Prime Time eCommerce: Sprint 2

CSC 4350 Software Engineering, Fall 2023 | Group 3 | September 12th, 2023

Members: Isaac Harris, Yonatan Teklu, Natania Tedla, Leah Cheong, Amrith Balaguhan

SECTION 1:

Assignee Name	Email	Task	Duration (hours)	Dependency	Due Date	Evaluation
Natania Tedla	ntedla3 @student.gsu.edu	Use Cases/System Requirements		Problem statement	9/20/23	100%
Yonatan Teklu	yteklu1@student.gsu.edu	Database Specs, System Modeling		System Requirements Database specs	9/28/23	100%
Leah Cheong	lcheong1@student.gsu.edu	Use Cases/System Requirements		Problem Statement	9/20/23	100%
Amrith Balaguhan	ebalaguhan1@student.gsu.edu	Problem Statement and Revisions		Original Problem Statement	9/16/23	100%
Isaac Harris (Coordinator)	iharris9@student.gsu.edu	Report, Database Specs, additional use cases		System Requirements	9/28/23	100%

SECTION 2: Revised Problem Statement

a. **What is your product, on a high level?**
   The product is a full e-commerce marketplace themed around allowing users to find, track, and purchase limited-run shoes while promoting the related social media pages.
b. **Whom is it for?**
   The product is a general single-seller e-commerce platform for anyone who wants to limit store access to an approved set of users.
c. **What problem does it solve?**
   It allows the client to formalize the e-commerce business and give it a public webpage, in addition to being a source for the client to advertise a Discord subscription.
d. **What alternatives are available?**
   eBay is an alternative, but does not offer features like tracking shoe launches. Amazon is an alternative, but does not allow the seller to direct users to a paid Discord server for discussion. Sellfy and Shopify both provide the tools to buy and sell, but are fully digital marketplaces without the ability to physically ship items.
e. **Why is this project compelling and worth developing?**

This project is worth developing because it will provide an organized, tailored space for our client to display the products that he procures for his business while also making it so he can directly and quickly complete transactions with his clientele.

f. **Describe the top-level objectives, differentiators, target customers, and scope of your product.**
The top-level objectives are to deliver a webpage that the client can offer to customers to buy products directly, and simultaneously advertise a Discord server that would give the user insight into what shoes are going up on sale, among other things. The incorporation of this Discord server is a differentiator in this product, as it is not a usual feature of many e-commerce websites. The target customers are people who are looking to buy limited-run shoes at prices that are cheaper relative to the market.

g. **What are the competitors and what is novel in your approach?**
The competitors are other shoe sellers with the same objectives as primetime LLC, which is to have a marketplace to sell shoes to any wanting customers. Many of these sellers do not have very technical systems and it makes for questionability on the buyer if they should trust these sellers. Our approach is to make the company website something that feels very trustworthy and as official as any other LLC.

h. **Make it clear that the system can be built, making good use of the available resources and technology.**
Because there are many e-commerce websites on the Internet, we have a better and broader idea of what to incorporate into our project as well as being able to add qualities that would make our project more unique than the ones that are available online. Adding a Discord server to our site is taking advantage of the communication resources that are already available to us as well as using functions such as bots to make things liek URL scraping and tracking URL changes much easier and more accessible.

i. **What is interesting about this project from a technical point of view?**
This project is interesting from a technical point of view in how it incorporates both sellers and buyers in the same platform for the sale of shoes, which is something that will be accomplished through the service's backend using a database language, such as SQL via SQLite.

j. **Do you have a client login and an admin login?**
Yes. Clients will be able to log in through a traditional user+pass combination or sign in a third-party sign-on service not yet decided upon (likely Discord or Google). The admin will use the same login mechanism.

SECTION 3: Use Cases and System Requirements

Use Case 1

Use Case Name: Browse website

Actors: Customer

Description: User has the ability to navigate across the e-commerce website to look at the wide variety of shoes available for purchase. They can filter the shoes down to the color and style of shoes to their preference to allow for easier browsing.

Exceptional path: Customer may get an error result or "no results found" when searching up a specific shoe.

Alternate Path: Instead of searching for a specific shoe, they can use filtering that will fit the criteria for the desired shoe.

Pre-Condition: Users must have access to the website such as logging in as a guest or making an account.

Post-Condition: Users may continue to look for more shoes that are listed on the webpage.

System Requirements:

Requirement: 1.1
Use Case: 1
Name: Shoe Catalog
Introduction: The Shoe Catalog is a core component of the e-commerce website, providing users (Customers) with the ability to browse and search for a wide variety of shoes available for purchase.
Rationale: It is essential that the e-commerce website functions as intended and provides a positive user experience, which leads to increased sales and customer satisfaction.
Input: User input on filtering criteria.
Requirement Description: The catalog should support filtering options based on various attributes such as color and style, allowing users to refine their search along with displaying basic information about the shoe such as name and price.
Output: A refined catalog of shoes that the user's filtering.
Test Cases: TBD

Requirement: 1.2
Use Case: 1
Name: Search Query

Introduction: Requirement allows users to search for their specified shoe based on their desired criteria.
Rationale: It allows users to be able to quickly find the shoes they are looking for in a quick and user-friendly manner.
Input: Search query
Requirement Description: The system provides a feature for the users that'll easily allow them to enter a search query for their specified shoe and then search for that shoe. If available in the shoe catalog then the system will return a successful result to the user.
Output: The search result will return whatever it is that the user searched for and if the particular shoe is in stock or not.
Test Cases: TBD

Use Case 2

Use Case Name: View detail of shoes

Description: When a customer would like to view a description on the selected shoe, they can choose to see it by clicking on the "View Details". Here it will show the style of shoe as well as the color.

