**Project Documentation: My Masjid Web Application**

**1. Project Overview**

The "My Masjid" web application is a client-side project designed to provide users with essential information related to daily prayers. The core functionality includes:

* **Prayer Times:** Displaying accurate prayer times for a user's selected masjid or their current location.
* **User Authentication:** Secure login and logout functionality using Firebase.
* **User Profile & Settings:** Allowing users to manage their profile and customize settings like the Asr calculation method.
* **Qibla Direction:** Calculating and displaying the Qibla direction based on the user's location.
* **Masjid Management:** Providing a list of masjids, with details and directions available for authenticated users.

The project is built with a clean, modular structure, ensuring maintainability and scalability.

**2. File Structure**

The project's file structure is organized to separate concerns and create a clear hierarchy.



**3. Application Flow**

**3.1. Startup Flow**

1. A user loads index.html in their browser.
2. The browser executes main.js, which is imported as a module.
3. main.js imports all necessary functions and the Firebase services from firebase-config.js.
4. The onAuthStateChanged listener is set up, which checks if a user is already logged in. This dynamically updates the header with either "Login" or the user's name and profile options.
5. populateMasjidDropdown() is called, which fetches all masjids from Firestore and populates the <select> element.
6. displayPrayerTimes() is called initially to either fetch times for the first masjid or a fallback based on the user's current location.
7. The application's clock and Hijri date are started via updateCurrentTime() and updateHijriDate().

**3.2. User Interaction Flows**

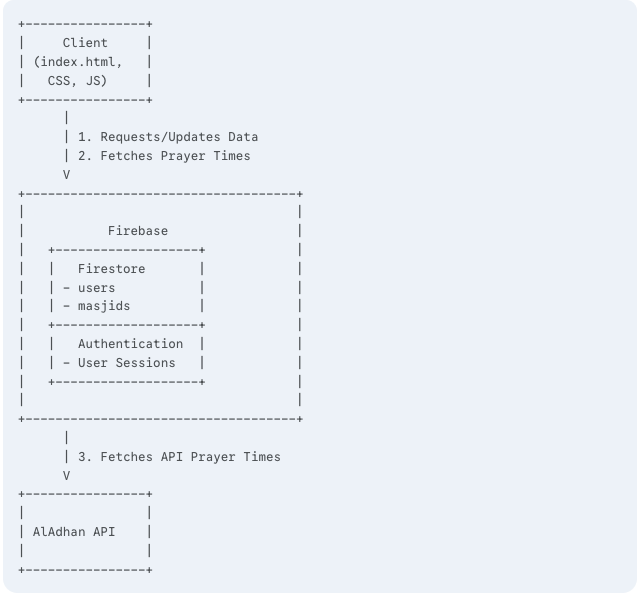
* **Selecting a Masjid:**
  1. The user selects a masjid from the dropdown.
  2. An event listener on the masjid-select element fires.
  3. displayPrayerTimes() is called with the selected masjid's ID.
  4. displayPrayerTimes() queries Firestore for the specific masjid's details and prayer times, then renders them.
* **Changing Settings:**
  1. The user clicks "Settings" in the profile dropdown.
  2. displaySettings() is called, which fetches the user's asrCalculationSchool from Firestore and sets the radio button.
  3. The user changes the setting and clicks "Save."
  4. saveSettings() updates the user's document in Firestore.
  5. displayPrayerTimes() is called again with the new setting to refresh the prayer times display.

**4. Connectivity**

* **Firebase Firestore:** The primary database for the application.
  + **Masjids:** A masjids collection stores details for each masjid, including their name, address, location coordinates, and prayer times. This data is static and managed by an administrator.
  + **Users:** A users collection stores additional profile data for each authenticated user, linked by user.uid. This includes firstName, lastName, and custom settings like asrCalculationSchool.
* **Firebase Authentication:** Handles all user session management, including login status and user identification.
* **AlAdhan API:** Used as a fallback. If a user has not selected a specific masjid or if a masjid's prayer times are not available in Firestore, the application fetches daily prayer times based on the user's location via the AlAdhan API.

**5. High-Level Design (HLD)**

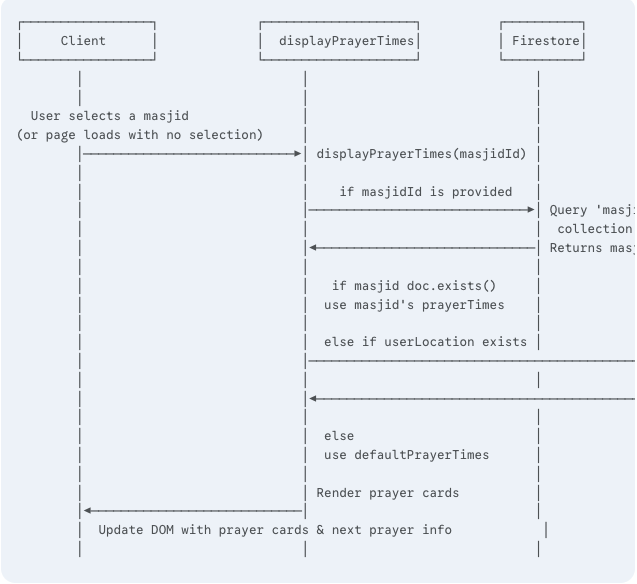
The HLD illustrates the primary components and their interactions.



**Flow Description:** The **Client** is the user interface. It communicates with **Firebase** to handle user sessions and retrieve masjid and user data from Firestore. For prayer times, it first tries to get data from Firestore. If that fails, it queries the **AlAdhan API** for location-based times.

**6. Low-Level Design (LLD): Displaying Prayer Times**

This sequence diagram details the displayPrayerTimes function, a core part of the application's logic.



**7. Database Schema (Firestore)**

**masjids collection**

Each document in this collection represents a single masjid. The id of the document is the masjidId used in the application.

* name: string
* address: string
* city: string
* state: string
* contact: string
* latitude: number (or string)
* longitude: number (or string)
* prayerTimes: object
  + Fajr: object
    - azanTime: string ('HH:mm')
    - prayerTime: string ('HH:mm')
  + Dhuhr: object
    - azanTime: string ('HH:mm')
    - prayerTime: string ('HH:mm')
  + Asr: object
    - azanTime: string ('HH:mm')
    - prayerTime: string ('HH:mm')
  + Maghrib: object
    - azanTime: string ('HH:mm')
    - prayerTime: string ('HH:mm')
  + Isha: object
    - azanTime: string ('HH:mm')
    - prayerTime: string ('HH:mm')

**users collection**

Each document in this collection represents a user's profile data. The id of the document is the uid provided by Firebase Authentication.

* firstName: string
* lastName: string
* email: string
* role: string (e.g., 'user', 'admin')
* asrCalculationSchool: string ('0' for Hanafi, '1' for Shafi'i)