Title: Google Student

Group Name: CSS 360

Name of the Group Members: Borna Borhani, Jonathan Dycaico, Taylor Eyler, Behnam Khabazan, Hadassah Latchague, Gabriel Smith-Dalrymple

Executive Summary SINGLE-SPACED

<This describes the goals and objectives of your project, primary requirements (note: a bulleted list is fine), the selected SDLC process, timeline and a proposed team structure (roles and responsibilities). >

BREAK PAGE - DOUBLE-SPACED

**Section 1. Introduction**

Through this project, our goal has been to produce an aggregated, easy-to-use, information resource for UW Bothell students. Our measurement for success would be a number of active users equivalent to 25% of the student body within 2 years of release.

Industry

The industry for student resource applications includes the following types of players:

* Learning Management System (LMS) software vendors
* Information hub applications, similar to our own
  + Ex. Modo Campus
    - <https://www.modolabs.com/products/modo-campus/>
* Single-purpose websites, as an alternative to more convenient applications
  + Ex. MyUW + Canvas + OrgSync

The industry for Actions on Google / Alexa Skills includes the following categories of applications (<https://www.cnet.com/how-to/amazon-echo-most-useful-alexa-skills/>):

* Home automation / smart device management
* General information retrieval
  + News, Finance, Weather
* Interfaces to service providers
  + Food ordering, Ridesharing, Travel booking
* Audio entertainment
  + Podcasts, Music streaming, Radio, Audio games

Our platform intends to fulfill two purposes. First, it will act as middleware, combining information from multiple sources into one integrated system. Second, it provides users with a convenient way of accessing the most important information from this system. This application will be accessible where customers already are - on smartphones, tablets, smart speakers, smart TVs, and chromebooks.

**Existing applications**

In the UW Bothell student software ecosystem, Google Student doesn’t intend to offer a 1-for-1 replacement of any individual service. Instead, the application will provide an interface to the most crucial information from a number of existing applications that is easiest to serve in a voice assistant interaction format.

Canvas, for instance, is a complex application that can provide detailed information on assignments, courses, and students. It offers integrations with Panopticon and Zoom, Google Apps and Vericite. The Google Student application is not envisioned as a replacement interface for the entirety of what Canvas can provide, but rather a subset of useful functionality. For instance, Google Student could access information from Canvas’ API on the assignments that will come due the soonest for a student, and read out only a few attributes of those assignments. Users would still have to navigate to Canvas in their web browser to receive the most granular information on those assignments, but Google Student would still provide a useful function.

However, it is feasible that students who previously had the Canvas mobile app installed would find it redundant, once they could access the limited subset of information they need on their mobile device using the Google Student application. Many students find the mobile app much less useful than Canvas’ web interface. In a sense, this would entail Google Student replacing the Canvas app on these user’s phones. Even though Google Student only accesses a limited subset of the information available through the Canvas app, it may be that Google Student provides enough of the information they need in a more user-friendly format, resulting in the replacement of the Canvas app.

Google Student also is designed to provide commute estimates to users, using both traffic/routing information from Google Maps as well as parking availability information from the application’s parking module. In this case, Google Student is not providing 1-for-1 functionality, and should not be seen as a replacement for Google Maps.

In domains related to campus-provided information, such as upcoming campus events and the food truck schedule, no existing mobile application exists. Instead, this information is displayed on pages of the UW Bothell website. This information is simple, and so is well-suited to a voice-based interaction model. For accessing this information on a mobile device, we believe Google Student to offer a superior interface than the alternative of viewing charts on a mobile browser, and having to remember the various URLs. As such, we aim to replace the existing ‘applications’ here, at least for mobile users.

**Justification**

As UW Bothell students, we’ve experienced ourselves the pains involved with planning and managing student life. First, the Canvas app is not user-friendly, leading to a situation in which most students can only check on their courses from a desktop browser. In addition, due to parking constraints, commuting to campus on-time for for classes and events poses some difficulty. We have existing parking sensors, but the campus is not making that information useful for the student body. Beyond that, campus information on events and food truck availability is kept on websites that are difficult to find and read on mobile devices.