Actors: Customer, Product database

Exceptional path: There could be a chance that a description is missing for the shoe.

Alternate Path: Instead of going to look at the detail of the shoes, the customer may just want to choose to buy it without checking product details.

Pre-Condition: The display of the shoes should be listed.

Post-Conditions: After the product details are displayed, customers will be able to see the description of the shoes.

System Requirements:
Requirement: 2.1
Use Case: 2
Name: View shoe details
Introduction: The "View Shoe Details" feature allows customers to access and view detailed information about a selected shoe product.
Rationale: This requirement is essential to provide customers with the information they need to make informed purchasing decisions and ensure a positive user experience.

Input: A “view details” button under the shoe catalog that is clicked by the user.
Requirement Description: when the user clicks on the "View Details" button for a specific shoe, the system should retrieve and display detailed information about that shoe.
Output: An expanded view/window of the shoe that was clicked that includes detailed information about the shoe that is displayed after being retrieved from the database, such as shoe size, price, how many are left in stock, etc.
Test Cases: TBD

Use Case 3

Use Case Name: Add to Cart

Description: Customers will have the option to add shoes they would potentially like to buy to a shopping cart. They can use the cart as a way to store their shoes as they continue to shop through the e-commerce website. Once they are ready to purchase, they can press the shopping cart icon.

Actors: Customer, Product Database

Exceptional path: If the product is unavailable then users will not be allowed to add it to their cart.

Alternate Path: Instead of users adding a product to cart, they can choose to buy it immediately instead and go straight to checkout with that one item.

Pre-Condition: An item has to be available in order to add to the cart.

Post-Condition: An item will be added to their cart and stored there until the customer is ready to decide if they would want to purchase the items.

System Requirements:
Requirement: 3.1
Use Case: 3
Name: Adding item(s) to cart
Introduction: The system will provide users with a shopping cart where they can add shoes that they wish to purchase.
Rationale: This requirement is a significant aspect that gives users the option to add all their chosen items into one place and view them before deciding to purchase them
Input: Shoes currently in the shopping cart.

Requirement Description: The system should be able to store all of the user's selected items into the cart. System should update the cart and save new items that are being added. Adding new items in the cart should not affect items that were previously in the cart.
Output: There should be a numerical display on a shopping cart of updated items added to the shopping cart
Test Cases: TBD

Use Case 4

Use Case Name: Remove from cart

Description: If a customer no longer chooses to purchase a pair of shoes, they can remove the selected shoes from their shopping cart while keeping the ones they would still like to purchase in the cart.

Actors: Customer, Product Database

Exceptional path: The shopping cart database attempts to remove items from the cart for the customer but fails to do so and it shows an error or “try again later” for the user.

Alternate Path: Customer is given a confirmation to whether or not they would like to remove items from the shopping cart.

Pre-Condition: There must be items in the cart for a customer to be able to remove from the cart.

Post-Condition: Once a customer chooses the selected item to be removed, the shopping cart database will remove that item from cart.

System Requirements:
Requirement: 4.1
Use Case: 4
Name: Removing item(s) from cart
Introduction: The system will allow users to remove items out of their cart whether it is the item all together or reducing quantity.
Rationale: This essential requirement gives users the ability to edit their cart and remove items they would no longer like to purchase from it.
Input: Items currently in cart, remove functionality
Requirement Description: The users should have full access to their shopping cart that allows them to view the shoe and its quantity and style. They should be able to remove one

or more items in their cart. System should be able to make those changes in the cart by removing only the selected items to remove from the user's cart.
Output: Only the selected items that were chosen to be removed should no longer be stored in the shopping cart.
Test Cases: TBD

Use Case 5

Use Case Name: Checkout Process

Actors: Customer, Product Database, Payment gateway

Description: Customers will be taken to a payment system where they will be able to confirm and finalize their order. They can choose their preferred method of payment here and be given details on their order such as when the product will be shipped out.

Exceptional path: An error between the customer and payment gateway where the payment failed to process. A message pops up to let customers know that they have to try again.

Alternate Path: Customers will be given multiple methods of payment and once they choose to their liking, the system will lead the customers to where they can input their payment information.

Pre-Condition: There must be items in the customer's shopping cart that they have chosen to purchase

Post-Condition: Customers will be given a confirmation that their payment has successfully went through

System Requirements:
Requirement: 5.1
Use Case: 5
Name: Choosing method of payment
Introduction: The users will need to purchase their items through a checking out process where they'll be able to pay using their preferred method of payment.
Rationale: Items have to be checked out through a manageable process so users will be able to purchase their items.
Input: Preferred method of payment, confirmation of payment

Requirement Description: When items are being checked out, the method of payment must be chosen, and after it's chosen, it's essential that all required fields are filled out so that payment can be processed and finalized.
Output: All required fields are filled, accepted, and confirmed.
Test Cases: TBD

Use Case 6

Use Case Name: Discord Subscription Sign-up

Actors: Customer

Description: Customers will be given the option to sign up for a subscription that'll be connected to their Discord account. In this process, they will have to pay for their subscription to be given the discord server link where paying customers will be granted exclusive access to new shoe releases and an opportunity to be part of a large community looking to sell and buy shoes.

