CSE470: Software Engineering
Project Report
Project Title:

Interactive Storytelling Platform

Group No-06, CSE470 Lab Section-13, Spring 24

ID	**Name**
21301080	Md Sayem Mottakee
21301016	Ariful Islam
21301088	Md Muntasir Mahmud Amit
23241085	Omor Bin Amjad Chowdhury

Table of contents

Name	**Role**	**Development Roles**	**Remark**
Ariful Islam	**Scrum Leader**	Database	
Md Sayem Mottakee	Team Member	Signup Page	Remark
Md Muntasir Mahmud Amit	Team Member	Login	
Omor Bin Amjad Chowdhury	Team Member	Forget password	Remark

Introduction

The **Interactive Storytelling Platform** aims to create an online community where users can create, share, and interact with storytelling content. The primary goal is to provide a collaborative environment for writers, readers, and other content creators. This project provides students with hands-on experience in software development, following Agile methodologies to build a fully functional software application from scratch.

Functional requirements

Functional requirements are divided into multiple modules:

Module-1:

1. Login
2. Signup/Registration
3. Password reset/Forgot password
4. Firebase connection with the project

Module-2:

5. First requirement
6. Second requirement
7.
8.
9.

Module-3:

10. First requirement
11. Second requirement
12.
13.
14.

Module-4:

15. First requirement
16. Second requirement
17.
18.
19.
20.

User Manual

The user manual will guide users through the platform, including account setup, story creation, engaging with other users, and using advanced features like chatbots and notifications. The user manual provides step-by-step instructions for users to navigate the platform. It covers:

- **Account Management**: How to sign up, log in, reset passwords, and manage profiles.
- **Story Creation**: Instructions on how to create, edit, and share stories.
- **Engagement**: How to participate in group chats, send direct messages, and interact with content.
- **Advanced Features**: Using chatbots, setting up notifications, and managing privacy settings.

Frontend Development

The frontend is developed using **Flutter**, a UI toolkit for building natively compiled applications for mobile, web, and desktop from a single codebase. Key aspects include:

- **User Interface**: Designing a responsive and intuitive interface.
- **User Experience**: Ensuring seamless navigation and interaction.
- **Cross-Platform Compatibility**: Ensuring the application works on different devices and screen sizes.6

Backend Development

The backend utilizes **Firebase**, providing robust backend services such as real-time databases, authentication, and cloud storage. Key components include:

- **Database Management**: Real-time synchronization of user data and stories.
- **Authentication**: Secure user authentication and authorization.
- **Cloud Functions**: Serverless functions for handling backend logic.

Database

The database is structured to store and manage user data, stories, multimedia content, and interactions efficiently. Using Firebase, the database ensures:

- **Scalability**: Handles large volumes of data and high traffic.
- **Real-Time Updates**: Immediate synchronization of changes across all clients.
- **Security**: Encrypted data storage and access controls.

Technology (Framework, Languages)

- **Visual Studio Code**: Integrated Development Environment (IDE)
- **Dart**: Programming language for Flutter
- **Flutter**: UI toolkit for cross-platform development
- **Firebase**: Backend services including real-time database and authentication
- **HTML/CSS**: For additional web content styling
- **Figma Design**: For creating responsive design templates
- **REST API**: For external service integration

GitHub Repo Link

A GitHub repository will be maintained for version control and collaboration. The repository link will be shared with the section faculty and used for project tracking and evaluation.

