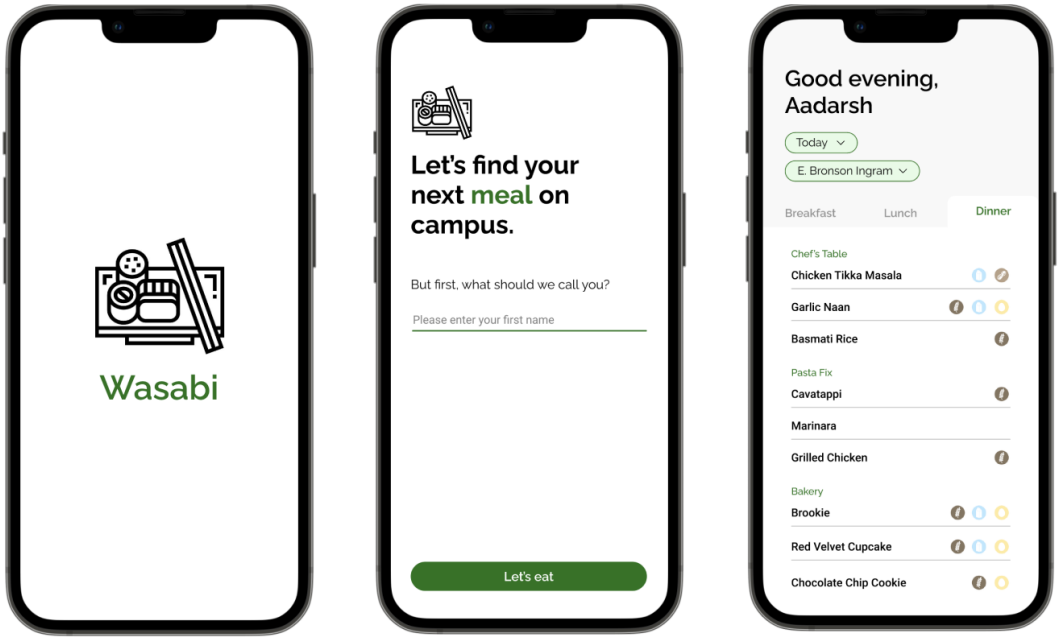
Wasabi

Towards End-To-End Nutrition Scraping From CBORD For Accessible Collegiate Dining Experiences At Vanderbilt University



Aadarsh J., Raahul N., Izzy H.

ENGM 3700 — Fall 2021

November 30, 2021

Table Of Contents

Executive Summary

Wasabi is a cross-platform, quality-of-life application for finding, understanding, and viewing Vanderbilt dining menu options. The motivation behind this project is increasing the accessibility of Vanderbilt campus dining options to the end-user. The current method for viewing dining options is the website: [Net Nutrition Vanderbilt](). However, upon an initial user study, it was found that this website is slow, hard-to-use and inconvenient. For instance, to view lunch options at Rand, one must go through three screens to see what is available. This makes it incredibly time consuming, considering the latency of the page, and difficult to compare several options at once. This has several impacts: (1) students are unable to view and choose different options based on their daily preferences; (2) people with dietary restrictions or other related issues may have a harder time finding accessible options; and (3) students become generally disillusioned with dining options at Vanderbilt due to the perceived difficulty of even finding food options to compare.

