Alliance Beverage Distributing

Kevin Holkeboer

Justin Houstin

Zack Poorman

Joseph Turnbull

Table of Contents

Overview

The primary focus of this project is to harness several years of data and analyze it in real-time based on user entered fields. This will aid ABD in data driven decision-making and enhance their customer experience. By utilizing React, Flask and R in a web app, anyone at ADB will be able to enter product information or product traits and receive a list of best-fitting customers.

Languages

We'll use React.JS for the frontend. It is our web development tool of choice. For the backend, we will be using Python/Flask. This is because python is both great for backend development as well as data analysis. R was originally used by ABD for mining and cleaning data. We will use R when interoperability is required.

Code Repository Organization

Due to the nature of this project, we are only utilizing the main branch since the majority of the changes shouldn't break the app. If we encounter any issues, we can easily create development and production branches to solve them. The main reason behind this is that it'll have a simple UI and a simple database which greatly lowers the risk for large scale issues. More branches may be added later if needed.

Development

Our application will be developed in our own local environments. Our initial prototype files will be used as the starting point for the teams development. We've already been provided with ADB's data from their initial data mining as well as the steps to reproduce their results.

ZenHub

We will be utilizing a standard board in ZenHub to follow the SCRUM methodology.

Icebox

The Icebox is for future features or ideas that aren't part of the current sprint.

Backlog

The Backlog will contain all issues that will be worked on in the current sprint.

In Progress

In Progress will contain all issues that are currently working on.

Review/QA

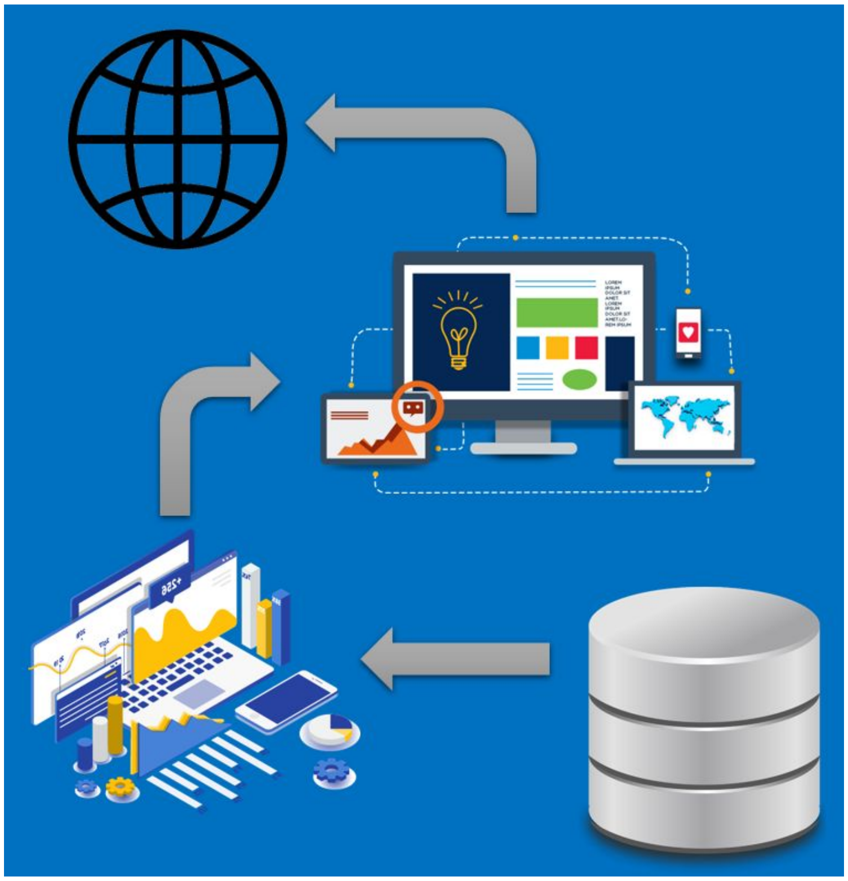
Review/QA will contain all issues that have been finished and are waiting for/are in review.

