# **HSBC – RoboAdvisor Portfolio Rebalancer Service**

**Internal Technical Design Document**

**CPSC 319 – Team Formation**

Justin Tao, Yao Liu, Yuntian Wan, Scott Wang

Jingwei Zhang, Lauren Bentley, Sam McConnell

April 2019

Version 1.1

Table of Contents

[Overview 3](#_Toc536828172)

[Document Purpose and Goal 3](#_Toc536828173)

[Assumptions 3](#_Toc536828174)

[Programming Environment Setup 4](#_Toc536828175)

[Development Requirements – Backend and Database 4](#_Toc536828176)

[Development Requirements – Frontend 4](#_Toc536828177)

[Architecture 5](#_Toc536828178)

[System Architecture 5](#_Toc536828179)

[Software Architecture 5](#_Toc536828180)

[Architecture Diagram 6](#_Toc536828181)

[Deployment and Operations Design 7](#_Toc536828182)

[Google Cloud Platform 7](#_Toc536828183)

[Continuous Integration and Continuous Deployment 7](#_Toc536828184)

[Data Design 8](#_Toc536828185)

[Portfolio Preference Model 8](#_Toc536828186)

[Rebalance Recommendation Model 9](#_Toc536828187)

[Entity Relationship Diagram 10](#_Toc536828188)

[API Design 11](#_Toc536828189)

[Portfolio Preference Management API 11](#_Toc536828190)

[Portfolio Fund Rebalancing API 13](#_Toc536828191)

[Swagger Documentation 14](#_Toc536828192)

[Algorithm Design 15](#_Toc536828193)

[Fund Rebalancing 15](#_Toc536828194)

[Fund Recommendation Ranking (Stretch Goal) 15](#_Toc536828195)

[User Interface / User Experience Design 16](#_Toc536828196)

[UI Mockups 17](#_Toc536828197)

[Notable Tradeoffs 21](#_Toc536828198)

[Message Queues 21](#_Toc536828199)

[Notable Risks 21](#_Toc536828200)

[Failure to Rebalance Assets 21](#_Toc536828201)

[Appendix 22](#_Toc536828202)

# Overview

RoboAdvisor Fund Rebalancer is a fund and asset management service which aims to help HSBC customers make informed decisions regarding their investments, and to provide high-level investment recommendations. Using the RoboAdvisor Fund Rebalancer service, customers will be able to review their portfolios with HSBC, set a portfolio rebalancing preference which helps determine how their assets are allocated within their portfolio, and ultimately rebalance their portfolio to realign investments with desired risk tolerance and asset allocations.

# Document Purpose and Goal

This document is intended to record all technical aspects of the Fund Rebalancing APIs and systems required by RoboAdvisor to sufficiently execute the desired requirements. The document details the architecture, back and frontend design, and other systems required to implement the fund rebalancing service. Where appropriate, required versions for dependencies and software requirements will be indicated. A list of all abbreviations used in this document can be found in the appendix under **Table A**.

# Assumptions

RoboAdvisor Fund Rebalancer is a service dependent on other HSBC services to provide the necessary data. This includes services which track customer’s portfolios, services which track all assets available for purchase and their performance over time and other metrics, services which execute transactions to buy or sell an asset among many other services. Since RoboAdvisor relies on all these HSBC services to be functional and available in order to provide customers with the ability to store portfolio preferences and execute rebalancing recommendations, the assumptions are that HSBC will reliably provide these services as stated in their initial requirements contract document in order for RoboAdvisor Fund Rebalancer to function as intended.

Another important assumption to note is that security will be provided by HSBC. RoboAdvisor Fund Rebalancer does not currently implement security such as Oauth or API keys. API requests currently only require a customer Id header to be provided along with the correct request body. It is assumed that HSBC will provide the proper authentication once RoboAdvisor Fund Rebalancer is handed over to HSBC on project completion.

# Programming Environment Setup

RoboAdvisor Fund Rebalancer is a Java Spring Boot MVC application with a MySQL database and a React frontend. The required development tools are indicated in the **Tables** **1** and **2** below.

## Development Requirements – Backend and Database

***Table 1*** *– Tools and systems required for backend and database development*

|  |  |  |
| --- | --- | --- |
| **Tool** | **Version** | **Comments** |
| IntelliJ IDEA IDE | Community 2018.3.4 | Powerful IDE for Java based development |
| Docker | Community 18.09 | Container manager for local development |
| Postman | 6.7.2 | API testing tool |
| Java | 8 | Base language for Spring |
| Springfox Swagger2 | 2.9.2 | API Building, Designed, and Documenting tool |
| Flyway | 5.2.4 | Database migration tool |
| Spring Boot | 2.1.2 | Framework for Spring based applications |
| MySQL | 8.0.14, latest | Latest official docker image of MySQL |
| Adminer | 4.7.1, latest | Database management, latest official docker image |
| Apache Maven | 3.5.3 | Dependency and build manager for Java |
| GitLab | Web | Git version control and DevOps integrated in one |
| Draw.io | Web | Diagram drawing tool for UML and Flow charts etc |

## Development Requirements – Frontend

***Table 2*** *– Tools and systems required for frontend development*

|  |  |  |
| --- | --- | --- |
| **Tool** | **Version** | **Comments** |
| Visual Studio Code | 1.30.2 | Powerful text editor for various development needs |
| React | 16.7 | JavaScript library for building user interfaces |

# Architecture

## System Architecture

The major systems involved in RoboAdvisor Fund Rebalancer are the Spring Boot Application hosted on GCP’s GAE, cloud database provided by GCP’s CloudSQL, and the Fund System (also hosted on GCP) provided by HSBC in order to provide customer and fund information as well as execute financial transactions.