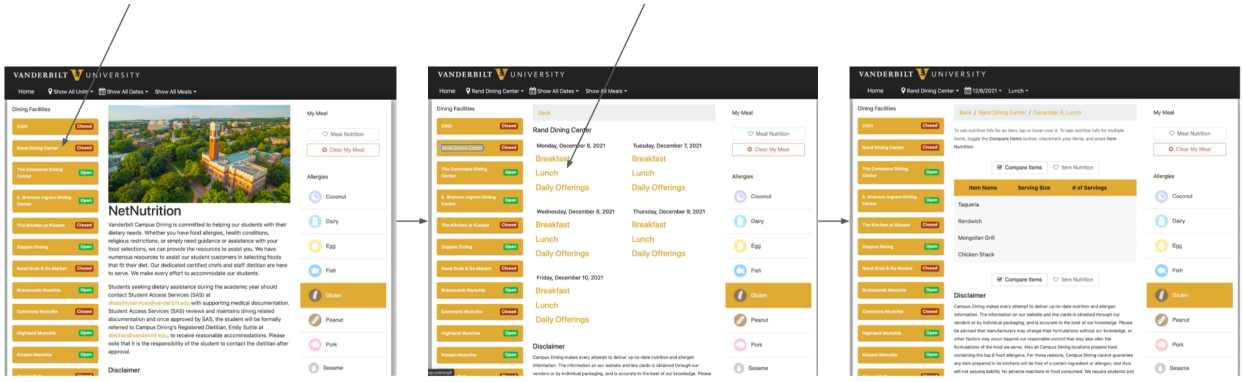


Figure 1 : Current Net Nutrition Workflow Of Finding Lunch Options At Rand

Herein, we propose an application, Wasabi, that makes dining at Vanderbilt wholly accessible to the Vanderbilt student. We plan on doing so in two main ways: (1) building a web scraping architecture that pulls information from the Vanderbilt Dining website; and (2) creating an easy-to-use application that displays this information in a one-stop-shop fashion. We believe this will bridge the gap between the stochastic and unknown nature of knowing what food options are available to eat on-campus, due to the poorly built website that is the currently used NetNutrition.

To achieve this, within this report, we lay out a five-week plan in achieving our goal to build a cross-platform application, meant for both iOS and Android. To begin, we estimate a $223 total cost of materials needed for this project. These materials include prototyping tools, development environment tools, programming toolkits, UI / UX frameworks, as well as developer

accounts to build and deploy our application. A further discussion of the tooling may be found both in the **Materials and Expenditures** and **Prototypical and Technical Specification** sections.

Next, we divide our project into six main stages: (1) prototype, (2) web scraping architectural development; (3) application development; (4) integration; (5) testing; and (6) deployment. More information about each step can be found in the **Prototypical and Technical Specification** section.

The division of labor is split in such a way that each individual on the team is responsible for the tasks for which they have the most experience and strength. In particular, Izzy will serve as the UI/UX Designer and the Product Owner. Raahul will be the engineer in-charge of the web scraping architecture and system design as well as application testing. Aadarsh, then, will be responsible for mobile application development and application deployment. More information on discrete tasks for each individual may be found in the **Responsibility Division** section.

Our **Risk** section enumerates nine potential risks our project may encounter, such as technical, personal, and team-based dynamics. In particular, each risk is associated with a likelihood, impact, consequence, and mitigation strategy. A net consequence is computed as 0.0446, and a total project cost is computed as $232.95 due to an added contingency fee of $9.95. So, our project is low cost and has mild risk.

Our project will utilize an Agile methodology with three main sprints, as described in our **Planning** section. The project will commence on November 4, 2021 and conclude on December 9, 2021 with buffer days built into the sprints. The first sprint is focused on product design and user research. The second sprint is reserved for developing the web scraping logic and the application. The final sprint is reserved for application testing and deployment of our final application to respective market stores.

Anticipated problems include: poor communication, personal issues, and general mismanaged team dynamics. Future direction of this work, as enumerated in **Future Work**, discusses on how we plan to scale this application up by allowing for personalization, increased functionality, as well as introducing this application to other schools that use a similar system.

Prototypical and Technical Specification

Wasabi is planned to be an end-to-end solution, which scrapes web application data and delivers it to an accessible application for the end-user. In particular, building an application of this scale requires six main stages: 1) prototype, 2) web scraping architectural development; 3) application development; 4) integration; 5) testing; and 6) deployment. Our codebase and other technical details may be found on our public GitHub repository: https://github.com/aadarshjha/wasabi. The motivation behind this section is to further explain each material used from the Materials list, as well as how each material will help us complete each story from our Kanban board (visualized later in this document).

To address the prototyping stage (1), Izzy will primarily lead the design, UI / UX, as well as a small-scale case study to receive feedback and ideas on how to improve the application. In order to prototype and design the application, Izzy will use a tool called Figma (https://www.figma.com/), which is free and open source. This tooling will allow us to prototype a usable application and will allow for initial user testing and feedback. A draft of the prototype can be seen below:

