

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

*<TO DO: Please provide a brief introduction to your project.>*

## Project Overview

The main purpose for the designed 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 site. 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 benefits of this application is to provide users 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 individually.

An high overview of the architecture of the system consists of: -----

## 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

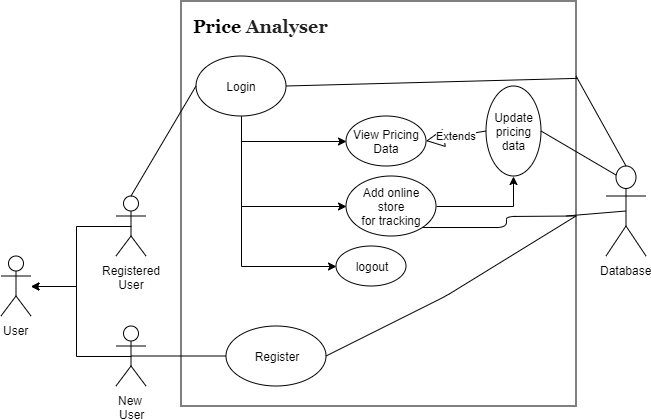
## System Modeling

< Update your UML diagrams in milestone 2, to reflect the real implementation of this software.

TO DO: Provide an updated version of the UML diagrams, including use case diagrams, sequence (or state) diagrams, activities diagrams, and class diagrams. If you don’t have an updated version, just mention: “our implementation strictly follows the design document (milestone 2)”. >

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

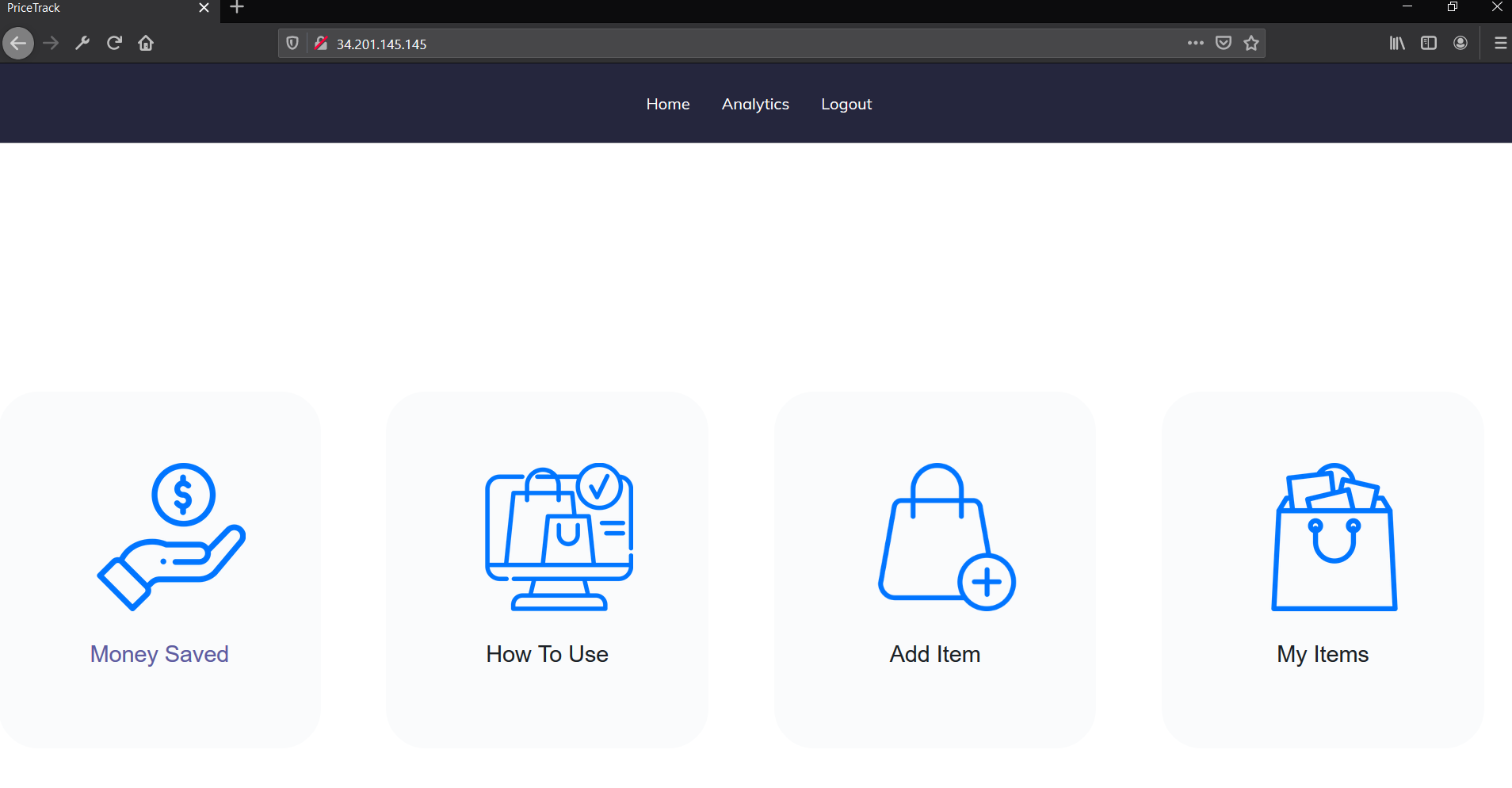
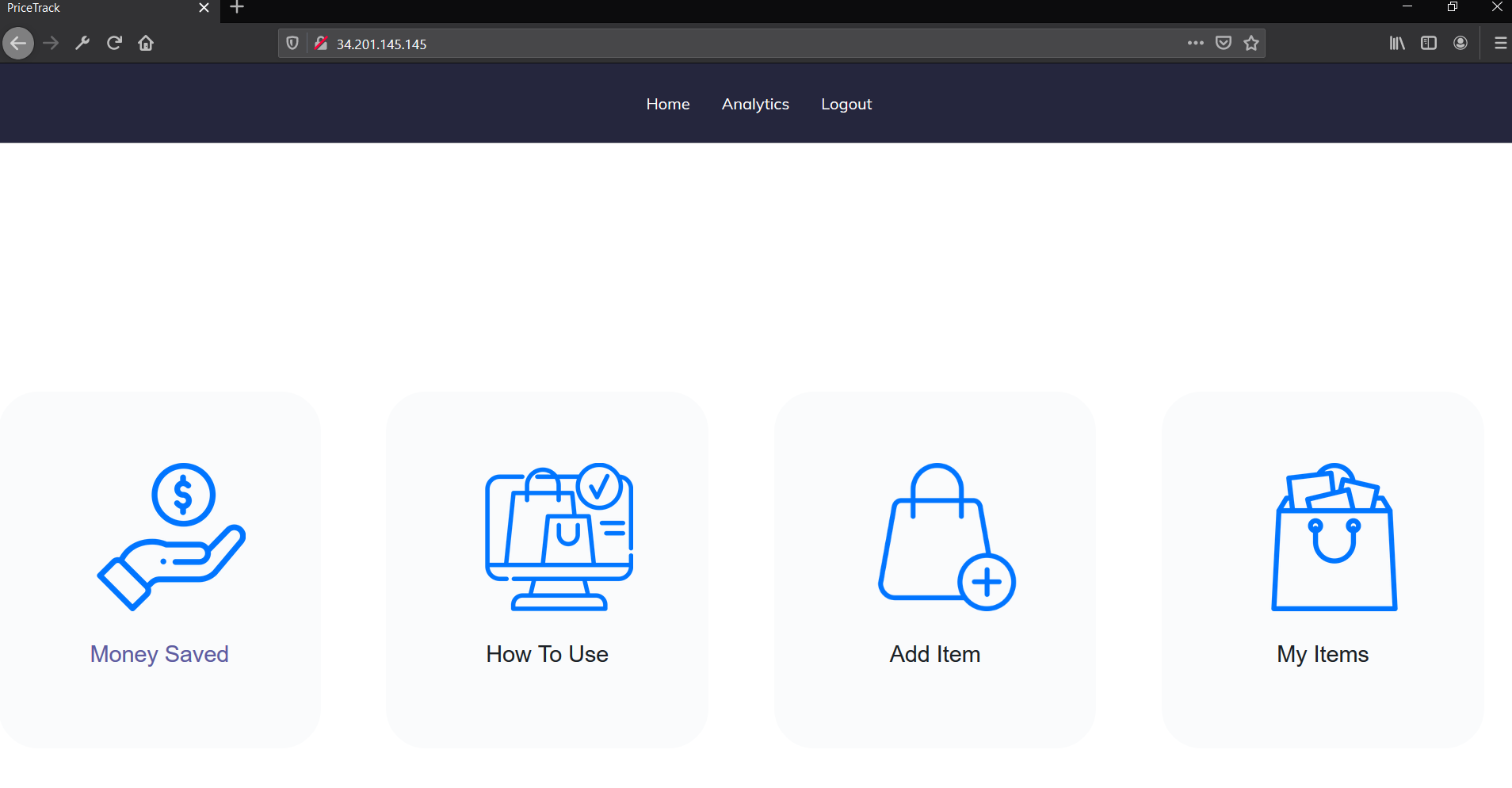
****