## Software Architecture

RoboAdvisor Fund Rebalancer is a Spring Boot application which implements Spring MVC. The application is required to store customer’s portfolio asset allocation preferences, provide and alter recommendations for portfolio rebalancing, and finally executing these recommendations. As with other MVC applications, RoboAdvisor employs model, view, and controller components. In addition, a service layer is added to allow for additional business logic handling.

The model component serves are the primary layer which interacts with the models and data. The two key models required are the *Portfolio Preference* object and the *Rebalance Recommendation* object. The model component will serve these objects to provide implementations for storing, retrieving, updating and other needs required to work with these objects.

The view component is involved with the representation of the data for the viewer. The primary output for this component will be the data retrieved in JSON format. This JSON object will then be passed to the frontend where it will be displayed visually on the UI.

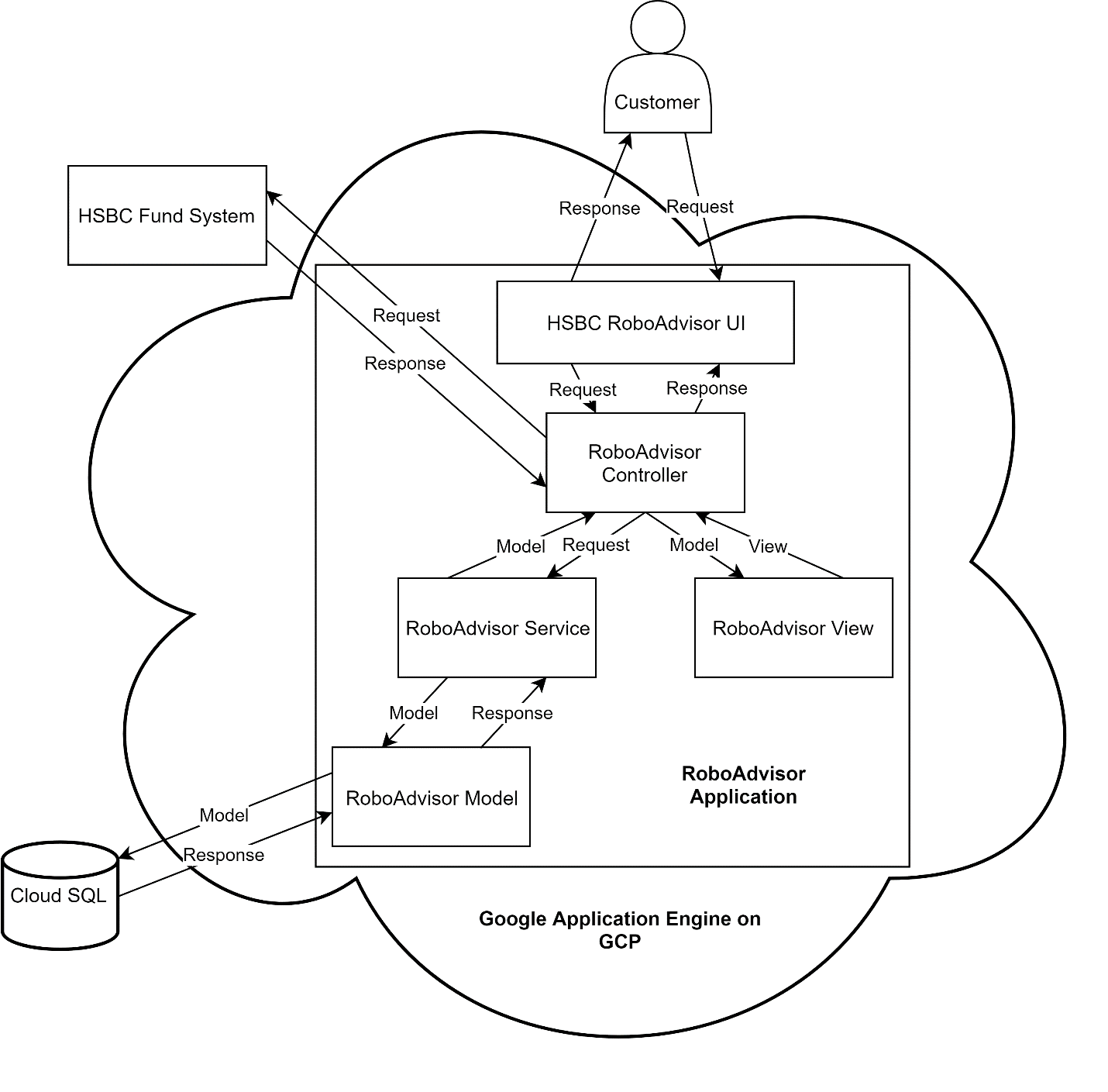
The controller component is central component which manages connections to and from the rest of the different components. REST endpoints will be handled in the controller component, receiving incoming requests and where appropriate, providing response bodies and the corresponding response codes. The controller acts as the central relay for input, output, and validation between the model, service, and view layers.

