

CS3354 Software Engineering
Final Project Deliverable 2

Green Home

Adam Shafi
Anand Menon
Emma Tung
Hamdiya Abdulhafiz
Lillian Chen
Noah Lauer
Sydney Khamphouseng

1. Delegation of Tasks

- Adam Shafi - Create class diagrams, attach draft and address feedback, Github task 1.6, project planning, created Gantt timeline, attach deliverable 1, and format final document
- Anand Menon - List software requirements, create sequence diagrams, Github task 1.6, and JUnit Testing.
- Emma Tung - Create github and add team members along with TA to repository, create class diagrams, Github task 1.6, project planning, and GitHub management.
- Hamdiya Abdulhafiz - Architectural design : apply and explain how the project will utilize the MVC model, Github task 1.6, and conclusion.
- Lillian Chen - Github task 1.5 (adding project scope), create use case diagrams, Github task 1.6
- Noah Lauer - List software requirements, create sequence diagrams, Github task 1.6, and Final Presentation.
- Sydney Khamphouseng - Gitbhub task 1.4, create use case diagrams, Github task 1.6, Project Comparison, Design Mockup, and Demo

2. Project Deliverable 1 content

1. Draft and Feedback

Project Title: Green Home

Group Members:

- Sydney Khamphouseng
- Adam Shafi
- Emma Tung
- Noah Lauer
- Lillian Chen
- Anand Menon
- Hamdiya Abdulhafiz

What the team will be doing:

Engineering an application providing a marketplace for zero-waste, environmentally-friendly household products.

Description of Motivation:

We aim to reduce shoppers' carbon footprints by offering accessible and zero-waste products; our product will help shoppers make eco-friendly choices to improve their quality of life.

Feedback to Learner

9/20/22 4:44 PM

Great project topic with a fringe benefit, as it will help protecting our mother nature.

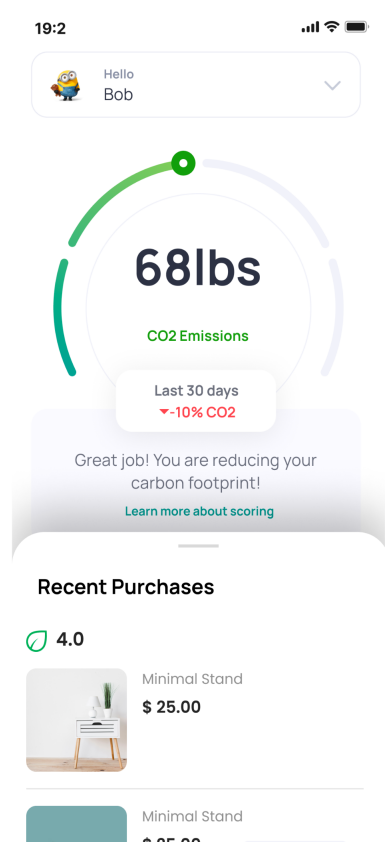
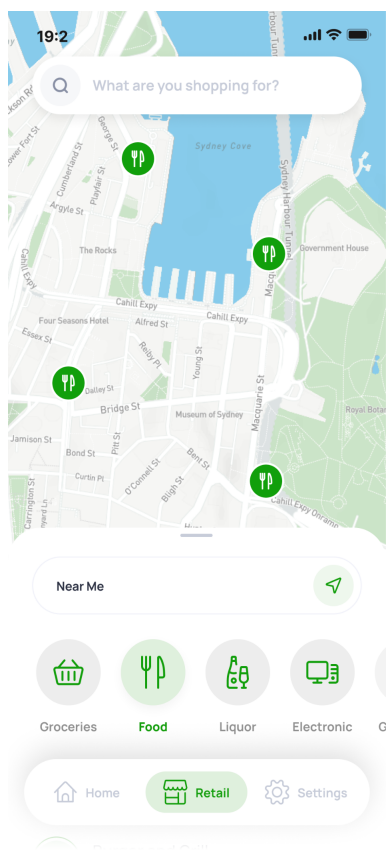
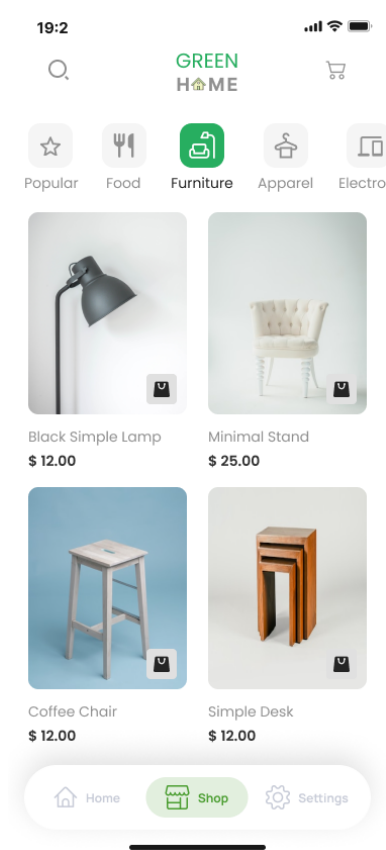
In the final report, please make sure to include comparison with similar applications -if any- and make sure that you differentiate your design from those and explicitly specify how.

Please share this feedback with your group members.

You are good to go. Have fun with the project and hope everyone enjoys the collaboration.

To comply with the feedback, we will compare our app with similar competitors like Amazon. We will explore how our design differs from other applications and exemplify our product's unique features. Unlike Amazon, our app will provide shoppings options at nearby stores to conserve travel resources and costs and to minimize packaging materials. The app will also provide shipping options. Our app's UI will differ from competitors as it will offer users the option to calculate their carbon footprint from the items they have purchased, as shown below in the figures.

Design Mockups for Green Home



2. Github Repository

<https://github.com/emmaxtung/3354-GreenHome>

3. Delegation of Tasks

- Adam Shafi - Create class diagrams, attach draft and address feedback, Github task 1.6,
- Anand Menon - List software requirements, create sequence diagrams, Github task 1.6
- Emma Tung - Create github and add team members along with TA to repository, create class diagrams, Github task 1.6
- Hamdiya Abdulhafiz - Architectural design : apply and explain how the project will utilize the MVC model, Github task 1.6
- Lillian Chen - Github task 1.5 (adding project scope), create use case diagrams, Github task 1.6
- Noah Lauer - List software requirements, create sequence diagrams, Github task 1.6
- Sydney Khamphouseng - Github task 1.4, create use case diagrams, Github task 1.6

4. Software Process Model Employed

The software process model that we will be employing is the waterfall model. The waterfall model is an approach that is popularly used in product development as it emphasizes a sequential progression of steps. The team has utilized this software process model already by outlining requirements, deadlines and guidelines for the project. Our approach while working on the final project will follow the logical progression of defining and planning the project, analyzing system specifications for business logic, outline design specifications which includes deciding on the programming language, data sources, and architecture. We will then discuss source code implementations, test our product, and at the end of the project we will be ready to present our final product.