With the advent of voice assistants as development platforms, we have an opportunity to create a one-stop solution for students. We can integrate the various sources of information relevant to student life into one application that targets an entire range of devices. Google Student will help users plan their days and weeks, and reduce the pains involved in doing so on modern computing devices.

**Stakeholders**

1. Campus staff

* We will need to gain their support in order to pull data from the event and food truck calendars’ backends. We could scrape the data, but getting the data directly will decrease our chance of encountering errors. In addition, we will need their assistance/approval in order to access or provide an interface to the parking garage utilization data.

1. Application users

* We will do our best to provide an application that is actually useful. To gauge this, it would be prudent to conduct studies about how they are using the application, and perhaps interview some of them as well.

1. Google, Inc

* Our team will attempt to add value to the Google Assistant platform, in order to increase adoption rates and overall brand value.

1. Campus event organizers and Food truck vendors

* These stakeholders will see differing levels of revenue/attendance depending on how well our application informs students.

1. Students not using the application

* They may find that app users are quicker to find available parking, leading to an even bleaker parking outlook for non-users.

1. Instructure, the company that sells Canvas

* They may see a reduction in downloads for their mobile application.

**Section 2. Software Requirements and User Stories**

2.1 Software Requirements (your top, most critical for success set)

* to provide a list of functional requirements
* to provide a list of non-functional requirements

Note: make sure these requirements will be fully implementable (through the Context Diagram and/or the selected architecture) and will also be verified and validated by your testing plan.

2.2 User Stories (even if you are using plan driven, just for the practice)

<Based on the requirements, this section will include a set of a minimum of 4 user stories (each user story will be approximately 120 - 140 words or about half a page long, double-spaced).>

**Section 3. Context Diagram and Architecture**

This section will include the following:

* the system’s context diagram including all key system elements and external entities (with any data flows labeled clearly).
* detailed text description of the context diagram
* the system’s architecture with your SW system clearly identified and marked (diagram to be included - copied image will be accepted if the image fits the selected architecture.)
* detailed text description of the architecture(s)
* the justification of why the architecture(s) were selected.

**Section 4. SDLC Process**

This section will include the following: (note: this is for project management.)

* detailed description of the selected SDLC process (with the SDLC diagram)
* justification of why the SDLC process was selected
* what deliverable(s) each stage (or iteration) of the SDLC will generate
* a rough estimate how long each SDLC stage/step will take (give your assumption about project team size, see below on Team)
* code management: how the team will manage the release of the code (e.g. how often, how big/small each release will be, how the versions will be controlled, etc.)
* how the engineers plan to communicate within the project team and with stakeholders throughout the proposed process (and with customers, if necessary).

**Section 5. Testing Approach**

* to describe testing approach(s) your group is proposing.
* to show how the testing approach(s) will be linked back to the requirements and/or user stories (e.g. V model shown in the lecture slide)

**Section 6. Team Description**

* rough in a size and description of a proposed team, given your SDLC process, the size and complexity of the release, etc.
* to describe the proposed roles and responsibilities of the team members

Note: if you build and use a detailed “responsibility matrix” for each member of the project group – including specific activities and deliverables, you’ll get extra points

**Section 7. Conclusion**

This section will include the following:

* project deliverables and artifacts
* description of benefits and changes if and when your proposed project is successfully completed

**Your final paper must be submitted using the following format: • Use 11- or 12- point font size (Calibri or Times New Roman) • Single Spaced for the executive summary, Double Spaced for the group report • Bottom of each page (except the Title page): group name, page number (right aligned)**

{beginning of submission}

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

**Executive Summary (JON)**

<This describes the goals and objectives of your project, primary requirements (note: a bulleted list is fine), the selected SDLC process, timeline and a proposed team structure (roles and responsibilities). >