Exceptional path: If a user attempts to sign into their discord account but fails to put the wrong username and/or password, it will ask them to re-enter it.

Alternate Path: The user may choose to reject the subscription offer.

Pre-Condition: The user should have an existing Discord account.

Post-Condition: The user will receive access to enter into the paid Discord server.

System Requirements:
Requirement: 6.1
Use Case: 6
Name: Subscription Sign Up Process
Introduction: The system enables users to sign up for Prime Time subscription using the user's Discord account.
Rationale: The requirement is essential as it gives customers the option to access exclusive benefits and enhance customer experience.
Input: Customer Discord account information (email/phone number and password), subscription payment.
Requirement Description: The system will grant a user-friendly process that'll easily transition customers from Prime time to Discord once the customer correctly inputs their account information and pays for their subscription service.

Output: Users will have full access to their Discord subscription.
Test Cases: TBD

Use Case 7

Use Case Name: Managing Inventory

Actors: Seller, Product Database, Log In System

Description: The seller can operate and update the shoe inventory needed for the website. This includes when the seller needs to consider new stocks that need to be added or removed from the website.

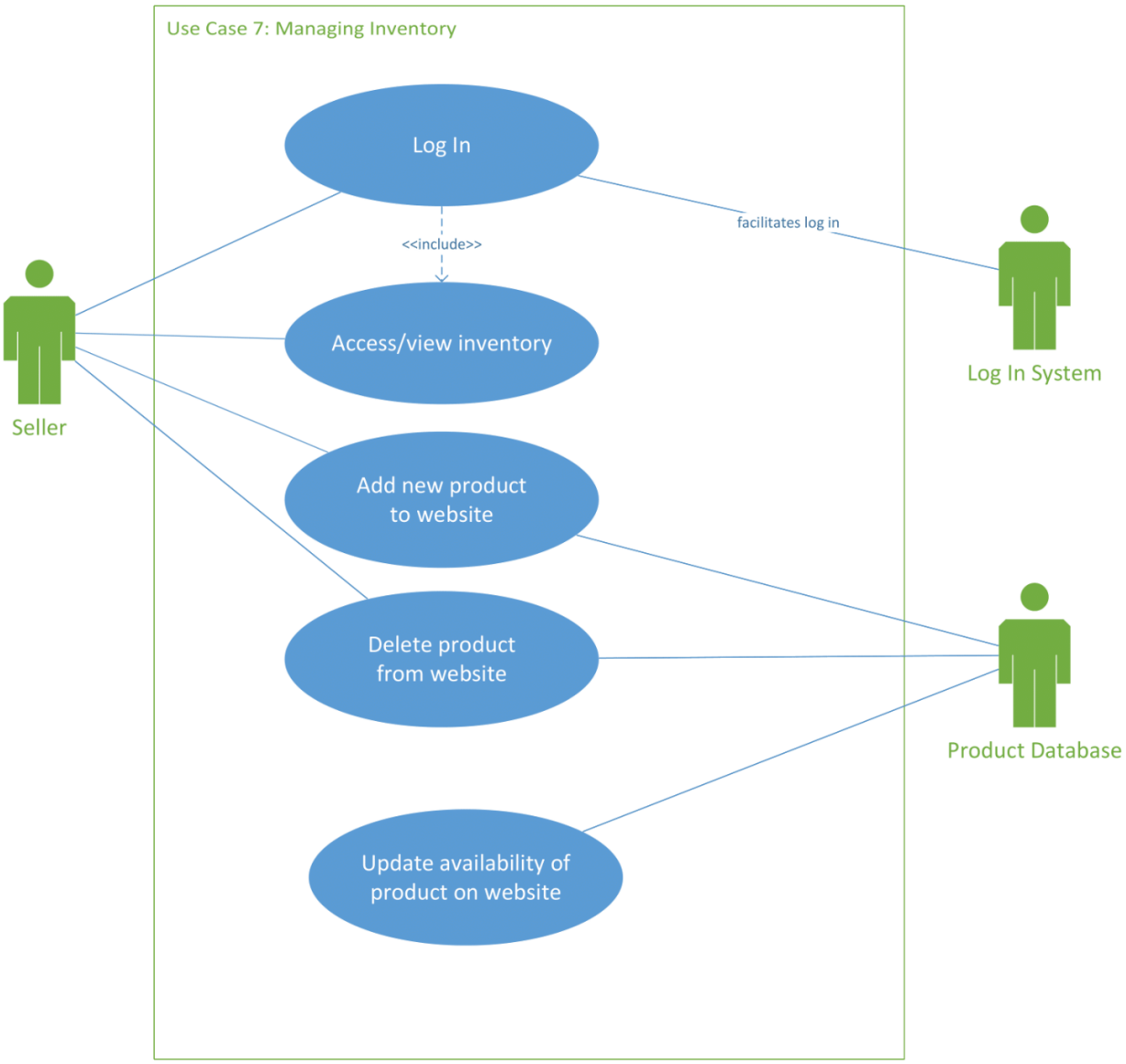
Exceptional path: The product of the shoe is not available in the database and system can't update the inventory of that specific shoe.

Alternate Path: System can keep an inventory history and update it whenever a shoe is purchased to a customer. That way the seller can keep track and manage the inventory this way.

Pre-Condition: Seller has authorization to access inventory. Seller must log in to verify themself as someone who has access to the inventory.

Post-Conditions: Inventory has been successfully up to date.