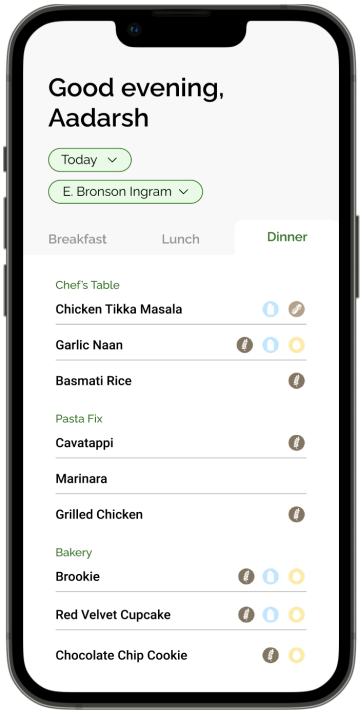


Figure 2 : Mockup of Main Menu Page

We will conduct usability testing and studies of the prototype by surveying a random sample of individuals and allowing them to use our prototype designed in Figma. We will then add or remove additional features in order to create a scaffolding for a final application.

To address the web scraping step (2), Raahul will be in charge of the system design and architecture of the application, specifically focusing on building the infrastructure for scraping the NetNutrition website from CBORD. To do so, Raahul will primarily work within the Visual Studio Code environment, using JavaScript (JS), and related JS libraries such as Request and Cheerio, in order to parse and architect the scraping logic. Raahul will ensure the code built is reusable and largely extensible to any large changes, and can easily be integrated into step (3).

Within steps (3) and (4), Aadarsh will primarily use React Native, a cross-platform framework that allows a developer to write a singular code base in JavaScript and compiles to iOS and Android applications. With React Native, Aadarsh will translate the finalized designs from Izzy into a functional MVP, with no logical functionality. Then, once Raahul has finished his work in step (2), Aadarsh will then use tooling like Redux to implement state management and dependable, predictive, as well as centralized state within the application, encapsulating both user data (such as their name), as well as cache and organize data that is fetched from the web scraping codebase.

Within Step (5), Raahul will use free JS libraries such as Jest (a JS testing library) and Cypress (an end-to-end testing JS testing framework) in order ensure: 1) the end-to-end nature of our application; 2) the web scraping logic is fully functional; and 3) the design, functionality, and feel of the actual mobile application across the different platforms for which it is available.

Finally, within step (6), a build will be produced for the react native application to be productionized in the Android and iOS environments. This build will then be deployed via our Apple and Google accounts, respectively. After which, our project will be completed and our deliverable will be available for Vanderbilt students.

In order to self-contain and organize our work, we will use a free Notion account to create a shared workspace upon which we will collaborate, discuss, and note down ideas, new stories, as well as product feedback.

Materials and Expenditures

In this project, the material and the related expenditures directly derive from the technological stack that will support our web scraping-based, cross-platform mobile application. In particular, the only cost of our project will come from the **Deployment Services** we utilize to publish our application to the app store

Materials Cost			
Item	**Cost**	**Quantity**	**Cost**
Figma Account (https://www.figma.com/)	$0.00	1	$0.00
React Native (https://reactnative.dev/)	$0.00		$0.00

Visual Studio Code (https://code.visualstudio.com/)	$0.00	2	$0.00
XCode (https://developer.apple.com/xcode/)	$0.00	1	$0.00
Android SDK (https://developer.android.com/studio)	$0.00	1	$0.00
Redux (https://redux.js.org/)	$0.00	1	$0.00
Jest (https://jestjs.io/)	$0.00	1	$0.00
Cypress (https://www.cypress.io/)	$0.00	1	$0.00
Notion (https://www.notion.so/)	$0.00	1	$0.00
Slack (https://slack.com/)	$0.00	1	$0.00
Vanderbilt Campus Dining Website (https://netnutrition.cbord.com/nn-prod/vucampusdining)	$0.00	1	$0.00
Apple Developer Account (2 Year Subscription)	$198.00	1	$198.00
Google Developer Account	$25.00	1	$25.00
Total			$223.00

Table 1: Materials Cost

All of the development tools we are using do not cost anything. Specifically, Figma, React Native, Visual Studio Code, XCode, Android SDK, and all the other related technical tooling do not cost anything as they are open source frameworks that can be found online for developers to use at their own discretion. The links of such open source tooling can be found in the table above. However, the deployment tools like the Apple Developer account and Google Developer account do have costs tied to them. The Apple Developer account is an annual subscription that allows developers to publish to the Apple App Store for iOS devices. It costs $99 per year for one account, and we are purchasing a two-year subscription for the initial deployment as well as future application updates. So, the total cost for the Apple Developer account will be $198. On the other hand, the Google Developer account is a one-time fee payment that allows developers to publish to the Google Play Store for Android devices. We will be using one developer account for Android deployment. So, the total cost for the Google Developer account will be $25. Therefore, the total cost of the project will be $223.

