CS3354 Software Engineering
Final Project Deliverable 2

Team Syrup

Daniel Pulido
Paul Collins
Nirranjan Akilan
Nikhil Dasari
Brandon Coffey
Scott Pedley
Monish Ravishankar

1. Well described delegation of tasks, i.e. who did what in the project

Nirranjan:
- Major contributor to creating the software requirements.
- Overall contributor to designing the architectural design diagram.
- Responsible for submitting to elearning (including GitHub URL).
- Contributed to slides & Cost, Effort, and Estimation Price calculations.
- Main contributor to Unit Testing

Monish:
- Overall contributor to the modeling process, responsible for the Class Diagram.
- Overall contributor to the report, specifically organization and writing.
- Responsible for making an additional commit about project scope .

Daniel:
- Overall contributor to the sequence diagram.
- Overall contributor to the conclusion

Paul:
- Assist Monish with documentation of report and editing.
- Responsible for facilitating architectural design pattern diagrams.
- Responsible for creating the live demo.

Brandon:
- Create a GitHub Repository and add all team members and the TA as collaborators.
- Overall contributor to the presentation slides.
- Contributor to the report organization and references.

Nikhil:
- Responsible for creating the Use Case diagram(s) and contributed to the Sequence diagrams.
- Assisted Paul with live demo creation.
- Estimated project scheduling with FP calculations.

Scott:
- Software process model contributor.
- Contribute to setting up the GitHub.
- Estimation of cost for project
- Contributed to conclusion

2. Everything required and already submitted in Final Project Deliverable 1

Project Deliverable 1 Content:

1) Project Description

Draft Description: We will design an app to save users money by displaying the price history of an item to help find the lowest cost of a product. This app will use various vendors to compare the price of the desired product, letting the user know if the product is above or below the Manufacturer's Suggested Retail Price (MSRP).

Instructor Feedback: "A lovely topic!! Once complete, it will save a lot of time and effort for those who are in dire need to find an item at the best price possible. Please consider implementing it fully, if you can. No pressure w.r.t. grade on implementation. In the final report, please make sure to include comparison with similar applications -if any-, make sure that you differentiate your design from those, and explicitly specify how. Fair delegation of tasks. Please share this feedback with your group members. You are good to go. Have fun with the project and hope everyone enjoys the collaboration."

As per our instructor's feedback, we will research other applications that share a similar niche with our topic to draw inspiration for features to ensure that our application is unique. This will help us get familiar with our "competitors" should we consider making the full implementation.

2) GitHub Repository

https://github.com/BrandonMCoffey/3354-TeamSyrup

3) Individual Tasks

Nirranjan:
- Major contributor to creating the software requirements.
- Overall contributor to designing the architectural design diagram.
- Responsible for submitting to elearning (including GitHub URL).

Monish:
- Overall contributor to the modeling process, responsible for the Class Diagram.
- Overall contributor to the report, specifically organization and writing.
- Responsible for making an additional commit about project scope .

Daniel:
- Overall contributor to the sequence diagram.

Paul:
- Assist Monish with documentation of report and editing.
- Responsible for facilitating architectural design pattern diagrams.

Brandon:

- Create a GitHub Repository and add all team members and the TA as collaborators.
- Assist Daniel with the Sequence Diagram.

Nikhil:

- Making first commit to the repository (README).
- Responsible for creating the Use Case diagram(s).

Scott:

- Software process model contributor.
- Contribute to setting up the GitHub.
- Estimation of cost for project
- Contributed to conclusion

4) Software Development Model

Our group has decided to utilize the incremental process model as it allows small teams to set manageable deadlines while keeping our resources and expenditure in check. The incremental process model will allow us to start with simple features that we make more complex as time progresses. Our project's first increment would be retrieving price history information from APIs and store it in our application's backend since price data is the core aspect of our application. Our project's second increment would be designing our application's front end since this is where our users will interact with our service, and our goals are to make it easy to navigate, visually appealing, and support our required features. Our project's third increment would be connecting the front end to the backend so that our website isn't entirely static and actively retrieves & renders price history data from our database [1].