https://github.com/MdSayemMottakee/Interactive-Storytelling-Platform

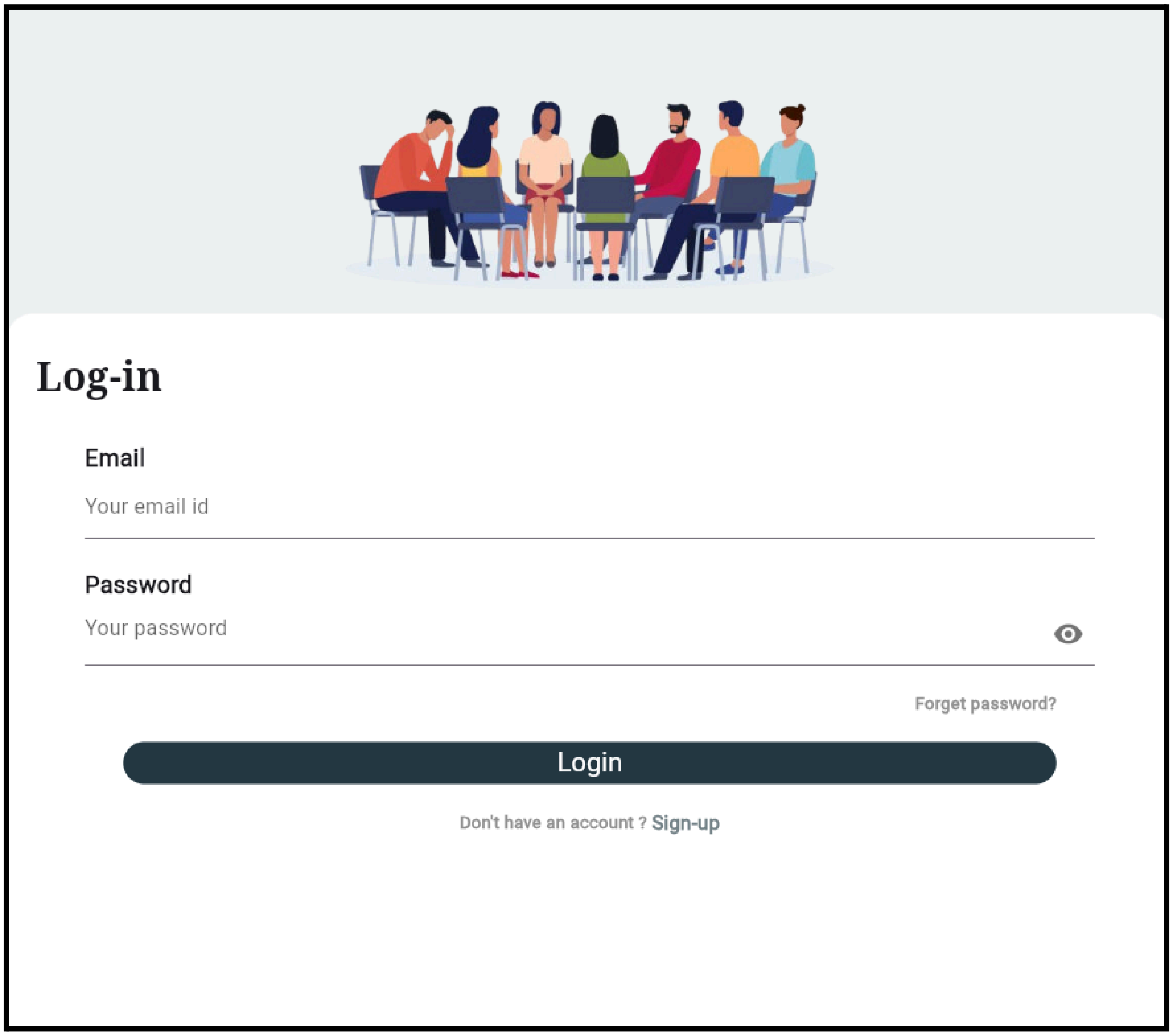
Individual Contribution

Each team member's specific contributions will be documented, detailing their roles and tasks during the project. This ensures accountability and highlights individual efforts in the project development.

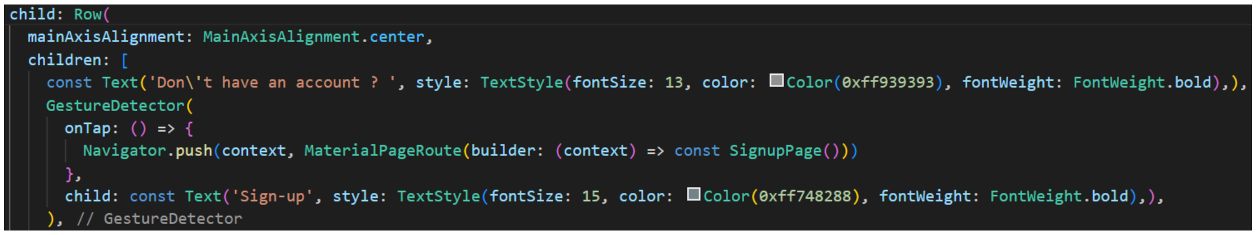
Login page:

The login feature allows users to authenticate themselves by providing their credentials (typically an email and password) to access the services. This feature is crucial for personalizing the user experience, maintaining user-specific data, and ensuring secure access to the app.

The login screen consists of input fields for the user's email and password. There are buttons for logging in, Sign-up a new account, and resetting the forgotten password.

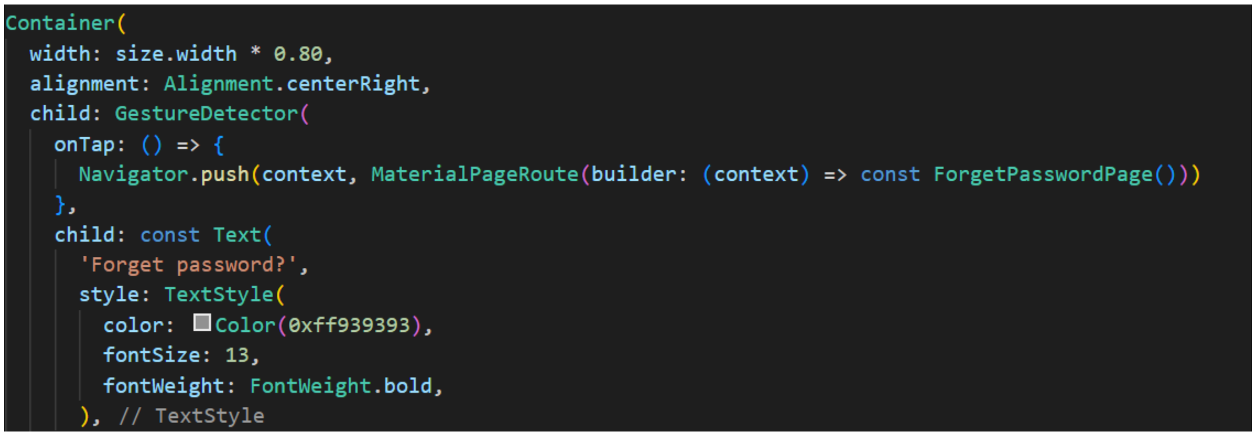


Using the following code we are going to sign-up.



```
child: Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    const Text('Don\'t have an account ? ', style: TextStyle(fontSize: 13, color: Color(0xff939393), fontWeight: FontWeight.bold),),
    GestureDetector(
      onTap: () => {
        Navigator.push(context, MaterialPageRoute(builder: (context) => const SignupPage()))
      },
      child: const Text('Sign-up', style: TextStyle(fontSize: 15, color: Color(0xff748288), fontWeight: FontWeight.bold),),
    ), // GestureDetector
```

Using the following code we are going to the password reset page.



