



**Dividend Stock Investment Strategy System**

|  |  |
| --- | --- |
| **Name** | Muhammad Aidan bin Abdul Shukor |
| **ID** | 100042177 |
| **Programme** | AmBank Structured Internship Programme |
| **Supervisor** | Dr. Chew Yew Choong |

|  |  |
| --- | --- |
| **Student’s Signature** |  |
| **Submission Date** | 5/6/2022 |
| **Supervisor’s Signature** |  |

# 1.0 Introduction

The first section discusses the introductory and preliminary information of AmBank: Dividend Stock Investment Strategy System. Subsections include Background, Stakeholders, Project Team and Glossary.

## Background

This project is established and developed for AmBank’s Structured Internship Programme to demonstrate the capacity of Data and Software Engineering within the Stock Market. Investing in stocks is considered as a dynamic environment that offers opportunities for financial growth and security. However, there are a lot of inherent uncertainties that makes informed decision a crucial aspect of successful investing. The Dividend Stock Investment Strategy System is an attempt to mitigate the uncertainty by providing a platform for calculating and analyzing financial ratios, with a focus on dividends.

Stock investment involves purchasing and selling shares of ownership in a company, entitling the shareholder to a portion of the company profits. The value of these shares can increase and decreased based on various factors, and one key element that can potentially offset this is dividends. Dividends are a portion of a companies earnings given to shareholders as a reward for investing in the company which can potentially mitigate risks and can also increase returns. However, companies that pay dividends often demonstrate financial stability and commitment to shareholder value, assuming they do so wisely considering their earnings. This commitment is crucial especially for long-term investors seeking not only profit returns but protect from losses.

## Project Overview

The Dividend Stock Investment Strategy System aims to demonstrate how the fusion of web development and data engineering can contribute to the creation of a robust investment strategy. By collecting data from diverse sources such as Bursa Marketplace, KLSE Screener, and Yahoo Finance, the system performs intricate calculations and analyses on fundamental financial and dividend ratios.

The heart of the project lies in the recognition that a well-informed investor is better equipped to navigate the uncertainties of the stock market. By utilizing web development tools, the system visualizes the results of these analyses, making complex financial information accessible to users in a user-friendly manner.

## Link to Project Work

The project is maintained at a GitHub repository and requires invitation for access. The link for the repository is provide below:

Link: [aidan-swin/Stock-Investment-Strategy: Internship - Stock Investment Strategy implemented with Python (github.com)](https://github.com/aidan-swin/Stock-Investment-Strategy)

# 2.0 System Analysis

The purpose of the System Analysis phase is to thoroughly investigate and understand the existing system or problem domain, identify user requirements and expectations, and define the scope and objectives of the proposed system.

## 2.1 Project Scope

In this project, scopes were set to define the boundaries of the project, such as what the project would and would not accomplish. The scopes that will be covered are defined systematically as follows:

* The project will involve the development of Dividend Stock Investment Strategy System that delivers a web application and various Python scripts.
* The project will involve fundamental research on stock investment for preliminary knowledge.
* The project will involve analysis and framing problem statement, project solution and architecture, as well as the tools and techniques necessary (ie. Web scraping).
* The project will involve research and analysis on various data sources and data quality assessment, to get the financial statements of stocks and the creation of advanced Python scripts to scrape these data accordingly.
* The project will involve preprocessing on collected data to ensure proper format before proceeding with further analysis.
* The project involves additional calculation to be implemented on Python scripts on scraped data to get more meaningful values for stock assessment.
* The project involves proper data formatting to be stored in a well-designed SQL database
* The project involves a web application that offers a subscription service that presents the calculation results, insights and recommendations towards users to help them make informed investment decisions.
* The project involves the research of backtesting, simulation and technical analysis to be implemented in the web application to aid users the right time to purchase a particular stock.
* The project involves basic operational functionalities in the web application such as login and administration process.

The scope that will be not within this project is mainly in the frontend aspect, where no advanced implementations or frameworks were carried out. It was also decided to develop one front-end platform, which is the web application, with no plans for a secondary platform such as a mobile application. And of course with the nature of stock investments, further recommendations in stock investments are never concluded with guaranteed certainty but as a guideline and estimation.