The purpose of each requested tooling can be found in the next section, **Prototypical and Technical Specification**. We have also listed Notion and Slack. These toolings are how our team internally will communicate with each other in order to make sure there is full transparency and all tasks are completed on-time. Furthermore, we plan on communicating with our Professor, Dr. Pence, with E-Mail exchanges.

Planning

In this project, per the specification, we are are planning to have exactly 5 weeks to complete the project — additionally, we imagine as though we started the project a month ago. As such, the start date for our project is November 4, 2021 with a final due date of December 9, 2021. Since this is a technical developmental project, with discrete tasks with continuous feedback and development, we plan on using an Agile methodology, defined by sprints. In this sense, we would like to divide our project into 3 main sprints. The first two sprints will last 2 weeks long each, and the last sprint will only be a week long. Each sprint will incorporate two to three buffer days for about one week of a buffer total. This buffer will allow team members to mitigate risks in completing stories and any possible story rollovers from one sprint to another.

Below, one can see a Kanban board, which organizes the salient tasks of each sprint into a workable story. We plan on further pointing the stories via difficulty through the fibonacci scale during the actual project timeline, via initial sprint meetings. We will also have a sprint conclusion meeting in which we discuss our burndown, progress, what to push back, as well as any feedback we have on each group member. A full Kanban board can be found on our GitHub repository, as in the following link: https://github.com/aadarshjha/wasabi/projects/1. A screenshot of our Kanban board is visualized below in Figure 3.

