

CS 3354 Software Engineering
Final Project Deliverable 2

Project Green Home

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

November 19, 2022

1 Delegation of Tasks

Adam Shafi created class diagrams, attached draft and address feedback, completed GitHub task 1.6, planned project, created the Gantt timeline and sequence diagrams, attached Deliverable 1, and formatted the draft.

Anand Menon listed software requirements, created sequence diagrams, completed GitHub task 1.6, and wrote the unit tests.

Emma Tung created the GitHub repository, added the TA and team members to the repository, created class diagrams, completed GitHub task 1.6, planned the project, and managed the GitHub repository.

Hamdiya Abdulhafiz composed the architectural design (e.g., application and explanation of how the project will use the MVC model), completed GitHub task 1.6, and wrote the conclusion.

Lillian Chen completed GitHub tasks 1.5 and 1.6, created the use-case diagrams, and composed and revised the current version of this document.

Noah Lauer listed software requirements, created sequence diagrams, completed GitHub task 1.6, and composed the final presentation.

Sydney Khamphouseng completed GitHub tasks 1.4 and 1.6 and created the use-case diagrams, project comparison, design mock-ups, and demos.

2 Project Deliverable 1 Content

2.1 Draft and Feedback

To comply with the feedback (Figure 2), we will compare our app with similar competitors like Amazon. We will explore how our design differs

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.

Figure 1: The header for our team's first deliverable.

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.

Figure 2: Feedback for our team's first deliverable.

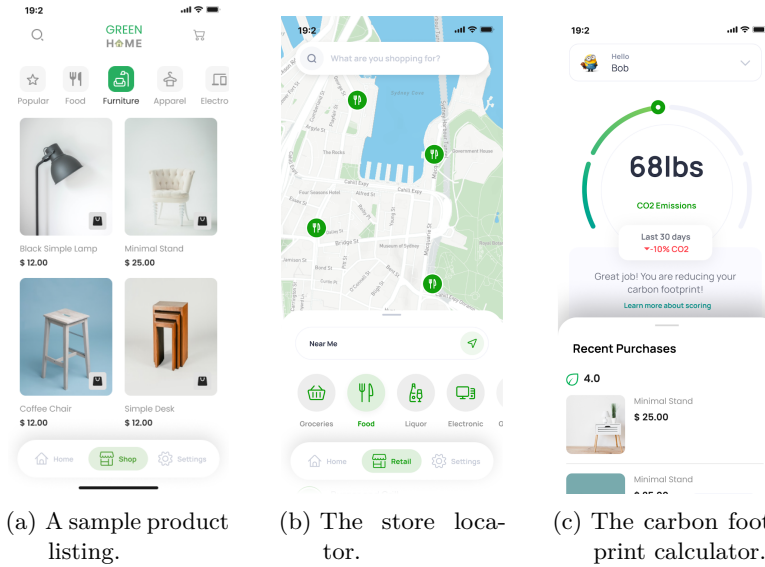


Figure 3: Design mockups for Green Home.

from other applications and exemplify our product’s unique features. Unlike Amazon, our app allows users to shop from nearby stores both to conserve travel expenses and to minimize packaging materials. The app also provides shipping options. Our app’s UI differs from competitors as it offers users the option to calculate their carbon footprint from the items they have purchased. See Figures 3a to 3c for the relevant mockups.

2.2 GitHub Repository

Our first deliverable can be accessed via the link: <https://github.com/emmaxtung/3354-GreenHome>.

2.3 Delegation of Tasks

Adam Shafi created class diagrams, attached draft and address feedback, and completed GitHub task 1.6.

Anand Menon listed software requirements, created sequence diagrams, and completed GitHub task 1.6.

Emma Tung created the GitHub repository, added the TA and team members to the repository, created class diagrams, and completed GitHub task 1.6.

Hamdiya Abdulhafiz composed the architectural design (e.g., application and explanation of how the project will use the MVC model), and completed GitHub task 1.6.

Lillian Chen completed GitHub tasks 1.5 and 1.6 and created the use-case diagrams.

Noah Lauer listed software requirements, created sequence diagrams, and completed GitHub task 1.6.

Sydney Khamphouseng completed GitHub tasks 1.4 and 1.6 and created the use-case diagrams.

