

CS 320 Course Project Final Report

for

Price Analizer

**Prepared by**

|  |  |  |
| --- | --- | --- |
| **Trever Hibbs** | **11577861** | [**trever.hibbs@wsu.edu**](mailto:trever.hibbs@wsu.edu) |
| **Muniker Aragon** | **11632391** | [**muniker.aragon@wsu.edu**](mailto:muniker.aragon@wsu.edu) |

|  |  |
| --- | --- |
|  |  |
| **Date:** | **12/13/2019** |
|  |  |
|  |  |
|  |  |

**Contents**

**Contents ii**

**1** **Introduction 1**

1.1 Project Overview 1

1.2 Definitions, Acronyms and Abbreviations 1

1.3 References and Acknowledgments 1

**2** **Design 2**

2.1 System Modeling 2

2.2 Interface Design 2

**3** **Implementation 3**

3.1 Development Environment 3

3.2 Task Distribution 3

3.3 Challenges 3

**4** **Testing 4**

4.1 Testing Plan 4

4.2 Tests for Functional Requirements 4

4.3 Tests for Non-functional Requirements 4

4.4 Hardware and Software Requirements 4

**5** **Analysis 5**

**6** **Conclusion 6**

**Appendix A - Group Log 7**

# 

# Introduction

## Project Overview

The main purpose for the design of this software is to provide users with an efficient tool for overseeing the price of multiple online products that they wish to purchase. Users are able to create a personalized profile where they can store multiple products by simply entering the URL of the products web page. This application saves the URL of the item and continuously monitors and informs the user if a drop in the product’s price is detected. The main benefit of this application is that it provides users with a place where they can save and monitor all the items that they wish to buy while eliminating the time that it would take to search each of these items individually.

A high overview of the architecture of the system consists of implementing models to represent an User object and an Item object. These structures hold a one to many relationships where an User is able to be mapped to many Items. Another key structure for this application consits of a developed web scraping script that holds all the functions utilized for retrieving the price of a product. Lastly, we implemented a script that handles the updating of each User in the system. This python script stores the User in a queue where it then begins to iterate to the users and update the price for all of their items.

## Definitions, Acronyms and Abbreviations

**Abbreviations**

*URL: Uniform Resource Locator*

*MVC: Model View Controller*

*AWS: Amazon Web Services*

*URI: Uniform Resource Identifier*

*HTTPS: Hypertext Transfer Protocol Secure*

*HTML: Hypertext Markup Language*

*SQL: Structure Query Language*

*WSGI: Web Server Gateway Interface*

*IP: Internet Protocol*

**Definitions**

*user agent: Strings that reveal a catalog of technical data about the device and software that a website visitor is using.*

## References and Acknowledgments

*<List any other documents or Web addresses to which this document refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document.*

*TO DO: Use the standard IEEE citation guide for this section.>*

**Documentation**

[1] Docs.djangoproject.com. (2019). *Django documentation*. [online] Available at:

<https://docs.djangoproject.com/en/2.1/> [Accessed 13 Dec. 2019].

[2] Richardson, L. (2019). *Beautiful Soup: We called him Tortoise because he taught us.*.

[online] Crummy.com. Available at: <https://www.crummy.com/software/BeautifulSoup/>

[Accessed 13 Dec. 2019].

# Design

## System Modeling

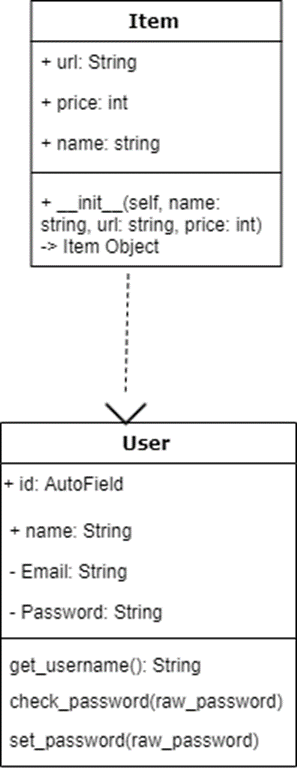
Not every UML diagram remained the same through the implementation of our web app. The diagrams that required changes are included below.