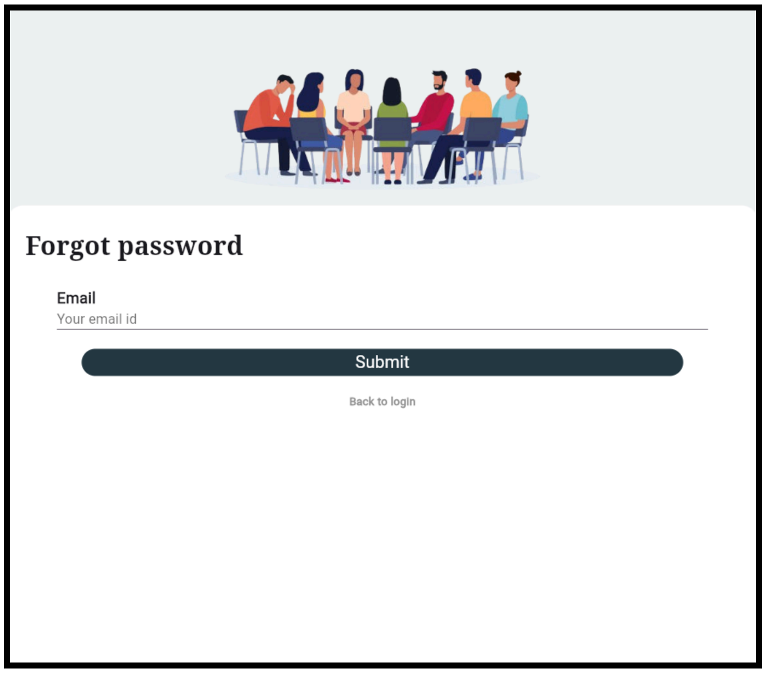
```
Container(
  width: size.width * 0.80,
  alignment: Alignment.centerRight,
  child: GestureDetector(
    onTap: () => {
      Navigator.push(context, MaterialPageRoute(builder: (context) => const ForgetPasswordPage()))
    },
    child: const Text(
      'Forget password?',
      style: TextStyle(
        color: Color(0xff939393),
        fontSize: 13,
        fontWeight: FontWeight.bold,
      ), // TextStyle
```

Forget password:

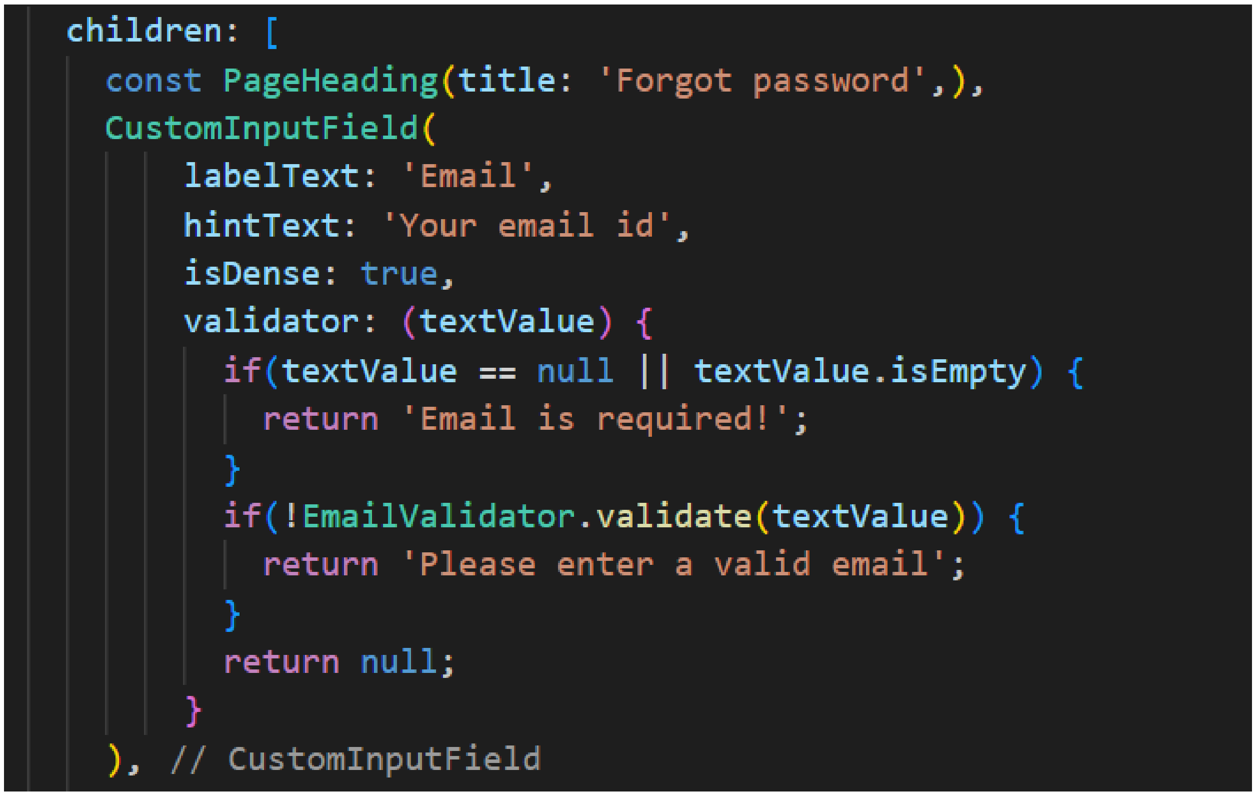
Password reset functionality allows users to regain access to their accounts if they forget their passwords. In a typical flow, users request a password reset, receive a reset link via email, and then use that link to set a new password.

