EXPRESS AND FIREBASE NOTES

The following document is a compilation of my notes for studying Express and Firebase. As I have been learning about these technologies, I have been taking notes on important concepts, commands, and code snippets that I have found useful. This document serves as a collection of these notes, organized by topic, in order to provide a comprehensive reference for myself as I continue to learn and work with these tools. I hope that these notes may also be helpful to others who are new to Express and Firebase and are looking for a concise and organized resource to help them get started.

1 Creating a Server

To create an Express server using JavaScript, follow these basic steps:

1. Import the Express module:

```
const express = require('express');
```

2. Create an instance of the Express application:

```
const app = express();
```

3. Define the routes for handling incoming requests. This is done using the 'app' object's methods such as 'get()', 'post()', 'put()', 'delete()', etc. Here's an example of defining a route that handles GET requests:

```
app.get('/', (req, res) => {
  res.send('Hello, World!');
});
```

4. Start the server by listening on a specified port using the 'listen()' method:

```
const PORT = 3000;
app.listen(PORT, () => {
  console.log(`Server listening on port ${PORT}`);
});
```

With these steps, you should have a simple Express server that can handle incoming requests and respond to them accordingly. Of course, there are many more advanced features and techniques you can use with Express, but these basic steps should get you started. Now if you visit the Express server at localhost:3000, you should be able to see the "Hello, World!" string you entered in the res.send() function.

2 HTTP Requests

In the previous example, we demonstrated the use of the GET request to handle incoming requests. The GET request is one of the HTTP request methods used to retrieve data from a server. In addition to the GET request, there are other HTTP request methods that can be used to perform different actions on server data.

Here's a list of some of the other common HTTP request methods including GET:

- **GET:** retrieves a resource from the server. This is typically used for reading data from the server, such as fetching a list of items or a specific item.
- **POST:** Used to submit data to be processed to a specified resource
- **PUT:** Used to update a specified resource with new data
- **DELETE:** Used to delete a specified resource
- **PATCH:** Used to update a part of a specified resource with new data

Each of these request methods has a specific use case and can be used to perform different actions on server data. This is all nice and dandy, but we also want to be able to create a Firebase database so that we can successfully run CRUD operations on our application.

3 Firebase Configuration

To connect to Firebase and use Firestore, the first step is to install the firebase-admin package in your project.

You can do this by running the following command in your terminal:

```
npm install firebase-admin --save
```

After the package is installed, you need to create a new file called firebase.js in your project. This file will contain the code that sets up the connection to Firebase and exports a Firestore client instance that can be used throughout your application We will now enter the following code within the firebase.js file:

```
// Import the necessary functions from the Firebase Admin App library
const { initializeApp, cert } = require('firebase-admin/app')

// Import the necessary function from the Firebase Firestore library
const { getFirestore } = require('firebase-admin/firestore')

// Load the service account credentials from a JSON file
const serviceAccount = require('./creds.json')

// Initialize the Firebase Admin App with the service account credentials
initializeApp({
    credential: cert(serviceAccount)
})

// Get a Firestore client instance
const db = getFirestore()

// Export the Firestore client instance for use in other parts of the application
module.exports = { db }
```

This code is a simple Firebase admin module that sets up a Firestore client instance and exports it for use in other parts of the application.
Here's a breakdown of the code:

1. **firebase-admin/app**: the Firebase Admin App library.
2. **firebase-admin/firestore**: the Firebase Firestore library.
3. The **serviceAccount** object is loaded from a JSON file, which contains the credentials for the Firebase project.

4. The **initializeApp**: function is called and passed the **serviceAccount** object as the **credential** option. This function initializes the Firebase Admin App with the given credentials.

5. The **getFirestore**: function is called to get a Firestore client instance.

6. Finally, the **db** object is exported as a module so that it can be used in other parts of the application.

The overall goal of this code is to provide a centralized way of setting up a Firestore client instance that can be reused throughout the application, rather than having to set up the Firestore client in every module that needs it. This makes it easier to manage the Firestore client and makes the code more organized and maintainable.