Making Recipes Readable Again

DESIGN DOCUMENT

Team Number: 12
Client: Mat Wymore
Advisers: Mat Wymore
Team Leader: Jack Paton
Meeting Scribe: Bret Knous
Design Facilitator: Vismay Gehlot
Test Facilitator: Luke Knous
Report Facilitator: Rithwik Gokhale
E-mail: sddec21-12@iastate.edu
Website: https://sddec21-12.sd.ece.iastate.edu

Revised: 4/20/2021

Development Standards & Practices Used

There are two standards that have been considered for this project; engineering development standards and engineering coding standards. We shall be following the Agile method using two week sprints for the engineering development standards and we will be following the coding standards outlined by the below link. In addition, we shall be adhering to the COVID-19 safety guidelines of Iowa State University.

Coding Standards:
https://www.sos.state.co.us/pubs/elections/VotingSystems/DVS-DemocracySuite511/documentation/SD-DVSJavaScriptCodingStandards-5-11-CO.pdf

Summary of Requirements

Requirements:

- Must work on a typical mobile device/smartphone
- May be platform specific (e.g. Android), though cross-platform is ideal
- Interface must be user-friendly (typical user is expected to be viewing device from roughly one meter away and using one non-primary finger to navigate)
- Capable of loading/upgrading a recipe from an arbitrary URL
- "Upgraded" recipes should load in 5s or less
- Solution should not interfere with the source's revenue stream (e.g. ads should still display)
- Should not require a user account (may be optional if it would help with desired features)
- Should not require backend infrastructure

Applicable Courses from Iowa State University Curriculum

1. COM S 309 - Software Development Practices
2. COM S 363 - Introduction to database management
3. COM S 227 - Introduction to Object oriented programming
4. COM S 228 - Data Structures
5. COM S 319 - Construction of User Interfaces
6. CPR E 388 - Android development
7. COM S 311 - Introduction to Algorithms

New Skills/Knowledge acquired that was not taught in courses

1. Javascript
2. React Development
3. Jest Javascript testing frameworks
4. Project Management and team-work through remote interactions
5. Maintaining COVID-19 health standards

Table of Contents

List of figures/tables/symbols/definitions

1 Introduction

1.1. Acknowledgement

Our client and advisor, Mat Wymore, is acknowledged for their contributions towards the design of this project and remaining available to provide assistance as needed.

1.2. Problem and Project Statement

Online recipe blogs and websites are often confusing to understand and take a long time to navigate. This project aims to reformat and modify what the user sees to make the recipes more comprehensible and easier to navigate.

Our project plans to create a mobile app that makes these changes to a website based on a provided URL. We want to accomplish this by creating a web scraper that will take all the necessary details from the website and reorganize them into a clear and concise manner that will allow users to easily identify ingredients and follow instructions for their desired recipes.

We hope that by using this app users will be able to easily read and understand recipes and make the cooking experience more enjoyable. In addition, we hope that recipes that once might have been difficult to understand can become more straightforward, allowing more users to utilize them.

1.3. Operational Environment

Our project is intended for online use only and will not be expected to function without access to the internet. Because of this, users may have different experiences with the application depending on their GPS signals. The constraint of needing the internet, and a decent internet connection, may have an effect on the location our application is used and limit some potential users. In terms of environmental constraints, this app is expected to work in the context of an active kitchen, so that will impact the UI. It should not require the user to touch it too much and it should be easy to read. It should also be able to easily flow into different sections of the recipe with minor interaction from the user.

1.4. Requirements

Requirements:

- Must work on a typical mobile device/smartphone
- May be platform specific (e.g. Android), though cross-platform is ideal

- Interface must be user-friendly (typical user is expected to be viewing device from roughly one meter away and using one non-primary finger to navigate)
- Capable of loading/upgrading a recipe from an arbitrary URL
- "Upgraded" recipes should load in 5s or less
- Solution should not interfere with the source's revenue stream (e.g. ads should still display)
- Should not require a user account (may be optional if it would help with desired features)
- Should not require backend infrastructure

Features:

- Hyperlink to images of different ingredients for visual assistance
- Easy unit conversion for the different quantities of ingredients
- Relevant substitutes/alternatives for specific ingredients in select recipes
- Hyperlinks to online/nearby stores where rare/specific ingredients can be purchased
- Social and online sharing abilities for the recipes enhanced through the app
- Ability to bookmark and save the recipes which have been enhanced by the app for easy access in the future
- Ability to scale recipes, as in multiply or divide ingredients proportionally.
- Listing quantities of the different ingredients in the instructions part of the webpage

1.5. Intended Users and Uses

Based on the requirements which have been described above, this product will have a wide range of end users. The primary objective of the product is to provide enhanced recipes to individuals who would benefit from any of the features which have been listed above. Therefore, it is safe to conclude that a large number of the end users for this app will be individuals who are just getting into culinary practices or beginner cook, However, due to the availability of specific features such as providing the option to convert units or access visual representation of certain ingredients as well as the ability to make this app compatible with virtually any online recipe, individuals who are experienced with cooking may also be able to benefit from this app

1.6. Assumptions and Limitations

Assumptions:

1) The end product can be used globally with no restriction
2) The make and model of a preferred device will not affect app compatibility
3) There will be no limitations with access to the required software and APIs during development

Note that these are all the assumptions which can be made before the beginning of the development phase. Additional points will be added as and when potential roadblocks are faced during the actual development of the application.

Limitations

1) This app will require constant internet connection since the websites are 'enhanced in real time'
2) Specific features in the app will also require constant GPS data to provide shopping information for the specialized ingredients
3) A new instance of the app will be loaded at each use due to the lack of external database (one of the functional requirements put forward by the client)
4) The end product will only work on mobile devices. Thus a web version of the product will not be available.

5) The application will only work on english websites

As stated above in the assumptions section, these are all the limitations which can be determined at this stage of the project. Additional technical limitations or modifications to current limitations may be applicable once the team begins coding.

1.7. Expected End Product and Deliverables

Project Deliverables

1) The final product which will be delivered to the client is a cross platform mobile application which meets all the features and functional requirements which have been listed in the above sections
2) Design and product mockups will be delivered to the client on a bi-weekly basis once the development has begun
3) While the application will be designed for maximum ease of maneuverability by the end user, an onboarding/user guide can be created and delivered with the end product if required by the client. This project is yet to be finalised after further discussions with the client.

2. Project Plan

2.1 Task Decomposition

1. Planning
    a. Framework Selection - Our application needs to work on both Android and IOS, so we will be using a framework to determine what will work best for us. The decision was between Flutter and React Native. We have selected React Native for our purposes, due its larger support and more established nature.

    b. Feature Selection - The last part of our planning is picking which features are most important and most feasible to create. The decided on features are described below in the Feature Functionality section.

    c. UI Design - To build a usable application interface we need a user focused design. We will plan our design around the needs of a non-technical user, using the application on their cellular device while cooking.