System Requirements:
Requirement: 7.1
Use Case: 1
Name: Adding New Product into Inventory
Introduction: This requirement indicates the functionality that the system should exhibit when the seller adds a new product to the inventory.
Rationale: It ensures that the system allows the seller to efficiently add products to the product database and keep a steady catalog of up-to-date shoes
Input: Product detail, Stock quantity
Requirement Description: The system should add the new product to the product database and provide all the product details including shoe name, size, quantity. It should then create a unique identifier for the new product, and then finally update the catalog.
Output: The new product will officially be added to the shoe inventory with the rest of the products that the website sells.
Test Cases: TBD


Use Case 7: Managing Inventory
Log In
facilitates log in
<<include>>
Access/view inventory
Add new product to website
Delete product from website
Update availability of product on website
Seller
Log In System
Product Database

Use Case 8

Use Case Name: Scrape upcoming shoe releases

Actors: Seller, Product Database

Description: The seller adds an entry to the database that contains info for a shoe listing, including release date (or speculated, for all info), price, a picture url, and optionally a url that will have info about it as release closes in. The program then can scrape that link for updates automatically.

Exceptional path: No URL can be provided, so an info-only post is made without any scraping or data update features.

Alternate Path: The seller uses a webhook and the site displays the latest posts from it rather than scraping the data.

Pre-Condition: A new shoe is being released, the backend has access to the database, and the seller is signed into an account and on the seller dashboard.

Post-Conditions: When the seller adds an entry to the database, the necessary information is successfully scraped from the URL of that shoe listing and updated for the customers to see.

System Requirements:
Requirement: 8.1
Use Case: 8
Name: Scrape for text in an HTML document or loaded webpage
Introduction: This requirement is necessary to give users a feed to see upcoming releases, like a watchlist. The shoe data can be gathered from other sites if the seller chooses.
Rationale: This feature will allow sellers to increase storefront interest.
Input: URL, full selector for resource
Requirement Description: To execute this, BeautifulSoup (Jsoup for Java) will connect to the webpage with Jsoup.connect(URL). Next, the given selector will be used in Jsoup (for example, document.select("#comment-4325091_wrap > div.cPost_contentWrap > div.ipsType_normal.ipsType_richText.ipsPadding_bottom.ipsContained > p:nth-child(2)"))
Output: For this link, https://stackoverflow.com/questions/16335820/convert-xpath-to-jsoup-query the output for path #question-header > h1 > a should be "Convert xPath to JSoup query"

Test Cases: Provide five websites with selector paths and return the text within that element. Test passes if content in the events table where source is identical to the given URL and content is identical to the given text.

System Requirements:
Requirement: 8.2
Use Case: 8
Name: Watch for changes on a remote page
Introduction: This requirement is linked with requirement 8
Rationale: This requirement is linked with requirement 8
Input: URL, full selector for resource
Requirement Description: Jsoup will scrape from the page given in requirement 8 at the same selector. The data will be fetched from the latest timestamp where source is equal to URL. If the content or title is different, store the new one.
Output: **NOTE: Jsoup does not support Javascript, so this exact output is impossible. This is solely for example.**

For this link, https://stackoverflow.com/questions/16335820/convert-xpath-to-jsoup-query and the selector "#question > div.post-layout > div.votecell.post-layout--left > div > div.js-vote-count.flex--item.d-flex.fd-column.ai-center.fc-theme-body-font.fw-bold.fs-subheading.py4," the current votes on the question is 20. The backend will check the data at this URL and selector every (frequency) hours. If the content of this element is no longer 20, it will create a new entry into the events table, with the current timestamp, identical URL, but new content.

Test Cases: The backend's health check page can be made as a template for Spring that updates on every refresh. Scrape it twice. If there are now two entries, one with a later timestamp (both in the table and in the content), pass. Then, delete the two entries so they aren't shown to the user. **TODO: Maybe keep a testing table to do this in instead**

Use Case 9

Use Case Name: Output Events as Webhook

Actors: Discord server channel, backend

Description: Upon detecting a change in a tracked URL, the change is posted to webhooks that have been added to the service. This is so the customers that use the website can stay up

to date with the changes occurring on the e-commerce website without having to check themselves.

Exceptional path: If there are no webhooks that are configured, nothing is sent.

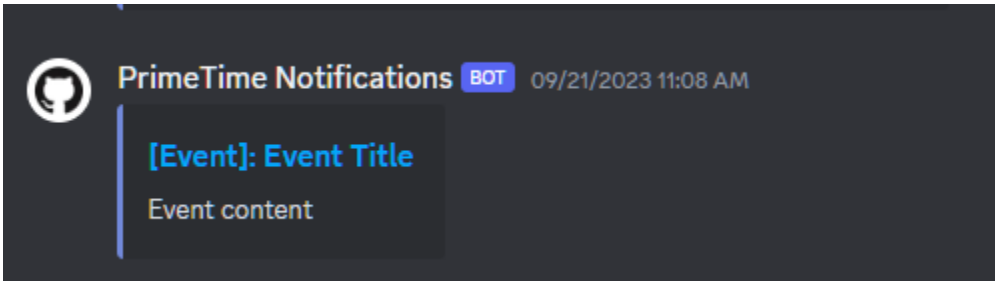
Alternate Path: The backend doesn't detect a change in the tracked URL, so no data payload is generated to send to the webhooks.

Pre-Condition: A tracked URL must go through a change.

