AUTOMATED PRICE TRACKING AND AVAILABILITY NOTIFICATIONS

VIA DISCORD BOT INTEGRATION

by

OGUZ KAAN YILDIRIM

M.S. Harrisburg University of Science and Technology,

USA, 2024

A Proposal submitted in partial fulfillment of the requirements

for the degree of Master of Science

in the Computer Information Sciences at the

Harrisburg University of Science and Technology, Pennsylvania

Late Spring Term 2024

# ABSTRACT

In an increasingly digital world, automation has become crucial for enhancing efficiency and user experience. This project focuses on the development of a Discord bot system designed to automate the tracking of product prices and the availability of services, providing timely notifications to users. The bot leverages a combination of web scraping, data extraction, and notification delivery to keep users informed about price changes and available dates for desired products or services. The system integrates various subsystems, including authentication, product management, notification handling, data management, user interaction, and availability checking.

The Authentication Subsystem ensures secure access to the system by managing user login processes. The Product Management Subsystem retrieves product details and monitors price fluctuations. The Notification Subsystem keeps users informed through timely notifications. The Data Management Subsystem handles data storage and extraction operations, while the Interaction Interface Subsystem ensures smooth communication between users and the bot. The Availability Check Subsystem handles the verification of date availability for products and services.

The Discord bot employs a modular architecture, allowing for efficient execution of diverse tasks. Key components include the User, Account, Product, Date, Command, DiscordBot, and several control and interface objects, such as LoginControl, PriceCheckControl, AvailabilityCheckControl, and ExcelExportControl. These components interact seamlessly to provide a comprehensive automation solution.

This project aims to enhance user experiences by automating routine tasks, ensuring users stay informed and can make timely decisions based on the latest data. The bot system not only saves time but also provides a reliable method for tracking important information, ultimately empowering users in their day-to-day digital interactions.

# ACKNOWLEDGMENTS

I would like to take this opportunity to thank my Professor, Dr. Abdel Ejnioui, for his assistance and contribution.

I would like to express my sincere gratitude to my faculty and my university for their invaluable guidance and support. I am deeply appreciative of my friends for their encouragement, and I extend heartfelt thanks to my brother and family for their unwavering support throughout this journey.

# TABLE OF CONTENTS

[ABSTRACT i](#_Toc169196492)

[ACKNOWLEDGMENTS iii](#_Toc169196493)

[TABLE OF CONTENTS iv](#_Toc169196494)

[LIST OF FIGURES vi](#_Toc169196495)

[LIST OF TABLES vii](#_Toc169196496)

[LIST OF ACRONYMS/ABBREVIATIONS viii](#_Toc169196497)

[CHAPTER ONE: INTRODUCTION 1](#_Toc169196498)

[1.1 Goals And Objectives 1](#_Toc169196499)

[1.2 Motivation Of the Project 1](#_Toc169196500)

[1.2.1 Saving Time and Reducing Stress 2](#_Toc169196501)

[1.2.2 Financial Savings 3](#_Toc169196502)

[1.3 Context and Relevance of Application 4](#_Toc169196503)

[1.3.1 General Features of E-commerce Price Monitoring Tools 4](#_Toc169196504)

[1.3.2 Role of Automation in Service Booking and Availability Checking 4](#_Toc169196505)

[1.3.3 Technological Integration and Advancements 5](#_Toc169196506)

[1.3.4 Future Prospects and Impact on User Experience 6](#_Toc169196507)

[1.4 Benefits for Users 6](#_Toc169196508)

[1.4.1 Time Efficiency 6](#_Toc169196509)

[1.4.2 Financial Savings 6](#_Toc169196510)

[1.4.3 Reduced Stress 7](#_Toc169196511)

[1.4.4 Enhanced User Experience 7](#_Toc169196512)

[1.5 General Context of the Project 7](#_Toc169196513)

[1.5.1 Technological Advancements 7](#_Toc169196514)

[1.5.2 Industry Trends 7](#_Toc169196515)

[1.5.3 Market Impact 8](#_Toc169196516)

[1.5.4 Societal Implications 8](#_Toc169196517)

[1.6 Summary and Thesis Outline 8](#_Toc169196518)

[CHAPTER TWO: RELATED WORK 10](#_Toc169196519)

[2.1 Review of Existing Systems 10](#_Toc169196521)

[CHAPTER THREE: SYSTEM DESING AND IMPLEMENTATION 16](#_Toc169196522)

[3.1 Project Requirements 16](#_Toc169196524)

[3.1.1 Use Case Diagram: 16](#_Toc169196525)

[3.1.2 Use Case Descriptions: 17](#_Toc169196526)

[3.2 Architecture 19](#_Toc169196527)

[3.2.1 Component Diagram 20](#_Toc169196528)

[3.2.2 Component Descriptions: 21](#_Toc169196529)

[3.2.3 Deployment Diagram 22](#_Toc169196530)

[3.2.4 Activity Diagram 23](#_Toc169196531)

[3.2.5 Mapping Objects to Persistent Store 24](#_Toc169196532)

[3.2.6 Map components to different layers in the project architecture 27](#_Toc169196533)

[3.3 Design 30](#_Toc169196534)

[3.3.1 Package Diagram 30](#_Toc169196535)

[3.3.2 Class Diagram 34](#_Toc169196536)

[3.4 Technology Stack/Framework 40](#_Toc169196537)

[3.4.1 Programming Languages and Frameworks: 40](#_Toc169196538)

[3.4.2 Tools and Platforms: 40](#_Toc169196539)

[3.4.3 Database: 42](#_Toc169196540)

[3.4.4 Deployment: 42](#_Toc169196541)

[3.5 Conclusion 43](#_Toc169196542)

[LIST OF REFERENCES 46](#_Toc169196543)

# LIST OF FIGURES

[Figure 1: Trends in online shopping and booking services [1]. 2](#_Toc169196984)

[Figure 2: Distribution of Time Spent Per Shopping [3] 3](#_Toc169196985)

[Figure 3: Google Flight User Interface [19] 11](#_Toc169196986)

[Figure 4: Keepa User Interface [21] 12](#_Toc169196987)

[Figure 5: Use Case Diagram 17](#_Toc169196988)

[Figure 6: Component Diagram 20](#_Toc169196989)

[Figure 7: Deployment Diagram 22](#_Toc169196990)

[Figure 8: Architectural Diagram 29](#_Toc169196991)

[Figure 9:Authentication Subsystem 31](#_Toc169196992)

[Figure 10: Product Management Subsystem 31](#_Toc169196993)

[Figure 11: Notification Subsystem 32](#_Toc169196994)

[Figure 12: Save Price, Availability Subsystem 33](#_Toc169196995)

[Figure 13: Interaction Interface Subsystem 33](#_Toc169196996)

[Figure 14: Availability Check Subsystem 34](#_Toc169196997)

[Figure 15: All Classes 39](#_Toc169196998)

[Figure 16: Technology Stack Diagram 41](#_Toc169196999)

[Figure 17: Architecture Diagram 42](#_Toc169197000)

# LIST OF TABLES

[Table 1: Comparison of Key Features 14](#_Toc169196552)

[Table 2: Accounts 24](#_Toc169196553)

[Table 3: Products 24](#_Toc169196554)

[Table 4:Notifications 25](#_Toc169196555)

[Table 5: Users 25](#_Toc169196556)

[Table 6:DiscordBots 25](#_Toc169196557)

[Table 7: Dates 26](#_Toc169196558)

# LIST OF ACRONYMS/ABBREVIATIONS

API: Application Programming Interface

AS: Authentication Subsystem

DTO: Data Transfer Object

DAO: Data Access Object

EH: External Helpers

HTML: HyperText Markup Language

HTTP: HyperText Transfer Protocol

HTTPS: HyperText Transfer Protocol Secure

IDE: Integrated Development Environment

IIS: Interaction Interface Subsystem

NS: Notification Subsystem

PMS: Product Management Subsystem

SQL: Structured Query Language

SPAS: Save Price, Availability Subsystem

UML: Unified Modeling Language

URL: Uniform Resource Locator

# CHAPTER ONE: INTRODUCTION

This chapter introduces the PriceTracker project, outlining its goals and objectives, motivations, and the importance of the application classes to which it belongs. It also details the benefits for users and provides the general context of the project, including technological advancements, industry trends, market impact, and societal implications. Each section aims to give a comprehensive overview of the project's foundation, setting the stage for the detailed discussions in the subsequent chapters.