Done
Done will contain all issues that have been completed, reviewed, and approved, in the current sprint.

Closed
All issues that have been completed and aren't part of the current sprint will be in Closed.

System Organization

Users will have to login before they can use the app. Users will be able to enter product names or information into a field and a list of the best fitting customers will be rendered on-screen. This process will work for both new and existing products. The existing products will be enterable by both options, whereas new products can only be entered by their traits. Prior to being rendered, the list of best-fitting customers will be generated by the real-time data analysis based upon the entered information. The customer information whether it's stored in a database or an Excel file, will be referenced by the data analysis to ensure the product is suggested for the customers where it'll have the highest sales and fit the customer. Depending on whether the app has a database or an excel file will decide whether it's client-server or pure client.



Rationale
We decided to develop a web based application for a couple of reasons. The first is that deployment and maintenance will both be simpler by not requiring local installs that need to be updated. The second is that this application wouldn't benefit from being a desktop application. There is little work required from a user (selecting a customer or selecting a product), so being able to work offline wouldn't be meaningful. There is also not a large amount of processing needed, so the likelihood of overloading a server is small. Security will always be a concern with any web app but because our application will only be sending sales recommendations not raw customer data, the damage caused by a security failure would be moderate.

Libraries, Frameworks, & APIs

We are looking to use Amazon Web Services or Netlify to help simplify the deployment process. This will also provide reliable access and enhanced security by giving us access to tools that have been tested through their large scale use.

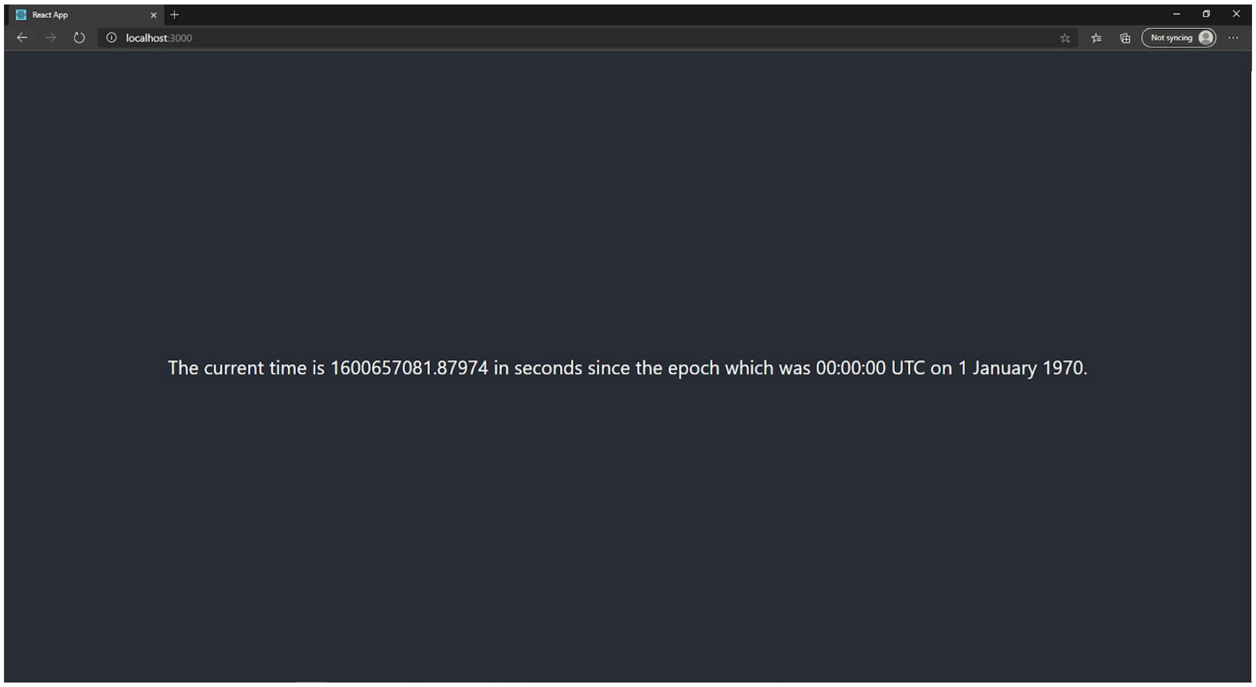
We are using Flask for our backend as it is a very simple and easy to use backend and is Python-based. This is convenient as Python is a very popular and powerful language for data mining/analysis. Also, everyone on the team is familiar with Python and it is interoperable with R (which the client is currently using) with Rpy2.