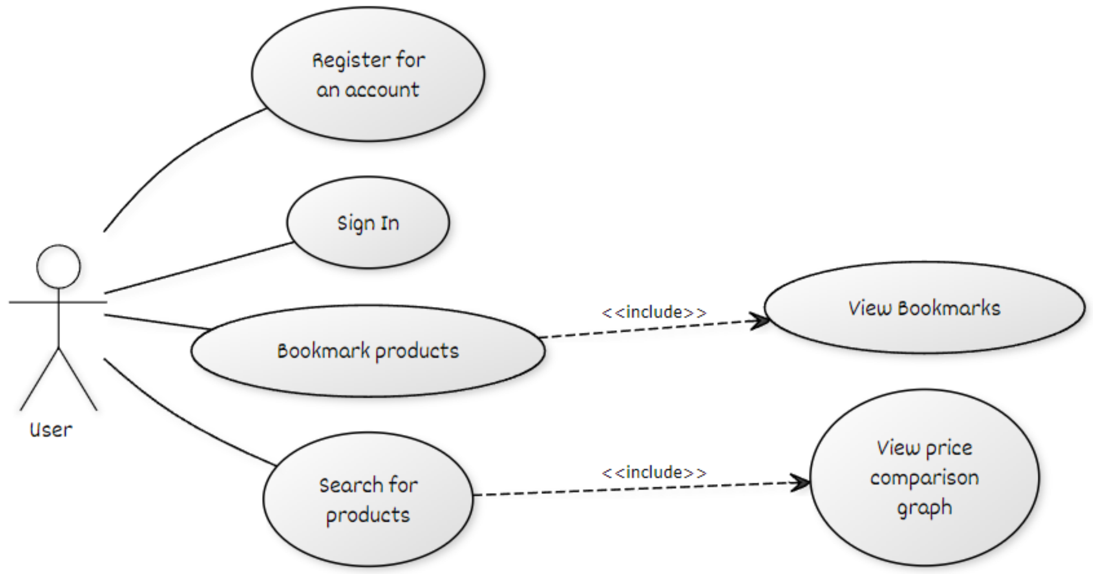
5) Functional Requirements [1]

1. API calls to get product information that we update once a day and track for a set time (ex: a year).
2. A search bar where the user can type in an item and click on it to go to its page
3. A query function to find the cost of the item from all the different shopping websites
4. Have functionality to graph a time vs price graph visual for users to view
5. Functionality to register/login in and save user information
6. Allow users to bookmark products to their account
7. Display bookmarked items on the user's homepage

Non-Functional Requirements [1]

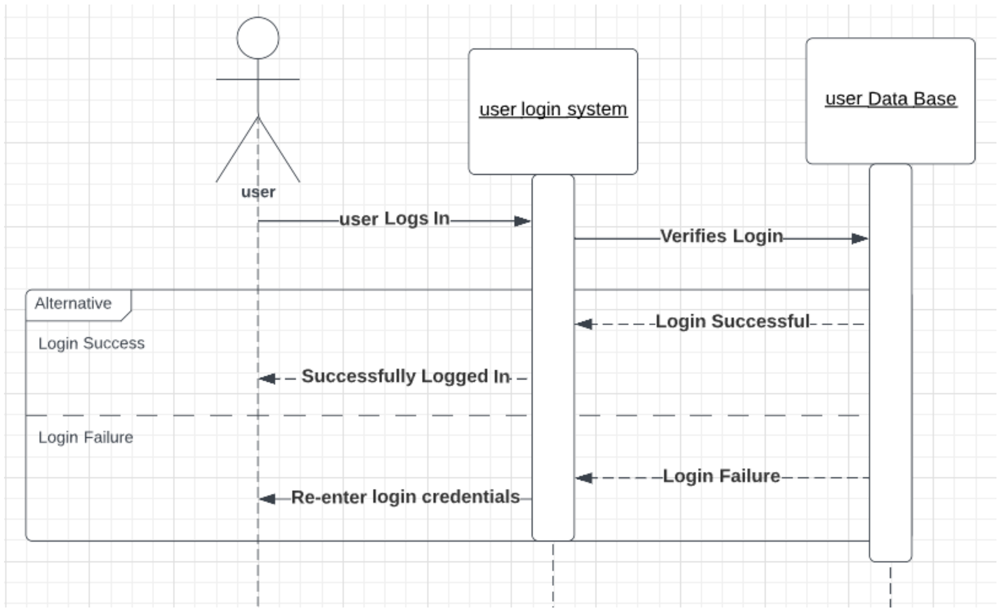
Usability requirement: Should be able to be used on desktop and mobile.
Performance requirement: All requests must be generated in 2 seconds or less.
Space requirement: No data should be stored on the user's computer.
Environmental requirement: The computer hosting the application's server must be located in a dust free environment per American dust laws about server maintenance.
Operational requirement: The application must be able to gather price history once a day and track it for a year.
Development requirement: The application must implement object-oriented programming practices to ensure that price data calculations are abstracted from the user.
Accounting requirement: The application's shareholders get a small commission for products purchased through the application.
Safety requirement: All user and cite data should be encrypted.