5. Software Requirements

Functional Requirements

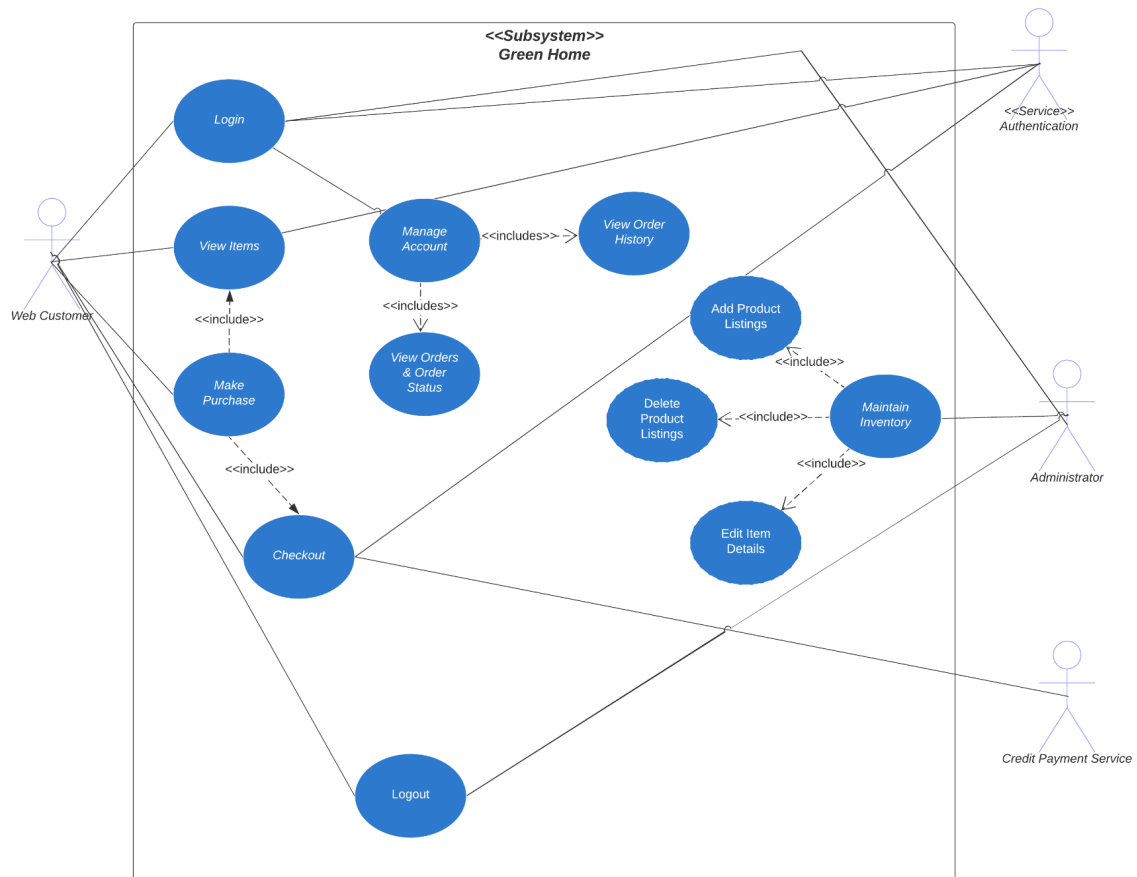
1. Customers must be able to create, modify and delete a user account
2. Customers must be able to browse, search, and filter products by criteria (from at least one provider) as a registered user or as a guest
3. Customers must be provided with a carbon footprint calculator.
4. Providers must be able to create, modify and delete a vendor account.
5. Providers must prove that products meet the environmental standards set by Green Home (EPA/ESG Score)
6. Providers must be notified when one of their items have been purchased
7. Providers must be provided a tax break and incentive calculator if qualified

Non-Functional Requirements

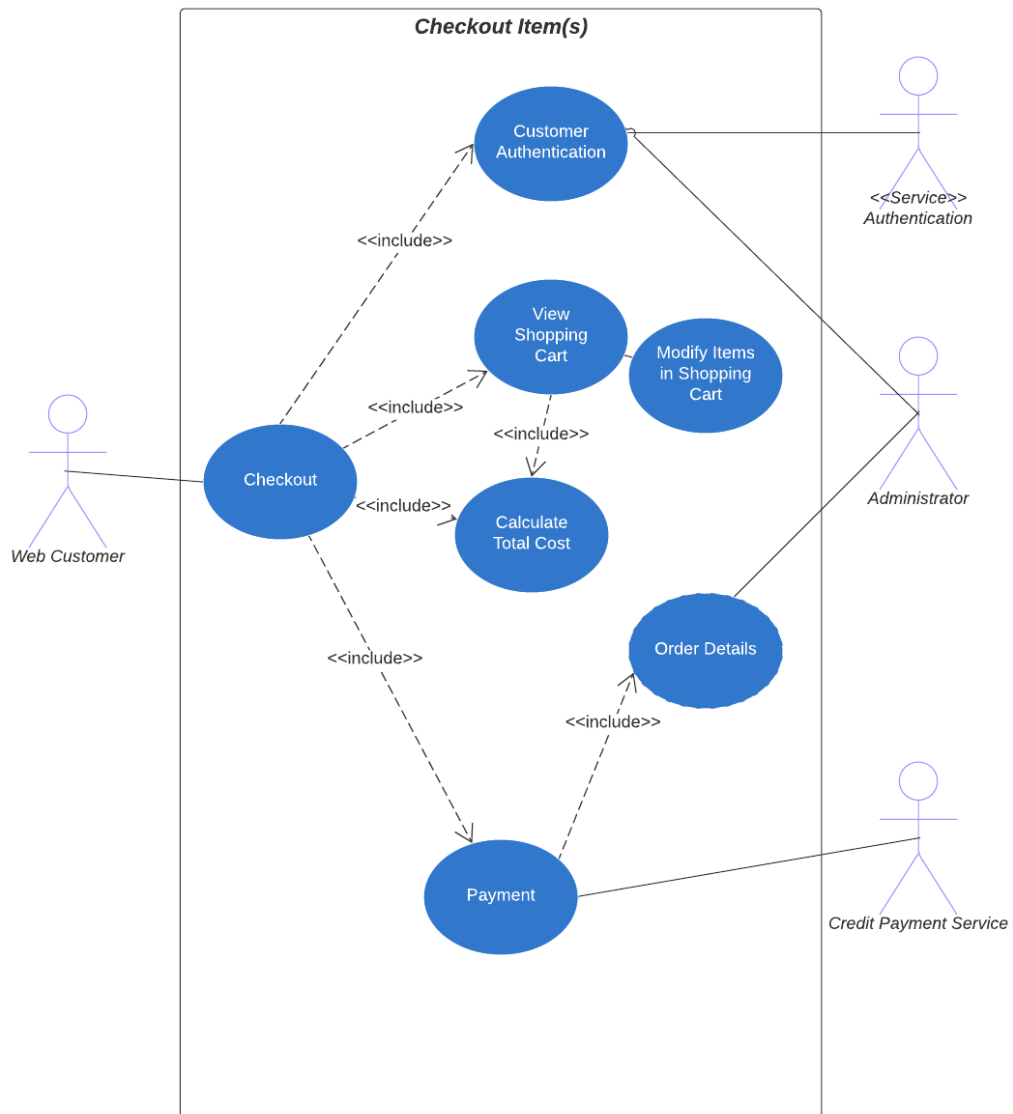
1. Service must be available 24/7 with a downtime of at most ten (10) minutes
2. Sellers need to provide proof of legitimacy and verify they are a registered business.
3. Database to hold seller and user information; server communicates with the database.
4. Any request made by customers or Providers must be handled in an average time of 5 secs.
5. Payment and customer information must be secure and protected.
6. System maintenance must not take more than 1 hour, and backups must be in place.
7. Systems must be in place to prevent unexpected downtime (e.g., from a DDoS)
8. Maintain lawyers on retainer to assist with legal regulations
9. Advertise ethical standards that will be maintained company wide.

