CAL STATE FULLERTON

Department of Computer Science

This project has been satisfactorily demonstrated and is of suitable form.

This project report is acceptable in partial completion of the requirements for the Master of Science degree in Software Engineering.

RoboStock - Stock Price Forecasting Using Machine Learning

Project Title (type)

Carlos Gibson

Student Name (type)

Dr. Chang-Hyun Jo, Ph.D

Advisor's Name (type)

Advisor's signature Date

Dr. Ning Chen, Ph.D

Reviewer's Name (type)

Reviewer's signature Date

Abstract

RoboStock is a web application built on the Django Model-Template-View Framework that allows users to view same-day closing price predictions for a selected stock symbol. Traditionally day traders resort to pure speculation when placing bets on the stock market. There are other options for making trading decisions, such as paying expert individuals or well-established organizations to invest for the customer. Having a deep understanding of the markets can also provide investors a higher level of confidence than pure speculation alone. Algorithmic day trading is another option that has become increasingly popular in recent years. This option requires the investor to have not only a good understanding of the markets but also possess good computer science and programming skills that enable them to create automated trading programs. RoboStock aims to provide users an easy-to-use tool that does not require a deep understanding of the market for making day trading decisions. RoboStock uses the Long-Short Term Memory architecture, an artificial recurrent neural network, to predict closing prices for selected stocks based on historical time-series data. In addition to short-term price predictions, RoboStock aims to provide users a simple tool for viewing current market trends based on major market indexes and historical trends of selected stocks.

Keywords

(Architecture) Model Kind, Agile, Architecture View, Architecture Viewpoint, Atom Text Editor, Bootstrap, Cascading Style Sheets (CSS), conda, Django, Git, GitHub, GitHub Desktop, GitHub Repository, Google Charts, Hypertext Markup Language (HTML), Integrated Development Environment, Long-Short Term Memory (LSTM), Machine Learning, Model-Template-View (MTV), Recurrent Artificial Neural Network, Scrum, Scrum Master, Sprint, Sprint Burndown Chart, Sprint Planning, Sprint Retrospective, Sprint Review, unittest, Virtual Environment.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Problem Description

The recipe for being a successful investor in the stock market is quite simple, buy when prices are low and sell when prices are high to gain a return on these transactions [2]. However, market volatility and other factors make price trend predictions a daunting task. There are several ways investors navigate through the markets. The most prevalent market navigation tool is pure human speculation based on accounting books and a general 'gut feeling' about the company behind the stock. At times investors lack the general understanding of a company's accounting books and make decisions based on overall trends and general public opinion. Another way of investing in the markets involves paying another party who has a solid reputation in the financial sector. The subject-matter-expert individual or organization charges a fee for investing customers' money into securities such as stocks and bonds. *Algorithmic trading* is a trend that has become popularized in recent years [19]. This novelty has been adopted by well-established financial organizations, but also by day traders look to gain short-term returns from their trading hobby. One of the main advantages of algorithmic trading is the removal of the human emotion factor in trading decisions. However, algorithmic trading requires a deep understanding of how the markets work and computer science, programming, and coding knowledge. Mastering these skills allow hobbyists to code 'bots' who then trade for short-term profits.

The aforementioned options require an initial investment of either money (investment fees or a percentage of the gains) or knowledge to get started. Based on this, the need for a simple tool that can predict the closing price can be appreciated. A simple tool that does not require the user to have a deep understanding of the financial markets, computer science, programming, and coding. RoboStock aims to do just that. RoboStock is a tool that can provide investors a higher level of confidence than pure human intuition alone, a tool that is easy to use and does not require a lot of user input in order to provide some meaningful and reliable information to the user.

1.2 Project Objectives

At the core of the objectives lies the web application RoboStock. RoboStock uses the MVT (Model, View, Template) Django Framework [8] studied in CPSC 545 Software Design and Architecture course [5].

RoboStock will make use of a highly effective recurrent neural network (RNN) architecture called Long-Short Term Memory (LSTM), which is a type of gated RNN [3]. The LSTM architecture has shown promising results for being able to predict with some reasonable level of accuracy the stock prices given some historical time-series data [6]. The Keras [15] and Scikit-Learn [17] Python libraries are levered for coding the LSTM program.

In addition to developing the RoboStock web application, this project will demonstrate some of the core Software Engineering concepts, practices, and methodologies learned in this MSE

(Master's in Software Engineering) program. The following six objectives make up the overarching aim of this project.

- **Objective 1:** Develop a web application, RoboStock, using the MVT Django Framework.
- **Objective 2:** Develop and maintain a Software Requirements Specification (SRS) document throughout the project.
- **Objective 3:** Deliver a User Manual document.
- **Objective 4:** Deliver an Architectural Description conformant to ISO/IEC/IEEE 42010.
- **Objective 5:** Demonstrate proficiency in the Agile software development process Scrum by defining, sustaining, and executing a Scrum process.
- **Objective 6:** Deliver a Final MSE Report consolidating all work performed for this graduate project.

The achievement of Objective 1 is demonstrated through the RoboStock web application video demonstration and the source code present in the GitHub repository. The SRS mentioned in Objective 2 is showcased in Section 2. The User Manual document, as well as a short Administrator Manual, are presented in Section 7. An Architectural Description has been developed and is included in Section 2 of this report. Scrum proficiency is demonstrated through the Agile process definition and development implementation execution documented in Section 4. Objective 6 is reached through the delivery of this final project itself.

1.3 Development Environment

The RoboStock web application was developed on an iMac using macOS. The Atom text editor was used as the IDE (Integrated Development Environment) for programming. A conda virtual environment, `MyDjangoEnv`, was created in order to install and use the necessary Python libraries. Please see Table 1 for more details.

Computer	iMac (Retina 5K, 27-inch, 2019)
Processor	3 GHz 6-Core Intel Core i5
Memory	8 GB 2667 MHz DDR4
Graphics	Radeon Pro 570X 4 GB
Operating System	macOS Big Sur (version 11.2.1)
IDE	Atom Version 1.56.0 x64
Terminal Emulator	Terminal Version 2.11 (440)
Software Version Control	GitHub Desktop Version 2.7.2

Table 1: Development Environment, Hardware, and Operating System

Table 2 contains a list of the web development technologies used. This list also includes key Python libraries used for the LSTM program:

Technology	Description
HTML	Hypertext Markup Language for front-end development
CSS	Cascading Styling Sheet for front-end-development
Bootstrap	CSS styling open-source technology
JavaScript	Front-end programming language
JQuery	Front-end programming tool
Python	Open-source, object-oriented programming language
Pandas	Python library for data manipulation and analysis
NumPy	Python library, adding support for large, multi-dimensional arrays and matrices, in conjunction with a large set of high-level mathematical functions to operate on the data arrays
Keras	Open-source software library provides a Python interface for artificial neural networks. It acts as an interface for the TensorFlow library.
Scikit-learn (sklearn)	Free software machine learning library for Python.
Django	Web application framework using the MVT design patter
Git and GitHub	These software version control technologies will be used to track changes through the development of the RoboStock web applications
Google Charts	This technology will be used to render charts through HTML files visually.

Table 2: Web Development Technologies, Languages, Libraries, and Programming Languages

1.4 Operation Environment

RoboStock can be accessed through any web browser (Google Chrome, Safari, Firefox, Internet Explorer or Microsoft Edge). This means that the web application is agnostic to the hardware and even operating system it is launched on. RoboStock can launch on an Apple Mac, PC, Windows computer, and so forth. As long as the computer has a web browser installed and access to the internet, the web application will be reachable

2 Software Requirements Specification

Section 2 of this report serves as the final draft of the SRS (Software Requirements Specification) Document. This final SRS's core content is taken from the Software Requirements Specification example presented in the Software Requirements text by Karl Wiegers and Joy Beatty [20].

2.1 Overall Description

Section 2.1 contains an overall description of the SRS. It includes information about the product perspective or scope, the high-level product features, user classes, user characteristics, operating environment, design and implementation constraints, user documentation, and assumptions and dependencies.

2.1.1 Product Perspective (Scope)

The requirements described in this section pertain to the entire RoboStock web application. Outside of the scope lies the host or server service that will permanently house the web application and client-side software and hardware for accessing RoboStock.

2.1.2 High-Level Product Features

Table 3 contains the RoboStock four high-level product features.

FE-1:	The RoboStock web application shall be able to launch from any major desktop web browser, regardless of the underlying operating system or hardware.
FE-2:	The RoboStock application is accessible to the public, but a user must be logged in in order to access the ML Predictions page.
FE-3:	The RoboStock web application shall be able to display historical stock price data for any stock in the NYSE (New York Stock Exchange).
FE-4:	The RoboStock web application shall use some machine learning algorithm to predict the stock price based on past historical time series data.

Table 3: Product Features

2.1.3 User Classes and User Characteristics

The key stakeholders from a user class perspective are comprised of the front-end user and the website administrator.

- Front-end User:
  - Can access the website without being logged in, with the exception of the ML Predictions page.
  - Can interact with the entire website once they are logged in.
  - Can select different stocks and visualize a display of the historical stock price for the selected stock.
  - Can select a stock and trigger a machine-learning algorithm to predict the stock price based on some historical time series data.
- Website Administrator:
  - Can access the website's administrator page.
  - Can access the website's database content (Django Models) through the 'admin' page.

2.1.4 Operating Environment

See Section 1.4

2.1.5 Design and Implementation Constraints

Table 4 contains a shortlist of the design and implementation constraints. Not design and implementation constraints exist beyond these three.

IC-1:	The web application shall use the Model Template View design pattern. More specifically, it shall use the Django framework.
IC-2:	The front-end technologies to be used shall be HTML, CSS, JavaScript, Bootstrap (and JQuery if needed).
IC-3:	The back-end technologies and programming languages shall be SQLite and Python.

Table 4: Design and Implementation Constraints

2.1.6 User Documentation

Table 5 contains the list of documentation required to be delivered along with the RoboStock web application.

UD-1:	The web application shall be delivered with a user manual
UD-2:	The web application shall be delivered with an administrator manual

Table 5: User Documentation

2.2 External Interface Requirements

The following three sections contain details for external interface requirements: user interfaces, hardware interfaces, and software interfaces.

2.2.1 User Interfaces

Table 6 contains the four user interface requirements.

UI-1:	The website is accessible to users while not logged in. Only the Machine Learning prediction page is disabled if the user is not logged in.
UI-2:	A navigation bar shall be displayed at the top of the page,or the left or right edge vertically. The navigation bar shall provide navigation to all the pages on the website.
UI-3:	An icon on the upper right corner shall clearly indicate whether or not the user is logged in
UI-4:	Navigation to the ‘admin’ page shall be accessible only by an administrator.

Table 6: User Interfaces

2.2.2 Hardware Interfaces

No explicit hardware interfaces between the web application and client-side computer hardware are needed or need to be specified. Any hardware interfaces are taken care of by the Atom text editor, an Integrated Development Environment (IDEs) for development.

2.2.3 Software Interfaces

No explicit communication interfaces are required for the Django application to function. Besides the communication to the world-wide-web, internet. For communicating with Yahoo finance. Raw data from Yahoo Finance is obtained through Django's View component and manipulated using Python libraries, commands, and functions.

2.3 Use Cases

The use case diagram is shown in Figure 1, Section 2.3.1. Use case descriptions are derived from the use case diagram and are shown in Section 2.3.2.

2.3.1 Use Case Diagram

Figure 1 depicts the different use cases for both the front-end user, nd administrator user classes. The Use Case Diagram shows the interaction with Yahoo Finance for retrieving stock price data, as well as TensorFlow for performing the LSTM predictions. The interaction with TensorFlow passes through the Python library Keras. More details on this Python library are given in the Architectural Description (Section 3.8.2).

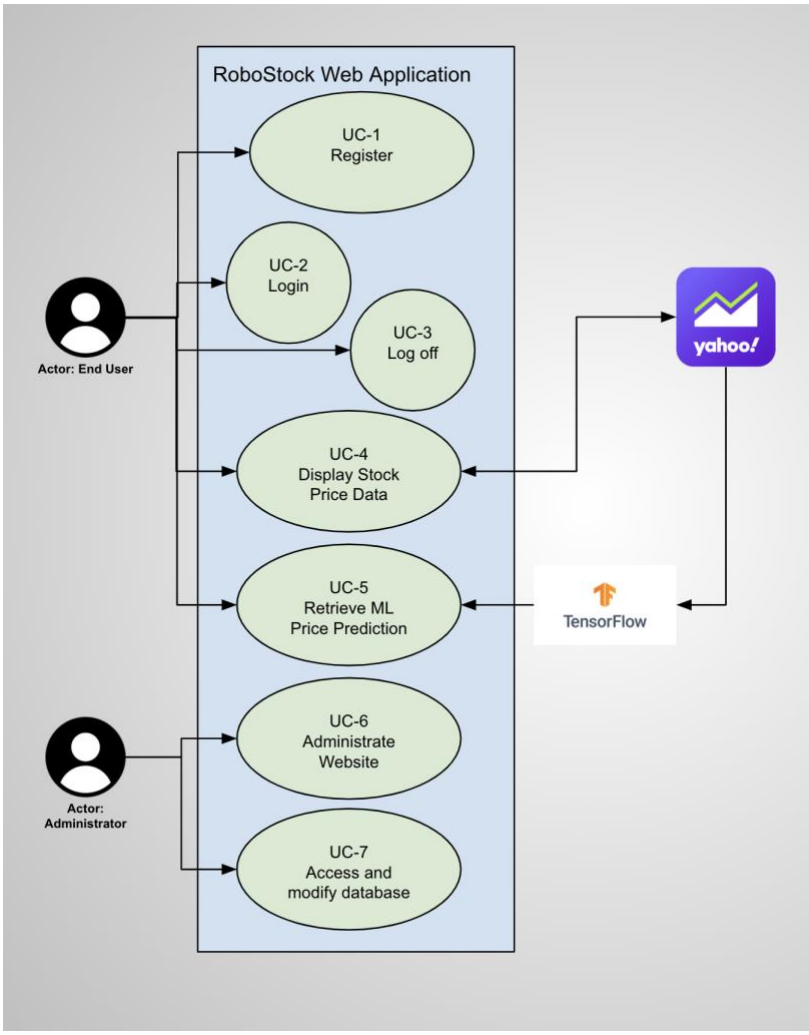

Figure 1: Use Case Diagram

2.3.2 Use Case Descriptions

ID	UC-1 [Version 2]
Name	User Registration
Description	The User can access the registration page by navigating to “login” > “Register.” The website is accessible without the need to log in with the exception of the “ML Predictions” page.
Primary Actor	End User
Secondary Actros	Super User (Administrator)
Trigger	• User visits the website by typing the site’s URL
Preconditions	• User has access to the internet and an internet browser
Postconditions	• User sees the registration page when the navigate to “login” > “Register.” • The user can access the entire website without being logged in or registered with the exception of the “ML Predictions” page.

Normal Flow	• User clicks on “login” in the upper right corner of the Navigation bar.
	• User clicks on the “Register” button below the login input fields.
	• User lands on the registration page.
	• User enters registration information:
	  ○ Username
	  ○ Password
	  ○ Email
	• User clicks “Register.”
	• Webpage returns a confirmation message.

Alternate Flow	None
Exceptions	Invalid Email Format • The email entered in the Django form must be of valid format
Includes	None
Extends	None
Priority	High
Assumptions	None
Notes and Issues	On February 26, this Use Case was revised and updated to Version 2. Summary of changes: • The navigation details for accessing the registration page were specified. The user is able to access the website, with the exception of the “ML Predictions” page without being logged in or registered. • Included Super Users, or Administrators, to the Secondary Actors • Registration Form input fields updated • Password requirements removed for simplicity.

Table 7: UC-1, User Registration Use Case Description

ID	UC-2 [Version 2]
Name	Login
Description	User logs into the website with the created credentials (email and password) from UC-1
Primary Actor	End User
Secondary Actros	Administrators
Trigger	• The user clicks on the ‘login’ Icon

	• The user enters a valid (registered) Username and Password combination and clicks the “Login” button.
Preconditions	• The user has the RoboStock website loaded on their browser • The user is not logged into the website
Postconditions	• The user successfully logs on
Normal Flow	• The user enters their username (email) • The user enters their password • The user is registered with the credentials entered • The user clicks the ‘Log on’, ‘Login’, or ‘Sign on’ button
Alternate Flow	None
Exceptions	• The user enters the wrong email and password combination • The user does not enter anything into the input fields
Includes	None
Extends	None
Priority	High
Assumptions	None
Notes and Issues	On March 8, this Use Case was revised and updated to Version 2. Summary of changes: • Added “the user does not enter anything into the input fields” as a possible exception. • Changed the ‘Log on’ phrase to ‘Login,’ which is more commonly used.