How It Works

1. **User Requests Password Reset**:
   - The user navigates to the reset password screen.
   - They enter their email address and submit the request.
2. **Server Sends Password Reset Email**:
   - The authentication service (Firebase in this case) sends an email containing a password reset code to the provided email address.
3. **User Receives Email**:
   - The user receives the email and uses it to set the new password.
4. **User Resets Password**:
   - The user then can enter a new password.
   - The new password is saved, and the user can now log in with the new password.



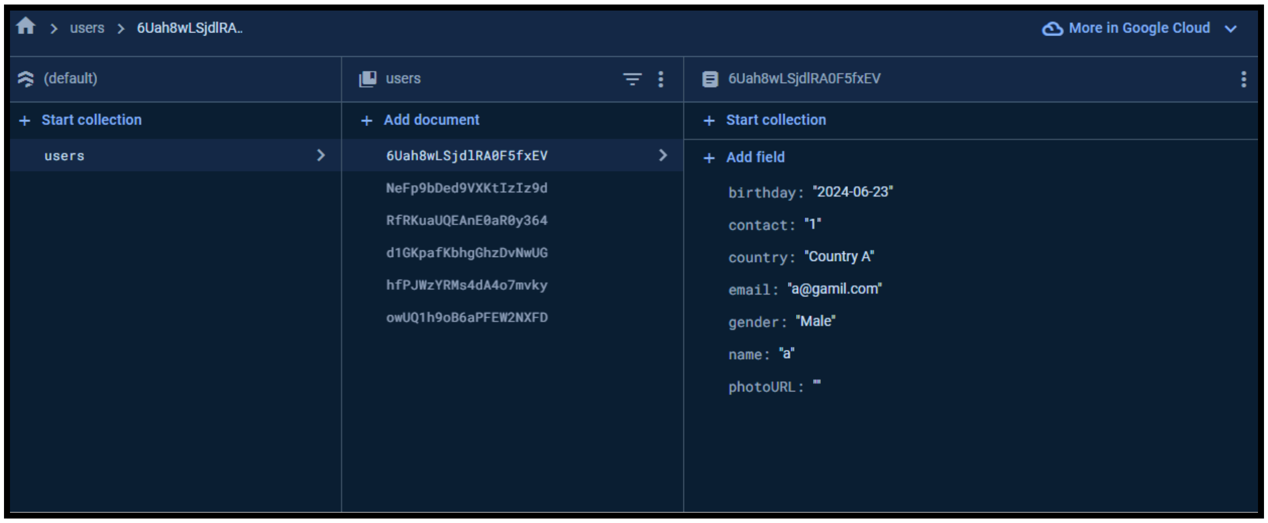
Future Development:



```
children: [
  const PageHeading(title: 'Forgot password',),
  CustomInputField(
      labelText: 'Email',
      hintText: 'Your email id',
      isDense: true,
      validator: (textValue) {
        if(textValue == null || textValue.isEmpty) {
          return 'Email is required!';
        }
        if(!EmailValidator.validate(textValue)) {
          return 'Please enter a valid email';
        }
        return null;
      }
  ), // CustomInputField
```

- We will enhance the UI for reset confirmation and error handling for a smoother user experience.
- We will Implement client-side validation to ensure valid email format before submitting reset requests.

Firebase:



users > 6Uah8wLSjdlRA..

More in Google Cloud

(default) | users | 6Uah8wLSjdlRA0F5fxEV

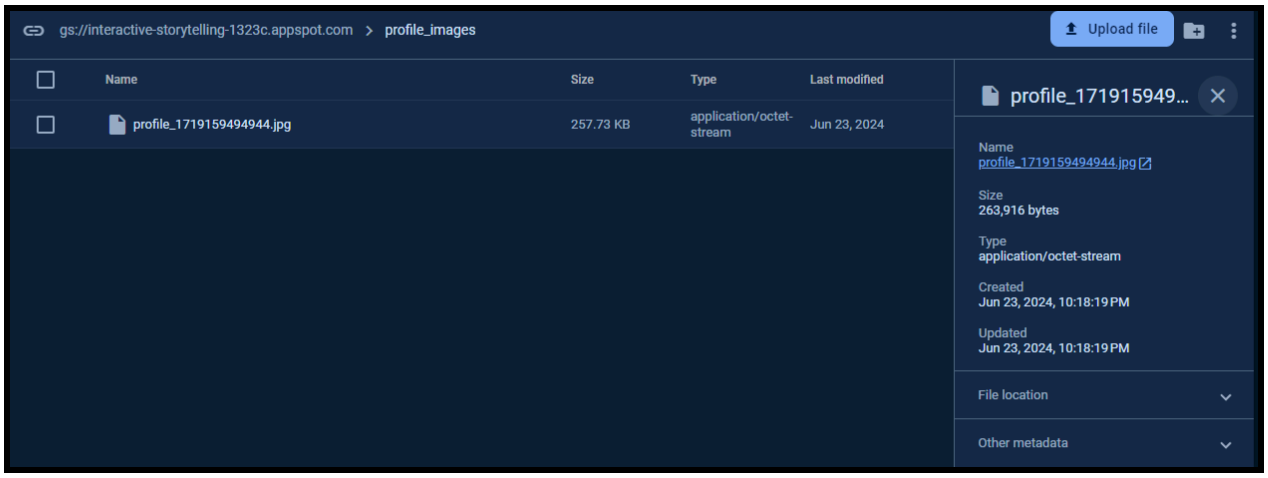
+ Start collection | + Add document | + Start collection

users | 6Uah8wLSjdlRA0F5fxEV | + Add field

```
NeFp9bDed9VXKtIzIz9d
RfRKuaUQEAnE0aR0y364
d1GKpafKbhgGhzDvNwUG
hfPJWzYRMs4dA4o7mvky
owUQ1h9oB6aPFEW2NXFD
```

```
birthday: "2024-06-23"
contact: "1"
country: "Country A"
email: "a@gamil.com"
gender: "Male"
name: "a"
photoURL: ""
```

- Database: In the database, Here all the user data is saved in the users by default unique key generated by firebase.



gs://interactive-storytelling-1323c.appspot.com > profile_images

Upload file

Name	Size	Type	Last modified
profile_1719159494944.jpg	257.73 KB	application/octet-stream	Jun 23, 2024

profile_171915949...