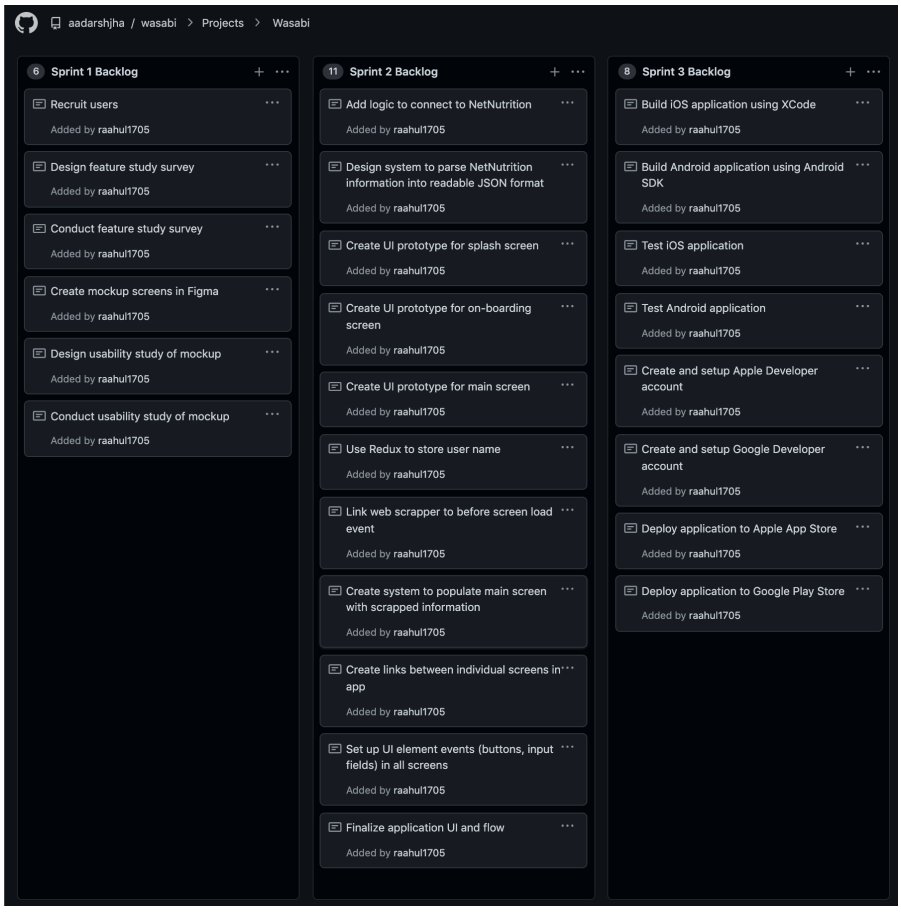


Figure 3 : Sprint Kanban Board

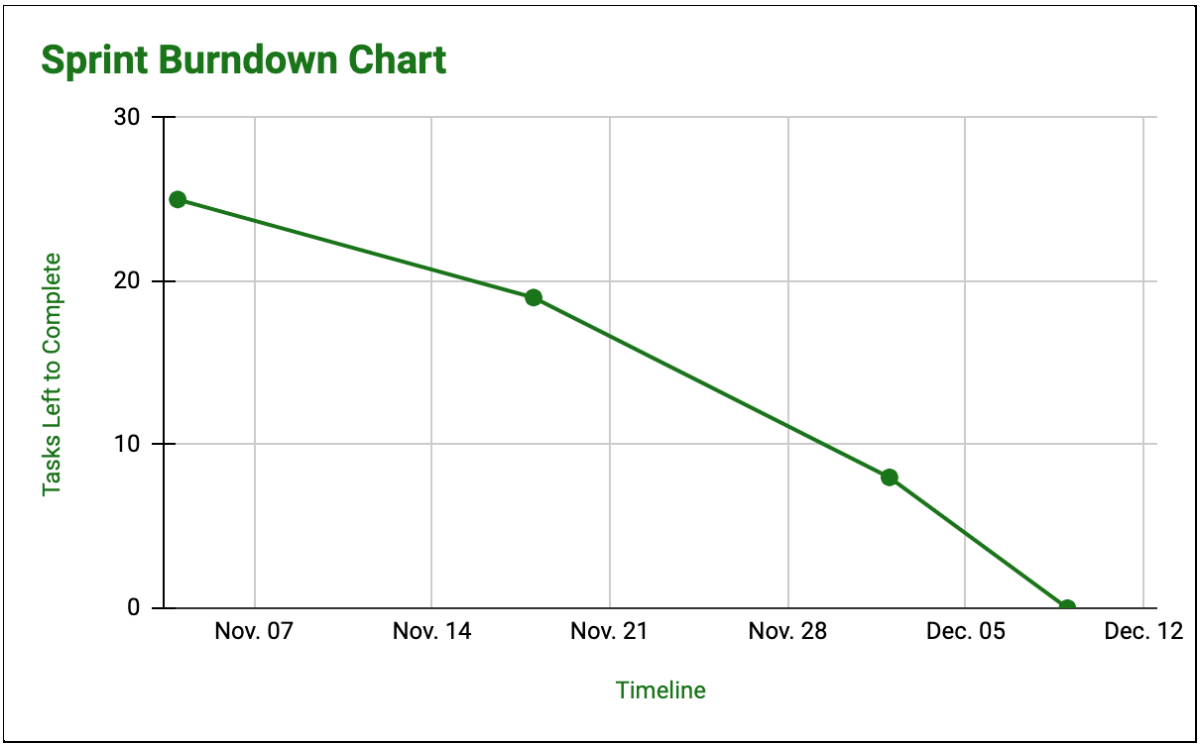


Figure 4 : Sprint Burndown Chart

In **Sprint 1**, we will emphasize feature and usability studies, as well as surveys, creating prototypes and mockups, and recruiting users for our application. Izzy will primarily drive this sprint as the Scrum Master and Product Owner, making sure we are able to deliver a proper study, survey, and understanding of the type of product we are going to build (in terms of UI/UX, and usability). Then, she will proceed to create mockups and prototypes of the application, as well as recruit users for continuous feedback. The termination of her works leads directly into Sprint 2.