2. Testing
    a. Testing Framework Selection - As we are developing in an agile development style, we need to pick a testing framework for our app so we can write tests before coding the app itself. We have chosen the Jest React testing framework.

    b. Unit Test Creation - Using the Jest framework, we will need to create unit tests for our application and ensure full coverage of our code to ensure completeness.

    c. User Testing - After the app is fully functional tests will be conducted by having possible users test the app and give feedback so we can make improvements where it is needed.

3. UI
    a. Interface Functionality - Getting our interface up and running will be one of our first priorities since it is the glue that holds everything together and will be used for user testing.

    b. Styling - After functionality is completed and tested we will make the application more responsive and presentable to users by styling the application with a combination of CSS stylesheet containers and React props.

4. Feature Functionality
    a. Web Scraping - To display recipe website data correctly, we utilize web scraping libraries such as jsonld.js to pull recipe data and display it to our users.

    b. Formatting Data - After the data is gathered from a users chosen website we need to display it properly in a more readable format. We will code the application to cut away extraneous information on the page such as advertisements and blog content. We will also insert ingredient amounts into the recipe direction section.

    c. Data Modifying Features - There are several data modifying features we will be adding such as an ingredient amount converter which will allow users to convert units on the recipe from imperial to metric. Adding hyperlinks for ingredient shopping and others. These can all be worked on concurrently as they do not depend on each other.

    d. Toggle Switch - To give users a choice in what they see, we will implement a toggle switch to allow users to see the previously cut blog content if they wish to see it.

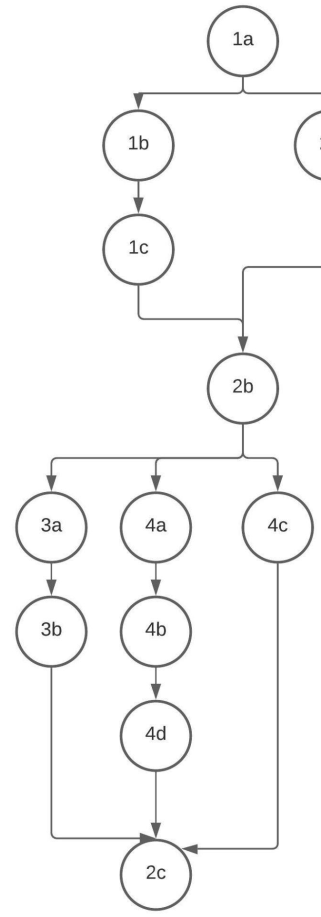


figure 2.1 - Dependencies

2.2 Risks And Risk Management/Mitigation

Planning - While we might have missed something in our planning process it will most likely be able to be fixed at a later time. Rating: 0.1

Testing - It is possible that bugs and other issues may escape our testing when our application is used on a wide variety of websites. 0.7

> Mitigation Plan: The best way for us to deal with this is to test our application on as many different websites as possible, making sure it is tested on the largest and most commonly used websites. As long as it works on the websites that get the majority of

the traffic it will not be as big of an issue if it doesn't work well on small obscure websites that few people use.

UI - The risks here are that the app might not be suitable for use in the kitchen or to people with disabilities like poor eyesight or color blindness. Rating: 0.5

Mitigation Plan: We will do what we can to make the app as readable as possible by using colorblind safe colors and making the font as large and readable as possible.

Feature Functionality - While it is possible for some of the libraries we use to depreciate otherwise have issues, this is unlikely and as long as our application is coded well the risks here should be minimized. Rating: 0.3

URL Safety - It is possible that a URL is not a valid URL, could redirect the user to an unintended site, or contain hidden JavaScript. Our project is a local application and our assumption is the user knows what is on the site they are trying to scrape. Therefore, this should not be a large issue. Rating: 0.3

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Milestone 1 : All unit tests are created

- Test cases are created and work as intended for the application's ability to properly pull recipe data from a given url and format it correctly.
- Test cases are created for the applications secondary functions as well.

Milestone 2: Main functionality completed

- The application's ability to retrieve and reformat data is implemented and can be tested using the already existing unit tests.

Milestone 3: Application is functional on major recipe websites

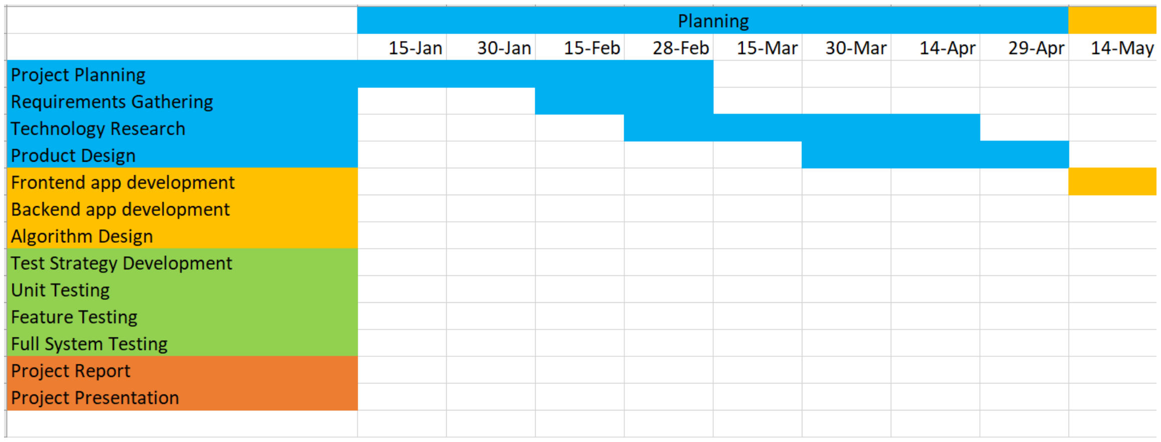
- The application is able to perform its main functionality on major recipe websites with 90% success rate.

Milestone 4: Full Functionality

- All features are implemented
- All unit tests are passed
- App is fully functional on 90% of recipe websites

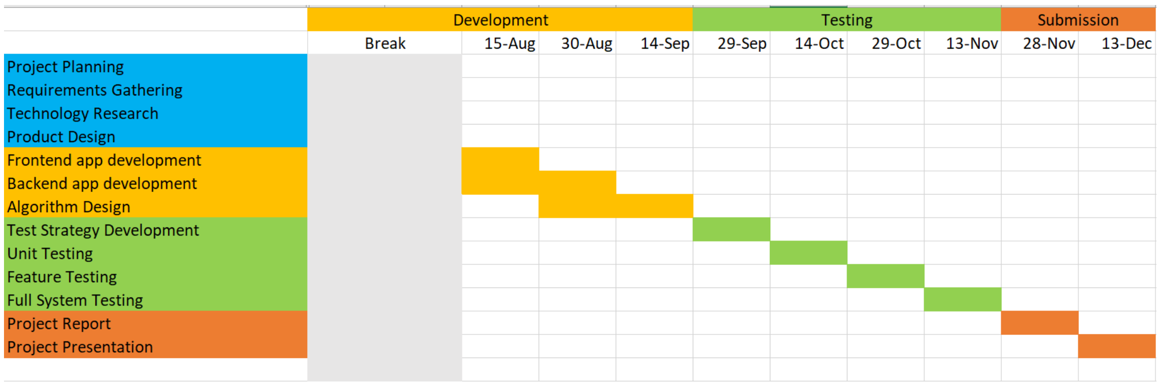
2.4 Project Timeline/Schedule

It is important to have a realistic and detailed project plan to ensure that all the tasks are completed on time and the dependencies/hierarchy of the different project phases are clearly defined. The project plan has been developed in the form of a Gantt chart which has been pasted below. Note that the Gantt chart has been divided up into two images for easy readability.



