**Deliverable**: In a single MS Word document “CSS360\_Implementation.doc”

**Description**:

Background: Class time will be provided for the exercise to check in on your specifications with your “customer.” The exact process for this check will be provided in class.

Assignment: Based on the feedback you received from your customer on your requirements, determine what changes to your implementation strategy and approach make sense.

Tasks: Given the requests for requirements changes and the new constraint/requirement, revisit whether your process still makes sense and if not, select a new one. Then revisit your architecture and module design, and develop a revised design and implementation plan which consists of the following parts -

1. For each module/component in your design (those that have changed, and those that have not changed), specify your strategy (build from scratch, purchase, or reuse with some modifications).
2. Explain your strategy decision.
   1. For example, if you are reusing software (e.g. software packages, third party components, course-grained reuse), provide one reason why you have chosen this particular approach over other technologies or options.
3. Applying a similar level of granularity used in the examples in class identify any specific technologies you will use (e.g., MySQL for backend database) and explain how the technology satisfies the design. If you don’t know because of uncertainty, either make a best guess or describe what you will evaluate to make the decision in the future.
4. Explain how you will manage the software development process involving multiple software versions. Refer to Sommerville Ch.25.

Write up a summary of your implementation plan beginning with a general summary of your plan, and then describing the specifics in a, b, c and d. Submit in Canvas using your team name in the title.

Parts **A** and **B**: apply on a per-module basis

Parts **C** and **D**: apply on a global basis

\*For pre-built Google components, check:

<https://en.wikipedia.org/wiki/Category:Google_services>

<https://developers.google.com/actions/>

Ex: DialogFlow - for intent matching; ImmersiveResponse - for visual output

###############################################################

{beginning of submission}

Summary

Our customers did not suggest any new requirements, however they were most interested with the parking and food truck tracking functionalities to our product. Overall, they were unsure whether they were going to be interested in the product if it were only a voice assistant application, however using a Actions on Google platform allows us to display some functionalities onto the user’s smartphone.

Google Student is a smart device application that provides a variety of functionalities to make UW Bothell students’ lives easier. While the primary mode of interaction will be voice, it shall also display some visual information on the user’s smartphone.

Implementing the Parking module

We will work directly with UWB to design and implement an API within Google Student which enables the Google Student client to fetch parking data from universities. The responsibility will be on the university to collect their own real-time parking availability data and store it in a way that is compatible with our API. Due to our partnership with UWB in the development of this API, no changes will be made to their existing system due to the fact that the module is directly modeled after their system.

Updated parking data is copied every five minutes from the university’s server onto the Google Cloud Machine Learning Engine, where we will digest the data to provide information to the user. The ML engine will analyze trends in parking availability in each lot for different times of the day and use that information to make predictions about what availability will look like in the future.

Why we chose this approach for Parking

We chose to partner with UWB in developing this API primarily because they are among one of very few universities across the US to have implemented a system that is able to identify the occupancy status of every individual parking space in real-time. This will also give other universities who want to implement a similar system the opportunity to model their system off an existing system that is proven to work, and to allow students to leverage that data more readily through Google Student.

Implementing the Traveling module

We will be using the Google web mapping service, Google Maps, to connect it to Google Student. We will be using the Google Maps API to integrate it. This will allow students to find out how long their drive to school will be and connects to the parking module to add on the estimated time it will take them to park their cars as well. We will not be using the Google Maps current interface and the students must give consent to Google Student to be able to share their location.

Why we chose this approach for Traveling

Due to the fact that the company we chose to go with is Google, it makes sense to use their web mapping service rather than going and purchasing one from another company or creating a whole other one.

Implementing the Canvas Integration module

We will **build** functionality within Google Student that leverages the Canvas API, a RESTful service, that delivers JSON responses over HTTPS (https://canvas.instructure.com/doc/api/).

Authentication will be handled by the Google Home app’s ‘Services’ functionality, as seen in Figure 1.