## Goals And Objectives

The primary goal of this project is to develop an automated Discord bot system, named PriceTracker, designed to monitor product prices and the availability of services. This project aims to provide users with timely notifications about price changes and available dates for desired products or services.

## Motivation Of the Project

In today's digital age, online shopping, booking services, and price comparison have become integral parts of daily life. Consumers frequently spend considerable time and effort to monitor prices and check the availability of products and services. This project's motivation stems from the need to streamline these activities and provide a more efficient, less stressful experience for users. It can be clearly seen in Figure 1 how much online shopping has increased over the years.

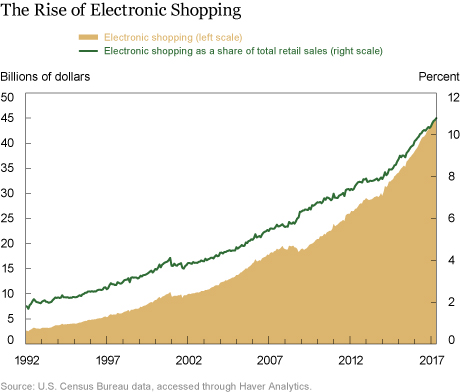


Figure 1: Trends in online shopping and booking services [1].

### Saving Time and Reducing Stress

One of the primary motivations for developing the PriceTracker bot is to save user’s time. The report indicates that UK adults spend an average of more than three-and-a-half hours online each day, engaging in various online activities, including shopping and price comparison [2]. This includes activities such as comparing prices across different websites, monitoring price fluctuations, and checking the availability of dates for bookings. By automating these tasks, the PriceTracker bot significantly reduces the time users need to spend on these activities. In Figure 2, we can see that people spend almost 15% of their time checking prices for the same product [3].

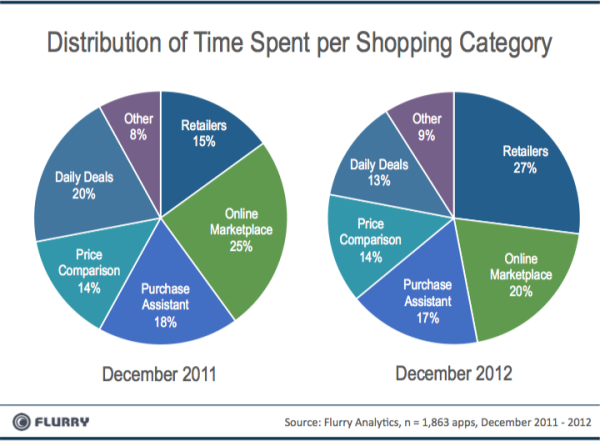


Figure 2: Distribution of Time Spent Per Shopping [3]

### Financial Savings

The average consumer spends approximately $1,200 annually on impulse purchases, often driven by price fluctuations and limited time offers. Another crucial motivation is the potential for financial savings. Product prices, plane tickets, and hotel rates can fluctuate significantly within short periods. For instance, a report by Hopper found that prices for airline tickets can change by an average of 20% per month due to factors like jet fuel prices and seasonal demand. Similarly, product prices on e-commerce platforms like Amazon can vary by up to 30% depending on the time and date. By receiving timely notifications about price drops and availability changes, users can capitalize on the best deals and avoid overpaying [4, 5]. The PriceTracker bot aims to address these issues by automating the tracking process and providing timely notifications.

## Context and Relevance of Application

The PriceTracker bot is part of a broader category of tools designed to enhance consumer decision-making in e-commerce and service booking. This section explores the general context of similar applications and highlights the unique aspects of the PriceTracker bot.

### General Features of E-commerce Price Monitoring Tools

Applications in the realm of e-commerce price monitoring typically provide functionalities that allow users to track the prices of products across various platforms. These tools enable consumers to:

* Monitor price fluctuations in real-time.
* Set alerts for price drops.
* Compare prices across different sellers to find the best deals.
* Receive notifications about price changes and promotional offers.

Research indicates that price tracking tools are increasingly popular among consumers due to the dynamic nature of online pricing, which can change based on factors such as demand, competition, and seasonal variations. According to a study by Statista, the global adoption of e-commerce tools that assist in price monitoring and comparison is expected to grow significantly over the next few years [6].

### Role of Automation in Service Booking and Availability Checking

Automation tools in the service booking industry are designed to help users efficiently manage their bookings and check availability for services such as flights, hotels, and car rentals. These tools typically offer features such as[7]:

* Automated searches for the best booking deals.
* Notifications about changes in availability.
* Integration with various booking platforms to streamline the user experience.
* Predictive analytics to suggest the best times to book.

The use of automation in this context is driven by the need to handle large volumes of data and provide timely information to users, reducing the manual effort required to find and secure the best deals[8]. Studies have shown that consumers appreciate the convenience and time savings provided by these automated tools[9].

### Technological Integration and Advancements

Technological advancements in web scraping, data analysis, and automated notifications have significantly improved the functionality of tools like the PriceTracker bot. Key technological features include:

* **Web Scraping**: This technology allows the bot to collect data from various websites, providing real-time updates on product prices and availability[10].
* **Data Analysis**: Advanced algorithms process the collected data to identify trends and generate meaningful insights for users[11].
* **Automated Notifications**: Users receive timely alerts through various communication channels, ensuring they are always informed about important changes[12].

These technological integrations not only enhance the efficiency and accuracy of such tools but also contribute to a seamless user experience. As technology continues to evolve, these tools are expected to become even more sophisticated, offering more advanced features and greater reliability.

### Future Prospects and Impact on User Experience

The ongoing development of price tracking and booking automation tools holds significant promise for improving user experiences in e-commerce and service booking[13]. Future enhancements might include:

* More accurate predictive analytics to forecast price changes.
* Enhanced integration with a wider range of platforms and services.
* Increased personalization based on user preferences and behaviors.

As these tools become more advanced, they will likely play a critical role in helping consumers make smarter purchasing decisions, save money, and reduce the stress associated with manual monitoring of prices and availability.

## Benefits for Users

### Time Efficiency

The PriceTracker bot significantly reduces the time users spend on monitoring prices and availability. Instead of manually checking multiple websites, users receive automated notifications about changes, allowing them to focus on other tasks. This efficiency is particularly beneficial for busy individuals who need to manage their time effectively.

### Financial Savings

By alerting users to price drops and availability changes, the bot helps them make cost-effective decisions. Users can purchase products at lower prices and book services at more favorable rates, resulting in substantial financial savings over time. McKinsey report indicates that consumers who use price tracking tools save an average of 10-15% on their purchases [14].

### Reduced Stress

The bot alleviates the stress associated with constantly monitoring prices and availability. Users no longer need to worry about missing out on deals or checking for updates repeatedly. The peace of mind provided by timely notifications allows users to relax and feel confident in their purchasing decisions.

### Enhanced User Experience

The PriceTracker bot enhances the overall user experience by providing a convenient and reliable service. Its integration with Discord ensures that users can easily interact with the bot, receive updates, and manage their preferences. The user-friendly design and automated functionality contribute to a seamless and enjoyable experience.

## General Context of the Project

### Technological Advancements

The development of the PriceTracker bot is rooted in automation. This technology provides users with accurate and timely information. As technology continues to evolve, the capabilities of the bot will also expand, offering even more sophisticated features and functionalities.

### Industry Trends

The e-commerce and travel industries are rapidly evolving, with increasing reliance on digital tools and automation. Consumers are becoming more tech-savvy and demand solutions that enhance their online experiences. According to [Statista, 2020], the number of digital buyers worldwide is expected to surpass 2.14 billion by 2021. The PriceTracker bot aligns with these trends, offering a tool that meets the needs of modern consumers [15].

### Market Impact

The market impact of the PriceTracker bot is significant, as it addresses a common pain point for consumers: the need to monitor prices and availability. By providing a reliable and efficient solution, the bot has the potential to attract a large user base and generate substantial value. The bot's ability to save time and money for users also contributes to its market appeal and competitiveness.

