Superformula QR Code Scanner

Overview

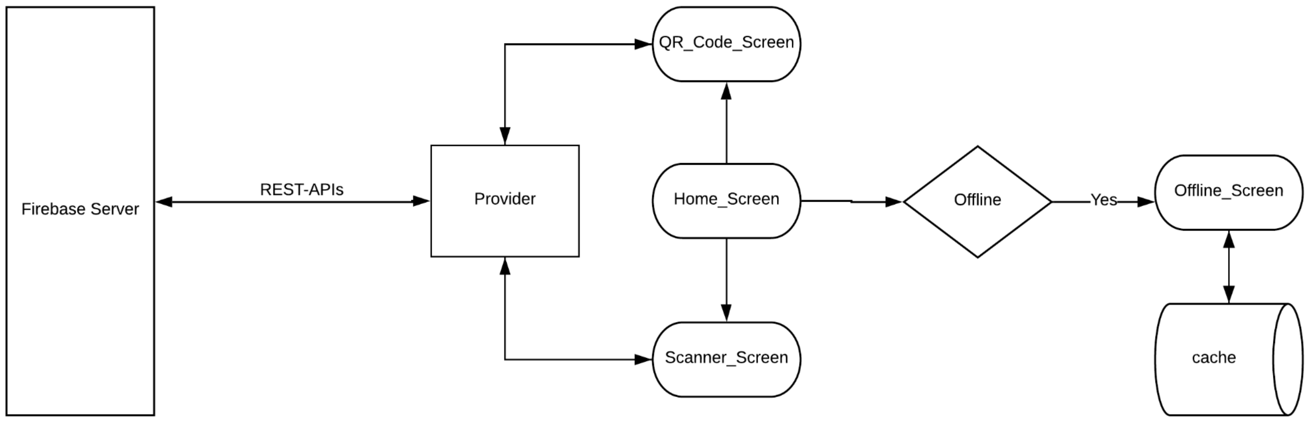
Mobile application developed in Flutter that has the functionality to display a QR code based on a seed value obtained through a REST-API and also the ability to scan a QR Code and verify whether it contains a valid seed value.

Technologies Used

1. **Flutter**: Used for developing the cross-platform mobile QR code application.
2. **NodeJS**: Server-side code that contains seed generation and seed verification modules.
3. **Firebase**: Used for hosting the server-side code.

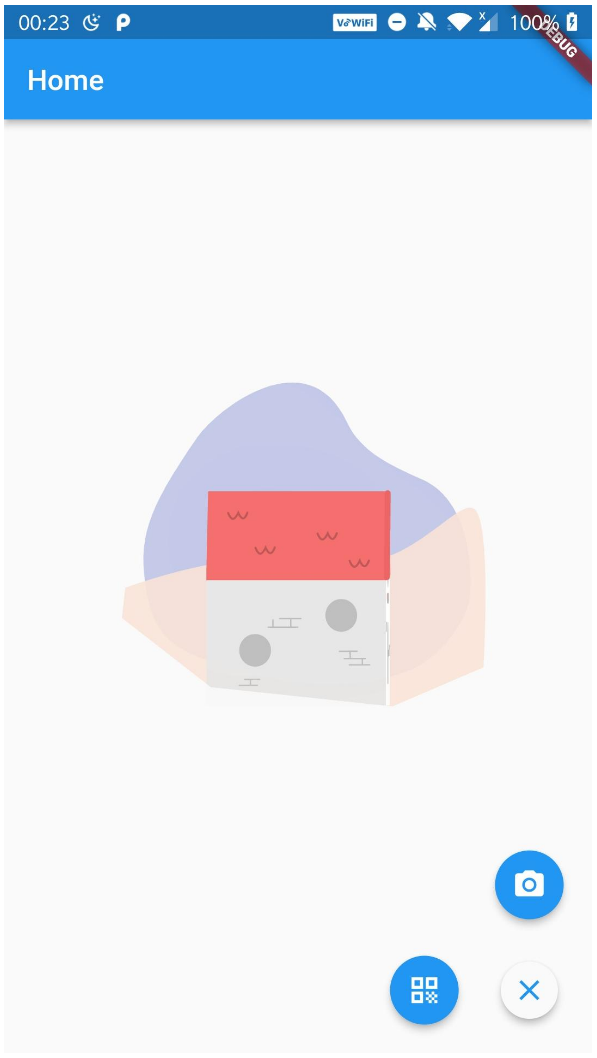
Project Structure

The app is designed based on an MVC architecture with each component as a separate entity but interconnected through a central controller. In this project, this is achieved using the Provider-Consumer feature provided by Flutter. A central multi-provider called Superformula provider is created that handles all the network functions that provide and notifies the values obtained from the REST-APIs to the listening consumers. Every time a seed is retrieved it is stored offline in the local device cache. Whenever the client mobile device goes offline this cached seed value is displayed in the QR Code of the offline screen.