We will be using a React front end for our application as it integrates nicely with Flask and is a very popular and powerful Javascript framework. We also have team experience with React and we are looking for a responsive UI with forms and React excels in this area.

Prototype

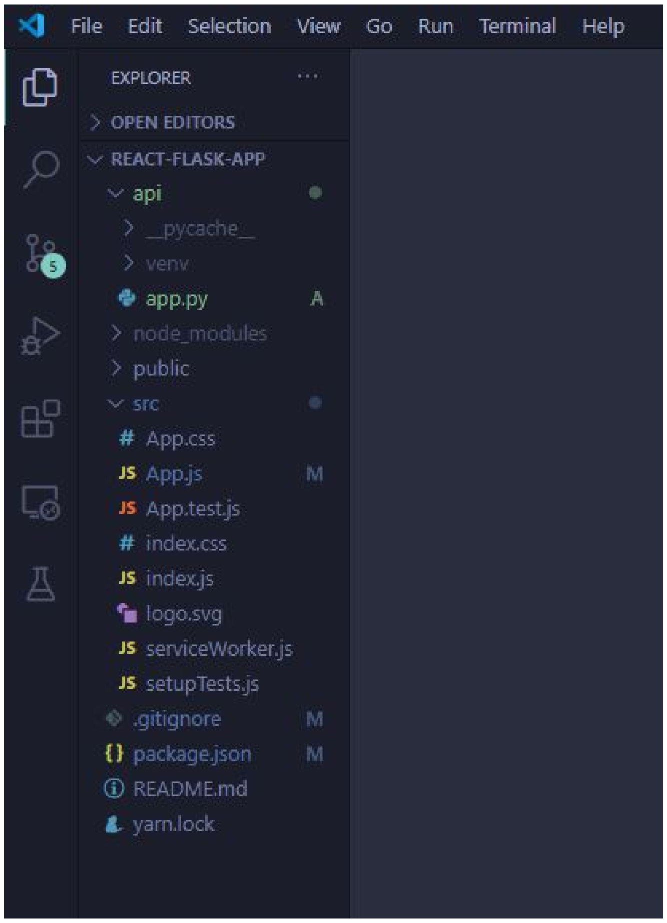
We built a prototype using a flask backend with a React front end and wired them up together. We also stored both the front and back end on one git repo for easy deployment testing.
The prototype was a simple idea to have the API return the time since the epoch (which was in 1970) in seconds. This number is continually increasing so our prototype ought to make new calls to the backend every time the front end is refreshed.The necessary requirements for our prototype are: Yarn, Python, Node, Flask. Optional requirements were: VScode

The prototype running on Edge:



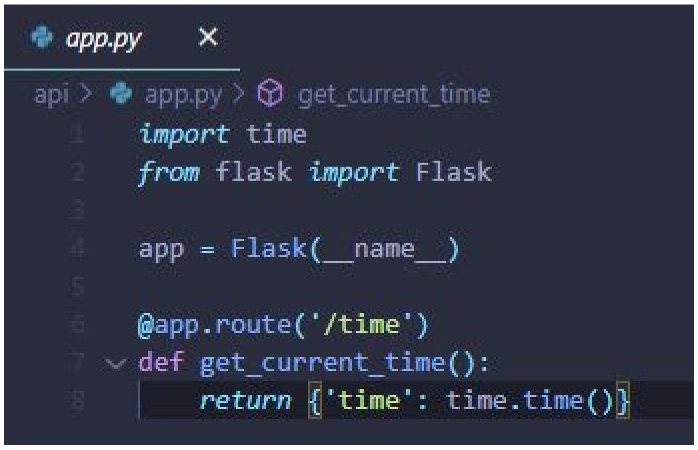
The current time is 1600657081.87974 in seconds since the epoch which was 00:00:00 UTC on 1 January 1970.