## 2.2 Functional Requirements

The products’ specification components will be covered in this section. As the name suggests, it alludes to a particular product feature that will be created as part of this project. It includes topics such as what inputs the system accepts, how it processes data, and what results it produces:

* To gather and save the basic stock information such as stock code, stock name and the industry type.
* The system must collect historical price for every stock code data efficiently and correctly.
* The system must collect and save historical dividend data for every stock code, including the announce date and payment date .
* The system must collect and save quarterly report values for every stock code which are net income, preferred dividends, total common shares outstanding, operating income, total equity, revenue, cash from operating activities and capital expenditures.
* The system must utilize collected data to calculate more meaningful financial ratios in Trailing 12 Months such as Earnings Per Share, Price to Earnings, Return on Equity, Operating Margins, Dividend Yield, Payout Ratio and Free Cash Flow
* The system must assess company performance by assessing each of the ratios to classify them based on rank “S”, “A”, “B”, “C” and “D”.
* The system must store these collected and calculated data into a well designed and normalized SQL Database.
* The system must involve a front-end interface (web application) that performs the CRUD process with the database.
* The system web application must display web pages for every stock code and their respective collected, calculated data, and recommendations for subscribed users.
* The system must have a watchlist function where users can add their desired (or highly ranked) companies for tracking purposes.
* The system must have a portfolio function that simulates when users buy a stock at a certain quantity and date, and displaying the real time dividend and price status accordingly.
* The system web application must contain technical analysis for every stock code to communicate and recommend users the best time to purchase a stock.
* The system must be able to update the database to collect any new prices, dividends and quarterly reports from external sources, and calculate any new financial ratios accordingly
* The system web application must contain charts and visualization techniques on historical prices, dividends, financial ratios, and portfolio performance to see trends.
* The system must contain login process and stored user data according to their watchlist and portfolios.
* The system must contain admin users where they can carry out admin operations such as adding, edit and delete user, as well as update all stocks.

## 2.3 Non-Functional Requirements

The non-functional requirements are the other aspect of a product demand that also has to be met. The system’s capabilities and quality characteristics as they relate to how it fulfills its functional requirements are described in this section. The following are the functional requirements for the proposed system.

* The system should have a strong processing performance and should be able to produce responses less than 1 second after any queries are received. The system should be able to function on lower-end hardware, such as on a single core CPU, to accommodate as many devices as feasible.
* The system should be interoperable with a variety of desktop operating systems, including Windows, Mac OS, and Linux. It should also adhere to best standards for user experience
* To provide the greatest availability, the system should able to achieve at least 99.999% uptime annually.

## 2.4 Software Required

The software required to develop the system are as follows:

* Python (and its related stock libraries)
* Anaconda Navigator
* Jupyter Notebook
* Visual Studio Code
* Flask-SQLAlchemy
* backtesting.py
* driver.py
* Beautiful Soup.py
* Tabula.py
* Yfinance.py

## 2.5 Hardware Required

The hardware required to develop the system are as follows:

* Storage (Store database data)
* Device (Development Environment, to open Browser to access the web application, to run and deploy backend web scraping, calculation and backend process)
* Network adapter (Connect to the network for connection to access the webpage via web server)

# 3.0 System Design

This section will discuss the overall design of the system, such as how the system architecture works in a high level, as well as the database design that illustrates how data stored data is structured.

## 3.1 System Architecture

A screenshot of a computer

Description automatically generated

Figure 3.1.1 – System Architecture

The figure above illustrates how the system architecture looks like, as well as the processes involved to assemble the complete product. Firstly, a data source is needed to gather the basic information of the stock market, such as the stock code, their historical prices, quarterly financial reports, and dividend history. This information is available publicly in various stock websites. Upon further research, the websites that will be focused on are Bursa Marketplace, Yahoo Finance and KLSE Screener, since gathering from these three will result in all the data required for this project.