**Introduction (JON)**

* to provide a summary of SW engineering project goals and major objectives (of the proposed SW engineering project) to provide a detailed description of the industry and/or device/application (e.g. microwave oven, mobile payment market and industry, etc.)
* to describe current SW application(s) your project intends to replace (or enhance). This includes business/engineering functions the existing SW applications serve, platforms the SW applications operate, technical features, etc.
* to explain the reasons why your proposed SW system/application is needed to replace (or enhance) the current SW application(s), what your proposed SW system/application will add/change
* to describe stakeholders (e.g. users that may use the SW application(s), people who will influence the decisions regarding your proposed SW project, people who will be influenced by the product/service your proposed SW project will produce.)

**Software Requirements**

Functional:

1. The system shall issue an alert if a parking lot is completely full for the general public (not including handicapped or EV charging spots).
2. The system shall alert you every time there is a calendar event.
3. The system shall send information to your phone or a display system based on the response.
4. The system shall calculate the amount of time needed to get to class on time.
5. The system shall implement a GUI-based application to utilize functionality without using voice prompts.
6. The system will alert the user when favorited food trucks will arrive on campus.
7. The system will read to the user a listing of the most imminently-due assignments.

Non-Functional:

1. The system shall not disclose any personal information that falls under FERPA restrictions.
2. The system shall ask you security questions when logging in.
3. The system shall have the ability to know where the user is located for navigation purposes.
4. The system shall have the ability to integrate with CANVAS.
5. The system shall keep a history of the conversation, and responses.
6. The system shall keep a record of food trucks that arrive on campus and when.
7. The system shall be able to accommodate simultaneous usage by the entire UW Bothell student body.

**User Stories**

User Story #1 (Reminders):

Lisa is a full-time student at UW Bothell who works part time to keep up with her bills. When she gets home from school and work, she is often too mentally drained to keep track of everything she needs to do for school, but Google Student is there to remind her of what assignments she has to work on that night. The next morning, she’s tired and woke up late for school, but Google Student notified her of exactly when she needed to leave for school to get there on time, and which parking spot she should head to in order to get to class as quickly as possible. The following afternoon, Lisa is all caught up with her assignments, so Google Student reminds her to take some time to herself and relax after all her hard work.

User Story #2 (CANVAS Integration):

Dan is a full-time student that lives on-campus and does not like using voice assistant applications. He tries to do all of his assignments digitally and lives on his laptop and smartphone. He requests for Google Student to let him know what assignments are due in the next week so he can get them done in time. Google Student gets the data from CANVAS and asks if Dan would like to view or hear the assignment’s names and due dates. Dan requests to hear, and GS starts listing out all of the assignments that are categorized as incomplete on CANVAS and their due dates. If new assignments are added, Dan gets a notification on his smartphone. Dan is happy he knows exactly what is due this week because now he will not have any surprise assignments.

User Story #3 (Parking & Travel):

