# **GNotes app Project Report**

## **Project Overview**

* **Project Name:** GNotes
* **Project Duration:** [07.09.2023] - [Present]
* **Project Manager:** [201145 Georg Mucunski]

## **Project Goals and Objectives**

* **Goal 1:** Create a note-taking web application with user authentication.
* **Goal 2:** Implement Firestore as the backend database for storing notes.
* **Goal 3:** Provide a user-friendly interface for creating, editing, and managing notes.
* **Goal 5:** Deploy the application to Firebase Hosting for public access.

## **Project Features**

* User Authentication
* Users can sign in using their Google accounts.
* Private notes are associated with the user's account.
* Note Management
* Users can create new notes.
* Users can edit existing notes.
* Users can delete notes.
* Text Formatting
* Users can align text (left, center, right).
* Firestore Integration
* Firestore is used as the backend database.
* Private and public notes are stored in Firestore collections.
* The application is deployed to Firebase Hosting.

## **Development Process**

* **User Authentication: Sign-In, Create Account, Sign-In with Google Popup, Log Out, Private Notes, Public Notes, Toggle, Delete, Create New note, Title Implementation**
* **Sign In**
* Retrieve user input values for email and password fields.
* Use the **signInWithEmailAndPassword()** method to authenticate the user with the provided email and password.
* Handle authentication success and errors using **.then()** and **.catch()**.
* Display appropriate error messages if the authentication fails.
* **Sign Up**
* Retrieve user input values for email, password, and username fields.
* Create a new user account using the **createUserWithEmailAndPassword()** method with the email and password.
* Handle account creation success and errors using **.then()** and **.catch()**.
* Store user information (email, password, username, timestamp) in the Firebase Realtime Database.
* **Sign In with Google Popup**
* Implement Google Sign-In using **signInWithPopup()** with a GoogleAuthProvider instance.
* Implement the **handleGoogleSignIn()** method to allow users to sign in using Google Authentication.
* Handle the successful sign-in event, extract user data, and store it in the Firebase Realtime Database.
* **Log Out**
* Attach an event listener to the "Log Out" button.
* Use the **signOut()** method to log the user out.
* Update the UI to reflect the user's logged-out state.
* **User Interface**
* Design and develop the user interface for the GNotes app.
* Implement features for creating, editing, and deleting notes.
* Implement text alignment (left, center, right).
* **Authentication and User Management**
* Implement user sign-in and sign-out functionality.
* Use Firebase Authentication for user authentication.
* Create functions for:
* **handleGoogleSignIn()**: Signs in users with Google.
* **handleSignOut()**: Signs out users.
* **isUserAuthenticated()**: Checks if a user is authenticated.
* **Note Table Component**
* Develop a Note Table component to display a list of notes.
* Retrieve user-specific notes from Firestore and display them in the table.
* Implement functions for:
* **handleNoteSelect(note)**: Sets the selected note.
* **handleDeleteNote(noteToDelete)**: Deletes a note.
* **handleCreateNote()**: Creates a new note.
* **Note Selection**
* Implement the **handleNoteSelect)()** method to set the selected note when a user clicks on a note in the table.
* **Note Creation**
* Implement the **handleCreateNote()** method to create a new note.
* Create a new note object with a unique ID, title, and empty content.
* Determine whether to add the new note to the **privateNotes** or **publicNotes** collection based on the active toggle.
* **Toggle between Private and Public Notes**
* Implement the **handleToggleChange()** method to switch between private and public notes.
* Update the active toggle state accordingly (**activeToggle**).
* **Create New Note**
* Implement the ability to create new notes.
* Allow users to set note titles and initial content.
* Save new notes to Firestore under the user's collection.
* Create a function:
* **createNewNote()**: Creates a new note in Firestore.
* **Edit Note Content**
* Develop a note editor component for editing the content of selected notes.
* Enable users to format text (e.g., bold, italics, underline).
* Allow users to change text alignment (left, center, right).
* Implement functions for:
* **handleNoteContentChange(event)**: Updates the content of the selected note.
* **handleTitleChange(event)**: Updates the title of the selected note.
* **handleTextAlignment(alignment)**: Changes text alignment.
* **Note Content Editing**
* Implement the **handleNoteContentChange()** method to update the content of the selected note.
* Update the content in the state and Firestore based on the active toggle.
* Ensure that changes in content are reflected in real-time in the Firestore database.
* **Title Editing**
* Implement the **handleTitleChange()** method to update the title of the selected note.
* Update the title in the state.
* **Title Blur Event**
* Implement the **handleTitleBlur()** method to update the title in the notes array when the input field loses focus (blur).
* **Text Alignment**
* Implement the **handleTextAlignment()** method to change the text alignment of the selected note (left, center, right).
* Update the text alignment in the state and Firestore based on the active toggle.
* **Delete Notes**
* Implement the ability to delete selected notes.
* Remove notes from Firestore upon deletion.
* Implement a function:
* **deleteNote(noteToDelete)**: Deletes a note from Firestore.
* **Note Deletion**
* Implement the **handleDeleteNote()** method to delete a note.
* Determine the collection (**privateNotes** or **publicNotes**) to delete the note from based on the active toggle.
* Delete the note from Firestore using the **doc** method.
* Update the state by filtering out the deleted note from the respective notes array.
* Clear the selected note if it matches the deleted note.
* **Firestore Integration**
* Integrate Firestore as the backend database for storing notes.
* Create Firestore collections for private and public notes.
* Implement Firestore security rules to restrict access to user-specific data.
* **Firestore Collections**
* Create two Firestore collections: **privateNotes** and **publicNotes** for storing private and public notes, respectively.
* **Deployment**
* Deploy the GNotes app to Firebase Hosting.
* Make the app accessible to users online.
* **User Authentication Persistence**
* Check if the user is currently logged in with Google. If not, retrieve their username from the Firebase Realtime Database.
* **User Authentication State**
* Use the **useEffect()** hook to listen for changes in the user's authentication state.
* When a user signs in, they fetch their private notes from Firestore.
* Fetch public notes as well.
* **User Sign Out**
* Implement the **handleSignOut()** method to allow users to sign out.
* Clear user data, private notes, public notes, and the selected note when signing out.
* **Error Handling**
* Implement error handling for various authentication and database operations.
* Display error messages to users when necessary.
* This development process outlines the key steps required to build the GNotes App, including user authentication, Firebase integration, user interface development, and deployment. It also includes error handling and user state persistence for a seamless user experience.