The three targeted website will be used to carry out web scraping process with the use of Python scripts that will collect all the data into a DataFrame. Each website requires different techniques required to web scrape properly, with Yahoo Finance requiring an API, KLSE Screener and Bursa Marketplace requiring complex HTML parsing and pathing to target the specific element and retrieving the value.

Once the large amount of data has been scraped and preprocessed into a DataFrame, further calculation is required to come up with more values. These new values are useful financial ratios in assessing the company’s performance in the stock market such as Earnings per Share, Price to Earnings and so on. Afterwards, each stock code is automatically assessed and graded accordingly based on their financial ratios which will be the system’s recommendation for potential investment. All of the collected and calculated data will be loaded in a SQL Database which can then be used to display on a website developed with a Flask Framework.

## 3.2 Database Design (Entity Relationship Diagram)

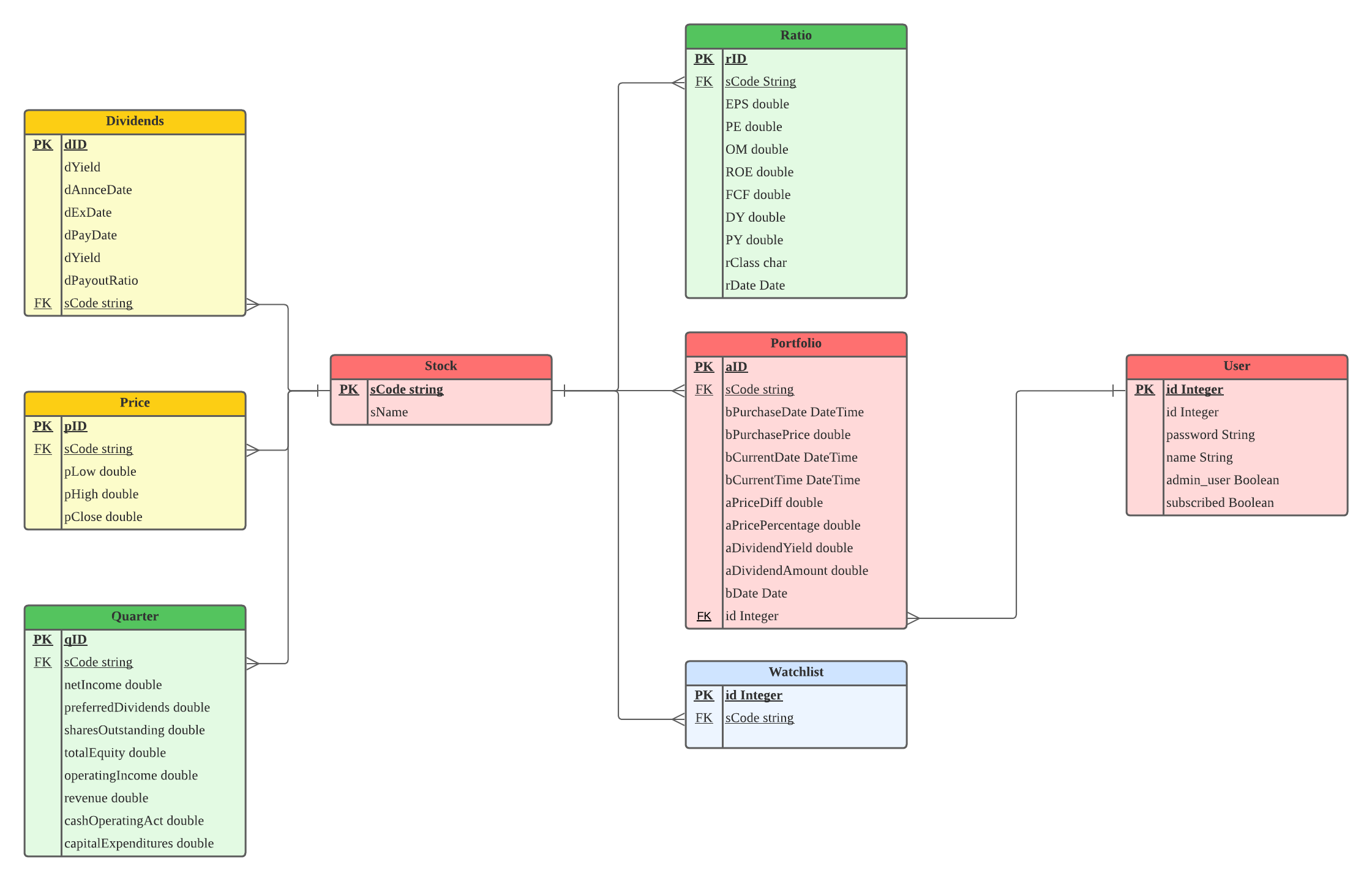


Figure 3.2.2 – Entity Relationship Diagram

The figure above shows the database relational design with the Entity Relationship Diagram. The tables are normalized accordingly based on proper techniques and ensure no data redundancy are present. Firstly, the center table contains the basic stock code information such as the stock name. Each stock code will contain records that were scraped and calculated in the Python Backend which are Dividends, Prices, Quarterly Reports and Financial Ratios. These tables will be populated accordingly when the Load process in the system is being carried out.

Focusing on tables that are mainly the website’s functionality, the database also contains a user table that contains the login information and account type that determines the access level of the web application. Every stock code may contain inside of a user’s Portfolio which simulates the event that the user has invested in them. It will keep records of the purchase date and purchase amount where the system can load and calculate the live analysis of their investments. Finally, stock codes may be contained inside of a watchlist depending on their graded stock performance analyzed by the Python Backend.

# 4.0 System Development

This chapter discusses the development phase of the project. It outlines the development environment, version control and repository.

## 4.1 Development Environment

The development environment specifications involve programming languages, frameworks, libraries, tools and technologies utilized during system development. This section discusses the overview of software and hardware infrastructures used.