	Planning								
	15-Jan	30-Jan	15-Feb	28-Feb	15-Mar	30-Mar	14-Apr	29-Apr	14-May
Project Planning	🟦	🟦	🟦	🟦					
Requirements Gathering			🟦	🟦					
Technology Research				🟦	🟦	🟦	🟦		
Product Design						🟦	🟦	🟦	
Frontend app development									🟨
Backend app development									
Algorithm Design									
Test Strategy Development									
Unit Testing									
Feature Testing									
Full System Testing									
Project Report									
Project Presentation									

figure 2.2 - Project Plan (Spring)



	Development				Testing				Submission	
	Break	15-Aug	30-Aug	14-Sep	29-Sep	14-Oct	29-Oct	13-Nov	28-Nov	13-Dec
Project Planning	▨									
Requirements Gathering	▨									
Technology Research	▨									
Product Design	▨									
Frontend app development	▨	🟨								
Backend app development	▨	🟨	🟨							
Algorithm Design	▨		🟨	🟨						
Test Strategy Development	▨				🟩					
Unit Testing	▨					🟩				
Feature Testing	▨						🟩			
Full System Testing	▨							🟩		
Project Report	▨								🟧	
Project Presentation	▨									🟧

figure 2.3 - Project Plan (Fall)

As it can be seen from the above images, a clear project plan has been developed by the team which accounts for the different project phases and the major milestones which need to be met before the completion of a specific phase of the project. The main project itself has been divided into 4 phases; planning, development, testing and submission. Furthermore, each of these project phases have smaller milestones and deliverables associated with them. Since the different phases have been clearly color coded, it is also easy to identify the different project dependencies. For example, in fig. 2.2, it is indicated that unit testing cannot take place until test strategy has been defined.

As explained further below in the document, initial experimental development work will begin at the end of this semester and throughout the summer break. As indicated in the project plan above, this will allow the project team to familiarize themselves with the coding environment and gain the necessary technical knowledge. All major modules will be completed by September 14th so that testing can begin. The project plan lists that all testing work is completed by November 14th so that the last two weeks can be dedicated to submitting the app and completing any associated documentation.

Note that the project will be managed using an Agile form of software management but this will be only during the development and testing phase. Due to the number of dependencies between the different phases, the overall project will follow a waterfall approach of software project management. This will allow the team to adequately plan before beginning any implementation or testing. Therefore the Gantt chart clearly shows the waterfall approach taken in this project. Within the 2 month development period, we will employ an Agile form of management with 2 weeks sprints and an MVP due at the end of each cycle.

2.5 Project Tracking Procedures

As detailed below in the development processes, we will be following an Agile form of software project management with 2 weeks sprints and a MVP due at the end of each cycle. The work will be broken down into tasks which will be selected for sprints from the backlog. The following technologies and project tracking processes will be adopted for our project:

1) Discord - This will be our IM channel for the project. Since all the team members are more used to the environment and features of Discord, we will be using this as our primary mode of communication with the team and the client (as an alternative to Slack).
2) Trello - Trello will be mainly used to manage the 2 weeks sprints during development. Tasks (tickets) will be assigned to trello boards for each project team member which have to be regularly updated and completed for the MVP to be presented at the end of the two week sprints.
3) Github - Github will be used as our primary project code repository. This is a code management software which will allow multiple developers to collaborate on the project simultaneously. Github will also provide the team and the client to keep track of code commits made by different team members to get a better understanding of the overall project progress.

2.6 Personnel Effort Requirements

Task	Time Required
Framework Selection	4 hrs
Feature Selection	10 hrs
UI Design	10 hrs
Testing Framework Selection	10 hrs
Unit Test Creation	60 hrs
User Testing	20 hrs
Interface Functionality	40 hrs
Styling	40 hrs
Web Scraping	80 hrs
Formatting Data	80 hrs
Data Modifying Features	50 hrs
Toggle Switch	10 hrs
Total Time	414 Hours

figure 2.4 - Time Allocation

As seen in the table above the early stages of the project, namly the planning tasks, will add up to about 44 hours. The testing of the application is estimated to take 80 hours, and the brunt of the development will take 290 hours if done consecutively. However some of the tasks in the development stages can be split up amongst the team members and accomplished concurrently so it will actually take less than 290 hours in practice.

2.7 Other Resource Requirements

Since our main project deliverable is a mobile application, there are minimal external resource requirements for this project to be successfully completed. Since all the development work will be completed through computers, the only additional hardware required for this project would be smartphones to test the end product on. While coding the different modules of the app, testing will take place through Android and iOS emulators. But to ensure thorough system testing, the app will be loaded on an Android and iOS smartphone after completing development to get a proper idea of the app's user experience.

2.8 Financial Requirements

There are no known financial requirements for this project at this stage. Since the project team members own Android/iOS smartphones, final system testing can be completed through personal smartphones. If there are any challenges with phone availability or security concerns, we will be using the test phones available through the Computer Science and Engineering department. These phones can be loaned for project work through an established program offered by the University itself.

3 Design

3.1 Previous Work And Literature

Detailed research was conducted by the team members during the design phase of the project specifically into pre-existing solutions/products in the market which work with the same topic as our project. While we were not able to find any applications which provide the same service as our product, there are a number of applications in the market which focus on improving the culinary experience with regards to finding and saving recipes from across the internet.

Therefore, while addressing the previous works and literature for this project, a direct comparison will not be made with another app which already exists in the market. Alternatively, the functionality, the strengths and weaknesses of the different products will be analyzed so that appropriate design decisions can be made during the implementation of our application.

1) Paprika

The main objective of this app is to provide users with the opportunity to save recipes from different web sources in one location for easy access. This app provides users with a built in web browser and the ability to bookmark any favorite recipes which they come across. The main strength of this application is that it provides users with the option to save all of their favorite recipes in one easy to access location. But beyond this main feature, the app does not provide users with any other options to further enhance the recipes which they are viewing. Our proposed solution will provide users the option to save recipes on the app while also providing other features to 'enhance' the recipes which they find on the web.

