

# PRICE WATCH

your personalized price tracking hub

*A Project report submitted in partial fulfilment of the requirements for the award  
of the Degree of **Bachelors of Technology***

*In*

***Computer Science and Engineering (CSE)***

*By*

**Pingili Shivani Reddy (21011A0536)**

**Power Vijaya (21011A0538)**

**Talari Varun (22015A0515)**

*Under the guidance of*

**Dr. O. B. V. Ramanaiah**

**Senior Professor**



**Department of Computer Science and Engineering,  
JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD  
University College of Engineering, Science & Technology Hyderabad  
Kukatpally, Hyderabad - 500 085.**

**Department of Computer Science and Engineering,  
JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD  
University College of Engineering, Science & Technology Hyderabad  
Kukatpally, Hyderabad - 500 085.**



## **DECLARATION BY THE CANDIDATES**

**We, Pingili Shivani Reddy(21011A0536), Power Vijaya(21011A0538), Talari Varun(22015A0515)** hereby declare that the mini-project report entitled **“PRICE WATCH –Your Personalized Price Tracking Hub”**, developed by our group under the guidance of **Dr. O.B.V. Ramanaiah**, is submitted in partial fulfilment of the requirements for the award of the degree of *Bachelors of Technology in Computer Science and Engineering*. This is a record of bona fide work carried out by our group and the results embodied in this project have not been reproduced/copied from any source.

The results embodied in this project report have not been submitted to any other University or Institute for the award of any other degree or diploma.

**Pingili Shivani Reddy (21011A0536)**

**Power Vijaya (21011A0538)**

**Talari Varun (22015A0515)**

**Department of Computer Science and Engineering,  
JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD  
University College of Engineering, Science & Technology Hyderabad  
Kukatpally, Hyderabad - 500 085.**



## **CERTIFICATE BY THE SUPERVISOR**

This is to certify that the project report entitled “**PRICE WATCH –Your Personalized Price Tracking Hub**”, being submitted by **Pingili Shivani Reddy(21011A0536), Power Vijaya(21011A0538), Talari Varun(22015A0515)**, in partial fulfilment of the requirements for the award of the degree of *Bachelor of Technology in Computer Science and Engineering*, is a record of bona fide work carried out by them. The results embodied in this project report have not been submitted to any other University or Institute for the award of any other degree or diploma.

**Dr. O. B. V. Ramanaiah,**  
Senior Professor

Date:

**Department of Computer Science and Engineering,  
JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD  
University College of Engineering, Science & Technology Hyderabad  
Kukatpally, Hyderabad - 500 085.**



## **CERTIFICATE BY THE HEAD OF DEPARTMENT**

This is to certify that the project report entitled “**PRICE WATCH—Your Personalized Price Tracking Hub**”, being submitted by **Pingili Shivani Reddy(21011A0536)**, **Power Vijaya(21011A0538)**, **Talari Varun(22015A0515)**, in partial fulfilment of the requirements for the award of the degree of *Bachelor of Technology in Computer Science and Engineering* is a record of bona fide work carried out by them.

**Dr. K.P. Supreethi**  
Professor & Head of CSE

Date:

## **ACKNOWLEDGEMENT**

We wish to extend our sincere gratitude to our supervisor, **Dr. O.B.V. Ramanaiah**, Senior Professor of the Department, for being a driving force all through the way. This project would not be so smooth and so interesting without his encouragement.

We are extremely grateful to **Dr. K. P. Supreethi**, Professor & Head of CSE for her immense support and cooperation that contributed a lot in the completion of the task. We show gratitude to our beloved Principal **Dr. G.V. Narasimha Reddy**, Senior Professor, Principal & Director of R&D for providing necessary infrastructure and resources for the accomplishment of our project report at JNTUH University College of Engineering, Science and Technology, Hyderabad. We also thank all the staff members of Computer Science & Engineering department, JNTUH University College of Engineering, Hyderabad for their valuable support and generous advices.

Finally thanks to our parents, all our family members and friends for their continuous support and enthusiastic help.