**Tools Used:**

1. Jupyter Notebook. Jupyter Notebook is an open-source web application that allows to create and share documents containing live code, equations, visualizations and narrative text. It provides and interactive environment for data collection and analysis during the ETL process of the system. It supports the Python programming language which makes it an ideal choice for the tasks outlined.
2. Visual Studio Code. A lightweight and versatile source code editor that offers a wide range of features, such as syntax highlighting, code completion and debugging. The Integrated Development Environment houses many features suited for various development frameworks. It is an ideal choice in creating the Flask web application.

**Libraries Used:**

1. Flask: A micro web framework written in Python. It is designed to be simple, lightweight and easy to understand. It provides the necessary tools and utilities to build web applications that follows the Web Serve Gateway Interface, supporting various extensions for added functionality. Its simplicity and flexibility makes it an ideal choice in developing our Dividend Stock Investment Strategy System website.
2. SQLAlchemy: Python SQL toolkit and Object-Relational Mapping (ORM) library. It provides a set of high-level APIs for interacting with databases such as SQL, allowing to work with relational databases that houses the collected and calculated data. It offers consistent API for querying, inserting, updating and deleting data.
3. Pandas. Popular open-source data manipulation and analysis library for python. Used for its wide range of functions and methods to clean, transform, filter, and aggregate data that works very well with other libraries.
4. Bootstrap. A front-end framework that allows making simplistic, stylish, and professional looking website much easier.
5. Chart.js. An integral part of the web application that displays interactive charts dynamically based on loaded data in the database.

**Languages Used:**

1. HTML, CSS, Javascript. The fundamental building blocks of the web development. These three languages work together to create engaging and responsive web interfaces. In the project, they are used to build the frontend of the web application, defining the structure, presentation, and behaviour of the user interface.
2. Python. Used for any data science and engineering processes due to its rich ecosystem of libraries and simplicity.

## 4.2 Version Control & Repository

The version control system was employed to manage the source code. Version control was used to track changes, collaborate with the supervisor, and ensure code integrity.