6. Use Case Diagrams

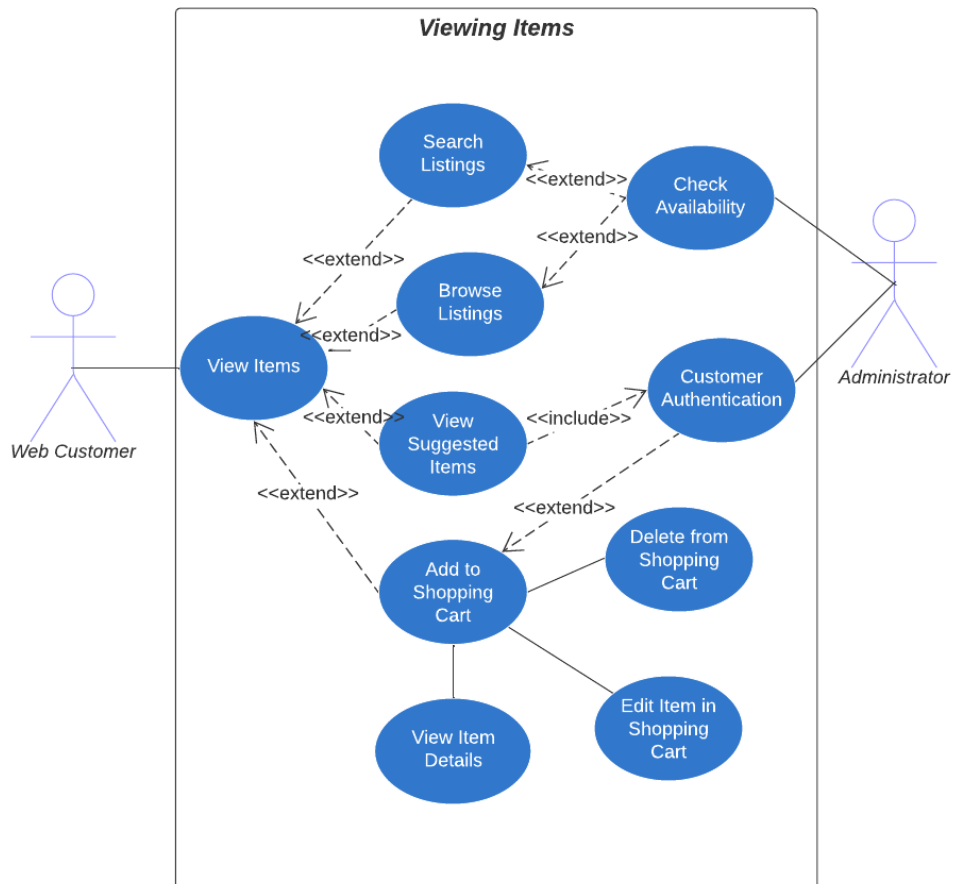
Top Level Use Case Diagram



Checkout Items Use Case Diagram

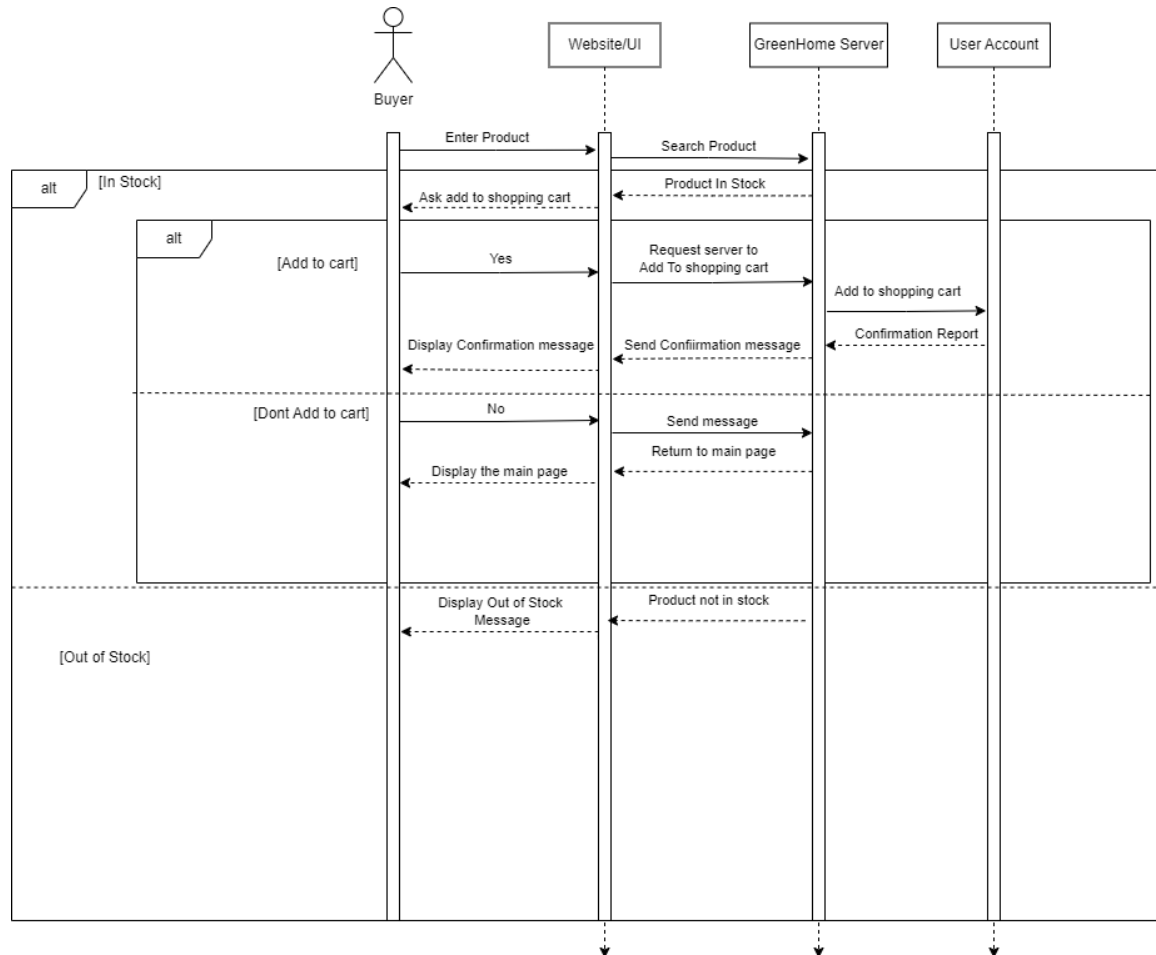


View Items Use Case Diagram

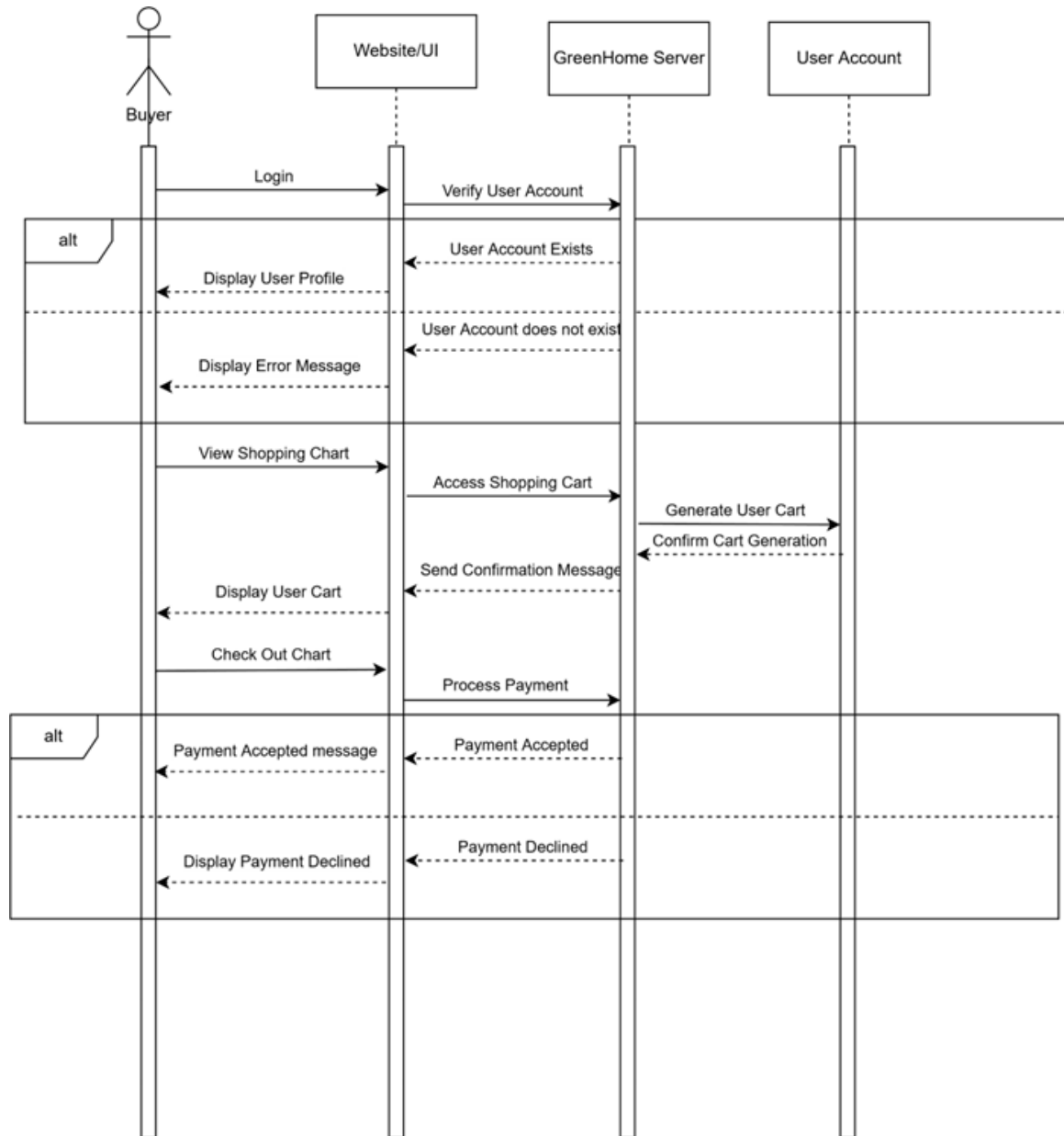


7. Sequence Diagrams

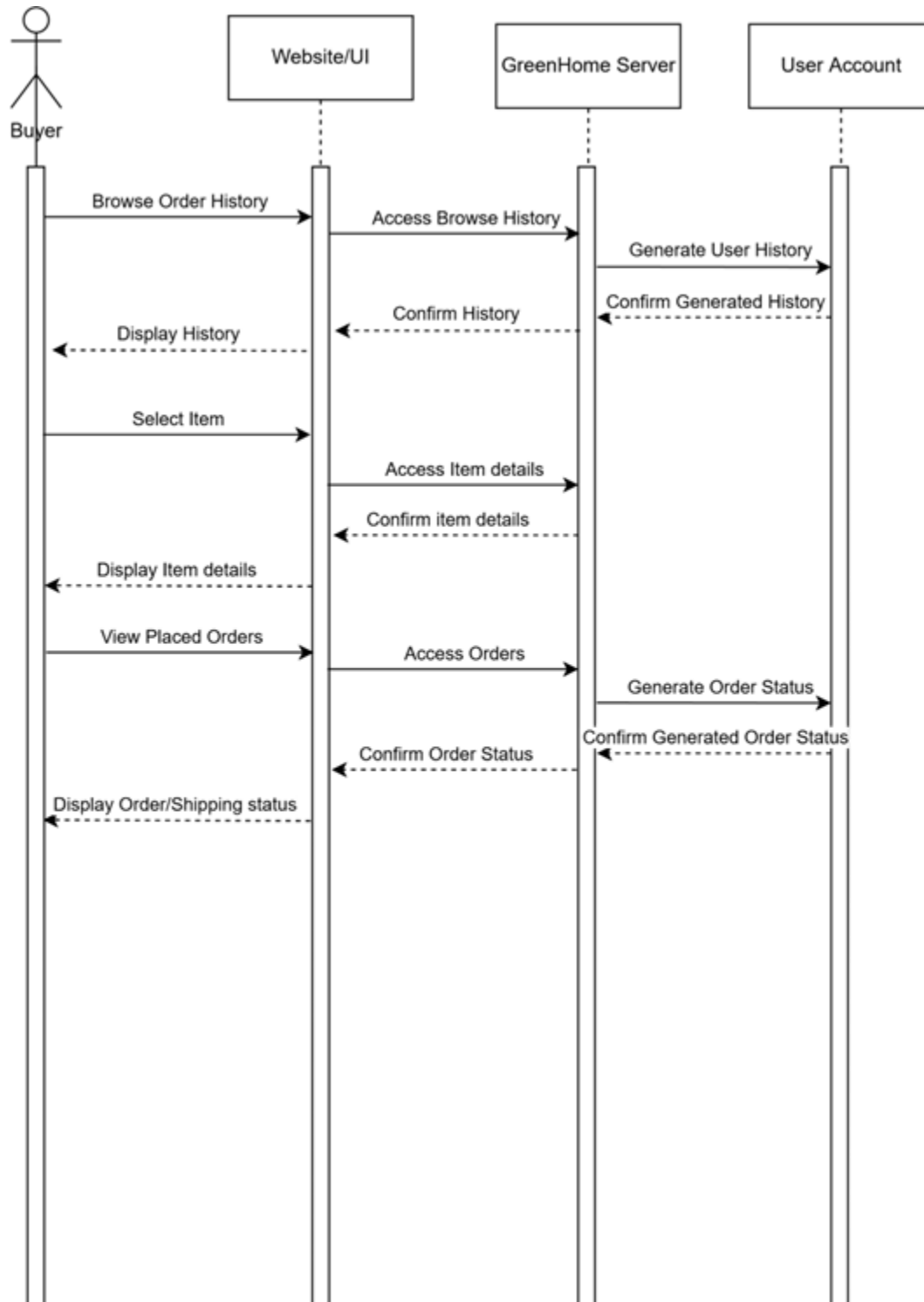
View Items Sequence Diagram



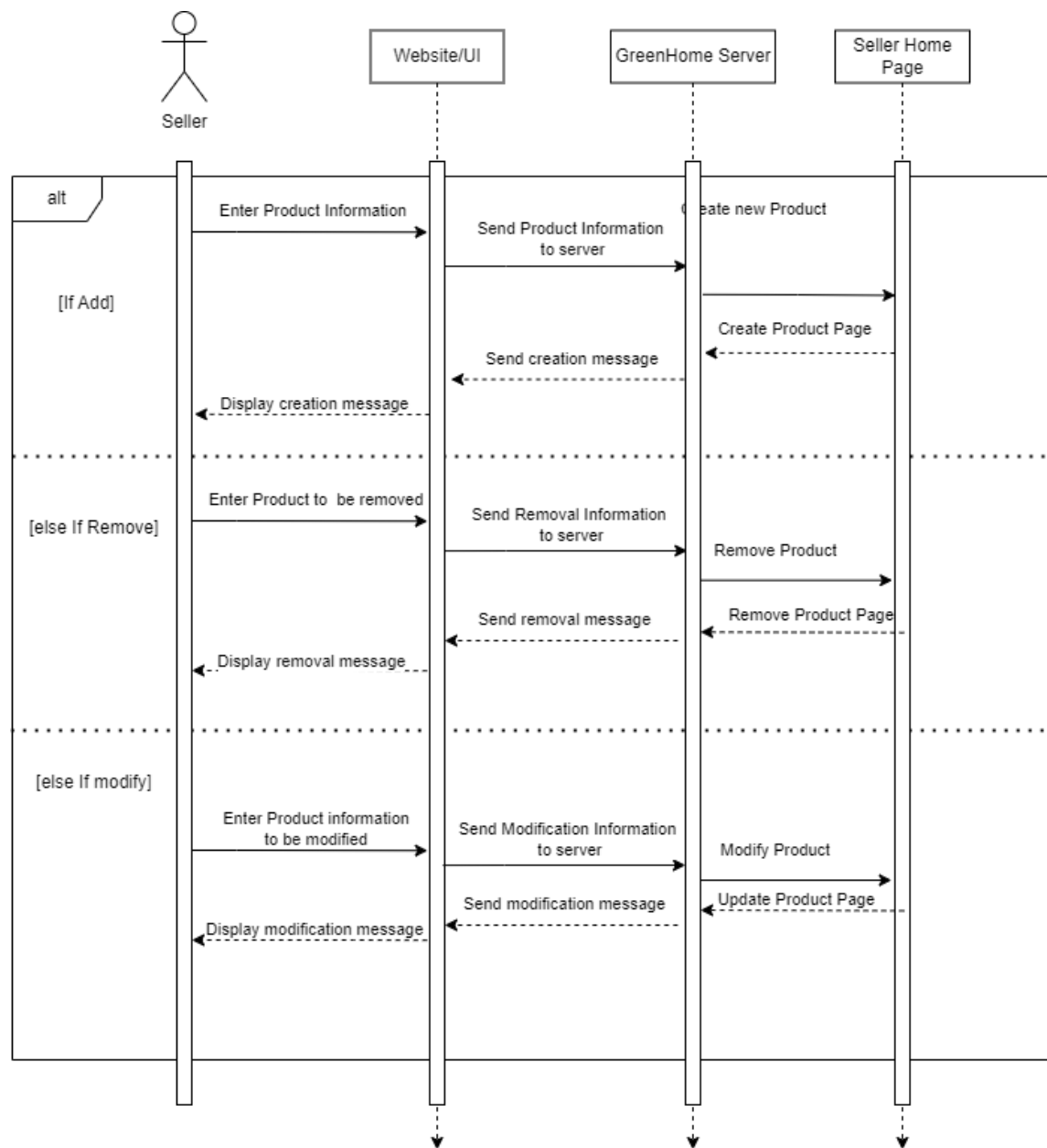
Checkout Sequence Diagram



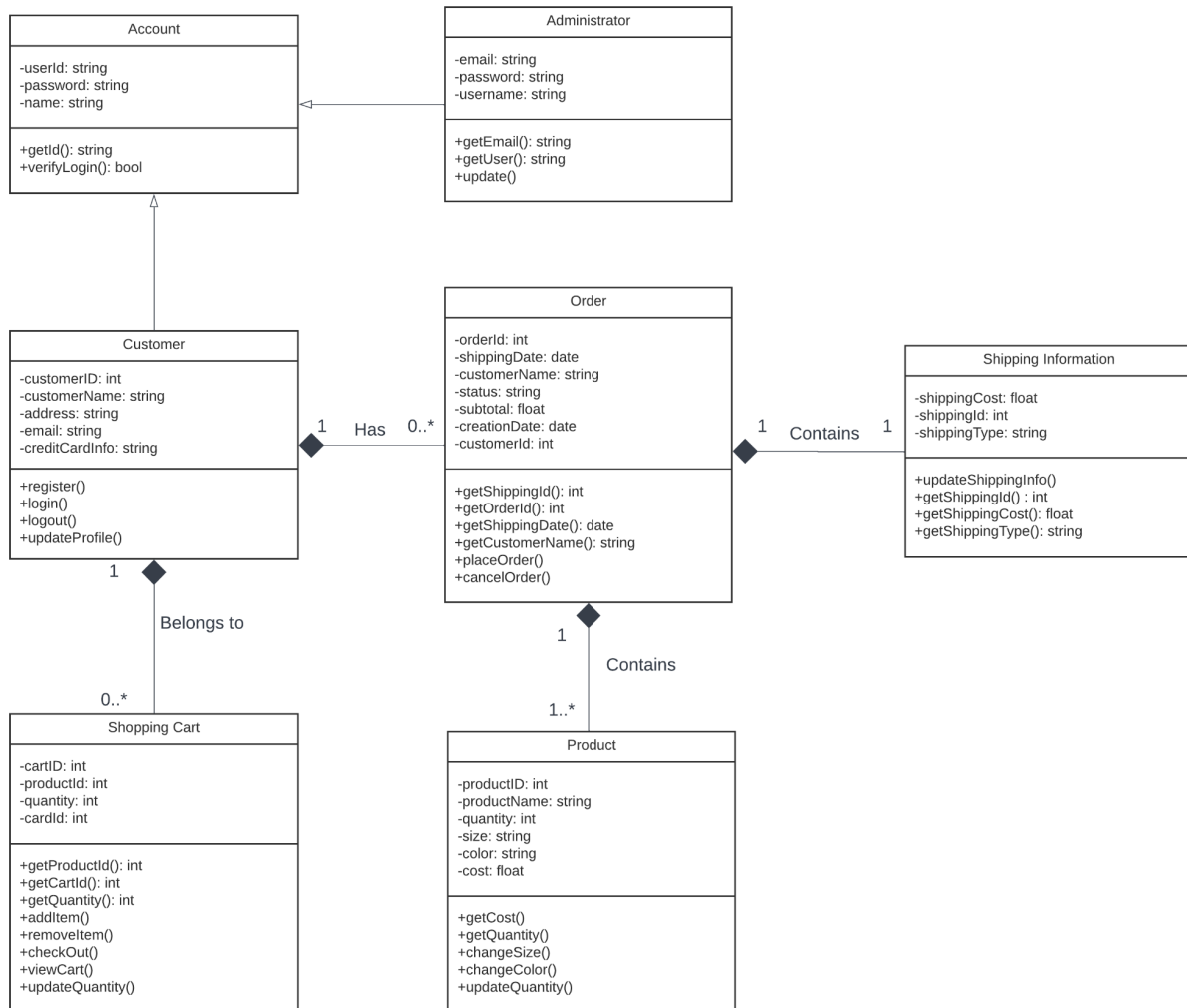
Check Order Status Sequence Diagram



Maintain Items Sequence Diagram

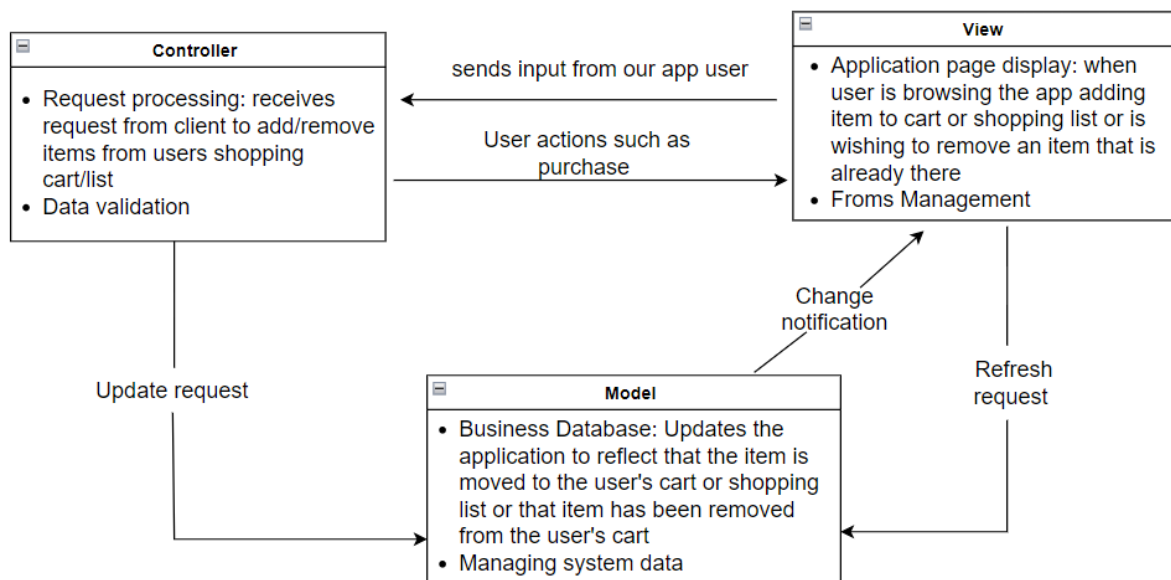


8. Class Diagram