**Pingili Shivani Reddy (21011A0536)**

**Power Vijaya (21011A0538)**

**Talari Varun (22015A0515)**

## ABSTRACT

We chose to develop this project to address the challenge of missing opportunities to save money, as constantly monitoring prices can be time-consuming and inefficient. Consumers face the challenge of tracking prices on e-commerce platforms to ensure they get the best deals. Prices on e-commerce platforms can fluctuate frequently due to sales, promotions, and stock levels. Manually checking prices across multiple websites is labour-intensive and inefficient. To overcome this, we have come up with a solution, that is “**PRICE WATCH**”. The Price Watch project is designed by leveraging web scraping techniques such as BeautifulSoup and Selenium, and also used the MERN stack for developing the website for users to interact, to monitor the prices of selected products across e-commerce platforms. The core objective is to provide users with real-time updates and alerts on price fluctuations, enabling them to make informed purchasing decisions."

## Table of Contents

ACKNOWLEDGEMENT .....	v
ABSTRACT .....	vi
INTRODUCTION .....	2
1.1 Motivation.....	2
1.2 Problem Definition .....	2
1.3 Software and Hardware Specification .....	3
1.4 Report Organization.....	3
LITERATURE .....	5
2.1 Price Scraping.....	5
Methodologies.....	6
3. DESIGN AND ANALYSIS.....	11
3.1 Unified Modelling Language (UML) .....	11
3.2 User Documentation .....	11
3.3 Assumptions and Dependencies .....	11
3.4 Use Case Diagram .....	11
3.5 Interaction Diagrams.....	13
3.5.1 Sequence Diagrams:.....	13
3.5.2 Collaboration Diagrams.....	14
3.6 Interaction Diagrams for PRICE WATCH .....	15
3.6.1 Signup.....	15
3.6.2 Login .....	16
3.6.3 Entering Product Details .....	18
3.6.4 Products in Watchlist .....	20
3.6.5 View product.....	21

3.6.6 Logging out.....	22
3.6.7 Deleting Product .....	23
4. INSTALLATIONS .....	26
4.1 Installation of Visual Studio Code .....	26
Windows .....	26
4.2 Installation of MongoDB Compass.....	26
Windows and macOS.....	26
4.3 Installation of Python.....	26
Windows .....	27
4.4 Installation of NodeJS .....	27
Windows and macOS.....	27
4.5 Installation of ReactJS .....	27
1. Install Node.js .....	27
2. Create a React Application .....	28
4.6 Installation of Axios.....	28
4.7 Installation of React .....	28
4.8 Installation of React-Dom .....	29
4.9 Installation of React-Hot-Toast .....	29
4.10 Installation of React-Icons.....	29
4.11 Installation of React-Router-Dom .....	29
4.12 Installation of React-Script:.....	29
4.13 Installation of Web-Vitals.....	30
4.14 Installation of Express .....	30
4.15 Installation of Mongoose .....	30
4.16 Installation of Nodemon .....	30
4.17 Installation of Pymongo .....	31
4.18 Installation of Selenium .....	31
4.19 Installation of ChromeDriver .....	31
5. PROCEDURE .....	33
5.1 Working of Frontend.....	33
5.2 Working of Backend .....	35
MongoDB.js.....	35



5.3 Working of Price Tracking. ....	36
5.4 How To Run The Frontend, Backend, Python .....	36
6. Testing and Results .....	40
6.1 Initial Screen.....	40
6.2 Login Successful Toast.....	41
6.2 is displayed when user logins successfully.....	41
6.3 Unknown User Login Fail .....	42
shown in Figure 6.3 is displayed when user failed to login by entering wrong email or password. ....	42
6.4 Empty Signup .....	43
6.5 Signup Filling .....	44
6.6 Signup Successful Toast .....	45
6.7 Entered Product Details .....	46
6.8 Product Added to Watchlist.....	47
6.9 Existing URL Failed .....	48
6.10 Watchlist .....	49
6.11 No Products in Watchlist .....	50
6.12 After Delete Product .....	51
6.13 Registration mail .....	52
6.14 Price fall alert .....	53
7. Conclusion .....	56
7.1 Work Carried Out .....	56
7.2 Scope for Future Work.....	56
REFERENCES .....	57

## LIST OF TABLES

Table 6.1 Products Data .....	54
Table 6.2 Pre-Registration Mail.....	54
Table 6.3 Login Data .....	54