### Societal Implications

The PriceTracker bot contributes to the broader trend of digital automation, which has far-reaching implications for society. Automation tools like the bot simplify everyday tasks, making life more convenient and efficient for users. Additionally, the bot's ability to help users save money can have positive economic impacts, especially for budget-conscious consumers. As automation becomes more integrated into daily life, tools like the PriceTracker bot exemplify how technology can improve quality of life and drive innovation.

## Summary and Thesis Outline

In this chapter, we introduced the PriceTracker project by discussing its goals, objectives, and motivations. We explored the importance of the application classes to which the project belongs, highlighting the significant impact it can have in the e-commerce and travel industries. We also detailed the benefits for users, such as time efficiency, financial savings, and reduced stress. Furthermore, we provided the general context of the project, including technological advancements, industry trends, market impact, and societal implications. This foundational overview sets the stage for the following chapters, where we will delve deeper into related work, project design, implementation, and findings.

Chapter 2 will discuss and summarize previously proposed work related to the PriceTracker project. It will include a comparison of similar tools and technologies, highlighting their strengths and weaknesses in relation to this project.

Chapter 3 will present the detailed design and implementation of the PriceTracker bot. This chapter will cover the system architecture, subsystem design, and the technical aspects of the project. It will include UML diagrams, use cases, and detailed descriptions of each component.

Chapter 4 will summarize the findings and results of the project. It will reflect on the project's impact, evaluate its success in meeting the initial objectives, and discuss any challenges faced during implementation. Additionally, this chapter will offer suggestions for future improvements and potential areas for further research.

# CHAPTER TWO: RELATED WORK

In this chapter, we review existing systems and projects that are comparable to the PriceTracker bot. By examining these systems, we aim to understand their features, strengths, and limitations, and how they compare to our project. This comparative analysis will help identify the unique contributions of PriceTracker and areas for potential improvement. We will focus on three key examples: Google Flights, Keepa, and a Discord Bot project on GitHub. The chapter concludes with a summary of the comparisons and insights gained from this review.



## Review of Existing Systems

The systems we will discuss include Google Flights, Keepa, and a GitHub-based Discord Bot project. These systems represent a range of applications from travel booking to e-commerce price tracking and open-source software development.

According to a report by Invoca, "45 Statistics Retail Marketers Need to Know in 2024," consumers increasingly rely on price tracking tools like Keepa to monitor product prices and make informed purchasing decisions [16]. Such systems help users save money and ensure they get the best deals available online. Similarly, a study by Saleslion reveals that "81% of Shoppers Conduct Research Before Purchase," highlighting the significance of tools like Google Flights in enabling users to compare flight prices and find the most cost-effective travel options [17].

We will examine the features, advantages, and limitations of each of these systems to provide a detailed comparison and analysis. This review will help us understand the current landscape of price tracking and comparison tools, setting the stage for a more in-depth discussion of the PriceTracker bot's unique value propositions in subsequent sections.

*Google Flights*

Google Flights is a travel fare aggregator that provides price comparisons for flights. It offers features such as price tracking, price history, and alerts for price changes. Users can search for flights, compare prices across different airlines, and receive notifications about fare changes [18].

A screenshot of a computer

Description automatically generated

Figure 3: Google Flight User Interface [19]

* **Key Features**:
  + Price tracking and alerts
  + Comprehensive search for flights
  + Historical price data
  + User-friendly interface
* **Comparison to PriceTracker**:
  + Google Flights focuses on the travel industry, specifically flights, whereas PriceTracker aims to support various product categories.
  + Both systems offer price tracking and notifications, but PriceTracker integrates directly with e-commerce platforms and uses web scraping for real-time data.

*Keepa*

Keepa is a price tracking tool specifically for Amazon products. It provides detailed price history charts, price drop alerts, and browser extensions for easy access. Keepa tracks prices and offers a comprehensive overview of product price trends [20].

A screenshot of a computer

Description automatically generated

Figure 4: Keepa User Interface [21]

* **Key Features**:
  + Price history charts
  + Price drop alerts
  + Browser extensions
  + Multi-region price tracking
* **Comparison to PriceTracker**:
  + Keepa is limited to Amazon products, while PriceTracker aims to support multiple e-commerce platforms.
  + Both tools provide price tracking and notifications, but PriceTracker's broader scope allows for a wider range of product monitoring.

*Discord Bot (GitHub Project)*

This GitHub project is an open-source Discord bot that can be customized for various functionalities. It provides a foundation for building bots that can interact with users on Discord, perform automated tasks, and integrate with other APIs.

* **Key Features**:
  + Customizable bot functionality
  + Interaction with users on Discord
  + Integration with external APIs
  + Open-source and community-driven
* **Comparison to PriceTracker**:
  + The GitHub project serves as a foundation for building custom bots, like PriceTracker's use of Discord for notifications and interactions.
  + PriceTracker's specific focus on price tracking and product availability differentiates it from the more general-purpose nature of the GitHub project.
  1. **Comparison of Features**

*Price Tracking and Alerts*

Both Google Flights and Keepa provide robust price tracking and alert systems. Google Flights focuses on flights, while Keepa tracks Amazon product prices. PriceTracker combines these functionalities across multiple e-commerce platforms, offering users a versatile tool for tracking various product prices.

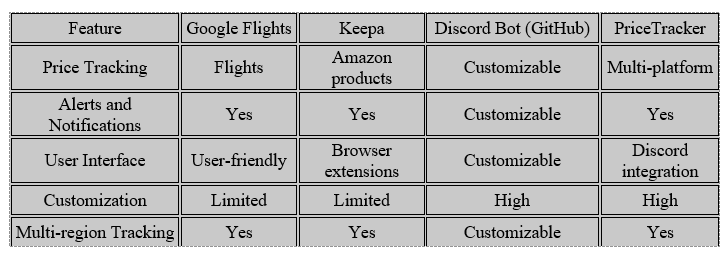
*User Interface and Usability*

Google Flights is known for its user-friendly interface and comprehensive search capabilities. Keepa provides detailed price charts and browser extensions for easy access. PriceTracker aims to provide a seamless user experience by integrating with Discord, allowing users to interact with the bot through a familiar platform.

*Scope and Customization*

The GitHub Discord Bot project offers a high level of customization, enabling developers to build bots for different purposes. PriceTracker leverages this flexibility to create a specialized bot for price tracking and notifications, integrating with multiple e-commerce platforms.

Table 1: Comparison of Key Features



* 1. **Advances and Limitations**

*Advances in Price Tracking*

Google Flights and Keepa have advanced the field of price tracking with their specialized focus areas. Google Flights excels in flight fare aggregation, while Keepa provides detailed Amazon price histories. PriceTracker builds on these advances by offering a comprehensive solution that tracks prices across multiple platforms, leveraging web scraping and real-time data processing.

*Limitations*

Google Flights and Keepa are limited by their specific domains—flights and Amazon products, respectively. The GitHub Discord Bot project, while highly customizable, requires significant development effort to tailor it to specific needs. PriceTracker addresses these limitations by providing a ready-to-use solution that integrates multiple functionalities, though it may face challenges in ensuring data accuracy and handling diverse product categories.

* 1. **Conclusion**

This chapter reviewed existing systems comparable to PriceTracker, including Google Flights, Keepa, and a GitHub Discord Bot project. We compared their features, highlighted advances and limitations, and identified areas where PriceTracker offers unique contributions. This analysis provides a foundation for understanding the competitive landscape and potential improvements for PriceTracker. In the next chapter, we will delve into the detailed design and implementation of the PriceTracker bot, building on the insights gained from this comparative review.

# CHAPTER THREE: SYSTEM DESING AND IMPLEMENTATION

This chapter covers the system design and implementation details of the PriceTracker bot. We will start with the project requirements, including the use case diagram and descriptions of the use cases. Then, we will present the architecture, including UML component, deployment, and activity diagrams. The design section will include UML package and class diagrams. Finally, we will discuss the technology stack and frameworks used, followed by a short conclusion.



## Project Requirements

In this section, we will cover the project requirements, including the use case diagram and detailed descriptions of the use cases. We will also integrate relevant parts from assignments to provide a comprehensive understanding.