Within **Sprint 2**, the emphasis of the work revolves around finalizing the application design and UI based on the user feedback. Both Raahul and Aadarsh will work concurrently during this sprint, making sure to pick up stories as their work has no dependencies upon each other. In particular, Raahul will focus on adding the logic to scrape the NetNutrition Vanderbilt menu website, as well as designing an architecture to scrape and parse this online information. In the same vein, Aadarsh will use technologies such as React Native and Redux in order to create a functional application, that collects user information, and has a scaffolding and design translated from Izzy's initial market research and prototypes. Then, at the end of the sprint, Raahul and Aadarsh will pair programs in order to integrate both the web scraping logic and the application together, by linking the web scraper to the application itself, and feeding the data through pipe and flow logic. Raahul and Aadarsh will serve as the Scrum Masters during this sprint, making sure we are able to complete all necessary tasks. To show a minimum viable product and a proof of concept, we have pasted below a portion of our source code. In particular, here we provide a sample of the scaping code that will fetch all the dining halls open

on campus. In particular, the reason such a sample is provided is because the below process is replicated in all other filtering and fetching calls in the remainder of the application.

```
const request = require('request');
const cheerio = require('cheerio');

// Request Header for the request
// Must pass user information.
var options = {
   url: 'https://netnutrition.cbord.com/nn-prod/vucampusdining',
   jar: true
};

// Function that scrapes all the dining options on-campus
request(options, function (error, response, body) {
   if (!error && response.statusCode == 200) {
       const $ = cheerio.load(body);
       let diningOptions = $('.cbo_nn_unitNameLink')
       let diningOptionsText = diningOptions.map(function(i, el) {
           return $(this).text();
       }).get();
       return diningOptionsText;
   }
});
```

Figure 5 : Sample Code Snippet

Within **Sprint 3**, we need only a week, as this is the phase in which we will export our singular code base into two separate builds — iOS and Android. After exporting this build, we plan on letting Raahul lead us through stories of testing the application and making sure that we have a deliverable product. After which, Aadarsh will set up the Apple and Google developer accounts and publish such products to the respective applications stores.

Finishing all sprints concludes our work for our project, and realizes the deliverable application. As mentioned earlier, we will have two main meetings. One meeting to initiate the sprint, and then another to conclude the sprint. Furthermore, we will have several such meetings within each sprint, meant as informal times in which we can work, collaborate, and ensure that we are accessible to each other for help.

Responsibility Division

Herein, we describe the responsibility division that will guide the three workers (Izzy, Aadarsh, Raahul) in completing the project by dividing the technical, prototypical, as well as product-facing tasks according to particular strengths and weaknesses demonstrated by each group member. In particular, Aadarsh will serve as the Project Manager of the group, ensuring that we are on-track and on the same page with respect to the task management and as well as making sure that everyone has the sufficient resources, support, and motivation to complete the

project. In such a way, Aadarsh always has the “A” and “N” for each task in the Responsibility Matrix, as he should always be an approver and notified of any changes in the project. Then, enumerating through each task, it is clear that each member has differential responsibility. We partition our responsibilities into three main categories:

- **Izzy H.**: UX/UI Design, Product Owner
- **Raahul N.**: Web Scraping Architecture And System Design, Application Testing
- **Aadarsh J.**: Mobile Application Development, Application Deployment

Responsibility Matrix			
Task	**Aadarsh**	**Izzy**	**Raahul**
User Research and Initial Feature Study	S A N	P	N
Mockup Screens Of Application In Figma	S A N	P	N S
Small Usability Study Of Initial Application Draft	A N	P	N S
Creation Of The Web Scraping Logic	A N S	N	P
Translating The Prototypes Into Functional Designs	P A	N S	N
Integration Of Web Scraping Logic Into Base Application	A N S	N	P
Finalizing UI and Design Of Cross-Platform Application	P A N	S	N S
Build And Test iOS Application	A N S	N	P
Build And Test Android Application	A N S	N	P
Deploy Application To Apple Developer Store	P A N	S	N S
Deploy Application To Google Play Store	P A N	S	N S

Table 2: Responsibility Matrix

Izzy has strengths in design and user experience and management. As such, Izzy will be the Primary owner of tasks with respect to the User Research and Initial Feature Studies, Mockup Screen of Application in Figma (https://www.figma.com/) , and conducting the Small Usability Study of Initial Application Draft. Raahul and Aadarsh both serve as secondaries in some of these tasks as they are the main developers of the project on the technical side, so they naturally have some limited experience in user outreach and product mockups.

Next, Raahul has primary ownership over tasks related to Web Scraping and fetching live, real-time data from the CBORD Net Nutrition website (https://netnutrition.cbord.com/nn-prod/vucampusdining). In particular, Raahul will be in charge of three main tasks as well. These include: (1) creation of the web scraping logic and architecture; (2) integration of the web scraping logic into the base application; (3) as well as building and testing the iOS and Android Application. Due to Raahul’s technical strength in JavaScript as well as his familiarity with GET, POST, PUT, and DELETE requests following the HTTP informational flow architecture, it makes most sense to allow him to lead these tasks.