9. Architectural Design

We chose to apply the Model-View-Controller pattern. The MVC would let the client make a request for what they may be looking for and the controller would deal with that request by passing it on to the model which will handle all of the data logic of a request and managing the system data. Once the model sends a response back to the controller, the controller will then interact with the view and then view will then send the final presentation to the controller and the controller will handle sending that presentation back to the user.



3. Project Planning

3.1 Project Scheduling

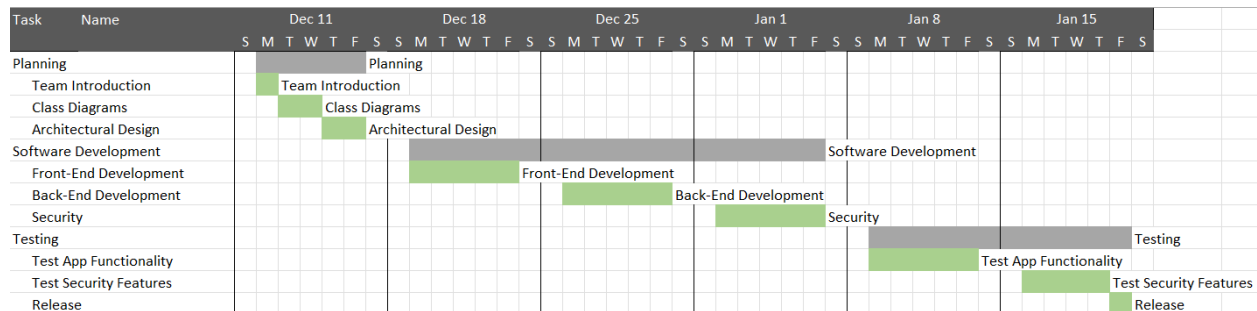
6 Week Program. 8 Hours/Day, Weekend excluded.

Start: December 12, 2022

End: January 20, 2023

This is a large-scale project that will take numerous of hours of development and additional planning. Therefore, we have established a 6 week program, with 8 hour workdays and weekends excluded, that gives ample time for development of the application. Furthermore, this accounts for minor setbacks in the process as we have allocated a generous time frame.

Gantt Timeline



3.2 Cost, Effort, and Price Estimation

Our team chose to implement the Application Composition modeling technique since it is an effective model for prototyping an application in the early stages. It is commonly used for web applications as it includes steps to plan out the number of screens and pages displayed as well as what features are produced per screen. It provides an accurate estimation of difficulty which is calculated from various factors such as the required elements, programming languages, and developer experience. This model also allows for reuse of existing components which is extremely beneficial for web applications.

