

ACTION PLAN

BY

TEAM 33

HASURA PRODUCT DEVELOPMENT FELLOWSHIP

Members:

| | | |
|--------|--|----------------|
| T33PF1 | Shivam shivamd20@gmail.com | ReactJS |
| T33PF1 | Swetha Gunta svgunta@gmail.com | React-Native |
| T33PF1 | Joel Kingsley joelkingsley.r@gmail.com | Python-Flask |
| T33PF1 | Uddhav Wani uddhavwani9@gmail.com | Python-Flask |
| | Abhinav Anand | |
| T33NE1 | abhinavanand.rockstar@gmail.com | NodeJS-Express |
| T33NE1 | Swathi Ainala swathi.ainala@gmail.com | ReactJS |

Objective:

To create an application that allows users to add data to a table, triggers a zap (from Webhook to Google Sheets) that adds the name of the table to Google Sheets as a new row. The user will be notified that the data is saved, and the zap is triggered.

Architecture:

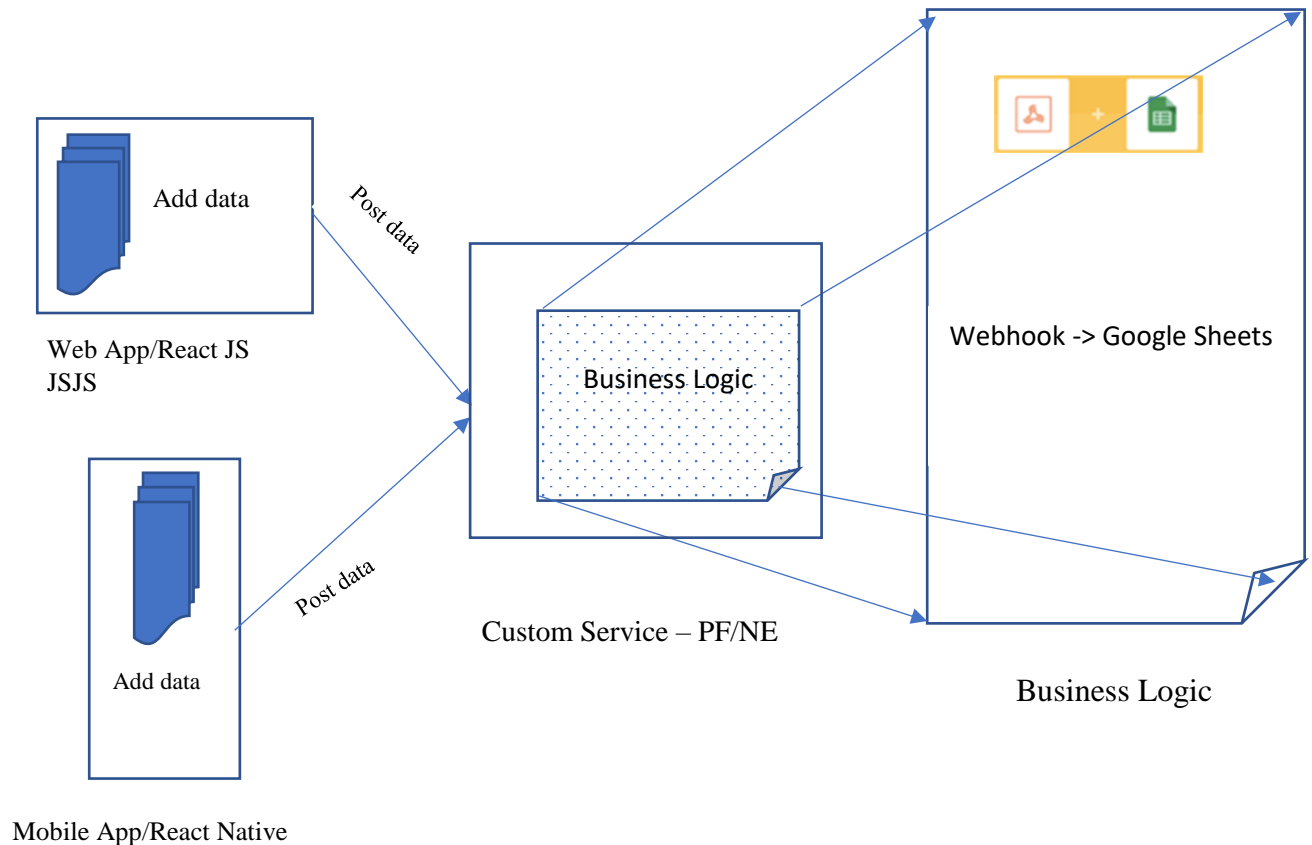


Figure 1: Architecture of the App