### Use Case Diagram

### Structural Modeling

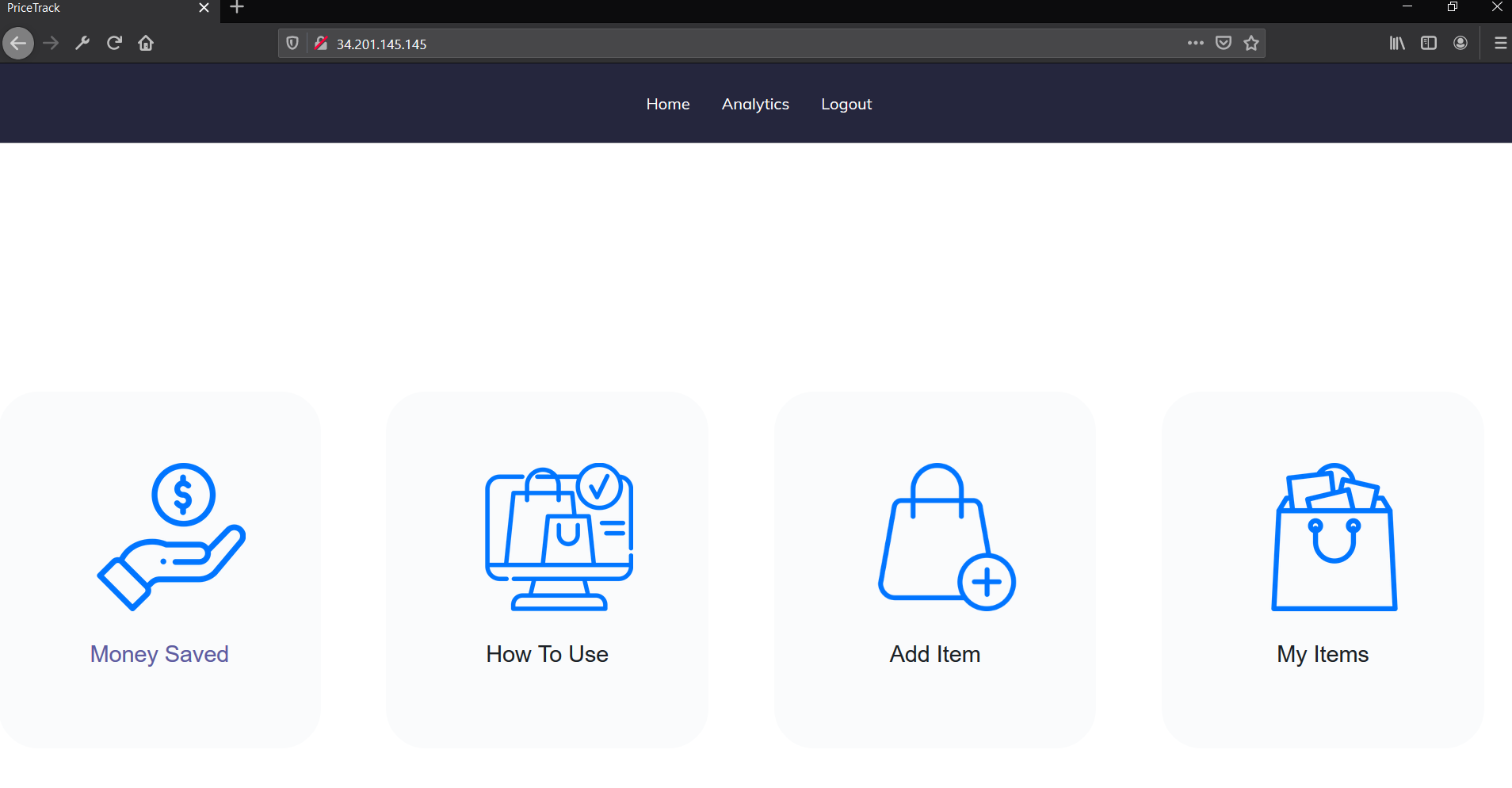
#### Item Class