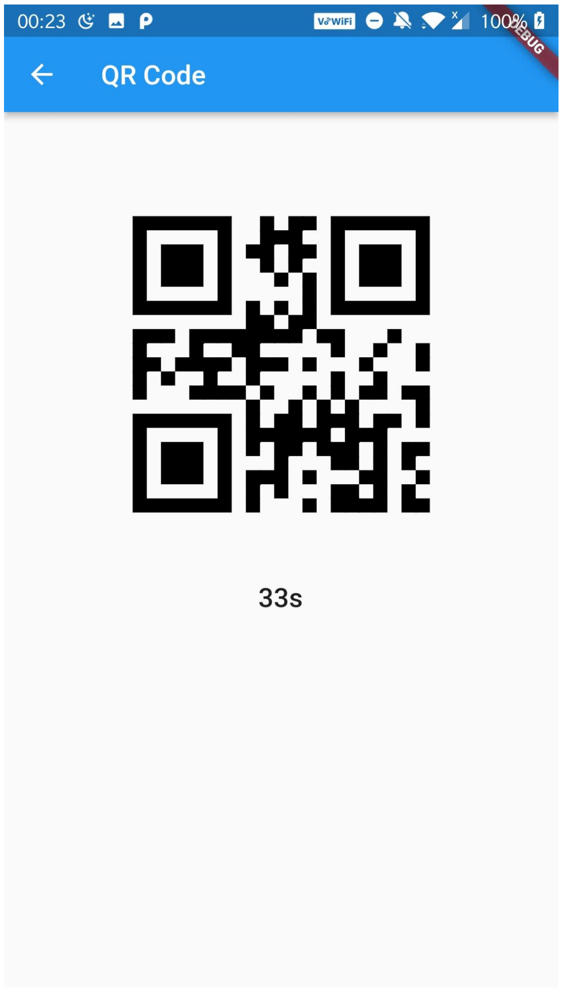


App Screens

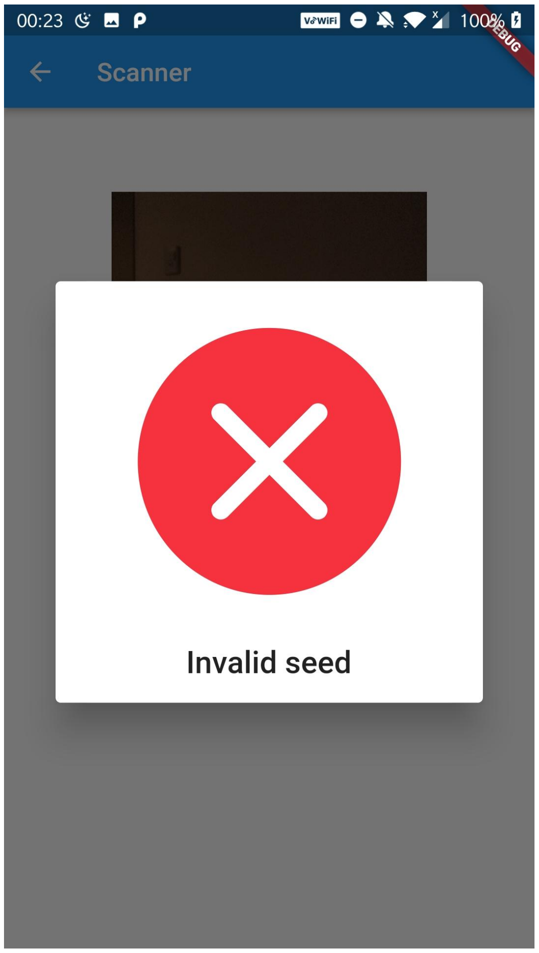
1. **Home**: Home page that provides navigation to the functional screens of the app.



2. **QR Code Screen**: Screen that displays a QR Code based upon the seed fetched from the server.



3. **Scanner Screen**: Screen that contains a camera view that scans a QR Code and extracts the data of the QR Code and verifies it against the server seed.



Seed Generation & Verification

The seed generation code is written on the server-side which generates a unique integer depending upon the current time. The seed generation function accepts the current UTC time's hour and minute combined as a string as the input and returns a unique integer based on the input. The seed is automatically refreshed every 60 seconds on both the server and mobile application. In the mobile application, a timer is implemented based on the current UTC time and pings the seed function every 60 seconds.

The seed verification is done by passing the scanned QR code data to the API and if it matches with the seed generated in the server, then the seed is determined as valid.

Libraries Used

1. **flutter_screenutil**: A flutter plugin for adapting screen and font size. Letting the UI display a reasonable layout on different screen sizes.
2. **top_snackbar_flutter**: A flutter plugin used for displaying user feedback through a top snackbar.
3. **flutter_easyloading**: A flutter plugin used to display a loading overlay UI
4. **qr_flutter**: Flutter plugin used for displaying QR Codes.
5. **provider**: A wrapper around InheritedWidget to make them easier to use and more reusable.
6. **dio**: A powerful http client made for Flutter with advanced networking features.
7. **qr_code_scanner**: Package that handles QR code scanning and data extraction.
8. **shared_preferences**: Used for offline persistent data storage, Used for storing seed value to generate offline QR code.
9. **simple_connection_checker**: Flutter package that checks a device's online status.
10. **test**: Package used to implement Unit Tests,
11. **dependency_validator**: Ensure there are no unused packages in the application.
12. **flutter_launcher_icons**: Command line tool to automatically generate launcher icons for both iOS and Android.

Unit Tests

The tests can be run by navigating to the flutter folder and executing the following command "flutter test test/seed_test.dart". Here are the unit tests defined in the seed_test.dart file

1. Seed Verification Fail: All test conditions are passed, the input seed is any random number less than 100 which differs from the server-side seed.
2. Seed Verification Success: This test is implemented by getting the server-side seed first and immediately verifying it thus returning a success condition.

Executing the code

The project structure is divided into 2 directories.

1. Nodejs: Navigate to functions subdirectory and execute the command "npm run serve" to execute the server-side code on the localhost. The url can be copied to Postman and run the API calls /api/seed to receive a seed value and /api/verify-seed to verify if a seed is valid or not (Pass the seed as a JSON parameter in postman in the form {"seed":<value>}).

2. Flutter: Open this folder in Android Studio or a supported IDE of your choice. Make sure the Android, iOS and Flutter SDKs are installed. The IDE will be able to execute the project directly.