## LIST OF FIGURES

Figure 3.1 Use- case diagram.....	12
Figure 3.2 Signup Sequence diagram.....	16
Figure 3.3 Signup Collaboration diagram.....	16
Figure 3.4 login Sequence diagram.....	17
Figure 3.5 Login Collaboration diagram.....	18
Figure 3.6 Entering product details Sequence diagram.....	19
Figure 3.7 Entering product details Collaboration diagram.....	20
Figure 3.8 Watchlist Sequence diagram.....	20
Figure 3.9 Watchlist Collaboration diagram.....	21
Figure 3.10 View product Sequence diagram.....	21
Figure 3.11 View product Collaboration diagram.....	22
Figure 3.12 Log out Sequence diagram.....	23
Figure 3.13 Log out Collaboration diagram.....	23
Figure 3.14 Deleting product Sequence diagram.....	24
Figure 3.15 Deleting product Collaboration diagram.....	25
Figure 5.1 Commands of Frontend .....	37
Figure 5.2 Commands of Backend .....	38
Figure 5.3 Commands for python .....	38
Figure 6.1 Initial screen .....	40
Figure 6.2 Login Successful Toast.....	41
Figure 6.3 Unknown user login fail .....	42
Figure 6.4 Empty Sign Up.....	43
Figure 6.5 Sign up filling.....	44
Figure 6.6 Sign up successful toast.....	45
Figure 6.7 Entered Product Details.....	46
Figure 6.8 Product added to Watchlist.....	47

Figure 6.9 Existing URL failed.....	48
Figure 6.10 Watchlist.....	49
Figure 6.11 No products in Watchlist.....	50
Figure 6.12 After Deleted Product .....	51
Figure 6.13 Registration Mail .....	52
Figure 6.14 Price Decrease Mail .....	53

# **Chapter 1**

## **INTRODUCTION**

# INTRODUCTION

## 1.1 Motivation

The motivation behind developing the Price Watch project is to address the challenges faced by consumers in navigating the complex e-commerce landscape. By automating the process of monitoring the prices on e-commerce platforms, Price Watch empowers users to make informed purchasing decisions, saving them time effort and money. It tackles issues such as missed savings opportunities due to frequent price fluctuations by providing real-time alerts on price drops and discounts. Ultimately, Price Watch aims to enhance consumer satisfaction and loyalty by providing a streamlined, efficient tool for maximizing savings and making strategic purchasing decisions.

## 1.2 Problem Definition

Price Watch aims to simplify the shopping experience, help consumers make informed decisions, and ensure they never miss out on the best deals. By providing an intuitive and user-friendly platform, it allows users to effortlessly track price changes and make smarter purchasing choices. This not only saves time but also maximizes savings by highlighting the most cost-effective options available.

The system offers automated real-time price monitoring, comprehensive historical data analysis, and personalized alerts tailored to individual preferences. Users receive timely notifications about significant price drops and trends, enabling them to act swiftly on great deals. Additionally, the analysis of historical pricing drop

empowers consumers with insights into the best times to buy, further enhancing their shopping experience and financial efficiency.

### **1.3 Software and Hardware Specification**

#### **Software:**

Operating system : Windows 11

Emulator : MongoDB Atlas or MongoDB Compass, Selenium, Chrome Driver

Development Tools : Visual Studio Code, Node.js, MongoDB, Express.js, React, Git

#### **Hardware:**

System : Desktop or Laptop

Processor : 2 GHz dual-core processor or higher

RAM : 4 GB+

ROM : 128GB+

### **1.4 Report Organization**

**Chapter 2** Provides an overview of the Literature Survey.

**Chapter 3** Provides detailed information on UML design.

**Chapter 4** Gives the details on the installations of Modules.

**Chapter 5** Gives the procedure for implementation.

**Chapter 6** Deals with testing and results of the proposed system.

**Chapter 7** Gives conclusion deducted from the results followed by references.

## **Chapter-2**

# **LITERATURE**



# LITERATURE

## 2.1 Price Scraping

Price scraping, also known as **web scraping** for prices, involves the automated extraction of price information from online sources such as e-commerce websites, marketplaces, and other online retailers. This process uses specialized software tools, scripts, and algorithms to collect data efficiently and accurately, often in large volumes. The collected data is then used for various purposes including competitive analysis, market research, and dynamic pricing strategies.

The core mechanism of price scraping typically involves sending automated requests to a target website, mimicking human browsing behaviour. The scraper parses the website's HTML structure to locate and extract specific pieces of information, such as product names, prices, descriptions, and other relevant data. Popular tools and libraries for web scraping include **BeautifulSoup** and Scrapy for Python, and Puppeteer and Cheerio for JavaScript. These tools facilitate the handling of complex web pages, including those with dynamic content generated by JavaScript.

