My steps taken (Frontend)

1. This is a React functional component that renders a form for a business loan application. The form consists of input fields for the company name, year established, loan amount, and accounting provider. There is also a button to request the balance sheet, which will fill in the profit or loss and asset value fields once the request is completed. Finally, there is a button to submit the application and display the final loan outcome

2. The component defines three functions: handleSubmit, handleOptionChange, RequestBalanceSheet, and GetFinalOutcome

3. handleSubmit is called when the user submits the loan application form. It prevents the default form submission behavior and logs a message to the console

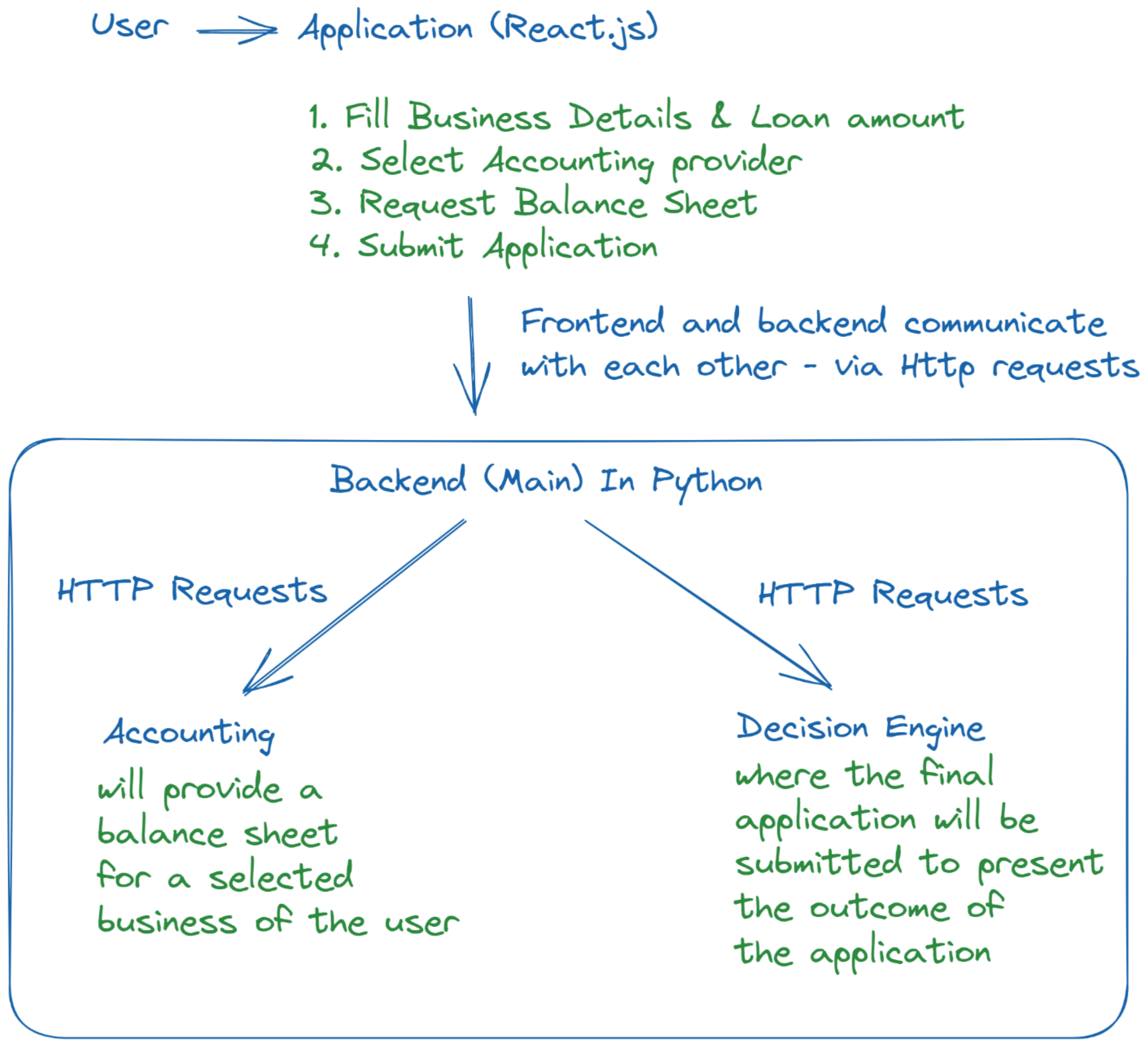
4. handleOptionChange is called when the user selects an accounting provider from the dropdown menu. It sets the accountingProvider state variable to the selected value

