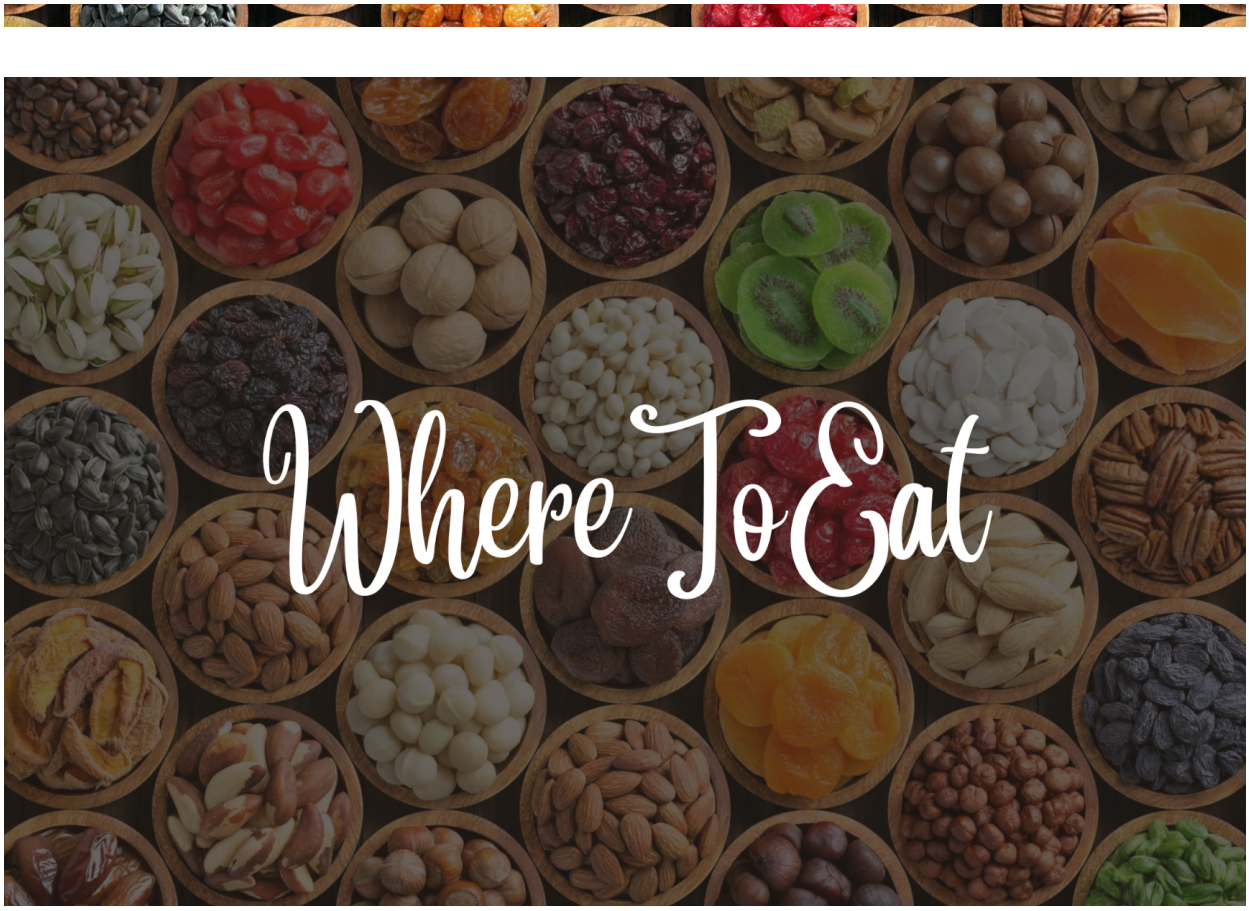


WhereToEat

Team Noobs

Ibrahim Alkuwaifi

Sai Kumar Gurram

Teja Peddi

Ashesh Kumar Pinisetti

Mohammad Khaja Moinuddin

Vishal Kandunuri

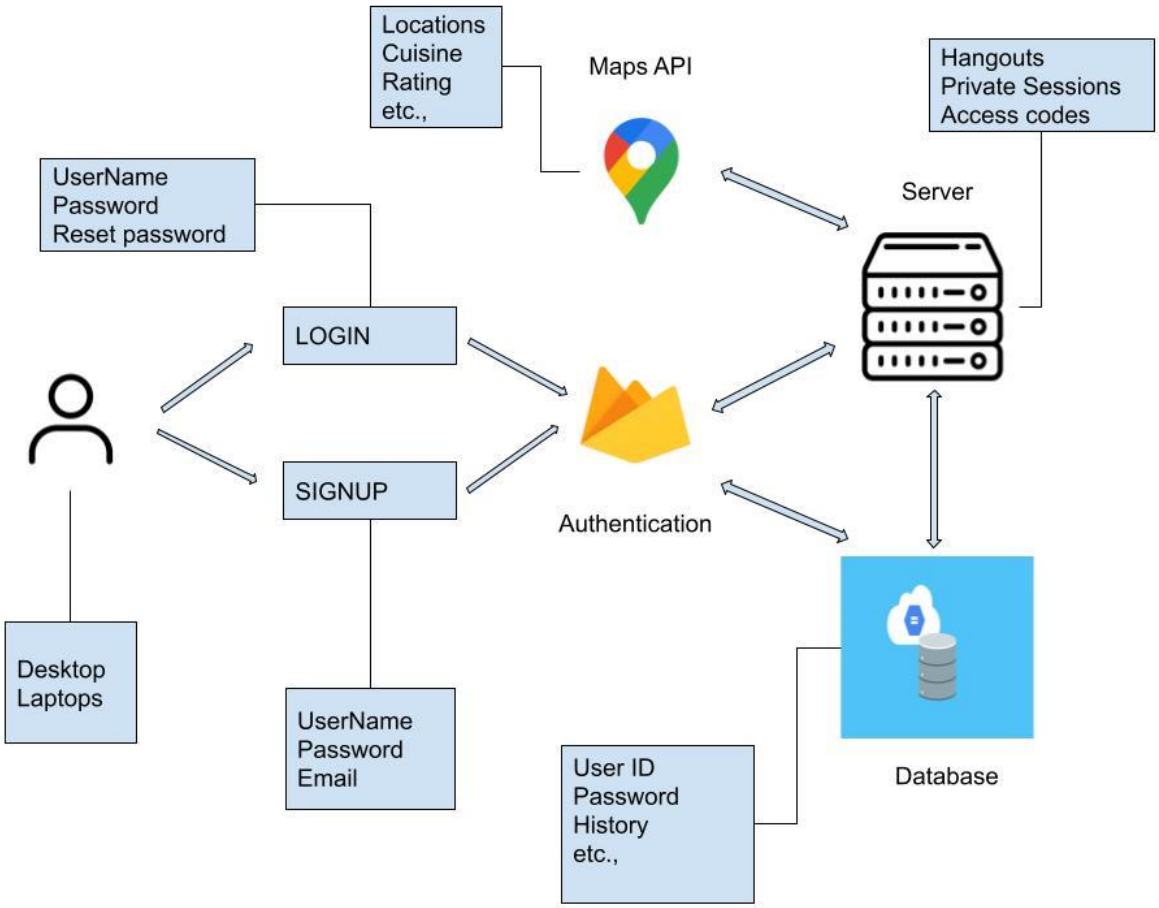
Mohammed Abdul Wahid

Gouthami Gongidi

Mallikarjun Narra

CSCE 54320.008

System Structure:



Our project utilizes a three-tier architecture. Upon accessing our webpage, the client is redirected to a login page featuring four key components: username, password, password reset, and signup. These components are solely applicable to existing users who can log in using their account details. Should a client forget their login credentials, we have incorporated a password reset feature that overwrites pre-existing passwords. New clients

can utilize our signup module to enter their details, create an account, and subsequently log in using their new credentials. We rely on Google Firebase authentication and fire storage for both login and signup, enabling real-time data for a seamless user experience. Once authenticated and verified, the client is redirected to our home page where they can access either a hangout or private hangout, and input their desired restaurant, cuisine, or location. The incorporation of Maps API is particularly useful for clients with specific preferences such as ratings, cuisine, and location. Once all details are finalized, and a restaurant is chosen, it is saved to the client's profile's history tab, and all data is written onto fire storage for disaster recovery purposes.

Requirements:

Functional Requirements:

- Authentication
    - Users can sign up.
    - Users can log in.
    - Users can reset their password.
- Hangout Creation
    - Hangout creators can specify the hangout's name.
    - Hangout creators can specify the hangout's location.
    - Hangout creators can provide an optional description.
    - Hangout creators can choose whether users can add their own restaurants.
    - Hangout creators can choose to generate random restaurants.
- Adding Restaurants
    - Users can add their own restaurants.
    - Users can provide restaurant details (name, address, cuisine type, etc.).
    - Users can manually add nearby restaurants using the Google Places API if desired.
    - When generating random restaurants, the app uses the Google Places API to fetch a list of nearby restaurants.