The service layer is an intermediate layer between the controller and model components. The service layer provides service segregation and is required to perform complex business logic with the model and repository to provide the required models requested by the controller. The inclusion of a service layer abstracts out the business logic from the controller and therefore Fund Rebalancing services can be used by other different services, not only the Fund Rebalance controller. The controller component would communicate the request with the service layer which in turn will communicate with the model component. The received object would then be manipulated if required and passed back to the controller for the view component.

These system and software interactions are best illustrated in the architecture diagram following.

## Architecture Diagram

***Diagram 1*** *– System and Software Architecture diagram for RoboAdvisor Fund Rebalancer*



# Deployment and Operations Design

## Google Cloud Platform

HSBC has requested that RoboAdvisor Fund Rebalancer be hosted on GCP. RoboAdvisor Fund Rebalancer services are currently deployed onto GCP’s serverless platform Google Application Engine. This allows for easy management, fast scalability, high availability as well as good affordability. GAE’s Service Level Agreement agrees to a monthly availability of at least 99.95%.

RoboAdvisor Fund Rebalancer backend and frontend applications and services are currently deployed in a 2-instance configuration on the flexible environment plan in the closest available Google recommended region – us-west2 region. Auto-scaling for instances is enabled for up to 20 instances. Upon deployment, GAE ensures that traffic is redirected properly and that the previous version is stopped as to not incur additional costs. Patching and upgrades will be administered as new versions to be deployed to GAE.

The CloudSQL database is configured as a MySQL 2nd Gen 5.7 database located in the Google recommended region – us-west1-c region. It is configured with 1vCPU, 3.75GB of memory, and 10GB of SSD Storage, the smallest standard CloudSQL MySQL configuration available.

Connections and routes are provided by GCP. Database connection is established by the generated Instance connection name.

RoboAdvisor Fund rebalancer can be accessed at <https://hsbc-roboadvisor.appspot.com/>.

Backend APIs can be targeted at <http://fund-rebalancer.hsbc-roboadvisor.appspot.com/>.

Swagger doc for API located at <http://fund-rebalancer.hsbc-roboadvisor.appspot.com/swagger-ui.html>.

## Continuous Integration and Continuous Deployment

We are looking into the possibility of CI/CD. Currently, all RoboAdvisor Fund Rebalancer services are manually deployed through GCP’s Cloud SDK CLI. We plan to use Gitlab’s native CI/CD service as our source code is already stored with Gitlab. This CI/CD offering is dependent on Google’s Kubernetes Engine. Additional details for GKE setup and selected configurations will be updated on this document once implemented.

# Data Design

Database management is provided by Flyway and Adminer. Flyway scripts permit easy database migration and verification. Adminer provides an easily accessible UI for MySQL. Our target database is named “roboadvisor”.

RoboAdvisor Fund Rebalancer functions primarily with two main models. The portfolio model and recommendation model. These models will be stored in MySQL. Their design is indicated below.

## Portfolio Preference Model

The main model in RoboAdvisor Fund Rebalancer is the *Portfolio Preference* model. This model stores each customer’s portfolio preference including how much deviation is allowed (between 0 – 5%), the portfolio type (“FUND” or “CATEGORY”), as well as the portfolio’s desired asset allocations.

Each *Portfolio Preference* object is composed of a unique integer portfolio Id, a portfolio type, an integer deviation and an array of assets and their respective desired percentage.

The “portfolio” table and columns can be created in the “roboadvisor” DB with the SQL command found below.



## Rebalance Recommendation Model

The *Rebalance Recommendation* model stores the required transactions in order to satisfy a portfolio’s desired asset preference. The model also stores a recommendation Id and an associated portfolio Id so that further adjustments can be made for a specific recommendation prior to executing the transactions.

Each *Rebalance Recommendation* object is composed of a unique integer recommendation Id, an integer portfolio Id, and an array of *transaction* objects.