Diagram

#### Item Class Diagram

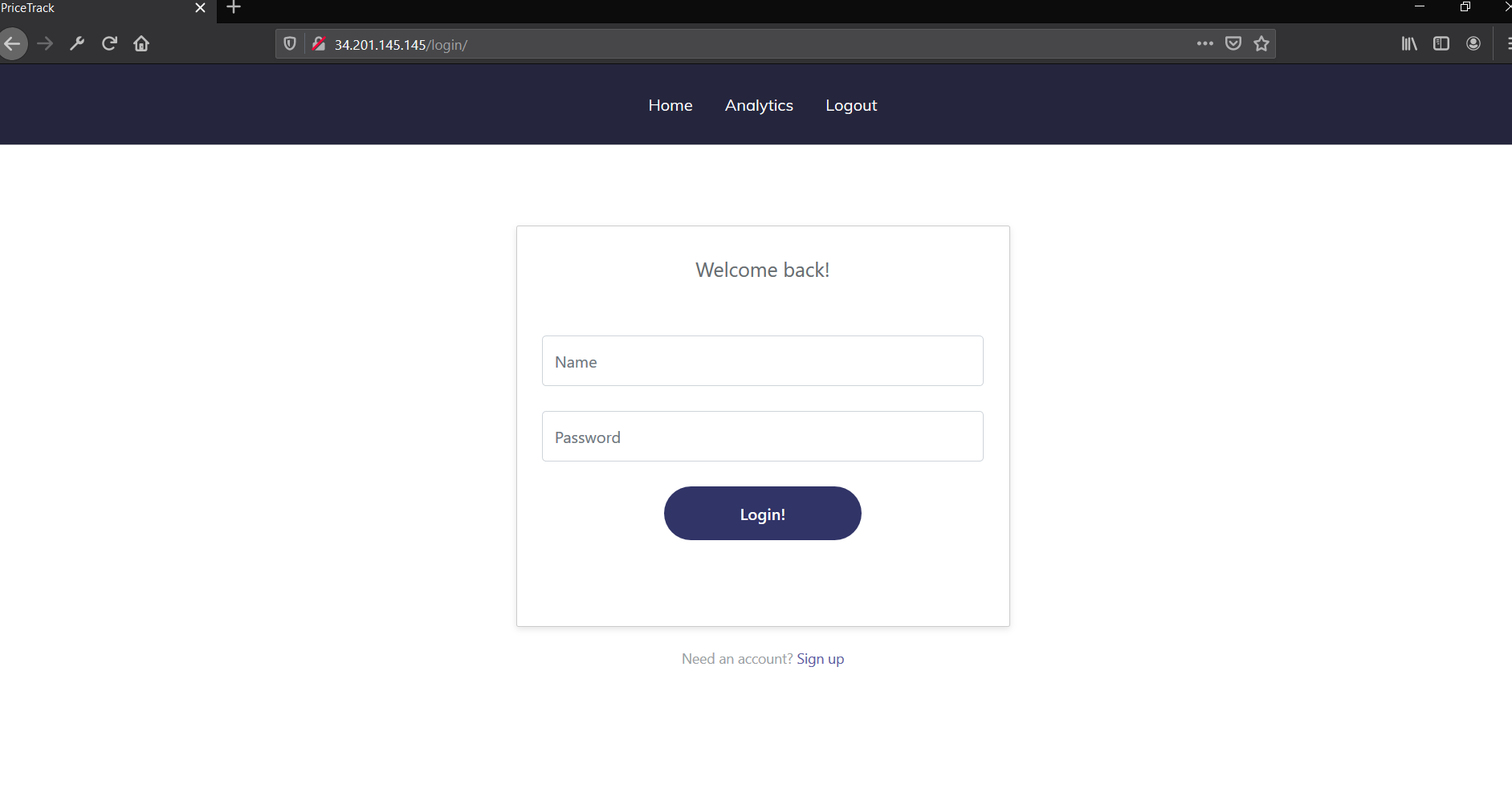


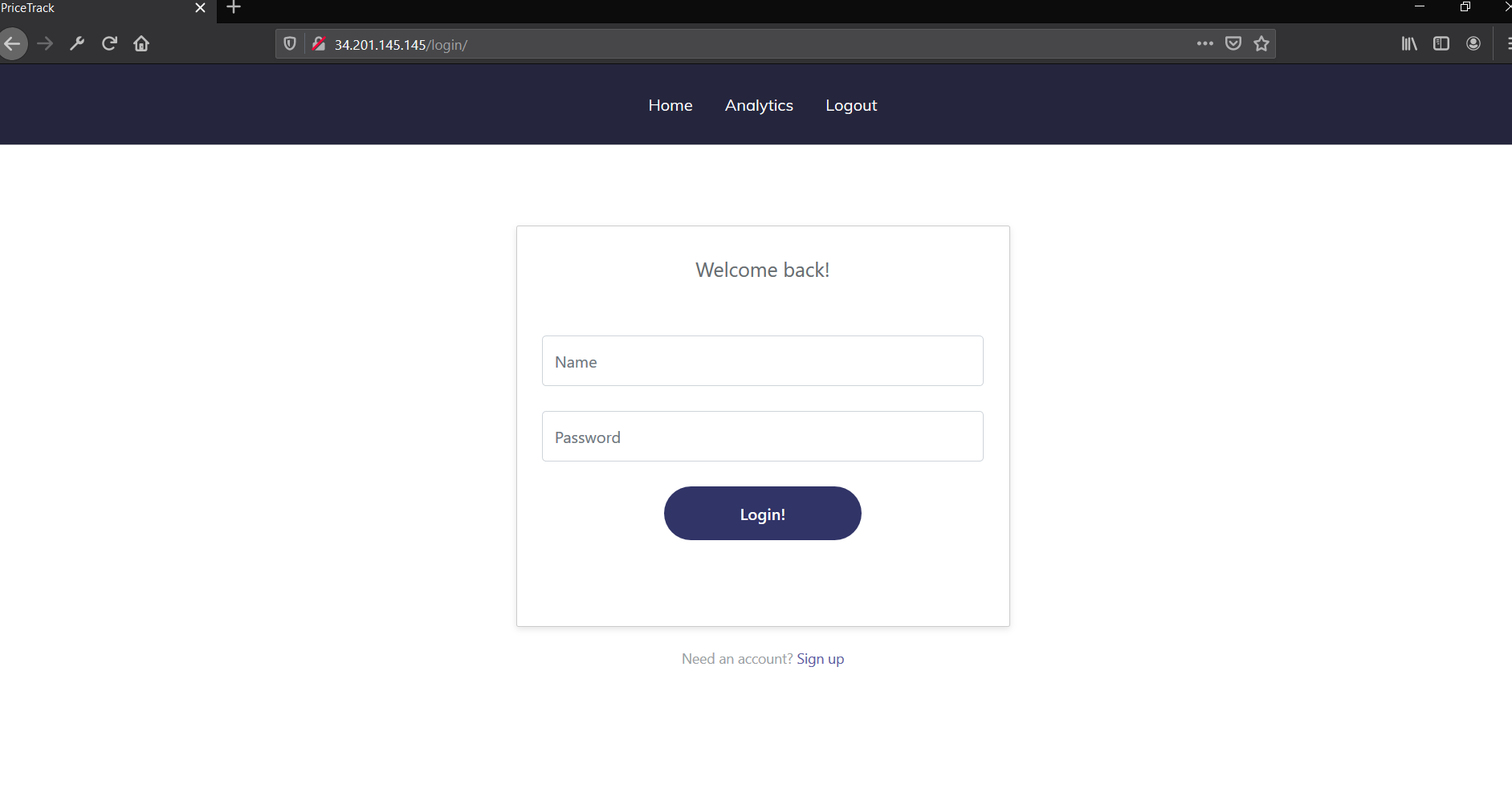
## Interface Design

*Live view at:* [*http://34.201.145.145/*](http://34.201.145.145/)

