# TABLE OF CONTENTS

[TABLE OF CONTENTS i](#_Toc178950115)

[CHAPTER THREE: PROJECT ORGANIZATION/STRUCTURE 3](#_Toc178950116)

[1.1 Project Requirements 3](#_Toc178950118)

[1.1.1 Use Case Diagram: 3](#_Toc178950119)

[1.1.2 Use Case Descriptions: 4](#_Toc178950120)

[1.2 Architecture 6](#_Toc178950121)

[1.2.1 Component Diagram 7](#_Toc178950122)

[1.2.2 Component Descriptions: 8](#_Toc178950123)

[1.2.3 Deployment Diagram 9](#_Toc178950124)

[1.2.4 Activity Diagram 10](#_Toc178950125)

[1.2.5 Mapping Objects to Persistent Store 11](#_Toc178950126)

[1.2.6 Map components to different layers in the project architecture 13](#_Toc178950127)

[1.3 Design 16](#_Toc178950128)

[1.3.1 Package Diagram 17](#_Toc178950129)

[1.3.2 Class Diagram 21](#_Toc178950130)

[1.4 Technology Stack/Framework 26](#_Toc178950131)

[1.4.1 Programming Languages and Frameworks: 27](#_Toc178950132)

[1.4.2 Tools and Platforms: 27](#_Toc178950133)

[1.4.3 Database: 28](#_Toc178950134)

[1.4.4 Deployment: 29](#_Toc178950135)

[1.5 Conclusion 29](#_Toc178950136)