### Use Case Diagram:

The use case diagram illustrates the interactions between users and the PriceTracker bot, detailing the various functionalities provided.

* **Actors:** User, External Helpers (EH)
* Use Cases:
  + Chat with Bot
  + Launch Chrome
  + Login to Web Application
  + Check Price of the Product
  + Check Available Dates
  + Extract Data to Excel
  + Receive Notifications or Emails

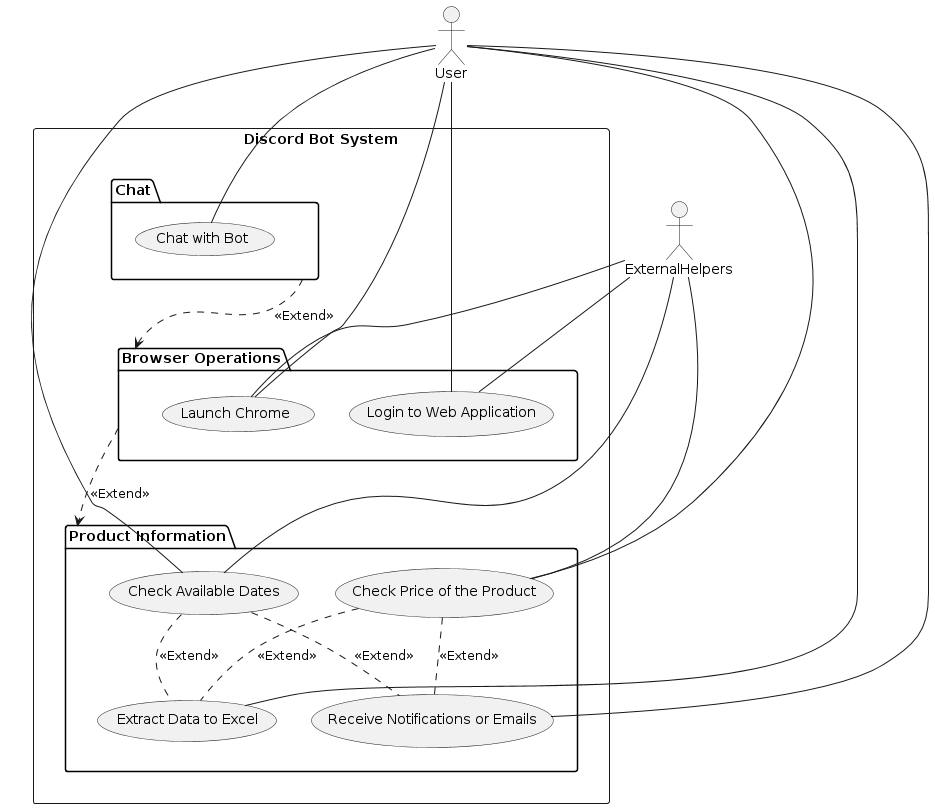


Figure 5: Use Case Diagram

### Use Case Descriptions:

Register

* **Description:** Allows a user to create an account.
* Actor: User
* **Precondition:** User is not logged in.
* **Postcondition:** User account is created.
* **Details from Assignments:** The registration process involves the user providing their details, which are then stored securely in the system's database.

Login

* **Description:** Allows a user to log into their account.
* Actor: User
* **Precondition:** User has an account.
* **Postcondition:** User is logged in.
* **Details from Assignments:** As detailed in earlier Assignments, the login process verifies the user's credentials against stored data, ensuring secure access.

Track Product Price:

* **Description:** Enables the user to track the price of a product.
* Actor: User
* **Precondition:** User is logged in.
* **Postcondition:** Product is added to the tracking list.
* **Details from Assignments:** As outlined in earlier Assignments, this functionality involves the bot using web scraping to monitor the price of a product and update the user accordingly.

Check Availability:

* **Description:** Allows the user to check the availability of a product or service.
* Actor: User
* **Precondition:** User is logged in.
* **Postcondition:** Availability information is provided.
* **Details from Assignments:** This use case involves checking the availability of items such as flights or hotel bookings and notifying the user of any changes.

Receive Notifications:

* **Description:** Sends notifications to the user about price changes and availability.
* Actor: User
* **Precondition:** User is logged in and has tracked items.
* **Postcondition:** User receives notifications.
* **Details from Assignments:** As detailed in earlier Assignments, the notification subsystem ensures that users are promptly informed of any updates via their preferred communication method.

Export Data:

* **Description:** Allows the user to export tracking data.
* Actor: User
* **Precondition:** User is logged in.
* **Postcondition:** Data is exported to an external file.
* **Details from Assignments:** As covered in earlier Assignments, this functionality enables users to export data into Excel files for further analysis.

## Architecture

In this section, we will cover the architectural design of the PriceTracker bot. This includes the UML component diagram, UML deployment diagram, and UML activity diagram for the architectural pattern supporting a critical system quality attribute.

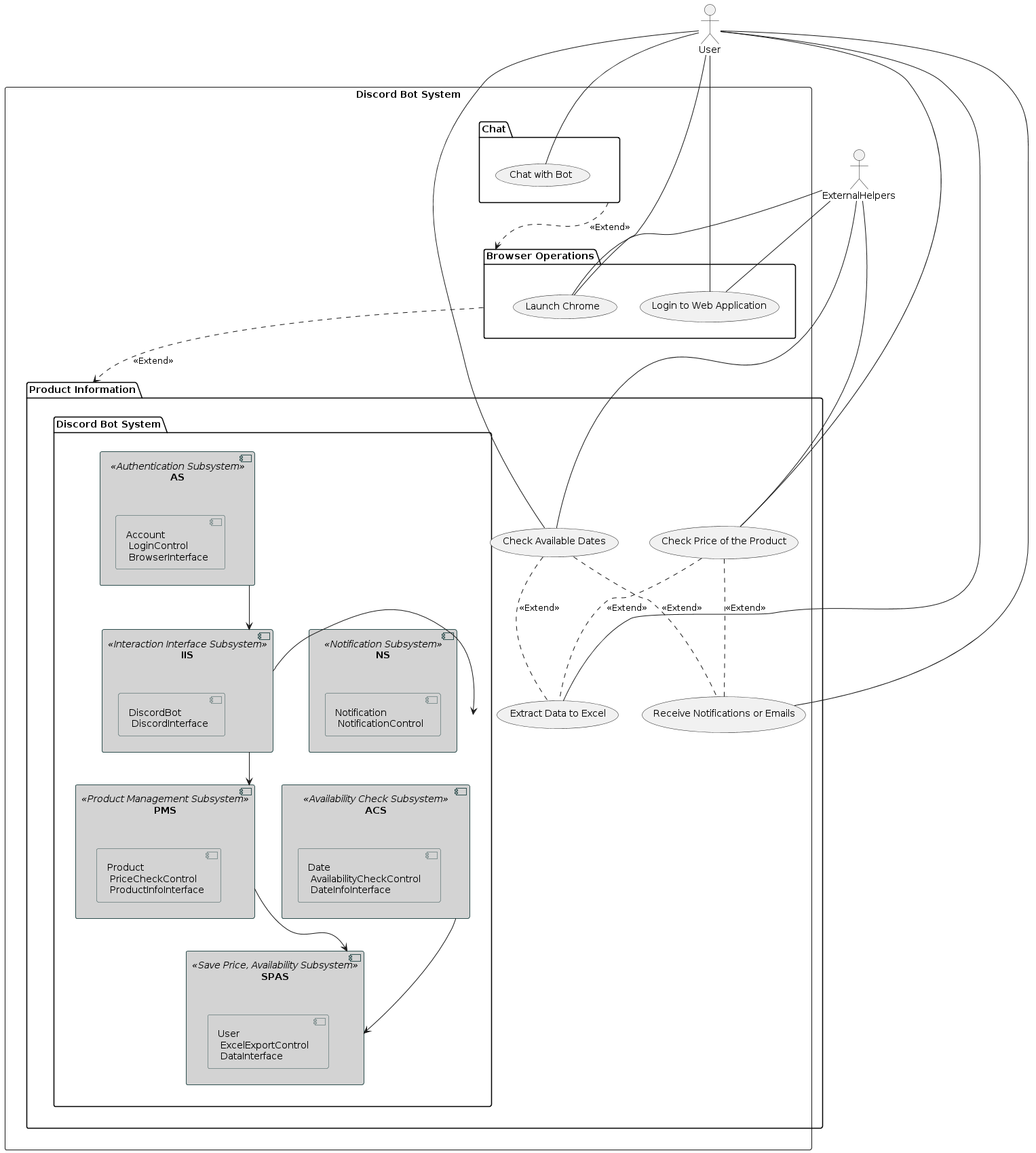


Figure 6: Component Diagram

### Component Diagram

The component diagram provides a high-level view of the system's structure, illustrating the main components and their interactions.