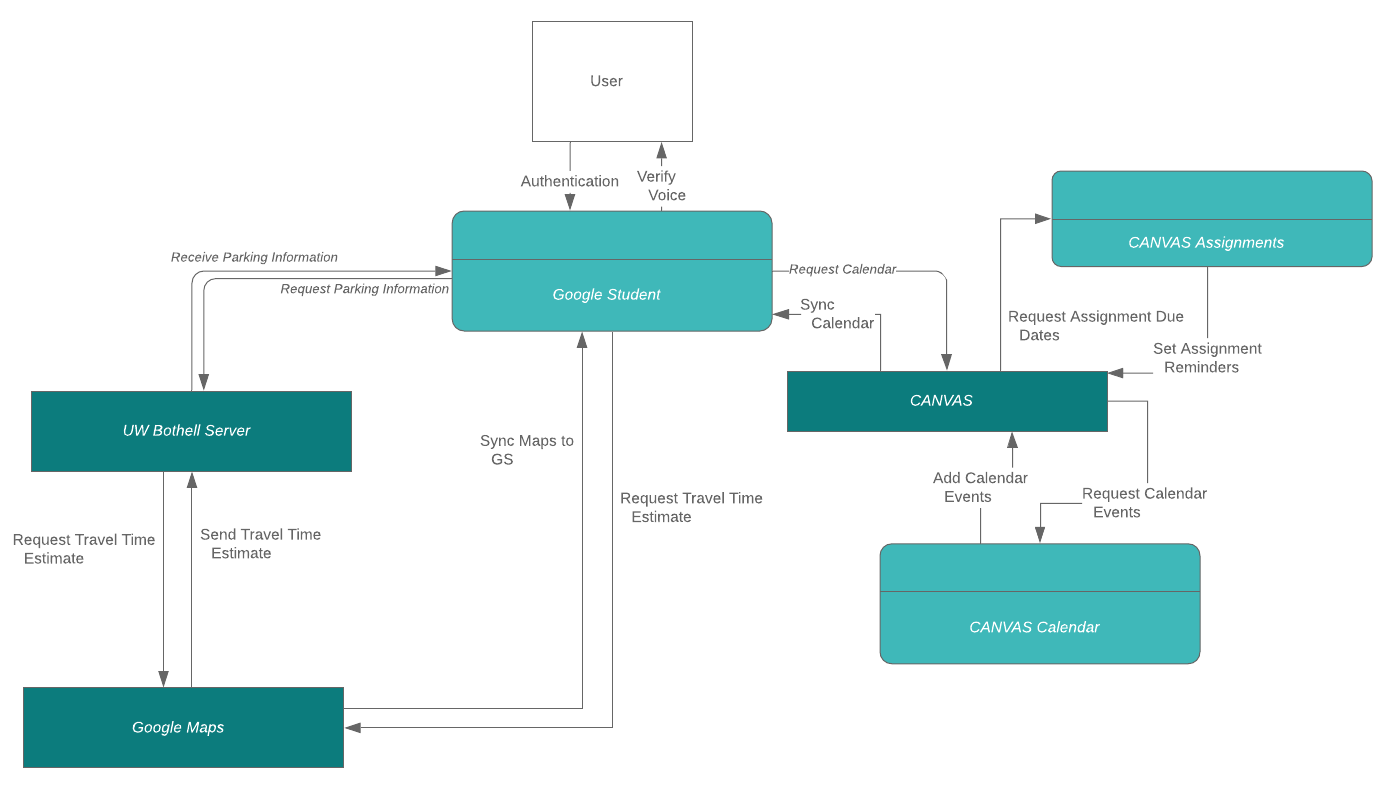
Chris is a full-time student who is also a full-time bartender. He lives off campus and struggles getting to class on time after he works night, which is pretty usual for him. Chris wants to know exactly when he should live his house to be able to get to class on time, making sure it considers the time it will take him to park his car in the desired parking lot. He wakes up at 8am on a Monday morning, and his first class starts at 8:45am in UW2. He requests an estimate on the time at which he should leave to get to the closest parking garage to his class so that he is not late. Google Student asks the user whether they want to view or hear the information. Chris needs a quick answer so he asks to hear the information. Google Student requests information from UWB server and Google Maps to give Chris an estimate on when he should leave his house to get to school on time. Chris leaves for school immediately, and gets to his 8:45am class on time for once.

User Story #4 (Food Truck Tracking):

Patricia is a part-time student who has a full-time job and is a full-time mom to two toddlers. She splits her time between taking her children to daycare and working. Patricia’s life is extremely busy, and she often doesn’t have time to go check which food trucks are available between her classes. After dropping off her children to daycare, she is off to school. Once she gets to school, Patricia asks Google Student on her smartphone which food trucks are available on campus for that day. Google Student retrieves the food truck location data from UWB server and asks Patricia if she would rather view or hear the information. Patricia requests to view the food trucks information, which makes Google Student display a list of the food trucks that will be available that day and the times at which they are planning to stay on campus. Patricia favorites the Poke food truck that will be on campus between 11am and 2pm. Patricia receives a notification when the food truck arrives with its exact location on campus. Patricia is excited about picking up her lunch between her classes.