Number of screens, reports, and 3GL components

- a. 7 screens
 - i. Login
 - 1. Accesses 1 data table (account info)
 - ii. Homepage
 - 1. View: Selection
 - a.
 - 2. View: Options
 - 3. View: Account
 - a. Accesses 1 data table
 - 4. View: Map
 - iii. Catalog
 - 1. View: Search screen
 - 2. View: Product information
 - iv. Shopping Cart
 - v. Checkout
 - 1. Accesses data table (inventory)
 - vi. Order Confirmation
 - vii. Profile
- b. 1 Reports
 - i. Catalog Report
 - 1. 1 section
 - 2. Accesses 1 data table
- c. 4 3GL Components
 - i. Java: backend
 - ii. CSS: frontend
 - iii. JavaScript: frontend
 - iv. React.js: frontend

2. Complexity Levels

For Screens				For Reports			
# of Views Contained	# of data tables			# of Sections Contained	# of data tables		
	< 4	< 8	8+		< 4	< 8	8+
< 3	simple	simple	medium	0 or 1	simple	simple	medium
3-7	simple	medium	difficult	2 or 3	simple	medium	difficult
> 8	medium	difficult	difficult	4+	medium	difficult	difficult

- a. 7 screens
 - i. Login
 - ii. Homepage
 - iii. Catalog
 - iv. Shopping Cart
 - v. Checkout
 - vi. Order Confirmation
 - vii. Profile

3. Complexity Weights

Object Type	Complexity Weight		
	Simple	Medium	Difficult
Screen	1	2	3
Report	2	5	8
3GL Component			10

Screens: $7 \times 1 = 7$

Reports: $5 \times 1 = 5$

3GL components: $4(10) = 40$

4. Object Point (OP) count
 - a. $7 \text{ (screens)} + 5 \text{ (reports)} + 40 \text{ (3GL components)} =$
 - b. $OP = 52$

5. New Object Points (NOPs)
 - a. ~40% reuse
 - b. $NOP = 52 \times (100 - 40) / 100 = \mathbf{31.2}$

6. Object Point Productivity:

a.

ICASE maturity & capability; Developers experience & capability	Low	Very Low	Nominal	High	Very High
PROD (NOP/ month)	4	7	13	25	50

- b. The development environment is low and the team has little experience in the application domain (i.e., very low)
 - i. $(7 + 4) / 2 = \mathbf{5.5}$

7. Person-month effect:

- a. $31.2 / 5.5 = 5.68 \text{ person-month}$

3.3 Estimated Cost of Hardware Products

- AWS Server:
 - \$4.44/month
 - ~\$53.28/year
- Total Cost
 - \$53.28/year

3.4 Estimated Cost of Software Products

- IntelliJ IDEA Ultimate
 - \$169/year per person
 - ~\$845/year for a team of 5 developers
- Jira Premium Plan
 - \$15.25/month per person
 - ~\$122/month for a team of 8
 - ~\$1,464/year for a team of 8
- Figma Organization Plan
 - \$45/month per person
 - \$540/year per person
- Total Cost
 - \$845/year + \$1,464/year + \$540/year = \$2,849/year

3.5 Estimated Cost of Personnel

- Product Manager (1 Person)
 - $\$15/\text{hour} * 8 \text{ hours/day} * 5 \text{ days/week} * 6 \text{ weeks} = \3600
- Product Designer (1 Person)
 - $\$12/\text{hour} * 8 \text{ hours/day} * 5 \text{ days/week} * 6 \text{ weeks} = \2880
- Software Developer (5 People)
 - $\$10/\text{hour} * 8 \text{ hours/day} * 5 \text{ days/week} * 6 \text{ weeks} = \$2400/\text{developer}$
 - $\$2400/\text{developer} * 5 \text{ developer} = \12000
- Quality Assurance Engineer (1 Person)
 - $\$10/\text{hour} * 8 \text{ hours/day} * 5 \text{ days/week} * 6 \text{ weeks} = \2400
- Total Cost
 - $\$3600 + \$2880 + \$12000 + \$2400 = \$20880$

4. Test Plan

We employ Unit and Integration testing strategies to test our application. In the following example we test the encryption and decryption modules of the app which is written in Python [3] using RSA modules from the pycryptodome library [4]. These are important for storing sensitive information and it is vital they can be decrypted correctly. As stated before we use the integration test and the unit test and test the encryption and decryption on a sample sql database and a sample library source file. The test code and a ReadMe.md with more details has been attached to this document and uploaded to github.

Note: Due to the nature of this file Windows Defender may flag the code as a Virus. Use Linux or the github link [<https://github.com/emmaxtung/3354-GreenHome>] to view and test the contents.

Description

There are two test cases:

- * An unit test, test_unittest.py
- * An integration test, test_integrationtest.py

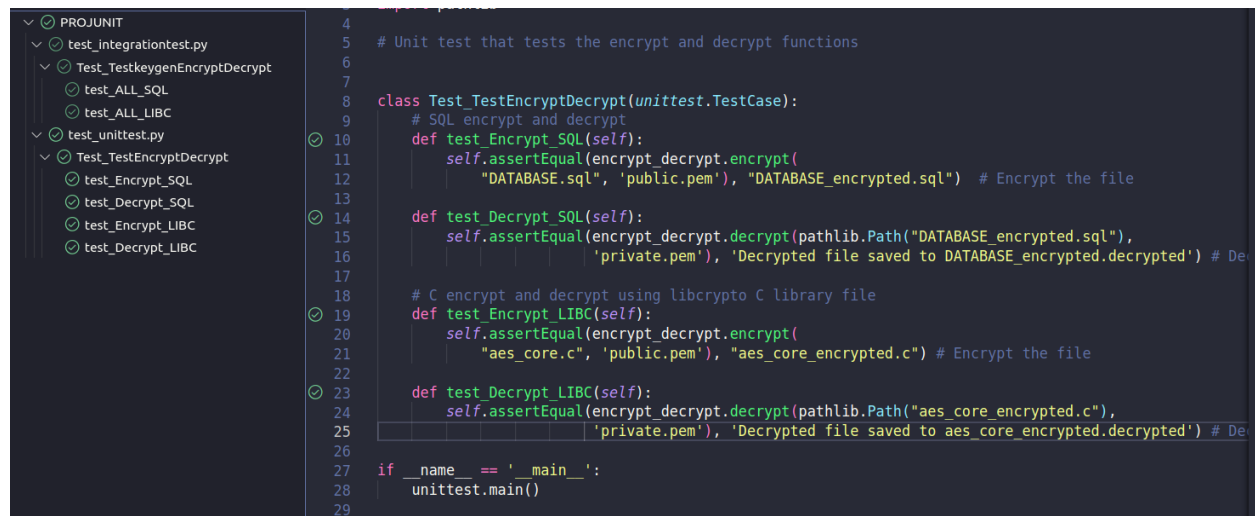
Unit Test

The unit test is used to test the encrypt_decrypt.py file

The unit test tests the following functions:

- * encrypt()
- * decrypt()

