IMPLEMENTATION

It is a Next.js project with components for reservations, restaurants, custom search bar, and layout elements. Firebase is used for backend functionality. The repository also includes configuration files and global styles. The README.md contains additional information about the project, its setup, and usage.

Source Files

The source files in hierarchical order and their intended purposes are as follows:

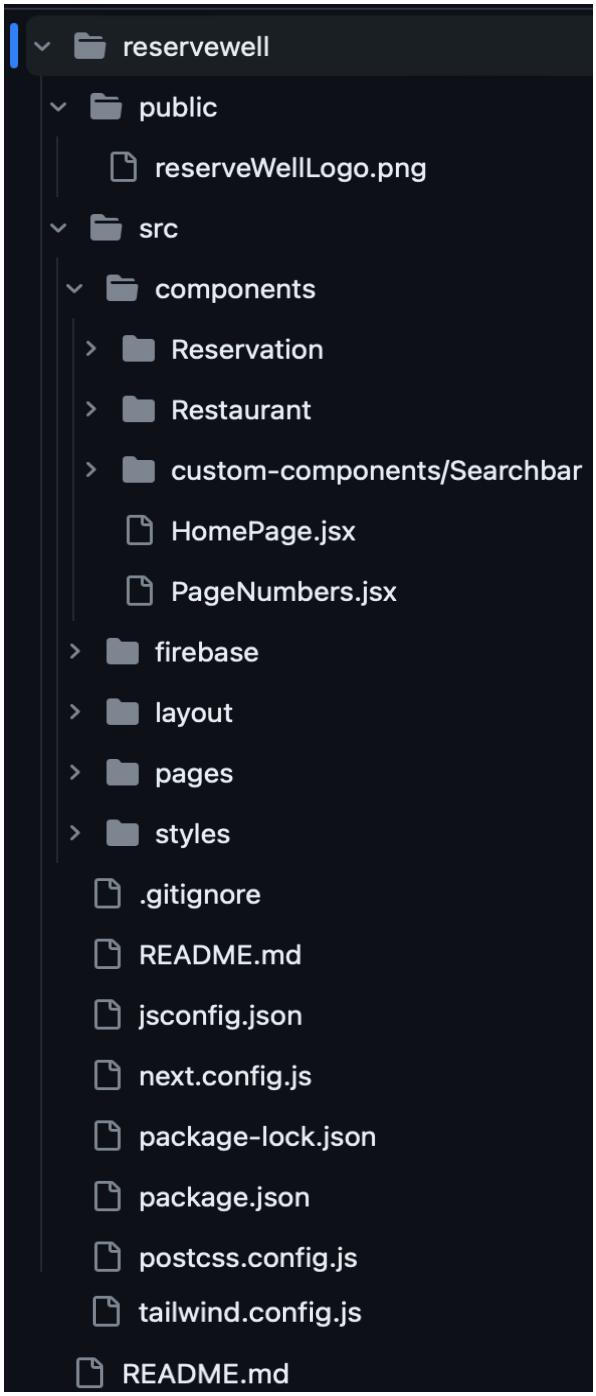
Public:

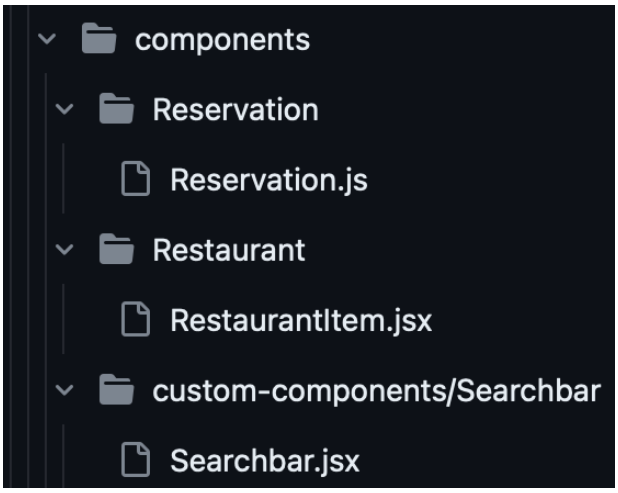
- **reserveWellLogo.png:** Logo image for ReserveWell.