2) BBC Good Food

The main selling point for this app is the wide variety of recipes available which have been created by industry professionals and reviewed by thousands of users from across the world. This provides the end user of this app with a well curated range of recipes with feedback from other users to improve the overall cooking experience. Furthermore, users are able to bookmark/save their preferred recipes on the phone for easy access in the future. But on the other hand, this app does not specifically offer any other features for the users to customize the recipes to their needs and view only the relevant information. This would be the main difference between BBC Good Food and our proposed application.

3) Tasty

Tasty uses a crowdsource approach to its recipes where users are able to actively comment on the different sections of the recipe and offer suggestions and tweaks which can be adopted by others who view the same recipe in the future. This allows the app to provide a wide range of recipes with essential feedback and reviews provided by other users of the app. While this app focuses on improving the recipes which are currently available on the web, the approach taken by Tasty is very different from our proposed solution which will not offer users the ability to rate and review recipes but provide them with machine generated improvements such as easy unit conversion, improved information layout and relevant visual aid.

3.2. Design Thinking

During the design phase of the project, as a team, we focused on determining the most effective product as the solution for the problem presented by our client. Over multiple brainstorming sessions, we considered different options such as websites plugins extensions. After thorough evaluation, we came to the conclusion that a mobile application would be the best option. This option would provide the service to the largest population of users since smartphones have become a major part of modern day living. Furthermore, smartphones also come with preinstalled settings which are designed to improve the user's app experiences. The next step in the define phase was to brainstorm the different features which could be included in our app to provide the users with an 'enhanced' recipe. During this phase, the team conducted research on different existing apps in the market and discussed different services which can be provided by the app and can be translated into features. And the final step of the define phase was to start coming up with different UI mockups for the app and create a draft of the system design for the application. During this process, mockups for the home screen and the recipe screen were developed by the team to get a better idea of the placement of the different buttons and features on the app screens.

During the ideate phase of the project, our main goal was to finalize the project requirements (the features for our application), discuss the non functional requirements of the project and conduct sufficient technology research so that we do not face many roadblocks during the actual implementation of the app. And the last step of the ideate phase was to finalize the design of the product in terms of system diagrams and UI mockups. This would adequately prepare us to begin development of the application in the next phase of the project.

3.3. Proposed Design

During the design phase of the project, our team had the following objectives to complete:

- Identify all the functional and nonfunctional requirements
- Develop mockups of app UI
- Conduct research on technologies which will be used for this project

Before creating the design document for this project, the team finalized on the requirements (seen in chapter 1). Mockups were also developed for the home page and the recipe page of the mobile application. The objective of these mockups were to provide the team with an idea of how the app will look from a design perspective and to spark conversations between the team members and the client on where the different components would go on each of the screens and the overall structure of the app. The following mockups are for the home screen and the recipe screens:

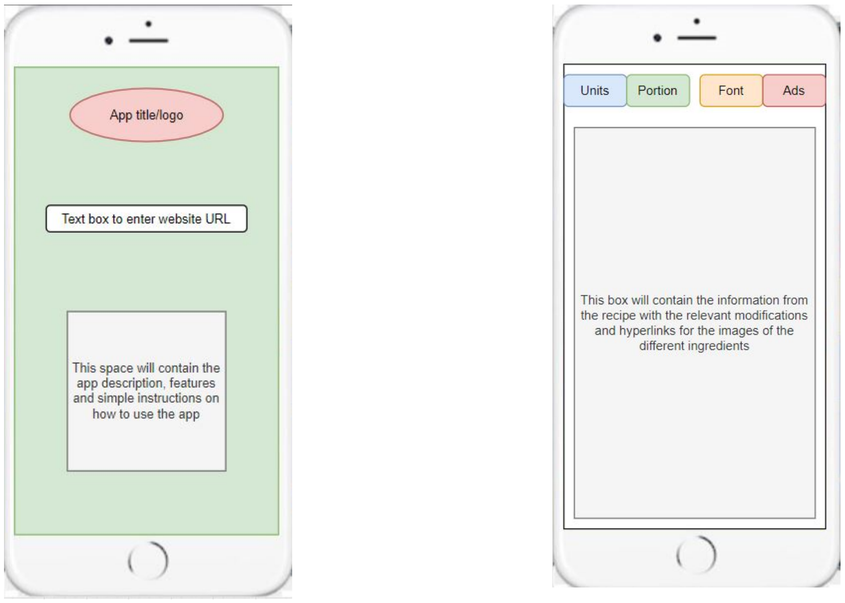

App title/logo
Text box to enter website URL
This space will contain the app description, features and simple instructions on how to use the app
Units
Portion
Font
Ads
This box will contain the information from the recipe with the relevant modifications and hyperlinks for the images of the different ingredients

Image 3.1 - screen mockups

As seen above, the first mockup is of the home screen/URL screen. This will be the first page which opens up when the app is launched. Since our app prioritizes ease of use, the home screen will have very easy to understand structure. This page will contain the title/logo, a text box for the user to enter the recipe website URL and a simple set of user instructions on how to navigate through the app. The second screen mockup is of the actual recipe page itself. This page will contain the enhanced recipe once the app has processed all the information from the original website. A majority of the screen will be taken up to display the recipe information. This screen also contains a number of button components at the top to use the different features provided by our app. These include the option to change units, change portion sizes, enable/disable advertisements and change the font to make it easy to read from long distances. As mentioned earlier, these mockups were designed to meet all the feature requirements of the app which were finalized with our client during the initial brainstorming sessions of this project.

The design proposed above was developed after careful consideration on how they could help meet the functional requirements of this project which were provided by the client. As it can be seen in the above images, this program is designed to run on a standard iOS and Android smartphone. Therefore, it follows the basic design principles of mobile applications. Furthermore, another major requirement listed in chapter 1 is ease of use with regards to the UI. Hence, the different components on the screen have been labeled clearly to ensure that users are able to easily navigate through the app while cooking or in the kitchen.

3.4 Technology Considerations

Since it was determined that we will be developing a cross-platform mobile application which can function on both iOS and Android devices, we believe that using React is the most appropriate option. Developing the app through React ensures a smoother development process and reduces the additional effort of optimizing the app for iOS and Android devices. While independent development for the two OS would provide users with more features and an overall better app experience but this would double the work for the team and increase the overall implementation time since we would have to learn different software and libraries which are native for Android and iOS development. Additional research is currently underway on different development platforms which can be used and APIs and libraries which are available for our use during the implementation of the app.

Testing is going to be a major part of this project. Before launching the application it is important to conduct a number of different tests to ensure that there are no logical errors or UI bugs in the code. Since the app is being developed with React, we will also be using a native Javascript testing framework called Jest. The use of this library will provide us with the option to conduct unit tests and system integration tests on the app and ensure that the overall app experience remains smooth.