To illustrate the desired functionality, we will provide an example request and the associated response.

|  |
| --- |
| User: ‘Okay Google, talk to Google Student’  System: ‘Now connected with Google Student’  User: ‘Read me my upcoming assignments’  {PROCESSING}  System: ‘On Tuesday the 3rd, *Software Project Charter* is due for *Software Engineering*. On Thursday the 5th, *IP Law Report* is due for *Business Law*.’ |

Processing

* First, the user’s request is handled by the Natural Language Processing capabilities built in to Google Assistant (‘*Dialogflow*’)
* The user’s intent is determined, and transformed into a query, which is then submitted to the Canvas API
* The response is returned to Google Student, which then generates the response

Query generation

In this case, the query passed to the Canvas API will concern Assignment objects, particularly the due\_at attribute. Google student will default to a date range of 5 days, which can be overridden by the user specifying a range. If we were using a standard RDBMS, the query might look something like this:

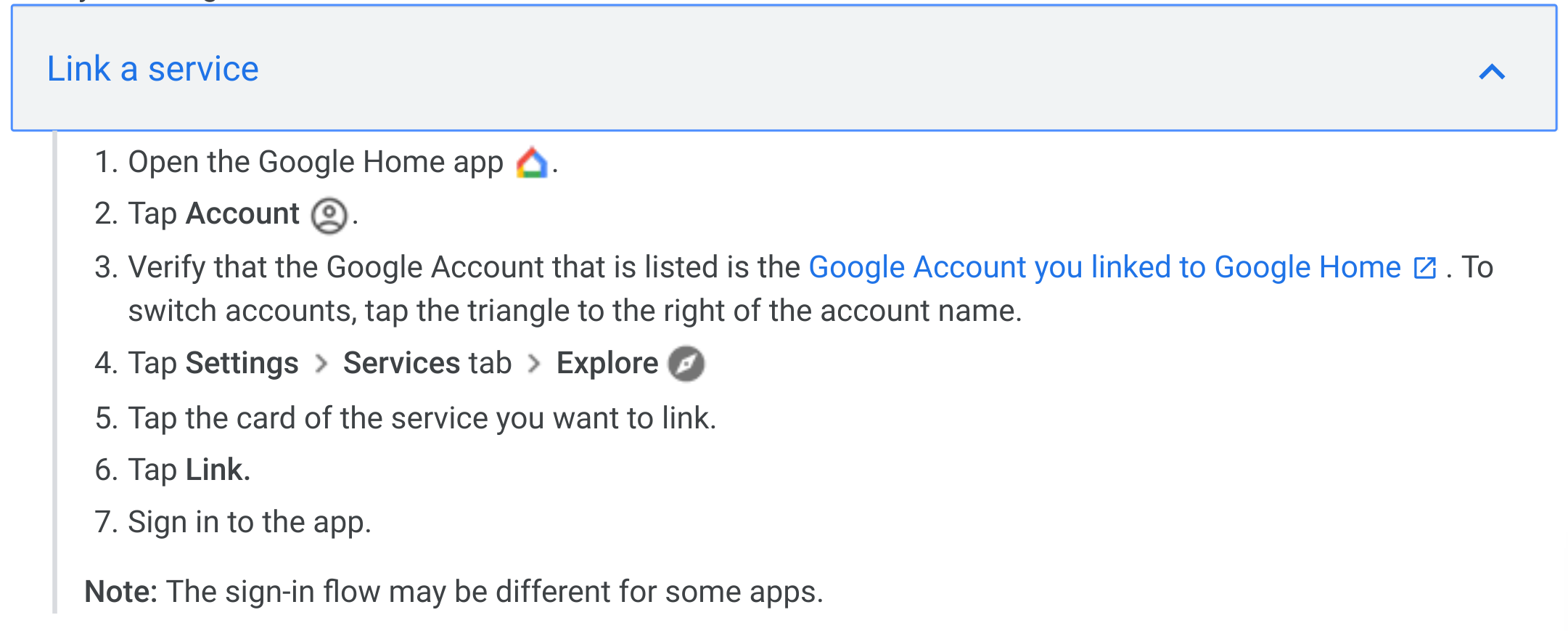
RETURN Assignment.name, Assignment.due\_at, Course.name

FROM Assignment JOIN Course on Assignment.course\_id = Course.course\_id

WHERE Assignment.due\_at > $NOW AND Assignment.due\_at < ($NOW + 5 days)

Figure 2 serves as an example of a real API query, used to return all Assignment objects associated with a particular courseID.