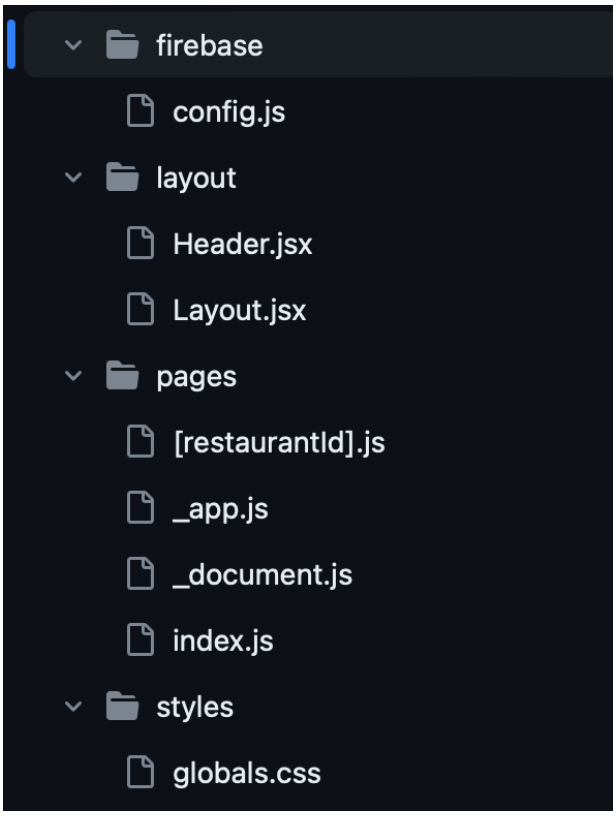
Src:

- **Components:**
  - **Reservation:**
    - **Reservation.js:** React component for managing reservations.
  - **Restaurant:**
    - **RestaurantItem.jsx:** React component representing a restaurant item.
  - **Custom Components/Searchbar:**
    - **Searchbar.jsx:** Custom React component for a search bar.
  - **HomePage.jsx:** React component for the home page.
  - **PageNumbers.jsx:** React component for displaying page numbers.
- **Firebase:**
  - **config.js:** Firebase configuration file.
- **Layout:**
  - **Header.jsx:** React component for the application header.
  - **Layout.jsx:** React component for the overall layout.
- **Pages:**
  - **[restaurantId].js:** Dynamic route page for displaying information about a specific restaurant.
  - **_app.js:** Custom App component to initialize pages.
  - **_document.js:** Custom Document component for server-side rendering.
  - **index.js:** Main page of the application.
- **Styles:**
  - **globals.css:** Global styles for the application.

GitHub file tree can be seen in Figure1, Figure2, Figure3







Data Files

In our project, we use Firebase Firestore. Our decision lies in its ability to provide a real-time, scalable, and secure database solution with minimal backend development, making it an attractive choice for modern web and mobile applications and team members' skill sets.

The data files include a variety of data types distributed across five collections:

1. **reservations:**
    - Documents store reservation details, including creation timestamp, date, group size, notes, restaurant ID, status array, time slot, and user ID.
2. **reservations_status:**
    - A single document holds an enumeration of reservation statuses, including "created," "updated," "completed," and "cancelled."
3. **restaurants:**
    - Documents represent restaurant information, encompassing address (geopoint), capacity, cuisine, maximum group size, name, rating, waitlist array, working days array, working hours array.
4. **users:**
    - Documents contain user details such as creation timestamp, email, full name, reservation counter, and user type (used for membership, with 0 indicating anonymous).
5. **waitlists:**
    - Documents store waitlist entries with creation timestamp, group size, restaurant ID, timeslot, and user ID.

This structured organization facilitates efficient management and retrieval of data related to reservations, restaurant details, user information, and waitlist entries within the Firebase Firestore database.

