Team: Imposters 4156 T1

Name	Yumeng Luo	Kevin Hwang	Sang Jun Chun	Shengtan Mao
UNI	yl4655	jch2169	sc4658	sm4954

Part 0:
Meeting time: Nov. 4th
Changes:

1. using Auth0 API for authentication.
2. Google Places API for geolocation
3. Using multiple store APIs to search up desired items
4. Baseline price is determined by the item the user originally looked at. The user receives alternate recommendations based on the item he first selected.
5. Change UI to responsive webpage
6. Add weekly email notification to users
7. Change reward and charity to email

Disagreements: we were advised by our IA to produce an average price using store APIs for a product based on the user location. We could not find any suitable API that completes this task for us. If we are to develop this ourselves, there would be many difficulties. It is difficult to produce a price per unit as many products are sold in bulk or varying quantities. We also have to consider how to account for the quality difference between items. The price average should be specific to the quality of the particular item.

Instead of this, we changed how our tool operates. The user would search up an item, and he would select the item he wants from the list. The item he selected would serve as the baseline price for calculating the savings. The tool would then give recommendations to cheaper products of similar types. The final item he chose would be treated as the item he purchased.

Part 1:
1.1 Description:
Our tool is called “saving and spending’s tracker” which will keep track of the user's spending and savings. Once a user specifies a product that they want to purchase, the tool will search for nearby alternatives in the same category that are cheaper and can fulfill the same needs. If the user did choose the alternative, the tool will calculate the difference in price between the initial item that the user chose and a cheaper item available nearby that we suggest. (For example, one user is planning to buy coffee. The user will search for coffee and select the product he/she wants, say, a $5 can of coffee, as delivered by the store APIs. The tool will set that as the baseline price and propose alternatives available nearby, such as a cup of Mcdonald’s coffee for $2, a cup of ShareTea latte for $4. It will then record the difference between the baseline price and what the user ultimately chooses as the user’s saving. That is, if the user initially chose a $5 coffee but follows our recommendation and chooses McDonald’s coffee for $2, he/she has saved $3.) At the end of each week, the tool will show balance accumulated so far and send an automated email to users. In addition to saving balance, the email also includes suggestions for ways to spend this savings: Purchase something expensive or donate the savings. (Email

shows the user spent a total of $1000 and saved $100 at the end of the week and suggests the user to reward himself with a gaming mouse or donate to a charity group of choosing.)

1.2 Users for our tool include:

1. Users who are on a tight budget and want to know cheaper alternatives to make ends meet
2. People who're saving money to purchase particular items or occasional indulgence
3. Advanced user who wants to customize which purchases to save from
4. People who wish to give back to charity in small ways (with money saved from daily expenses)
5. Sellers who're promoting their products that's a cheaper alternative to an existing one

1.3 Saving data:

Our tool will require each user to register and login before using and will be forced to timeout after 20 minutes of inactivity. We will be using the OAuth protocol from Auth0 API to authenticate the users. The account balance, spending and saving's history of each individual user will be saved persistently using a database and will only be shown to the account owner after login. The database also saves the emails of each user for weekly emails.

1.4 Demonstration:

Our current plan for demonstration is through a webpage. An example demonstration could go like this:

1. User visits the webpage
2. Selects a category and types in an item
3. The app presents items that meet the criteria of what the user searched for
4. The user chooses the most appropriate item from the search result, and the app sets that as the baseline price
5. The app then shows cheaper alternative options to the user's initial selection if it is available in the nearby store. It will use Google Maps API to find if there are applicable stores near the user.
6. The user then decides whether to buy cheaper products or not. If the user chooses instead a cheaper option, the saving amount (baseline price - cheaper item price) is calculated.
7. Savings and spendings are updated to new amount
8. Change time to Sunday, the tool sends users an email about savings accumulated this week and suggests users spend it on expensive items or donations.

1.5 API

For API usage, we are planning to use "Charity Search" to search for charities the user can donate their savings to."UPC lookup API" to find product ID by name, and "Walmart Open API","Best Buy" and "Wegman" to get product prices by product ID. To find cheaper products (and where to buy them) that are alternatives to what users specify, we will use "Google Maps Places" to search for nearby stores that carry similar, cheaper products. We will largely rely on the Google API to collect and analyze nearby stores' proximities and prices (via cost indicator). We will use Auth0 API for authentication, and "Mailgun API" for sending weekly savings emails.