Table 8: UC-2, Login Use Case Description

ID	UC-3
Name	Log off
Description	User logs off from the website
Primary Actor	End User
Secondary Actros	Administrators
Trigger	• User clicks the ‘log off’ button
Preconditions	• User is on the RoboStock website • User has an internet connection • User is logged on
Postconditions	• User is no longer logged in • User does not have access to the ML Predictions page
Normal Flow	• User clicks the log off button
Alternate Flow	None
Exceptions	None
Includes	None
Extends	None
Priority	High
Assumptions	None
Notes and Issues	None

Table 9: UC-3, Log Off Use Case Description

ID	UC-4
Name	Display Stock Price Data
Description	User selects a stock quote from the NYSE and historical data for that stock is displayed.
Primary Actor	End User
Secondary Actros	Administrators
Trigger	• The user navigates to ‘Watchlist’ window

	• The user selects a quote from the NYSE • The user clicks ‘Display data’
Preconditions	• The user is in the RoboStock website • The user is signed on • The user is in the historical stock price page • The user has selected inputs to query data
Postconditions	• The user sees a line chart display showing historical data
Normal Flow	• The user selects a NYSE quote • The user selects a time range • The user clicks ‘Display data’
Alternate Flow	None
Exceptions	None
Includes	None
Extends	None
Priority	High
Assumptions	None
Notes and Issues	None

Table 10: UC-4, Display Stock Price Data Use Case Description

ID	UC-5
Name	Retrieve ML Price Prediction
Description	Forecast stock price for a given quote based on the historical time series data. The prediction is a day-to-day prediction and overlayed ontop of the historical data.
Primary Actor	End User
Secondary Actros	Administrators
Trigger	• The user navigates to the ML page • The user selects a stock quote from the NYSE • The user selects conditions for ML algorithm • The user triggers the ML algorithm
Preconditions	• The user is in the RoboStock website • The user is signed on • The user is in the ML stock price page • The user has selected inputs to query data
Postconditions	None
Normal Flow	• The user navigates to the ML page • The user selects a stock quote from the NYSE • The user selects conditions for ML algorithm • The user triggers the ML algorithm
Alternate Flow	None
Exceptions	•
Includes	None
Extends	None
Priority	High
Assumptions	None
Notes and Issues	None

Table 11: UC-5, Retrieve ML Price Prediction Use Case Description

ID	UC-6 [Version 2]
Name	Administrate Website
Description	Perform administrative tasks on the website
Primary Actor	Administrators

Secondary Actros	None
Trigger	• Administrator logs into the Django ‘Admin’ page
Preconditions	• The admin is in the RoboStock website • The admin is signed on
Postconditions	• The admin lands on the admin page
Normal Flow	• The admin sings onto the website • The admin link to the admin page is visible and accessible • The admin clicks on the link and is taken to the admin page
Alternate Flow	None
Exceptions	• The user is not an admin • The user cannot access the /admin page • The user does not have access to the admin page
Includes	None
Extends	None
Priority	High
Assumptions	None
Notes and Issues	On March 22 this Use Case was updated in order to remove the requirement for an explicit button for the admin page. Instead, there should be no admin page visible to the end users who are not administrators. The admin page should only be accessed by the URL http://127.0.0.1:8000/admin/

Table 12: UC-6, Administrate Website Use Case Description

ID	UC-7
Name	Access and modify database(s)
Description	Access the website’s administrator’s page and view/access/modify the Django Models (database)
Primary Actor	Administrators
Secondary Actros	None
Trigger	• Admin accesses the admin page and modifies the pre-existing Django models (database)
Preconditions	• The user interacting with the website is an admin • The admin is in the Django admin page
Postconditions	None
Normal Flow	• The admin is in the Django admin page • The admin can view and modify the Models within the admin page
Alternate Flow	None
Exceptions	• The user is not an admin and does not have access to the admin page
Includes	None
Extends	None
Priority	High
Assumptions	None
Notes and Issues	None

Table 13: UC-7, Access and Modify Database(s)

NOTE: These uses cases are translated into User Stories for Scrum Agile software development defined in Section 4.1 and represented as User Stories through the Product Backlog and Sprint Backlogs in Section 4.2.

2.4 Functional Requirements

Table 14 is a comprehensive list of the functional requirements for the RoboStock web application. These twelve functional requirements are extracted from the Use Cases and are meant to represent the specific function requirement of the application.

FR-1:	The website shall not allow users to interact unless they have successfully signed on
FR-2:	The website shall register users using a valid email address and password
FR-3:	The website shall encrypt the passwords used to register
FR-4:	The website shall display the historical price trend as a line chart
FR-5:	The website shall allow users to select any stock quote from the NYSE
FR-6:	The website shall not allow users to select stock quotes not in the NYSE
FR-7:	The website shall use a separate window for historical stock price and ML predictions
FR-8:	The website shall allow users to log off from any window
FR-9:	The website shall provide a navigation bar allowing users to navigate anywhere on the site
FR-10:	The website shall restrict access to the admin page to admin accounts only
FR-11:	The website shall allow users to select a time range to query from
FR-12:	The website shall allow users to select the date range for training the ML algorithm and for testing the algorithm

Table 14: Functional Requirements

2.5 Nonfunctional Requirements

There are several quality attributes discussed by Bass *et al* in the text *Software Architecture in Practice, 3rd Ed*. [1]. Some of the quality attributes discussed included availability, security, modifiability, and performance. The following nonfunctional requirements pertain to the quality attributes mentioned above.

NFR-1:	The website shall be available 99.9996% of the time
NFR-2:	The website shall use password encryption for security
NFR-3:	The website shall be easy to modify (easy to maintain)
NFR-4:	The website shall have a latency of less than 30 seconds for ML stock price prediction.

Table 15: Nonfunctional Requirements

3 Architectural Description

The Following Architectural Description is conformant to ISO/IEC/IEEE 42010 Clauses 5 and 7 [12].

3.1 System-of-Interest

RoboStock is the system-of-interest for whose architecture this architecture description is written. The RoboStock overarching architecture follows the Model-Template-View design patter [8].

3.2 Architecture Description Identifying Information

The identifying information for this architecture description is shown in Table 16.

Date of issue	**April 24, 2021**
Status	Final Draft, Pending Approval
Author	Carlos Gibson
Advisor	Professor Chang-Hyun Jo, Ph.D.
Reviewer	Professor Ning Chen, Ph.D.
Approving Authority	California State University, Fullerton - Computer Science Faculty
Issuing Organization	California State University, Fullerton

Table 16: Architecture Description Identifying Information

3.3 Scope

The scope of this Architecture Description pertains to the architecture exhibited by the RoboStock Django Web application Framework (system-of-interest). The concerns regarding the architecture and system, and the typical stakeholders sharing such concerns, are listed in Section 3.5. Other specific web applications running on this framework are not in the scope of this architecture description. Only the RoboStock web application is within the scope of the architectural description.

3.4 Architecture and Architecture Description Evaluation Results

When preparing this document, no evaluations have been made to the architecture or its architecture description by the author. The author has not made evaluations on behalf of the issuing organization, California State University, Fullerton. Therefore, no results from evaluations are available yet. Evaluations and their respective results are TBD (to be determined).

3.5 Identification of Stakeholders and Concerns

The following sections identify the stakeholders and the concerns they poses with regards to the system-of-interest.

3.5.1 Stakeholders

The following are system stakeholders with fundamental concerns about the architecture of the system-of-interest:

- **Users of the system** - any individual or group of individuals who interacts with the internet web page built on the Django Framework, RoboStock.
- **Operators and maintainers of the system** - any individual or group of individuals who operate, maintain, sustain, and modify or make improvements to a web application built on the Django Framework, RoboStock.
- **Acquirers of the system** - any individual, group of individuals, or organizations who purchase a web application built on the Django Framework, RoboStock.
- **Owners of the system** - any individual, group of individuals, or organization(s) who owns the system, financially or otherwise.
- **Suppliers of the system** - any individual or organizations supplying services, computational resources, hardware, and any other service or product that the system-of-interest uses to be developed or operated. Any individual or organization supplying services, computational resources, hardware, and any other service or product that interfaces with the system of interest (e.g., databases, web hosting services, etc.)
- **Developers and builders of the system** - any individual or group of individuals who develop, design, build, document, architect, deploy the web application using the Django Framework; specifically, the RoboStock web application.

3.5.2 Concerns

The following concerns are considered to be fundamental to the architecture of the system-of-interest:

- Purpose: The Django Framework aims to provide a high-level, open-source, python-based framework for the development of web application. The purpose of RoboStock is to provide users with a simple, easy-to-use tool for navigating the stock market and making short term decisions.
- The architecture's suitability for achieving the system's purpose has proven to be effective through its popularity and common use in web development.
- There is a high level of feasibility for constructing and deploying web pages running on the Django framework.
- The following are some potential risks and impacts of the system to its stakeholders:
    - Although the system has robust security measures, there is always the possibility of risk when it comes to security.
    - Technical difficulties implementing, developing, and maintaining websites using the Django Framework as a foundation are also possible.
    - The RoboStock web application, although built on the Django framework, uses very specific and unique python libraries for conducting the machine learning algorithm, the two main Python libraries are Keras and Sklearn. Technical difficulties working with these libraries pose a risk.
    - Another technology used for the visual rendering of the charts is the Google Charts API [9]. This is a unique technology that poses technical risks when integrating it within the Django framework. More specifically, the integration of Google Charts [9] within the data transfer interaction between the Views and

Template requires an understanding of both the Google Charts [9] technology and the Django framework.

- The Django Framework offers a high level of maintainability and evolvability.
- Other fundamental concerns to the architecture of the Django Framework are:
    - **Availability** – this concern and associated risks are dependent on the hosting service, the hosting service infrastructure, and other factors outside of the scope of this report. The availability concern within the scope of this architectural description is discussed in terms of software related issues (bugs, faults, errors, error-handling, etc.), and how they are mitigated through the design.
    - **Performance** – performance is another quality attribute that is dependent on factors other than the software program(s) themselves. Performance can be impacted by the host's infrastructure, as well as client-side factors (internet speed, client machine performance, etc.)
    - **Modifiability** – The system-of-interest inherently has a high degree of modifiability. This is thanks to the Model-Template-View framework exhibited by the Django framework. The separation of these components allows for high cohesion, and low coupling. Modification to the components can impact the system as a whole, but the risk of impact is minimized by the separation of functionality between the Model, the Template, and the View.
    - **Usability** – This quality attribute concern is addressed through a simplistic and elegant design. The site has a uniform look and feel across all its pages. By using a base.html, the templates inherit the base's extension across all the other HTML templates. The website requires little to no user input in order to display desired information. The website shows a simplistic view that does not overwhelm users.
    - **Security** – Django offers a wide range of up-to-date security features that can be appreciated through the Django's extensive and thorough documentation [8].

3.6 Concern and Stakeholder Association

The following table, Table 17, depicts the relationship between the aforementioned concerns and stakeholders.

Concern	Stakeholders
System's Purpose	• Users • Operators and maintainers • Acquirers • Owners • Suppliers • Developers and builders
System's Suitability	• Operators and maintainers • Acquirers • Owners • Developers and builders
System's Feasibility	• Developers and builders • Acquirers • Owners

System’s Security Risk	• Users • Operators and maintainers • Acquirers • Owners • Suppliers • Developers and builders
System’s Technical Difficulties Impact	• Operators and maintainers • Suppliers • Developers and builders
System’s maintainability and Evolvability	• Operators and maintainers • Acquirers • Owners • Developers and builders
Availability	• Users • Operators and maintainers • Acquirers • Owners
Performance	• Users • Operators and maintainers • Acquirers • Owners
Modifiability	• Operators and maintainers
Usability	• Users

Table 17: Concerns and Stakeholder Association

3.7 Architecture Viewpoints

There are three architecture viewpoints summarized in this description: the structural viewpoint, the file directory viewpoint, and the data flow viewpoint. These viewpoints contain an overview, a list of concerns and anti-concerns, typical stakeholders concerned or interested in the viewpoint, and the model kinds.

3.7.1 Structural Viewpoint

3.7.1.1 Viewpoint Overview

This viewpoint establishes the conventions for the construction, interpretation and use of the structural architecture view which frames high-level concerns with regards to the component-and-connector architecture of the system-of-interest, RoboStock.

3.7.1.2 Concerns (and anti-concerns)

The concerns framed by this architecture viewpoint are the following:

- The system’s purpose
- The system’s suitability
- The system’s feasibility

- The system’s security risk
- The system’s technical difficulties impact
- The system’s maintainability and evolvability
- The system’s availability
- The system’s performance
- The system’s modifiability
- The system’s usability

This viewpoint is intended to answer the following concerning questions:

- What is the general structure of the RoboStock web application?
- What is the suitability of the RoboStock web application with regard to its purpose?
- What is the feasibility of the RoboStock web application for realizing its purpose?
- What is the general component-and-connector pattern Django exhibits?

The following is a list of anti-concerns or concerns not framed by this viewpoint.

- Concerns regarding the allocation or computational resources
- The hardware requirements for clients or servers

3.7.1.3 Typical stakeholders

The following is a list of typical stakeholders for concerns framed by this viewpoint:

- Users
- Operators and maintainers
- Acquirers
- Owners
- Suppliers
- Developers and builders

3.7.1.4 Model kinds

Only one model kind is used for the architecture viewpoint “Overall Architecture Viewpoint.” The name of this model is “Overall Architecture Diagram.” The “Overall Architecture Diagram” uses the informal language, notation, and convention (diagram was made using Google Slides).

3.7.2 File Directory Structure Viewpoint

3.7.2.1 Overview

This low-level viewpoint establishes the conventions for the construction, interpretation and use of the associated File Directory Structure View to frame specific system concerns.

3.7.2.2 Concerns

The concerns framed by this architecture viewpoint are the following:

- The system’s functionality and maintainability
- The system’s file structure
- The system’s technical difficulties impact
- The system’s maintainability and evolvability

3.7.2.3 Typical Stakeholders

The following is a list of typical stakeholders for concerns framed by this viewpoint:

- Operators and maintainers
- Suppliers

- Developers and builders

3.7.2.4 Model Kinds

Only one model kind is used for the architecture view governed by “File Directory Structure Viewpoint.” The name of this model is the “File Directory Structure Diagram.” The File Directory Structure Diagram” uses the informal language, notation, and convention.

3.7.3 Data Flow Viewpoint

3.7.3.1 Overview

This viewpoint establishes the conventions for the construction, interpretation and use of the architecture view “Data Flow View” to frame specific system concerns.

3.7.3.2 Concerns

The concerns framed by this architecture viewpoint are the following:

- The system’s functionality and maintainability
- The system’s technical difficulties impact
- The system’s maintainability and evolvability
- The system’s security and security risks
- The system’s performance
- The system’s usability

3.7.3.3 Typical Stakeholders

The following is a list of typical stakeholders for concerns framed by this viewpoint:

- Operators and maintainers
- Suppliers
- Developers and builders

3.7.3.4 Model Kinds

Only one model kind is used for the architecture view governed by “Data Flow Viewpoint.” The name of this mode is “Data Flow Diagram.” The “Data Flow Diagram” uses the informal language, notation, and convention (diagram was made using Google Slides).

3.8 Architecture Views

3.8.1 Structural View

This view is governed by the Structural Viewpoint outlined in Section 3.7.1. The following architecture model is version 1.0. Its governing model kind is the Structural Viewpoint outlined in Section 3.7.1.4. The model in Figure 2 is governed by the informal model kind.

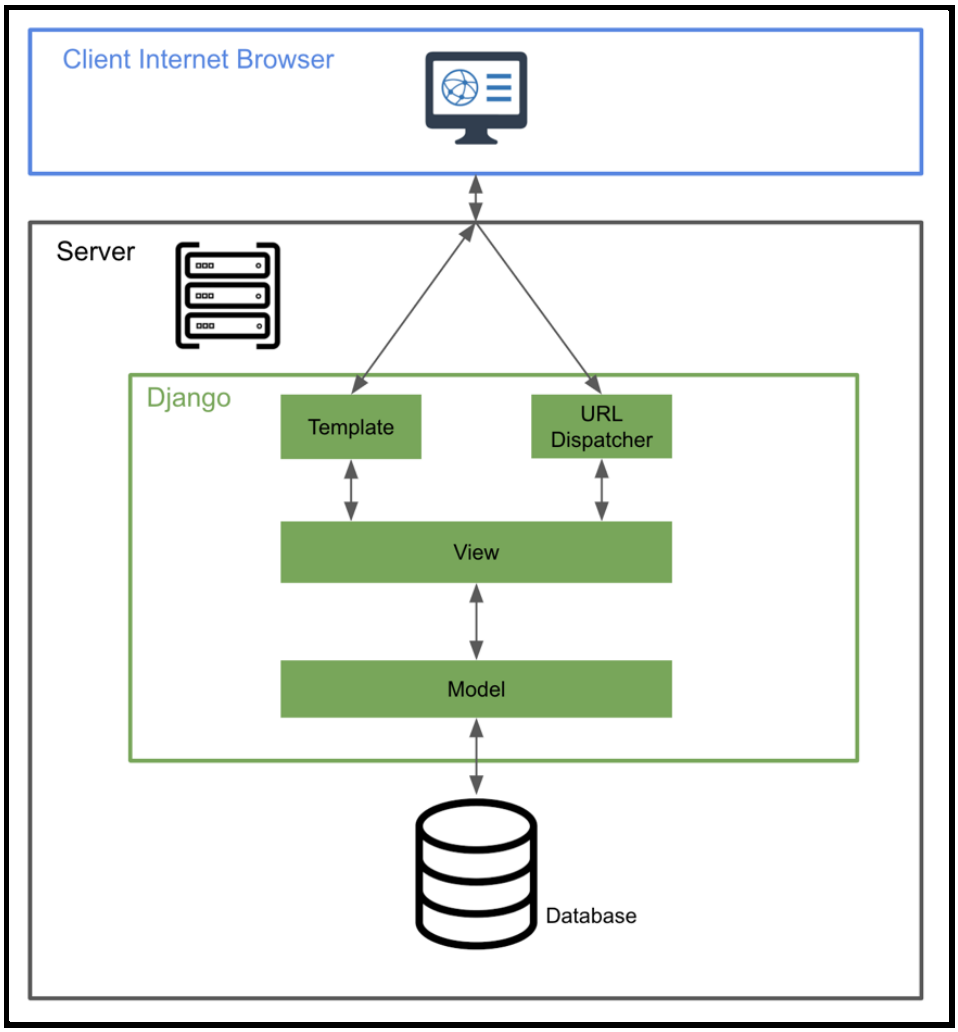

Figure 2: Architecture Model - Structural View, governed by the Structural viewpoint, v1.0

- The purpose of this view is to outline the typical structural layout for Django. This typical layout is inherited by the RoboStock web application, since it is built upon the Django framework.
- Django makes use of the MVT (Model-View-Template) design pattern.
- Django resides in a hosting service and interacts with the outside world through typical internet communication protocols (HTTP & HTTPS).
- A URL dispatcher serves as a router for the different web pages, which are then handled by the View. The View then returns a render of the Templet pertaining to the URL's request.
- The Model serves as a database.
- The View interacts with the Model in order to fetch and provide data.
- The Model serves as the data access layer and performs read and write actions against the database. The Model also serves as a database constructor.
- Templates are then used in order to render the information and present the webpage to the client's browser.