### Structural Modeling

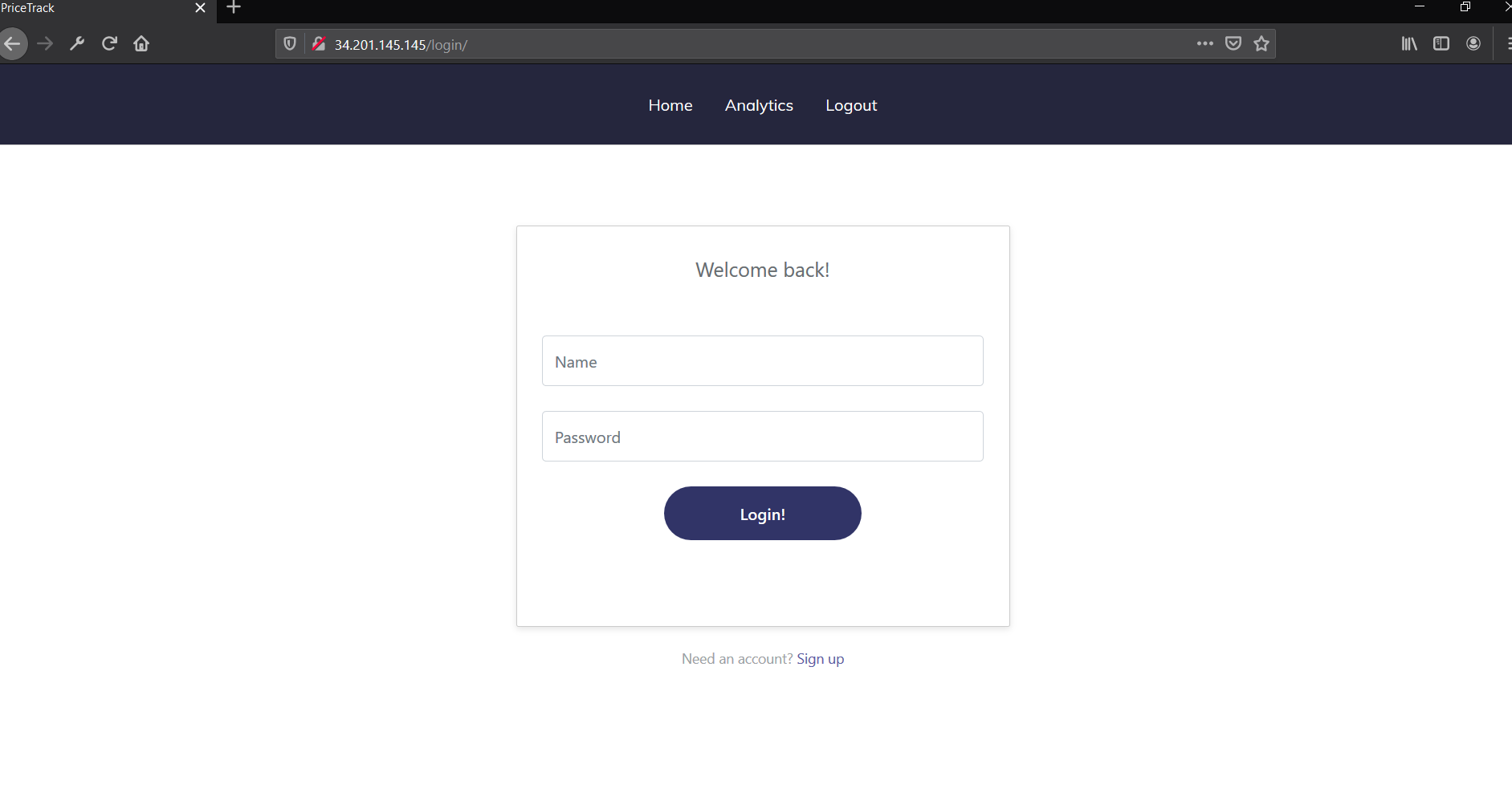
#### Item Class Diagram