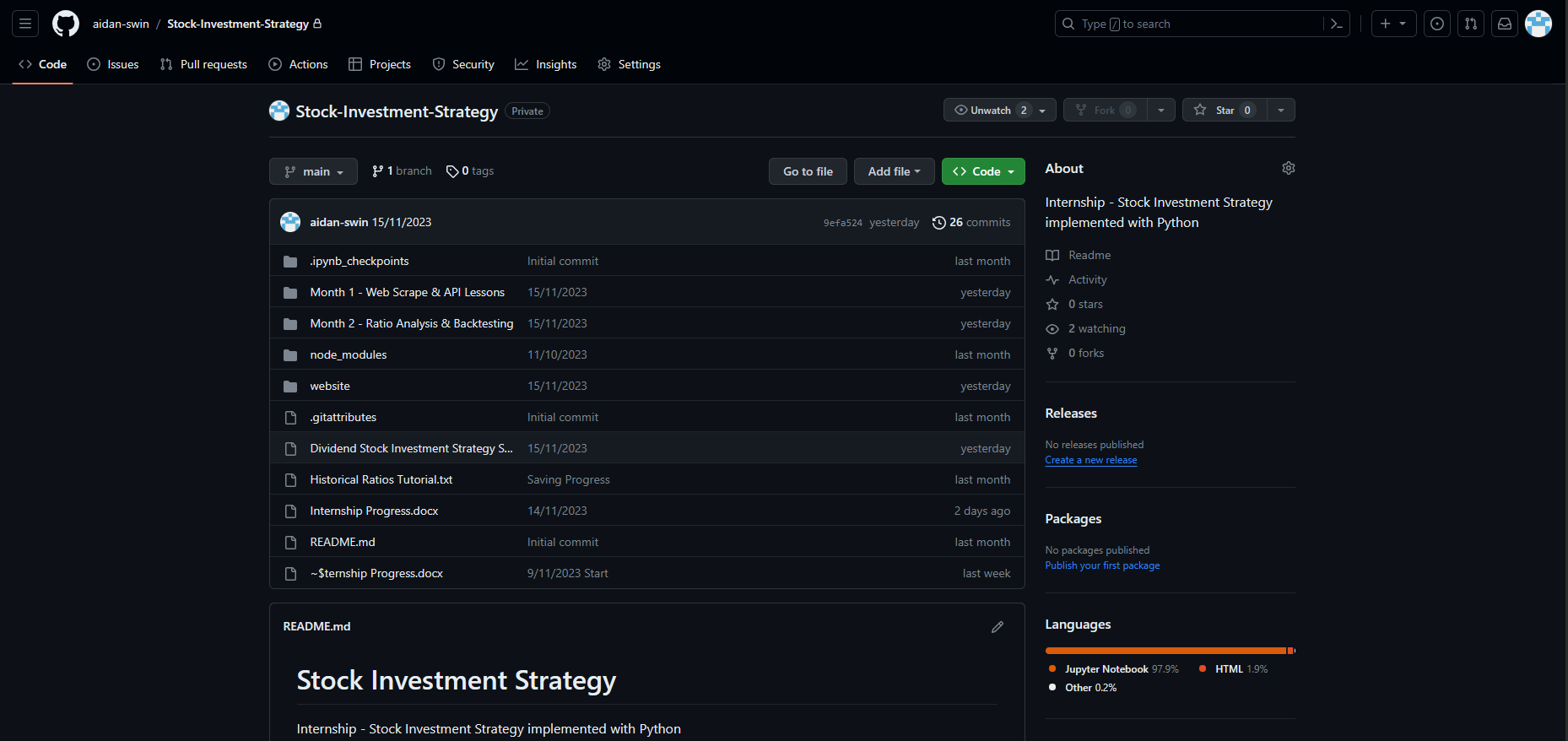


Figure 4.2.1 – Project GitHub Repository

GitHub is a web-based version control platform that enables teams to collaborate on software development projects. The repository provides a platform to store and manage the code, allowing members to work and supervise the project simultaneously and real time. It allows to clone the project repository locally, make changes in the local environment, and then push the changes back to GitHub, ensuring that everyone has access to the latest version of the codebase. It also provides as means as backup where every commit can be restored to the previous version of the system.

Link: [aidan-swin/Stock-Investment-Strategy: Internship - Stock Investment Strategy implemented with Python (github.com)](https://github.com/aidan-swin/Stock-Investment-Strategy)