Name
profile_1719159494944.jpg

Size
263,916 bytes

Type
application/octet-stream

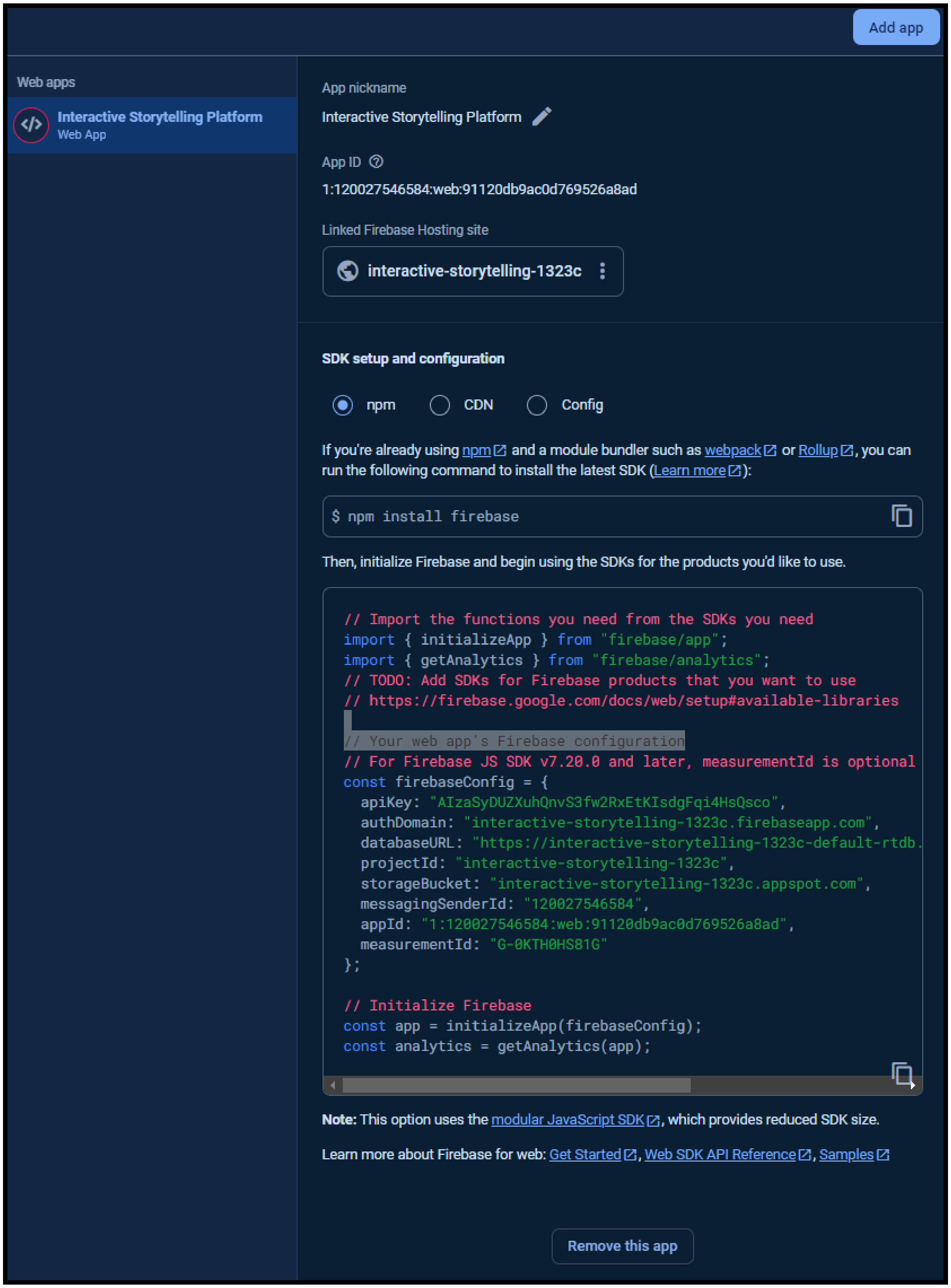
Created
Jun 23, 2024, 10:18:19 PM

Updated
Jun 23, 2024, 10:18:19 PM

File location

Other metadata

- Storage: Here, all the pictures of users will be saved here .