Additionally, Raahul has experience building and publishing web applications to the real-world, so his skills in application testing should be useful in making sure we have an end-to-end product that is fully-functional and deliverable to the end user. Aadarsh serves as the secondary to Raahul since Aadarsh's technical skill set complements Raahul's skills quite well.

Finally, Aadarsh has primary ownership with tasks related to mobile application development and deployment. In particular, Aadarsh will use the free and open source (FOSS) library called React Native (https://reactnative.dev/) to build a cross-platform (both iOS and Android) application with a singular codebase. In doing so, Aadarsh is responsible for the following tasks: (1) translating the built prototypes into functional designs and code; (2) finalizing the UI and design of the cross-platform application based on the user feedback; (3) and deploying the application to the Google Play Store and the App Store. Aadarsh has extensive experience in JavaScript and TypeScript, as well as using front-end design libraries that are supported by React Native. As such, Aadarsh is placed as a primary on these tasks, and Raahul is placed as a secondary due to their complementing technical skillset, with the exception of the task of translating the prototypes to code, where Izzy is placed as the secondary, due to her strength in design.

In all, differential tasks are partitioned into P, A, N, S based on the role, strengths, and abilities of each group member in order to allow for concurrency, optimization, and a streamlined project timeline.

Risks And Contingency

Risk	Likelihood	Impact	Consequence	Mitigation
Learning JavaScript and React takes longer than expected	0.01	0.2	0.002	Create a training session for workers to learn JS and React together
Create very buggy code during development	0.05	0.3	0.015	Use AGILE methodology and pair-programming to mitigate risk of bugs
Need more time learning how to prototype specific and complex figures in Figma and Adobe XD	0.05	0.3	0.015	Go through a Figma and Adobe XD workshop to learn the essentials
NetNutrition changes API or webpage structure	0.0001	0.5	0.00005	Regularly review NetNutrition for changes and rebuild code accordingly. Make code flexible
NetNutrition blocks	0.0001	0.5	0.00005	Add session cookie storage to app

public GET requests from users				
Workers get sick due to COVID-19 Pandemic	0.02	0.3	0.006	Keep workspaces socially distanced and require masks
React Native vulnerabilities causes critical security issues in app	0.01	0.6	0.006	Use the latest version of React Native and set up NPM to review and update critical packages
App design is not user-friendly and we need to rework functionality to meet user requirements	0.001	0.3	0.0003	Use an incremental design-review feedback system for quicker design iterations
Developers fail to correctly use version control leading to corrupted or lost code	0.001	0.2	0.0002	Educate all developers on how to use Git version control, create a Git cheatsheet
Total			0.0446	

Table 3: Risk & Contingency Matrix

Project Costs	
Type	**Cost**
Materials	$223.00
Contingency Free Based On Risks	$9.95
Total	$232.95

Table 4: Updated Costs with Contingency

To ensure that the project runs smoothly, we identified possible risks in the project. Specifically, we first estimated the likelihood and impact of each possible risk based on our intuition. Then, we calculated the risk consequence for each risk by multiplying the likelihood by the impact. The above risk matrix shows the results of our calculations.

Based on the calculations, we find that our total risk consequence is 0.0446. So, the risk for our project is very mild. However, we still mitigate these risks so that our project proceeds without problems. To mitigate the risk of our developers taking longer than expected to learn JavaScript (JS) and React, we propose a training session for developers that will allow them to learn the fundamentals of JS and React that they will need for the project. This session will be conducted before beginning the application development process. As for the risk of creating very buggy code during development, we mitigate this risk by using the AGILE methodology and pair-programming to keep the developer's workload manageable and to give them the