3.5 Design Analysis

So far in the project, the proposed design plan is serving the purpose effectively. The team has been able to complete all the tasks on time and the project schedule is being closely followed. One of the small limitations of the proposed design plan is that due to the lack of development work being done in phase, the team is unable to fully prepare for any technical challenges which we might encounter while coding the application. While the current proposed design meets all the functional requirements provided by the client, there may be alterations to the design during project implementation due to any technical limitations the team might face. One potential solution to this problem in the design process could be allowing the team to experiment with partial implementation of different screens or backend systems during the break in order to ensure that everyone is comfortable with the development environment.

3.6. Development Process

We will be using an Agile software development process for the scope of this project. We will have 2 week sprints throughout the development period and each meeting, new tasks will be added from the backlog which will be curated at the beginning of the implementation phase. A minimum value product (MVP) will be presented to the client and the rest of the team at the end of the two week sprints where items from the backlog are completed. This development process will ensure that the client and the team members have a realistic idea of the project progress and therefore will be able to better plan out the next steps which need to be completed in the upcoming sprint cycle.

3.7. Design Plan

The team initially started the design process by creating mockups of the different screens in the app to get a better understanding of the placement of the different components in the app. The next step of the design process would be to develop simple UI mockups through Javascript development and display it to smartphone emulators. This step will provide the team and the client with a realistic idea of how the UI will be implemented for this project and any changes can be made based on the feedback received from the client. Mock up implementation of the databases will also be completed before main project implementation so that we have a better understanding of the backend logic required for this project.

Next semester, the majority of the time will be spent on developing an efficient web scraping algorithm so that only the relevant information is collected from the recipe websites, enhanced and displayed in the app. The team expects this process to take the longest in the overall project since we will have to train models to collect only the most essential data from the recipe websites and then display it in the app in an effective manner. Once this process has been completed and tested with the UI, additional system testing will be done in the app to ensure that there are no bugs within the code.

4 Testing

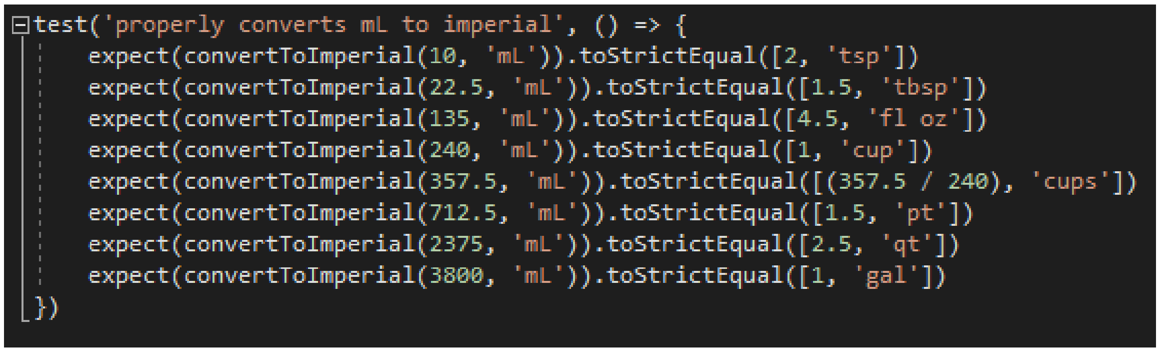
1. Our project will need unit tests for modules, integrity tests for interfaces, acceptance tests for all functional and nonfunctional requirements, black and white box testing for end functionality and performance, and security testing.

2. The following lists items needing to be tested and the types of tests we plan to use.
   a. Units: Unit Conversion for measurements, Retrieving Data from URL, Scaling Measurements, Ensuring URL is safe.

b. Interfaces: Between UI and Retrieving Data from URL, Between Retrieving URL and Ensuring URL is safe, Between UI and Scaling Measurements/Unit Conversions.
c. Acceptance: All the functional and non-functional requirements and any features that have been implemented. This includes, but is not limited to, loading a recipe from an arbitrary URL, solution should not require a user account, and solution should be user-friendly.
d. Black Box: Ensure end functionality is working properly. Such as our functions for retrieving the website data from the URL are doing so.
e. White Box: Ensure the flow of input and output of the code inside our functions is correct and to find improvements in design.
f. Security: URL retrieved should be valid, preventing redirect URLs, and checking for hidden javascript in URLs.

3 & 4. Below is an example of a test case we have defined, designed, and developed. The test below is part of a test suite designed to make sure the function we wrote to convert from units in the metric system to the imperial system is working as intended. This test has the goal to make sure the function converts properly between mL, specifically its abbreviation, to an appropriate unit in the imperial system. These tests were written in JavaScript in Visual Studio and tested using Jest, a native testing framework for JavaScript. The example below has eight different test cases. The first test case demonstrates converting 10 mL to imperial. This is done by passing a value of 10 and a string of mL to our function. Our expectation is that the result will be 2 tsp.

Image 4.1



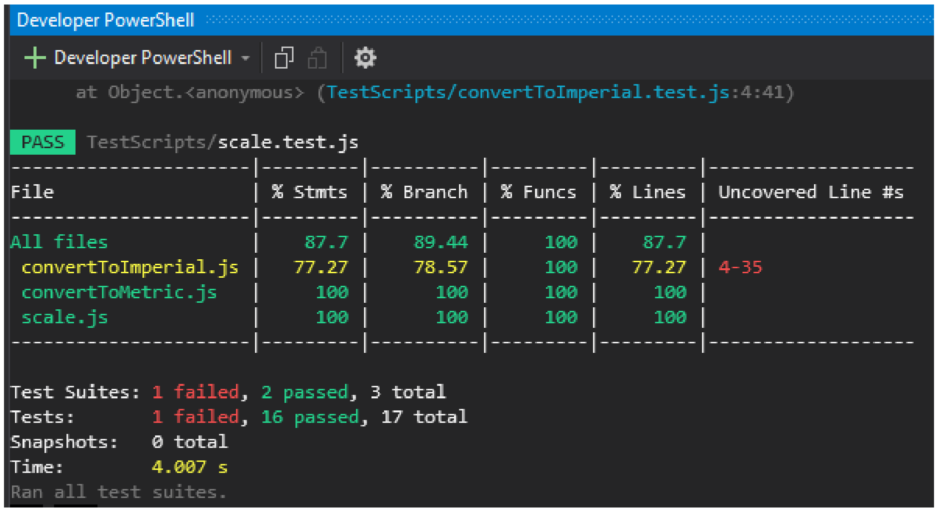
```
test('properly converts mL to imperial', () => {
    expect(convertToImperial(10, 'mL')).toStrictEqual([2, 'tsp'])
    expect(convertToImperial(22.5, 'mL')).toStrictEqual([1.5, 'tbsp'])
    expect(convertToImperial(135, 'mL')).toStrictEqual([4.5, 'fl oz'])
    expect(convertToImperial(240, 'mL')).toStrictEqual([1, 'cup'])
    expect(convertToImperial(357.5, 'mL')).toStrictEqual([(357.5 / 240), 'cups'])
    expect(convertToImperial(712.5, 'mL')).toStrictEqual([1.5, 'pt'])
    expect(convertToImperial(2375, 'mL')).toStrictEqual([2.5, 'qt'])
    expect(convertToImperial(3800, 'mL')).toStrictEqual([1, 'gal'])
})
```

5. We ran the tests in the Jest framework by running our suites. We ran our suites with the --coverage tag to provide more insight to how our functions are behaving, such as what percentage of lines are being reached by our tests.