Post-Conditions: Changes detected in the tracked URL are received by the webhooks and displayed in the Discord server channel so that server members can be aware of the change.

System Requirements:
Requirement: 9.1
Use Case: 9
Name: Post data to a webhook
Introduction: The system needs to notify users in the Discord server of updates that happen on a tracked URL, as an alternative to going to the site and reading the feed
Rationale: This provides users more ways to stay up to date with products that sell out fast
Input: Webhook URLs, tracking table must have URLs and a change must have happened since the last one
Requirement Description: The system sends a POST request to the Discord webhook using a created webhook class. The class will send a request in the format '{"username": "PrimeTime Notifications", "content": "null", "embeds": [{"title": "[Event]: Event Title", "description": "Event content"}]}'

Output:



Test Cases: Human verifies webhook contents succeeding test for use case 8

Use Case 10

Use Case Name: Customer Reviews

Actors: Customer

Description: Customers can submit their reviews based on their satisfaction with their purchases.

Exceptional path: If a customer submits an invalid response or doesn't complete a required field then it may not go through.

Alternate Path: Customers can discuss their feelings about the product they purchased on other platforms in a more informal way, such as on social media or in person.

Pre-Condition: The customer must have bought the product in question before submitting a review in order to maintain the veracity of the reviews. Customers who haven't bought a certain product will not have the ability or option to leave a review on that product's detail page.

Post-Conditions: The customers' reviews will be visible under teh product's detail section, allowing other customers to read over their satisfaction with their purchases before deciding to also buy this product.

System Requirements:
Requirement: 10.1
Use Case: 10
Name: Review System
Introduction: The review system allows customers to submit their reviews based on their satisfaction with their purchases. These reviews provide valuable feedback for other potential customers and help in maintaining the credibility of product reviews.
Rationale: This system is essential for promoting transparency, trustworthiness, and customer engagement on the ecommerce platform.
Input: User typed review and rating
Requirement Description: Text box that users can type into along with a rating system that lets the user choose the value (5-star system). The reviews need to be public so other users can see the reviews.
Output: Display of customer reviews and ratings on the product detail page
Test Cases: TBD

Use Case 11

Use Case Name: Wishlist

Actors: Customers, Product Database, Log In System, User Database

Description: Customers can choose to create a wishlist of shoes that they can save for later and purchase in the future. Customers can view this list at any time as well as add to or delete from it as they see fit.

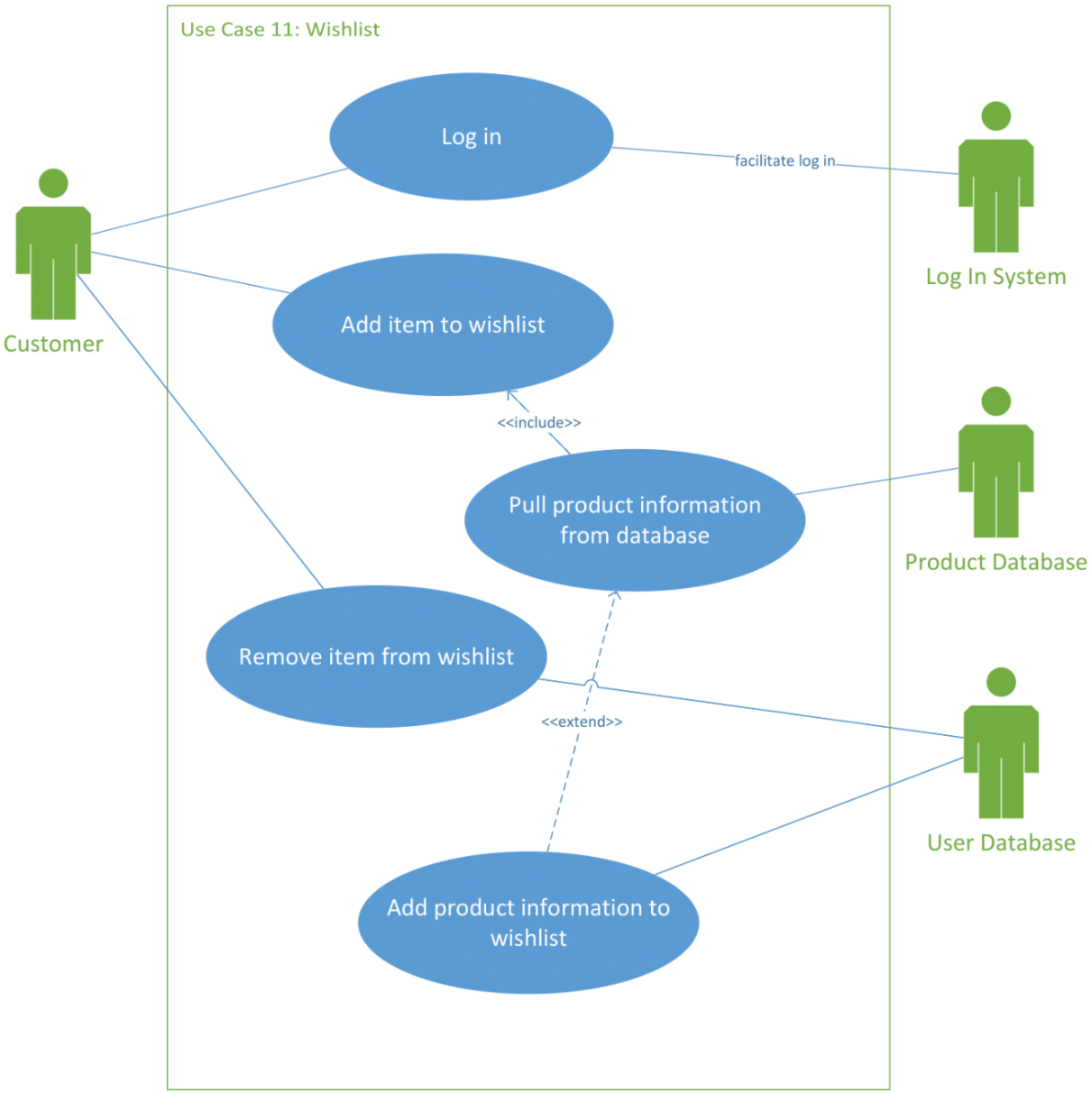
Exceptional path: Item cannot be added to wishlist because it's already in the wishlist duplicates are prevented