2.4 Software Process Model Employed

We will be using the waterfall software process model, a popular approach in product development for its emphasis on the sequential progression of steps. Our team has already used the waterfall model by outlining requirements, deadlines, and guidelines for the project. Our approach when working on the final project will follow the logical progression of defining and planning the project, analyzing system specifications for business logic, and outlining design specifications (e.g., choice of programming language, data sources, architecture, etc.). We will then discuss source code implementations and test our product. We will present our final product at the end of the project.

2.5 Software Requirements

2.5.1 Functional Requirements

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

2.5.2 Non-Functional Requirements

1. The Service must be available 24/7 with a downtime of at most ten (10) minutes.
2. Vendors need to provide proof of legitimacy and verification as a business entity.
3. The Service must use a database to store vendor and customer information; the Service's servers will communicate with this database.
4. The Service must handle any request made by customers or vendors over an average time of five (5) seconds.

5. The Service must take adequate measures to secure and protect customer and vendor information.
6. System maintenance must take at most one (1) hour.
7. The Service must maintain sufficient backups of relevant information.
8. The Service must maintain lawyers on retainer to assist with legal regulations.
9. The Service must advertise and maintain company-wide ethical standards.

2.6 Use-Case Diagrams

See Figures 4 to 6 for our product's use-case diagrams.

2.7 Sequence Diagrams

See Figures 7 to 10 for our product's sequence diagrams.

2.8 Class Diagrams

See Figure 11 for our product's class diagrams.

2.9 Architectural Design

We chose to apply the Model-View-Controller (MVC) pattern. The MVC pattern would let the client request their desired products, and the controller passing this request to the model. The model will handle all request processing and manage system data. Once the model sends a response back to the controller, the controller modify the view. The view will then send the final presentation to the user through the controller (Figure 12).

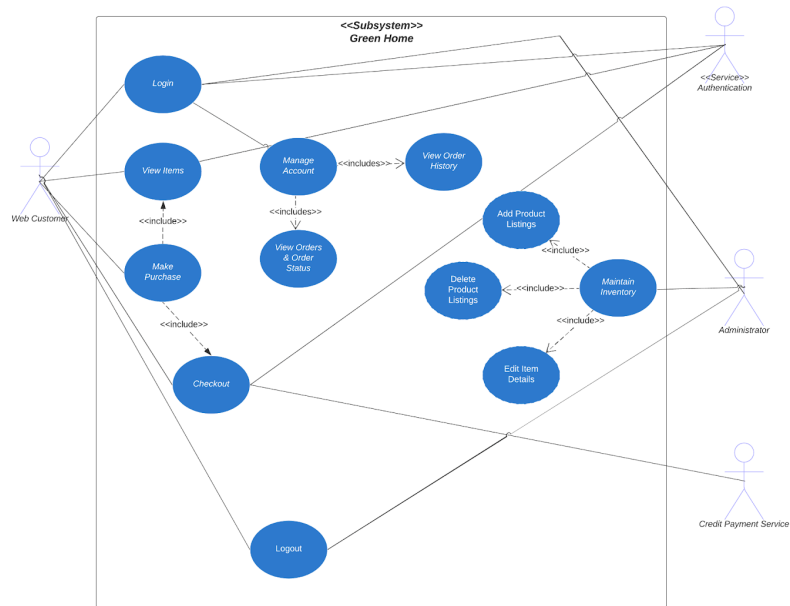


Figure 4: The top-level use-cases for Green Home.