**Context Diagram**

Figure 1: Context Diagram



This context model shows how Google Student interacts with other systems in its environment. The user first needs to authenticate themselves to be able to use their Google Student. Once they are authenticated by the system, which systems are interacted with depends on the functionality the user requests. If the user requests parking information, GS will fetch parking data by interacting with the API to request parking information, which will also trigger a request for a travel time estimate from Google Maps. If the user requests assignment or calendar information, GS will interact with CANVAS to sync the calendar and send out appropriate reminders of upcoming assignments.

**Architecture**

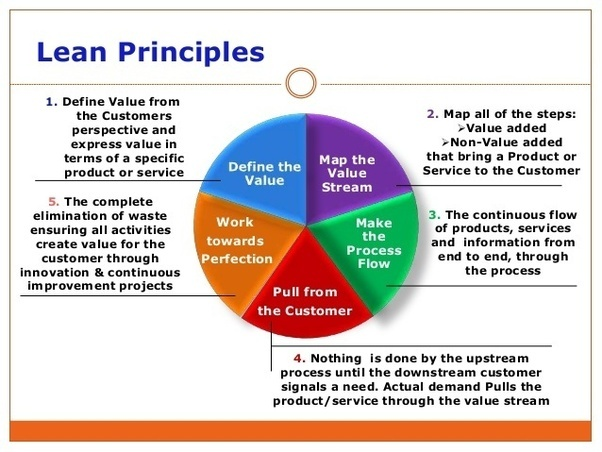
As shown in Figure 2, we chose the Layered Architecture pattern because it will allow us to be modular with the services that the product can provide by only needing to adjust a single layer as we add/remove services. This pattern organizes the system into separate layers, with related functionality associated with each layer. If we add more services that the application offers it doesn’t need to affect the utility or the configuration layers. This approach will also support incremental development.

Figure 2: Layered Architecture Pattern

|  |
| --- |
| Voice Assistant Application Phone Application |
| **Configuration Services** |
| Group Management Application Management  Identity Management |
| **Application Services** |
| Parking Availability Weekly Food Truck Schedule & Availability  Upcoming Events Reminders Degree Auditing Calls & Messaging |
| **Utility Services** |
| Logging in/out Authentication Interfacing User Storage |

**SDLC Process (TAYLOR)**

We chose to use Lean Software Development to develop Google Student because it will allow us to be flexible and optimize efficiency. It is focused on removing everything that is unnecessary, and results in continuous communication with the customer to ensure the end result is exactly what they need.



Lean development follows seven guiding principles:

1. **Eliminate Waste:** eliminate anything that does not add value to the customer by avoiding unnecessary code, ambiguous requirements, task switching, or ineffective communication
2. **Build Quality In:** focuses on producing quality code through pair programming practices, test-driven development, or incremental development with constant feedback
3. **Create Knowledge:**  encourages teams to retain and share valuable knowledge through extensive documentation, code reviews, training, or knowledge sharing sessions
4. **Defer Commitment:** avoids making decisions without the necessary data, but instead remains flexible until a reasonable determination can be made
5. **Deliver Fast:** remains cognizant of common roadblocks to delivering a product to a customer quickly (such as thinking too far in advance about future requirements or over-engineering solutions) and actively works to avoid them by building the simplest solution and seeking continuous feedback from the customer
6. **Respect People:** promotes respect for people by addressing work-related issues as a team, encouraging healthy conflicts, and empowering each other to do their best work
7. **Optimize the Whole:** focuses on viewing the team as a ‘value stream,’ which represents the sequence of activities required to design, produce, and deliver a product or service to customers, and making changes to optimize the team through that lens