3.8.2 File Directory Structure View

This view is governed by the File Directory Structure Viewpoint outlined in Section 3.7.2. The following architecture model is version 1.0. Its governing model kind is the File Directory

Structure Viewpoint Model Kind outlined in Section 3.7.2.4. The model in Figure 3 is governed by the informal model kind.

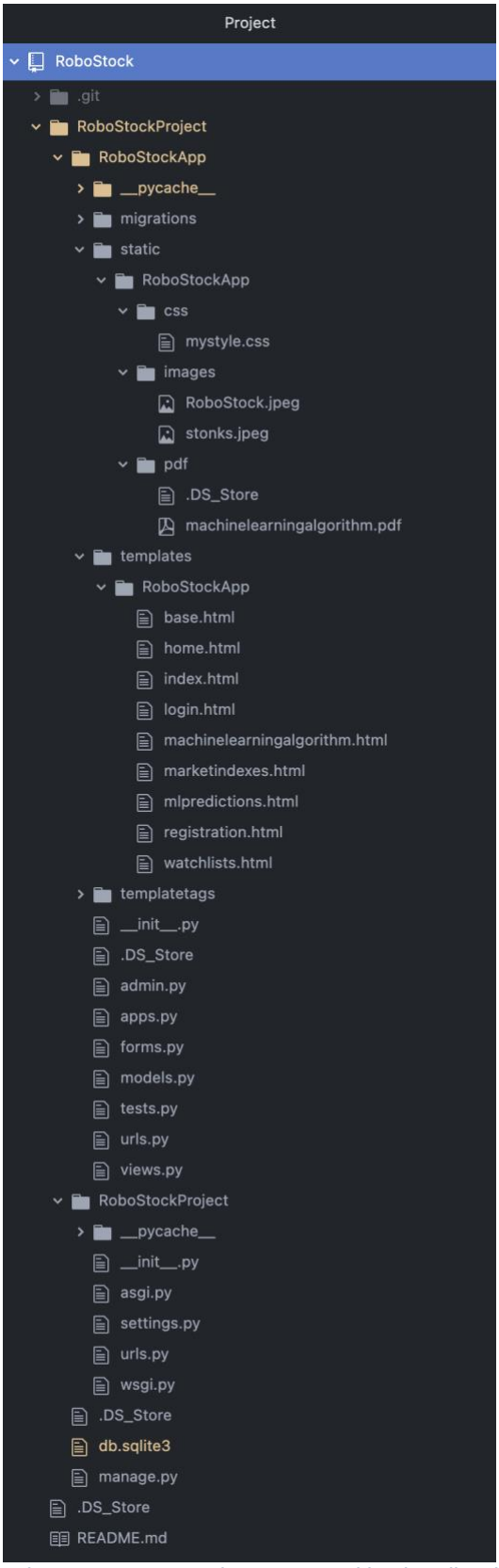

Figure 3: Architecture Model - File Directory Structure View, governed by the File Directory Structure Viewpoint, v1.0

- Figure 3 depicts the architecture view showing the RoboStock Project file directory structure.
- Templates - the templates are HTML files, and they are located in subfolders with the name of the application. In this case, it is templates > RoboStockApp.
- There are two URL dispatchers, the RoboStockApp > urls.py is the application's URL dispatcher. The RoboStockProject > urls.py is the project's URL dispatcher for the entire project and contains a URL path to the admin page, it also extends to the URL paths within the RoboStockApp > urls.py dispatcher.
- Views.py is the View component of the MVT pattern. This has all the logic and performs the data computations that are passed through to the Templates. The views return a response and data is embedded within the dictionaries.
- Models.py is the data access layer component of the MVT pattern
- For information on the file structure, see the Django documentation at https://www.djangoproject.com/ [8]

3.8.3 Data Flow View

This view is governed by the Data Flow Viewpoint outlined in Section 3.7.3. The following architecture model is version 1.0. Its governing model kind is the Data Viewpoint outlined in Section 3.7.3.4. The model in Figure 4 is governed by the informal model kind.

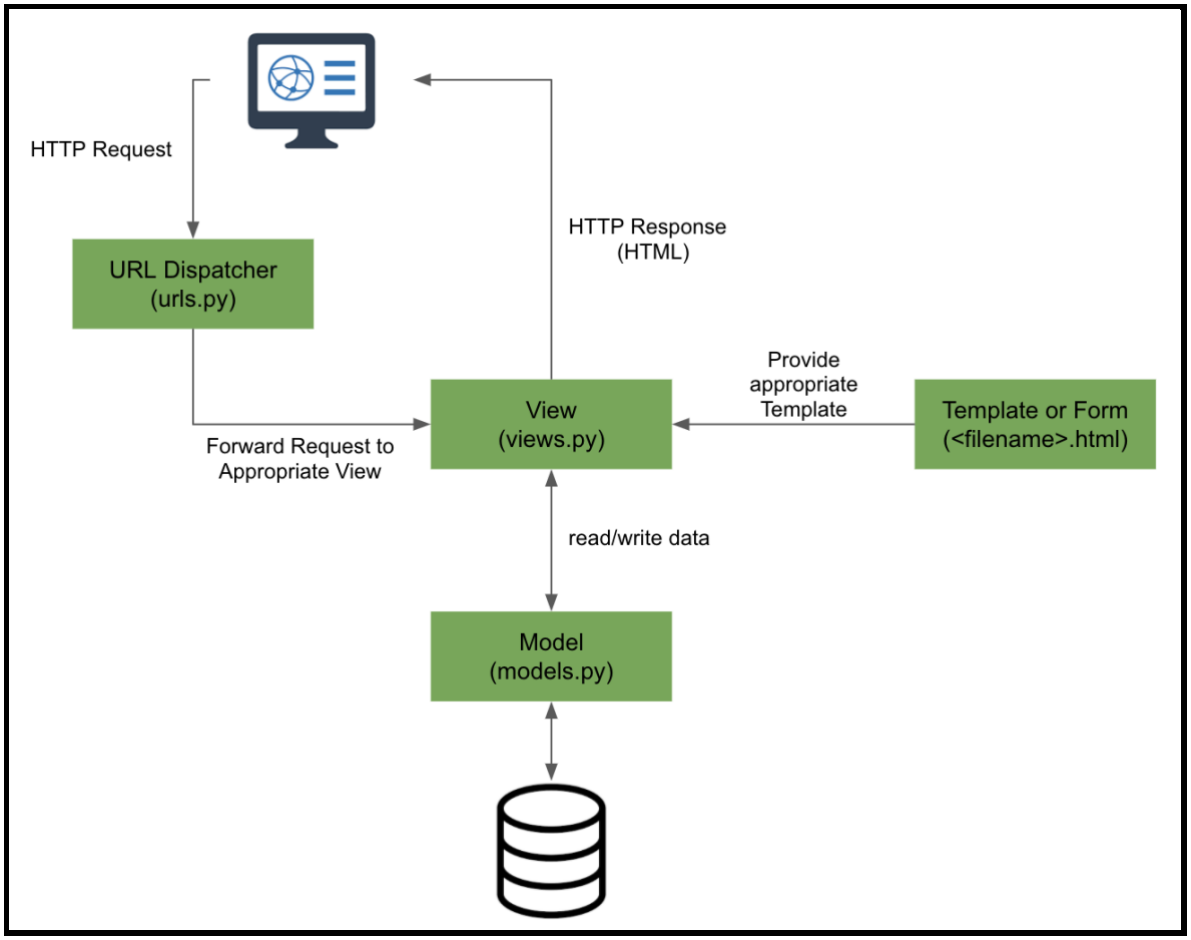

Figure 4: Architecture Model - Data Flow View, governed by the Structural Viewpoint, v1.0

- HTTP requests are initiated by the browser and are executed through request middleware.

- The URL dispatcher routes the request and forwards this to the appropriate view
- The view then orders read/write commands against the Model, which can interface with a multitude of databases.
- The view also uses templates or forms to render data and static elements on the web page.
- The HTTP response is then passed to the web client in the form of an HTML page.

3.8.4 Technologies Data Flow View

This view is governed by the Structural Viewpoint outlined in Section 3.7.3. The following architecture model is version 1.0. Its governing model kind is the Structural Viewpoint outlined in Section 3.7.3.4. The model in Figure 5 is governed by the informal model kind.

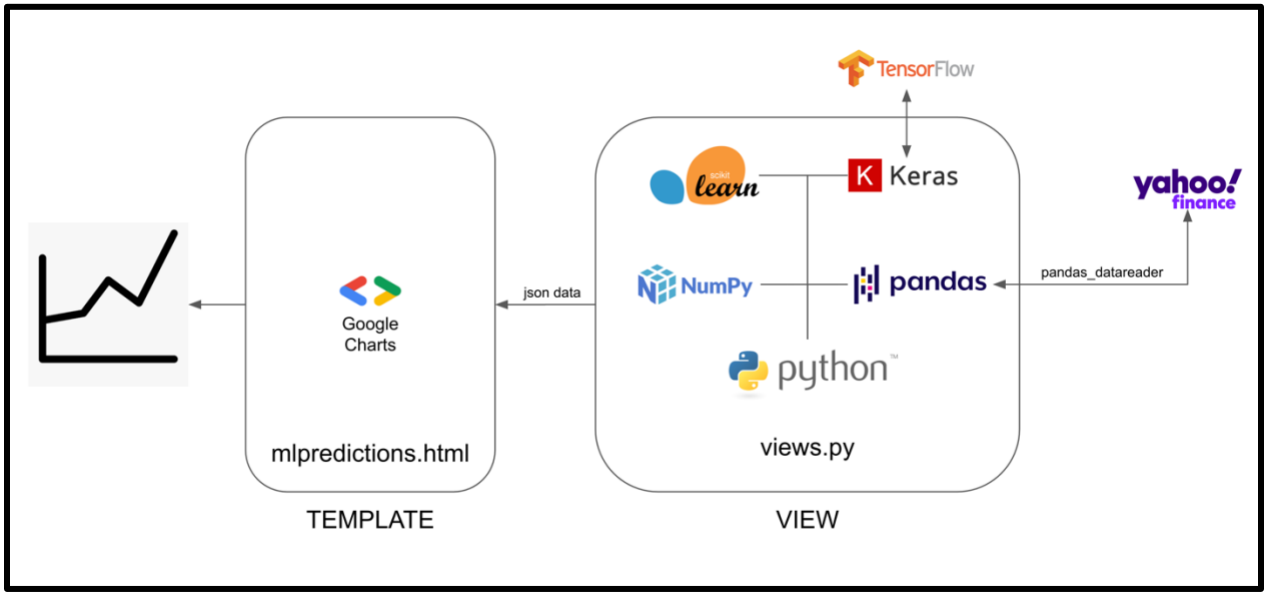

Figure 5: Architecture Model – Technologies Structure View, governed by the Structural Viewpoint, v1.0

- The purpose of this architecture model is to represent the key technologies used for the machine learning algorithm program displayed in the RoboStock web application.
- The two key components from the Django Model-Template-View framework are the View (views.py file) and the Template (html files, the mlpredictions.html file in particular).
- Data for the ML algorithm training is queried through the web form Yahoo Finance [21] using pandas_datareader.
- The key ML libraries used are Sklearn and Keras. Sklearn is used for data processing while Keras is used as a “middleman” between the Python program and TensorFlow (an ML library).
- Pandas and NumPy are used for data analysis and manipulation
- The data is passed to the Template as a JSON object. The view returns the data in the form of a dictionary.
- The Template uses this data through the use of Template Tags. Google Charts API is used to render the data visually and represent it to the end user.

3.9 Architecture Relations

3.9.1 Consistency with an architecture description

No inconsistencies were identified across the architecture models and views of this architecture description. All information depicted across the three models and three views is consistent. The informal model kind allows for the creation of meaningful models with valuable information without being constrained to some annotation formality (UML, ADL).

Analysis of consistency between the models and views was executed. The analysis revealed that the models and views are consistent across. They all revolve around the Mode-View-Template pattern. This pattern is expressed in all three views consistently.

3.9.2 Correspondences

The correspondence between architecture viewpoints and architecture views are outlined in the following table, Table 18. The relation here is governance of the viewpoint over the view.

Architecture Viewpoint	Corresponding Architecture View
Structural Viewpoint	Structural View
File Directory Structure Viewpoint	File Directory Structure View
Data Flow Viewpoint	Data Flow View
	Technologies Data Flow View

Table 18: Architecture Viewpoint and Architecture View Correspondence

The correspondence between concerns and stakeholders having those concerns is outlined in Table 18 above.

3.9.3 Correspondence rules

- Viewpoint and view correspondence rule: Viewpoints govern viewpoints
- Model kind and model correspondence rule: the model kind governance over its model, and the viewpoint governance over its view.
- Stakeholder and concern correspondence rule: Stakeholders have concerns with regards to the system-of-interest, the RoboStock web application.

3.10 Architecture Rationale

3.10.1 Rationale recording

The Model-View-Template architecture was chosen due to its effectiveness at separating concerns. The components of the pattern separate functionality while maintaining sound communication and connection. The MVT pattern was chosen over its predecessor, the MVC pattern, in order to make the pattern suitable for Web application development. The chosen technologies and data flow expressed in the Technologies Data Flow View were chosen for their web development suitability and machine learning programming effectiveness.

3.10.2 Decision Recording

The following major decision was recorded in accordance to ISO/IEC/IEEE 42010:2011(E) Clause 5.8.2.

3.10.2.1 Decision to use the Django Model, Template, View architecture

The following table, Table 19, contains a summary of the major decision to use the Model, Template, View design pattern offered by the open-source Django Framework:

Unique ID	ROBOSTOCKDECISION001
Decision Statement	Use the Django Model, Template, View design pattern for the overarching architecture of the RoboStock web application.
Pertaining Concerns	• The system's functionality and maintainability • The system's technical difficulties impact • The system's maintainability and evolvability • The system's security and security risks • The system's performance • The system's usability
Decision Owner	Carlos Gibson
Affected AD Elements	All viewpoints and views
Rationale	The Django MTV design pattern was chosen for the following reasons: • It is open-source and does not impose a financial cost. • It is fully supported by the Django Software Foundation. • It uses Python as the programming language. Python is a ubiquitous programming language. • It is a new framework learned in this program. I good opportunity to expand programming skills.
Constraints and assumptions influencing the decision	The following are constraints that influenced the decisions: • Commitment to using the Django framework was made through the initial project proposal. • Lack of skills, knowledge, and experience with other web application frameworks. The following assumptions were made: • The Django Software Foundation would continue support throughout the Lifecyle of this project.
Alternatives	1. Ruby on Rails – Programming language is Ruby, zero experience programming with Ruby. 2. React – difficult to learn, built on JavaScript (not a lot of experience with JavaScript).
Decision consequences	The consequence of choosing Django over Ruby on Rails or React (the other two alternatives) is that the learning curve of learning a new web development framework and programming languages is drastically decreased. Previous experience both with Django and Python made the choice an easy one.
Timestamp	Friday, June 26, 2020
Relevant citations	[5], [8]

Table 19: Decision to use the Django Framework for the RoboStock web application

4 Development

The software process for developing RoboStock used a Scrum process definition. The Scrum process is defined in Section 4.1. The Scrum process was simulated throughout the development of the application. The work products and the progress tracking of each sprint are described in Section 4.2. The entirety of this section is dedicated to the definition and simulation of the Agile software development process Scrum to underscore the importance and ubiquitous proliferation of Agile practices in the industry. Several sources served as a guide for defining and simulating this Scrum process [7], [10], [13], and [14].

4.1 Scrum (Agile) Process Definition

The software development process will follow a traditional Scrum (Agile) Process. The Sprint durations will be two weeks long over the sixteen weeks of the Spring 2021 Semester for a total of eight sprints. Each sprint will be kicked off with a Sprint Planning meeting, where User Stories from the product backlog will be chosen to be implemented during the upcoming sprint.

4.1.1 Scrum Team

Since this is an individual graduate project, the scrum team will only be composed of one individual (myself, Carlos Gibson). However, in a typical Scrum team, around seven team members will compose the Scrum team.

4.1.2 Scrum Master

The scrum master's role is to ensure that the team has all the resources they need to be successful. They act as barrier from outside distractions.

4.1.3 Product Owner

The product owner is the customer representative and liaison between the customer and the development team. They provide critical input on user stories and feature prioritization; and play an essential role in the Sprint Planning and Sprint Review meetings.

4.1.4 Development Team

The remainder of the team works on developing the RoboStock website.

4.1.5 Sprints

As stated above, the sprint duration will be two weeks long throughout the entire semester. Below is a visual representation of what a sprint would look like during each week (Figure 6).