App nickname

Interactive Storytelling Platform

App ID

1:120027546584:web:91120db9ac0d769526a8ad

Linked Firebase Hosting site

interactive-storytelling-1323c

SDK setup and configuration

[✓ npm] CDN Config

If you're already using npm and a module bundler such as webpack or Rollup, you can run the following command to install the latest SDK (Learn more):

```
$ npm install firebase
```

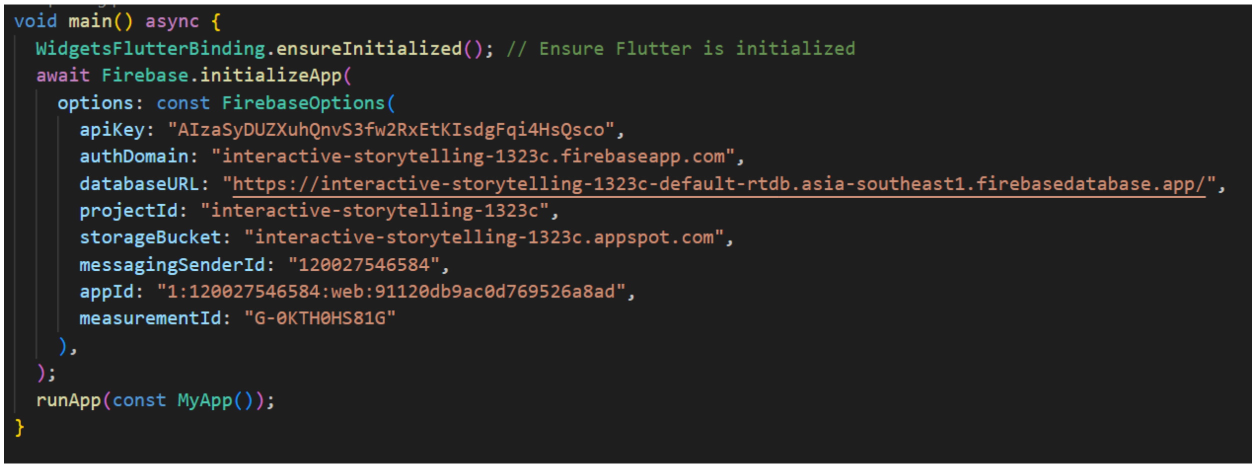
Then, initialize Firebase and begin using the SDKs for the products you'd like to use.

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
import { getAnalytics } from "firebase/analytics";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "AIzaSyDUZXuhQnvS3fw2RxEtKIsdgFqi4HsQsco",
  authDomain: "interactive-storytelling-1323c.firebaseapp.com",
  databaseURL: "https://interactive-storytelling-1323c-default-rtdb.
  projectId: "interactive-storytelling-1323c",
  storageBucket: "interactive-storytelling-1323c.appspot.com",
  messagingSenderId: "120027546584",
  appId: "1:120027546584:web:91120db9ac0d769526a8ad",
  measurementId: "G-0KTH0HS81G"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
const analytics = getAnalytics(app);
```

Note: This option uses the modular JavaScript SDK, which provides reduced SDK size.

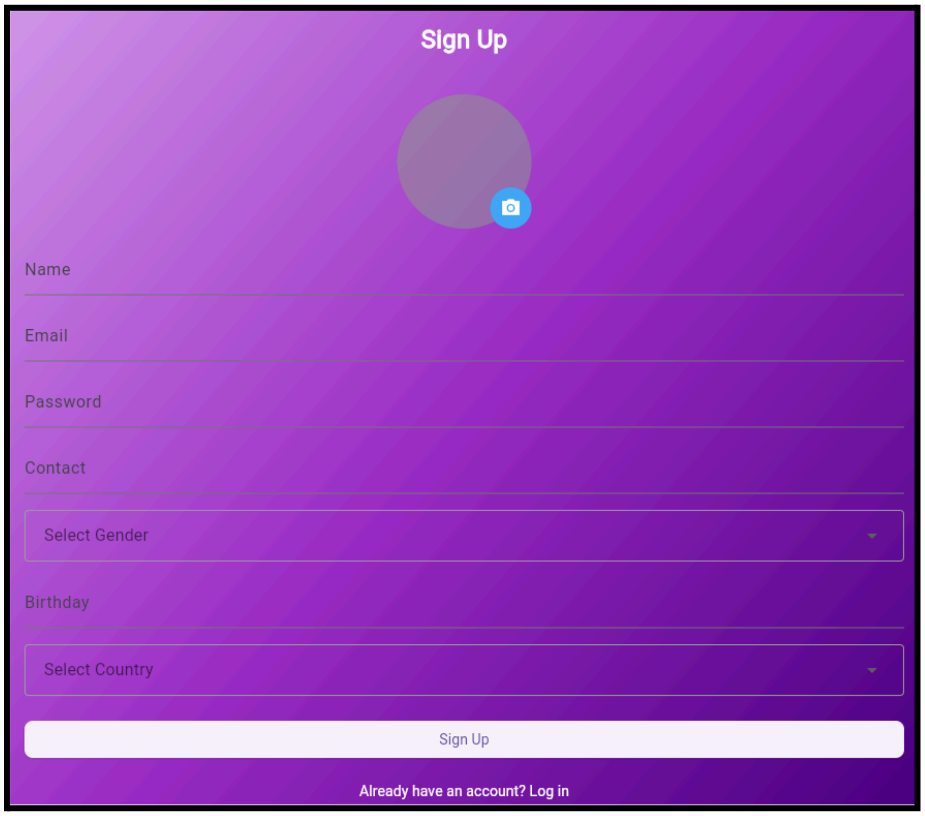
Learn more about Firebase for web: Get Started, Web SDK API Reference, Samples

- Here, we add our project to firebase and firebase gives us all the details of what we add to the main.dart file.

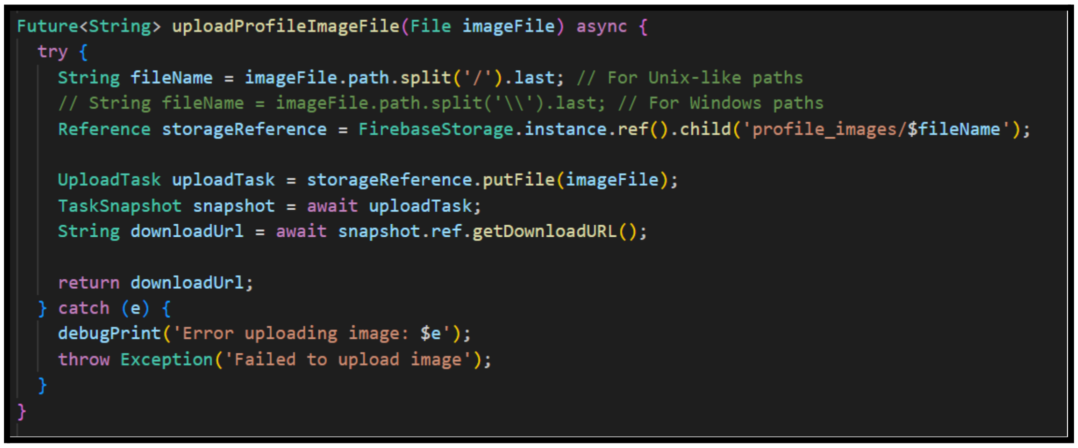