6) Use case diagram [2]

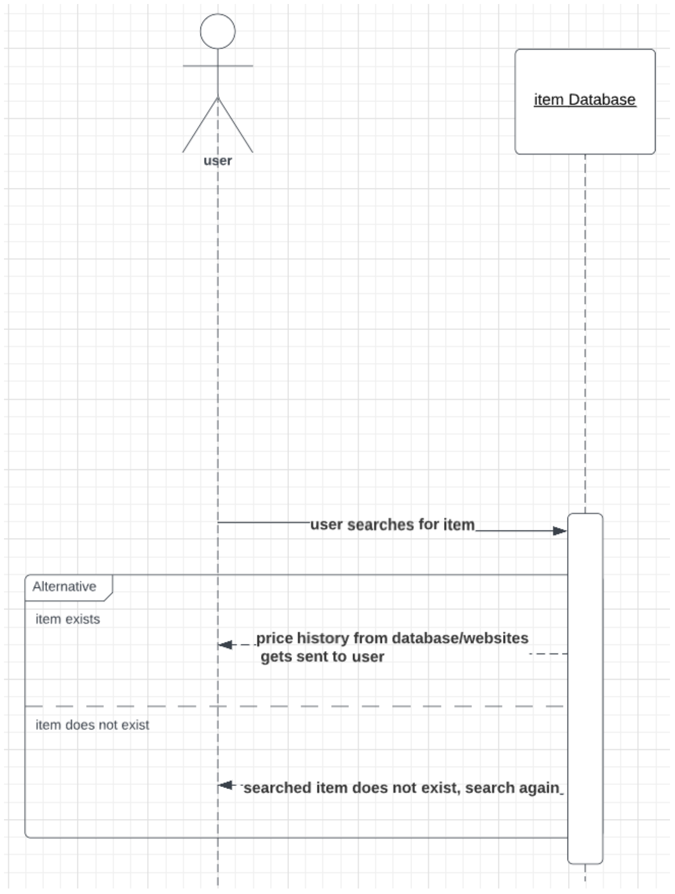


7) Sequence diagram

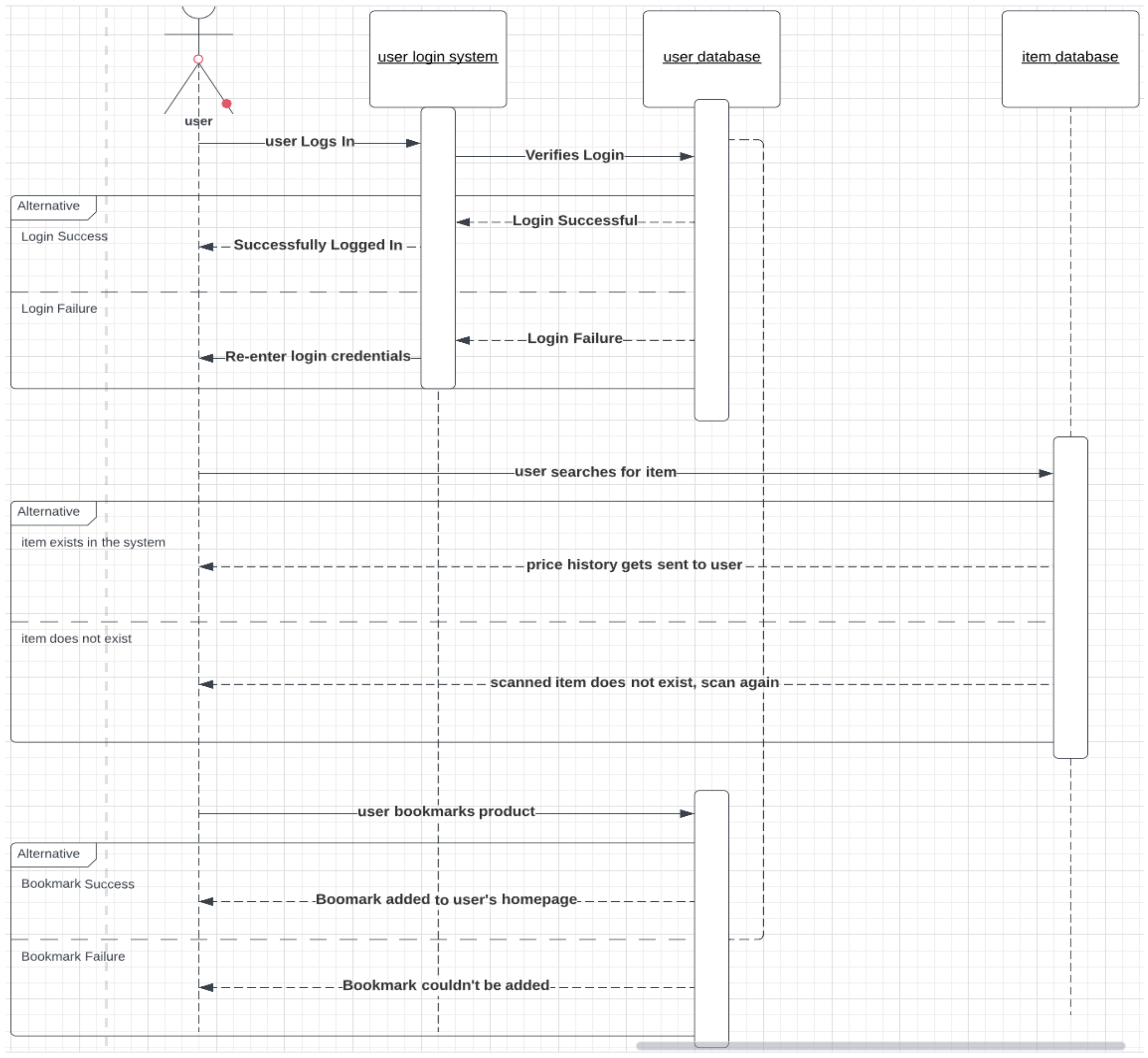
Login/Register:



Search for product:

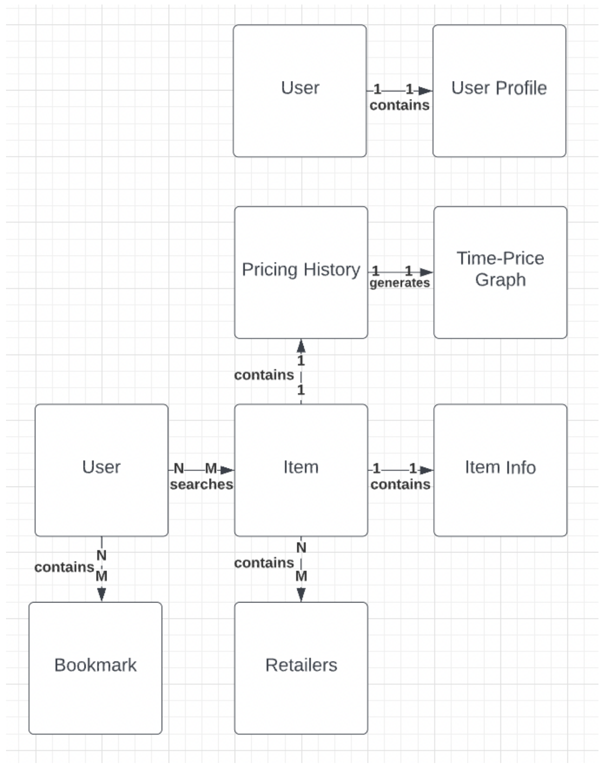

user
item Database
user searches for item
Alternative
item exists
price history from database/websites gets sent to user
item does not exist
searched item does not exist, search again

Bookmark Product (need to be logged in and have searched for an item)