- Restaurant Voting
    - All hangout participants can view and vote for their preferred restaurant.
    - Real-time vote calculations are displayed.
    - The winning restaurant is highlighted.
- Hangout Interaction
    - Users can join hangouts created by others.
    - Users can view hangout details.
    - Users can see the list of restaurants and their voting status.
    - Users can vote and change their votes as needed.
- Data Management
    - Firestore is used to store hangout details.
    - Firestore is used to store user information.
    - Firestore is used to store restaurant data.
    - Firestore collections are structured for hangouts, users, and restaurants.
- User Profile
    - Users can view their profiles.
    - Users can update their profiles.
    - User profiles display user-specific information.
    - User profiles display the hangouts they've joined or created.

Non-Functional Requirements:

- Security
    - Authentication must be secure to protect user data.
    - User passwords must be stored securely.
    - Access to hangouts and user profiles should be restricted to authenticated users only.

- Performance
    - Real-time vote calculations must be fast and responsive.
    - Data retrieval and updates from Firestore should be efficient.
    - The app should handle concurrent users smoothly during hangout interactions.
- Reliability
    - The app should be available and reliable, with minimal downtime.
    - Data stored in Firestore should be consistent and not prone to corruption.
- Usability
    - The user interface should be intuitive and easy to use.
    - Hangout details and restaurant information should be presented in a user-friendly manner.
- Scalability
    - The app should be able to handle a growing number of users and hangouts.
    - Firestore should scale to accommodate increased data storage requirements.
- Data Integrity
    - Data in Firestore should be accurate and free from errors.
    - User profiles and hangout details should be updated correctly.
- Integration
    - Integration with Firebase Authentication and Google Places API should be seamless.
    - Data integration between different app components should be smooth.

Interfaces:

User interface :

- Initially the User lands on the **Login** Page.

- He should be able to login if he has already registered (**note:** the user should have a firebase account in order to create an account here).Otherwise, the user should register his details in order to login to the account in the **Signup** Page.
- If the user forgets the password he will be able to reset the password in the **Forgot Password** page.
- Once the user fills the valid credentials he should be able to login to the website and **Homepage** will be displayed. This page will have all the information that is required for the user. He will be able to look through different restaurants nearby, and can add hangouts that he is interested in. Make interactions by creating voting criteria. Finally he can Logout from the website.

Software interface :

We have used Firebase for our website. It will handle all the interactions such as, storing the user login information in a secured format, handling the Api calls etc. Apart from that we also use Google Places API to fetch a list of nearby restaurants. All the data modification for a user like adding or deleting hangouts and updating user information will be done in the UI itself.

Planning :

Project scope:

The "WhereToEat" application is a social dining decision-making platform created to make pick a restaurant when dining out with friends or family more straightforward. The scope of this project includes the development of a user-friendly web application with the following key features:

- Authentication:
    - Users can sign up or log in using Firebase Authentication, ensuring secure access to the app. Only authenticated users can create and join hangouts. Users can also reset their password if they forgot it.

- Hangout Creation:
    - Hangout creators can specify the hangout's name, location, and an optional description. They can also choose whether users can add their own restaurants or generate random restaurants.
- Adding Restaurants:
    - Users can add their own restaurants, they can provide restaurant details, including name, address, cuisine type, and other relevant information. They can also manually add nearby restaurants using the Google Places API. If generating random restaurants, the app uses the Google Places API to fetch a list of nearby restaurants.
- Restaurant Voting:
    - All hangout participants can view and vote for their preferred restaurant. Real-time vote calculations are displayed, and the winning restaurant is highlighted.
- Hangout Interaction:
    - Users can join hangouts created by others, view hangout details, and see the list of restaurants and their voting status. An intuitive interface allows users to vote and change their votes as needed.
- Data Management:
    - Firestore is used to store hangout details, user information, and restaurant data. Firestore collections are structured for hangouts, users, and restaurants, ensuring efficient data management.
- User Profile:
    - Users can view and update their profiles, displaying user-specific information and the hangouts they've joined or created.

RISKS:

Technical Risks:

- Dependency on Third-party Services: The project depends on outside services like Firebase and the Google Places API. The functioning of the application can be affected if certain services go offline or change how they are used.
- Technology Stack Challenges: Compatibility issues or problems with the chosen technology stack (Angular, Firebase) may cause developmental delays or unforeseen technical issues.

Market Risks:

- Users' Preferences Might Change: The popularity of the "WhereToEat" app may change if users' tastes in eating out or using restaurant recommendation apps change.

Operational risks:

- Resource constraints:
    - Insufficient technical or human resources may cause project delays or poor results.
- Issues with Scalability:
    - A non-scalable application may have trouble managing a a growing user base.
- Risks to privacy and security:
    - Data breaches and adverse publicity could result from security flaws that allow unauthorized access to user data.
- Non-compliance:
    - Violations of data protection laws may subject violators to fines and destroy user trust.

Project Resources:

Human Resources:

The project manager is in charge of the overall planning, carrying out, and monitoring of the project.