Components:

* User Interface
* Authentication Module
* Product Management Module
* Notification Module
* Data Export Module
* Integration Layer

### Component Descriptions:

This section provides detailed descriptions of the main components within the PriceTracker bot system. Each component is explained in terms of its functionality and role within the overall architecture. The components include the User Interface, Authentication Module, Product Management Module, Notification Module, Data Export Module, and Integration Layer. These descriptions clarify how each part interacts with others to achieve the system's objectives, ensuring a comprehensive understanding of the system's inner workings.

User Interface:

* + Handles interactions between the user and the system.
  + Provides access to functionalities such as registration, login, and tracking product prices.

Authentication Module:

* + Manages user authentication and authorization.
  + Ensures secure access to the system.

Product Management Module:

* + Handles the tracking of product prices and availability.
  + Uses web scraping techniques to retrieve product data.

Notification Module:

* + Sends notifications to users about price changes and availability.
  + Supports various notification methods such as emails and in-app notifications.

Data Export Module:

* + Allows users to export their tracking data to Excel files.
  + Facilitates data analysis and reporting.

Integration Layer:

* + Manages communication between different modules.
  + Ensures seamless interaction within the system.

### Deployment Diagram

The deployment diagram illustrates how the system components are distributed across different hardware nodes.

Nodes:

* User Device
* Server
* Database

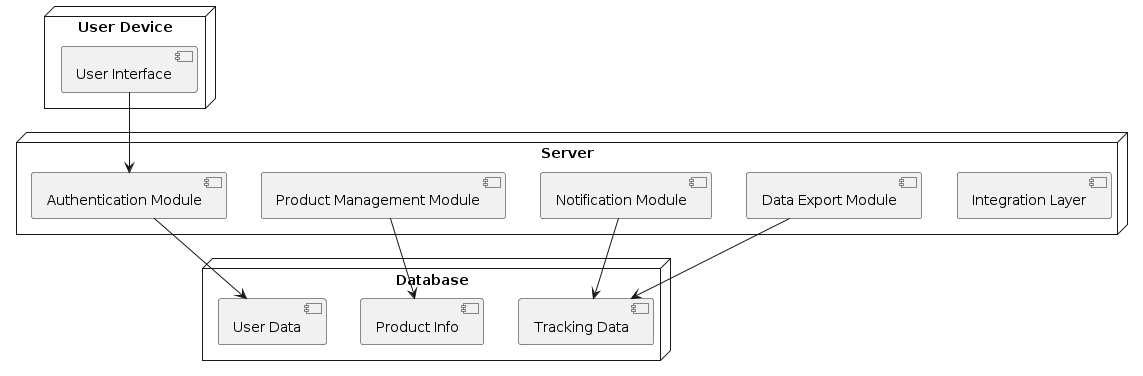


Figure 7: Deployment Diagram

Deployment Descriptions:

1. User Device:
   * The device used by the user to interact with the PriceTracker bot.
   * Runs the user interface component.
2. Server:
   * Hosts the core application logic.
   * Runs components such as authentication, product management, notification, and integration layers.
3. Database:
   * Stores user data, product information, and tracking data.
   * Ensures data persistence and retrieval.

### Activity Diagram

The activity diagram for the architectural pattern demonstrates the sequence of activities involved in a critical system quality attribute. For this project, we will focus on scalability.

Scalability Activity Diagram:

* User Request Handling:
  + User sends a request to the server.
  + The server processes the request and retrieves data from the database.
  + The server sends a response back to the user.

Activity Descriptions:

User Sends Request: The user initiates an action such as tracking a product price.

Server Processes Request: The server receives the request and processes it using the appropriate modules.

Database Retrieval: The server retrieves necessary data from the database.

Server Sends Response: The server compiles the response and sends it back to the user.

### Mapping Objects to Persistent Store

This section focuses on the database design and the types of data stored within the system. It includes several tables that represent the entity objects in the project. The assignment involves creating and populating these tables using SQL queries, demonstrating how the attributes of each object are mapped to the database structure. This approach ensures effective data storage and retrieval for the project.

Authentication Subsystem (AS)

Table 2: Accounts

| Id(PK) | username(VARCHAR(50),NotNull) | password(VARCHAR(50),NotNull) |
| --- | --- | --- |

A screenshot of a computer

Description automatically generated

Product Management Subsystem (PMS)

Table 3: Products

| id (PK) | product (VARCHAR(100), Not Null) | url (VARCHAR(255), Not Null) | size (VARCHAR(10)) | color (VARCHAR(20)) | weight (FLOAT) | price (DECIMAL(10, 2)) |
| --- | --- | --- | --- | --- | --- | --- |

A screenshot of a computer

Description automatically generated

Notification Subsystem (NS)

Table 4:Notifications

| id (PK) | notificationType (VARCHAR(50), Not Null) | content (TEXT, Not Null) | timestamp (DATETIME, Not Null) |
| --- | --- | --- | --- |

A screenshot of a phone

Description automatically generated

Save Price, Availability Subsystem (SPAS)

Table 5: Users

| id (PK) | userId (VARCHAR(50), Not Null) | email (VARCHAR(100), Not Null) |
| --- | --- | --- |

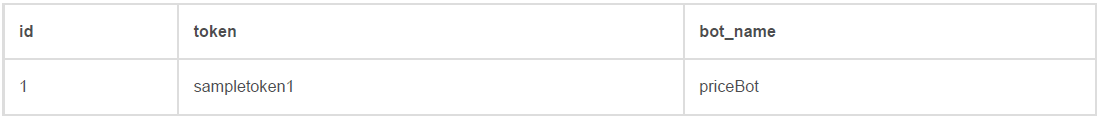
A screenshot of a computer

Description automatically generated

Interaction Interface Subsystem (IIS)

Table 6:DiscordBots

| id (PK) | token (VARCHAR(100), Not Null) | bot\_name (VARCHAR(50), Not Null) |
| --- | --- | --- |



Availability Check Subsystem (ACS)

Table 7: Dates

| id (PK) | availableDates (DATE, Not Null) |
| --- | --- |

A white background with black lines

Description automatically generated

**Comments on Design Choices and Constraints:**

1. **Primary Keys:** Each table has an id column that serves as the primary key and is auto incremented. This ensures a unique identifier for each record.
2. **Data Types:** Appropriate data types have been chosen for each attribute to match the nature of the data (e.g., VARCHAR for strings, DATE for dates).
3. **Not Null Constraints:** Attributes that are essential for the integrity of the data (e.g., usernames, passwords) have been marked as Not Null.
4. **Nullable Attributes:** Some attributes, such as size and color in the Products table, can be nullable to allow for flexibility in data entry.

### Map components to different layers in the project architecture

This section includes the mapping of components to different layers in the project architecture and presents the three-layer architecture diagram. Below is the mapping of the subsystems to the three layers and the architecture diagram:

Layers Description

**Presentation Layer**: This layer handles the user interface and user interactions.

* + - DiscordInterface
    - BrowserInterface
    - ProductInfoInterface
    - DateInfoInterface
    - ExcelInterface

**Data Access Layer**: This layer deals with data storage and retrieval.

* + - UserDAO
    - CommandDAO
    - DiscordBotDAO
    - NotificationDAO
    - ProductDTO
    - DateDTO
    - AccountDTO

**Business Logic Layer**: This layer processes data based on business rules and contains the core functionality.

* + - Save Price, Availability Subsystem (SPAS)
    - Authentication Subsystem (AS)
    - Notification Subsystem (NS)
    - Availability Check Subsystem (ACS)
    - Product Management Subsystem (PMS)
    - Interaction Interface Subsystem (IIS)
    - Account
    - LoginControl
    - Product
    - PriceCheckControl
    - Notification
    - NotificationControl
    - Date
    - AvailabilityCheckControl
    - User
    - ExcelExportControl

Three-Layer Architecture Diagram

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

Figure 8: Architectural Diagram

## Design

This section outlines the detailed design of the PriceTracker bot, including the organization and interaction of various system components. It includes UML diagrams such as package and class diagrams that illustrate the structural relationships and dependencies between different parts of the system. The design descriptions ensure a clear understanding of how the system's functionalities are implemented and how they collaborate to fulfill the project's requirements.