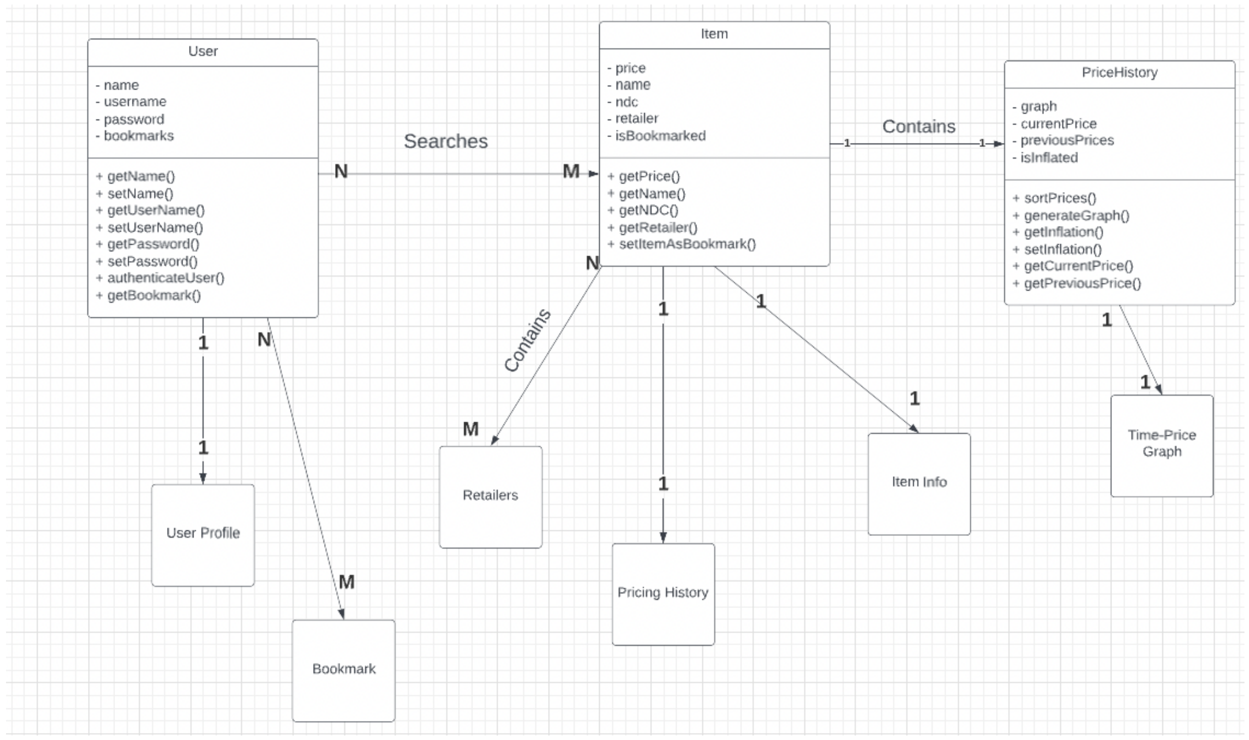


8) Class diagram [3]

Associations

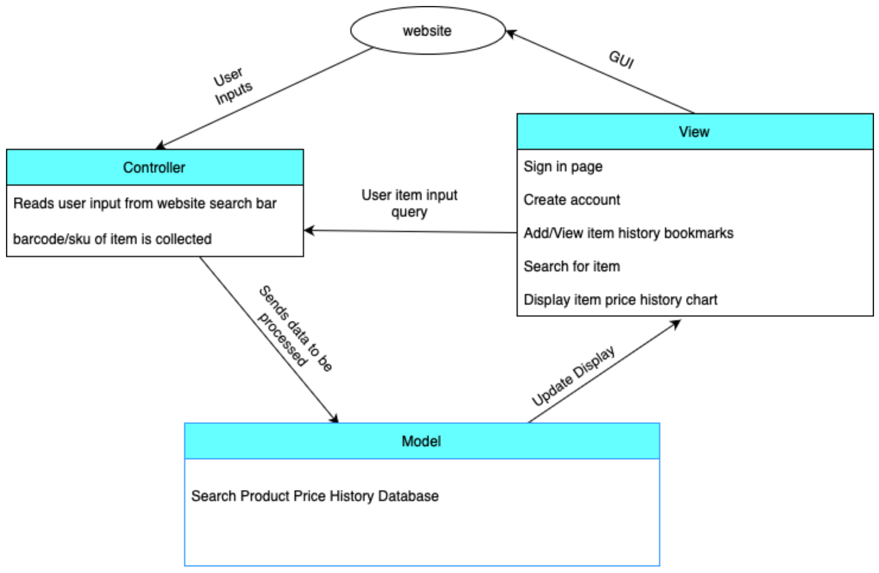


Classes



9) Architectural design

Model-View-Controller (MVC) pattern


website
User Inputs
GUI
View
Sign in page
Create account
Add/View item history bookmarks
Search for item
Display item price history chart
Controller
Reads user input from website search bar
barcode/sku of item is collected
User item input query
Sends data to be processed
Update Display
Model
Search Product Price History Database

3. Project Scheduling, Cost, Effort and Pricing Estimation, Project duration and staffing

3.1. Project Scheduling

Weekends will not be counted and the number of working hours per day will be around 4, as the people implementing may have other responsibilities and 20 hours a week is the average part-time schedule. Based on the math done in 3.2, the project will take approximately 6 weeks for a team of 7 people to complete.

Start date: April 3rd, 2023.

- This was chosen because starting a project at the end of the year would lead to rushes to meet deadlines and would not give this project enough time to be created. April allows enough time for any deadlines and past assignments to be completed that could potentially interrupt this project.

End date: May 15th, 2023

- This is 6 weeks after the starting date, which is how much time this project will take to accomplish as specified in 3.2.

3.2. Cost, Effort and Pricing Estimation

We will use the Function Point (FP) estimation for this project.

We chose function point because our application is largely centered around user input/output, and the amount of data files/tables that we need to keep track of. We felt that function point did a better job of tracking this information, and so it would provide us a better estimate of how long the project would take us.

Function Category Count

User Inputs: 10 – 2 for login, 5 for user registration and 3 for bookmarking, searching and filtering
User Outputs: 6 – Counted success prompts
User Queries: 1 – One per item
Data Files and Relational Tables: 50 – It was assumed 10 items and 5 retailers per item
External Interfaces: 5 – The 5 retailers

Gross Function Point

	Function Category	Count	Simple	Average	Complex	Count * Complexity
1	Number of user input	10	**3**	4	6	30
2	Number of user output	6	4	5	**7**	42
3	Number of user queries	1	3	**4**	6	4
4	Number of data files and relational tables	50	7	**10**	15	500
5	Number of external interfaces	5	5	**7**	10	35
					GFP	**611**

Assumptions: Based on the sequence diagram

Processing Complexity:

PC1: 4
PC2: 5
PC3: 3
PC4: 4
PC5: 4
PC6: 5
PC7: 5
PC8: 3
PC9: 3
PC10: 3
PC11: 4
PC12: 3
PC13: 2
PC14: 4

Processing Complexity Adjustment:

PCA = 0.65 + 0.01 (4 + 5 + 3 + 4 + 4 + 5 + 5 + 3 + 3 + 3 + 4 + 3 + 2 + 4)
= 0.65 + 0.52 = **1.17**

Function Point
FP = GFP * PCA = 611 * 1.17 = **714.87**

Estimated Effort
E = FP / Productivity = 714.87 / 20 = ~ **36 person-weeks**

Project Duration:
D = E / team size = 36 / 7 = ~ **6 weeks**

3.3. Estimated cost of hardware products

Target user base: 2000
Due to our project being on the smaller scale we have decided that a target user base could be up to 2000, with this we would need a host capable of storing lots of user information (username/ passwords etc.) to accommodate this we have decided to adapt Web Hosting using Domain.com's Ultra Plan which costs $13.75 per month [4], which consists of support for unlimited websites, unlimited storage, scalable bandwidth, unlimited FTP logins, Free SSL certificate, and a free domain for the first year. Totaling an amount of $165 for the first year.

3.4. Estimated cost of software products

This product would primarily be built from the ground up from coding platforms such as react and django meaning that there would be no upfront cost as far as software. Some optional software products that the team may find useful could be a premium slack or Trello account for increased organization however these are definitely optional

3.5. Estimated cost of personnel

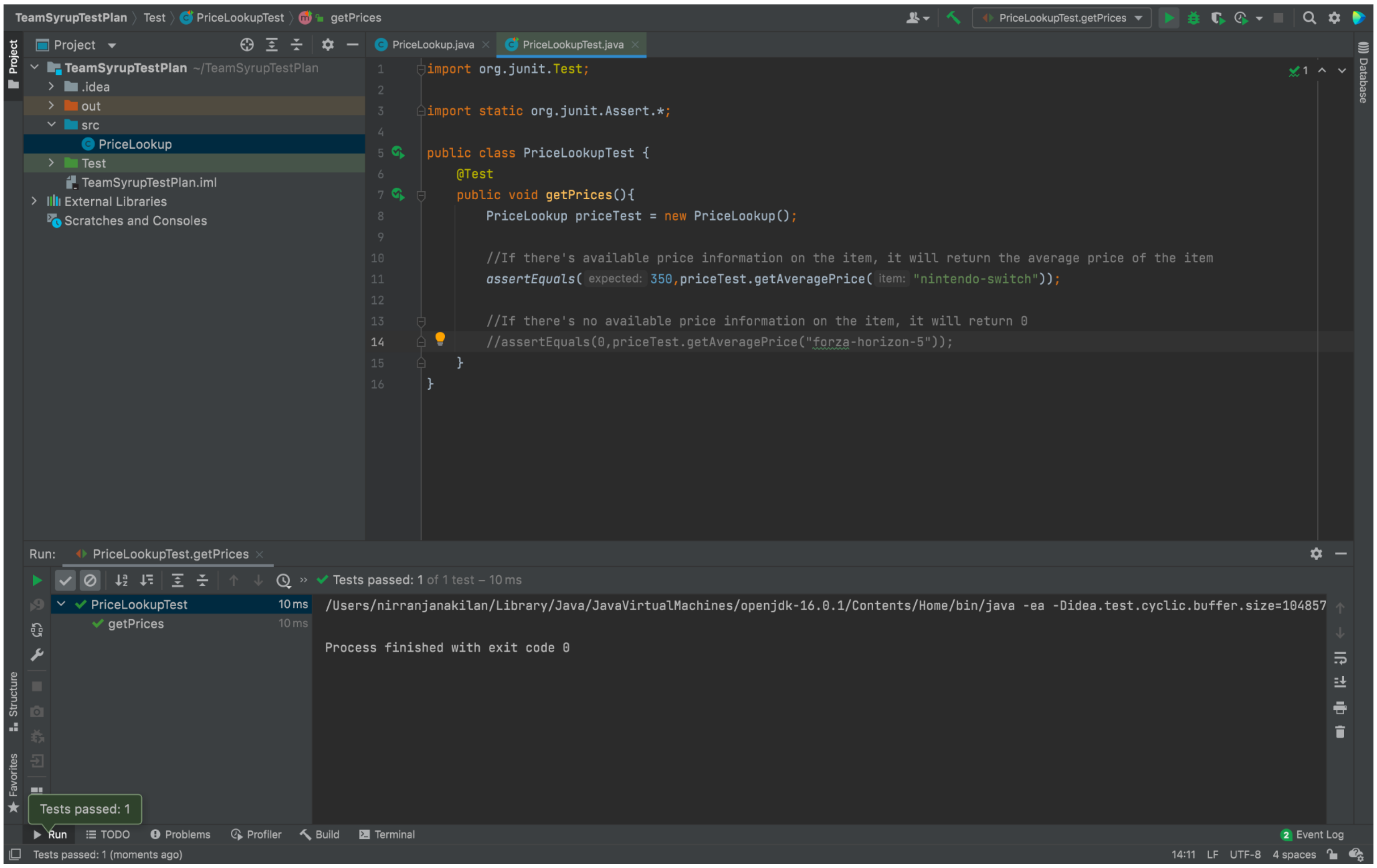
Given that the plan for the website is relatively small scale but efficient we believe that a group of 7 individuals is very capable of building, testing and deploying a website within the project timeline of 6 weeks. The team will consist of 4 front end engineers and 3 backend engineers. Based on the average salary for a front end engineer being 127,483 at 61$/hr [5] and a back end engineer being 131,789 at 64$/hr [5] we estimate that this will cost a total of $104,640 in personnel costs