Developers, designers, testers, analysts, subject matter experts, and other specialists make up the project team.

Project funders, users, managers, and outside consultants are all stakeholders.

Suppliers or Vendors: If outside parties offer products or services.

Financial Resources:

- Project Budget: Money set aside for costs associated with the project, such as wages, supplies, license fees, and operational costs.
- Contingency Funds: Funds set aside to deal with unforeseen problems or scope adjustments.

Other environmental Resources:

- Google Places API is a third-party API that is used to access restaurant data, boosting the functionality of the application's restaurant suggestion feature.
- The user interface is developed using Bootstrap and Angular Material, which offer pre-made UI components and styles for a unified and user-friendly experience.
- Deployment: Users can access the "WhereToEat" application over the internet thanks to Firebase Hosting, which is utilized to deploy and host it.
- These software and technological tools are ideal for creating web-based programs like "WhereToEat." The success of the application as a whole and the functionality of the Angular, Firebase, Google Places API, and UI/UX design libraries are both influenced by these technologies. For the creation and implementation of projects, these resources must be managed and licensed properly.

ESTIMATE COST & EFFORT: DECOMPOSE THE PROBLEM:

Phases:

Phase 1: Core Features and Basic Functionality

1. Authentication:
    a. Implement user sign-up, login, and password reset using Firebase Authentication.
    b. Ensure secure access to the app and restrict hangout creation and joining to authenticated users.
2. Hangout Creation:
    a. Allow users to create hangouts with a name, location, and optional description.
    b. Enable hangout creators to choose whether users can add their own restaurants or generate random restaurants.
3. Adding Restaurants:
    a. Enable users to add their own restaurants with relevant information (name, address, cuisine, etc.).
    b. Implement a feature to generate random restaurants for users who don't want to add their own.

Phase 2: Enhanced User Experience and Integration

1. Google Places API Integration:
    a. Integrate Google Places 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:
    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:
    a. Implement features to edit, delete, and manage hangouts.
    b. Allow users to invite friends to join hangouts and track RSVPs.

Phase 3: Advanced Features and Deployment

1. Voting System:
    a. Develop a voting system for users to vote on their preferred restaurants within a hangout.
    b. Implement a feature to display voting results and determine the winning restaurant.
2. Notifications:
    a. Implement notifications to inform users about hangout updates, voting results, and other relevant information.
    b. Allow users to customize notification preferences (email, push notifications, etc.).
3. Deployment:
    a. Deploy the application using Firebase Hosting, making it accessible to users over the internet.
    b. Ensure the application is scalable and can handle increased user traffic.

The project team can guarantee that the project is finished on schedule, on budget, and in accordance with its objectives by fulfilling these conditions during each step.

DEVELOP A PROJECT SCHEDULE:

Task 1: Set Up Angular Project (1 week)

Task 2: User Authentication (2 weeks)

Task 3: User Profiles (1 week)

Task 4: Hangout Creation (2 weeks)

Task 5: Restaurant Adding/Generation (2 weeks)

Task 6: Google Places API Integration (2 weeks)

Task 7: User Interface (UI) Design (2 weeks)

Task 8: Hangout Management (2 weeks)

Task 9: Voting System (2 weeks)

Task 10: Notifications (2 weeks)

Task 11: Project Integration (2 weeks)

Task 12: User Testing (1 week)

DEFINE A TASK NETWORK:

Task 1 (Set Up Angular Project).

Task 1 must be finished before Task 2 (User Authentication) may begin.

Task 2 and Task 3 (User Profiles) can work together in parallel.

Once the tasks for Phase 2 have been completed, Task 9 (Voting System) can begin.

USE SCHEDULING TOOLS TO DEVELOP A TIMELINE CHART:

Here, we used the **Gantt chart**: Gantt chart

Determining Milestones and Activities:

- Project start:
  - Milestone: Project initiation and approval.
  - Activities: initial scope definition.
- Development Phases
  - Milestones: Phase completions (e.g., Core Features, User Experience, Advanced Features).
  - Activities: Task execution, development, testing, documentation.
- Integration and Testing :
  - Milestone: Completion of project integration and testing.
  - Activities: Integration of components, user testing, security testing.
- Deployment :
  - Milestone: Successful deployment on Firebase Hosting.
  - Activities: Final testing, deployment planning, configuration.
- User Training and Documentation :
  - Milestone: Completion of user documentation and training.
  - Activities: Documentation development
- Compliance and Security Review
  - Milestone: Compliance documentation and security checks.
  - Activities: Review compliance with regulations and security standards.
- Project Closure
  - Milestone: Project closure, client acceptance, and handover.
  - Activities: Final project review, client acceptance, project handover