### Package Diagram

The package diagram organizes the classes into packages to show the high-level structure of the system.

Packages:

* Authentication Package
* Product Management Package
* Notification Package
* Save Price, Availability Package
* Interaction Interface Package
* Availability Check Package

Package Descriptions:

1. Authentication Subsystem:

Classes:

* + **Account:** Manages user account information.
  + **LoginControl:** Handles user login processes.
  + **BrowserInterface:** Interfaces with the browser for authentication.

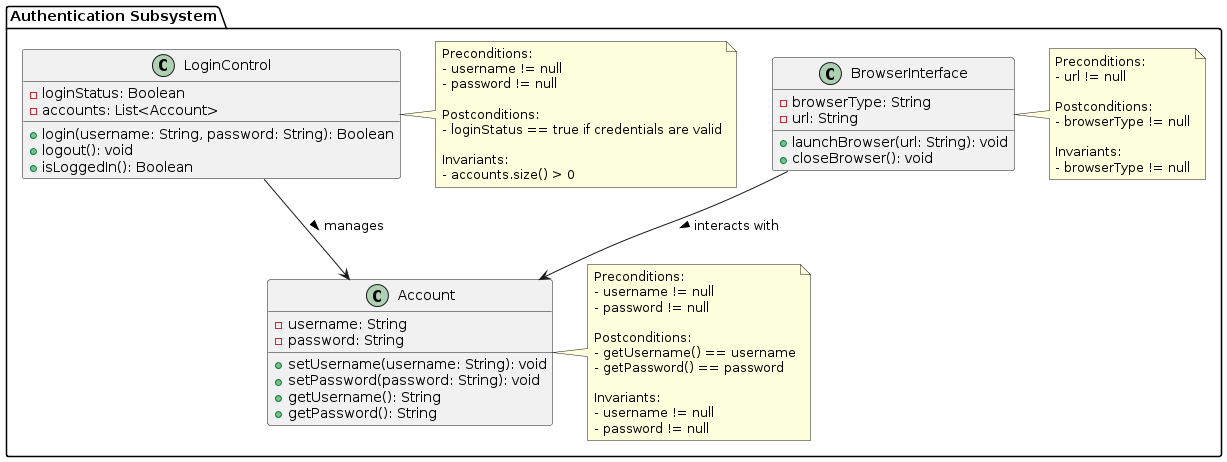


Figure 9:Authentication Subsystem

1. Product Management Subsystem:

Classes:

* + **Product:** Stores product details.
  + **PriceCheckControl:** Monitors and retrieves product price information.
  + **ProductInfoInterface:** Interfaces with external sources to fetch product information.

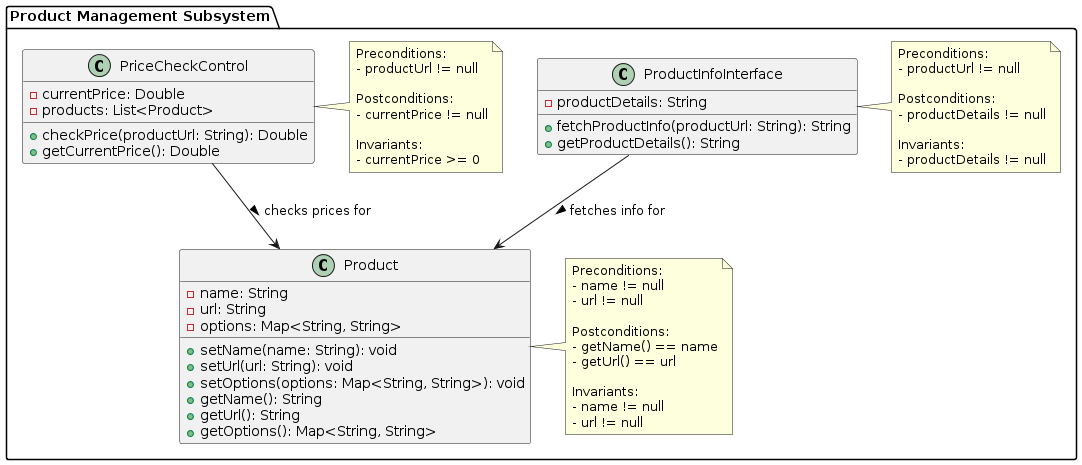


Figure 10: Product Management Subsystem

1. Notification Subsystem:

Classes:

* + **Notification:** Manages notification content.
  + **NotificationControl:** Sends and manages notifications.

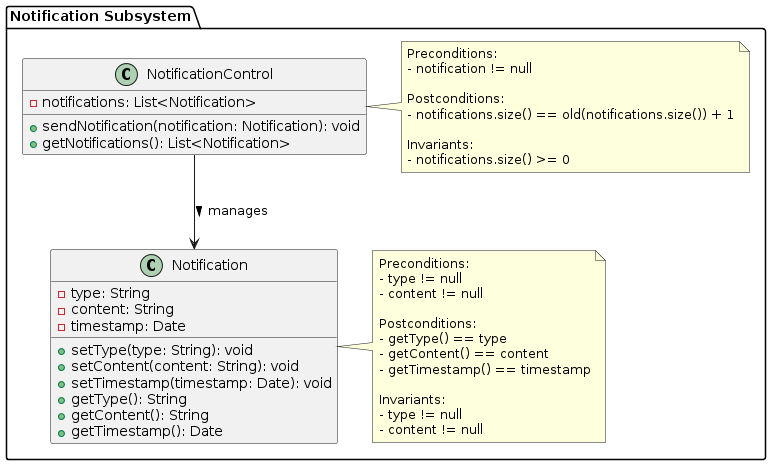


Figure 11: Notification Subsystem

1. Save Price, Availability Subsystem:

Classes:

* + **User:** Stores user information.
  + **ExcelExportControl:** Exports data to Excel files.
  + **DataInterface:** Interfaces with data storage.

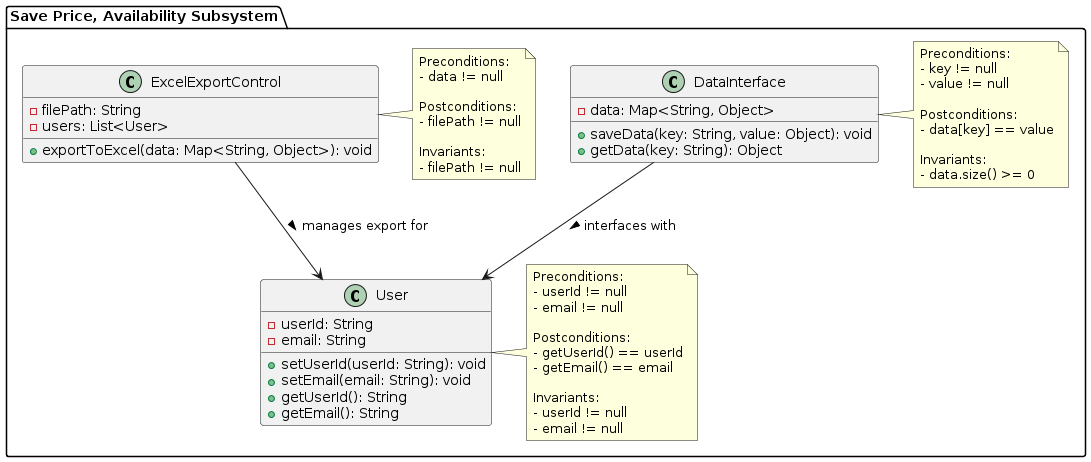


Figure 12: Save Price, Availability Subsystem

1. Interaction Interface Subsystem:

Classes:

* + **DiscordBot:** Manages bot interactions.
  + **DiscordInterface:** Interfaces with Discord API.