The prototype's file structure:



```
File  Edit  Selection  View  Go  Run  Terminal  Help
EXPLORER
> OPEN EDITORS
v REACT-FLASK-APP
  v api
    > __pycache__
    > venv
    app.py              A
  > node_modules
  > public
  v src
    App.css
    App.js              M
    App.test.js
    index.css
    index.js
    logo.svg
    serviceWorker.js
    setupTests.js
  .gitignore            M
  package.json          M
  README.md
  yarn.lock
```

The Flask API file is:



app.py

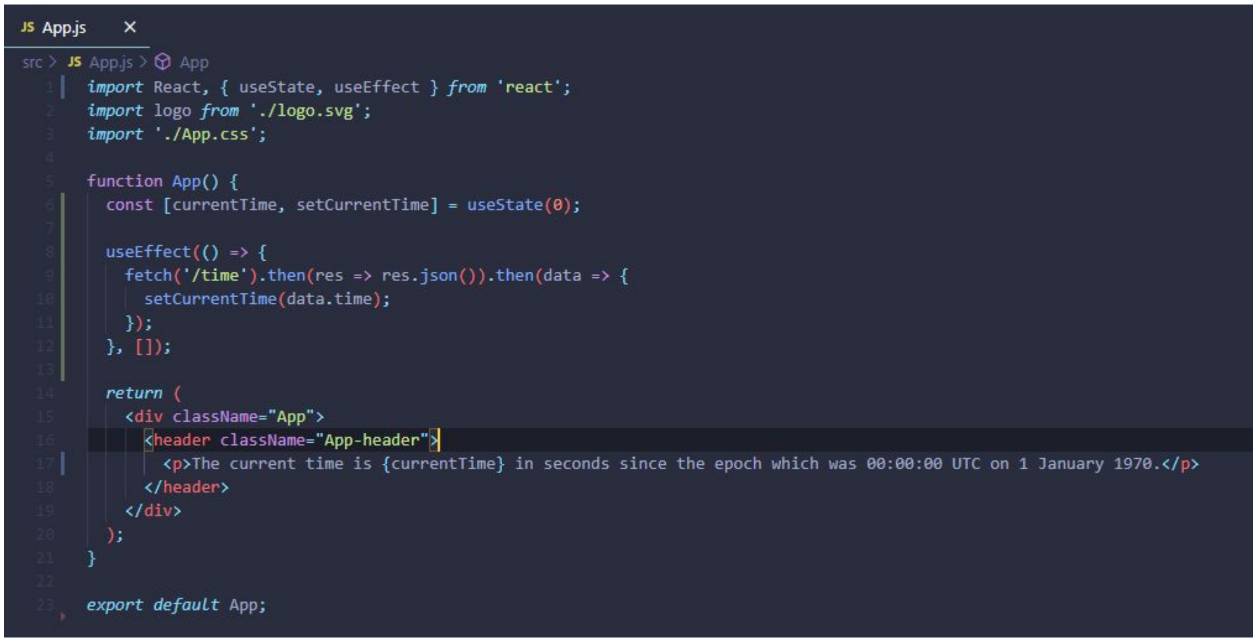
api > app.py > get_current_time

```python
import time
from flask import Flask

app = Flask(__name__)

@app.route('/time')
def get_current_time():
    return {'time': time.time()}
```