6. Below are the results we found on our initial run of our test suites. We discovered that one of the tests was failing in the convert to imperial suite, the failed test was part of the

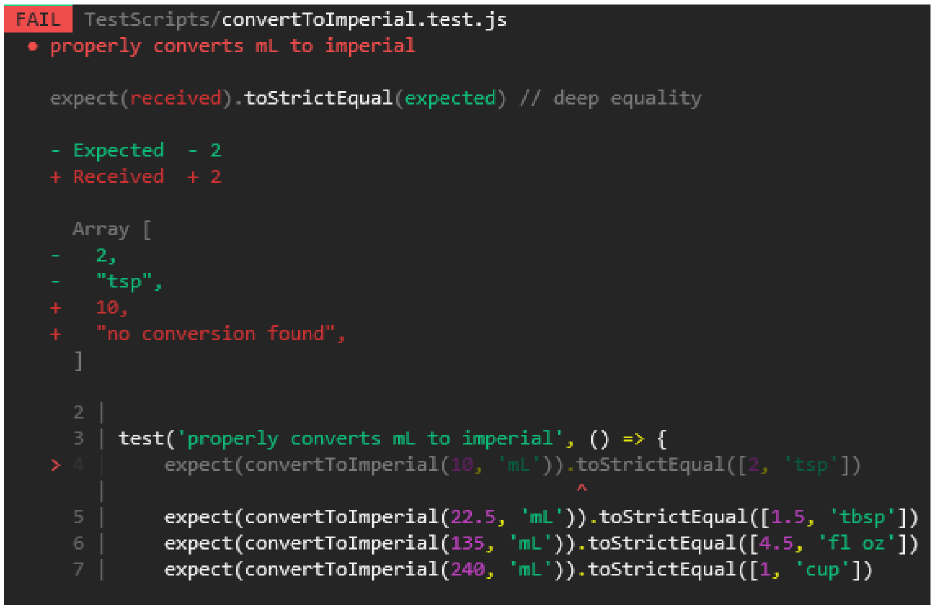
properly converts mL to imperial test that we outlined above, and the convert to imperial suite did not have full statement, branch, and line coverage. Jest provided an html which showed us the specific lines which were never reached by our code by highlighting them red, which was helpful in determining where our problem existed.

Image 4.2



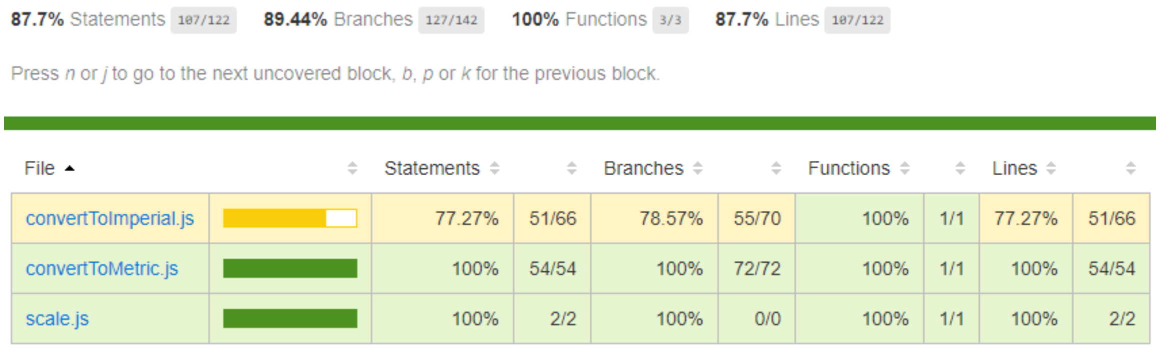
```
      at Object.<anonymous> (TestScripts/convertToImperial.test.js:4:41)

 PASS  TestScripts/scale.test.js
----------------------|---------|----------|---------|---------|-------------------
File                  | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
----------------------|---------|----------|---------|---------|-------------------
All files             |    87.7 |    89.44 |     100 |    87.7 |
 convertToImperial.js |   77.27 |    78.57 |     100 |   77.27 | 4-35
 convertToMetric.js   |     100 |      100 |     100 |     100 |
 scale.js             |     100 |      100 |     100 |     100 |
----------------------|---------|----------|---------|---------|-------------------

Test Suites: 1 failed, 2 passed, 3 total
Tests:       1 failed, 16 passed, 17 total
Snapshots:   0 total
Time:        4.007 s
Ran all test suites.
```

Image 4.3



```
 FAIL  TestScripts/convertToImperial.test.js
  ● properly converts mL to imperial

    expect(received).toStrictEqual(expected) // deep equality

    - Expected  - 2
    + Received  + 2

      Array [
    -   2,
    -   "tsp",
    +   10,
    +   "no conversion found",
      ]

        |
        | test('properly converts mL to imperial', () => {
    >   |     expect(convertToImperial(10, 'mL')).toStrictEqual([2, 'tsp'])
        |                                          ^
        |     expect(convertToImperial(22.5, 'mL')).toStrictEqual([1.5, 'tbsp'])
        |     expect(convertToImperial(135, 'mL')).toStrictEqual([4.5, 'fl oz'])
        |     expect(convertToImperial(240, 'mL')).toStrictEqual([1, 'cup'])
```

Image 4.4



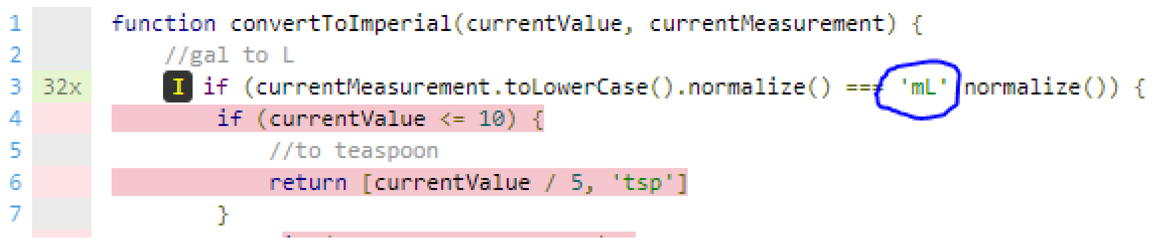
87.7% Statements 107/122 **89.44%** Branches 127/142 **100%** Functions 3/3 **87.7%** Lines 107/122

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

File		Statements		Branches		Functions		Lines	
convertToImperial.js		77.27%	51/66	78.57%	55/70	100%	1/1	77.27%	51/66
convertToMetric.js		100%	54/54	100%	72/72	100%	1/1	100%	54/54
scale.js		100%	2/2	100%	0/0	100%	1/1	100%	2/2

7. After some digging, we found that we were converting the measurement we passed to the function to lower case but trying to compare it to mL with a capital L. To fix this we changed that L to be lower case. The picture below shows where this was in our code.