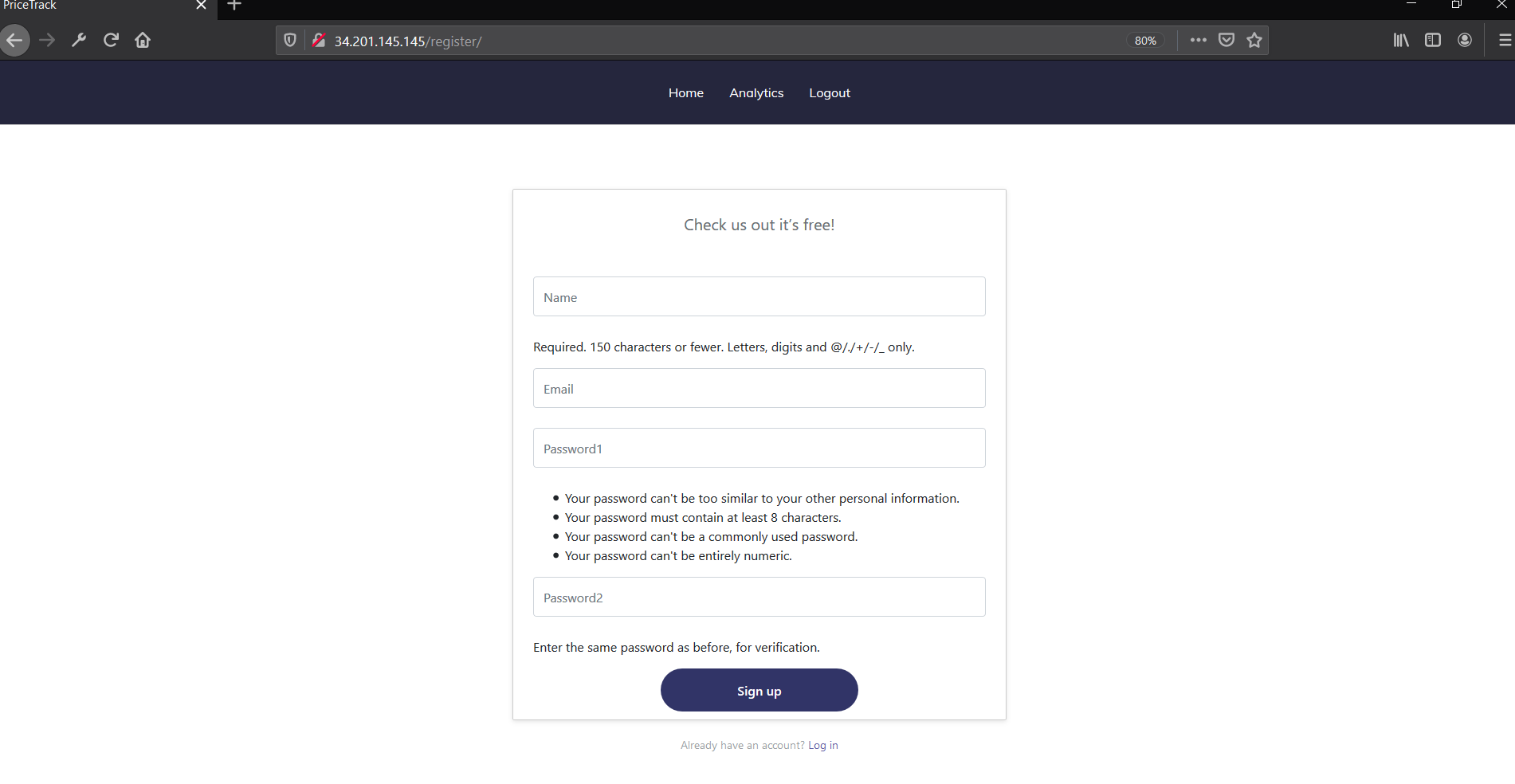
*Home\_page* 

*Login\_page*

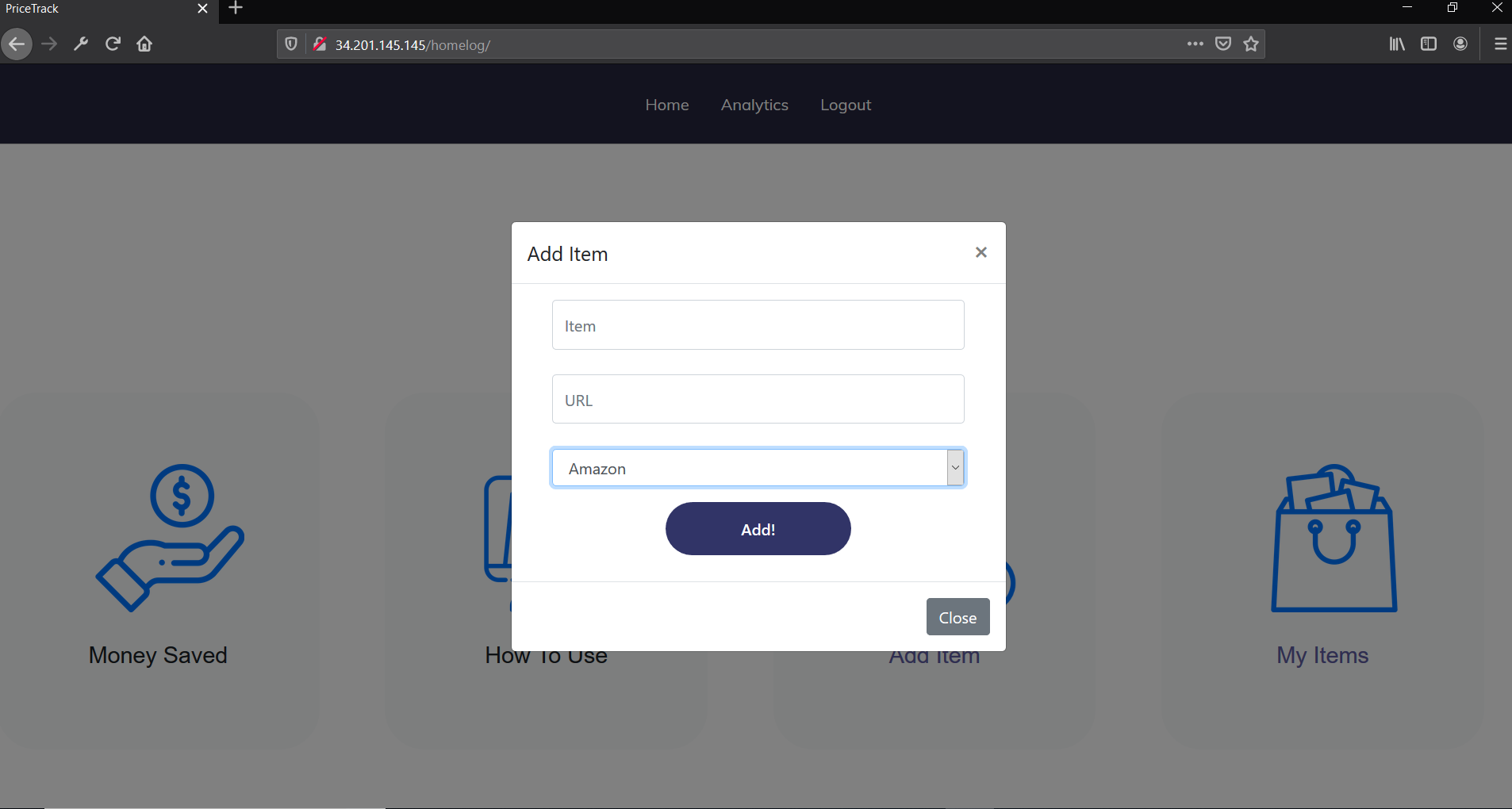


**

*Registration\_page*



*AddItem\_page*



# Implementation

## Development Environment

*Inplemented Programming languages, IDEs, and other tools:*

* *Amazon Web Services*
* *APACHE*
* *Visual Studio Code*
* *Python*
* *Django*
* *HTML*
* *CSS*
* *BootStrap*

## Task Distribution

Muniker Aragons’ implementation tasks

* the initial set up and configuration of the Django web application framework
* the development of the applications UI
* the set up of the AWS web server
* implementation of the best buy and walmart web scraping functionality
* implementation of the pricing update system
* implementation of the registration and login system
* the creation of the view and model classes.

Trever Hibbs’ implementation tasks

* implementing the Amazon web scraping function
* testing the webscraping functions
* creating the item class.

## Challenges

A major unexpected implementation challenge that came up during the project was, the Amazon store flagging our application as a bot and blocking its HTTP requests. To get around this our group implemented a system that randomized the IPs and user agents that the web scraping application used to send the HTTP requests to the amazon store. This system grabs several random IPs from an online IP database and stores them in a python list. Then it imports a list of user agents from a directory on the server. It then sends the amazon store HTTP requests using the IP and user agent list until it gets a valid response or it runs out of IPs or user agents. This process was only needed for the amazon store. The walmart store and the best buy store did not try to block our application.

# Testing

## Testing Plan

*<Describe your testing plan for the project.*

*TODO: Give a list of items or functions you want to test, and also a schedule for performing the testing. >*

## List of Item To Test

### Functional Requirements

* All webscriping functions
* database reading and writing
* User registration functions
* User login functions
* User logout functions
* User saving item functions
* Item view page function
* User item update system function

### Non-functional Requirements

* All web scraping functions

## Testing Schedule

All functional tests were performed as the corresponding function was being

implemented. All non-functional tests were performed after all functional

requirements were implemented and functional.