```
void main() async {
  WidgetsFlutterBinding.ensureInitialized(); // Ensure Flutter is initialized
  await Firebase.initializeApp(
    options: const FirebaseOptions(
      apiKey: "AIzaSyDUZXuhQnvS3fw2RxEtKIsdgFqi4HsQsco",
      authDomain: "interactive-storytelling-1323c.firebaseapp.com",
      databaseURL: "https://interactive-storytelling-1323c-default-rtdb.asia-southeast1.firebasedatabase.app/",
      projectId: "interactive-storytelling-1323c",
      storageBucket: "interactive-storytelling-1323c.appspot.com",
      messagingSenderId: "120027546584",
      appId: "1:120027546584:web:91120db9ac0d769526a8ad",
      measurementId: "G-0KTH0HS81G"
    ),
  );
  runApp(const MyApp());
}
```

Signup Page:

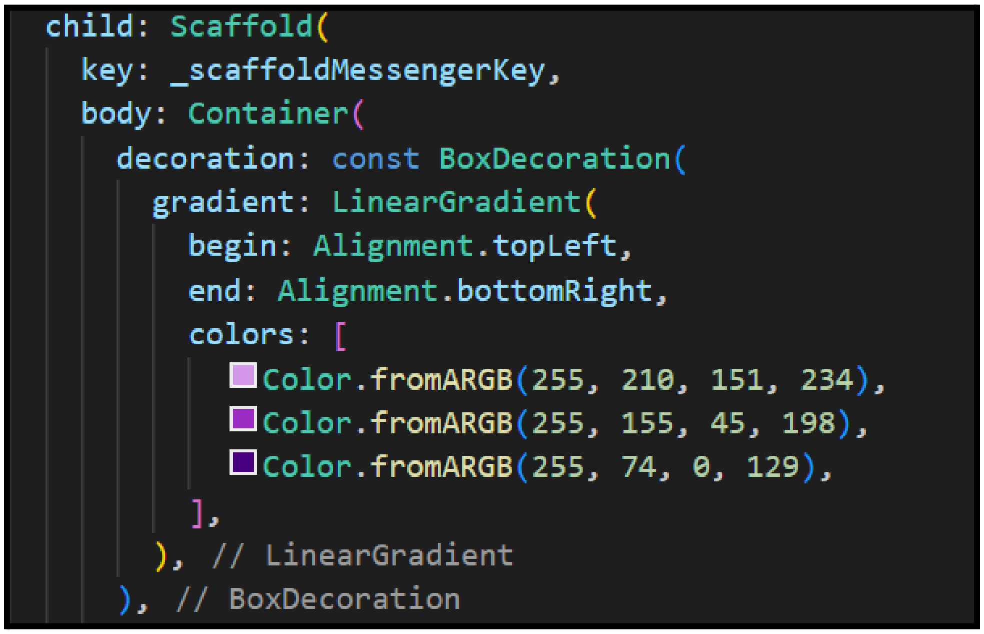


This is our sign up page. Here we are asking Profile picture, Name, Email, Password, Contact, Gender, Birthday, Country from the user.



```
Future<String> uploadProfileImageFile(File imageFile) async {
  try {
    String fileName = imageFile.path.split('/').last; // For Unix-like paths
    // String fileName = imageFile.path.split('\\').last; // For Windows paths
    Reference storageReference = FirebaseStorage.instance.ref().child('profile_images/$fileName');

    UploadTask uploadTask = storageReference.putFile(imageFile);
    TaskSnapshot snapshot = await uploadTask;
    String downloadUrl = await snapshot.ref.getDownloadURL();