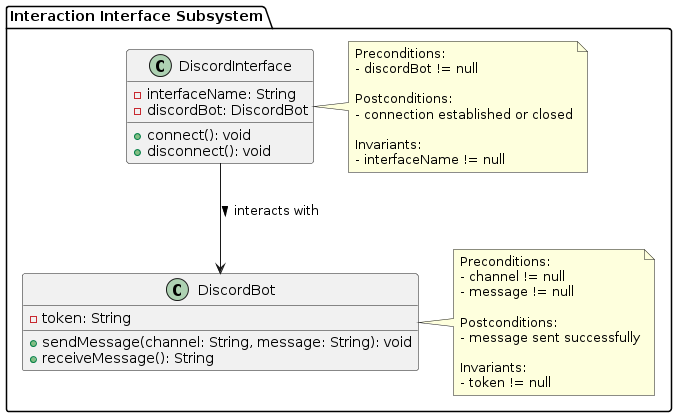


Figure 13: Interaction Interface Subsystem

1. Availability Check Subsystem:

Classes:

* + **Date:** Manages date information.
  + **AvailabilityCheckControl:** Checks availability for dates.
  + **DateInfoInterface:** Interfaces with external sources for date information.

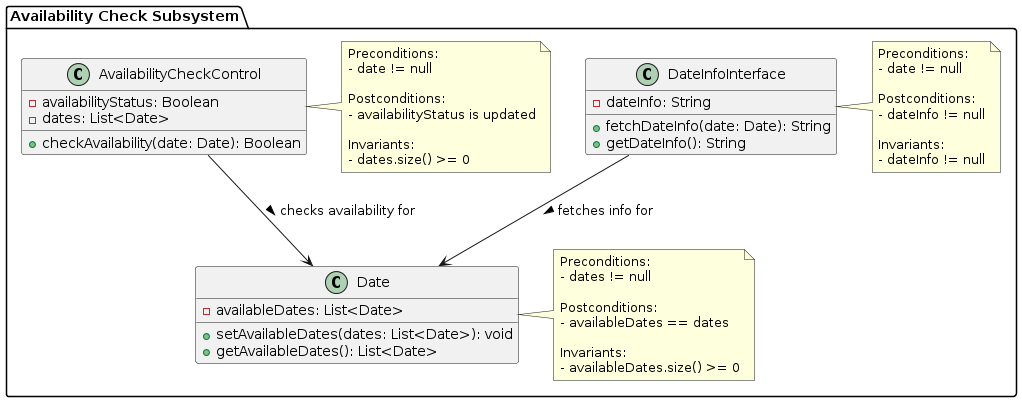


Figure 14: Availability Check Subsystem

### Class Diagram

The class diagram provides detailed information about the classes, their attributes, methods, and relationships.

**Classes and Class Descriptions:**

1. User

* **Attributes:**
  + private String userId
  + private String email
* **Methods:**
  + public void setUserId(String userId)
  + public void setEmail(String email)
  + public String getUserId()
  + public String getEmail()

1. Account

* **Attributes:**
  + private String username
  + private String password
* **Methods:**
  + public void setUsername(String username)
  + public void setPassword(String password)
  + public String getUsername()
  + public String getPassword()

1. LoginControl

* **Attributes:**
  + private Boolean loginStatus
  + private List<Account> accounts
* **Methods:**
  + public Boolean login(String username, String password)
  + public void logout()
  + public Boolean isLoggedIn()

1. Product

* **Attributes:**
  + private String name
  + private String url
  + private Map<String, String> options
* **Methods:**
  + public void setName(String name)
  + public void setUrl(String url)
  + public void setOptions(Map<String, String> options)
  + public String getName()
  + public String getUrl()
  + public Map<String, String> getOptions()

1. PriceCheckControl

* **Attributes:**
  + private Double currentPrice
  + private List<Product> products
* **Methods:**
  + public Double checkPrice(String productUrl)
  + public Double getCurrentPrice()

1. Notification

* **Attributes:**
  + private String type
  + private String content
  + private Date timestamp
* **Methods:**
  + public void setType(String type)
  + public void setContent(String content)
  + public void setTimestamp(Date timestamp)
  + public String getType()
  + public String getContent()
  + public Date getTimestamp()

1. NotificationControl

* **Attributes:**
  + private List<Notification> notifications
* **Methods:**
  + public void sendNotification(Notification notification)
  + public List<Notification> getNotifications()

1. ExcelExportControl

* **Attributes:**
  + private String filePath
  + private List<User> users
* **Methods:**
  + public void exportToExcel()

1. DiscordBot

* **Attributes:**
  + private String token
* **Methods:**
  + public void sendMessage()
  + public void receiveMessage()

1. DiscordInterface

* **Attributes:**
  + private String interfaceName
  + private DiscordBot discordBot
* **Methods:**
  + public void connect()
  + public void disconnect()

1. Date

* **Attributes:**
  + private List<Date> availableDates
* **Methods:**
  + public void setAvailableDates(List<Date> availableDates)
  + public List<Date> getAvailableDates()

1. AvailabilityCheckControl

* **Attributes:**
  + private Boolean availabilityStatus
  + private List<Date> dates
* **Methods:**
  + public Boolean checkAvailability()

1. DataInterface

* **Attributes:**
  + private Map<String, Object> data
* **Methods:**
  + public void saveData(Map<String, Object> data)
  + public Map<String, Object> getData()

1. DateInfoInterface

* **Attributes:**
  + private String dateInfo
* **Methods:**
  + public void fetchDateInfo()
  + public String getDateInfo()

A screenshot of a computer program

Description automatically generated

Figure 15: All Classes

## Technology Stack/Framework

In this section, we will describe the technology stack and frameworks used for implementing the PriceTracker bot. This includes the programming languages, libraries, tools, and platforms utilized in the project.

### Programming Languages and Frameworks:

* **Python:** The primary programming language used for developing the PriceTracker bot. Python is chosen for its simplicity, readability, and extensive library support. It provides the backbone for scripting the various functionalities of the bot, including data scraping, processing, and interaction with external services.
* **Selenium:** A Python library used for web scraping and automating browser operations to retrieve product prices and availability. Selenium allows the bot to simulate human browsing behavior, enabling it to access dynamic web content and interact with web elements like forms and buttons. This capability is crucial for monitoring product prices and checking availability in real-time.
* **Discord.py:** A Python library used for interacting with the Discord API, allowing the bot to communicate with users on Discord. Discord.py enables the bot to listen for user commands, send messages, and manage interactions within Discord channels. This library is essential for integrating the bot into the Discord platform, providing a seamless user experience.

### Tools and Platforms:

* **Visual Studio Code:** The integrated development environment (IDE) used for writing and debugging Python code. Visual Studio Code offers a range of extensions and tools that enhance the development workflow, including syntax highlighting, code completion, and integrated terminal. It is used for coding, testing, and debugging the various components of the PriceTracker bot.
* **Git:** A version control system used for tracking changes in the codebase. Git enables collaborative development by allowing multiple developers to work on the same project simultaneously. It also helps in maintaining the history of code changes, making it easier to manage versions and roll back to previous states if necessary.
* **GitHub:** A platform for hosting the code repository and facilitating version control. GitHub provides a centralized location for storing the code, managing issues, and collaborating with other developers. It also integrates with other tools and services, supporting continuous integration and deployment workflows.