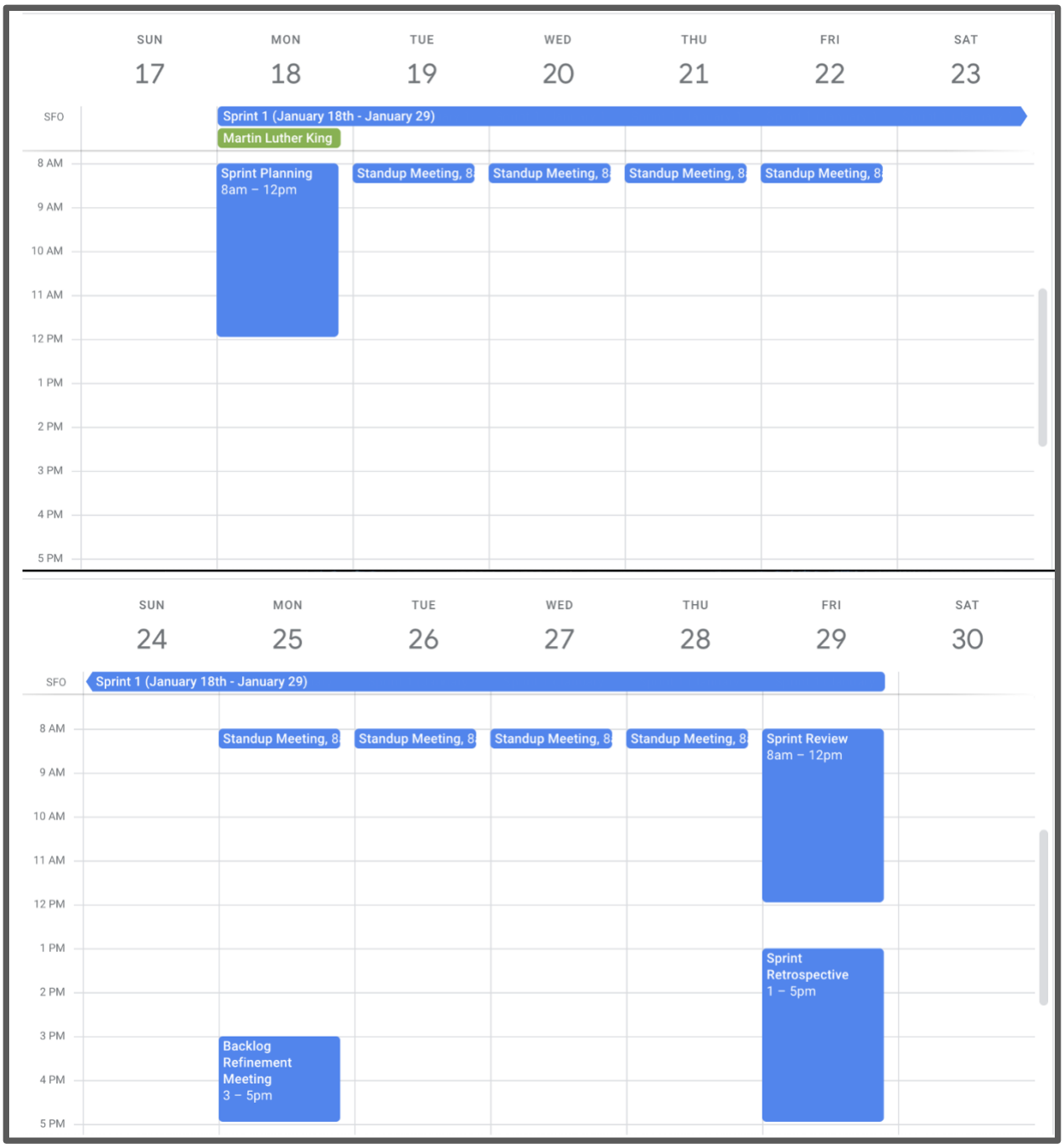

Figure 3: Typical Sprint Cadence Example

4.1.6 Daily Standup Meetings

During the daily standup meetings, the Scrum team discusses what they did the day before (if applicable), what they plan on doing today, and what obstacles they are facing. The move features from the user stories from the to-do bucket to the work-in-progress bucket, and finally to the completed bucket.

4.1.7 Sprint Planning

During the sprint planning meeting, the Product Backlog is analyzed, and User Stories are taken and put into the Sprint Backlog list. Typically, the User Stories with the highest priority (most important to the customer of the product owner) are taken first.

4.1.8 Sprint Review

Sprint Review meetings are used by the Scrum team to demonstrate working software to the customer(s) and/or product owner. They receive direct feedback from the customer(s) and/or product owner.

4.1.9 Sprint Retrospective

During the Sprint Retrospective meeting, the Scrum team alone meets together, and they discuss the sprint that they just closed. The discussions focus on what went well, what went wrong, and how to improve.

4.1.10 Backlog Refinement Meeting

This biweekly meeting was hosted by the product owner. During the Backlog Refinement Meeting, the Product Backlog Items (PBI) are refined and prioritized. Priorities are changed according to the most up-to-date requirements, and items are added or removed based on the customer’s and Product Owner's advice.

4.1.11 Scrum Work Products

The documentation required for the typical Scrum software development process can range from ceremonial to loosely defined. Besides the work products defined in this report (requirements specification, architectural description, etc.), typical Scrum work products will also be tracked. These work products include the product backlog, the sprint backlog, and the burndown chart.

4.1.12 Product Backlog

The product backlog contains all of the User Stories, ranked with respect to priority (value-added).

4.1.13 Sprint Backlog

The sprint backlog is a subset of the product backlog and is formulated by the Scrum team at the beginning of each sprint. Items from the product backlog are taken in order from highest priority to lowest priority. The Scrum team commits to working on these items only during the current sprint. They do not expand their commitment to additional User Stories once the sprint begins.

4.1.14 Burndown Chart

The burndown chart shows the number of stories committed for completion and the number of actual user stories committed as line charts overlayed on each other.

4.2 Scrum (Agile) Process Execution and Implementation

The web application development occurred between the beginning of the semester, on Monday, January 18th, 2021, and culminated on the project's due date on Thursday, April 29th, 2021.Between these start and end date, there were a total of six sprints or development iterations. The Scrum process defied on Section 4.1 was not simulated through a team environment due to the individualistic nature of this graduate project. The master's in software engineering graduate candidate played the Scrum master, development team, and product owner role. The following sections contain a summary of the Scrum sprint iterations. Figure 7 contains the final version of the Product Backlog

ID	USER STORY (Product Backlog Item or PBI)	Priority	Category	Status	Acceptance Criteria	Story Points
US-1	A user shall be able to register to the website	P0	User Story	Complete	Successful user registration	25
US-2	A user shall be able to login to the website	P0	User Story	Complete	Successful user Login	20
US-3	A user shall be able to log off the website	P0	User Story	Complete	Successful user logoff, user cannot access the ML Predictions page	10
US-4	A user shall be able to see historical stock price data	P0	User Story	Complete	User selects a stock symbol and historical stock price data for the selected dates is displayed.	50
US-5	A user shall be able to see ML stock price prediction	P0	User Story	Complete	User selects a stock symbol and prediction stock price data for the selected dates is displayed.	100
US-6	An administrator shall be able to administer the website	P0	User Story	Complete	The built-in Django admin page is accessible to administrators only.	10
US-7	An administrator shall be able to modify database through the admin page	P0	User Story	Complete	An administrator can add or remove users from the Users model through the admin page.	10
US-8	A user shall be able to see major stock indexes recent history	P1	User Story	Complete	Historical trends for top stock indexes is displayed	50
US-9	A user shall be able to perform a Google search from the website	P1	User Story	Complete	A user can type something on a search bar and trigger a Google search.	20
US-10	A user shall be able to access Yahoo Finance through the website	P1	User Story	Complete	A user can click on a Yahoo Finance link and navigate to the yahoo finance page.	10
US-11	A user shall be able to read a high level summary of how the algorithm works	P3	User Story	Complete	Page containing a summary of the ML algorithm is present	25
FE-1	The RoboStock web application shall be able to launch from any major desktop web browser, regardless of the underlying Operating System or hardware.	P0	Product Feature	Complete	The web application is able to be launched from multiple web browsers. Specifically, it should launch from these three popular web browsers: Google Chrome, Safari, and Firefox.	5
FE-2	The RoboStock application is accessible to the public, but a user must be logged in in order to access the ML Predictions page.	P1	Product Feature	Complete	The user can navigate through the website with the exception of the ML predictions page while not logged in.	20
FE-3	The RoboStock web application shall be able to display historical stock price data for any stock in the NYSE (New York Stock Exchange).	P0	Product Feature	Complete	Any of the NYSE symbols can be selected from a drop down menu and this will trigger a ML prediction.	50
FE-4	The RoboStock web application shall use some machine learning algorithm to predict the stock price based on past historical time series data.	P0	Product Feature	Complete	The application can predict stock prices using LSTM algorithm.	100
IC-1	The web application shall use the Model Template View design pattern. More specifically, it shall use the Django framework.	P0	Design and Implementation Constraints	Complete	The web application uses Django, and it's Model-Template-View design pattern as its framework	5
IC-2	The front-end technologies to be used shall be HTML, CSS, JavaScript, Bootstrap, (and JQuery if needed).	P0	Design and Implementation Constraints	Complete	These front-end technologies are used	5
IC-3	The back-end technologies and programming languages shall be SQLite, Python,	P0	Design and Implementation Constraints	Complete	These back-end technologies are used	5
UD-1	The web application shall be delivered with a user manual	P3	User Documentation	Complete	A user manual is provided	5
UD-2	The web application shall be delivered with an administrator manual	P3	User Documentation	Complete	An admin manual is provided	5
UI-1	The website is accessible to users while not logged in. Only the Machine Learning prediction page is disabled if the users is not logged in. (updated April 19, 2021)	P1	User Interface	Complete	The user can navigate through the website with the exception of the ML predictions page while not logged in.	5
UI-2	A navigation bar shall be displayed at the top of the page, or the left or right edge vertically. The navigation bar shall provide navigation to all the pages on the website	P0	User Interface	Complete	The navigation bar is consistent across all of the pages. The user is able to navigate through the pages using the navigation bar.	5
UI-3	An indicator on the upper right corner shall clearly display whether or not the user is logged in	P3	User Interface	Complete	The upper right corner says "login" if the user is not logged in yet, and it says "log off" if the user is logged in.	10
UI-4	Navigation to the 'Admin' page shall be accessible only to an administrator.	P0	User Interface	Complete	Only an admin, while logged in, can access the /admin page	5

Figure 4: Product Backlog

Figure 8 depicts the final Product Release Burndown Chart showing working days in the horizontal axis and effort as story points in the vertical axis. Given that the initial product backlog contained 555 points and the total project duration was 70 working days, the daily effort cadence of 8 story points per day was calculated. This eight daily story point cadence was used to calculate the forecasted progress line. The actual remaining line is a compilation of the iterative progress accomplished in each sprint.

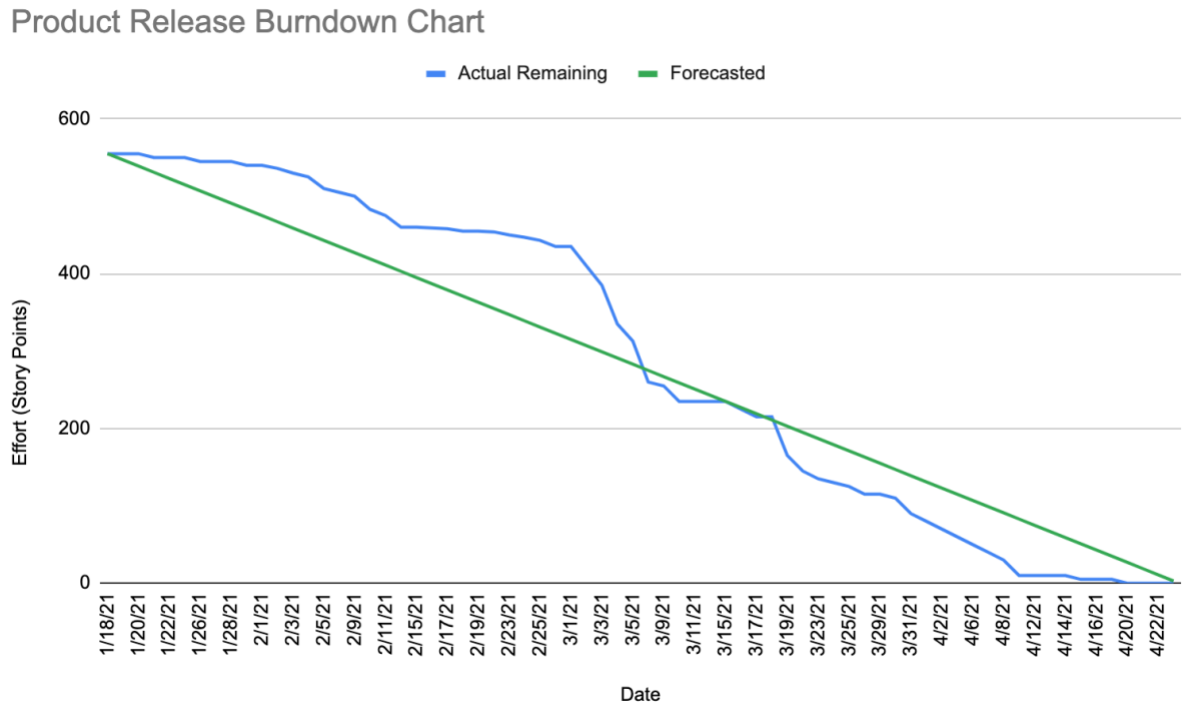

Figure 5: Product Release Burndown Chart

4.2.1 Sprint 1

In the first Sprint Iteration, the primary design and implementation constraints were taken out of the Product Backlog and into the Sprint Backlog. These IC (Implementation and Design Constraints) were considered to be pre-cursor before beginning work. No risks were identified at this stage. The Django project and application were initiated, and the general directory project structure was developed.

ID	USER STORY (Product Backlog Item or PBI)	Priority	Category	Status	Acceptance Criteria	Story Points
IC-1	The web application shall use the Model Template View design pattern. More specifically, it shall use the Django framework.	P0	Design and Implementation Constraints	Complete	The web application uses Django, and it's Model-Template-View design pattern as its framework	5
IC-2	The front-end technologies to be used shall be HTML, CSS, JavaScript, Bootstrap, (and JQuery if needed).	P0	Design and Implementation Constraints	Complete	These front-end technologies are used	5
IC-3	The back-end technologies and programming languages shall be SQLite, Python,	P0	Design and Implementation Constraints	Complete	These back-end technologies are used	5

Figure 6: Sprint Backlog - Sprint 1

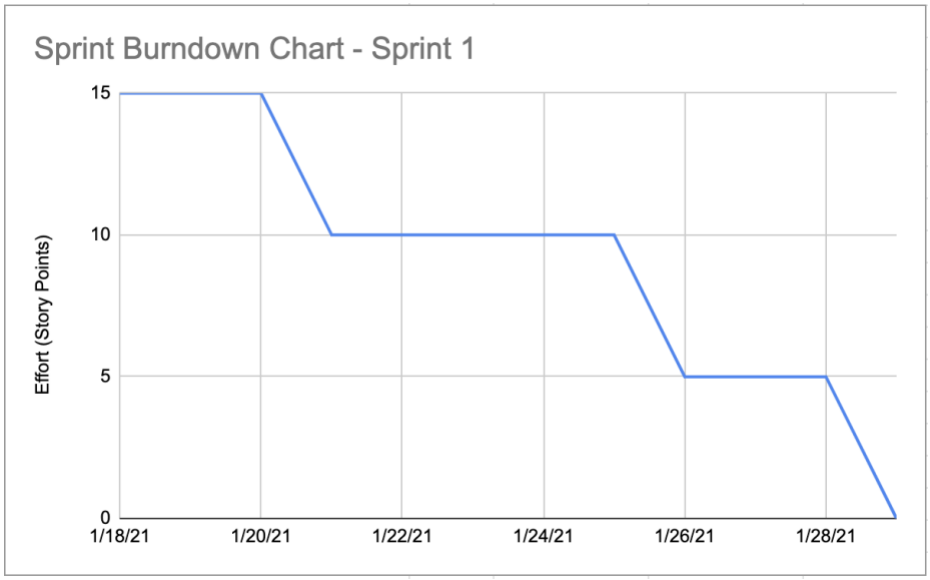

Figure 7: Sprint Burndown Chart - Sprint 1

4.2.2 Sprint 2

The focus of Sprint 2 was to begin work on the logging-in and logging off features. Registering a user was also part of the focus. The registration form was achieved through a Django form. The logic was added to the base.html file in order to disable the “ML Predictions” link on the navigation bar when the user is not logged in. No risks were identified during this Sprint.

ID	USER STORY (Product Backlog Item or PBI)	Priority	Category	Status	Acceptance Criteria	Story Points
US-1	A user shall be able to register to the website	P0	User Story	Complete	Successful user registration	25
FE-1	The RoboStock web application shall be able to launch from any major desktop web browser, regardless of the underlying Operating System or hardware.	P0	Product Feature	Complete	The web application is able to be launched from multiple web browsers. Specifically, it should launch from these three popular web browsers: Google Chrome, Safari, and Firefox.	5
FE-2	The RoboStock application is accessible to the public, but a user must be logged in in order to access the ML Predictions page.	P1	Product Feature	Complete	The user can navigate through the website with the exception of the ML predictions page while not logged in.	20
US-2	A user shall be able to login to the website	P0	User Story	Complete	Successful user Login	20
US-3	A user shall be able to log off the website	P0	User Story	Complete	Successful user logoff, user cannot access the ML Predictions page	10

Figure 8: Sprint Backlog - Sprint 2

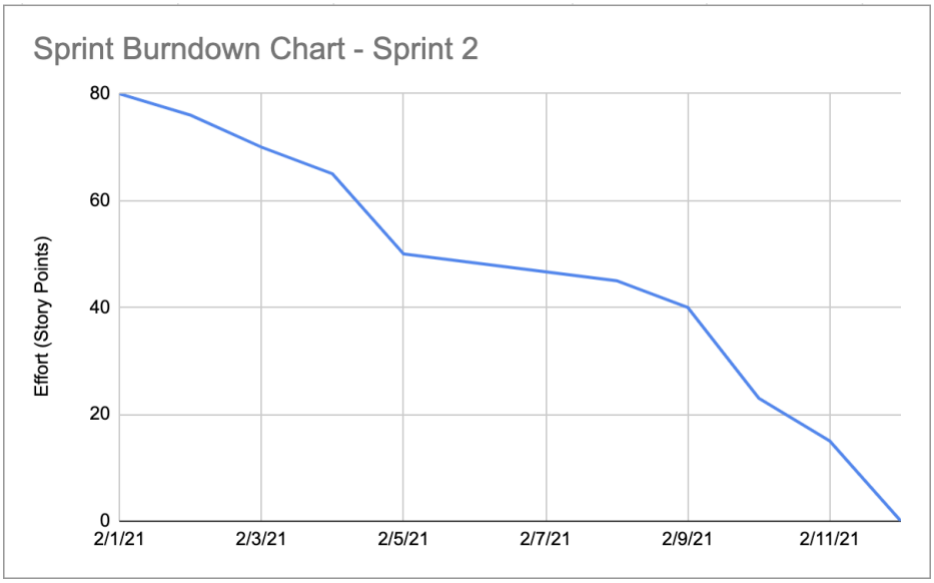

Figure 9: Sprint Burndown Chart - Sprint 2

4.2.3 Sprint 3

The focus of this sprint was User Interface items. All four user interface requirements were worked on, meaning that a great deal of work on the Templates (HTML and CSS files) was completed. The styling of the website uses Bootstrap 4.1.3 [16]. The Bootstrap classes such as navigation bar submit buttons and links were used.