5. RequestBalanceSheet is called when the user clicks the "Request Balance Sheet" button. It checks that the name, yearEstablished, and loanAmount state variables are not empty. If they are empty, they return early. Otherwise, it creates a requestInfo object with the name, yearEstablished, loanAmount, and accountingProvider variables. It then sends a POST request to http://localhost:8080/balancesheet with the requestInfo object as the request body. If the response is not OK, it throws an error. Otherwise, it sets the assetsValue, profitOrLoss, and balancesheetInfo state variables to the corresponding values in the response data

6. GetFinalOutcome is called when the user clicks the "Submit Application" button. It checks that all the required state variables are not empty. If any of them are empty, it returns early. Otherwise, it creates a requestInfo object with the name, yearEstablished, profitOrLoss, assetsValue, and loanAmount variables. It then sends a POST request to http://localhost:8080/finaloutcome with the requestInfo object as the request body. If the response is not OK, it throws an error. Otherwise, it sets the finalOutcome state variable to the corresponding value in the response data

My steps taken (Backend)

1. Create 3 separate folders, one for main service, decision service and accounting service
2. Under each backend service path, create a requirements file by entering this command: pip3 freeze > requirements.txt


User
Application (React.js)
1. Fill Business Details & Loan amount
2. Select Accounting provider
3. Request Balance Sheet
4. Submit Application
Frontend and backend communicate with each other - via Http requests
Backend (Main) In Python
HTTP Requests
HTTP Requests
Accounting
will provide a balance sheet for a selected business of the user
Decision Engine
where the final application will be submitted to present the outcome of the application

My steps taken (main.py)

1. This is a Python Flask application that serves as a middle layer between front-end and two backend services: an accounting service and a decision engine service. The application has three routes defined:
    a. / - A simple message is returned when this route is accessed
    b. /balancesheet - This route handles a POST request with JSON data containing information about a company's balance sheet, sends the data to the accounting service, and returns the balance sheet information
    c. /finaloutcome - This route handles a POST request with JSON data containing information about a loan application, including business details, profit or loss, assets value, and loan amount. The application uses a rule engine to process the balance sheet information and determine whether the loan application is approved or not. If approved, the loan application information is sent to the loan assessment service, and the result is returned. If not approved, a message explaining the reason is returned
2. The application uses the Flask-RESTful extension to define the routes as resources, and the Flask-CORS extension to allow cross-origin requests. It also imports two modules, model, and rules, that contain classes used to represent and process the loan application and balance sheet data. The application uses the requests library to send HTTP requests to the 2 services

My steps taken (accounting.py)

1. This is a Flask app that generates a balance sheet for a business
2. The app has a class BalanceSheet that generates the balance sheet data for a given year. It initializes with the year and creates two lists profit_or_loss and assets_value containing random values for each month of the year. It also has two methods net_profit() and net_average_assets() that calculate the net profit and average assets from the generated data
3. The Flask app has a route '/getbalancesheets' that can handle GET and POST requests
4. If a GET request is made, it returns a JSON response with a message that says "Balance Sheet present"
5. If a POST request is made with JSON data that contains the year in the businessdetails, the app creates a BalanceSheet object with the year and generates the balance sheet data for that year using the to_dict() method of BalanceSheet class. The generated balance sheet data is returned as a JSON response.
6. The CORS module is used to allow cross-origin resource sharing, which allows the app to be accessed from other domains.

My steps taken (decisionengine.py)

1. The code is a Flask app that defines an endpoint at '/getfinaloutcome'. This app and determines the loan amount and whether the loan request is approved or rejected based on a pre-assessment score
2. If the pre-assessment score is 100, the loan request is approved, and the calculated loan amount is returned in the response along with the outcome set to True. If the pre-assessment score is 20, calculated loan amount is returned in the response along with the outcome set to False. If the pre-assessment score is 60, the loan amount is reduced by 40%, and the calculated loan amount is returned in the response along with the outcome set to True
3. If the request does not have a valid 'Content-Type' or the JSON payload is missing required fields, a response with an error message is returned