The project consists of a Front End and a Back End. The user creates a new table or selects an existing table and populates the table with data. On the click of the button named “Save Data” button the front end posts a request, that carries the data, via an URL, a custom microservice. This custom microservice then triggers a zap, which is activated from WebHook app to Google Sheets app. The Table name, a JSON String is now carried via a POST request to the WebHook app. This WebHook app copies the table name to Google Sheets as a new row. Once a trigger is made by the backend the user is notified that the data is saved, and the zap triggered.

Basic Flow Diagram:

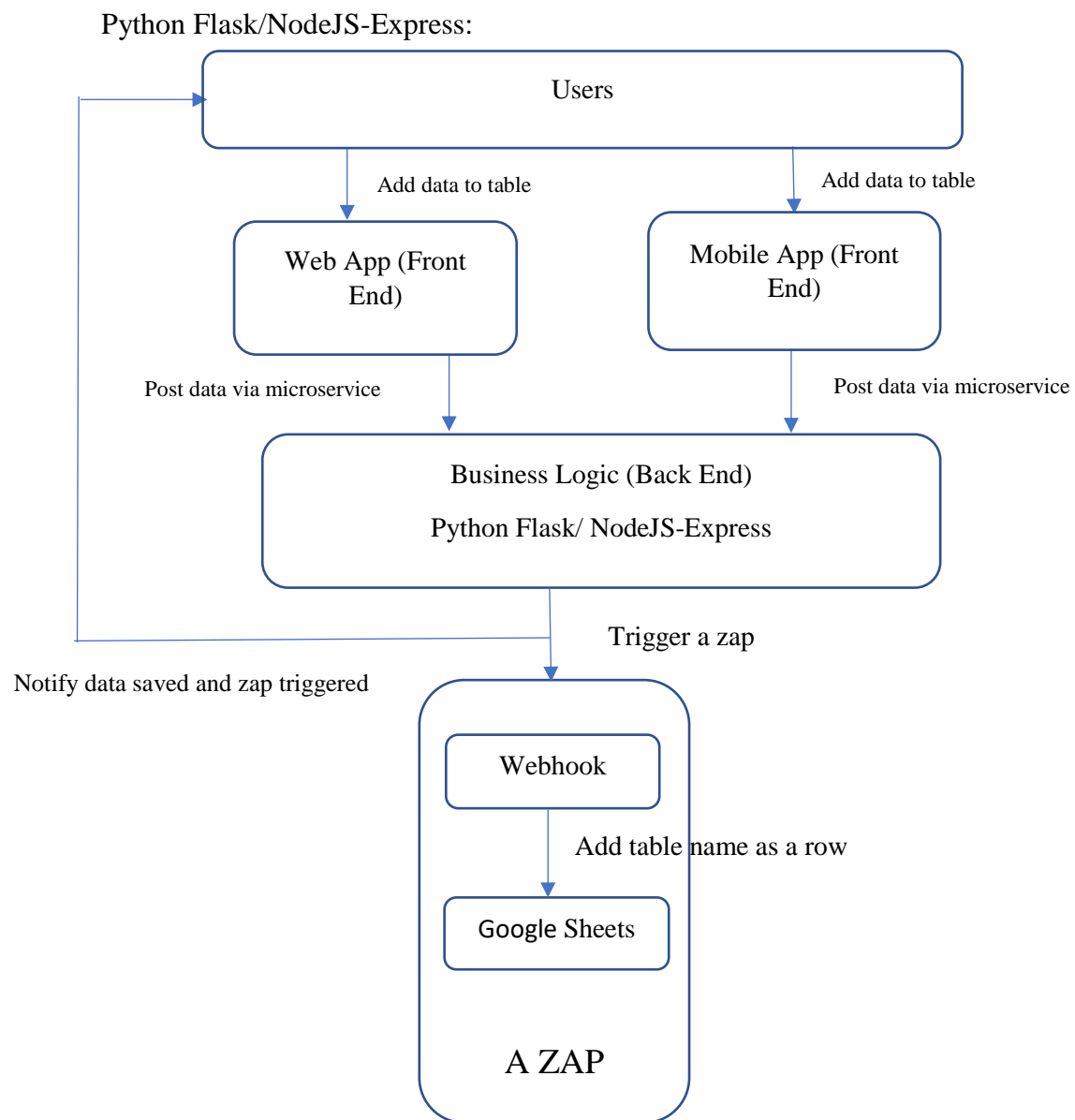


Figure 2: Basic flow diagram with Python Flask/NodeJS-Express as Back End

Each team consists of one Back-End developer and two Front-End developers, one for Web Application and the other for Mobile Application.