ID	USER STORY (Product Backlog Item or PBI)	Priority	Category	Status	Acceptance Criteria	Story Points
UI-1	The website is accessible to users while not logged in. Only the Machine Learning prediction page is disabled if the users is not logged in. (updated April 19, 2021)	P1	User Interface	Complete	The user can navigate through the website with the exception of the ML predictions page while not logged in.	5
UI-2	A navigation bar shall be displayed at the top of the page, or the left or right edge vertically. The navigation bar shall provide navigation to all the pages on the website	P0	User Interface	Complete	The navigation bar is consistent across all of the pages. The user is able to navigate through the pages using the navigation bar.	5
UI-3	An indicator on the upper right corner shall clearly display whether or not the user is logged in	P3	User Interface	Complete	The upper right corner says "login" if the user is not logged in yet, and it says "log off" if the user is logged in.	10
UI-4	Navigation to the 'Admin' page shall be accessible only to an administrator.	P0	User Interface	Complete	Only an admin, while logged in, can access the /admin page	5

Figure 10: Sprint Backlog - Sprint 3

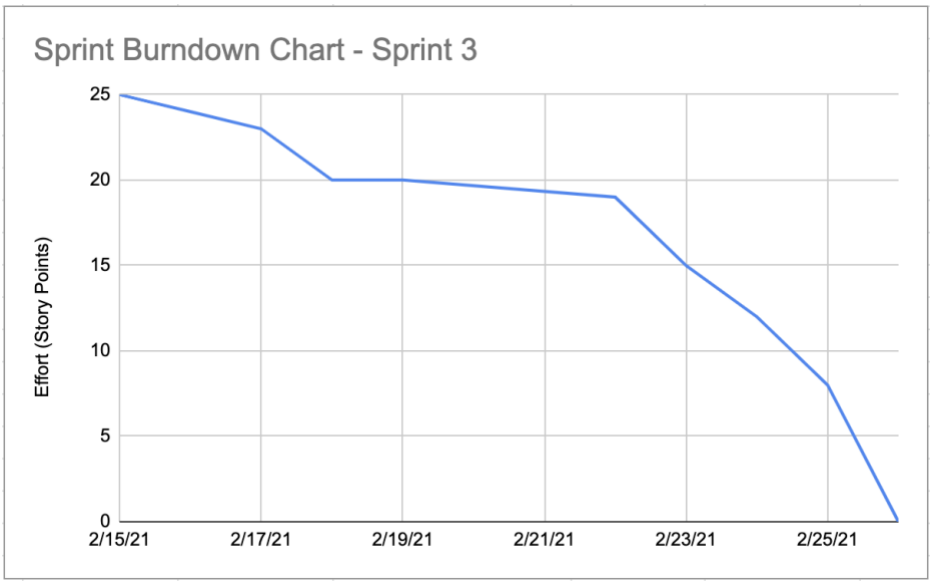

Figure 11: Sprint Burndown Chart - Sprint 3

4.2.4 Sprint 4

This sprint had some of the most significant User Stories and Feature Requirements for the entire project. The Long-Short Term Memory [3] was picked as the algorithm for making the predictions based on the historical time-series data. The python libraries used were sklearn [17] and Keras [15]. The total effort estimation for this sprint was 300 story points. 100 story points remained after the culmination of the sprint.

ID	USER STORY (Product Backlog Item or PBI)	Priority	Category	Status	Acceptance Criteria	Story Points
US-4	A user shall be able to see historical stock price data	P0	User Story	Complete	User selects a stock symbol and historical stock price data for the selected dates is displayed.	50
US-5	A user shall be able to see ML stock price prediction	P0	User Story	In Progress	User selects a stock symbol and prediction stock price data for the selected dates is displayed.	100
FE-3	The RoboStock web application shall be able to display historical stock price data for any stock in the NYSE (New York Stock Exchange).	P0	Product Feature	Complete	Any of the NYSE symbols can be selected from a drop down menu and this will trigger a ML prediction.	50
FE-4	The RoboStock web application shall use some machine learning algorithm to predict the stock price based on past historical time series data.	P0	Product Feature	In Progress	The application can predict stock prices using LSTM algorithm.	100

Figure 12: Sprint Backlog - Sprint 4

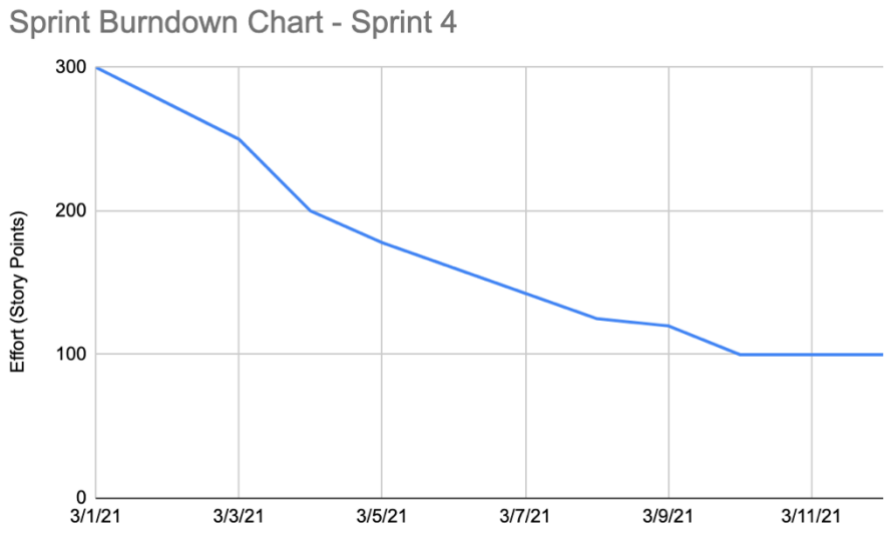

Figure 13: Sprint Burndown Chart - Sprint 4

4.2.5 Sprint 5

User Story US-5 and Feature Requirement FE-4 were carried over from Sprint 4 into Sprint 5. Miscellaneous user stories dealing with the administrator were completed. The Django framework offers a built-in administrator page and administrator functionality, so there was not much overhead when it came to effort (story points). A new user story was introduced, US-8 and the three major stock indexes were added to the website. Historical data showing these indexes, rendered in graphs using Google Charts [4], was used. Unfortunately, the User Story and Feature Requirement dealing with Machine Learning carried over and were not 100% completed in Sprint 5, making it the second sprint to carry over these two Product Backlog Items (PBI).

ID	USER STORY (Product Backlog Item or PBI)	Priority	Category	Status	Acceptance Criteria	Story Points
US-5	A user shall be able to see ML stock price prediction	P0	User Story	In Progress	User selects a stock symbol and prediction stock price data for the selected dates is displayed.	50
FE-4	The RoboStock web application shall use some machine learning algorithm to predict the stock price based on past historical time series data.	P0	Product Feature	In Progress	The application can predict stock prices using LSTM algorithm.	50
US-6	An administrator shall be able to administer the website	P0	User Story	Complete	The built-in Django admin page is accessible to administrators only.	10
US-7	An administrator shall be able to modify database through the admin page	P0	User Story	Complete	An administrator can add or remove users from the Users model through the admin page.	10
US-8	A user shall be able to see major stock indexes recent history	P1	User Story	Complete	Historical trends for top stock indexes is displayed	50
US-9	A user shall be able to perform a Google search from the website	P1	User Story	Complete	A user can type something on a search bar and trigger a Google search.	20
US-10	A user shall be able to access Yahoo Finance through the website	P1	User Story	Complete	A user can click on a Yahoo Finance link and navigate to the yahoo finance page.	10

Figure 14: Sprint Backlog - Sprint 5

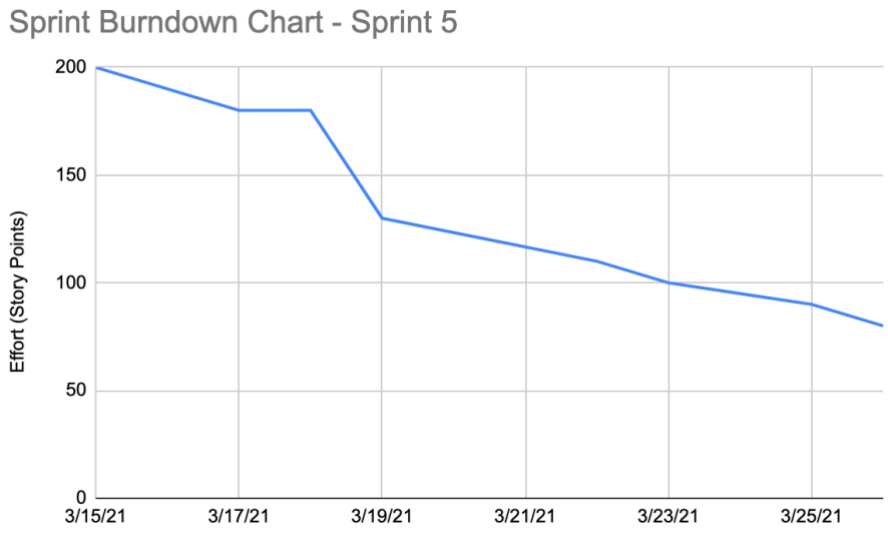

Figure 15: Sprint Burndown Chart - Sprint 5

4.2.6 Sprint 6

Sprint 6 focused on closing some of the more elaborate User Stories carrying over from previous sprints. Sprint 6 was successful at closing US-5, US-11, and FE-4.

ID	USER STORY (Product Backlog Item or PBI)	Priority	Category	Status	Acceptance Criteria	Story Points
US-5	A user shall be able to see ML stock price prediction	P0	User Story	Complete	User selects a stock symbol and prediction stock price data for the selected dates is displayed.	40
FE-4	The RoboStock web application shall use some machine learning algorithm to predict the stock price based on past historical time series data.	P0	Product Feature	Complete	The application can predict stock prices using LSTM algorithm.	40
US-11	A user shall be able to read a high level summary of how the algorithm works	P3	User Story	Complete	Page containing a summary of the ML algorithm is present	25

Figure 16: Sprint Backlog - Sprint 6

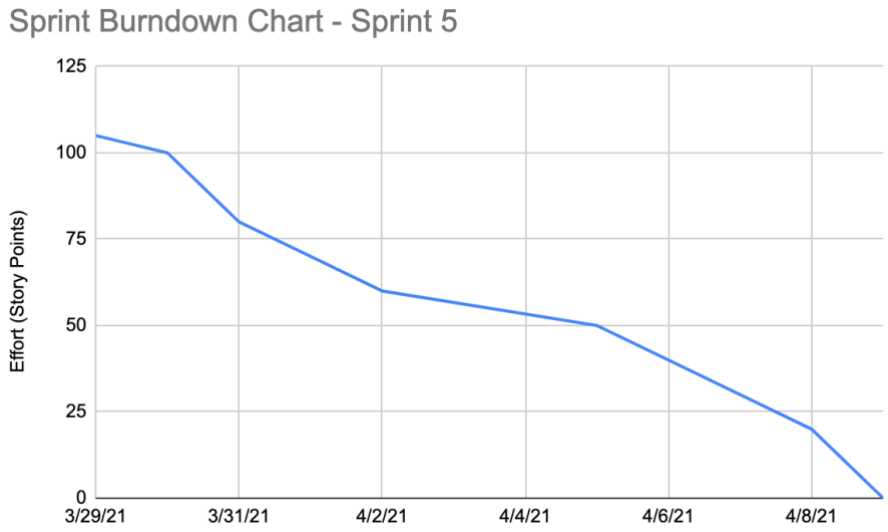

Figure 17: Sprint Burndown Chart - Sprint 6

4.2.7 Sprint 7

Sprint 7's focus was on the user documentation requirements (UD-1 and UD-2). The low Story Points in this sprint allowed for this report to be finalized.

ID	USER STORY (Product Backlog Item or PBI)	Priority	Category	Status	Acceptance Criteria	Story Points
UD-1	The web application shall be delivered with a user manual	P3	User Documentation	Complete	A user manual is provided	5
UD-2	The web application shall be delivered with an administrator manual	P3	User Documentation	Complete	An admin manual is provided	5

Figure 18: Sprint Backlog - Sprint 7

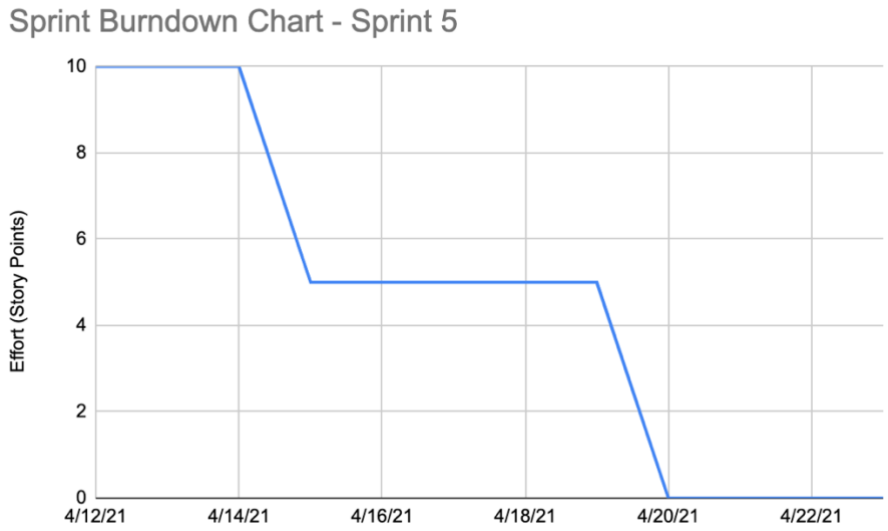

Figure 19: Sprint Burndown Chart - Sprint 7

5 Testing

The testing of the RoboStock application project entails two main parts. The first part, outlined in Section 5.1, outlines a Test Plan. The second part, outlined in Section 5.2 has a summary of the test results.

5.1 Test Plan

This Test Plan was formulated according to Test Plan presented by Burnstein on Chapter 7 of her textbook Practical Software Testing [4], and the Test Plan example presented on Appendix II of the same text. The test plan described by Burnstein is derived from the standard IEEE Std 829-1983 [11].

5.1.1 Test Plan Identifier

CSUF-RoboStock-2021-1

5.1.2 Introduction

- **Overall description of the project:** This project's final deliverable is the RoboStock web application. The RoboStock project leverages the Scrum software development process defined in Section 4.1.
- **Overall description of the software system being developed:** As described in this report, RoboStock is a web application built upon the Django Framework that intends to give users an easy-to-use solution for making daily trading decisions.
- **Overall description of the software items and features to be tested:** The software item to be tested is the RoboStock web application. The features to be tested are the features derived from the Use Cases and User Stories defined in the aforementioned sections.

5.1.3 Items to be tested

The RoboStock web application is the only item to be tested. The testing scopes starts and ends within the RoboStock web application, which is considered to be the system under test.

5.1.4 Features to be tested

The key features to be tested are outlined in Section 2.1.2. In addition to these features, a test for each User Story shall be performed in order to validate that the RoboStock web application performs according to the expectations of the user. The User Story requirements is the ultimate set of features to be tested.

5.1.5 Approach

One of the key obstacles to overcome in web application testing is the coupling of components and the need for the different MTV components to work appropriately in order to satisfy the requirements. For this reason, the Model, Template, and View shall work appropriately and together in order for the website to work appropriately. Unit Testing will be omitted since testing individual components can be cumbersome and will not add much value. Instead, Integration testing will be performed using Django's built-in testing suite. Django's built-in testing suite uses unittest [18] as it's backbone testing tool for performing some basic tests.

5.1.6 Pass/Fail Criteria

The pass and fail criteria are set forth initially by the Use Cases outlined in Section 2.3, and secondly by the acceptance criteria designated to each one of the PBI in the Product Backlog (see Section 4.2).

5.1.7 Suspension and resumption criteria

The first suspension and resumption criteria entail the end and start of working days respectively. Testing should be suspended at the end of a working day and carried on at the beginning of the next working day.

Whenever major issues are encountered during testing, testing should be suspended and the bug, error, failure, or fault, should be addressed in order to continue with testing. Examples of major issues are:

- The Django development server does not initiate or crashes.
- The Django server returns an HTTP status code 500 or 404
- The Django server does not return the expected Template (HTML file) or does not return the expected output.

The resumption criteria involve the bug, error, failure, or fault to be addressed prior to testing continuation.

5.1.8 Test deliverables

The testing artifacts shall be delivered:

- An Integration Testing Summary
- A System Testing Summary
- An Acceptance Testing Summary

5.1.9 Testing Tasks

The testing tasks shall be performed throughout every iteration. Testing shall be treated as an activity, and not a phase. Therefore, testing should be carried out on a continuous basis. The following three high level testing tasks shall be performed:

1. **Integration Testing:** perform integration testing using the Django's built in testing suite in order to verify that the correct templates are returned by the views when the URL dispatcher triggers a URL path. Integration Testing shall be conducted using a white box approach, since the name of the templates (HTML files) and URL extensions must be known.
2. **System Testing**: System testing shall verify that the high-level nonfunctional requirements are met. The quality attributes outlined in Section 2.5 are to be used as a basis for devising the acceptance criteria for the nonfunctional requirements tests.
3. **Acceptance Testing:** The User Stories shall be used in order to devise the acceptance criteria for final acceptance testing.

5.1.10 Test environment

The test environment to be used should be exactly the same as the development environment, `MyDjangoEnv`. It is recommended to create a conda virtual environment, install Django and all relevant Python libraries, in order to conduct the test.