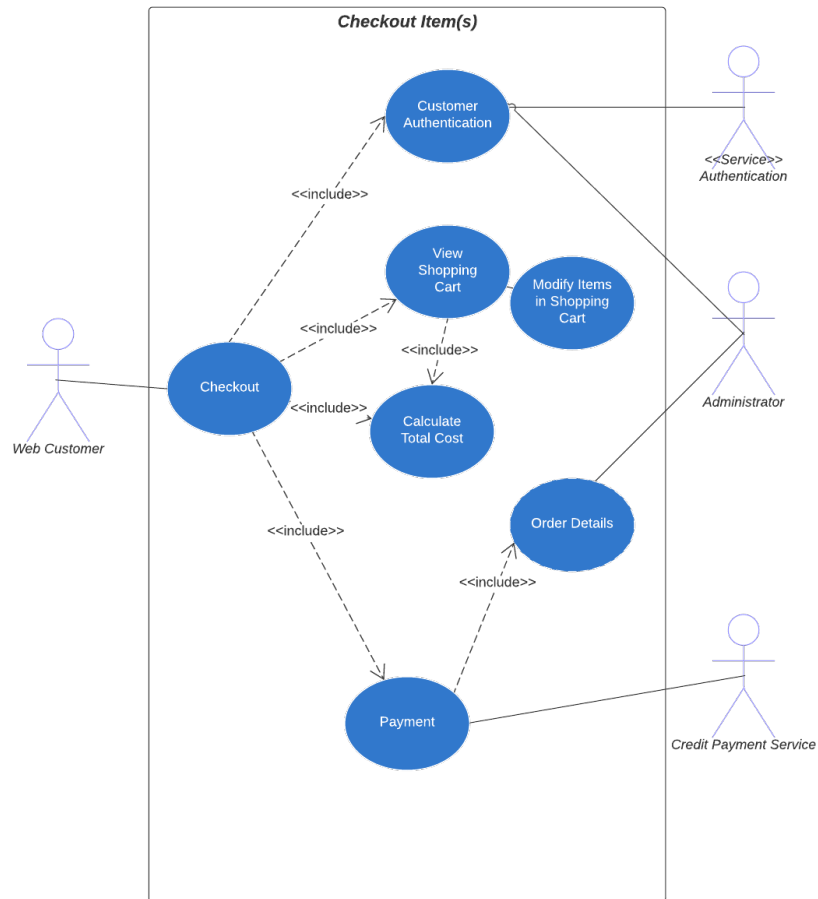


Figure 5: The use-case diagram modelling the checkout process for Green Home.

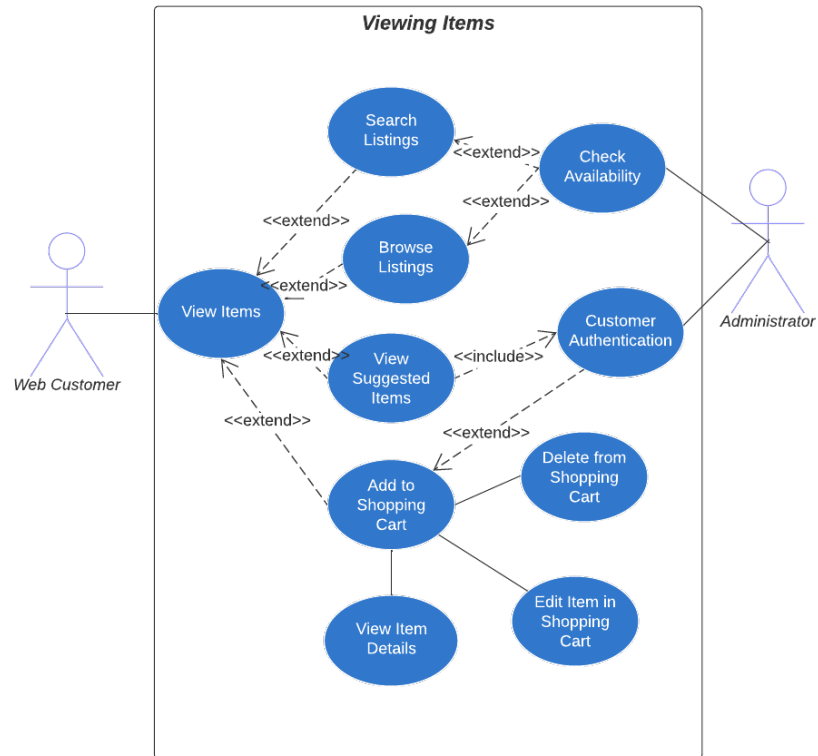


Figure 6: The use-case diagram modelling the item-view process for Green Home.