Part 2:
Our MVP should be able to keep track of spending and prompt cheaper alternatives; keep track of savings and reach user set goals; customize alternative option's categories. The detailed user stories are as following**:**

USId1. As **a user with a tight monthly budget**, I want to **know and choose cheaper alternatives for daily expenses** so that **I can stay under budget and save money**. My condition of satisfactions are:

a. I can set a budget I want to spend under
b. The tool correctly records my spendings
c. The tool proposes a cheaper option that also satisfy my need
d. The tool correctly records my savings
e. Following the tool's suggestions, I can spend less than my budget permits

USId2. As **an everyday user looking for occasional indulgence**, I want to **save on daily expenses** so that **I can purchase more expensive items**. My conditions of satisfactions are:

a. The tool correctly records my savings
b. I can set a goal to save up to
c. Following the tool's suggestions, I can save more than my goal

USId3. As **a user looking for cheaper alternative options**, I want to **specify what categories of purchases are extraneous to me** so that **I can save more in such categories**. My conditions of satisfaction are:

a. I can customize the items I feel are extraneous
b. The tool proposes the cheapest alternatives in these categories

Aside from the above 3 user stories, we propose the 2 following optional user stories that should be completed if time permits**:**

USId4. As **a user who wants to give back in small ways**, I want to **know and choose what charity organizations are seeking donations** so that **I can donate some of my savings from daily expenses to a charity of my choosing**. My conditions of satisfaction are

a. The tool shows descriptions of potential charities I can donate to
b. I can specify which charity I'm donating to and what amount I am donating
c. The tool returns confirmation that the charity has received my donations
d. The tool makes my donation anonymously

USId5. As **a consumer product company focusing on cheaper goods**, I want to **have my products shown as cheaper alternatives for similar purchases by tool's users**, so that I **can effectively market my products to the right customer base.** My conditions of satisfaction are

a. My product is proposed to users as an alternative to a more expensive product
b. Consumers should be able to purchase from the platform (app, website etc)

c. The tool provides report on who / how many of my products are being sold

Part 3:

USId1. Common Case

a. The tester sets a budget amount.
b. The tester purchases multiple products and verifies that the tool correctly records spendings based on the products chosen.
c. The tester requests multiple alternative recommendations and verify those products fulfills the purpose of the original product and are cheaper in price.
d. Tester makes choices for alternative options for multiple products and verifies that the tool correctly records the savings based on the product chosen and the original product.
e. Once the budget amount has been reached, the tool should warn the tester budget reached and proposed alternatives should be “do not make this purchase”.
f. The tool will pass this test if the final monthly spending is less than the set budget, otherwise the test fails

USId1. Special Case

a. The tester tries to set the budget as a negative number. We expect an error that the user is trying to set an invalid value.
b. If the tester does not set the budget and proceeds, the tool will warn the tester the budget is not set.

USId2. Common Case

a. The tester sets a savings goal. Testers should not be able to set the goal to negative numbers.
b. The tester records purchases for multiple products and makes choices for alternative options and verifies that the tool correctly records the savings based on the product chosen and the original product.
c. Once the savings amount has been reached, the tool should notify the tester that the goal met.
d. The tool will pass this test if the final monthly saving is more than the set goal, otherwise the test fails

USId2. Special Case

a. The tester tries to set the goal as a negative number. We expect an error that the user is trying to set an invalid value.
b. If the tester does not set the goal and proceeds, the tool will warn the tester the goal is not set.

USId3. Common Case

a. The tester unchecks all categories as “extraneous”
b. The tester makes purchases in multiple categories and records the price for each alternative options
c. The tester checks all categories as extraneous
d. The tester makes the same purchases as before

e. The test passes if the price of the alternative option is lower than the recorded price before checking as “extraneous”, otherwise the test fails

USId3. Special Case

a. The tester makes a purchase with a negative number as price.
b. The test passes if the tool warns the tester the price is negative and does not record this purchase or propose an alternative. Otherwise the test fails.

Part 4:

For our backend facilities, we plan to use the following:

- Eclipse for our IDE (using Java JDK 11)
- Maven for our build tool
- SQLite for our database

For our frontend, we plan to use the following:

- Javascript

For testing, we plan to use the following:

- JUnit for unit testing
- Emma for code coverage

For other tools, we plan to use the following:

- CheckStyle for our style checker
- Spotbugs for our bug finder

For our APIs, we plan to use the following:

- Auth0 for authentication
- Google Places API for geolocation
- Mailgun API for sending weekly summary emails
- Walmart, Best Buy, and Wegman APIs for finding products & prices