new Hangout when you give a verified **Hangout Name, Zip Code, and Description then search / Select a Hangout from the Dropdown of options available** Such that your hangout group can join.



- Current hangouts are displayed in the **Active Hangouts** section, where all the current sessions are displayed. User/s can hop from hangout to hangout.



- Our application also has a **Past Hangouts** where user/s view a list of past hangouts (History) i.e (Foodie Adventure Hangout, Movie Night Out, Coffee and Conversation, etc.



Compilation and Run Instructions:

- Take a clone of the project repository from your Git or other version control system.
    - git clone https://github.com/brofcb/software-engineering.git
    - cd WhereToEatApp
    - Use npm to install project dependencies - npm install
- Configure Firebase
    - Create a Firebase project via the Firebase Console to configure Firebase.
    - Set up Firebase using your Firebase credentials and initialize it in your app.

      ```
      // Firebase configuration setup
      const firebaseConfig = {
      apiKey: "<YOUR_API_KEY>",
      authDomain: "<YOUR_AUTH_DOMAIN>",
      projectId: "<YOUR_PROJECT_ID>",
      storageBucket: "<YOUR_STORAGE_BUCKET>",
      messagingSenderId: "<YOUR_MESSAGING_SENDER_ID>",
      appId: "<YOUR_APP_ID>"
      };
      ```
- Configure Yelp
    - Create an app via the yelp for developers to generate API key
    - Set up the Yelp API key and configure it in your app
    - YELP_API_KEY: process.env.YELP_API_KEY || '<Your API key>'
- Running the Web Application:
    - Start the web application server - npm start
- Access the web application:
    - Open a web browser and go to http://localhost:PORT (insert port number here to indicate the port your server is operating on).

    - You will be sent to the login page, where you may get the four essential elements: registration, password reset, username, and password.
- Test Cases:
    - Make sure your web application's test cases cover a variety of scenarios and features before beginning any testing.You may create and run these test cases using testing frameworks such as Mocha, Jasmine, or Jest.
- Run the test suite:
    - npm test
    - Verify that all features and components are operating as intended by reviewing the test results.

Code Inspection Workshop Summary:

We presented our “inspection-code-team-noobs” report, Before the meeting, which was reviewed by the stakeholders. Given their limited experience with Angular, they had few questions, and the only inquiry they raised is related to the security measures employed in our authentication process. We assured them that we rely on Google's Firebase API for handling all our security requirements, which satisfied their concerns. Towards the end of the meeting, we assured the stakeholders that they could always reach out to us via email if they had any doubts or needed further clarification on any aspect of the report/project. Overall, the stakeholders feedback was positive.

Feedback - Peer Review:

During the peer review session, the project received valuable feedback in several areas.

Documentation Clarity:

Feedback indicated that the documentation needed to be clearer and more comprehensive. In response, the team worked on improving the user documentation, ensuring that it was easy to understand and covered all essential aspects of the application.

Testing and Quality Assurance:

Feedback highlighted the importance of testing. As a result, the team implemented few test cases to identify and address any bugs or issues.

Security and Privacy:

The team made sure that Firebase Authentication was configured to the latest best practices to protect user data effectively.

Scalability:

Some feedback addressed the application's scalability, suggesting that it should be designed to handle potential growth in the user base. In response, the team made architectural changes and ensured that the database and infrastructure could scale as needed.

Feedback Implementation Tracking:

A systematic approach to tracking and documenting feedback and its subsequent actions. The team agreed to ensure that all suggestions were addressed and resolved effectively.

A brief reflection on the "WhereToEat" project:

Accomplishments:

We have made progress in implementing key features of the project. These include user authentication, hangout creation, restaurant addition, voting, and data management using Firebase and the Google Places API. Our focus has been on creating a user interface that allows navigation, easy hangout creation and joining, restaurant voting, and profile management.

We have effectively managed data using Firestore. Additionally, we have prioritized planning and documentation by following a defined timeline and providing user documentation on GitHub.

Areas for Improvement:

The report also acknowledges areas where enhancements can be made. It suggests conducting testing with an emphasis on unit testing and integration testing. As our application grows we need to consider its scalability while ensuring responsiveness across devices. Continuous feedback from users along with development can help refine the application by incorporating features based on user requirements.

Project Risks:

Furthermore, the document outlines risks that may arise during the project development process. These risks cover challenges, market uncertainties, and operational issues.

There are risks to consider such as relying heavily on external services facing technological challenges with the chosen framework adapting to changing user preferences dealing with limited resources and scalability concerns, addressing privacy and security risks, and ensuring compliance with data protection laws. It is vital to acknowledge and mitigate these risks in order to achieve success in the project.

Regarding project resources:

The document outlines resources involved in the project including resources, financial assets, and environmental elements like third-party APIs and UI/UX design libraries. Proper management and licensing of these resources play a role in ensuring the success of the project.

Project Timeline:

Here, we have used **Gantt Chart** to show the project timeline: Gantt chart

Members Contribution Table:

Member name	Contribution description	Overall Contribution (%)	Note(if applicable)
Ibrahim Alkuwaifi	I worked on adding the Yelp fusion API. I also worked on updating the progress for this week. I also worked on hosting the webapp	60%	
Mohammad Khaja Moinuddin	Worked on breif reflection on project and updated the timeline of Gantt chart.	5%	
Teja Peddi	Worked on UML,Sequence, Usecase including one normal case and one error case	5%	
Mohammed Abdul Wahid	I updated the meeting minutes for Deliverable -4 in the project repository on GitHub mentioning the Agenda Items/Topics Discussed and Action Items/Tasks Assigned.	5%	
Gouthami Gongidi	Helped to work on Unitcase diagram	5%	
Mallikarjun Narra	Drafted User manual with the summery all key components	5%	
Ashesh kumar Pinisetti	Added code inspection workshop summary Added compilation/run instructions	5%	

	Added note-deliverable-4 in gitlab Updated file directory structure in gitlab		
Sai Kumar Gurram	Added Unit test cases and updated the Test case Description	5%	
Vishal Kandunuri	Added Feedback-Peer review	5%	