RepliDocs App-> My replica to Google Docs App using Flutter, also using Node.js, Express Sockets, MongoDB, Riverpod, and various techniques. The replica application contains methods of authenticating users with Google, keeping users logged in, creating RestAPI to create and display new documents, building a powerful routing system to share documents links, and establishing socket connection to allow users to collaborate.

Functionalities/Demo:

- Contains a neat navbar.
- Allows users to sign in, sign out, authenticate with Google and staying logged in.
- Creating a New Document.
- Displaying all documents created by the user.
- Contains a well-designed Document Screen.
- Contains Collaborative Editing.
- Auto-Save functionality works.
- Sharing Link works.

Developing Languages, Tools, and Techniques Needed:

Flutter Dart Code v3.52.0 Vscode Extension
https://marketplace.visualstudio.com/items?itemName=Dart-Code.flutter
Flutter SDK https://docs.flutter.dev/get-started/install/macos
Vscode 1.73 https://code.visualstudio.com/updates/v1_73
Dart programming language https://dart.dev
dart google_sign_in dev dependency 5.4.2 https://pub.dev/packages/google_sign_in
Google Console Cloud API
https://console.cloud.google.com/projectselector2/apis/dashboard?pli=1&supportedpurview=project
Xcode 14.1
https://developer.apple.com/documentation/xcode-release-notes/xcode-14_1-release-notes
Flutter Riverpod Dependency
https://codewithandrea.com/articles/flutter-state-management-riverpod/
Node.JS v18.12.0 https://nodejs.org/en/
NPM JS https://www.npmjs.com
Socket Express https://socket.io/get-started/chat/
JSON Web Token https://www.npmjs.com/package/jsonwebtoken
Mongoose https://www.npmjs.com/package/mongoose
MongoDB Cloud https://www.mongodb.com
Thunder Client Vscode extension v2.0.0 https://www.thunderclient.com/
JSON Web Tokens https://jwt.io/
routemaster https://pub.dev/packages/routemaster
Flutter Riverpod Snippets
https://marketplace.visualstudio.com/items?itemName=robert-brunhage.flutter-riverpod-snippets
AutoDraw https://www.autodraw.com/
Flutter Quill https://pub.dev/packages/flutter_quill
socket_io_client https://pub.dev/packages/socket_io_client

Prerequisites & Setup:

Install Flutter Dart Code extension in Vscode empty starting page.

Install Flutter SDK to the desired local directory.
Configure and export Flutter to the local path in Console:
`export PATH="$PATH:`pwd`/flutter/bin"`
Use `Command Palette` in Vscode-> Flutter -> Create a new project -> Application.
To avoid macOS “zsh command not found” error:
In Console, run `vim $HOME/.zshrc`
Press the "I" key to go into INSERT mode.
Add the following line in the opened file:
`export PATH=$PATH:/Desktop/flutter/bin`
Press "Esc" then write `:wq!` in terminal and press enter to exit vim.
Restart the created Flutter project in Vscode.
In Vscode Terminal**, navigate to the Flutter project first:**
`cd FLUTTER_PROJECT_NAME`
Then run the Flutter project with:
`flutter run`
Choose Chrome Web/macOS Safari for testing.
Done setting up when **Flutter Demo** web server successfully launched in browser.
at http://localhost:52767/#/
flutter project web server done setting up.PNG
Synchronous Developing Notes:
Pass Google logo for the signin page in `login_screen.dart`:

```
import 'package:flutter/material.dart';
class LoginScreen extends StatelessWidget {
  const LoginScreen({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Center(
        child: ElevatedButton.icon(
          onPressed: () {},
          icon: Image.asset('assets/images/google-icon.png'),
          label: const Text(
            'Sign in with Google'
```

google icon imported.PNG
Resize it and we got:
google icon resized to height 20.PNG
Create OAuth client ID with Google Console Cloud API.
NOTE: Actively use command `flutter clean` and `flutter run` to rebuild from root.
Configure for Android and append `google_services.JSON`
Configure for iOS: Open it with Xcode-> input bundle ID-> download plist.
Configure web applications refer to the pub dev page.
Set preferable localhost at port 3000.
Now in Vscode run:

```
flutter run -d chrome --web-port 3000
```

Now passed the web server at localhost:3000 successfully:

web server passed to localhost 3000.PNG

Configure Google sign in using Node.js:

Install Riverpod dependency:

add `flutter_riverpod: ^2.0.0-dev.9` in `pubspec.yaml`.

Run `dart pub get` in the Vscode terminal.