opportunity to think methodically about the code while programming. We mitigate the risk of needing more time to learn how to prototype specific and complex figures in Figma and Adobe XD by participating in a Figma and Adobe XD workshop to learn the essentials of UI/UX. For the risk of NetNutrition changing its API or web page structure, we will mitigate this risk by regularly reviewing NetNutrition, rebuilding the codebase to match the updated NetNutrition, and making the codebase flexible so that the NetNutrition changes will not have a huge impact on project. To mitigate the risk of NetNutrition blocking public GET requests from our application, we will implement a session cookie storage mechanism in our app to avoid any blocked GET requests. For the risk of workers getting sick, we mitigate that risk by keeping workspaces socially distanced and requiring masks to be worn at all times. Additionally, to mitigate the risk of vulnerabilities in React Native causing critical security issues in our app, we will use the latest version of React Native and set up the NPM package manager to review and update critical packages. In this project, we also run the risk of designing a UI that is not user friendly. This setback will cause us to rework the app functionality to meet user requirements. To mitigate this risk, we will use an incremental design review feedback system that will help us iterate through design ideas very quickly and minimize wasted resources. Finally, as this project involves collaborative coding, we run the risk of developers failing to correctly use version control. This situation will lead to corrupted and lost code in the project. To mitigate this risk, we will educate all developers on how to use Git version control. In addition, we will provide all developers with a Git cheat sheet to help them with version control.

To account for these risks, we computed a value of $9.95 based on the total consequence, which increases our total net costs of the project to $232.95.

Anticipated Problems

As with most group projects, we anticipate having problems with group dynamics. Poor communication could result in misunderstandings and misinterpretations. For example, if Izzy does not effectively communicate the components of the design, Aadarsh and Raahul as developers may misinterpret the design and produce an incorrect app. To mitigate and resolve this anticipated problem, our team will focus on asking clarifying questions rather than assuming. We will also leverage our sprint meetings for full communication and make sure that we are able to be self-organized and have great group dynamics.

Another possible problem with group dynamics could be personal issues. This might occur when personalities clash. In instances like these, group members should respect and acknowledge each others' ideas, as well as make sure there is an open communication line to resolve issues.

Overall, while there are some anticipated issues that will arise, we believe with the enumerated mitigation strategies that we should be able to overcome these natural roadblocks.

Future Work

The MVP of Wasabi includes a fully working, cross-platform mobile application that allows users to view food offerings on Vanderbilt's location depending on location, date, and meal period. In future versions of Wasabi, following usability testing and feedback, we would implement personalization preferences, integration with real-time dining updates, and expansion to other university campuses. Following are expansions of these features, including an analysis based on their complexity of implementation and return on investment (ROI).

Personalization preferences would allow users to log in to Wasabi to set and store dietary preferences and other key personalization information. For example, if a user has a favorite dish served on campus, they may set a notification system for Wasabi to ping their mobile device when that dish is being served somewhere on campus. Adding this feature in a future version of Wasabi would require a more detailed look into CBORD's NetNutrition API, as well as using messaging-based notification API, such as Twilio (https://www.twilio.com/). We estimate this would require an additional 1 week as well as increase our budget by $2 per month, according to the Twilio API. Based on these details, personalization preferences have a low complexity with a high ROI.

Integrating Wasabi with real-time dining updates would allow users to feel confident that the information they are viewing in the app is as accurate as possible. Vanderbilt dining often announces updates via a mobile text alert or on social media. Wasabi could provide an announcements section where all dining announcements are collated in one location. Adding this feature in a future version would require adding a listener to the newstream of Vanderbilt Dining information. We expect this would cost an additional two weeks in order to set up the infrastructure to have a Publisher-Subscriber methodology to listen on Vanderbilt Events and Dining news, however we expect no changes in cost as this is all using FOSS tooling. Based on these details, real-time dining update integration is moderate complexity with a low ROI.

Expanding Wasabi to other university campuses, or even company campuses, would generalize the app to function beyond Vanderbilt. This would open up the app to a larger target user group. Scaling up Wasabi would require higher compute resources, scalable deployment strategies, and cloud-based architectures, including, but not limited to, Vagrant (https://www.vagrantup.com/), Ansible (https://www.ansible.com/), Docker (https://www.docker.com/), Kubernetes (https://kubernetes.io/), and AWS (https://aws.amazon.com/). All of these, with the exception of AWS, is a free tooling, which would net a cost of $120. We expect this would cost an additional 2 months of time, due the high complexity in being able to scale applications in the cloud, and distribute workloads in an efficient manner as more and more users sign up for our application. Based on these implementation details, expansion to other university campuses has a high complexity with a high ROI.

Based on the complexity and ROI analyses of the three features outlined above, we believe the highest priority would be personalization preferences, followed by expansion to other universities and real-time updates. All features would require additional user testing.