The React App.JS file is:



```
import React, { useState, useEffect } from 'react';
import logo from './logo.svg';
import './App.css';

function App() {
  const [currentTime, setCurrentTime] = useState(0);

  useEffect(() => {
    fetch('/time').then(res => res.json()).then(data => {
      setCurrentTime(data.time);
    });
  }, []);

  return (
    <div className="App">
      <header className="App-header">
        <p>The current time is {currentTime} in seconds since the epoch which was 00:00:00 UTC on 1 January 1970.</p>
      </header>
    </div>
  );
}

export default App;
```

Testing

By using SCRUM as our workflow methodology, we won't have time for dedicated testing. Instead, we are going to make sure our code is reliable by testing our code prior to pushing it to GitHub. Before each pull request is merged, someone on the team other than the author will quickly test the code and approve it unless there are issues. For our final sprint, we're going to focus on integration testing and an overall thorough test. At the end of each sprint, we'll demonstrate it to ADB for feedback. Prior to demonstrating to ADB, we'll quickly test what we're planning to demonstrate.

Work Environment

This project can generally be broken down into two large parts: The front end and the back end (this may not entirely encompass what is generally assumed for these roles). The front end will consist of creating a web app, the user interface, handling login credentials and security, and lastly connecting the front end to the back end. The back end will consist of data analysis, predictive analytics, database management, and connecting the database to the rest of the back end. The front end will be primarily done in VS Code, while the backend will be primarily done in PyCharm.

Zenhub will be used to delineate tasks and to track the progress of the project. Github will be used to host the project, where there will only be one primary branch (the master branch). Whenever somebody gets a new task, they will make a branch and work on that. Once the work is done they will merge the branches together. More intermediate branches may be added as needed. Basic testing will be done as sections of code are finished, and will be more thoroughly examined once the project is near completion.

User Interface

Our user interface is going to be very minimalistic. It will have the ability to select an existing product and get a list of customers who will be most likely to buy it. The other option is to enter traits about a new or existing product and a list of best-fitting customers will be rendered.

Database and Deployment

In terms of databases we are still waiting on the client for complete understanding for how they want us to store/access the product information. We currently have three options:

Firstly, that they have a SQL Server database that will house the product information that we will pull in from our API. We will also use their customer Excel file as data and upload it into the program.

Secondly, we create our own database which will hold both the customer information that we have in the Excel file and the relevant information from their product database that we will pull from.

Thirdly, we have no database at all and have options for them the Excel files of both their customer and product information.

In terms of deployment we are looking at both cloud-based paid options such as AWS, Azure or Netlify to hold this website or perhaps deploying on site if they have that capability.

Division of Labor

Labor will be divided up rather loosely. The purpose of this is for everyone to gain exposure to all parts of the process and to not tie anyone down to a strict role and allow flexibility. That being said, general roles will be in place. Joseph and Kevin will take charge of the front end of the system, while Justin and Zack will be in charge of the back end (as described in work environments).

These roles were chosen based off of a combination of skills, experience, and interests. Justin and Zack are the only two with data science experience which is why they were chosen for the back end. Joseph and Kevin both have a keen interest in web development, which made them the easy choice for front end.

These two larger sections (front end and back end) will be separated into smaller subsections (listed in the milestones section). Each individual will pick from the subsections available in each sprint, regardless of if they were assigned to that larger role (hence the “loose” division). They, however, will only do this if they are ahead of schedule for their own section. When an individual takes a task, they will be solely responsible for that task, however the other members will be obliged to help when requested.

Deliverables and Milestones

1. Set up the environment
   Design web page
   Identify data requirements
   Explore data and characteristics
   Connect to the database

2. Build home page structure
   Add basic features
   Develop Methodologies
   Determine important variables

3. Add advanced features
   Build and assess first model
   Build and assess second model

4. Create login page
   Build and assess third model
   Build and assess fourth model
   Choose model

5. Implement model
   Connect front end to back end
   In depth testing and debugging
   Polishing
   Finalize Documentation

6. Presentation
   Project submission