## **Challenges Faced**

* **Challenge 1:** Implementing real-time synchronization of notes between users.
* **Challenge 2:** Configuring Firestore security rules to restrict access to user-specific data.
* **Challenge 3:** Styling and responsive design for various screen sizes.

## **Lessons Learned**

* **Lesson 1:** Firebase provides powerful tools for authentication and real-time data storage.
* **Lesson 2:** Collaborative development in React.js requires effective component management and state handling.
* **Lesson 3:** User experience design is crucial for note-taking applications.

## **Future Improvements**

* Implement collaborative note editing with real-time synchronization.
* Add support for file attachments and multimedia within notes.
* Enhance security and access control mechanisms.
* Optimize app performance for larger datasets.

## **Conclusion**

The GNotes App project successfully achieved its goals of creating a user-friendly note-taking application with authentication and Firestore integration. The app demonstrates effective use of React.js, Material-UI, Firebase, and Firestore. Further enhancements and improvements can be made to enhance the app's functionality and scalability.

The app has a lot of room for improvement that will enable a more connected user experience. There are a lot of features that can be implemented to enhance the note creating and editing process. This project is a mock-up of what could be possible with seamless integration of a simple note application. The design and functionalities were hindered by the limited time remaining in finishing the project.

In conclusion, the development of the GNotesApp has been a journey marked by significant progress and achievement. This report has provided a comprehensive overview of the entire development process, from project initiation to the implementation of key features, such as user authentication, private and public notes management, and real-time collaboration.

In summary, the development of the Note App has been a successful endeavor, combining robust functionality with an appealing user interface. It has the potential to become a versatile platform for notetaking and collaboration. The focus on user authentication, data security, and real-time capabilities positions the app to meet the evolving needs of users. As development continues, incorporating user feedback and exploring innovative features will be key to its success.