Price scraping has become an essential tool for businesses in the e-commerce sector. Companies use this data to monitor competitors' pricing strategies, allowing them to adjust their own prices to remain competitive. By regularly collecting price data, businesses can identify market trends, detect pricing patterns, and forecast future pricing movements. This information is crucial for making informed decisions on product pricing, promotions, and inventory management.

Despite its widespread use, price scraping raises several legal and ethical considerations. Legally, the practice of web scraping exists in a gray area, with its acceptability varying by jurisdiction. Issues arise primarily around the terms of service of the websites being scraped, intellectual property rights, and potential breaches of data protection regulations. Ethically, there are concerns regarding the impact of scraping on the functionality of websites, particularly if scraping activities are too frequent or intensive, potentially leading to server overloads and other operational issues.

Overall, price scraping is a powerful tool that offers significant benefits to businesses and consumers alike. For businesses, it provides a means to gain competitive insights and optimize pricing strategies. For consumers, it can lead to greater price transparency and more competitive pricing in the marketplace.

## **Applications**

- **Competitive Analysis:**

- Businesses use price scraping to monitor competitors' prices and adjust their own pricing strategies accordingly.

- **Market Research:**

- Researchers collect price data to analyse market trends and consumer behaviour.

- **Dynamic Pricing:**

- E-commerce platforms use scraped price data to implement dynamic pricing strategies, adjusting prices in real-time based on market conditions.

## **Methodologies**

### **Web Scraping Techniques:**

- **HTML Parsing:** Extracting data by parsing the HTML structure of web pages using libraries like BeautifulSoup (Python) or Cheerio (JavaScript). This involves identifying specific HTML elements and attributes that contain the desired information, such as product prices, and programmatically retrieving this data. HTML parsing is especially useful for sites without structured APIs, allowing for comprehensive data scraping directly from the source code of web pages.
- **APIs:** Accessing price data through official APIs provided by websites. Many e-commerce platforms and price comparison services offer APIs that deliver structured data in formats like JSON or XML. Using APIs ensures reliable and up-to-date information, as data is provided directly by the source. This

method is efficient and minimizes the risk of scraping-related issues, such as being blocked by the website for excessive requests.

- **Headless Browsers:** Using tools like Selenium or Puppeteer to automate interactions with web pages and extract dynamic content. Headless browsers simulate a real user browsing experience but operate in the background without a graphical interface. This is particularly useful for websites that load content dynamically through JavaScript, which traditional HTML parsing cannot easily capture. By automating actions like clicking, scrolling, and filling forms, headless browsers can access and extract all visible data, ensuring comprehensive price monitoring.

#### 1. Data Processing and Storage:

- **Data Cleaning:** Ensuring the extracted data is accurate and formatted correctly.
- **Database Management:** Storing the scraped data in databases for further analysis.

### 1. Understanding the Basics of Price Scraping

Price scraping is the automated process of extracting pricing information from various websites using specialized software. This practice helps businesses and individuals gather essential data for competitive analysis, market research, and strategic pricing decisions.

### 2. Tools and Technologies for Price Scraping

Several tools and technologies facilitate the price scraping process, including:

- **Web Scraping Libraries and Frameworks:**
  - **BeautifulSoup:** A Python library for parsing HTML and XML documents to extract data.
  - **Scrapy:** An open-source web crawling framework for Python, used to extract data from websites.
  - **Puppeteer:** A Node.js library for controlling headless Chrome or Chromium, enabling scraping of dynamic web content.

- **Cheerio:** A fast, flexible, and lean implementation of jQuery designed for server-side web scraping.
- **Headless Browsers:**
  - **Selenium:** A suite of tools for automating web browsers, often used for testing web applications but also effective for scraping.
  - **PhantomJS:** A headless WebKit scriptable with a JavaScript API, useful for web scraping.

### 3.Steps in Price Scraping

#### 3.1 Identifying the Target Website and Data

Before initiating the scraping process, it's crucial to identify the target website and the specific data points needed, such as product names, prices, and descriptions. This involves:

- **Selecting the URLs:** Determining the specific web pages where the desired price information is located.
- **Understanding the HTML Structure:** Analysing the HTML structure of the target pages to locate the elements containing the price data.