## Tests for Functional Requirements

Requirement1)

User is able to create a new account in the system by inputting an username, email, and password.

**Test:** Created a query to the database to make sure the information had been correctly stored.

**Result:** Pass

Requirement2)

User is able to be authenticated by the system and directed to their personalized account.

**Test:** Performed black box testing where we input the information of an user and then waited to be redirected to the home page.

**Result:** Pass

Requirement3)

User is able to save an online item to their account by inputting the Item’s Name, Url and store.

**Test:** Created a query to the database to make sure the information had been correctly stored.

**Result:** Pass

Requirement4)

User is able to see a list of their previously stored items.

**Test:** Utilized the Django administrative interface to check if the information had been written to the database

**Result:** Pass

Requirement5)

All the user Items are updated every 10 minutes.

**Test:** Initiated the updateUser script and manually changed the price from one the user’s Items.Next, we waited to see if the price of this Item would be updated to the correct value.

**Result:** Pass

## Tests for Non-functional Requirements

*<Similar to the Section 4.2, but this section is for the non-functional requirements. >*

Requirement1)

The walmart web scraping function must return data within 2 seconds.

**Test:** Using pytest and the timeit python module assert that the walmart function return data in less than 2 seconds.

**Result:** time in seconds for walmart function call 0.9120285839999269 -> *PASS*

Requirement2)

The amazon web scraping function must return data within 2 seconds.

**Test:** Using pytest and the timeit python module assert that the amazon function return data in less than 2 seconds.

**Result:** time in seconds for amazon function call 1.8524380469998505 -> *PASS*

## Hardware and Software Requirements

*<Describe the hardware and software requirements for performing the tests. >*

The tools that our group utilized to perform test on this system consist of the Django administration Interface, and Django python shell, pytest, and python unittest framework. This project implemented various Models which represented information for the objects which would then be stored in the database. To test the implementation and storage of the model we utilized the Django administration interface as this provided us with an efficient and easy way of creating and saving a model object. Another tool we utilized for testing consists of the Django shell, this interactive shell allowed us to test the different queries that needed to perform in our database while preventing us from having to write a full script. The last tools utilized for testing was the unittest framework and the pytest test runner. Using these tools we tested the functionality and efficiency of the web scraping functions.

# Analysis

*<In this Section you need to analyze the effort that has been put on this project.*

*TODO: Describe how many hours (approximately) each team member spent on the project, for each milestone, which milestone takes the most effort and why. >*

*Trever Hibbs spent spent approximately 6 hours writing code for the project and another couple hours writing and performing non-functional tests. The milestone assignments all took about 2 to 4 hours to complete. The milestone assignment that required the most work from Trever was milestone 2. This milestone was the most challenging because it required detailed planning and diagramming.*

*Muniker Aragon spent about 20 hours coding the application and configuring the web server. And he spent about 2 to 3 hours working on each of the milestones. Out of all the milestone assignments milestone 2 took the most amount of work to complete. Milestone 2 took the most amount of work because the creation of the diagrams required detailed knowledge of the technical aspects of the project. There were many details to consider when thinking about all the software components and how they interacted.*

# Conclusion

*While developing this project our team has been able to learn how to design and implement a dynamic web application from the ground up. This project provided us with the experience of creating different Software related documents such as Requirements Specifications, System models, and Functional Specifications. In this project, we were also able to obtain experience in utilizing Github which is a version control tool that helped our team manage the changes we performed our source code. Aside from Software documentation, we were also able to acquire a lot of new programming knowledge in working with web page requests as well as utilizing different web scraping utilities. We strongly believe that we will be able to transfer over the knowledge we have gained from developing this project to any Internships or jobs that we may obtain in the near future. Lastly, this project helped us greatly improve our team working skills as we had to maintain close communication and find efficient ways of dividing up the work so that we could help cover each other's weaknesses.*

**Appendix A - Group Log**

*< Describe how frequently the group meembers meet during the semester, and how effective the communication is. This is optional for one-person projects.>*

*During the semester the group members ment about once every month to discuss the latest details of the project and upcoming milestone assignments. Most communication was handled by instant messaging. There were no serious communication issues.*