Result:



```
1  # Import packages
2
3  # Unit test that tests the encrypt and decrypt functions
4
5  # SQL encrypt and decrypt
6
7
8  class Test_TestEncryptDecrypt(unittest.TestCase):
9
10     def test_Encrypt_SQL(self):
11         self.assertEqual(encrypt_decrypt.encrypt(
12             "DATABASE.sql", 'public.pem'), "DATABASE_encrypted.sql") # Encrypt the file
13
14     def test_Decrypt_SQL(self):
15         self.assertEqual(encrypt_decrypt.decrypt(pathlib.Path("DATABASE_encrypted.sql"),
16             'private.pem'), 'Decrypted file saved to DATABASE_encrypted.decrypted') # De
17
18     # C encrypt and decrypt using libcrypto C library file
19     def test_Encrypt_LIBC(self):
20         self.assertEqual(encrypt_decrypt.encrypt(
21             "aes_core.c", 'public.pem'), "aes_core_encrypted.c") # Encrypt the file
22
23     def test_Decrypt_LIBC(self):
24         self.assertEqual(encrypt_decrypt.decrypt(pathlib.Path("aes_core_encrypted.c"),
25             'private.pem'), 'Decrypted file saved to aes_core_encrypted.decrypted') # De
26
27 if __name__ == '__main__':
28     unittest.main()
29
```

Integration Test

The integration test is used to test the entire system from the keygen.py file to the encrypt_decrypt.py file

The integration test tests the following functions:

- * generate_keys()
- * encrypt()
- * decrypt()

Result:

```
3 import encrypt_decrypt # Encrypts and decrypts the file
4 import pathlib
5
6 # Integration test that tests the encrypt and decrypt functions after creating a new public and private key
7 # The decryption will fail if the keys are not generated properly
8
9
10 class Test_TestkeygenEncryptDecrypt(unittest.TestCase):
11
12     def test_ALL_SQL(self):
13
14         self.assertEqual(keygen.generateKeyPair(),
15                          "Keys generated") # Generate the keys
16
17         self.assertEqual(encrypt_decrypt.encrypt(
18             "DATABASE.sql", 'public.pem'), "DATABASE_encrypted.sql") # Encrypt the file
19
20         self.assertEqual(encrypt_decrypt.decrypt(pathlib.Path("DATABASE_encrypted.sql"), 'private.pem'),
21                          "Decrypted file saved to DATABASE_encrypted.decrypted") # Decrypt the file
22
23     def test_ALL_LIBC(self):
24
25         self.assertEqual(keygen.generateKeyPair(),
26                          "Keys generated") # Generate the keys
27
28         self.assertEqual(encrypt_decrypt.encrypt(
29             "aes_core.c", 'public.pem'), "aes_core_encrypted.c") # Encrypt the file
30
31         self.assertEqual(encrypt_decrypt.decrypt(pathlib.Path("aes_core_encrypted.c"),
32                                                  'private.pem'), "Decrypted file saved to aes_core_encrypted.decrypted") # Decrypt the file
```

5. Project Comparison

Amazon's Climate Pledge Friendly

There are several web and mobile applications that focus on providing consumers with eco-friendly products. A major competitor to our application would be the multinational e-commerce company, Amazon. Amazon has a feature where users can shop for eco-friendly products under Amazon's "Climate Pledge Friendly" section. The basis behind "Climate Pledge Friendly" is that Amazon is "partnered with trusted third-party certifications [1]" and created their own certifications such as Compat by Design and Pre-Owned Certified to "highlight products that meet sustainability standards and help preserve the natural world [1]." If users wanted to shop for only eco-friendly products on Amazon, it is not easily accessible from the main landing page. Users must search in the search bar for "Climate Pledge Friendly" to access the list of eco-friendly products. Amazon does not provide alternatives on how to receive these products other than shipping them out from a facility to the user's residence. In comparison to Green Home, Amazon lacks the ability for users to easily access eco-friendly products immediately. Green Home is a single place where users can solely shop for eco-friendly products and have different alternatives on how to receive the products. Such as through shipment, or through finding nearby stores that have those items in stock. Green Home also allows users to track their carbon footprint through the product that they buy, which Amazon does not currently. Green Home will also provide options for eco-friendly restaurants and food choices which Amazon does not provide the same service.

GreenDay

This mobile app focuses on providing eco-friendly products and alternatives to customers. GreenDay allows users to "earn gems & be rewarded for (their) sustainable behaviors [2]." This application is very similar to Green Home's purpose and goal. GreenDay allows for the support of local businesses who are eco-conscious merchants as well allowing users to barter second hand goods within their community. However, GreenDay is only available in Singapore. Whilst our app, Green Home, will focus on providing a mobile marketplace for everyone globally. In respect to design, GreenDay's app design is similar to many e-commerce applications, which our app Green Home also takes after as well. The main difference between GreenDay and Green Home is that Green Home does not allow for the bartering of goods between the community nor does it have a reward system. Green Home focuses on allowing users to purchase eco-friendly goods and having users immediately see their impact by doing so through the carbon footprint calculator.

Green Home's Design

The notable difference between other applications such as Amazon and GreenDay is that Green Home focuses on giving the user a pleasant experience while they make eco-conscious choices. Green Home provides the user with a map to purchase a product if it is nearby the user, helping the user make eco-conscious transportation choices. Green Home will also provide the user with a carbon footprint calculator which helps track the user's carbon footprint based on the items they have purchased. The main page is the shopping catalog which provides immediate use and access to the user since Green Home is an e-commerce app, it is ideal to make the main purpose of the app be readily available and accessible.

6. Conclusion

[10 POINTS] Conclusion - Please make an evaluation of your work, describe any changes that you needed to make (if any), if things have deviated from what you had originally planned for and try to give justification for such changes.

7. References

- [1] A. Inc., “Amazon.com: Climate pledge friendly,” *Amazon Climate Pledge Friendly*. [Online]. Available: <https://smile.amazon.com/cdn.amazon.com/b?ie=UTF8&node=21221607011>. [Accessed: 10-Nov-2022].
- [2] G. Day, “About Us,” *GreenDay*, 2020. [Online]. Available: <https://www.greendayapp.com/about-us>. [Accessed: 15-Nov-2022].
- [3] "G. van Rossum, Python tutorial, Technical Report CS-R9526, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, May 1995." [Accessed: 17-Nov-2022]
- [4] *Welcome to pycryptodome's documentation*. Welcome to PyCryptodome's documentation - PyCryptodome 3.15.0 documentation. (n.d.). Retrieved November 17, 2022, from <https://pycryptodome.readthedocs.io/en/latest/> [Accessed: 12-Nov-2022]
- [5] “The Collaborative Interface Design Tool.,” *Figma*. [Online]. Available: <https://www.figma.com/>. [Accessed: 4-Nov-2022].
- [6] “Open-source tool that uses simple textual descriptions to draw beautiful UML diagrams.,” *PlantUML.com*. [Online]. Available: <https://plantuml.com/>. [Accessed: 4-Nov-2022].
- [7] I. Sommerville, *Software engineering*. Hallbergmoos/Germany: Pearson, 2018.

8. GitHub Repository

<https://github.com/emmaxtung/3354-GreenHome>