#### 3.2 Setting Up the Scraping Environment

- **Installing Necessary Libraries:** Ensure all required libraries and tools are installed (e.g., BeautifulSoup, Scrapy, Puppeteer).
- **Configuring the Scraper:** Set up the scraper to send requests to the target website and handle responses.

#### 3.3 Sending HTTP Requests

The scraper sends HTTP requests to the target website, mimicking a regular user's browsing behaviour. This can be done using libraries like requests in Python

#### 3.4 Parsing the HTML Content

Once the web page is retrieved, the scraper parses the HTML content to extract the necessary data. For instance, using BeautifulSoup in Python:

### 3.5 Handling Dynamic Content

Many modern websites use JavaScript to load content dynamically. To scrape such content, tools like Selenium or Puppeteer are employed:

### 3.6 Data Cleaning and Storage

Extracted data often requires cleaning to ensure accuracy and consistency. This might involve:

- **Removing Extraneous Characters:** Cleaning up currency symbols or whitespace from price values.
- **Converting Data Types:** Ensuring all prices are stored as numerical values for further analysis.

Cleaned data is then stored in a suitable format, such as CSV files, databases (e.g., MySQL, MongoDB), or cloud storage solutions.

### 3.7 Automating and Scheduling Scrapes

To keep data up-to-date, scrapers are often automated and scheduled to run at regular intervals. This can be achieved using cron jobs (Unix-based systems) or task schedulers (Windows).

### Legal and Ethical Considerations

Before implementing price scraping, it's essential to:

- **Review Website Terms of Service:** Ensure compliance with the target website's terms and conditions regarding data extraction.
- **Respect Robots.txt:** Adhere to the rules specified in the website's robots.txt file, which indicates permissible scraping activities.
- **Avoid Overloading Servers:** Implement rate limiting and respectful scraping intervals to prevent server overload.

## **Chapter 3**

# **DESIGN AND ANALYSIS**

## 3. DESIGN AND ANALYSIS

### 3.1 Unified Modelling Language (UML)

In this chapter we are going to use UML to describe and generate a blueprint for our system. UML will be used for the following purposes

- **Visualization** – This is used as a standardized tool to write and clearly specify a model for the system being developed
- **Specifying**– Models built using UML are precise, unambiguous and complete that leave no element of doubt for the developer
- **Constructing** – UML is object oriented in nature which means that after generation of these you can directly convert to code
- **Documenting** – These UML diagrams also serve as documentation for the system's architecture and intrinsic details

### 3.2 User Documentation

- The game shall provide a help system that will explain all function available on the user interface.
- The game shall provide the user with a tutorial that will explain the principles of the Dots and Boxes game.

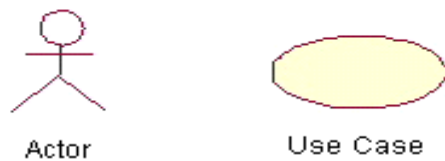
### 3.3 Assumptions and Dependencies

- The end-user must have a mobile or laptop.
- The mobile phone, laptop used by the end-user must have access to the internet.

### 3.4 Use Case Diagram

A use case is a set of scenarios that describing an interaction between a user and a system. A use case diagram displays the relationship among actors and use

cases. The two main components of a use case diagram are use cases and actors



An actor represents a user or another system that will interact with the system you are modelling. A use case is an external view of the system that represents some action the user might perform in order to complete a task.

Use cases are used in almost every project. These are helpful in exposing requirements and planning the project. During the initial stage of a project most use cases should be defined, but as the project continues more might become visible.

This use case describes the functionality of the Website Price Watch.