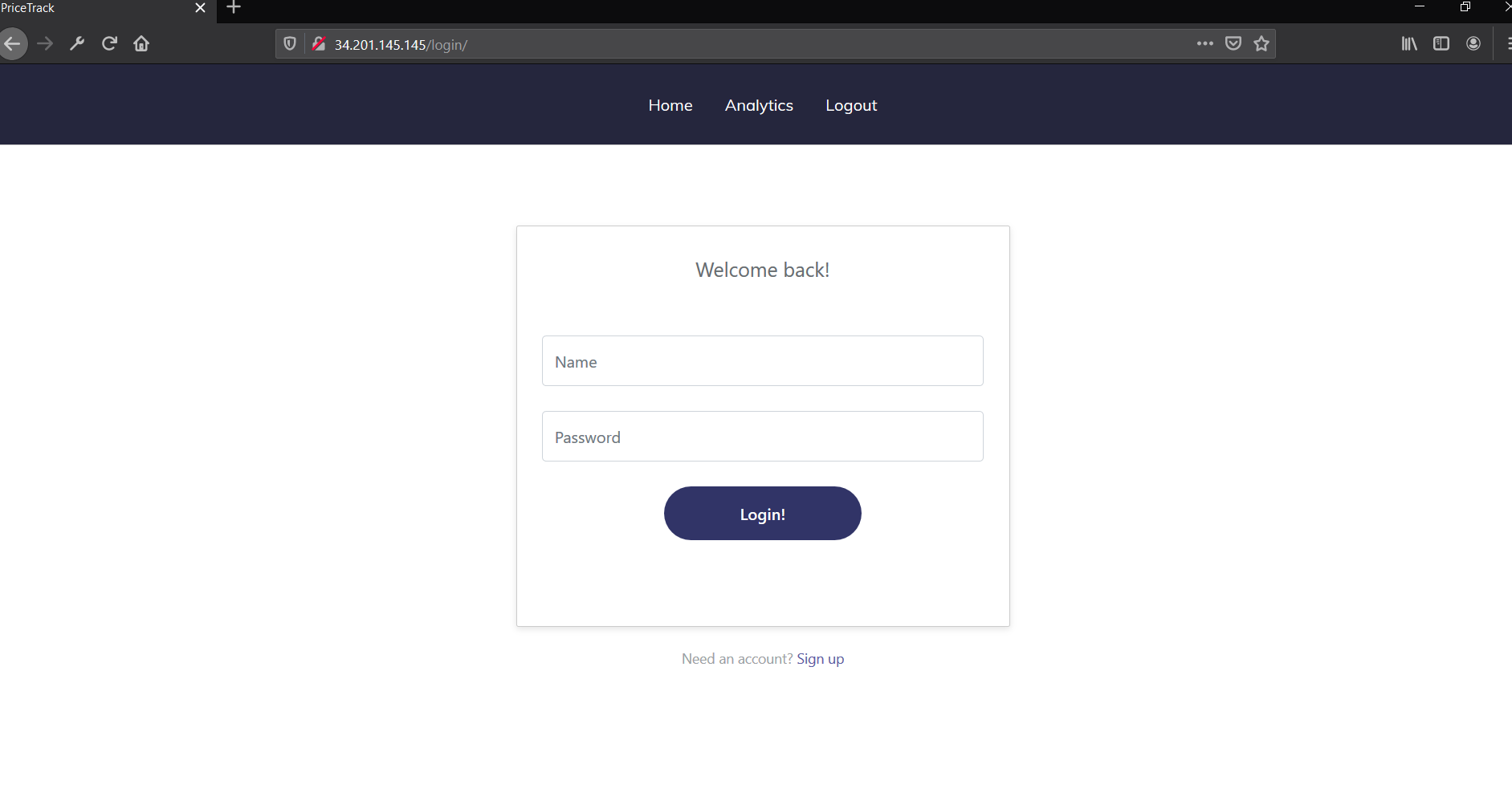
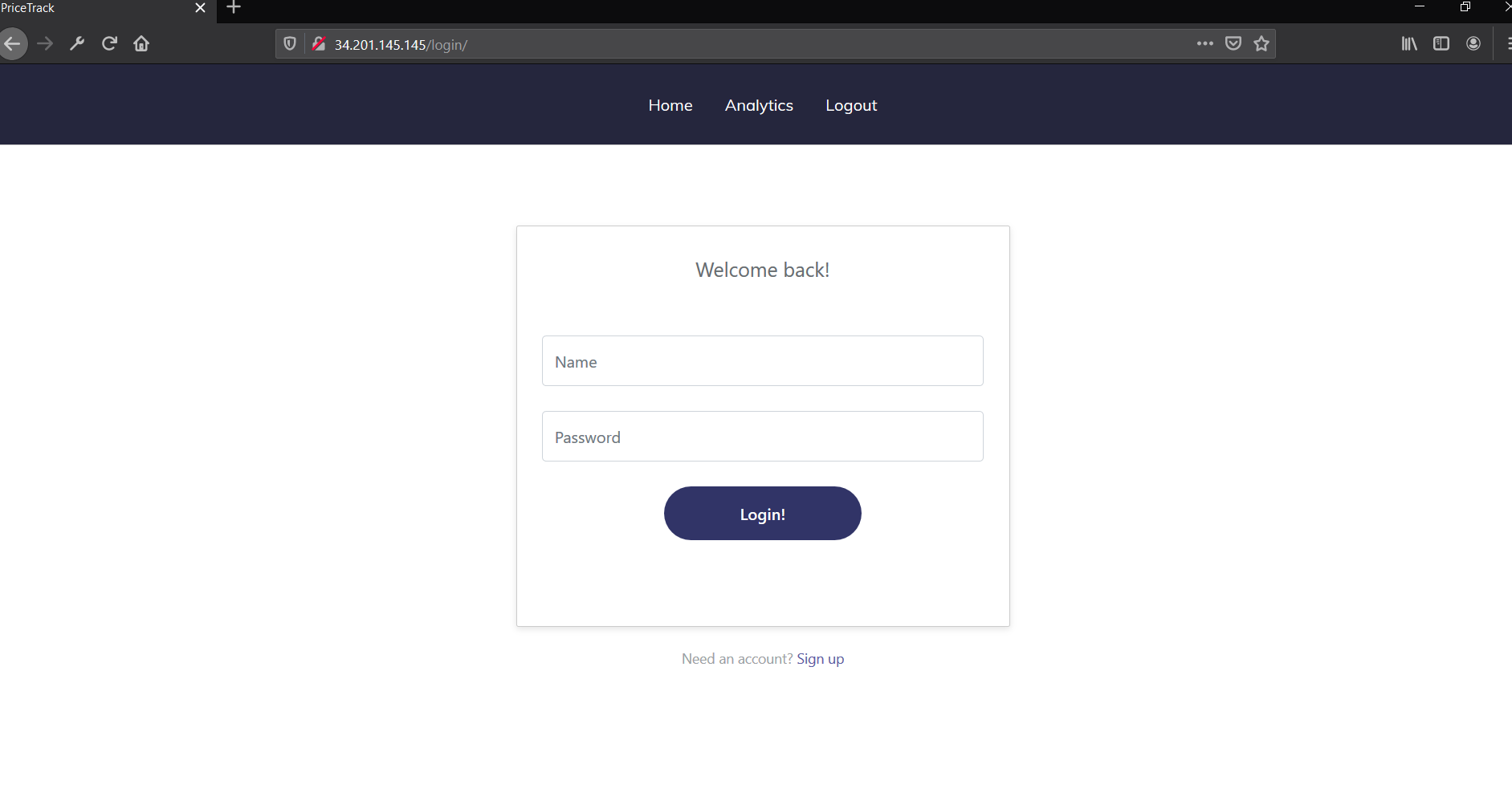
### 