# CHAPTER THREE: PROJECT ORGANIZATION/STRUCTURE

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

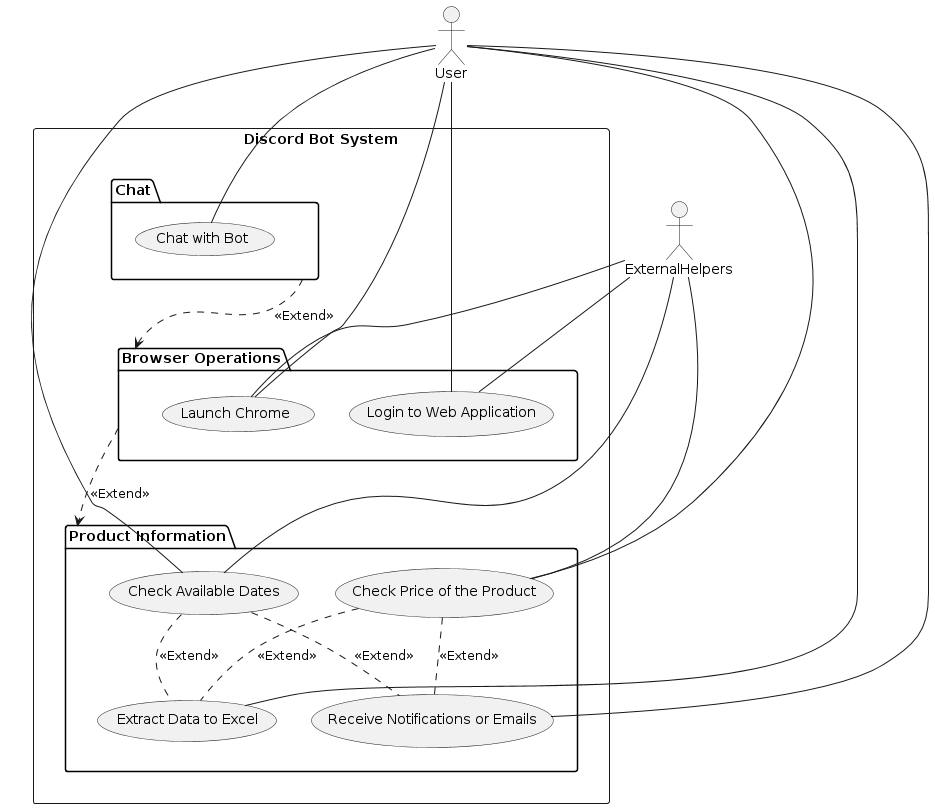


Figure 5: Use Case Diagram

### 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.

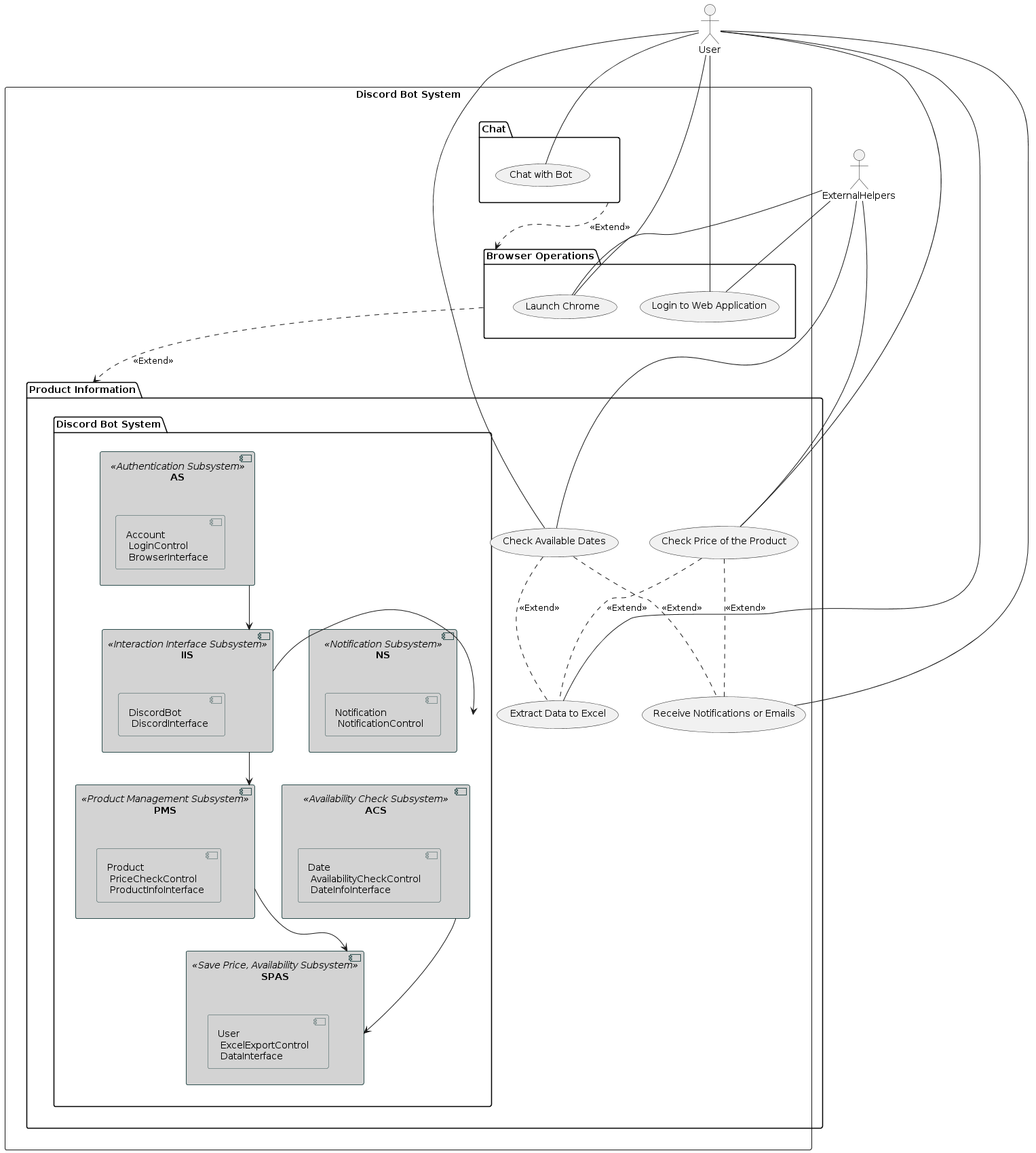


Figure 6: Component Diagram

### 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

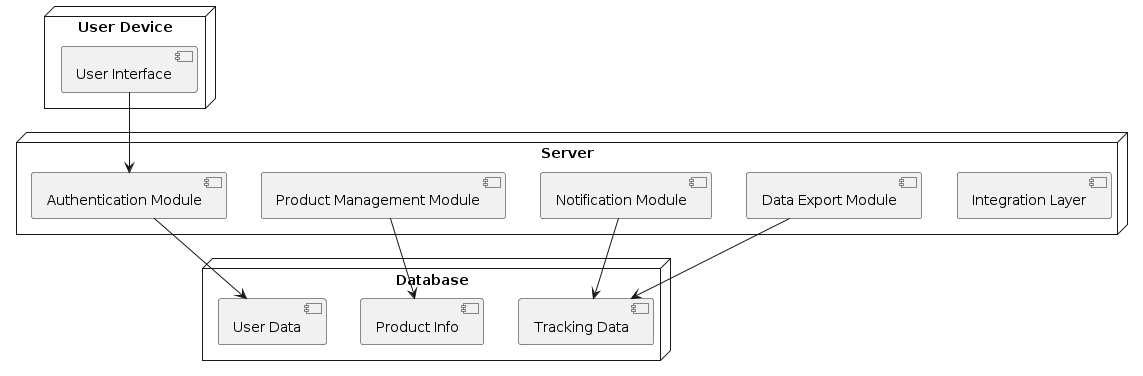