Back-End:

1. Python-Flask:

The high-level idea in the backend is to host end points for both the web as well as the mobile app, through which registered users can create and manage tables in their accounts. At the same time each user will get a shared link to a Google Sheet in their username, which automatically gets updated with the latest list of tables in their account.

Initially, we are going to start off by implementing the basic CRUD operations for the tables. When the user creates a new table, and clicks on the create button, a JSON POST request containing the table's details is made to the server, which would store the details in a database.

At the same time, a trigger in the form of a JSON POST request is made by the server to the Zapier Webhook, which takes the request and makes changes in the Google sheet allotted for the user from whom the request is made.

Subtasks:

1. Create endpoints for CRUD operations.
2. Create database.
3. Trigger Zap when changes are made in a table.

Dependencies:

- PostgreSQL - PostgreSQL is a powerful, open source object-relational database system. Here, it will be storing the tables of all the users.
- Zapier - Zapier is an online tool that is used to connect two or more apps to automate workflows. We'll be connecting the Zapier webhook and Google Sheets to get a spreadsheet of table names for each user.
- Hasura Authentication (in later stages) - Hasura Auth APIs let you create, authenticate and manage user accounts on your Hasura project. It also lets you manage sessions and roles for users. It will be used to manage users for the app.

2. NodeJS-Express:

Front End:

1. React JS

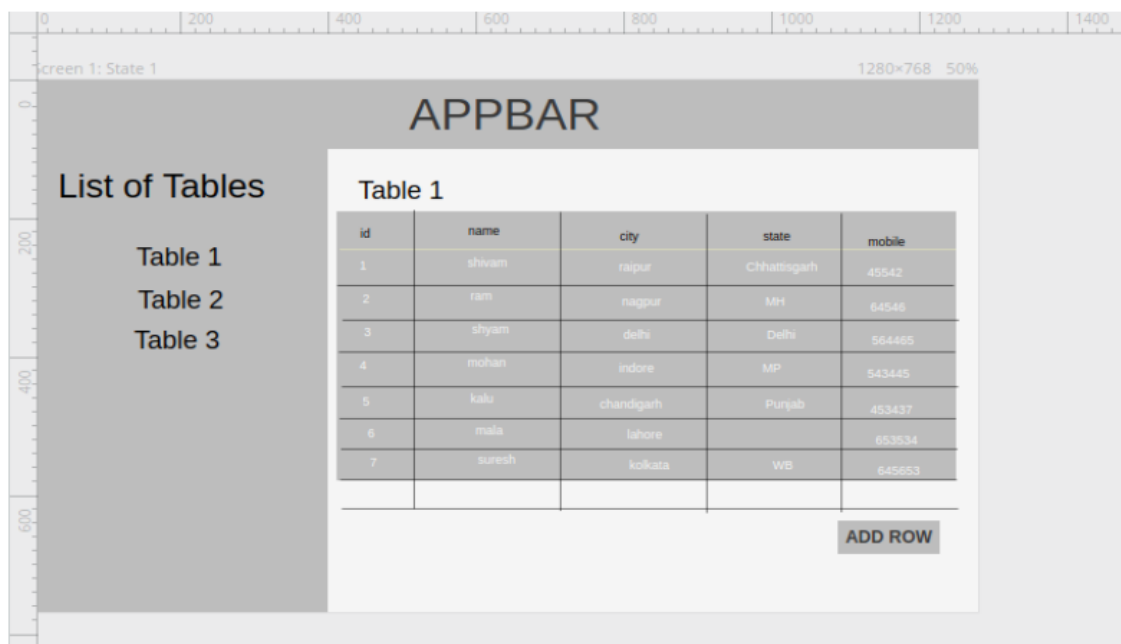


Figure 3: Wire-frame for ReactJS

2. React Native

Introduction:

React Native is a JavaScript framework that uses the same design as React to build the user interface for native mobile devices. In other words, it uses the same building blocks, declarative components, as that of iOS and Android Apps. It is used not to build a mobile web app but a mobile app that is indistinguishable from the app built using Objective-C or Java.