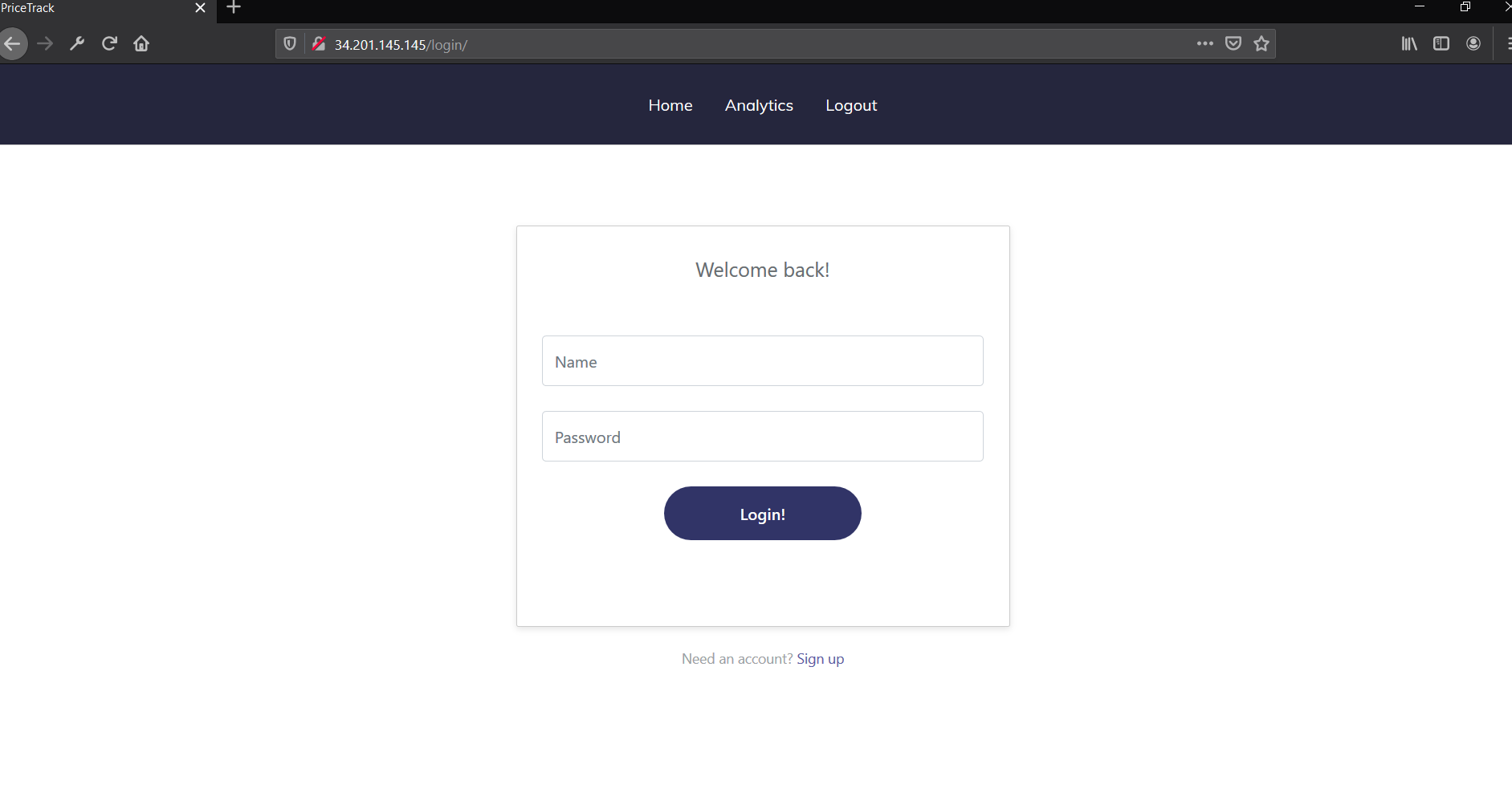
## Interface Design

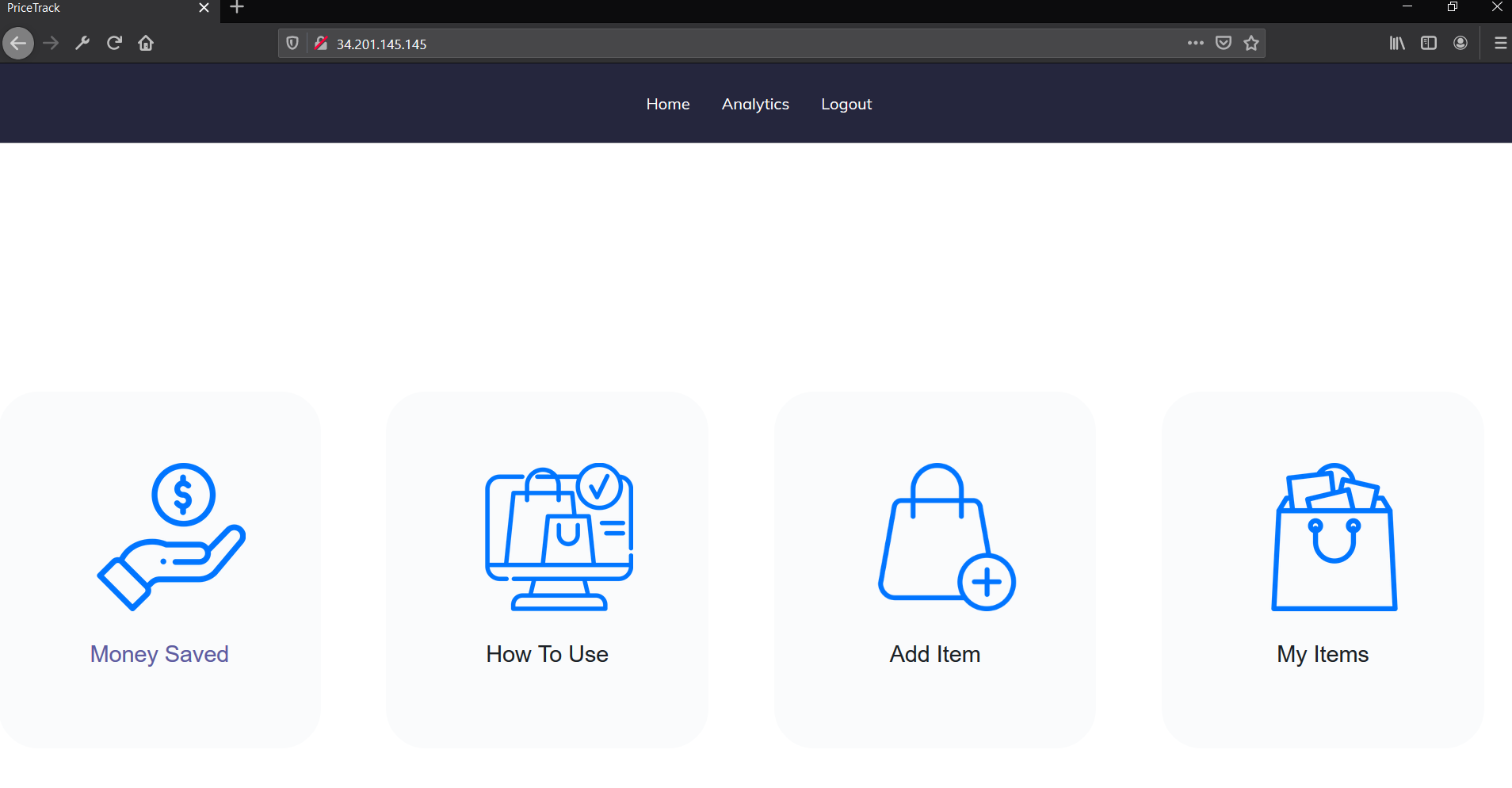
Home\_page

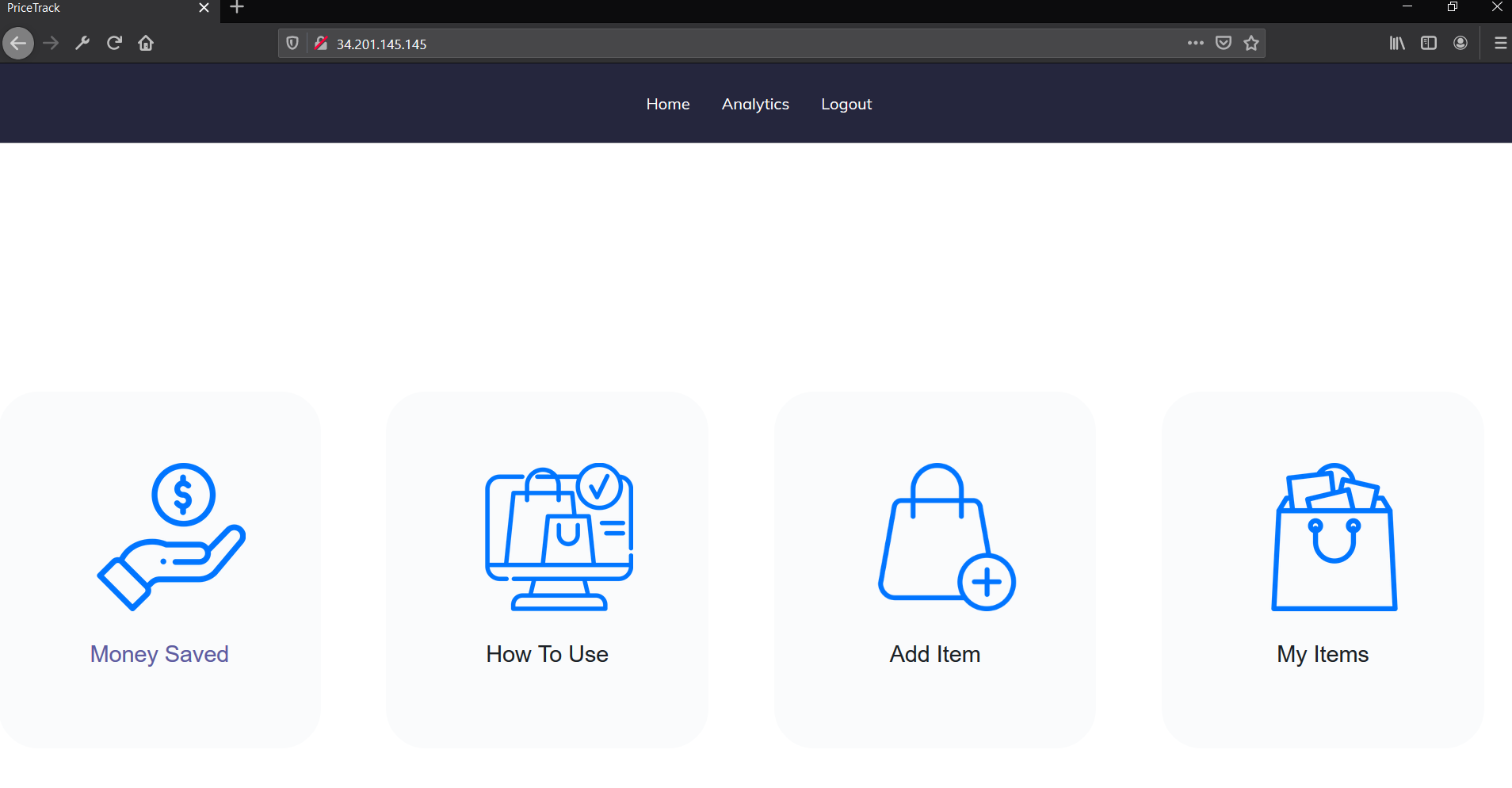
Login\_page

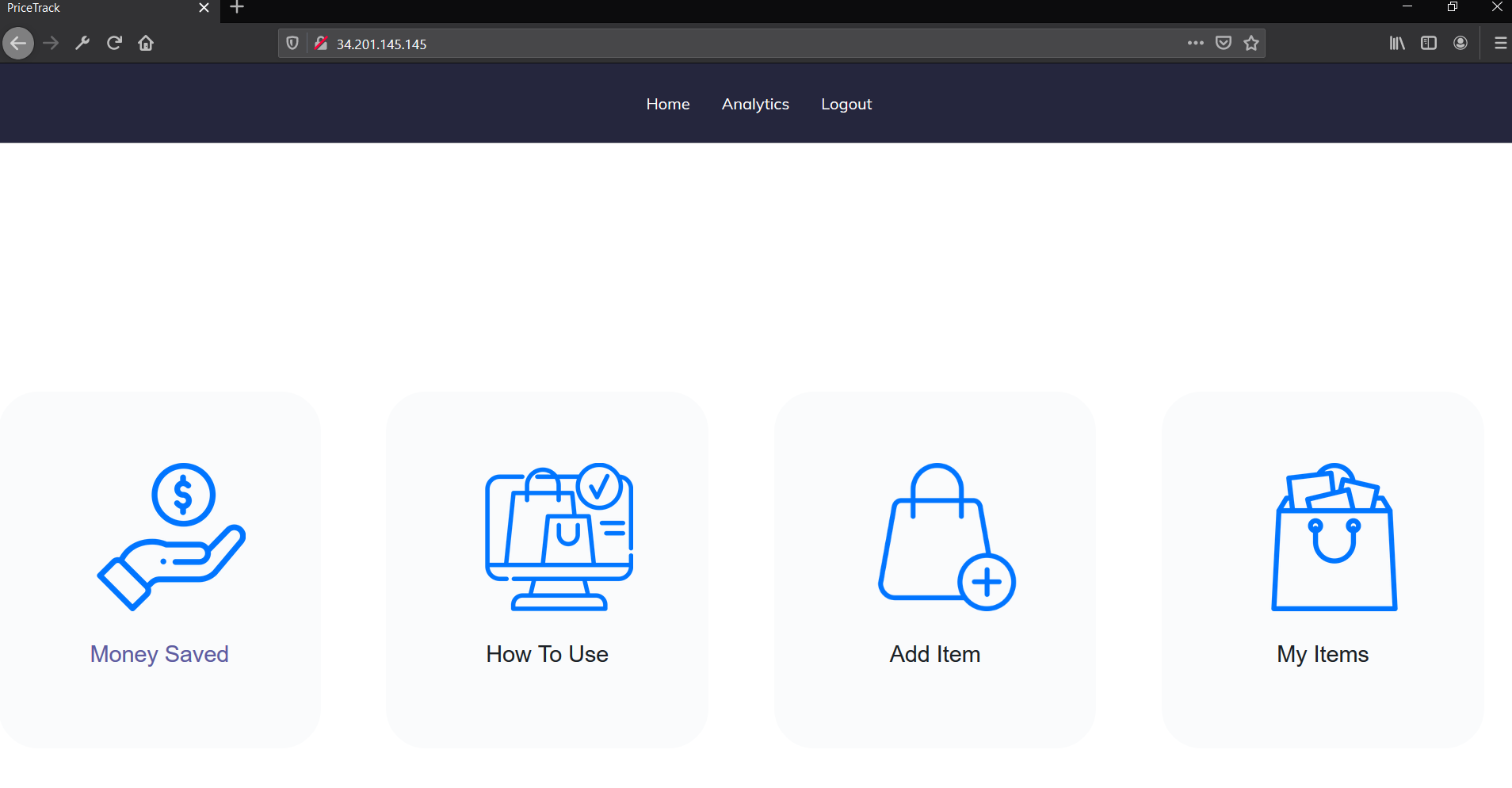


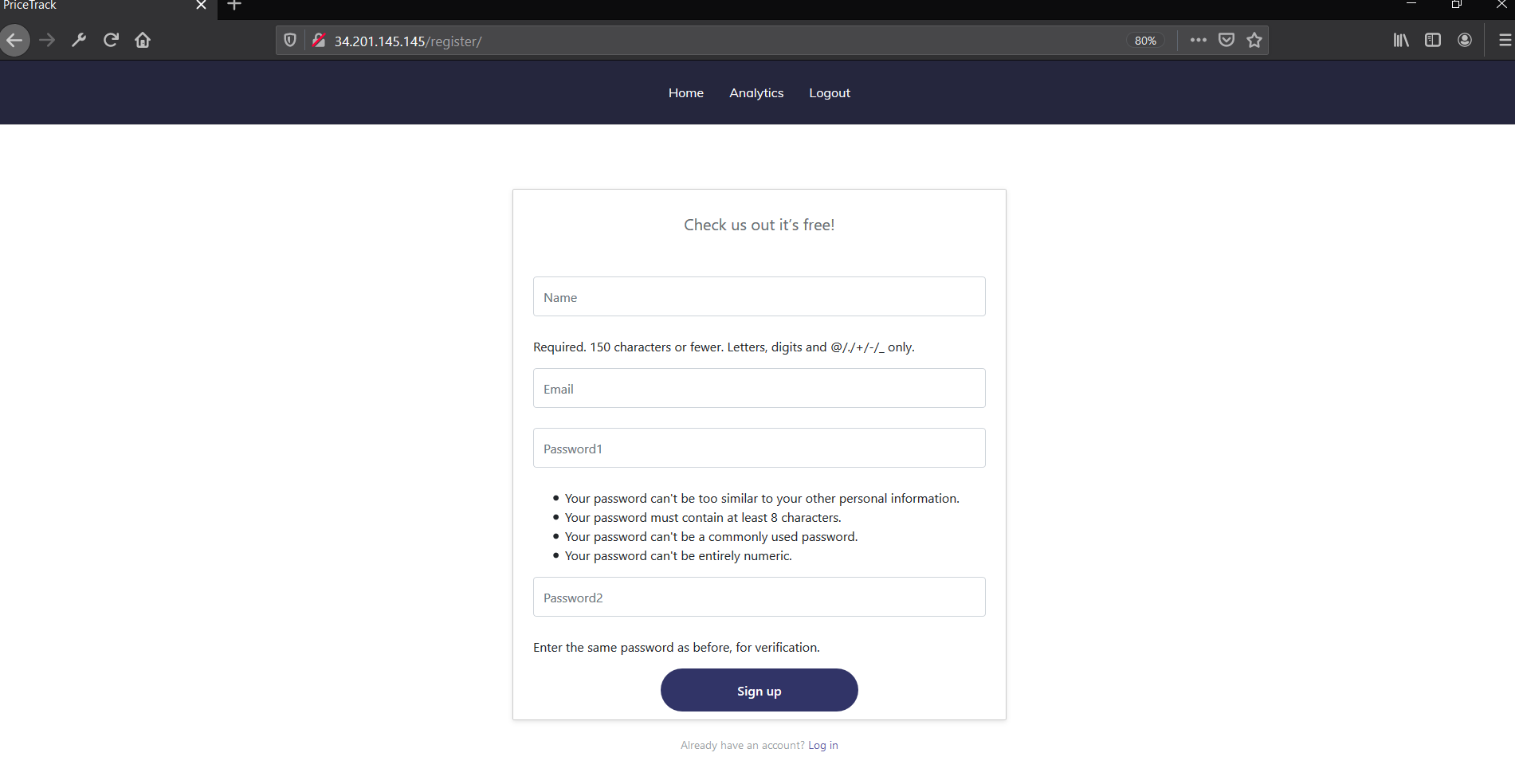
 

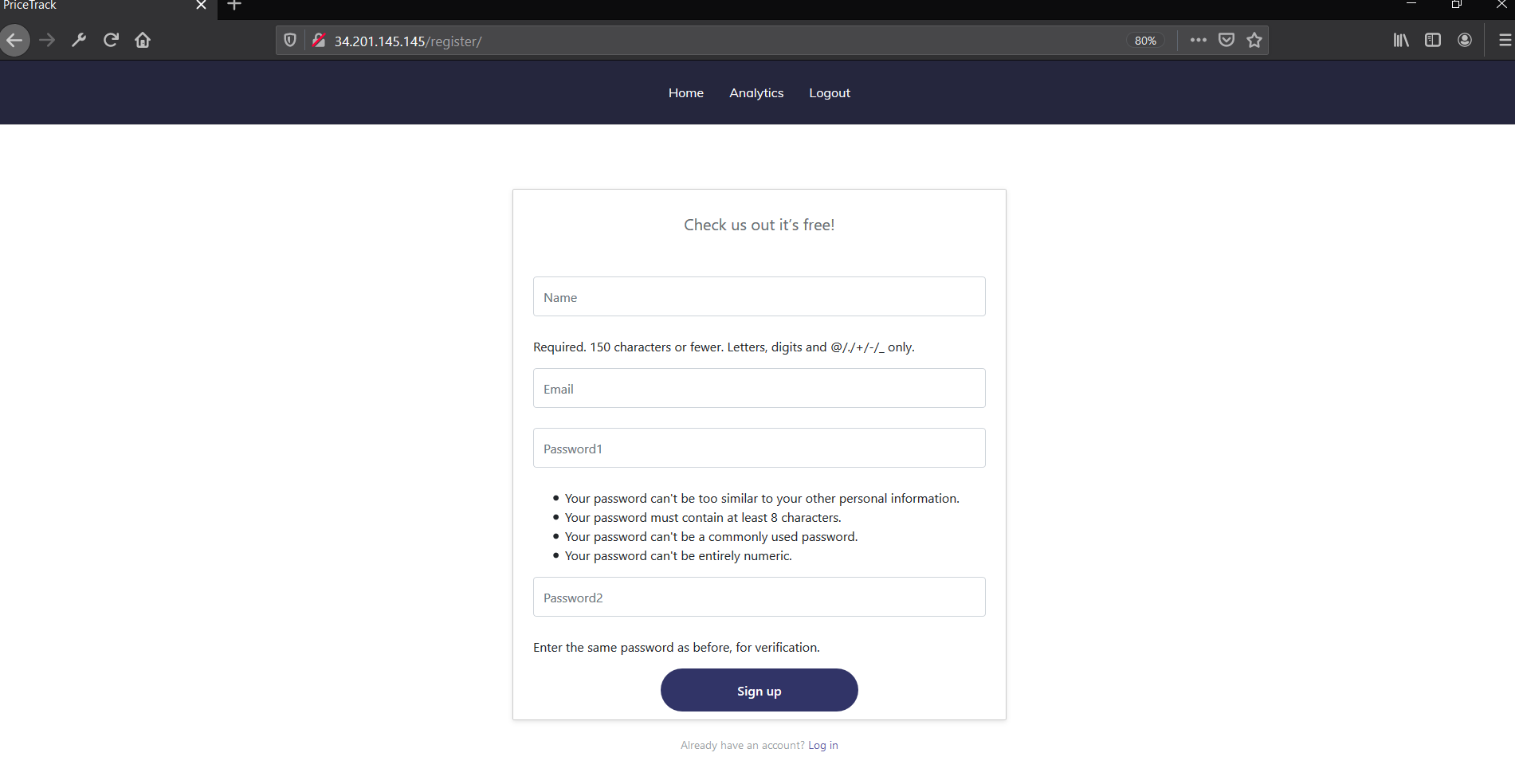




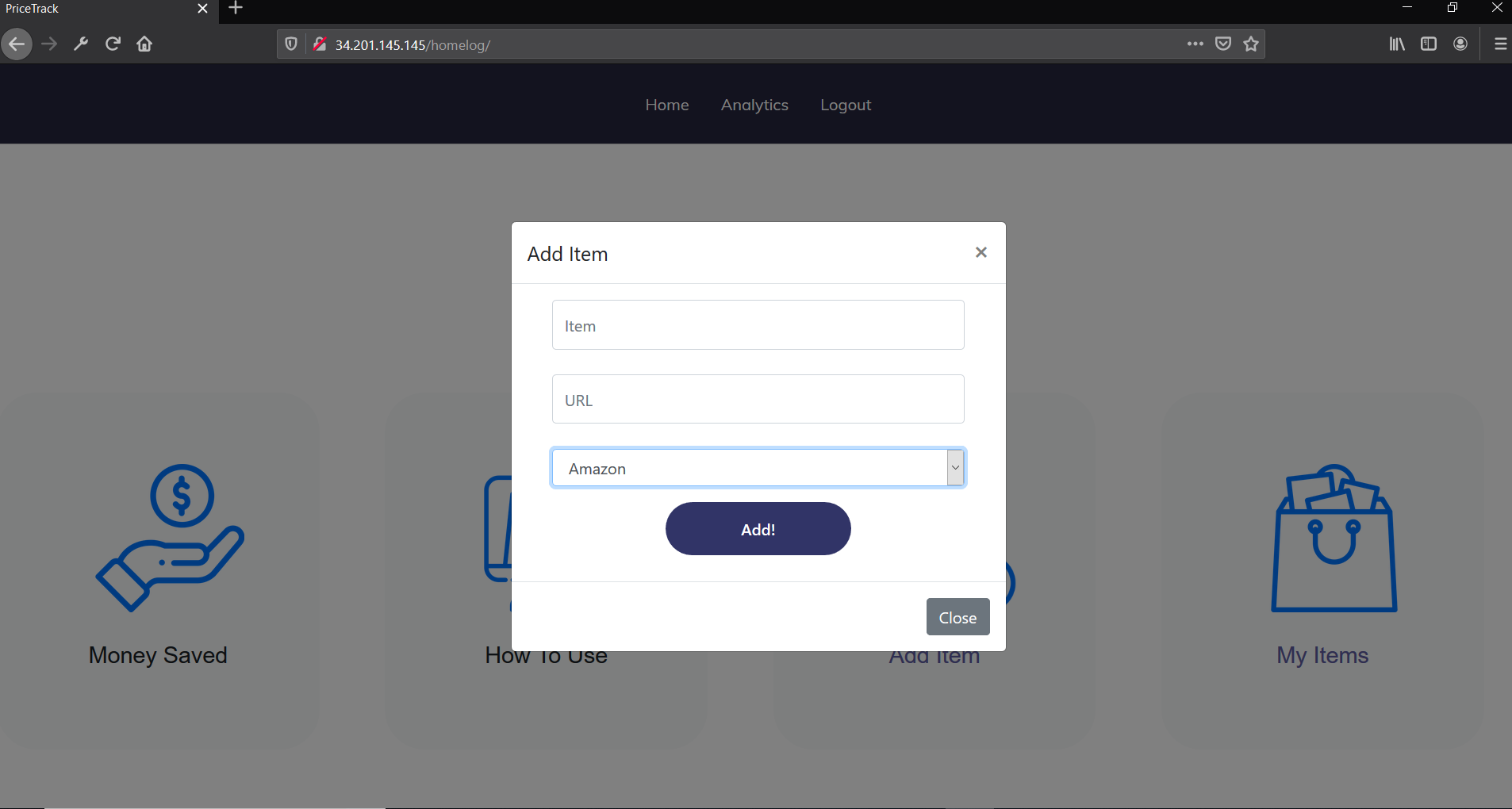


 Registration\_page





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

## <*This section is a summary of your testing report>*

## 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. >

## Tests for Functional Requirements

<Describe your test results for the functional requirements.

TODO: Provide a list of use cases or functions you have tested, as well as the testing results (whether or not the system passed the tests).>

Requirement1)

User is able to create a new account in the system by inputing 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 pesonalized account.

Test:

Result:

Requirement3)

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

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

Result:

Requirement4)

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

Test:

Result:

## Tests for Non-functional Requirements

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

## Hardware and Software Requirements

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

The tools our group utilized to perform test in this system consist of the Django administration Interface, and Django python shell. This project inplemented various Models which represented the information of an object which would then be store in the database. To test the implementiton and storage of this objects we utilized the Django administration as this provied us with an efficient and easy way of creating and saving a model object. Another tool we utilized for testing consits of the Django shell, this interactive sheel allowed to test the query of inforamtion from out models and prevented from having to write afull script. The last tools utilized for testing is called ……….. and this consits of.

# 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. >

# Conclusion

<Conclude the document with what you have learned through working on the project.>

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.>