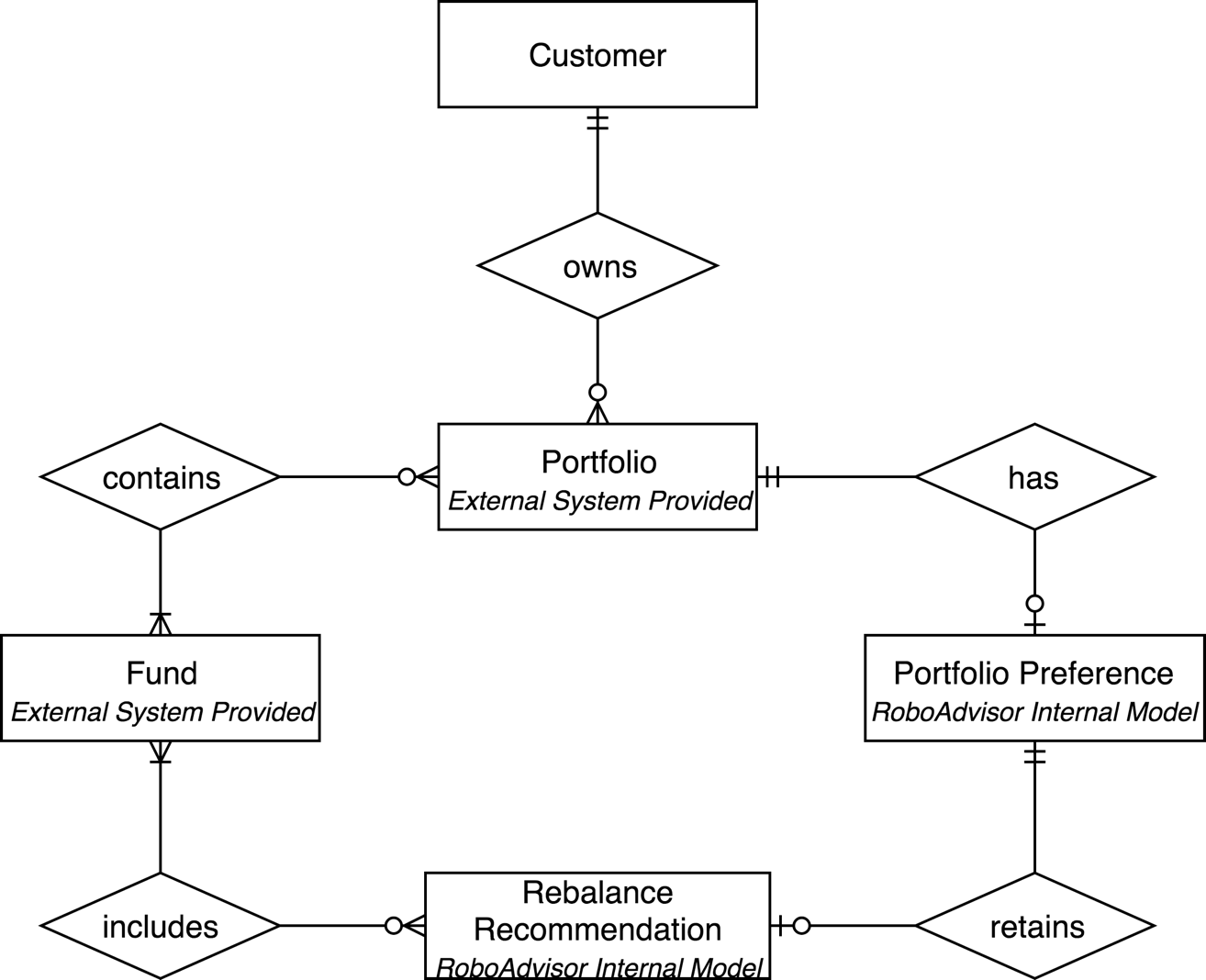
A *transaction* object is a JSON object which contains an action (“BUY” or “SELL”), a fund Id to be acted on, and finally the number of units of the fund to be acted on by the action.

The “recommendation” table and columns can be created in the “roboadvisor” DB with the SQL command found below.



The ER diagram below helps illustrate the different models and how they interact.

## Entity Relationship Diagram

***Diagram 2*** *– Entity Relationship Diagram for RoboAdvisor Fund Rebalancer*

# API Design

RoboAdvisor Fund Rebalancer APIs generally fall under two categories. APIs which serve the purpose of setting, reviewing or modifying the portfolio’s desired asset allocation preference, and APIs which serve the needs to modify or execute rebalancing recommendations and transactions. All APIs must be called with a specific x-custid header to serve the purpose of authentication. It is expected that HSBC will eventually replace x-custid with a proper authorization header. Standard HTTP response codes and error are employed. Expected successful response code will be 200 OK and 201 CREATED. Expected error codes will be 400 BAD REQUEST, 401 UNAUTHORIZED for missing x-custid, 403 FORBIDDEN for wrong x-custid, and 404 NOT FOUND.

## Portfolio Preference Management API

**POST /roboadvisor/portfolio/{portfolioId}**Creates a portfolio preference for the provided portfolio Id which includes the deviation, portfolio type, and desired asset allocation.

Sample Request Body, Expected Response Code – 201 CREATED  


Sample Request Body (Stretch Goal), Expected Response Code – 201 CREATED  


**PUT /roboadvisor/portfolio/{portfolioId}**Updates a portfolio preference for the provided portfolio Id which includes the deviation and desired asset allocation.