Figure 1. Linking Actions on Google with third-party services



Source: [https://support.google.com/googlehome/answer/7126338](https://support.google.com/googlehome/answer/7126338?co=GENIE.Platform%3DAndroid&hl=en)

Figure 2. An example Canvas API query



Source: https://canvas.instructure.com/doc/api/assignments.html

Why we chose this approach for Canvas Integration

Of course, using an API is far more straightforward than scraping text from the internet, so if an API exists, it is logical to use it. We could have chosen to purchase an existing module that generates queries to the Canvas API and interprets its responses, but as a large company, we will prefer to have control over this functionality, to fix errors and add functionality into the future. As the required logic is not too complex, it makes more sense to build this module ourselves, leveraging the API.

Implementing the Display module

The Actions by Google platform already contains a component that can resolve user queries to visual responses, known as Interactive Canvas. As described by Google, “This visual layer is an interactive web app that is sent as a response to the user in conversation” (https://developers.google.com/actions/interactivecanvas/).

For example, imagine that a particular parking query resolves to a visual response, showing the current utilization of both garages. First, the user’s intent is determined, and a query is passed to the parking module. Next, an *ImmersiveResponse* is received by the device that includes the URL of a custom web app and a state value. This state value contains variables, including the current garage utilization rates. This data is read by the custom web app and displayed, in visual form, to the user - all within the Google Assistant window.

This solution should be classified as building our own components on top of existing systems. The ‘infrastructure’ for creating these visual experiences already exists, but we will need to write the custom web app for each type of visual response we want to display to the user.

Why we chose this approach for Display

From the beginning, we chose the Google Assistant platform in part because of the rich set of existing functionality that it provides. By leveraging existing components, we are able to minimize the development time involved in providing visual results to users. This was not part of the core functionality of our application, and so we are more concerned with efficiency, and less concerned with having ultimate flexibility in how visual responses are handled.

Implementing the Food Truck Tracking module

For this module, we will have to work with UWB to get them to request data from the food trucks and store it. We will also need to implement an API within GS that will allow the student to receive information on which food trucks are available when and where. This module must be connected to Google Maps to be able to track the exact locations of the food trucks once they are on campus.

Why we chose this approach for Food Truck Tracking

We chose to work with UWB to be able to get the food trucks’ location data because they already manage which food trucks come in and out of the school and so it shouldn’t be too hard, as long as the food truck owner is willing to give information about the food truck’s locations, for them to save it in a database. Implementing a new API within Google Student makes the most sense because this is a unique functionality.

Technologies

* Python: Since the Google Assistant SDK supports this programming language, it makes sense to use it to code our product functionalities.
* GIT: Google is known for storing its code in a single-source large repository
* Google Assistant API for communicating between the devices, and tools given from the assistant
* Canvas API to get grades, calendar events, and other information from Canvas
* Google Maps API: allows integration of Google Maps service with Google Student
* UWB servers and services for location, events, and parking information

Managing Multiple Software Versions

Version Control

The first version of our product will be released and dedicated to only UW Bothell, and once we fix all the issues post-release coming from feedback from the users, we will start branching out to other universities through version upgrades. The second version might include different UI design, efficiency and removing errors, as well as compatibility to other Universities.

System Building

For system building, we will be using the tool of reporting to provide reports about the success or failure of the builds so as to also help after post-release to decide when it is the best time to move on with the product application to other universities. We will also be using test automation which will run tests to make sure the build doesn’t break from changes.

Change Management

When it comes to change management, cost will be taken a look at and made sure that there are enough benefits to the proposed changes. In this case, we may come up with a new system for checking for parking availability and so it’s up to the cost to determine whether it’s worth changing the old system to the new.

Release Management

Our form of release management will be of giving our finished product to UW Bothell as a minor release then through feedback will we change and better our product to then do a major release to many other universities.

{end of submission}

###############################################################