Figure 7: Deployment Diagram

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) |
| --- | --- | --- |

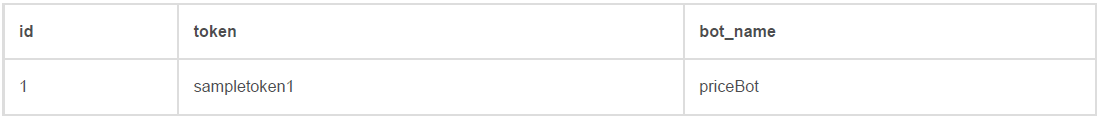
A screenshot of a computer

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.

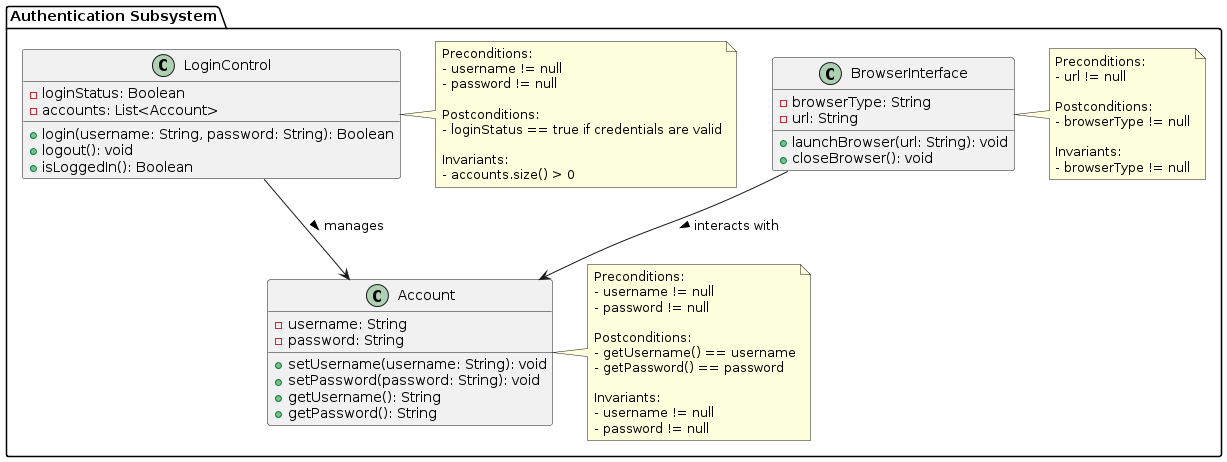


Figure 9:Authentication Subsystem

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.