We expect the following deliverables to be generated during the following stages of our development cycle:

* **Define the Value (*1-2 days*):** an interview takes place between the Google Student team and the customers, where the customer provides feedback on current design plans, as well as provides any new feature ideas not yet captured
* **Map the Value Stream (*1 week*):** the team identifies each step required to produce the full Google Student system, and a Kanban board is created to manage workflow among the team
* **Make the Process Flow (*3-4 weeks*):** as the workflow progresses, bottlenecks are identified and alleviated, and flow is optimized by avoiding task switching and keeping the number of tasks in progress on the team to a minimum
* **Pull from the Customer (*1-4 weeks*):** once a flow of work (iteration) is completed, new work is only started when there is a demand for it and the team has capacity for more work; requests pulled from the customer enter a queue of flows to be completed by the team
* **Work towards Perfection (*1 week*):** any activities that are found to not create value for our customers are eliminated; continuous improvement projects are implemented

Lean development favors a lower number of tasks in progress at any given time, so most processes will be completed as a team. Daily stand-up meetings will take place to ensure all team members are on the same page of what they are building and how it will function.

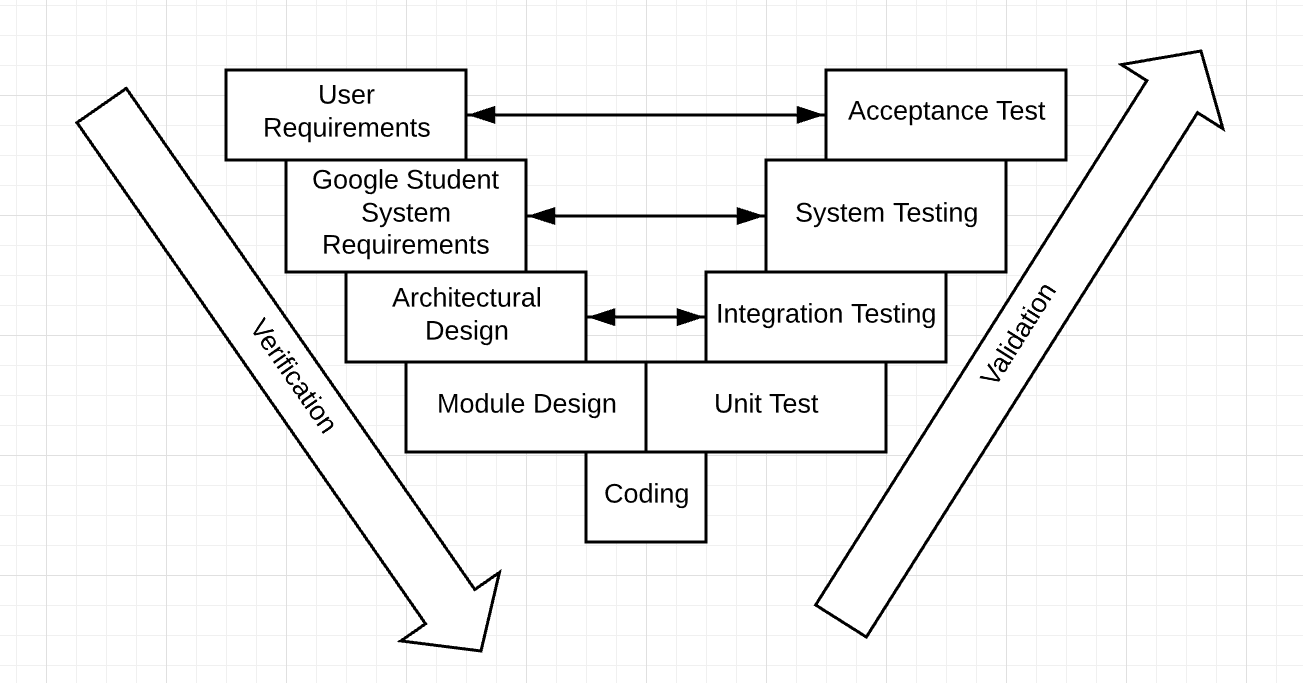
Google’s proprietary version control system, called Piper, will be utilized to allow for the merging of changes checked in by each team member. As each module is completed, it will be deployed to devices that have changed the “Testing State” setting on their Google Assistant-enabled device to “Alpha” or “Beta.” As this is a company-owned project, the codebase will not be provided to the public, although an API will be provided for universities to provide data to the Google Student service.