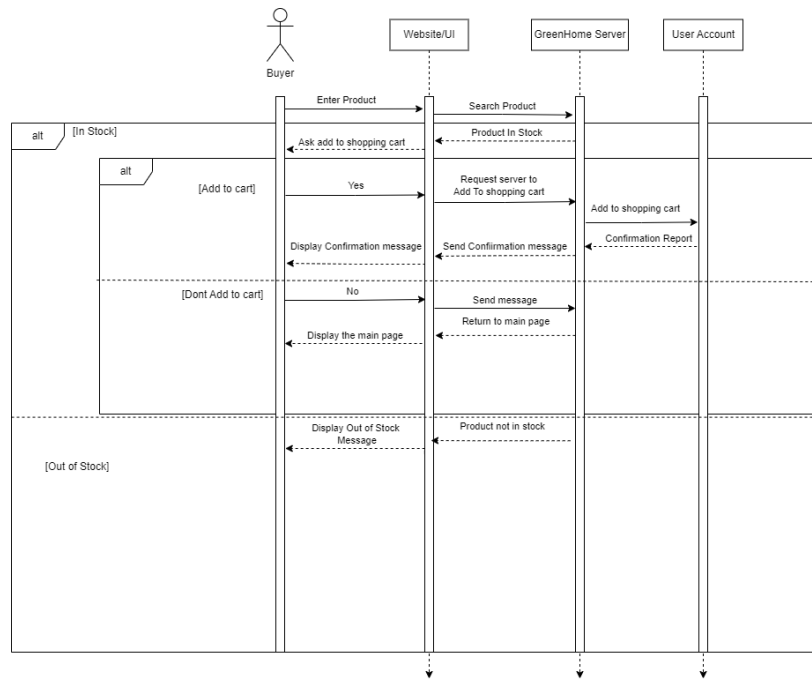


Figure 7: The sequence diagram describing the item-view process for Green Home.

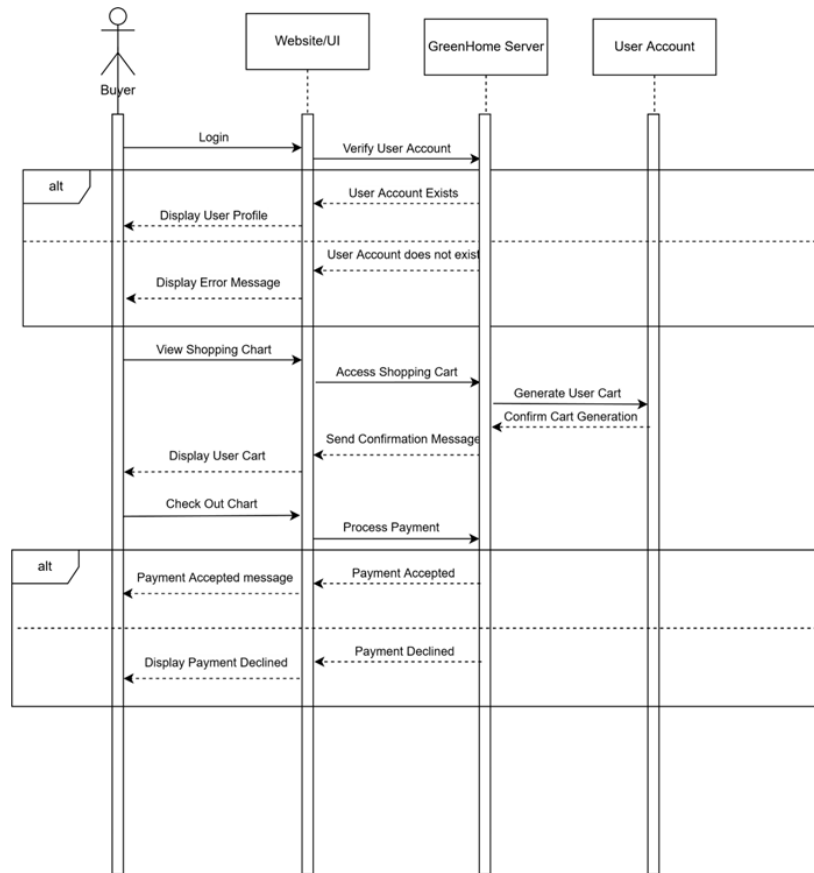


Figure 8: The sequence diagram describing the checkout process for Green Home.