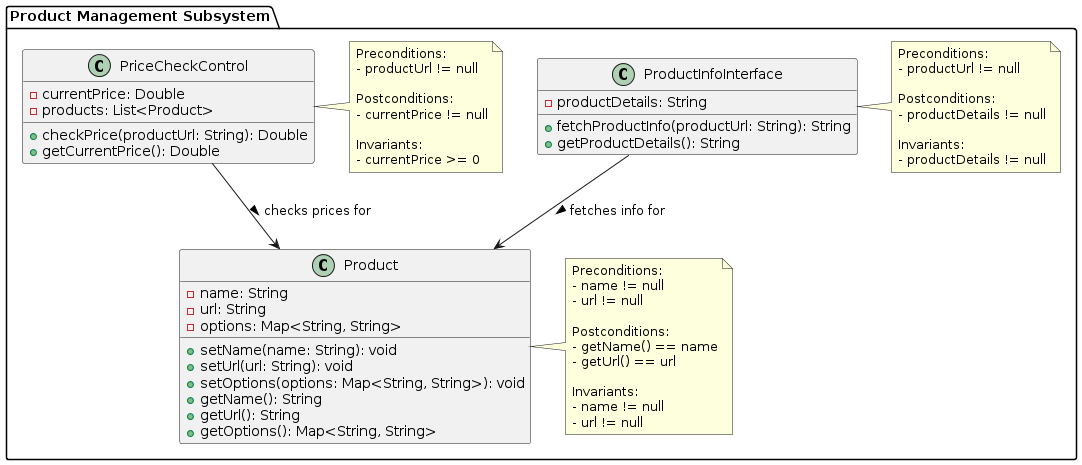


Figure 10: Product Management Subsystem

1. Notification Subsystem:

Classes:

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

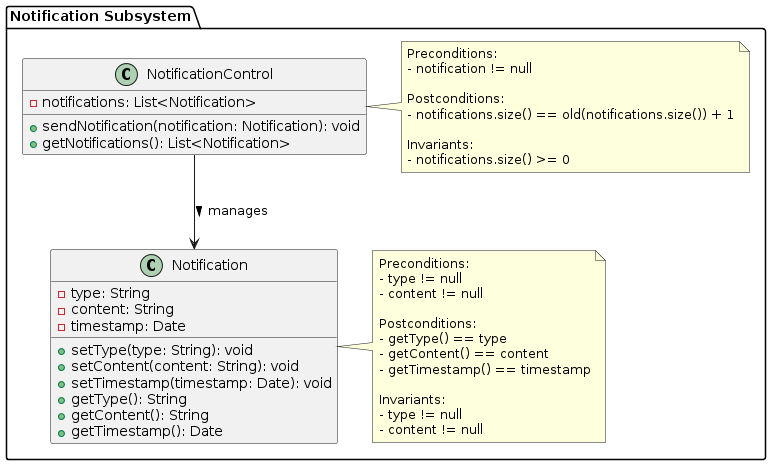


Figure 11: Notification Subsystem

1. Save Price, Availability Subsystem:

Classes:

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

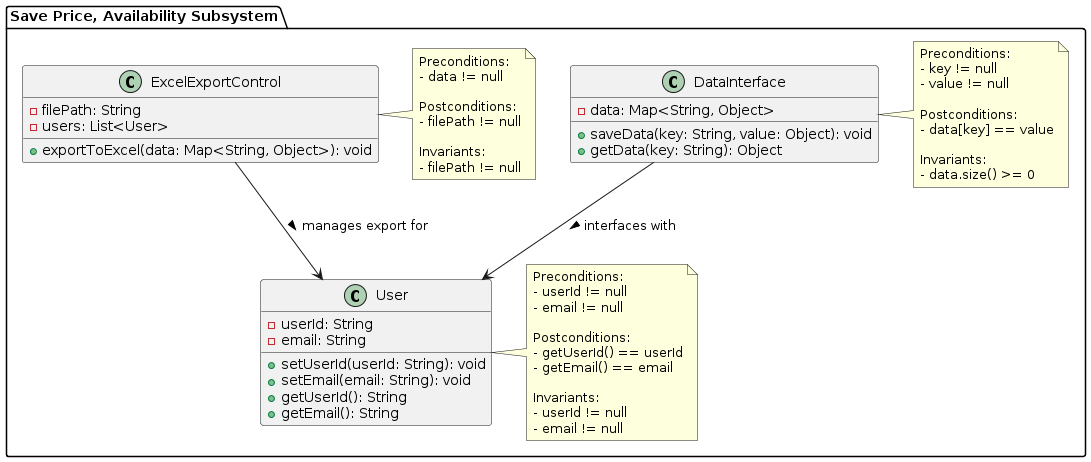


Figure 12: Save Price, Availability Subsystem

1. Interaction Interface Subsystem:

Classes:

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

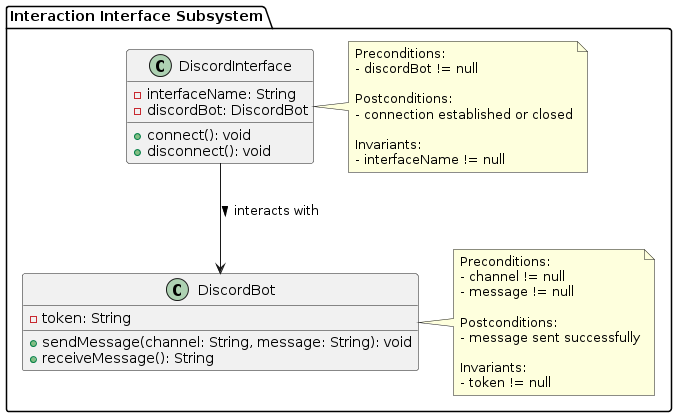


Figure 13: Interaction Interface Subsystem

1. Availability Check Subsystem:

Classes:

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

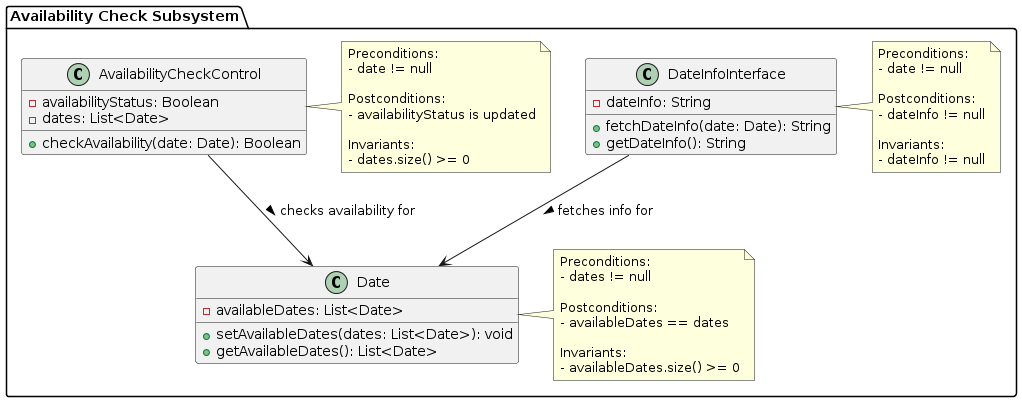


Figure 14: Availability Check Subsystem

### 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.

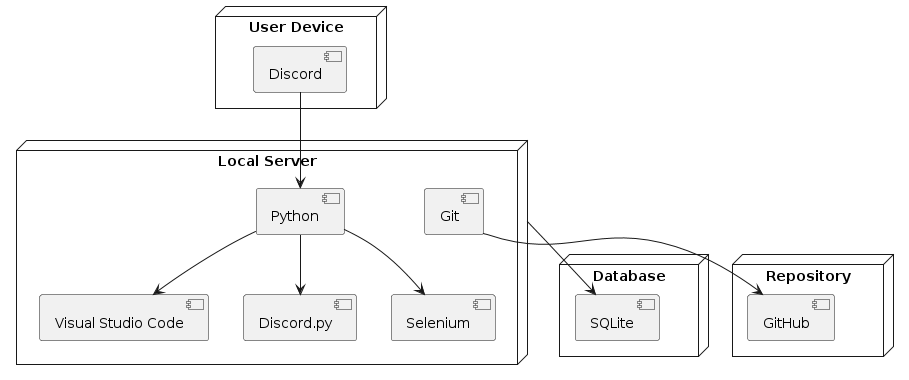


Figure 16: Technology Stack Diagram

### 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.