Sample Request Body, Expected Response Code – 200 OK  


Sample Request Body (Stretch Goal), Expected Response Code – 200 OK

**GET /roboadvisor/{portfolioId}**Retrieves a portfolio preference for the provided portfolio Id.

Sample Response Body, Expected Response Code – 200 OK

**PUT /roboadvisor/portfolio/{portfolioId}/allocations**Update the asset allocation distribution for the provided portfolio Id

Sample Request Body, Expected Response Code – 200 OK  


Sample Response Body  


Sample Request Body (Stretch Goal), Expected Response Code – 200 OK  


Sample Response Body (Stretch Goal)  


**PUT /roboadvisor/portfolio/{portfolioId}/deviation**Update the desired deviation limit for the provided portfolio Id.

Sample Request Body, Expected Response Code – 200 OK  


Sample Response Body  


## Portfolio Fund Rebalancing API

**POST /roboadvisor/portfolio/{portfolioId}/rebalance**Returns a *rebalance recommendation* object containing the recommended *transactions* to rebalance the portfolio to desired asset allocations, along with the created recommendation Id.

Sample Response Body, Expected Response Code – 200 OK  


As part of the stretch goal, a list of recommended transactions will also be provided for the given specified category. The UI will then modify the list of recommended transactions should the user prefer to purchase one fund over another.

Sample Response Body (Stretch Goal), Expected Response Code – 200 OK  


**POST /roboadvisor/portfolio/{portfolioId}/recommendation/{recommendationId}/execute**Executes the recommended transactions for the provided recommendation Id and portfolio Id.

Sample Request Body, Expected Response Code – 200 OK  


**PUT /roboadvisor/portfolio/{portfolioId}/recommendation/{recommendationId}/modify**Update the recommended transactions for a provided recommendation Id and portfolio Id.

Sample Request Body, Expected Response Code – 200 OK  


Sample Response Body  


**POST /roboadvisor/portfolio/rebalance/ranking**Create a list of suggested transactions for the given category and budget

Sample Request Body, Expected Response Code – 200 OK  


Sample Response Body – 200 OK   


## Swagger Documentation

RoboAdvisor Fund Rebalancer is employing Springfox Swagger2 for straightforward and accessible API documentation. Swagger annotations are used throughout the source code to provide necessary information for the swagger-ui to render an informative and practical document for all API endpoints offered. The swagger document is currently located at <http://fund-rebalancer.hsbc-roboadvisor.appspot.com/swagger-ui.html>.

# Algorithm Design

Two key algorithms will be used in RoboAdvisor Fund Rebalancer to help customers determine how they should rebalance their portfolio to attain their desired portfolio asset allocation preference.

## Fund Rebalancing

The fund rebalancing algorithm will determine which assets will need to be sold or bought and how many units of that asset will be required in order for the customer’s portfolio to reach their desired portfolio asset allocation preference. The algorithm is as follows:

1. For the current portfolio, for each asset, calculate the asset’s portfolio proportion given the asset’s performance and amount of asset owned.
2. For each asset, calculate how much of that asset must be purchased or sold in order to attain desired asset allocation limits.
3. Report back the recommended transactions in order to rebalance the portfolio.

## Fund Recommendation Ranking (Stretch Goal)

This algorithm would only apply for customers who have a portfolio type of CATEGORY. The portfolio’s asset allocations would consist of risk categories rather than specific funds. Should a customer need to buy a new fund they do not currently own in a given risk category, HSBC would like to rank the recommended funds available for purchase. The yearly performance weights will be provided by HSBC. The algorithm is as follows:

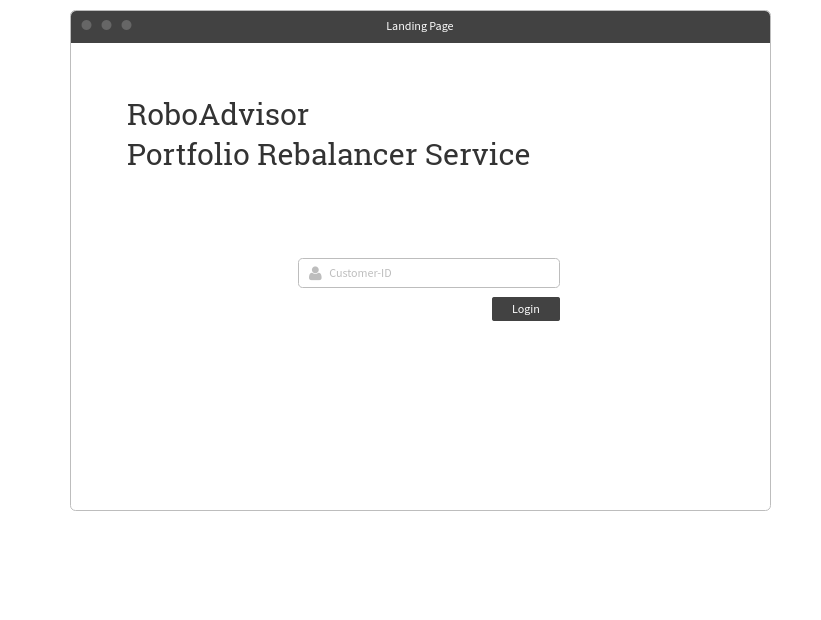
1. For the given risk category, determine which funds are available for purchase in that category.
2. For each fund, calculate the recommendation score based on the following sum of
   1. 5-year performance weight x 5-year average return
   2. 10-year performance weight x 10-year average return
   3. 3-year performance weight x 3-year average return
   4. 1-year performance weight x 1-year average return
   5. Sum of each asset composition sector weight x sector performance
3. Report back the ranked funds from highest to lowest score for recommended funds for purchase.
4. Transactions will buy and sell funds proportionately to the respective funds’ category percentage. For example, if 2 funds belong to category 3 and they occupy a 15/85 split of that category, then corresponding sell transactions will also see in proportionate amounts.