Import Riverpod in `auth_repository.dart`:

```
import 'package:flutter_riverpod/flutter_riverpod.dart';
```

Add Riverpod provider for login auth:

```
final authRepositoryProvider = Provider(
  (ref) => AuthRepository(
```

…

Run flutter web server again. The google account sign-in page popped:

google account sign-in page popped.PNG

Sign in to my google account:

signed in.PNG

Use npm in Vscode to install Express socket, JSONWEBTOKEN and Mongoose:

```
npm i express http socket.io@2.3.0 jsonwebtoken mongoose
```

Install Nodemon:

```
npm i nodemon --save-dev
```

Use `npm run dev` to start the server at port 3001 based on `index.js`:

```
const express = require("express");
const mongoose = require("mongoose");
const PORT = process.env.PORT | 3001;
const app = express();
app.listen(PORT, "0.0.0.0", () => {
    console.log(`connected at port ${PORT}`);
});
```

server connected.

Connect to MongoDB:

In `index.js`:

```
mongoose
    .connect(DB)
    .then(() => {
        console.log("Connection successful!");
    })
    .catch((err) => {
        console.log(err);
    })

//async -> await
```

```
// .then((data) => print(data))
app.listen(PORT, "0.0.0.0", () => {
    console.log(`connected at port ${PORT}`);
});
```

Set up MongoDB and run the server `npm run dev` again:

mongodb connection successfully.PNG

Error: `Cannot run with sound null safety, because the following dependencies don't support null safety: - package:http - package:http_parser For solutions, see https://dart.dev/go/unsound-null-safety`

DEBUGGING: Update flutter http to the latest version.

Configure local private IP address to be redirect to signup page at port 3001:

Use command in local terminal:

```
ipconfig getifaddr en0/1
```

To obtain the local private IP address, and in `constants.dart`:

```
const host = `<ip address>:3001`
```

Then in `auth_repository.dart`, pass host:

```
  var res = await _client.post(Uri.parse('$host/api/signup'),
            body: userAcc.toJson(),
            headers: {
              'Content-Type': 'application/json; charset=UTF-8',
            });
```

Now rerun both node and dart terminals on vscode:

redirected to signup 3001 page.PNG

Use `flutter pub add shared_preferences` in dart terminal to install shared_pref lib.

Use `flutter pub add routemaster` to install routemaster.

Create new documents:

XHTTPEmpty Error: Failed to connect to Mongoose and API server to get response.

DEBUGGING: Reconnect to Mongoose-> Node terminal-> `npm run dev` -> Locate errors and debug errors -> Thunder Client -> Send new Request http://localhost:3001/api/signup -> Node run again-> mongoose reactivated -> Dart terminal run.

Set up id and routers when creating new document in `router.dart`:

```
final loggedInRoute = RouteMap(routes: {
  '/': (route) => const MaterialPage(child: HomeScreen()),
  '/document/:id/:somethingelse': (route) => MaterialPage(
        child: DocumentScreen(
          id: route.pathParameters['id'] ?? '',
        ),
      ),
```

new created document id generated when click to create new document.PNG

Parse Uri of my docs in `document_repository.dart`:

```
 Uri.parse('$host/docs/me'),
```

new document id showed.PNG

list of untitled documents parsed.PNG

Design document sharing button in `document_screen.dart`:

```
Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: kWhiteColor,
        elevation: 0,
        actions: [
          ElevatedButton.icon(
            onPressed: () {},
            icon: const Icon(
              Icons.lock,
              size: 16,
```

document sharing button.PNG

Design document title in `document_screen.dart`:

```
 title: Row(
          children: [
            Image.asset(
              'assets/images/google-docs-logo.png',
              height: 40,
            ),
            const SizedBox(width: 10),
            SizedBox(
              width: 180,
              child: TextField(
                controller: titleController,
                decoration: const InputDecoration(
                  border: InputBorder.none,
                  contentPadding: EdgeInsets.only(left: 10),
```

document title design.PNG

Use `flutter pub add flutter_quill` to install flutter quill.

Import Quill editor:

```
import 'package:flutter_quill/flutter_quill.dart' as quill;
   body: Column(
          children: [
            quill.QuillToolbar.basic(controller: _controller),
            Expanded(
              child: quill.QuillEditor.basic(
                controller: _controller,
                readOnly: false, // true for view only mode
```

quill editor banner showed.PNG

Make the text editor canvas align at center:

```
 body: Center(
        child: Column(
          children: [
            quill.QuillToolbar.basic(controller: _controller),
            Expanded(
              child: SizedBox(
                width: 750,
                child: Card(
                  color: kWhiteColor,
                  elevation: 5,
                  child: quill.QuillEditor.basic(
                    controller: _controller,
                    readOnly: false, // true for view only mode
```

text editing canvas align at center.PNG

bold, italic, underline, different text colors and various text effects testing:

various text effects testing.PNG [MAKE IT VISIBLE IN README].

Install socket_io_client with:

```
 flutter pub add socket_io_client
```

generate public sharing link in `document_screen.dart`:

```
Clipboard.setData(ClipboardData(
             text:

'http://localhost:3000/#/document/${widget.id}'))
              .then(
                 (value) {
         ScaffoldMessenger.of(context).showSnackBar(const SnackBar(
                       content: Text(
                         'Link copied!'
```

Now click on Share button, a public sharing link will be generated.