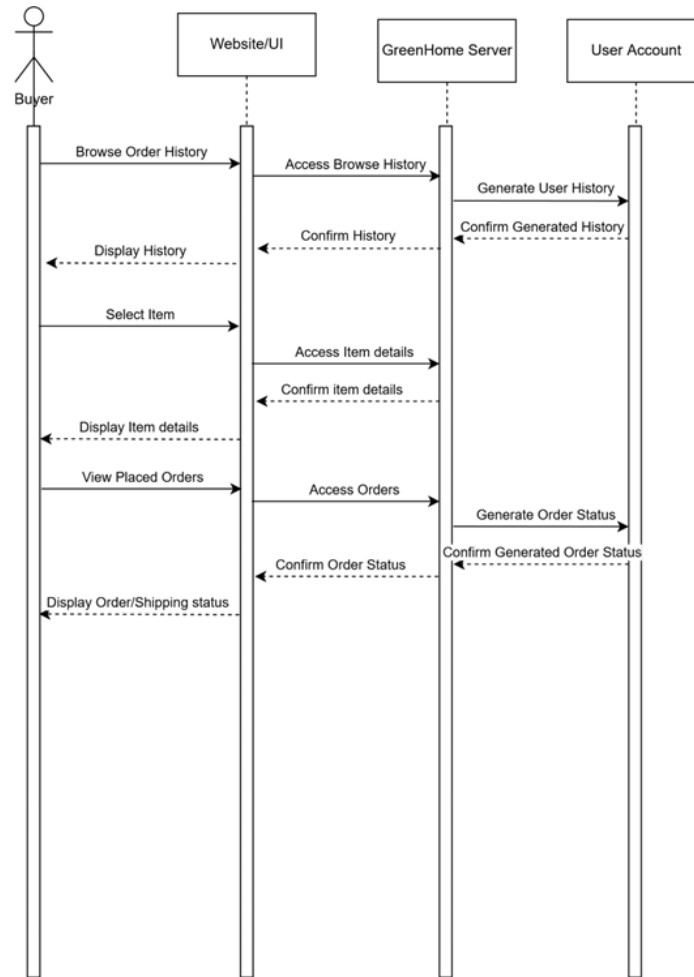


Figure 9: The sequence diagram describing the order checking process for Green Home.

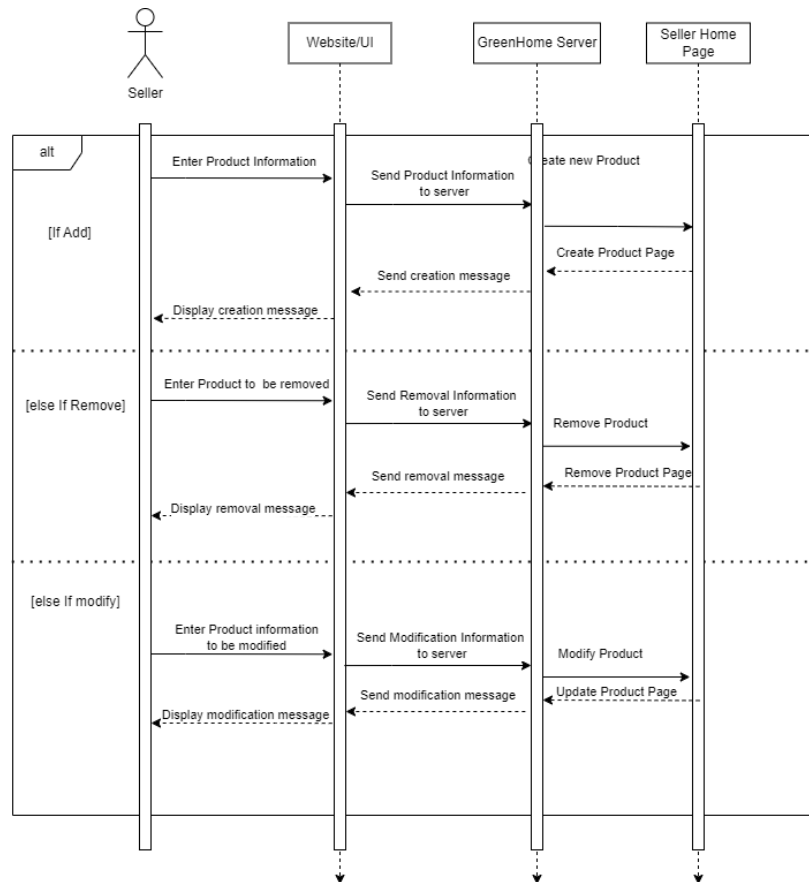


Figure 10: The sequence diagram describing the item maintenance process for Green Home.