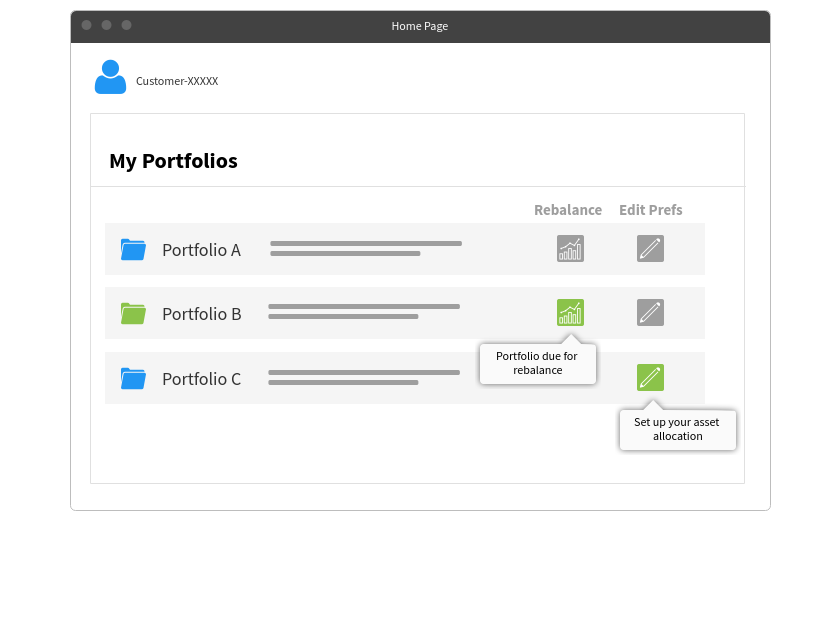
# User Interface / User Experience Design

In designing the User Experience Flow we aimed to maximize clarity while minimizing clutter and user pain points. We designed and sketched UI wireframe on whiteboard in order to draft user stories and validate correct user workflows. Then UI mockups were drawn with a consistent color scheme and interface in mind. A basic user workflow is as follows:

* As an HSBC customer seeking to rebalance their portfolios, they will first log into the Portfolio Rebalancer via the Landing Page (Diagram 3) by entering the customer ID. Upon successful login, the user will be directed to the Home Page (Diagram 4) in which a list of all currently owned portfolios will be shown.
* Each portfolio will have status icons indicating whether each portfolio is due for rebalance, and another set of status icons indicating if the corresponding asset allocation preferences have been set.
* Selecting a portfolio will bring the user to the Portfolio Page (Diagram 5) in which a list of all associated funds will be displayed. Each fund’s current percentage as part of the total portfolio will be calculated and displayed in graphical form (donut chart) and plain numerals. If no asset allocation has been set for the portfolio, a panel on the right will be active and display an alert to the user as a reminder to set their asset allocation.
* Clicking the button will bring up a modal window (Diagram 6) for the user to adjust his/her target asset allocation and the maximum deviation percentage.
* After the asset allocation has been successfully set, any portfolio that deviates outside of the chosen deviation range triggers another alert (Diagram 7) to remind the user to rebalance portfolios.
* Upon accessing the Rebalance function, the user will receive a list of recommended transactions to balance their portfolio (Diagram 8). The user can adopt the recommendations and execute or modify it accordingly.