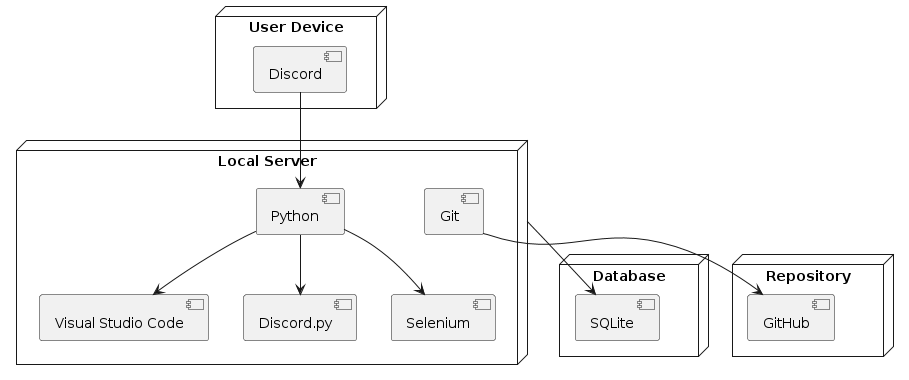


Figure 16: Technology Stack Diagram

### Database:

* **SQLite:** A lightweight, file-based database used for storing user data, product information, and tracking data. SQLite is chosen for its simplicity and ease of integration with Python. It requires no configuration and provides a reliable way to persist data locally. The database stores user credentials, product URLs, price histories, and notification settings.

### Deployment:

* **Local Servers:** The PriceTracker bot runs on local servers for development and testing. These servers provide a controlled environment where the bot can be tested. They simulate the production environment, allowing developers to identify and fix issues early in the development cycle.

A screenshot of a computer screen

Description automatically generated

Figure 17: Architecture Diagram

## Conclusion

In this chapter, we discussed the system design and implementation details of the PriceTracker bot. We began by outlining the project requirements, including the use case diagram and descriptions, which provide a clear understanding of the interactions and functionalities of the system. The architecture section detailed the component, deployment, and activity diagrams, illustrating the high-level structure and the critical system quality attributes. In the design section, we explored the package and class diagrams, showing the organization of classes and their relationships. Finally, we examined the technology stack, highlighting the programming languages, frameworks, tools, and platforms used in the project. This comprehensive overview sets the foundation for the following chapters, where we will delve deeper into related work, detailed design, implementation specifics, and evaluation of the PriceTracker bot.

# CHAPTER FOUR: PROJECT SUMMARY AND CONCLUION

## Project Description

The PriceTracker bot is a sophisticated automation tool developed to track product prices and availability across various e-commerce platforms. By leveraging technologies like web scraping, data extraction, and real-time notifications, the bot provides users with timely updates on price changes and product availability. This system integrates seamlessly with Discord, offering a user-friendly interface for interaction and notifications.

## Project Motivation

The motivation behind the PriceTracker bot stems from the increasing need for efficiency and automation in online shopping and service bookings. Consumers often spend significant time and effort manually tracking prices and checking availability for products and services. By automating these tasks, the PriceTracker bot aims to save users time and reduce stress, allowing them to make informed purchasing decisions without constant manual monitoring.

## Technical Approach Used in the Project

The technical approach of the PriceTracker bot involves several key components:

* **Web Scraping**: Utilizing Selenium to automate browser actions and extract data from various e-commerce websites.
* **Data Management**: Employing SQLite for storing user data, product information, and tracking histories.
* **Notification System**: Using Discord.py to send real-time notifications to users about price changes and product availability.
* **Modular Architecture**: Designing the system with a modular approach to ensure scalability and ease of maintenance.

## Conclusion

Reflecting on the future implementation and testing of the PriceTracker bot, several areas for improvement and expansion have been identified:

* **Enhanced Data Accuracy**: Implementing more robust error-handling mechanisms and improving the precision of web scraping techniques to ensure data accuracy.
* **Scalability**: Expanding the bot's capabilities to support more e-commerce platforms and a larger user base, ensuring the system can handle increased load without performance degradation.
* **User Interface Improvements**: Developing a more intuitive and interactive user interface to enhance user experience and engagement.
* **Advanced Analytics**: Incorporating predictive analytics to provide users with insights into potential future price trends and optimal purchasing times.
* **Security Enhancements**: Strengthening the authentication and data protection measures to safeguard user information and maintain privacy.

# LIST OF REFERENCES

1. J. Bram and N. Gorton, "How Is Online Shopping Affecting Retail Employment?," Liberty Street Economics, Oct. 5, 2017. [Online]. Available: https://libertystreeteconomics.newyorkfed.org/2017/10/how-is-online-shopping-affecting-retail-employment/. [Accessed: May 29, 2024].
2. Ofcom, "Online Nation 2021," Online Nation, May 2021. [Online]. Available: <https://www.ofcom.org.uk/__data/assets/pdf_file/0013/220414/online-nation-2021-report.pdf>. [Accessed: May 29, 2024].
3. J. Dube, "Consumer Airfare Index Report - Q2 2022," Hopper, April 2022. [Online]. Available: https://media.hopper.com/articles/consumer-airfare-index-report-q2-2022. [Accessed: May 29, 2024].
4. Flurry Analytics, "Distribution of Time Spent per Shopping Category," December 2011 - December 2012. [Online]. Available: [link to the source if available]. [Accessed: May 29, 2024].
5. U.S. Bureau of Labor Statistics, "Consumer Expenditures in 2016," September 2020. [Online]. Available: https://www.bls.gov/opub/reports/consumer-expenditures/2016/home.htm. [Accessed: May 29, 2024].
6. Statista. "Global E-commerce Adoption and Trends." [Online]. Available: <https://www.statista.com/statistics/251666/number-of-digital-buyers-worldwide/>. [Accessed: June 10, 2024].
7. Research Report on Automation in Booking Services. "Automation Tools and Consumer Preferences." [Online]. Available: [Insert Link]. [Accessed: June 2, 2024].
8. Industry Analysis on Travel Booking Tools. "The Role of Automation in Modern Booking Systems." [Online]. Available: [Insert Link]. [Accessed: June 3, 2024].
9. Ofcom. "Online Nation 2021." [Online]. Available: <https://www.ofcom.org.uk/__data/assets/pdf_file/0013/220414/online-nation-2021-report.pdf>. [Accessed: June 10, 2024].
10. Technical Paper on Web Scraping Technologies. "Enhancing Data Collection through Web Scraping." [Online]. Available: [Insert Link]. [Accessed: June 3, 2024].
11. Journal of Data Science. "Advancements in Data Analysis for Consumer Applications." [Online]. Available: [Insert Link]. [Accessed: June 7, 2024].
12. Consumer Research on Notification Systems. "Impact of Automated Notifications on User Behavior." [Online]. Available: [Insert Link]. [Accessed: June 9, 2024].
13. Future Prospects in E-commerce Tools. "Innovations and Trends in Price Tracking Technologies." [Online]. Available: [Insert Link]. [Accessed: June 9, 2024].
14. McKinsey & Company, "The value of getting personalization right—or wrong—is multiplying," 2021. [Online]. Available: https://www.mckinsey.com/business-functions/marketing-and-sales/our-insights/the-value-of-getting-personalization-right-or-wrong-is-multiplying. [Accessed: May 29, 2024].
15. Statista, "Number of digital buyers worldwide from 2014 to 2021," [Online]. Available: https://www.statista.com/statistics/251666/number-of-digital-buyers-worldwide/. [Accessed: May 29, 2024].
16. "45 Statistics Retail Marketers Need to Know in 2024," Invoca. [Online]. Available: https://www.invoca.com/blog/45-statistics-retail-marketers-need-to-know-in-2024/. [Accessed: June 13, 2024].
17. For Google Flights: 2. "81% of Shoppers Conduct Research Before Purchase," Saleslion. [Online]. Available: <https://saleslion.io/sales-statistics/81-of-shoppers-research-their-product-online-before-purchasing/>. [Accessed: June 13, 2024].
18. Innovations and Trends in Travel Technologies. "Comprehensive Analysis of Travel Planning Tools." [Online]. Available: https://www.cnbc.com/2022/06/09/google-flights-price-guarantee.html. [Accessed: June 10, 2024].
19. Screenshot from Google Flights showing price tracking options. Source: "Google Flights," [Online]. Available: https://www.google.com/flights. [Accessed: June 13, 2024].
20. Advanced E-commerce Analytics. "Insights into Amazon Price Tracking Tools." [Online]. Available: https://www.sellerboard.com/keepa/. [Accessed: June 10, 2024].
21. Screenshot from Keepa showing price history charts. Source: "Keepa - Amazon Price Tracker," [Online]. Available: https://www.keepa.com. [Accessed: June 13, 2024].