4. A test plan for your software

We are using Junit to test an average price function of a given item. The average price of an item would be a good metric that complements the graph of the price history, allowing users to estimate how much money the item would cost regardless of the retailer of choice. Our test cases for the average price function show that if there exists a price history for an item in our system, it would compute the average price and return it back to the user. If no price history exists for an item in our system, then it would return an average price of 0 back to the user.

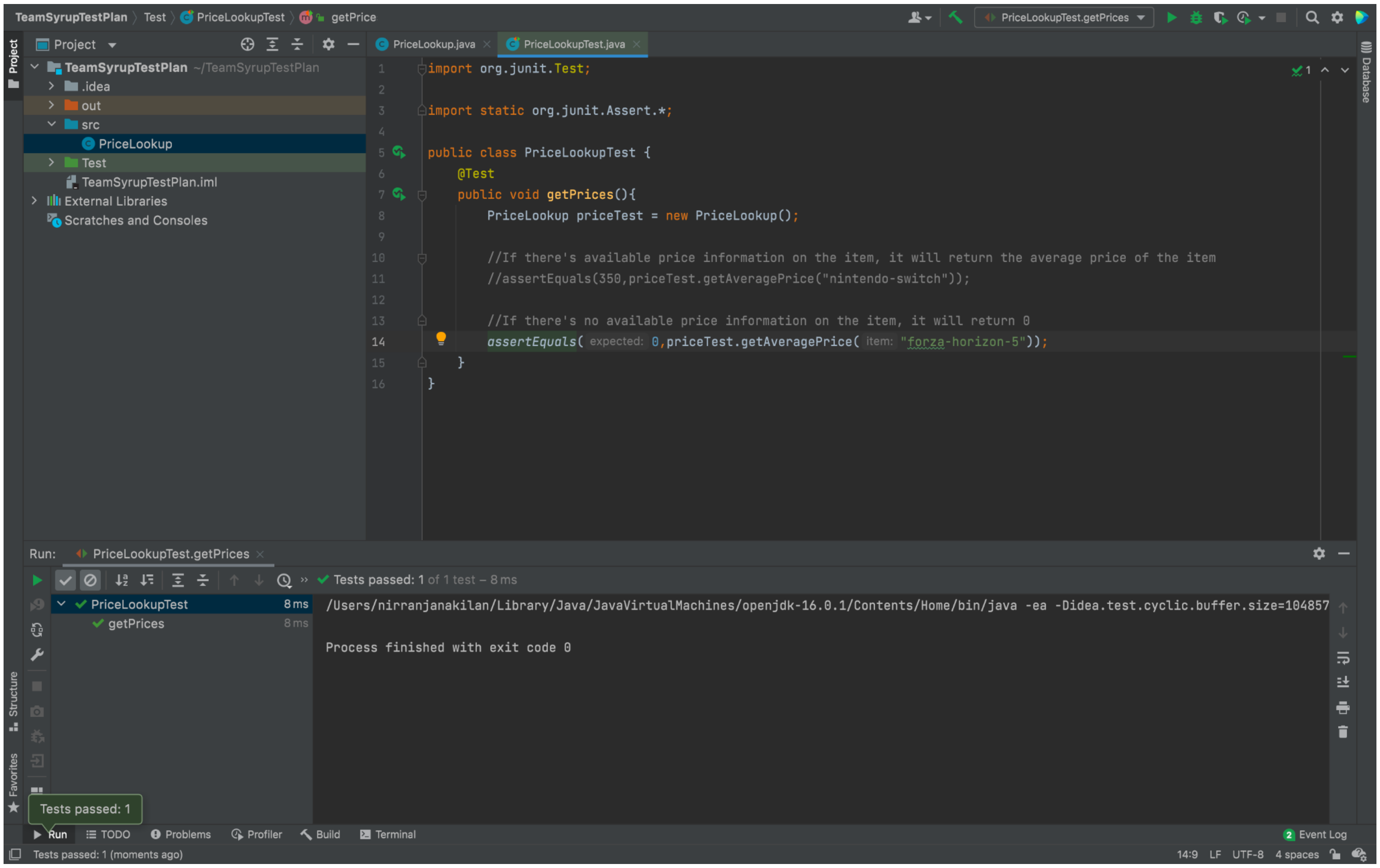
Test Code is included in the ZIP file and available on GitHub.