5.1.11 Responsibilities

The Scrum development team is responsible for conducting all testing as an activity. The individuals and roles outlined in Section 5.1.16 are responsible for approving this Test Plan.

5.1.12 Staffing and training needs

This test plan does not outline explicit training needs. The Scrum team shall perform all testing. Ideally, testing is performed by the developers for the code they have developed. No additional staffing needs beyond the scrum team are necessary.

5.1.13 Scheduling

Testing shall follow the two-week Scrum sprint cadence. Testing shall be carried out on a daily basis whenever a new feature or new User Story has been developed.

5.1.14 Risks and contingencies

Testing Risk ID	Risk Description	Occurrence Probability	Prioritization	Contingency
TR-1	Testing consumes more time than expected and takes away resources and time from development.	High	P0	Begin development early. Allocate daily time for testing. Leverage the built-in Django testing suite for automated testing.
TR-2	Low test coverage resulting from poor test case design.	Medium	P3	Design test cases early in the project in order to allow time for thorough testing.
TR-3	Errors and bugs escape to the customer due to code changes after features have been tested.	Medium	P2	Conduct regression testing whenever a major feature is introduced.

Table 20: Testing risks and contingencies

5.1.15 Testing costs

The testing costs are based on the total number of man hours required for testing during this project. Allocating one half (1/2) man hour per day and given that there will be roughly 70 working days in this project, the total man hours estimated for testing is 35 hours. Assuming that the cost is $100/hour, the total cost for testing results in $3,500. This figure is an approximation only and actual final cost may differ based on testing time differences, etc.

5.1.16 Approvals

The following is a list of required and optional approvers:

- Scrum Master (Optional)
- Tester(s) (Required)
- Developer(s) (Required)
- Product Owner (Required)

- Costumer (Required)
- Middle and Upper Management (Optional)

5.2 Test Results

This section contains the Integration, System, and Acceptance Testing Results.

5.2.1 Integration Testing Results

Django Web applications make use of the Model-Template-View design patter. RoboStock being a Django application uses these components as part of its framework architecture. The built-in test suite was leveraged for conducting integration testing that would verify the proper templates were rendered by the URL dispatcher through the views. Test cases for each one of the templates were formulated and ran. The integration testing was conducted using a white box approach. Understanding of how the URL dispatcher, View, and Templates interact with each other was necessary in order to develop the tests.

Figure 20: Django Testing – Templates

Figure 24 shows the Terminal output after running the `$ ./manage.py test` command.

```
(MyDjangoEnv) carloss-imac:RoboStockProject carlosgibson$ ./manage.py test
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
..-256.33 -0.75
....
----------------------------------------------------------------------
Ran 6 tests in 3.067s

OK
Destroying test database for alias 'default'...
(MyDjangoEnv) carloss-imac:RoboStockProject carlosgibson$
```

Figure 21: Django Testing Terminal Output

This testing confirmed that the URL extension was handled by the URL dispatcher, passed to the appropriate view, and rendered the desired template. Table 21 provides a test summary with results for the tests.py code script and Terminal output above (Figures 23 and 24 respectively).

Integration Test	Acceptance Criteria	Test Result
test_main_page	• Return response status code 200 “OK” • Verify that the template used is ***RoboStockApp/index.html*** • Verify that the html template contains the text ***Welcome to RoboStock!***	PASS
test_home_page	• Return response status code 200 “OK” • Verify that the template used is ***RoboStockApp/home.html*** • Verify that the html template contains the text ***Web application's Home Page and index***	PASS
test_marketindexes_page	• Return response status code 200 “OK” • Verify that the template used is *RoboStockApp/marketindexes.html* • Verify that the html template contains the text ***Major Stock Exchange Indexes***	PASS
test_mlpredictions_page	• Return response status code 200 “OK” • Verify that the template used is *RoboStockApp/mlpredictions.html* • Verify that the html template contains the text ***Machine Learning Stock Price Predictions***	PASS
test_userlogin_page	• Return response status code 200 “OK” • Verify that the template used is *RoboStockApp/login.html* • Verify that the html template contains the text ***Please Login***	PASS
test_register_page	• Return response status code 200 “OK”	PASS

	• Verify that the template used is *RoboStockApp/registration.html*	
Run the Django server and ensure no errors are present	• Run the Terminal command `$ Python manage.py runserver` and ensure that no glaring errors appear on the web browser • (Django performs some basic checks when the command `$ python manage.py runserver` is ran. This ensure that no syntax errors are present in the python components. Specifically, it compiles the code on important modules such as the views.py file and the models.py file. If there are errors, they will be flagged on the web browser.)	PASS

Table 21: Integration Test Cases, Acceptance Criteria, and Testing Results

5.2.2 System Testing Results

System testing was performed using a black box approach. Given that Acceptance Testing would cover all the features derived from the user stories, the focus for system testing was placed on the nonfunctional requirements outlined in Section 2.5.

ID	Nonfunctional Requirement	Acceptance Criteria	Test Results
NFR-1	The website shall be available 99.9996% of the time	Over the course of a year, the website shall be "up and running" or available for 99.9996% of the time. Note that this availability quality attribute is to be tested once the website has been fully deployed on a hosting web service. The website's availability is dependent on the hosting service availability. This metric cannot be tested at the time this report was finalized.	Not Testable – Pending website launch
NFR-2	The website shall use password encryption for security	Ensure that the passwords are stored using the PBKDF2 with a SHA256 hash (this is the default password encryption algorithm and hash offered by Django). Access any user through the Django 'admin' page and ensure that on the user's password says "**algorithm**: pbkdf2_sha256."	PASS
NFR-3	The website shall be easy to maintain	The website's 'admin' page is accessible with a set of superuser credentials. The files are modifiable, and the MTV components are separate from each other.	PASS

NFR-4	The website shall have a latency of less than 30 seconds for ML stock price prediction.	The visualization of the ML Predictions chart shall be returned in less than 30 seconds from the time the query is triggered.	PASS

Table 22: Nonfunctional Requirements Testing

5.2.3 Acceptance Testing Results

Acceptance testing Test Cases were derived from the Software Requirements Specification (see Section 2). These test cases were used to validate that the RoboStock web application met the design intent. Table 23 contains the user story IDs, the User Story description, the Acceptance Criteria for the test, and the Acceptance Test Results.

ID	User Story Description	Acceptance Criteria	Test Results
US-1	As a user, I want to be able to register to the website, in order to be able to log in, and access the Machine Learning page.	• The user is able to navigate to the login page, where they have the option to login or register. • The user is able to navigate to the registration form page from the login page. • The user is able to register by using a valid email format, a username, and password. • The user can successfully login with the newly created credentials	PASS
US-2	As a user, I want to be able to login to the website in order to access the Machine Learning Page	• The user cannot click on the “ML Predictions” link, which navigates to the ML Predictions page, unless the user is logged in. • The user is able to login using a valid set of credentials • The user is NOT able to login using an invalid set of credentials • Once logged in, the user is able to navigate to the ML Predictions page and is able to interact with the page.	PASS
US-3	As a user, I want to be able to logoff from the website in order to prevent someone else who isn’t me to interact with the ML Predictions tab.	• Clicking the logoff icon logs the user off the application. • The “ML Predictions” on the navigation bar is disabled. The user can no longer click	PASS

		on it to access the ML Predictions page.	
US-4	As a user, I want to be able to display stock price data, and not just the stock price predictions based on the Machine Learning algorithm. I also want to be able to see the major U.S. Stock Exchange Indexes, their current values, their day change, and their past trend for the last 12 months.	• The user is able to see the three major U.S. Stock Exchange Indexes (S&P 500, NASDAQ, and the Dow Jones Industrial Average). • The user is able to see the historical trend of a selected stock symbol from the N.Y.S.E.	PASS
US-5	As a user, I want to be able to see the Machine Learning price prediction (which is based on the time series data of the stock price), overlayed on top of the actual closing price line graph.	• The user is able to see the stock price predictions once they have selected the N.Y.S.E. symbol, a start date, and an end date. • The user is able to hover over the prediction trace and see the values for both the currently highlighted date and the prediction price.	PASS
US-6	As an administrator, I want to be able to access the built in Django Admin page in order to preform standard administrative actions.	• The administrator is able to access the Django Admin page with a superuser set of credentials. • The administrator is able to perform typical administrative actions on the Django Admin page.	PASS
US-7	As an administrator, I want to be able to access the Users and delete or add users through the Admin page.	• The administrator is able to add or delete users through the Admin page.	PASS

Table 23: User Stories, Acceptance Criteria, and Testing Results

6 Installation Instructions

The installation instructions described in this section pertain to the installation of the Django project, on a host machine (server). In order to run the Django server on a host machine, the environment must be set with things such as Python installations, etc.

6.1 Create A Virtual Environment

It is quite useful to run the Django commands on the Terminal (Mac OS), or Command Prompt (Windows) within a virtual environment. The following instructions show some basic virtual environment commands for working with virtual environments.

- Create a virtual environment using Conda as follows:

```
$ conda create --name MyDjangoEnv Django
```

- Activate the virtual environment with the following command:

```
$ conda activate MyDjangoEnv
```

- To deactivate the virtual environment, run the following command:

```
$ conda deactivate
```

- To list the currently available virtual environments, run the following command:

```
$ conda info --envs
```

- To install the latest version of Django within the virtual environment, run the following command:

```
$ conda install django
```

6.2 Obtain Source Code Files

The source code files are located in GitHub. Obtain these source code files and mount them on the host machine intended to host the Django project.

GitHub Repository Name	GitHub Repository Link
closgibs/RoboStock	https://github.com/closgibs/RoboStock

Table 24: Source Code Repository Information

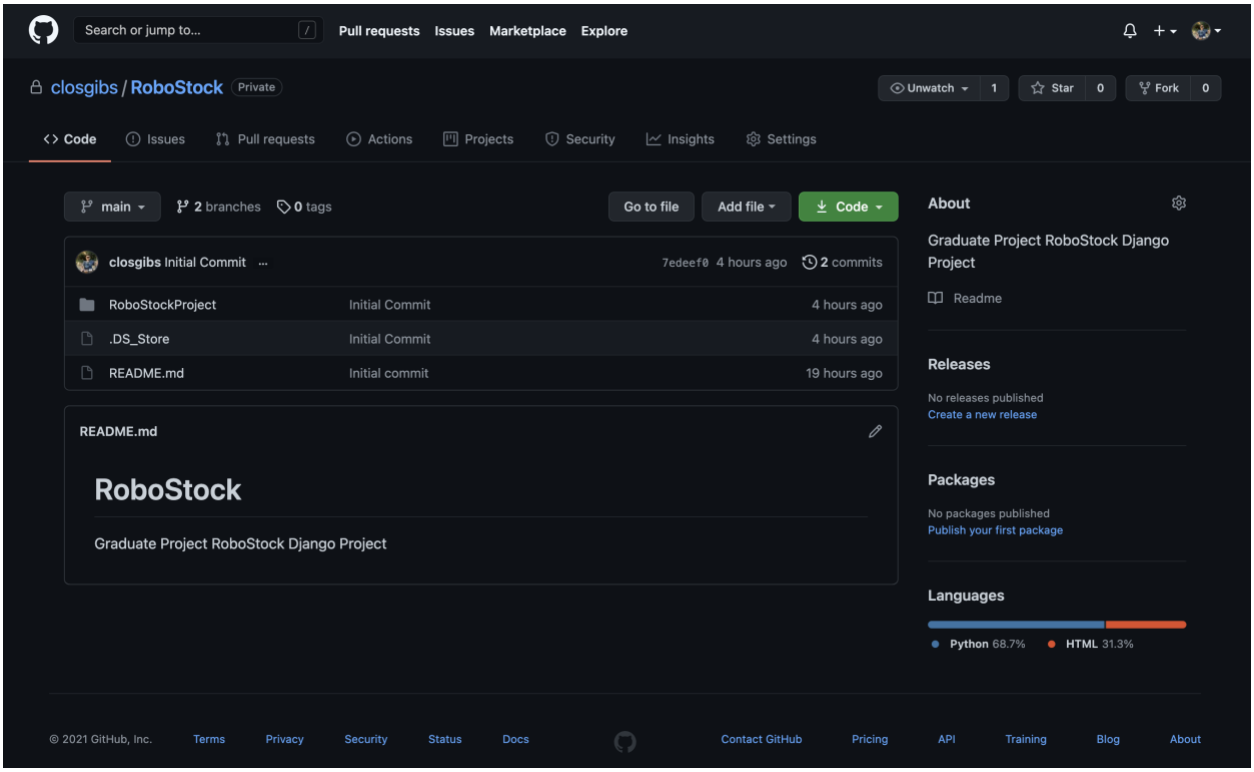

Figure 22: Source Code GitHub Repository

6.3 Run the Django server

To run the Django server, simply navigate to the RoboStockProject folder and run the following command within an activated virtual environment.

```
$ Python manage.py runserver
```

This will launch the Django server locally. Navigate to the URL from the Terminal or Command Line output using a web browser.

7 Operating Instructions

The operating instructions are provided through the following User and Administrator Manual. The User portion of the manual includes basic instructions on how to interact with the RoboStock web application. The Administrator portion of the manual has additional information on how to create a superuser, and how to access the Django “Admin” page using the set of credentials for said superuser.

7.1 User Manual

Step 1: In order to access the RoboStock web application, copy and paste this link in your browser’s URL field: http://127.0.0.1:8000/. The application should launch, and you should see the website’s landing page. (NOTE: The server link will change based on the deployment site, etc. The link shown above is for a locally running server).

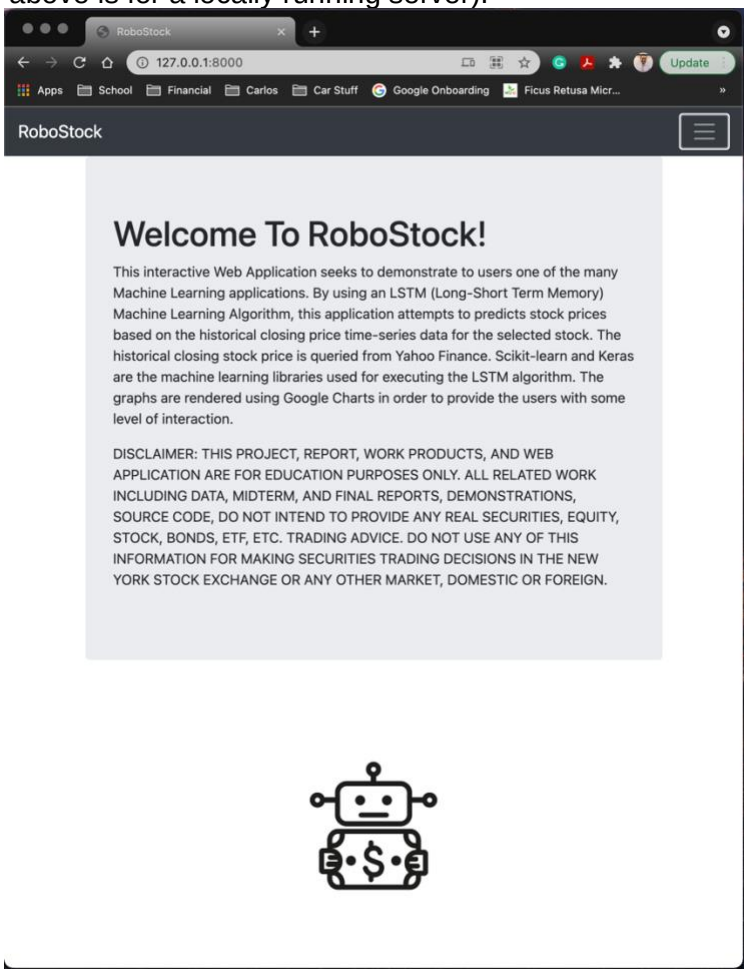

Figure 23: User Manual - Landing Page

Step 2: Next, access the navigation bar by expanding the menu (1) and click on the ‘login’ link which will take you to the login in page.

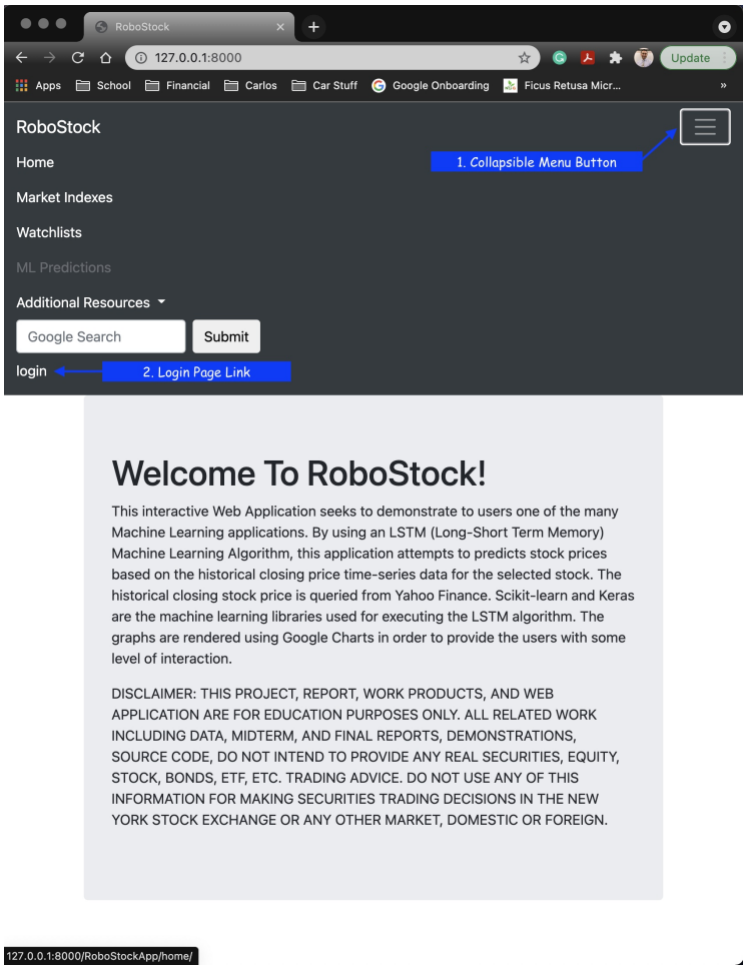

Figure 24: User Manual – Navigation Bar

Step 3: In the login page, enter a valid set of credentials or click on the 'Register' button in order to register to the site. For registering instructions, please see Step 4. Skip Step 4 if there is no need to register.