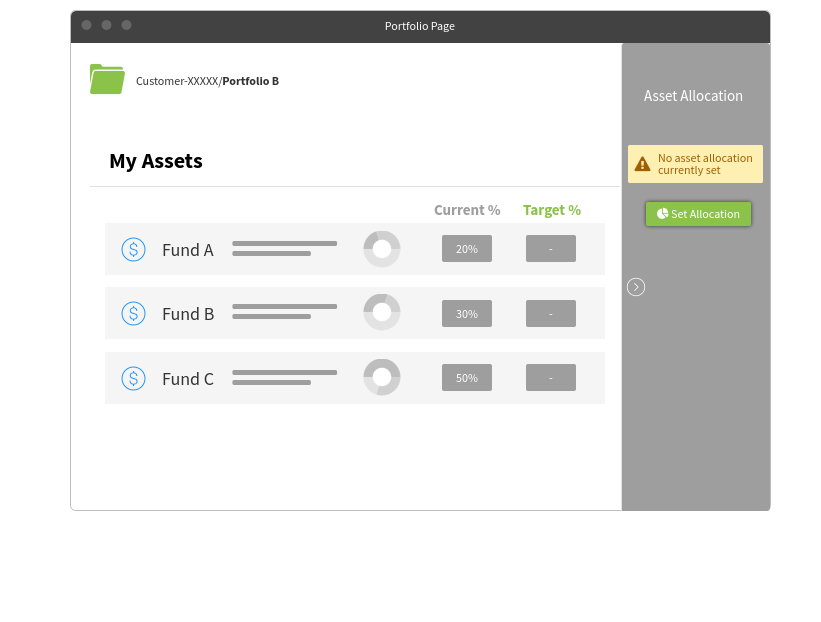
## UI Mockups

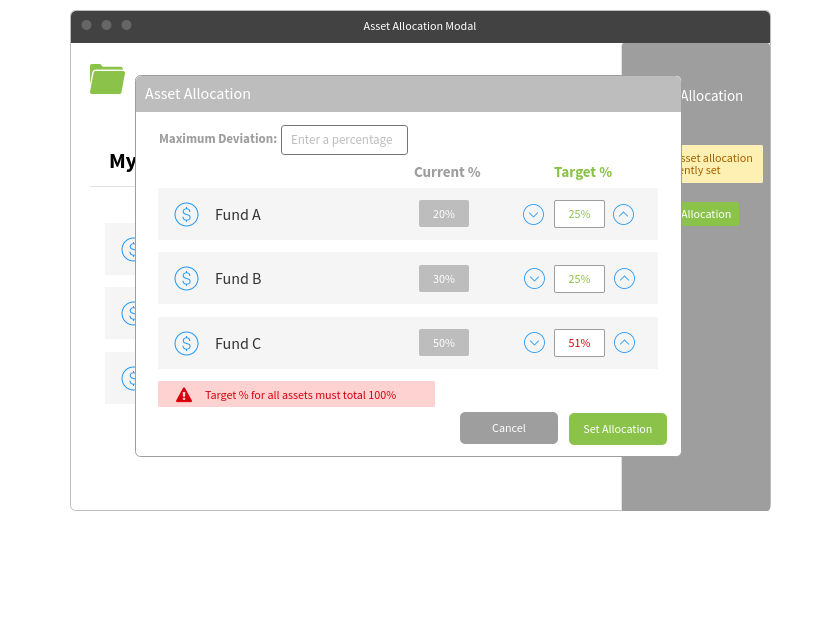
***Diagram 3*** *– RoboAdvisor Portfolio Rebalancer Initial Landing Page* *\*Note: User can login using their customer id. (It will be replaced with HSBC login system later.)*

***Diagram 4*** *– RoboAdvisor Portfolio Rebalancer My Portfolios page*

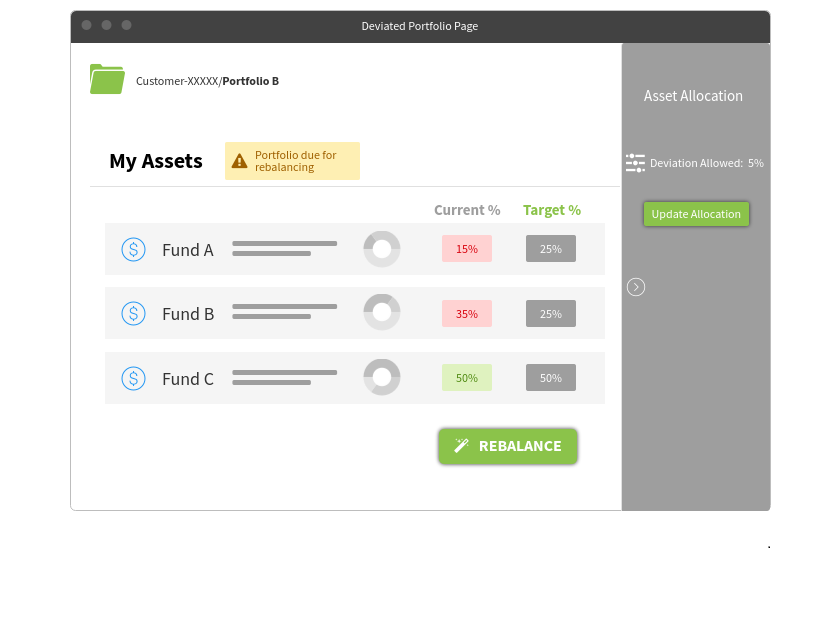
*\*Note:*

* *Color of portfolio icon indicates status. (Green indicates that it exceeds the and is due for rebalance. Blue indicates that it does not exceed the deviation.)*
* *User can also edit the preferred deviation by clicking the edit button.*

