ACTION PLAN

BY

TEAM 33

HASURA PRODUCT DEVELOPMENT FELLOWSHIP

Members:

|  |  |  |
| --- | --- | --- |
| T33PF1 | uddhavwani9@gmail.com | Python-Flask |
| T33NE1 | abhinavanand.rockstar@gmail.com | NodeJS-Express |
| T33NE1 | swathi.ainala@gmail.com | ReactJS |
| T33PF1 | [shivamd20@gmail.com](mailto:shivamd20@gmail.com) | ReactJS |
| T33PF1 | [svgunta@gmail.com](mailto:svgunta@gmail.com) | React-Native |
| T33PF1 | joelkingsley.r@gmail.com | Python-Flask |

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:



Business Logic

Add data

Add data

Post data

Post data

Webhook -> Google Sheets

Custom Service – PF/NE

Business Logic

Web App/React JS JSJS

Mobile App/React Native

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.

KEYWORDS:

***Zap:***

A zap is an automated workflow between two or more apps. It consists of at least two parts: a Trigger, one or more Actions. A zap has to be created, saved and activated. In this project, a zap connects the WebHook app and Google Sheets app. The WebHook app is the trigger which is followed by an action of coping the table name to Google Sheets.

***WebHook:***

A WebHook is an HTTP POST that posts data to a URL on the happening of an event in the Web Application or Mobile Application. In this project a WebHook is configured when the user clicks the “Save Data” button. The HTTP POST carries the Table Name as a JSON String.

***Google Sheets:***

A spread sheet that consist of many rows and columns facilitating the entry of information in an organised and categorical way.

***Trigger:***

A trigger is the event occurring in an app that starts the zap. In this project, once a zap is created and activated it monitors the Web Application or Mobile Application for any modification of a Table. A table can be modified in many ways, but the zap is not triggered until a table is populated with data.

***Action:***

An Action is the event that completes the zap. In this project, the Action is saving the table name to Google Sheet.

Basic Flow Diagram:

Python Flask:

Add table name as a row

A ZAP

Webhook

Google Sheets

Business Logic (Back End)

Python Flask

Post data via microservice se

Users

Web App (Front End)

Add data to table

Mobile App (Front End)

Add data to table

Post data via microservice

Trigger a zap

Notify data saved and zap triggered

Figure 2: Basic flow diagram with Python Flask as Back End

Basic Flow Diagram:

NodeJS- Express:

Add table name as a row

A ZAP

Webhook

Google Sheets

Business Logic (Back End)

NodeJS-Express

Post data via microservice se

Users

Web App (Front End)

Add data to table

Mobile App (Front End)

Add data to table

Post data via microservice

Trigger a zap

Notify data saved and zap triggered

Figure 3: Basic flow diagram with Python Flask 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.

Team PF (Python-Flask):

Back End: Python-Flask

< Description of tasks and sub tasks > < Sample Code Snippets >

Front End:

1. React JS

< Description of tasks and sub tasks > < Sample Code Snippets >

< Sample pictures/ wireframe >

1. 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 – Node.js is a cross-platform open source server framework using javascript on the server. A node.js file contains tasks that have to be executed when an event occurs. One such event is trying to access a port on the server. A node.js files must be initiated on the server before having any effect. They have a file extension of “.js”. npm stands for nodeJS package manager. It facilitates reuse of code or modules and allows download of many packages. To install npm and nodejs:

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](mailto:npm@4.6.1) //Downgrade to the required version

1. Install React Native – To install react native use the following command:

$npm install -g react-native-cli

1. 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.
2. Install a Text Editor – This project uses Microsoft Visual Studio editor. Download it from the website.
3. Create the app – Use the following command to create the app.

$react-native init <Project-Name>

1. 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
2. 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:

<TO DO>

“Swetha G”]

Team NE (NodeJS-Express):

Back End: NodeJS-Express

< Description of tasks and sub tasks > < Sample Code Snippets >

Front End:

1. React JS

< Description of tasks and sub tasks > < Sample Code Snippets >

< Sample pictures/ wireframe >

1. React Native

< Description of tasks and sub tasks > < Sample Code Snippets >

< Sample pictures/ wireframe >

< More Side headings such as “Integration”, “Milestones”, etc will be added later >

**P.S. : I request every one to edit their respective columns. Please enclose your description with [] brackets and your at the end so that later it is easier to edit your part without getting confused as to who wrote what.**

**For eg:**

**[ I belong to React Native PF group and I will be writing all my description like this. Hope everyone is comfortable with this “Swetha G”]**