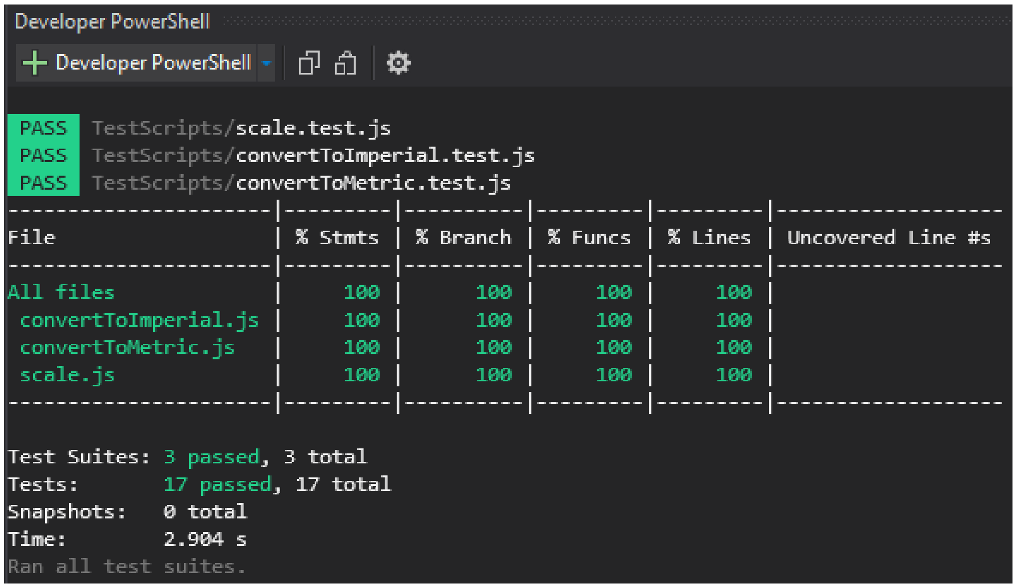
Image 4.5



```
                function convertToImperial(currentValue, currentMeasurement) {
                    //gal to L
        32x      I if (currentMeasurement.toLowerCase().normalize() === 'mL'.normalize()) {
                        if (currentValue <= 10) {
                            //to teaspoon
                            return [currentValue / 5, 'tsp']
                        }
```

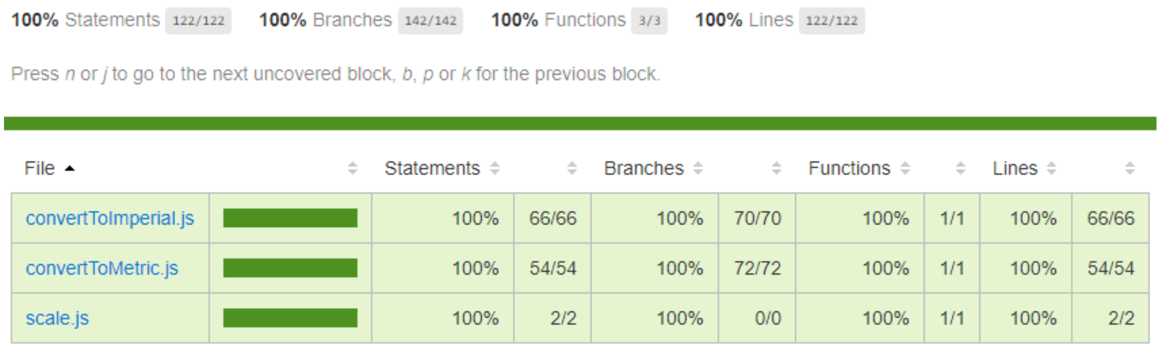
8. After making the changes and re-running our tests, we found the failed test now passes and we have complete coverage for the statements, branches, and lines. In addition, the html no longer has red lines indicating part of the code was never reached.

Image 4.6



```
PASS  TestScripts/scale.test.js
PASS  TestScripts/convertToImperial.test.js
PASS  TestScripts/convertToMetric.test.js
----------------------|---------|----------|---------|---------|-------------------
File                  | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
----------------------|---------|----------|---------|---------|-------------------
All files             |     100 |      100 |     100 |     100 |
 convertToImperial.js |     100 |      100 |     100 |     100 |
 convertToMetric.js   |     100 |      100 |     100 |     100 |
 scale.js             |     100 |      100 |     100 |     100 |
----------------------|---------|----------|---------|---------|-------------------

Test Suites: 3 passed, 3 total
Tests:       17 passed, 17 total
Snapshots:   0 total
Time:        2.904 s
Ran all test suites.
```

Image 4.7

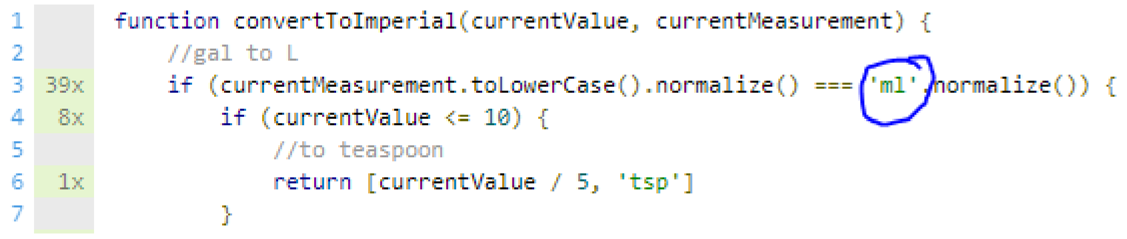


100% Statements 122/122 **100%** Branches 142/142 **100%** Functions 3/3 **100%** Lines 122/122

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

File		Statements		Branches		Functions		Lines	
convertToImperial.js	🟩	100%	66/66	100%	70/70	100%	1/1	100%	66/66
convertToMetric.js	🟩	100%	54/54	100%	72/72	100%	1/1	100%	54/54
scale.js	🟩	100%	2/2	100%	0/0	100%	1/1	100%	2/2

Image 4.8



```
     function convertToImperial(currentValue, currentMeasurement) {
         //gal to L
39x      if (currentMeasurement.toLowerCase().normalize() === 'ml'.normalize()) {
 8x          if (currentValue <= 10) {
                 //to teaspoon
 1x              return [currentValue / 5, 'tsp']
             }
```

9. The above process illustrates one example of how we plan to test our team's JavaScript code for this project. Our team will continue to use Jest to help ensure our code is fully tested. This is just an example for our unit testing but other types of testing might use a

different framework or program other than Jest. Those details will be established when we have started that phase of our testing.