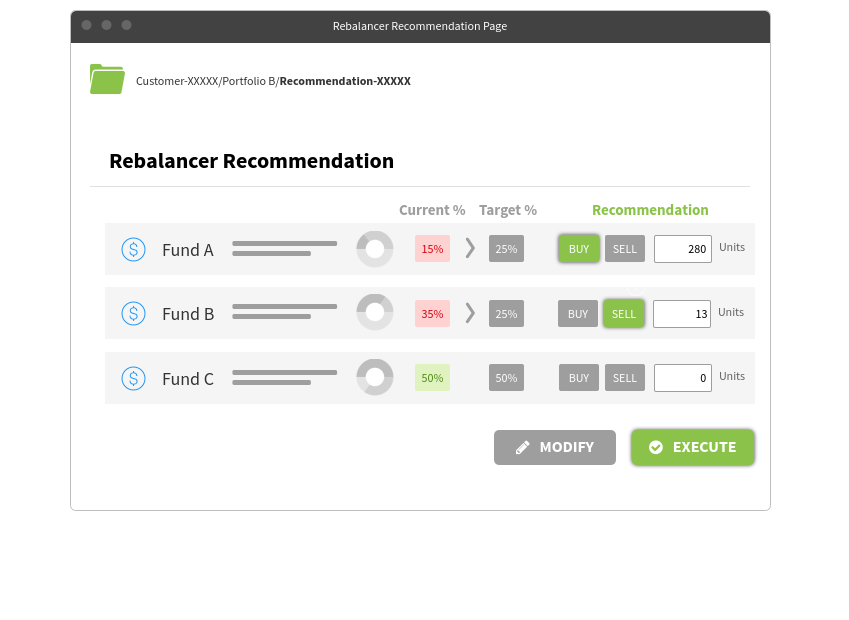
***Diagram 5*** *– RoboAdvisor Portfolio Rebalancer My Assets page in a selected portfolio*

***Diagram 6*** *– RoboAdvisor Portfolio Rebalancer asset allocation modal for a given portfolio*

*\*Note: The sum of all funds’ share should be exactly 100%, otherwise a warning will be shown and the “Set Allocation” button will be disabled.*

*****Diagram 7*** *– RoboAdvisor Portfolio Rebalancer deviated portfolio triggering rebalance*

*\*Note: By clicking the “REBALANCE” button, our back-end will rebalance the portfolio for that user.*

***Diagram 8*** *– RoboAdvisor Portfolio Rebalancer recommendations page*

*\*Note: Modify feature is part of the stretch goal*

# Notable Tradeoffs

## Message Queues

Message queues, such as Google’s Cloud Pub/Sub, RabbitMQ or Apache Kafka, provide an [asynchronous](https://en.wikipedia.org/wiki/Asynchronous_communication) [communications protocol](https://en.wikipedia.org/wiki/Communications_protocol), meaning that the publishers and consumers of the message do not need to interact with the message queue at the same time. A message placed onto the queue is stored until a consumer retrieves and executes it. The use of message queues brings numerous benefits. We can have multiple publishers and consumers so that users are never reliant on just one server instance. If a particular instance of a multi-instance services goes down, another instance of the same service can just as easily consume the message and complete the necessary tasks. Similarly, if there is suddenly an increase in published messages, the number of consumers can easily be scaled up to match. Another benefit would be in the resiliency of messages. Only when a consumer successfully consumes a message does the message get removed from the queue. Should some failures occur, the message is returned to the queue for another consumer to process. This ensures that all published messages are taken care of and no published messages goes unheard (provided there are enough functional consumers).

RoboAdvisor Fund Rebalancer would be more reliable and scalable with the help of the Message Queues. However, we do not employ the use of these messaging services in this project with the assumption that the HSBC will handle the reliability and scalability concerns when they take over our project.

# Notable Risks

## Failure to Rebalance Assets

A system failure in RoboAdvisor Fund Rebalancer is a critical risk to a customer’s portfolio. A properly balanced portfolio is important for maintaining a health risk reward ratio and for financial security. If rebalancing no longer functions, the portfolio risks becoming lopsided or risk heavy. This would result in excess investment risk and put pressure on customers to find another method to rebalance, or worse, move assets away from HSBC to another financial institution. This would be ill advised for HSBC. Therefore, it is essential that the systems in place for RoboAdvisor Fund Rebalancer be well tested and maintained to ensure value for both customers and HSBC.

# Appendix

***Table A*** *- A List of Acronyms and Abbreviations used throughout this document*

|  |  |
| --- | --- |
| **Term/Acronym** | **Definition** |
| **API** | Application Programming Interface |
| **CRUD** | Create, Read, Update, and Delete - the four basic functions of persistent storage |
| **CI/CD** | Continuous Integration, Continuous deployment |
| **CLI** | Command Line Interface |
| **DB** | Database |
| **ER Diagram** | Entity Relationship Diagram |
| **GAE** | Google Application Engine, GCP’s serverless hosting platform |
| **GCP** | Google Cloud Platform |
| **GKE** | Google Kubernetes Engine |
| **IDE** | Integrated Development Environment |
| **MVP** | Minimum Viable Product |
| **REST** | Representational State Transfer |
| **SDK** | Software Development Kit |
| **SQL** | Structured Query Language |