Alternate Path: Customers can save URLs for each product they want to buy manually if they do not wish to make an account to make a wish list

Pre-Condition: Customers must have an account that the saved items on the wishlist can be associated with and saved to. Items can only be added to a wishlist if the customer has an account and is currently logged into it.

Post-Conditions: The items added to the wishlist will stay stored in the database as a part of that user's wishlist until they buy the item or decide to remove it from the wishlist themselves.

System Requirements:
Requirement: 11.1
Use Case: 11
Name: Adding to Wishlist
Introduction: The system will provide users with the option to add a product to their personal wishlist
Rationale: This requirement is essential for customer engagement and retention because it allows user to save and revisit products whenever they want
Input: User clicked an add to wishlist prompt
Requirement Description: Having an "add to wishlist" button that can also remove an item if clicked on something already on the wishlist
Output: A list of products that the user added to the wishlist
Test Cases: TBD


Use Case 11: Wishlist
Log in
facilitate log in
Log In System
Customer
Add item to wishlist
<<include>>
Pull product information from database
Product Database
Remove item from wishlist
<<extend>>
User Database
Add product information to wishlist

Use Case 12

Use Case Name: Contact Seller/Customer Support

Actors: Customers, Seller

Description: This use case is for customer assistance where customers can contact the seller for issues pertaining to their order on the website. Issues may be related to the quality of product or product delivery.

Exceptional path: The seller left an invalid form of contact and cannot be reached (i.e. invalid number), making contacting the seller impossible.

Alternate Path: Customers may reach sellers through a social media platform, such as Instagram or Facebook.

Pre-Condition: Customer has an order from the company that they have a particular issue with. Order number must be included so that the specific transaction can be discussed.

Post-Conditions: The seller is able to reach out within an appropriate timespan to solve the customer's inquiry. The seller and customer discuss the issue and come to a solution through their discussion.

System Requirements:
Requirement: 12.1
Use Case: 12
Name: Seller contact
Introduction: The seller contact system allows the user to communicate directly to the seller regarding the product and any questions or issues the user might have.
Rationale: This requirement is important to resolve issues and maintain customer satisfaction
Input: Customer's inquiry, order details, contact information
Requirement Description: The system should provide a secure and reliable communication channel for customers to contact the seller while also keeping information private.
Output: A messaging thread that allows the user and seller to send messages in real time to each other.
Test Cases: TBD

SECTION 5:

Database Specification

Database Management System: SQLite3 (Note: Must use https://docs.oracle.com/javase/7/docs/api/java/sql/PreparedStatement.html for Java, else rely on library-specific sanitation for other languages. All TIMESTAMP values must be in the ISO-8601 format for SQLite)

Parent Tables:

1. Account table:

User ID	First Name	Last Name	Email	Address	Phone Number

id UUID NOT NULL PRIMARY KEY, email TEXT NOT NULL, pass VARCHAR(72) (bcrypt hashed+salted length), addresses TEXT[], phone_no TEXT[]

2. Item table:

Item ID	Name	Images	Price	Description	Tags	Seller ID

id SERIAL PRIMARY KEY, name TEXT NOT NULL, thumbnail TEXT, listing_images TEXT[], price FLOAT, description TEXT, tags TEXT[], seller_id TEXT[], sku TEXT

Tags could either be text, or a JSON-like structure for filtering: {“brand”: “Nike”, “Size”: “10”, “color”: “blue”, “releaseYear”: “2023”, etc.}

3. News/Events:

Time	Title	Content	Thumbnail	Source

time TIMESTAMP, title TEXT NOT NULL, content TEXT, thumbnail TEXT, source TEXT

Child Tables:

1.Customer table:

User ID	First name	Last Name	Cart	Preferences

id UUID NOT NULL FOREIGN KEY, first_name TEXT NOT NULL, last_name TEXT, cart_items INTEGER[][], preferences TEXT[]

2.Listings table:

Item ID	Name	Thumbnail	Price	Tags

id FOREIGN KEY, name TEXT, thumbnail TEXT, price INTEGER, tags TEXT[]