4.1. Unit Testing

The following units will be tested in isolation; unit conversion for measurements, retrieving data from a URL, scaling recipe measurements, ensuring the URL is safe, and displaying UI elements.

Unit Conversion for measurements: Our plan is to have a function that has the ability to convert recipe amounts between metric and imperial units and vice versa.

Retrieving data from a URL: Our plan is to have a function that takes an URL as input and returns back the data from the website the URL points to.

Scaling Measurements: Our plan is to have a function that can scale the amounts needed for each ingredient based on how many batches the user plans to make.

Ensure the URL is safe: We will make a function that examines the URL and detects whether it is safe or not.

4.2. Interface Testing

Our compositions of two or more interfaces will be tested through UI observations resulting from user input. We plan to use a native program of React Native called React Native CLI Quickstart, an Android emulator, and an IOS emulator to do so.

4.3. Acceptance Testing

Once all of our tests pass, we will create a presentation for our client that demonstrates and explains how all the requirements are being met. The client will also test the app themselves through hands-on interaction. After this, we will have our client check off the requirements they feel have been met.

4.4. Results

We found a failure when trying to compare a string we had converted to lowercase with one that we had set to have an upper case letter. After this problem was fixed, we successfully had created functions to scale batches and covert ingredient measurements between the metric and imperial systems.

We have learned we will need some way to retrieve the ingredient's original unit because we cannot perform a conversion without it. For example, if we are currently in teaspoons, then we need to know this information in our conversion method so we can multiply by 5 to get to mL. This means we will have to add to our current design to include some way to not only retrieve the value to be converted but to retrieve the unit it is currently in as well.

5 Implementation

Since our project is a mobile application, a large amount of the time this semester was spent on finalizing the different features and requirements for the app. Additionally, since successful mobile applications are known to have a very smooth user experience, a significant amount of time was also spent on discussing and finalizing the app structure and developing mockups of the different screens which the users would see. But as the semester comes to an end, the team has fully understood the requirements for this project, conducted sufficient technical research on the different technologies which will be utilized to develop this app and has finalized the UI designs for the app.

A majority of the technical implementation will be taking place during the second semester of this project but to ensure that everyone in the team is adequately prepared for the tasks, a small amount of work will also be done during the summer break. First, the front end developers of the team will create multiple mockups of the app using Javascript so that minimal time is spent making design decisions next semester. In addition to this, we will also aim to have all group members a little more familiar with REACT so that there's no need to take time out of the school year to try and learn it.

During the fall semester, the main objective will be to fully implement the app which was designed this semester. The team will be divided into UI design (frontend) and backend development which will be responsible for data management and the coding of the web scraping algorithm which will collect all the relevant information from the recipe websites, enhance it to fit the requirements of the app and then display this information using the decided structures. A considerable amount of time will also be dedicated to testing the app to ensure that there are no logic errors in the code and there is seamless integration within the app UI.

For a more detailed view of the project plan for the upcoming semester, visit section 2.4 of this document which contains the Gantt chart for the project. We strongly believe that this plan is feasible since all the design and features work has been completed this semester and all the effort will be on implementing the app successfully next semester.

6 Closing Material

6.1 Conclusion

The introductory phase of this project consisted of us identifying the problem statement and brainstorming an initial plan of action that we would use as a solution for the issue at hand. We identified our target audience as well and decided to choose a platform that would allow the widest range of users to access our final product. We chose to go with a mobile application for the end product as we believe this would allow people to quickly and easily get the information they need without taking too much space in the kitchen. We identified that the end users would most likely be someone who was interested in cooking, whether they're beginners or experienced in the culinary field already.

The design phase of this project was one of the more time intensive parts of the project since the team had several discussions on the structure and UI designs for the app. By the end of this phase, we were able to create realistic mockups of the URL screen and the recipe screen which provided the client and everyone in the team with an idea on how the end product might look like next semester. In this phase of the project, we also looked at pre existing apps in the market which provide similar services to the culinary industry. Research into 'competing' products gave us an opportunity to evaluate the features which the app plans on offering to the users and how they would be more beneficial compared to what is already available on the app stores. And finally, the team also had a chance to finalize the design plan for the upcoming semester. Since the app requirements, features and UI mockups have now been completed, a majority of the time during the second semester of this project will be spent on implementing the app itself. The project team will be divided into smaller groups to work on the frontend components and also write the web scraping algorithms which will be collecting the data from the original recipe websites.

For the testing part of the project we have started with unit tests for some JavaScript functions including scaling ingredient quantities and converting between metric and imperial units. After performing research, our team settled on using Jest to test our JavaScript code and we plan to continue using the framework for the remainder of the project. Jest allows us to write tests for our functions and see if our code passes them all. In addition, Jest allows us to view an html form with additional information regarding code coverage. Using Jest, our team will be able to ensure our code passes all the required tests and all of the code we write is being covered by our tests. For other forms of testing such as interface testing, our team will research ways to perform such tests as our project draws closer to needing them.

As detailed in the implementation plan above, the majority of our development work will be completed in the second semester of this project. Based on the information collected during the requirements gathering and design thinking phases, the team will be divided into smaller groups to work on the UI, frontend logic and any backend algorithms which support the functionality of the app. Members will be expected to familiarise themselves with REACT development environments so that there are minimal roadblocks next semester. Testing will also play a significant role in the implementation phase of the project where unit tests and system integration tests will be conducted after the completion of each module to ensure that there are no bugs and the app runs smoothly.

6.2 References

[1] Amit Thinks. *How to run JavaScript on Visual Studio Code.* (Oct. 13, 2020). [Online Video]. Available: https://www.youtube.com/watch?v=Z_G86SKXP3s. Accessed: Apr. 24, 2021.

[2] Ben Awad. *Running Create React Native App on Your Phone.* (Dec. 31, 2017). [Online Video]. Available: https://www.youtube.com/watch?v=mhoiwfShSnE. Accessed: Apr. 24, 2021.

[3] Jordan Walke. “React Native Documentation” React Native https://reactnative.dev/ Accessed: Apr. 25, 2021

[4] LevelUpTuts. *React Native For Everyone Preview.* (June 5, 2017). [Online Video Playlist]. Available: https://www.youtube.com/playlist?list=PLLnpHn493BHG30qU2Rw403x6w7XP8hHB5. Accessed: Apr. 24, 2021.

[5] “Setting up the development environment · React Native,” *React Native*, Mar. 12, 2021. [Online]. Available: https://reactnative.dev/docs/environment-setup. Accessed: Apr. 24, 2021.

[6] Web Dev Simplified. *Introduction To Testing In JavaScript With Jest.* (Sep. 24, 2019). [Online Video]. Available: https://www.youtube.com/watch?v=FgnxcUQ5vho. Accessed: Apr. 24, 2021.