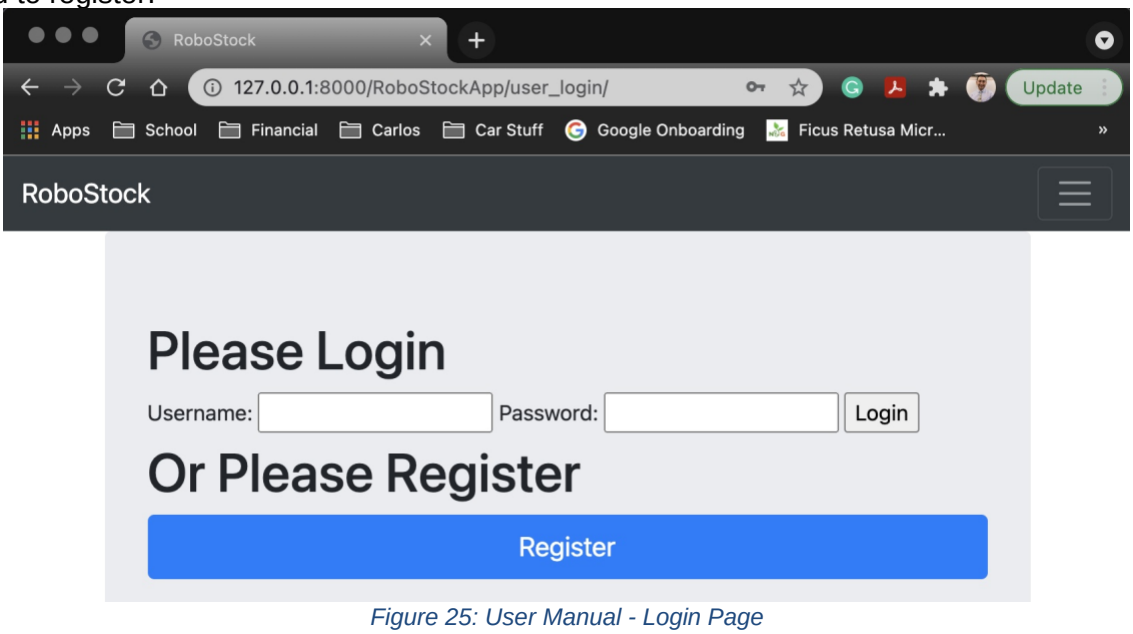

Figure 25: User Manual - Login Page

Step 4: After clicking on the ‘Register’ button, the website will navigate to the registration form page. Enter a username, password, and an email address with a valid email address format. Then click register. After successfully registering, you will see the message “Thank you for registering!” Go back to step 3 and use the newly created credentials to login to the website.

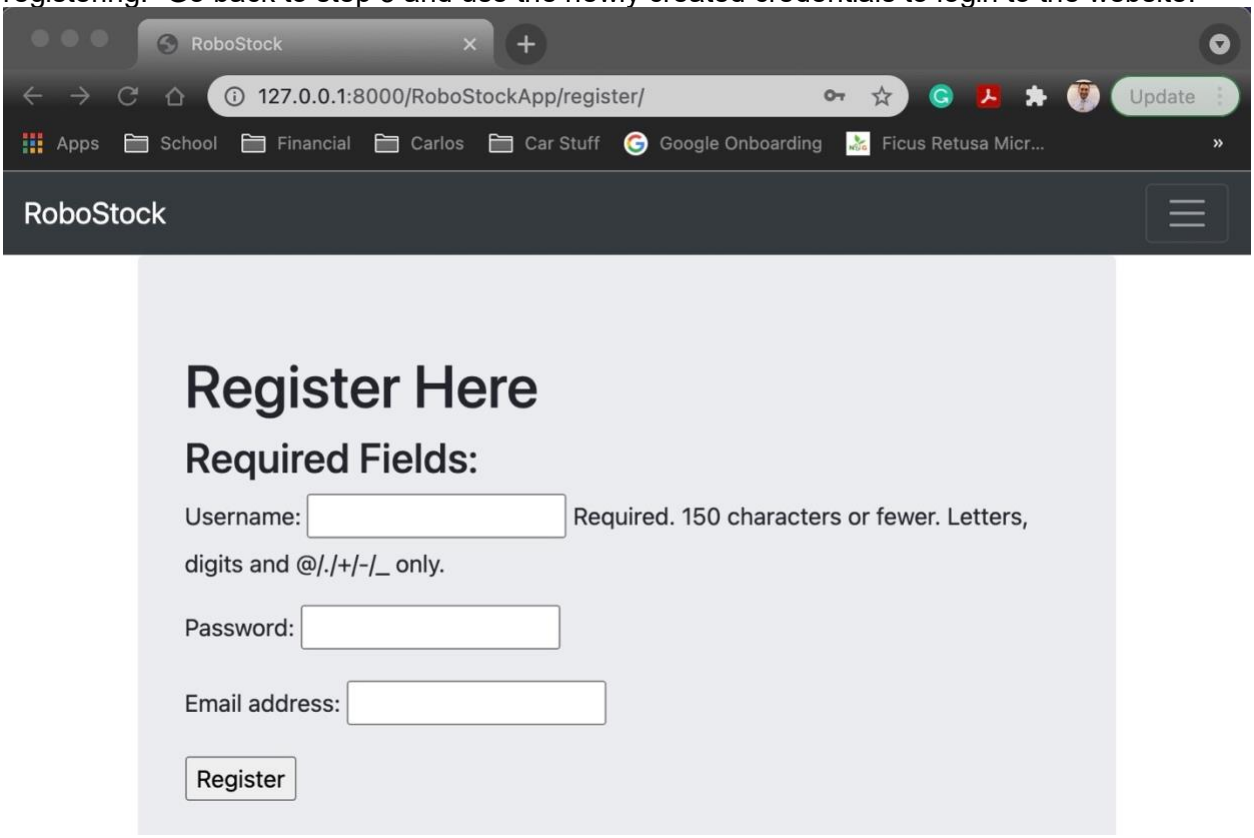

Figure 26: User Manual - Registration Page

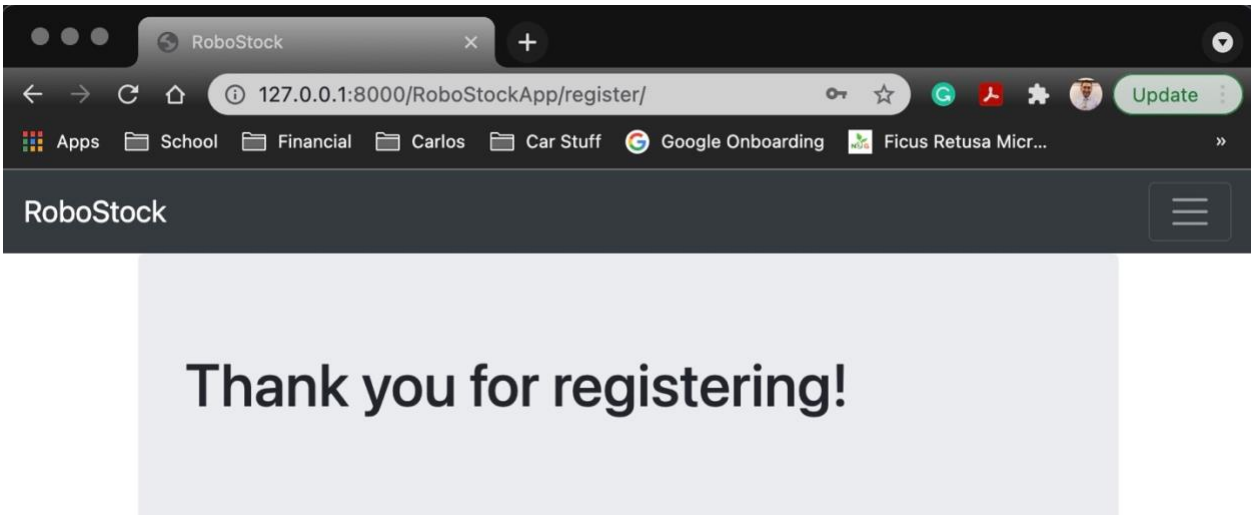

Figure 27: User Manual - Registration Message

Step 5: Navigate to the ‘Market Indexes’ page from the navigation bar. This will take you to a page summarizing the three major stock market indexes. The current point value, day change, and percentage day change are displayed, along with the historical trend since the beginning of the year 2021.

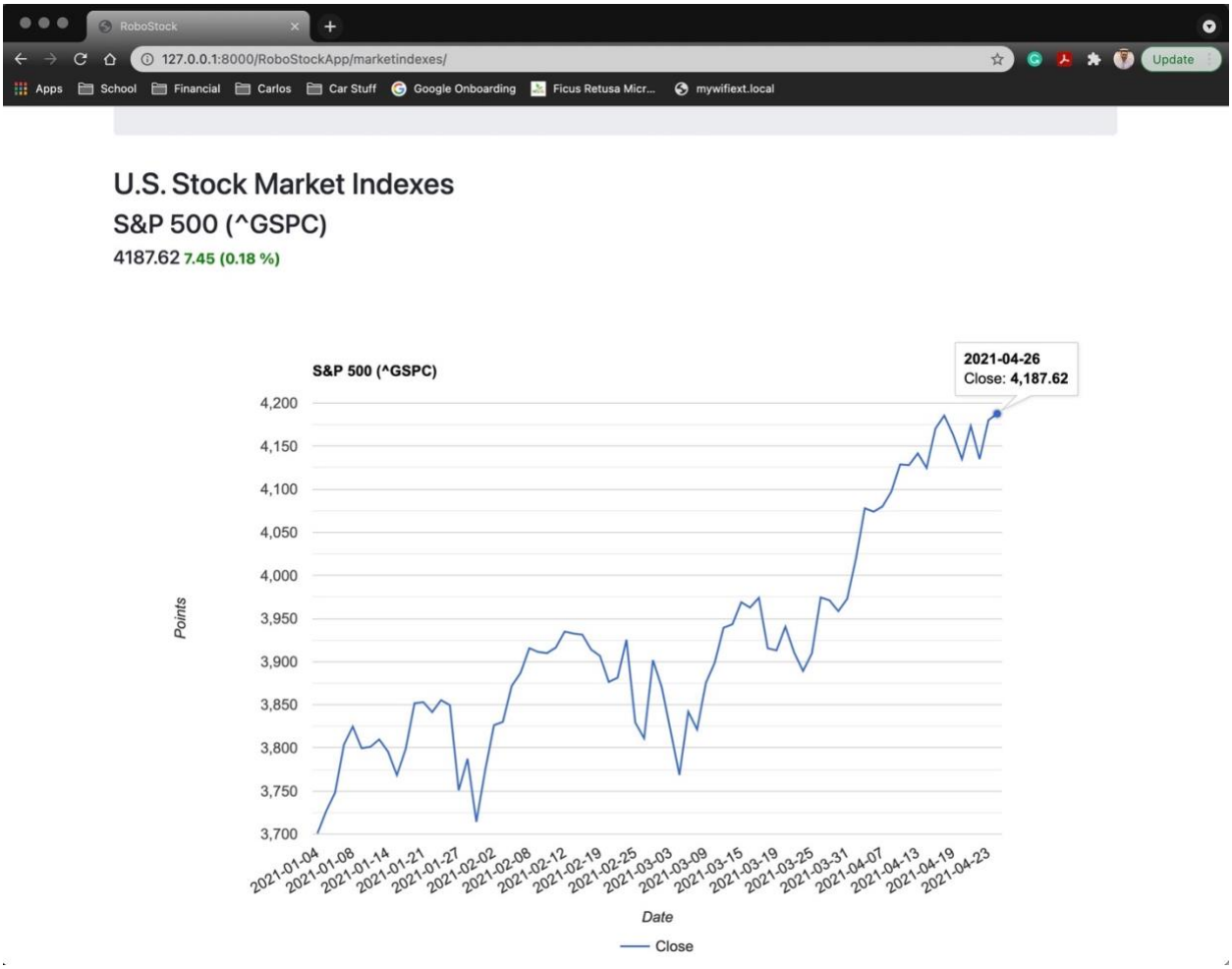

Figure 28: User Manual - Stock Indexes Page

Step 6: Click on the 'Watchlists' link through the navigation bar or through the website's Home page index. Select any stock symbol from the dropdown menu. Click the 'Query Date' button. The stock price data will be displayed, along with the current price value, the day change, and the day change as a percentage. Hover over the line graph to display the date and closing price value through the tooltip.

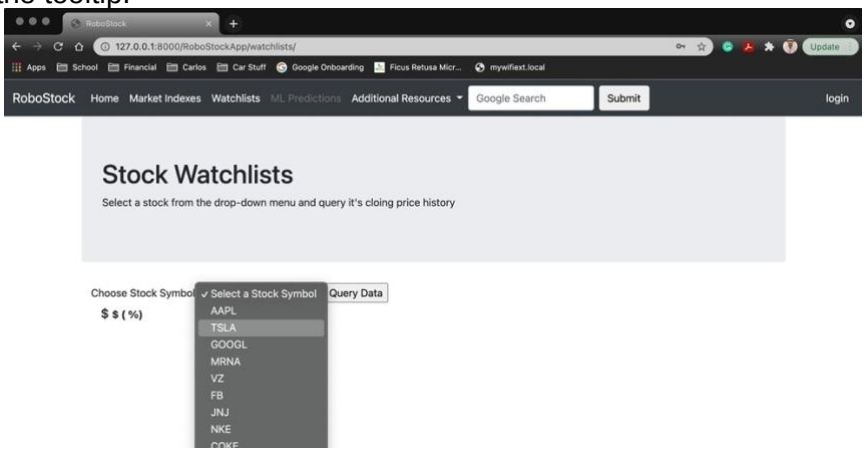

Figure 29: User Manual - Stock Symbol Dropdown Menu

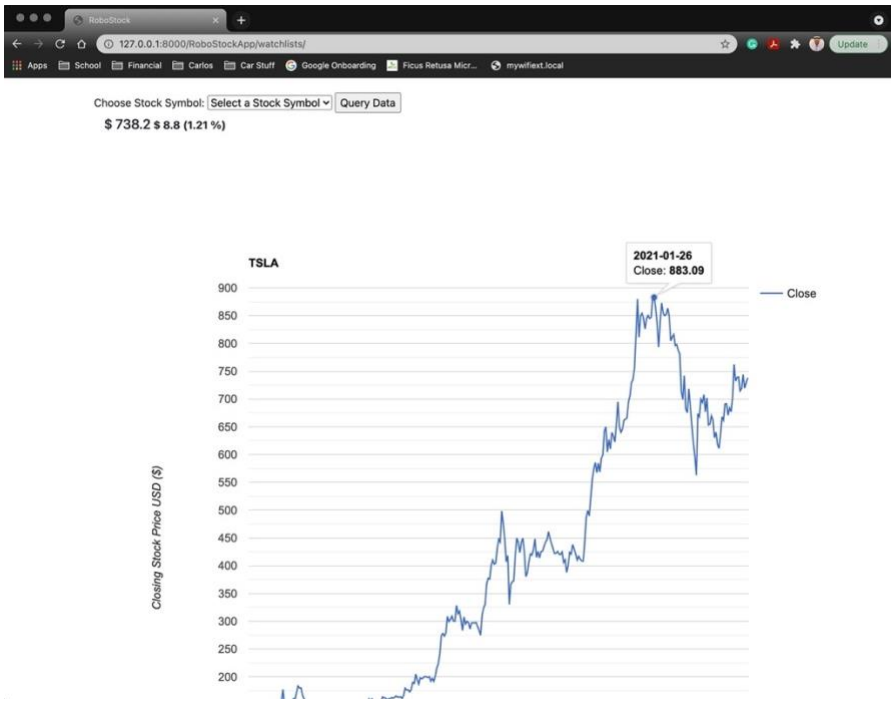

Figure 30: User Manual - Watchlists Page

Step 7: While logged into the website, click on the 'ML Predictions' link through the navigation bar or the home page website index. If the 'ML Predictions' link is disabled, you are not logged in. Select a stock quote to run predictions for, a start date for initiating the training algorithm, and an end date up to the current date (today's date). The algorithm will display the actual closing prices for the selected stock in blue, and the predictions in red.