**Testing Approach**

The desired features we will be testing Google Student for include:

1. Quick response
2. Easy accessibility
3. Increased student productivity
4. CANVAS integrated
5. Phone connectivity
6. Internet connectivity
7. Confirmation messages when task is completed
8. Modularity in design
9. Testing each feature individually:
   1. Parking
   2. Traveling
   3. Food Truck

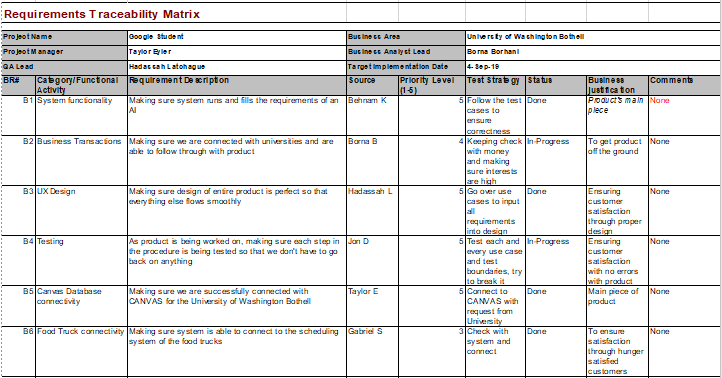
The testing approach we will be using to test our code is test-driven development. It is an approach which interleaves testing and the development of the code, which would allow us to have a better program structure and reduce costs. For acceptance testing, we plan on letting UWB students test the system to decide whether or not Google Student is good enough to be deployed.



**Team Description**

As the product is purely software we won’t need to harness any of the hardware teams so we will be looking to start with a small nimble team of 10-20 professionals, and will expand to bring in additional resources when we are nearing the end of the product if we want to include voice functionality, and localization functions. Since our procedure will be following the lean software development path, we will have small teams within. A development team, testing team, UX Design team, one or two mobile app specialists, and one or two systems integration specialists.

The development team will be in charge of how to develop the product itself and how it can be an artificially intelligent product that recognizes and responds to voices. The mobile app specialists will make sure it connects to the product in the first place. The development team will work with the UX Design team for requirements that need to be hit to make sure the design includes everything that is needed. All teams will be reliant on the systems integration specialists once the role of the design team is done for the most part. After everything seems to be done, the testing team will ensure everything works and all the user requirements are hit. Though the testing team will also work with the systems integration specialists while working on the system to ensure everything is working along the way.

****

**Conclusion**

The final deliverable for this product would be to have it be successfully implemented and connect with the CANVAS database, the parking availability system and the food truck availability system. You should be able to ask for a visual or an audio message from the product of assignments, reminders, calendar events, parking availability, and food truck availability.

Benefits that will come out of the release of this product will most definitely be happy students because they will have kept track of their schedules via reminders and less stress will be given to them with lesser time to find parking before class starts. Other than happy customers, universities will have a more organized system with better connections with their students eventually making them want to invest more in this product. With more investment from UW Bothell, more universities will want to use this product with its game changing success with a more convenient student life at their universities.

**References**

“7 Guiding Principles of Lean Development.” *LeanKit*, 2019, leankit.com/learn/lean/principles-of-lean-development. Accessed 7 June 2019.

(<https://www.cnet.com/how-to/amazon-echo-most-useful-alexa-skills/>