    return downloadUrl;
  } catch (e) {
    debugPrint('Error uploading image: $e');
    throw Exception('Failed to upload image');
  }
}
```

Here, we are uploading images directly to firebase storage, then getting a download link and saving the link with the user database so that picture doesn't mix with other users.



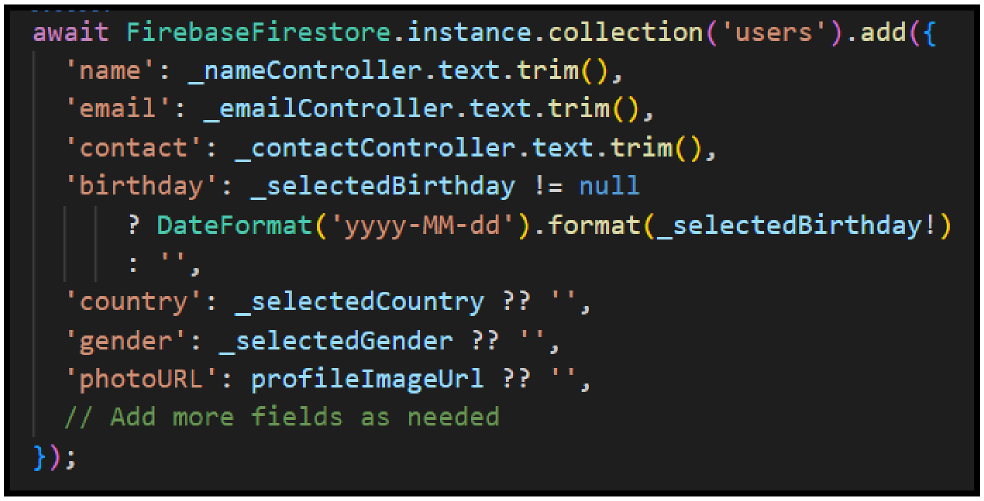
```
child: Scaffold(
  key: _scaffoldMessengerKey,
  body: Container(
    decoration: const BoxDecoration(
      gradient: LinearGradient(
        begin: Alignment.topLeft,
        end: Alignment.bottomRight,
        colors: [
          Color.fromARGB(255, 210, 151, 234),
          Color.fromARGB(255, 155, 45, 198),
          Color.fromARGB(255, 74, 0, 129),
        ],
      ), // LinearGradient
    ), // BoxDecoration
```

We used LinearGradient to design the frontend.

<u>Future development:</u>

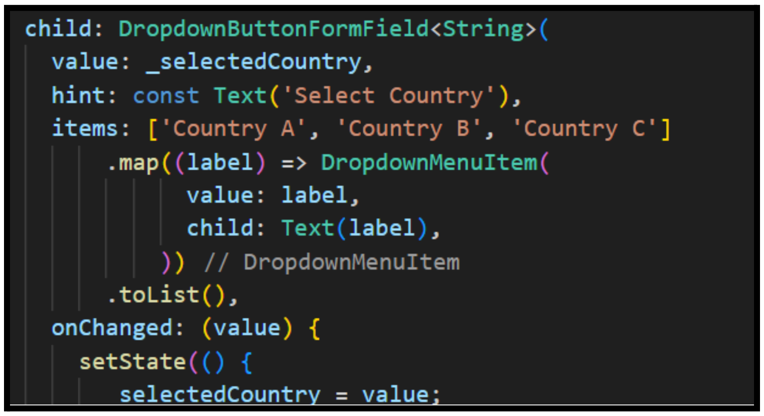
- In future, we will work on the frontend more.

- Passwords will be saved in hashing.That is why we are not saving the password yet.



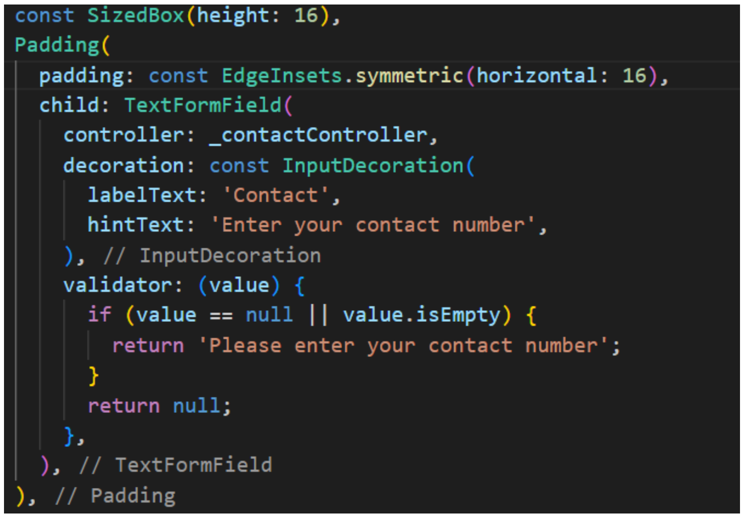
```
await FirebaseFirestore.instance.collection('users').add({
  'name': _nameController.text.trim(),
  'email': _emailController.text.trim(),
  'contact': _contactController.text.trim(),
  'birthday': _selectedBirthday != null
      ? DateFormat('yyyy-MM-dd').format(_selectedBirthday!)
      : '',
  'country': _selectedCountry ?? '',
  'gender': _selectedGender ?? '',
  'photoURL': profileImageUrl ?? '',
  // Add more fields as needed
});
```

- Here, there is no country name, only dummy names. We will update it in future.



```
child: DropdownButtonFormField<String>(
  value: _selectedCountry,
  hint: const Text('Select Country'),
  items: ['Country A', 'Country B', 'Country C']
      .map((label) => DropdownMenuItem(
            value: label,
            child: Text(label),
          )) // DropdownMenuItem
      .toList(),
  onChanged: (value) {
    setState(() {
      selectedCountry = value;
```

- There is no rule in contact that it must be 11 digit or something and no code to verify the contact.



```
const SizedBox(height: 16),
Padding(
  padding: const EdgeInsets.symmetric(horizontal: 16),
  child: TextFormField(
    controller: _contactController,
    decoration: const InputDecoration(
      labelText: 'Contact',
      hintText: 'Enter your contact number',
    ), // InputDecoration
    validator: (value) {
      if (value == null || value.isEmpty) {
        return 'Please enter your contact number';
      }
      return null;
    },
  ), // TextFormField
), // Padding
```