Screenshots:



```java
import org.junit.Test;

import static org.junit.Assert.*;

public class PriceLookupTest {
    @Test
    public void getPrices(){
        PriceLookup priceTest = new PriceLookup();

        //If there's available price information on the item, it will return the average price of the item
        assertEquals( expected: 350,priceTest.getAveragePrice( item: "nintendo-switch"));

        //If there's no available price information on the item, it will return 0
        //assertEquals(0,priceTest.getAveragePrice("forza-horizon-5"));
    }
}
```

```
Tests passed: 1 of 1 test – 10 ms

/Users/nirranjanakilan/Library/Java/JavaVirtualMachines/openjdk-16.0.1/Contents/Home/bin/java -ea -Didea.test.cyclic.buffer.size=104857

Process finished with exit code 0
```



```java
import org.junit.Test;

import static org.junit.Assert.*;

public class PriceLookupTest {
    @Test
    public void getPrices(){
        PriceLookup priceTest = new PriceLookup();

        //If there's available price information on the item, it will return the average price of the item
        //assertEquals(350,priceTest.getAveragePrice("nintendo-switch"));

        //If there's no available price information on the item, it will return 0
        assertEquals( expected: 0,priceTest.getAveragePrice( item: "forza-horizon-5"));
    }
}
```

Tests passed: 1 of 1 test – 8 ms

```
/Users/nirranjanakilan/Library/Java/JavaVirtualMachines/openjdk-16.0.1/Contents/Home/bin/java -ea -Didea.test.cyclic.buffer.size=104857

Process finished with exit code 0
```

5. Comparison of your work with similar designs

Our first model was based on a similar price-checking application named BuyVia [6] that utilizes barcode scanning(both UPC barcodes and standard QR codes) to find competitive pricing options from other vendors. BuyVia also has the option of a website that allows synchronous access to accounts in mobile devices, which became convenient for our application's design due to increased portability.

Applications like Karma coupons [7] and Honey [8] give customers the option of accepting coupons.

6. Conclusion

Originally Syrup was conceived to be a mobile price tracking application that displayed the price fluctuations of an item over a certain amount of time in order to tell a user whether the item they were scanning was at above or below the Manufacturer's Suggested Retail Price (MSRP). The app would compile several online retailers to compare the price of the desired product. Eventually Syrup evolved into becoming a website that would be accessible on desktop and mobile devices wherein a user could search for an item and be given a graph from several online retailers that showed the item's price change in price over time and the item's MSRP. Functionalities that survived between both iterations of Syrup were the ability to see charts that showed the user the price history of an item they were looking for. We began to shift away from price scanning via barcodes to a user guided search to find price histories. Once we began shifting away from price scanning the photo functionality for a mobile application then became unessential and the switch to a website became more reasonable. Once we became a website we added functionalities like bookmarking queried items so users could be alerted to an item's price drop.

7. References

[1] I. Sommerville, *Software engineering, Tenth Edition*. Harlow, Essex: Pearson Education, 2016.

[2] Creately, “UML diagram tool: UML Diagram Online,” Creately, 26-Oct-2021. [Online]. Available: https://creately.com/lp/uml-diagram-tool/. [Accessed: 18-Nov-2022].

[3] draw.io, “Diagrams.net - free flowchart maker and diagrams online,” Flowchart Maker &amp; Online Diagram Software. [Online]. Available: https://app.diagrams.net/. [Accessed: 18-Nov-2022].

[4] “Domain name search,” *domain.com* | *Domain Names - Buy a Website Domain*. [Online]. Available: https://www.domain.com/. [Accessed: 18-Nov-2022].

[5] “Glassdoor job search | you deserve a job that loves you back,” *glassdoor.com*. [Online]. Available: https://www.glassdoor.com/index.htm. [Accessed: 18-Nov-2022].

[6] Crunchbase, “Buyvia - Crunchbase Company Profile & Funding,” *Crunchbase*. [Online]. Available: https://www.crunchbase.com/organization/buyvia. [Accessed: 18-Nov-2022].

[7] “Karma | Best Coupons and Cash Rewards,” *karmanow.com*. [Online]. Available: https://www.karmanow.com/. [Accessed: 18-Nov-2022].

[8] “Automatic Coupons, Promo Codes, and Deals | Honey,” *joinhoney.com*. [Online]. Available: https://www.joinhoney.com/. [Accessed: 18-Nov-2022]

8. Presentation slides

Included in the zipped folder and available on GitHub.

10. GitHub requirement

https://github.com/BrandonMCoffey/3354-TeamSyrup