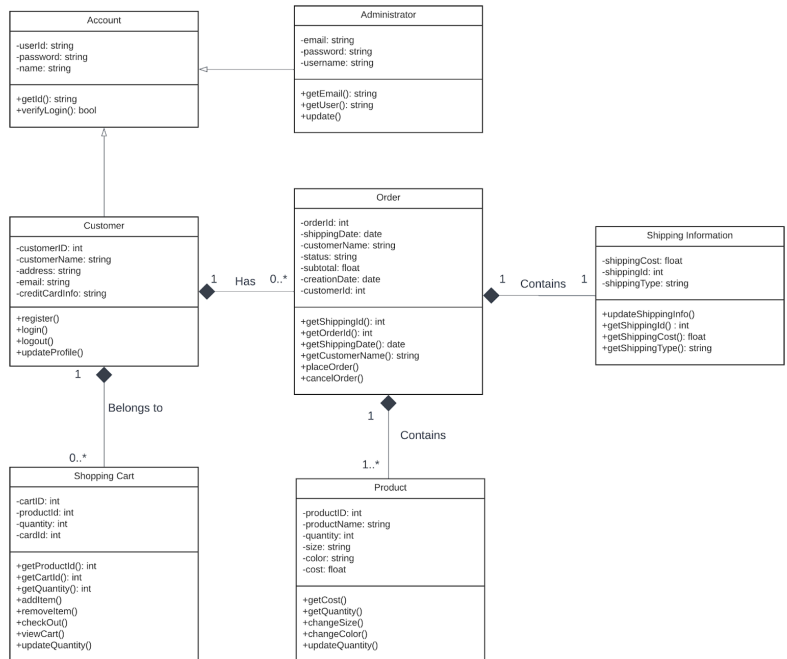


Figure 11: The class diagram for Green Home.

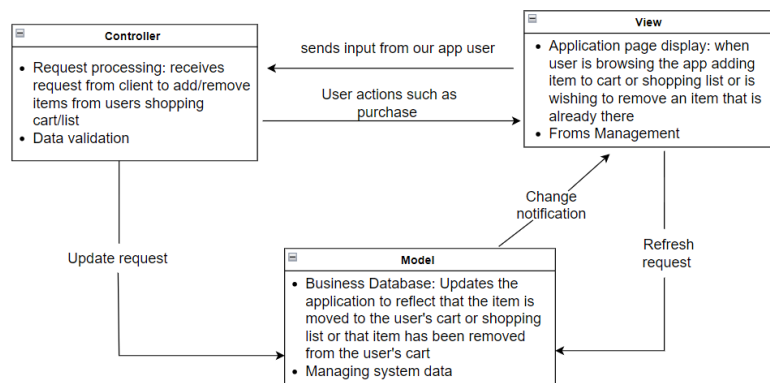


Figure 12: The annotated MVC model describing core interactions in Green Home.

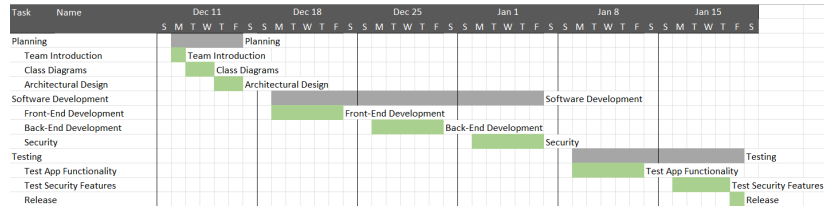


Figure 13: The Gantt timeline for Green Home.

3 Project Planning

3.1 Project Scheduling

We anticipate that our eight-person team can implement our product over six (6) weeks standard work weeks (i.e., eight (8) hours per day, excluding weekends). We will begin our project implementation on December 12, 2022, and we should finish the project by January 20, 2023. The allotted time should grant our team ample time to develop our product, accounting for minor setbacks. See Figure 13 for our project's Gantt timeline.

3.2 Cost, Effort, and Price Estimation

Our team chose to implement the Application Composition modeling technique for its effectiveness in prototyping an application. Web applications commonly implement this model as it includes steps to plan out the number of screens and pages displayed and the number of features produced per screen. It provides an accurate estimation of difficulty calculated from factors such as the required elements, programming languages, and developer experience. This model also allows for the reuse of existing components which greatly benefits web applications.

Our estimation begins with the enumeration of the screens, reports, and 3GL components of Green Home (Figure 14). We estimate that

- Seven (7) screens:
 1. Login (accesses one (1) data table for account information)
 2. Homepage
 - View: Selection
 - View: Options
 - View: Account (accesses one (1) data table)
 - View: Map
 3. Catalog
 - View: Search screen
 - View: Product information
 4. Shopping Cart
 5. Checkout (accesses one (1) data table for the inventory)
 6. Order Confirmation
 7. Profile
- One (1) report
 1. Catalog report
 - one (1) section
 - accesses one (1) data table
 2. Four (4) 3GL components
 - Java: backend
 - CSS: frontend
 - JavaScript: frontend
 - React.js: frontend

Figure 14: A breakdown of the screens, reports and 3GL components found in Green Home.