Setting up the environment:

- (1) Install NodeJS and npm – Download the windows installer from the Node.js website. Run the installer and follow the prompt messages. Once the installation is done test the versions using:
`$npm -v`
`$node -v` //Check the versions, required node>=6.0
`$npm install npm@4.6.1` //Downgrade to the required version
- (2) Install React Native – To install react native use the following command:
`$npm install -g react-native-cli`
- (3) Install Android Visual Studio – Download the installer from the website and follow the prompt messages. Set up an Android Marshmallow environment via Sdk tools. Also set up an Android Virtual Device.
- (4) Create the app – Use the following command to create the app.
`$react-native init <Project-Name>`
- (5) Install NativeBase, React Navigation and link peer dependencies – Change directory to the project folder, i.e., `$cd <Project-Name>` and install native base directory using the following command: `$npm install native-base --save`. Also install react-navigation using `$npm install --save react-navigation`. Link dependencies using the command: `$react-native link`
- (6) Running the app – Now run the packager using `$react-native start`. Keep the terminal window open and run the following command to run the app in the emulator `$react-native run-android`

Wireframe of the UI:

The project contains the following screens:

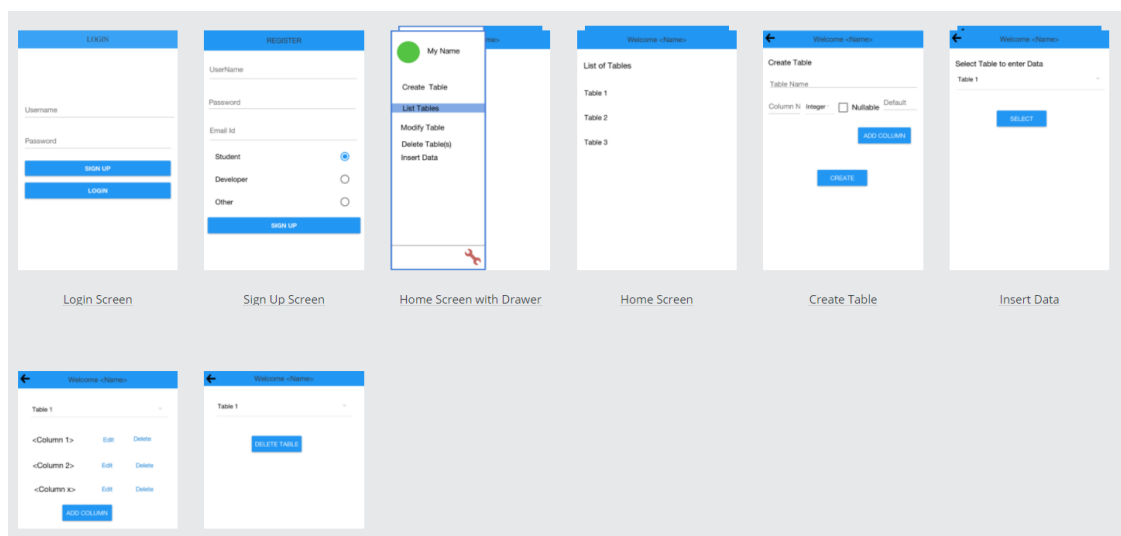


Figure 4: Wireframe for the Mobile App

The Screen Transitions are as follows:

Login Screen: The user logs in using the credentials. If the user isn't registered before he/she registers himself after clicking the "SignUp" button

Sign Up Screen: The user if isn't registered registers himself in this screen. As soon as the user registers himself he is taken back to the Login Screen

Home Screen with Drawer: As soon as the user logs in with the right credentials he/she is navigated to the Home screen which has different permissions. It displays a List of Tables that can be edited and modified. It also contains a drawer through which one can navigate to different screens.

Create Table: In this page the user enters the name a Table. The table name is accepted if the table doesn't exist already. The user is also asked to enter the column names and their types.

Insert Data: This page allows the user to select a table in which the user wants to insert data and then populate the data column-wise.

Modify Data: This page allows the user to modify the column names and add extra columns if necessary.