Example documents for each collection, with corresponding fields and datatypes are as follows:

Collection: reservations

- **Document ID: gWsiSRJjn40OOzTZopgx**
    - **created_at:** November 25, 2023 at 4:38:34 PM UTC+3 (timestamp)
    - **date:** "01.12.2023" (string)
    - **group_size:** "3" (string)
    - **notes:** "" (string)
    - **restaurant_id:** "tasty-burger-joint" (string)
    - **status:**
        - 0: "created" (string)
    - **time_slot:** "14:00-16:00" (string)
    - **user_id:** ""

Collection: reservations_status

- **Document ID: EY9e07TNAG6rc3alHAwQ**
    - **status:**
        - 0: "created" (string)
        - 1: "updated" (string)
        - 2: "completed" (string)
        - 3: "cancelled" (string)

Collection: restaurants

- **Document ID: vUVqdtjAjUmUxXr1Qohc**
  - **address:** [47.4979° N, 19.0402° E] (geopoint)
  - **capacity:** 40 (number)
  - **cuisine:** "casual" (string)
  - **max_group_size:** 12 (number)
  - **name:** "Twentysix_Restaurant" (string)
  - **rating:** 0 (number)
  - **waitlist:**
    - 0: "" (string)
  - **working_days:**
    - 0: "Monday" (string)
    - 1: "Tuesday" (string)
    - 2: "Wednesday" (string)
    - 3: "Thursday" (string)
    - 4: "Friday" (string)
    - 5: "Saturday" (string)
    - 6: "Sunday" (string)
  - **working_hours:**
    - 0: "10:00-12:00" (string)
    - 1: "12:00-14:00" (string)
    - 2: "14:00-16:00" (string)
    - 3: "16:00-18:00" (string)
    - 4: "18:00-20:00" (string)
    - 5: "20:00-22:00" (string)

Collection: users

- **Document ID: SxLEs0mBblgMibm6VLDU**
  - **created_at:** November 21, 2023 at 12:00:00 AM UTC+3 (timestamp)
  - **e_mail:** "tunaliezgimetu@gmail.com" (string)
  - **full_name:** "Ezgi Tunali" (string)
  - **reservations_counter:** 1 (number)
  - **type:** 0 (type will be used for membership, anonymous)

Collection: waitlists

- **Document ID: i1ys23vydw2SpsOlHxz6**
  - **created_at:** November 28, 2023 at 12:00:00 AM UTC+3 (timestamp)
  - **group_size:** 4 (number)
  - **restaurant_id:** "vUVqdtjAjUmUxXr1Qohc" (string)
  - **timeslot:** "21:00-21:30" (string)
  - **user_id:** "SxLEs0mBblgMibm6VLDU" (string)

Adjustments can be made based on specific requirements.

The Firebase Firestore security rules are as follows, which is convenient for development and testing. However, they are planned to be enhanced in upcoming iterations, to align with the principle of

least privilege, granting users only the permissions necessary for their specific tasks. By this way, better balance between functionality and security will be achieved.

Firebase FirseStore Rules

```
rules_version = '2';

service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read; allow write;
    }
  }
}
```

Build Scripts

The commands below, outline the process of setting up the development environment, creating the application, managing dependencies, and deploying it using Vercel, with an assumption that a text editor is available for use in the development environment.

1. **Installation of Node.js and Node Modules:**
    - Node.js is installed locally to run the application. Having Node.js installed is a prerequisite for both building and running the application. However, the node_modules folder, which is responsible for managing dependencies, is not pushed to GitHub. Therefore, anyone external to the project, needs to use *"npm install"* to install the required packages.
2. **Cloning the Repository:**
    - The repository is cloned from the source, by cloning from GitHub with *"git clone https://github.com/MustafaZemin/IS502_ReserveMasters.git"*
3. **Creating a Next.js Application:**
    - The command "*npx create-next-app"* is used to create a Next.js application. This command also includes the necessary packages with "*npm install",* making it convenient for developers.
4. **Adding Additional/External Packages:**
    - To add other external packages, the command is *"npm install package_name"* for each specific package.
5. **Building and Deploying with Vercel:**
    - Browsers cannot directly execute React code, so the source code needs to be built. The deployment is done using Vercel, a platform commonly used and created by Next.js developers. Vercel automates the build process by connecting to the GitHub repository and progressing in parallel.

Other Files

- **.gitignore:** Specifies files and directories to be ignored by version control.
- **README.md:** Project documentation providing an overview of the repository.
- **jsconfig.json:** Configuration file for JavaScript (JS) development in Visual Studio Code.
- **next.config.js:** Configuration file for Next.js.
- **package-lock.json:** Auto-generated file for package dependency versions.
- **package.json:** Configuration file for Node.js project dependencies and scripts.
- **postcss.config.js:** Configuration file for PostCSS, a tool for transforming styles.
- **tailwind.config.js:** Configuration file for Tailwind CSS, a utility-first CSS framework.