Data tables (t)			
Views (v)	$t < 4$	$t < 8$	$t \geq 8$
$v < 3$	simple	simple	medium
$3 \leq v \leq 7$	simple	medium	difficult
$v > 7$	medium	difficult	difficult

Figure 15: Complexity levels for screens.

Data tables (t)			
Sections (s)	$s < 4$	$s < 8$	$s \geq 8$
$s < 2$	simple	simple	medium
$2 \leq s \leq 3$	simple	medium	difficult
$s > 3$	medium	difficult	difficult

Figure 16: Complexity levels for reports.

Green Home’s screens will have a “Simple” complexity (Figure 15), its reports, a “Medium” complexity (Figure 16), and its 3GL components, a “Difficult” complexity (Figure 17). We denote the complexity weights for screens, reports, and 3GL components with the variables (c_s, c_r, c_c), respectively. It follows that

$$c_s = 7(1) = 7 \quad \text{and} \quad c_r = 1(5) = 5 \quad \text{and} \quad c_c = 4(10) = 40. \quad (1)$$

The object point (OP) count o follows from Equation (1):

$$o = c_s + c_r + c_c = 52, \quad (2)$$

and the new object points (NOP) o_n with a reuse $r = 40$ is given by

$$o_n = o \left(1 - \frac{r}{100} \right) = \frac{156}{5} = 31.2 \quad (3)$$

We derive the object point productivity p according Figure 18: our team is foreign to the application domain (i.e., very low competency)

Complexity weight			
Type	Simple	Medium	Difficult
Screen	1	2	3
Report	2	5	8
3GL component	1	2	10

Figure 17: Complexity weights for common object types.

ICASE and developer competence	Very low	Low	Nominal	High	Very high
PROD (NOP/month)	4	7	13	25	50

Figure 18: The scale for production points per ICASE maturity and developer competence.

and has a low development environment, so

$$p = \frac{7 + 4}{2} = \frac{11}{2} = 5.5. \quad (4)$$

From Equation (3) and Equation (4) we derive the approximate person-months m :

$$m = \frac{o_n}{p} = \frac{312}{5.5} \approx 5.68 \text{ person-month} \quad (5)$$

3.3 Estimated Cost of Hardware Products

We anticipate that our project use a single Amazon Web Service (AWS) server which costs \$4.44 per month. The annual cost should approximate \$53.28 notwithstanding other unpredictable hardware costs.

3.4 Estimated Cost of Software Products

We will be using IntelliJ IDEA Ultimate for implementation, Jira for project management, and Figma for product design. Our choice of development environment, IntelliJ IDEA Ultimate, runs at \$169 per year per person: for five (5) developers, this totals \$4,225 annually. For Jira, we have opted for the premium plan running at \$15.25 per month per person; the total monthly cost for our eight-person team is \$122 which totals \$1,464 annually. For Figma, we have opted for the organization plan running at \$45 per month per person; the monthly cost totals \$225 which totals \$2,700 annually. The total estimated annual cost is thus \$8,389.

3.5 Estimated Cost of Personnel

The number of billable hours per person totals 240 hours (subsection 3.1). We anticipate having one (1) product manager billed at \$15 per hour, one (1) product designer billed at \$12 per hour, five (5) software developers billed at \$10 per hour, and (1) quality assurance engineer \$10 per hour. The total estimated annual cost is thus \$20,880.

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 [1] using RSA modules from the `pycryptodome` library [2]. 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.

```

4
5 # Unit test that tests the encrypt and decrypt functions
6
7
8 class Test_TestEncryptDecrypt(unittest.TestCase):
9     # SQL encrypt and decrypt
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

```

Figure 19: The results of the unit test for Green Home’s cryptographic module.



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.

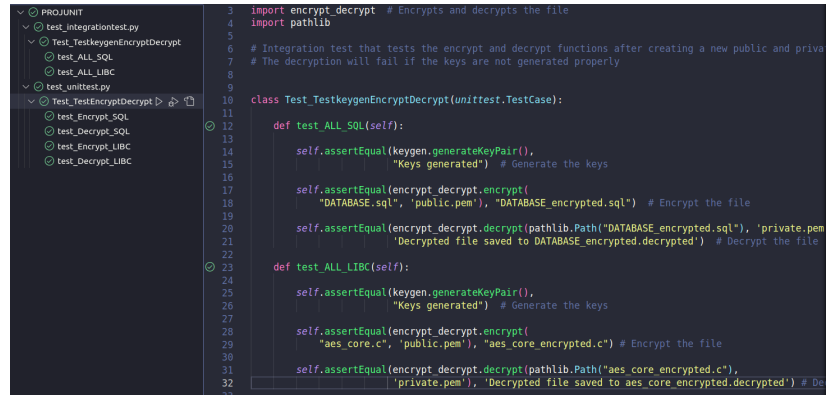
4.1 Description

We have two (2) test cases:

- a unit test (`test_unittest.py`)
- an integration test (`test_integrationtest.py`)

4.2 Unit Test

The unit test tests the functions `encrypt` and `decrypt` of the script `encrypt_decrypt.py` (Figure 19).



```
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
```

Figure 20: The results of the integration test for Green Home’s cryptographic module.

4.3 Integration Test

The integration test tests the functions of the entire cryptographic system comprising the scripts `scriptgen.py` and `encrypt_decrypt.py` (Figure 20).

5 Project Comparison

5.1 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 has “partnered with trusted third-party certifications [3]” and created their own certifications (e.g.,

“Compact by Design” and “Pre-Owned Certified”) to “highlight products that meet sustainability standards and help preserve the natural world [3].” Users who want to shop only for eco-friendly products on Amazon will find it difficult to as this the main landing page obscures this feature. Users must search in the search bar with the filter “Climate Pledge Friendly” to access the list of eco-friendly products. Amazon provides no alternatives on how to receive these products besides shipping them out from a facility to the user’s residence. Compared to Green Home, Amazon lacks the ability for users to immediately access eco-friendly products. Green Home is a single place where users can shop solely for eco-friendly products with alternatives for order fulfillment (e.g., through shipment, delivery from nearby stores, etc.). Green Home also allows users to track their carbon footprint through the products that they buy, an option absent from Amazon’s current offerings. Green Home will also provide options for eco-friendly restaurants and food choices, a service that Amazon has yet to provide.

5.2 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 [4].” This application is aligns with Green Home’s purpose. GreenDay allows for the support of local eco-conscious merchants and allows users to barter second-hand goods within their community. However, GreenDay is only available in Singapore, while our app, Green Home, will cater to a global, mobile marketplace. GreenDay’s app design is similar to many e-commerce applications, inspiring the design of Green Home. 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 emphasizes purchases of eco-friendly goods and having users immediately see their impact through the carbon footprint calculator.

5.3 Green Home's Design

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

6 Conclusion

Today, it is more important than ever to find ways to be environmentally friendly in our lives. Many existing products that one sees on grocery store shelves proliferate wasteful packaging and contribute to unnecessary waste. We wanted to provide a solution through our project, Green Home, a user-based application that helps users find environmentally-conscious alternatives to household consumer products. Throughout the planning phase, our project went smoothly; we did not need to make any serious changes due to group planning and collaboration. We implemented the application composition modeling technique as it felt more appropriate due to the fact that our project involved web screens and pages. Though we stated a six (6) week estimate for this project, we estimate a small margin of error depending on features and changes and our co-development team. Our functional and non-functional requirements have remained roughly the same as they employ functionalities that we want to deliver in our final product. According to our project scheduling, our project will take six (6) weeks, from December 12, 2022, to January 20, 2023. With a team of eight (8), we will have a total cost of \$20,800 for personnel and an \$8,442 annual upkeep cost.

7 GitHub Repository

Our project's repository is located at <https://github.com/emmaxtung/3354-GreenHome>.

References

- [1] G. v. Rossum, “Python tutorial,” Centrum voor Wiskunde en Informatica (CWI), Amsterdam, Tech. Rep. CS-R9526, May 1995.
- [2] Helder Eijs, “Welcome to PyCryptodome’s documentation.” [Online]. Available: <https://pycryptodome.readthedocs.io/en/v3.15.0/>
- [3] Amazon Incorporated, “Amazon.com: Climate Pledge Friendly.” [Online]. Available: <https://www.amazon.com/b?ie=UTF8&node=21221607011&pldnSite=1>
- [4] “About Us,” 2020. [Online]. Available: <https://www.greendayapp.com/about-us>
- [5] “Figma: the collaborative interface design tool.” [Online]. Available: <https://www.figma.com/>
- [6] “Open-source tool that uses simple textual descriptions to draw beautiful UML diagrams.” [Online]. Available: <https://plantuml.com/>
- [7] I. Sommerville, *Software Engineering*, 10th ed., ser. Pearson. Pearson, Aug. 2015.