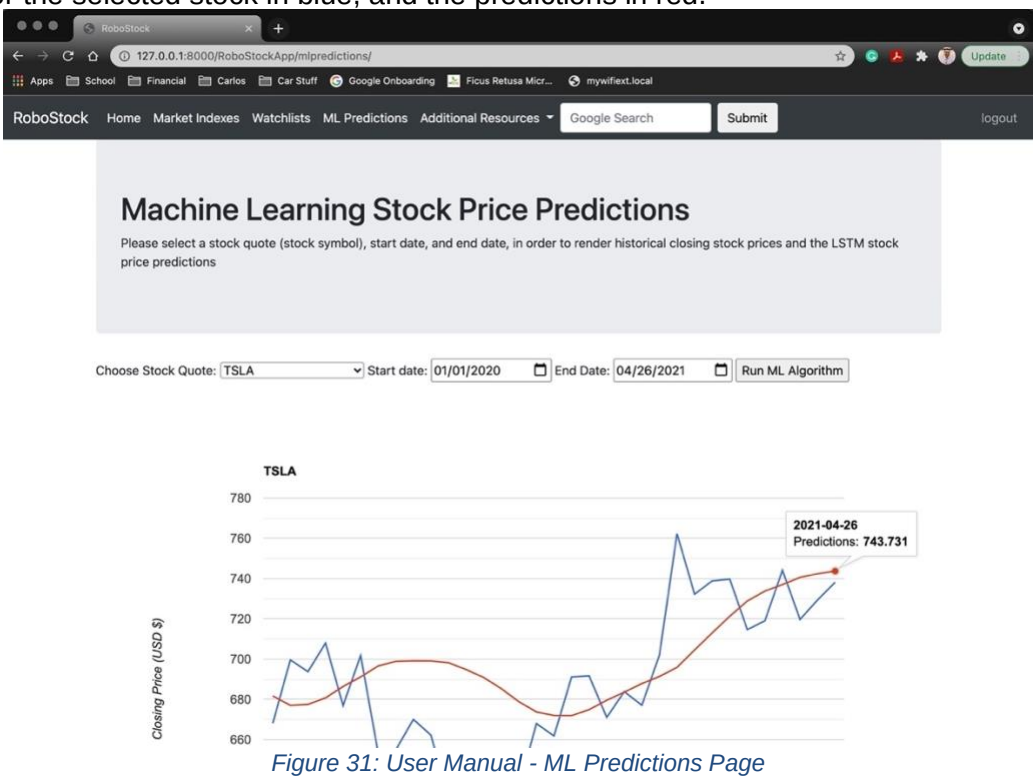

Figure 31: User Manual - ML Predictions Page

Step 8: The 'Additional Resources' dropdown menu contains (1) a link to the details on how the LSTM recurrent artificial neural network architecture works. It has a code walk-through that

details the logic behind the algorithm. (2) a link to Yahoo Finance. And (3) a link to the graduate student's LinkedIn profile. In addition to these resources, the website has a Google Search bar for performing internet searches outside the website (4).

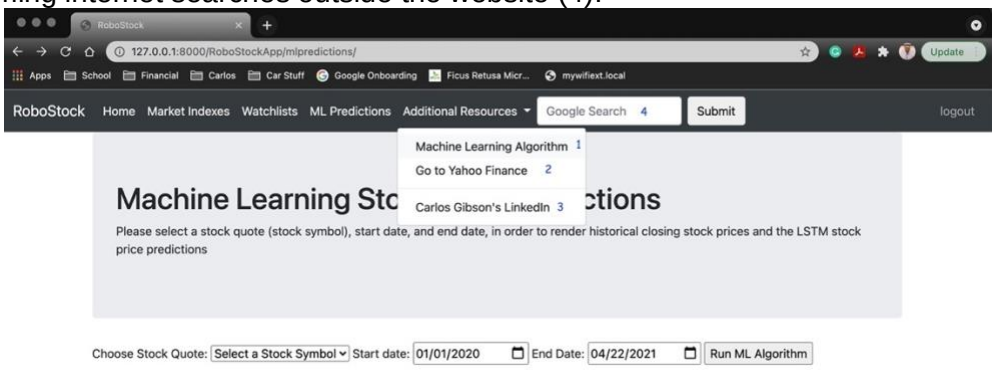

Figure 32: User Manual - Additional Features

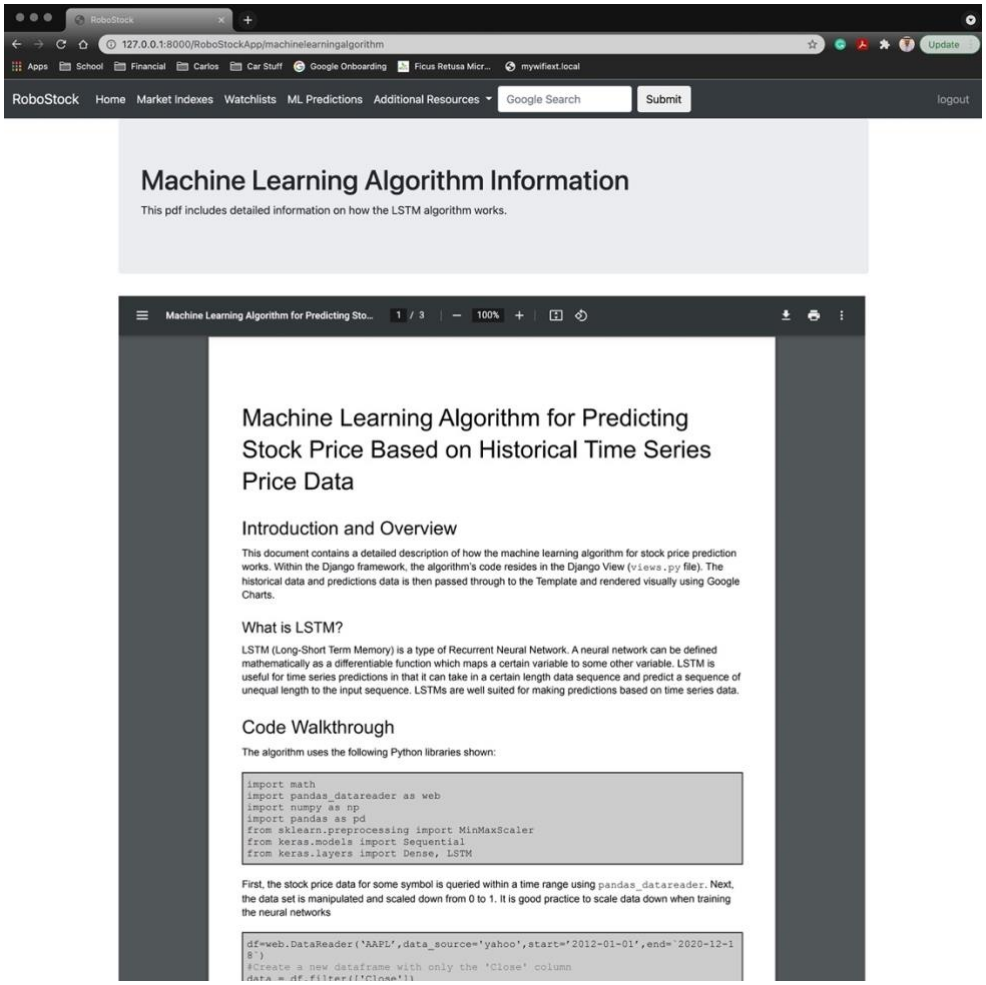

Figure 33: User Manual - Machine Learning Algorithm Information Page

7.2 Administrator Manual

The administrator features are inherited from the Django framework. In order to access the admin page, simply add the extension 'admin/' to the application's URL. For example, use the URL http://127.0.0.1:8000/admin/ for accessing the admin page. In order the access the

administrator page, one must have ‘Super User’ credentials, or administrator credentials. To create a super user, navigate to the RoboStockProject working directory where the manage.py file resides. Execute the following command:

```
$ Python manage.py createsuperuser
```

The Terminal (Mac) or Command Prompt Window (Windows) will guide you through the step-by-step process of creating a ‘Super User.’

For more information on how to administer the website, please visit the Django admin site documentation at https://docs.djangoproject.com/en/3.2/ref/contrib/admin/.

8 Recommendations for Enhancement

There are several recommendations for enhancement for improving the RoboStock web application. Ideally, this web application would be developed by a team of full-time developers with a wide range of skills (e.g., Machine Learning, front-end web development, back-end web development, testing, Python programming, UI and UX subject-matter-experts, product owners who are subject-matter-experts in finance and securities/stock investment, etc.). In addition to additional human resources, additional time would be ideal. Here are three primary recommendations for enhancement.

8.1 Investigate Other Machine Learning Algorithms

The LSTM (Long-Short Term Memory) Recurrent Neural Network architecture was chosen for its reputation with predicting natural phenomena using time series data. Investigating other more suitable architectures is at the top of the priority list when it comes to recommendations for enhancement. Another recommendation is to investigate how to take into account other relevant data into the algorithm that can affect stock prices. Although the algorithm proved to be somewhat accurate by using historical time-series data alone, it had difficulties adjusting to abrupt and sudden price changes. For example, the Google Sock price saw an abrupt price increase around March 31st. The predictions took a while to be able to adjust and account for this prediction change and undershot the forecast for a couple of weeks after.

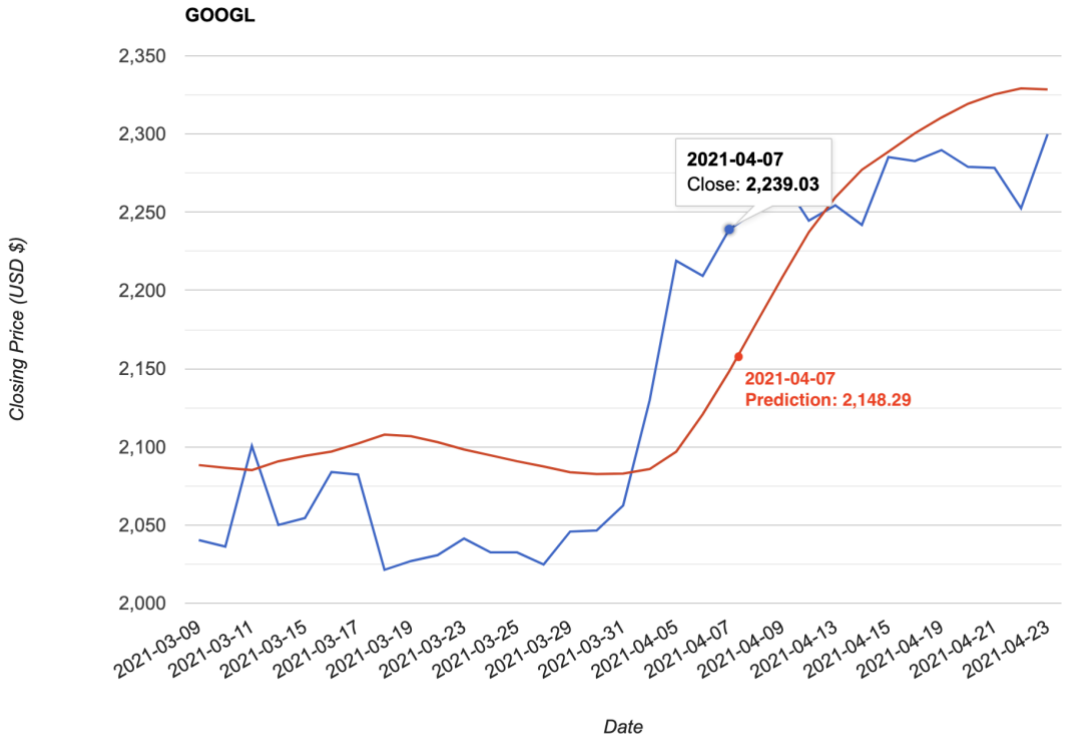

Figure 34: Actual vs Prediction Closing Price Discrepancy

Taking into account other data such as internet searches, communication between internet users, news articles, etc. can make a better prediction by understanding what the general sentiment is with respect to some stock price speculation. Understanding the general sentiment the public has with regards to a stock price can result in more accurate predictions. By anticipating whether the public wants to buy, sell, or feels like the stock price is overvalued or

undervalued, a more sophisticated algorithm might be able to predict sharp changes in prices better.

8.2 Add Additional Features to The Website

There are several additional features, and even additional User Stories, that were thought of as the website was being developed. Due to time constraints, these features and User Stories were not added to the scope or Product Backlog. Here's a list of some of the features that are worth noting for future enhancement efforts:

8.2.1 Add dashboard controls to the charts

Adding dashboard controls to the charts, beyond the simple tooltip, would enhance the user experience. Allowing the user to toggle a slider control in order to display data for a certain time range would be useful to the user (similar to stock charts provided by websites like E*TRADE and Yahoo Finance, which have interactive time ranges).

8.2.2 Add a loading bar to the ML Predictions page

Adding some form of visual indicator showing that the ML Prediction chart is being loaded would improve the user experience. Currently, the view takes some time (roughly 15 seconds) to retrieve the data and run the LSTM algorithm. The user has no way of knowing what is happening during this time. Although the time is relatively short, the user might wonder if the site is responsive when the computation time increases.

8.2.3 Making the stock symbol input field a search bar rather than a dropdown menu

Instead of having an extensive dropdown menu list listing all the stock symbols, the stock symbol should be searchable through a search bar. The stock symbols should be searchable by the symbol and the company name, and other related terms.

8.2.4 Adding more inputs to the ML Predictions page

The reason why only three input fields were included in the ML Predictions page was to keep things simple. However, there are many parameters that can be fed into the ML Predictions view in order to run the LSTM algorithm. Here are some of the inputs that might be worth adding in order to make the training of the LSTM recurrent neural network more interactive:

- Number of epochs
- Number of days in the past, from the present day, where the predictions should begin (the current setting is 60 days)
- The number of LSTM neurons (the current setting is 50 neurons)

8.3 Investing More Time and Effort on Testing, Verification, and Validation.

As described in Section 5, RoboStock was tested with enough thoroughness and detail to ensure that no glaring defects are present upon delivery. However, it must be recognized that more thorough and comprehensive testing can still be done. In addition to improved test cases, focus on verification and validation activities can improve the quality of RoboStock. Finally,

incorporating an automated test tool such as Selenium would add robustness to the testing activities and would reduce human labor timer.

8.3.1 Build More Thorough Test Cases

Developing test cases that trigger data queries from Yahoo Finance at different times would be ideal. It is possible that Yahoo Finance does not respond in a timely manner or does not respond at all. Testing unique edge cases could identify how the system behaves when other points outside the scope of the RoboStock web application don't behave as intended. This would in turn allow for better error-handling design.

8.3.2 Verification and Validation

Verification and Validation are two aspects of the project that should be expanded. Performing verification activities to confirm (or *verify*) that the code and work products are being built correctly would add a lot of value to the project. The validation of the RoboStock application was somewhat done through the User Story (and Use Case) testing. Through this testing, the suitability of the design for the design intent and use case was validated. However, more extensive validation activities should be performed with real-world scenarios and users to improve the confidence that what we built was the right tool for the job. In essence, more verification would better answer the question *"is what I'm building right?"* and validation would answer the question *"am I building the right thing?"* Lastly, it is not ideal that I tested the application since I was the developer as well. My testing execution might have been affected by the preconception of how the application *should* work form developing it.

8.3.3 More Test Automation

Django offers an automated test suite which was leveraged for Integration Testing (see Section 5.2.1). These tests were written in order ensure that the correct HTML Templates were rendered when a URL extension was called. This checked confirmed that the right URL Dispatcher/View/Template interactions took place. Automated testing could also be expanded to Acceptance Testing. The use of tools like Selenium can reduce the time required for manual testing. Regression testing with Selenium would allow for rapid testing whenever major modifications are made to the project/application during development.

9 Bibliography

[1] Bass, L., Clements, P., & Kazman, R. (2021). *Software Architecture in Practice 3/E (3rd ed.)*. PEARSON INDIA.

[2] Bernstein, S. H. (2018*). Buy Low / Sell High: A Commonsense Guide on Becoming a Better Investor* (1). BookBaby.

[3] Burkov, A. (2019). *The Hundred-Page Machine Learning Book*. Andriy Burkov.

[4] Burnstein, I. (2004). *Practical Software Testing: A Process-Oriented Approach*. Springer (India) Private Limited.

[5] Chen, N., 2020. CPSC 545: Software Design and architecture. Fullerton, CA.

[6] Computer Science. (2019, December 21). *Stock Price Prediction Using Python & Machine Learning [Video].* YouTube. https://www.youtube.com/watch?v=QIUxPv5PJOY&t=135s

[7] Cong, B. 2020. CPSC 546: *Modern Software Management*. Fullerton, CA.

[8] Django Software Foundation and individual contributors. (2005). *Django Documentation | Django.* Django. https://docs.djangoproject.com/en/3.2/

[9] Google Developers. (2020). *Google Charts*. https://developers.google.com/chart

[10] Highsmith, J. (2010). *Agile Project Management* (2nd ed.). Addison-Wesley.

[11] IEEE Std 829-1998: *IEEE Standard for Software Test Documentation*. IEEE, 1983. Web.

[12] ISO/IEC/IEEE 42010:2011(E): *Systems and software engineering – architecture description*. ISO/IEC/IEEE, 2011. Web

[13] James, M., & Walter, L. (2010). *Scrum Reference Card*.

[14] Jo, C. 2019. CPSC 544: *Advanced Software Process*. Fullerton, CA.

[15] K. Team. (2015, March 27). *Keras: The Python deep learning API*. Keras. https://keras.io/

[16] Otto, M. J. T. (2011). *Bootstrap 4.1 Documentation Introduction*. Bootstrap. https://getbootstrap.com/docs/4.1/getting-started/introduction/

[17] Pedregosa et al., *Scikit-learn: Machine Learning in Python*, JMLR 12, pp. 2825-2830, 2011.

[18] Python Software Foundation. (2021, April 26). *unittest — Unit testing framework — Python 3.9.4 documentation*. Python. https://docs.python.org/3/library/unittest.html

[19] Seth, S. (2020, March 7). *Basics of Algorithmic Trading: Concepts and Examples*. Investopedia. https://www.investopedia.com/articles/active-trading/101014/basics-algorithmic-trading-concepts-and-examples.asp

[20] Wiegers, K., & Beatty, J. (2013). *Software Requirements* (3rd ed.). Microsoft Press.

[21] Yahoo! (1997, January 19). *Yahoo Finance*. Yahoo Finance. https://finance.yahoo.com/

10 Appendix A: Disclaimer

DISCLAIMER: THIS PROJECT, PROJECT REPORT, PROJECT WORK PRODUCTS, AND WEB APPLICATION ARE FOR EDUCATION PURPOSES ONLY. ALL RELATED WORK INCLUDING DATA, MIDTERM AND FINAL REPORTS, DEMONSTRATIONS, SOURCE CODE, DO NOT INTEND TO PROVIDE ANY REAL SECURITIES, EQUITY, STOCK, BONDS, ETF, ETC. TRADING ADVICE. DO NOT USE ANY OF THIS INFORMATION FOR MAKING SECURITIES TRADING DECISIONS IN THE NEW YORK STOCK EXCHANGE OR ANY OTHER MARKET, DOMESTIC OR FOREIGN.