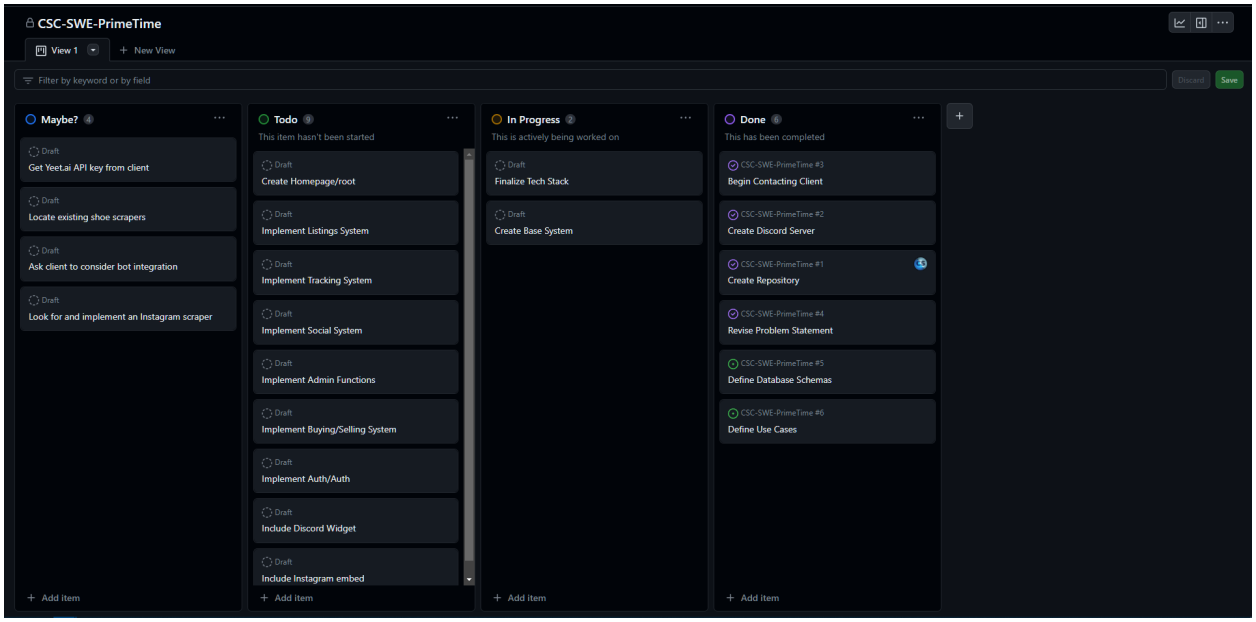
3. Tracking:

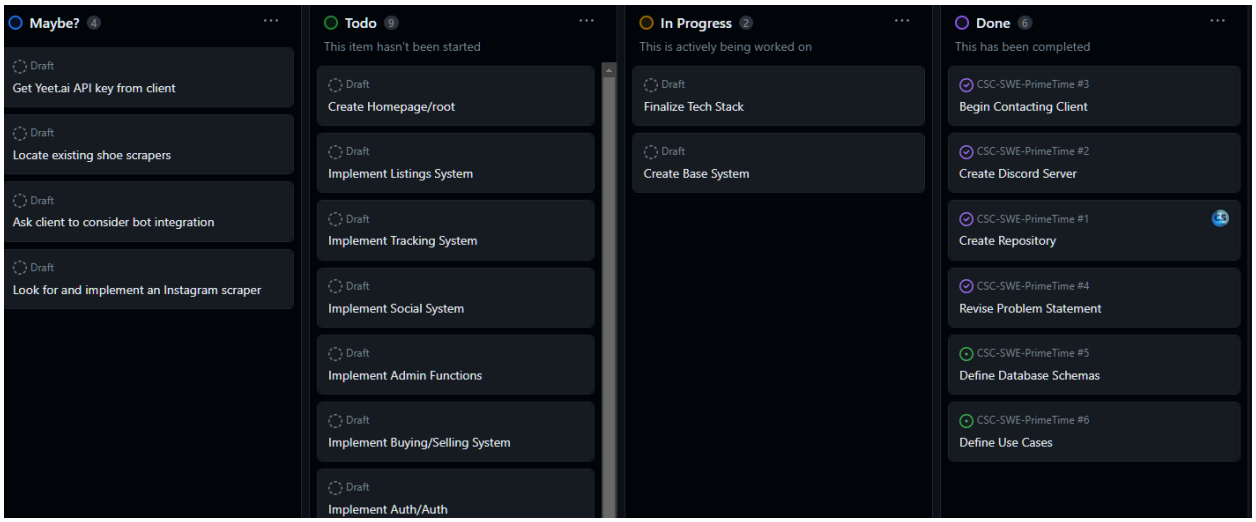
URL	Frequency	Selector

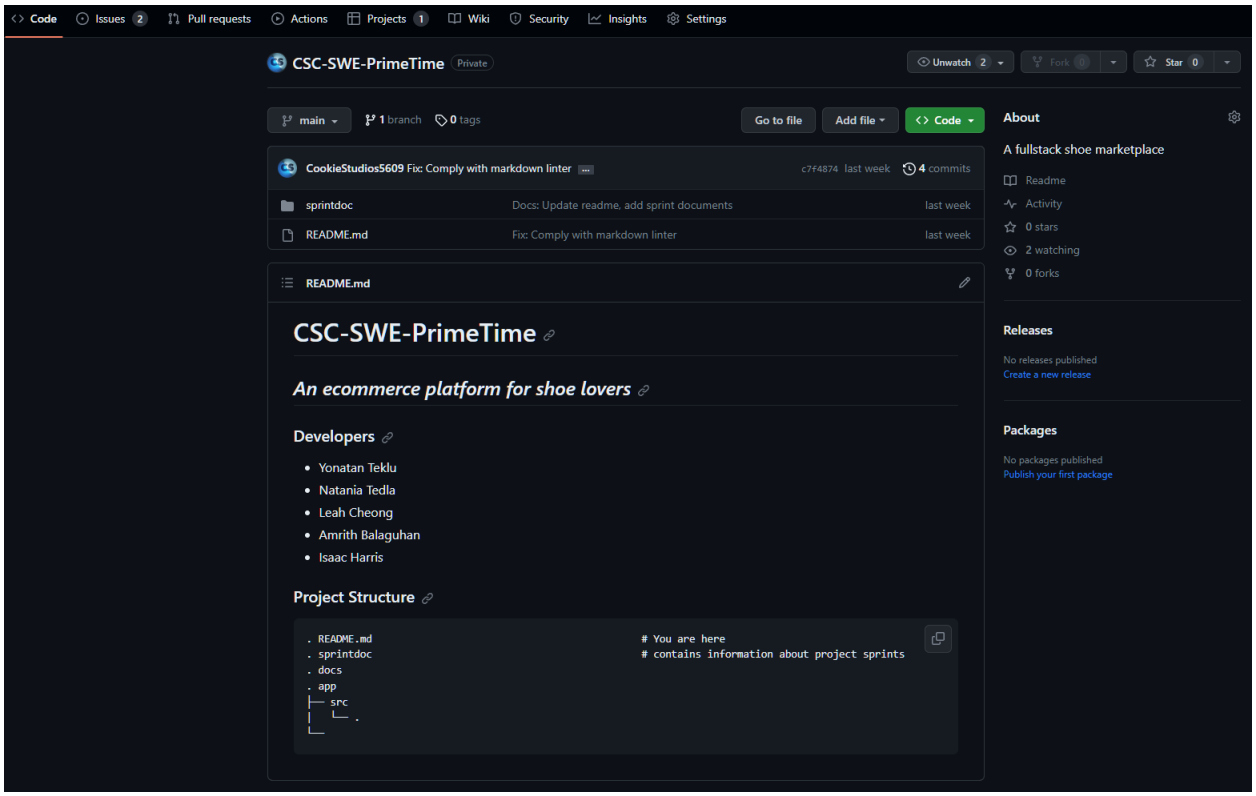
url TEXT, frequency INTEGER, selector TEXT (frequency stored in hours maybe?)

SECTION 6: Appendix

GitHub Screen Shots


CSC-SWE-PrimeTime
View 1
New View
Filter by keyword or by field
Discard
Save
Maybe? 4
Draft
Get YeetAi API key from client
Draft
Locate existing shoe scrapers
Draft
Ask client to consider bot integration
Draft
Look for and implement an Instagram scraper
Add item
Todo 9
This item hasn't been started
Draft
Create Homepage/root
Draft
Implement Listings System
Draft
Implement Tracking System
Draft
Implement Social System
Draft
Implement Admin Functions
Draft
Implement Buying/Selling System
Draft
Implement Auth/Auth
Draft
Include Discord Widget
Draft
Include Instagram embed
Add item
In Progress 2
This is actively being worked on
Draft
Finalize Tech Stack
Draft
Create Base System
Add item
Done 6
This has been completed
CSC-SWE-PrimeTime #3
Begin Contacting Client
CSC-SWE-PrimeTime #2
Create Discord Server
CSC-SWE-PrimeTime #1
Create Repository
CSC-SWE-PrimeTime #4
Revise Problem Statement
CSC-SWE-PrimeTime #5
Define Database Schemas
CSC-SWE-PrimeTime #6
Define Use Cases
Add item


Maybe? 4
Draft
Get Yeet.ai API key from client
Draft
Locate existing shoe scrapers
Draft
Ask client to consider bot integration
Draft
Look for and implement an Instagram scraper
Todo 9
This item hasn't been started
Draft
Create Homepage/root
Draft
Implement Listings System
Draft
Implement Tracking System
Draft
Implement Social System
Draft
Implement Admin Functions
Draft
Implement Buying/Selling System
Draft
Implement Auth/Auth
In Progress 2
This is actively being worked on
Draft
Finalize Tech Stack
Draft
Create Base System
Done 6
This has been completed
CSC-SWE-PrimeTime #3
Begin Contacting Client
CSC-SWE-PrimeTime #2
Create Discord Server
CSC-SWE-PrimeTime #1
Create Repository
CSC-SWE-PrimeTime #4
Revise Problem Statement
CSC-SWE-PrimeTime #5
Define Database Schemas
CSC-SWE-PrimeTime #6
Define Use Cases


Code
Issues 2
Pull requests
Actions
Projects 1
Wiki
Security
Insights
Settings
CSC-SWE-PrimeTime
Private
Unwatch 2
Fork 0
Star 0
main
1 branch
0 tags
Go to file
Add file
Code
CookieStudios5609 Fix: Comply with markdown linter
c7f4874 last week
4 commits
sprintdoc
Docs: Update readme, add sprint documents
last week
README.md
Fix: Comply with markdown linter
last week
README.md
CSC-SWE-PrimeTime
An ecommerce platform for shoe lovers
Developers
Yonatan Teklu
Natania Tedla
Leah Cheong
Amrith Balaguhan
Isaac Harris
Project Structure
. README.md # You are here
. sprintdoc # contains information about project sprints
. docs
. app
├── src
│ └── .
└──
About
A fullstack shoe marketplace
Readme
Activity
0 stars
2 watching
0 forks
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package