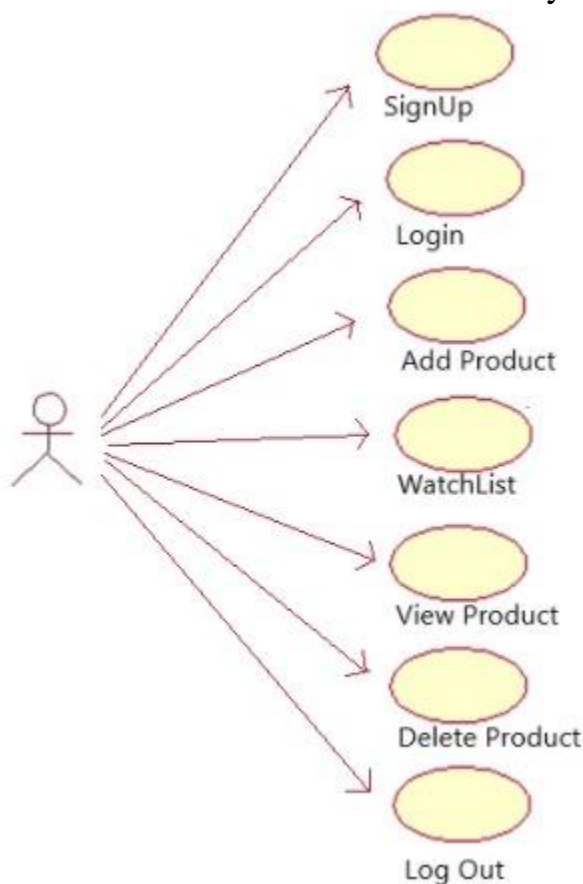


Figure 3.1 Use-Case Diagram



This example shows the user as an actor because the user is using the website. The diagram takes the simple steps listed above and shows them as actions the user might perform. From this simple diagram the functionalities of the web application can be easily understood. The system will need to be able to perform actions for all of the use cases listed.

### 3.5 Interaction Diagrams

Interaction diagrams model the behaviour of use cases by describing the way groups of objects interact to complete the task. The two kinds of interaction diagrams are **sequence** and **collaboration** diagrams.

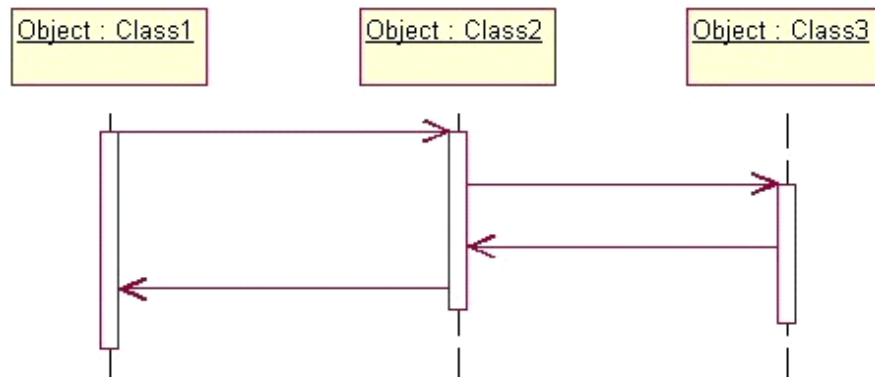
Interaction diagrams are used when you want to model the behaviour of several objects in a use case. They demonstrate how the objects collaborate for the behaviour. Interaction diagrams do not give an in-depth representation of the behaviour.

Sequence diagrams, collaboration diagrams, or both diagrams can be used to demonstrate the interaction of objects in a use case. Sequence diagrams generally show the sequence of events that occur. Collaboration diagrams demonstrate how objects are statically connected. Both diagrams are relatively simple to draw and contain similar elements.

#### 3.5.1 Sequence Diagrams:

Sequence diagrams demonstrate the behaviour of objects in a use case by describing the objects and the messages they pass. The diagrams are read left to right and descending. The example below shows an object of class 1 start the behaviour by sending a message to an object of class 2. Messages pass between the different objects until the object of class 1 receives the final message

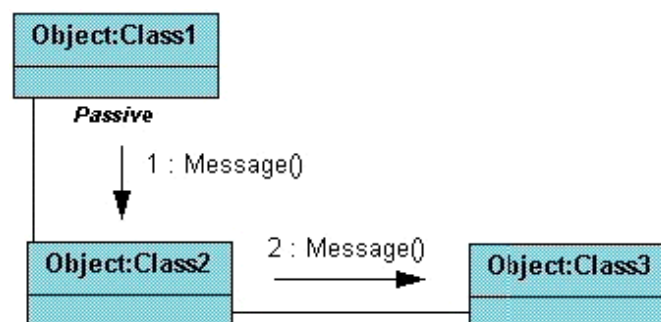
Example:



### 3.5.2 Collaboration Diagrams

Collaboration diagrams are also relatively easy to draw. They show the relationship between objects and the order of messages passed between them. The objects are listed as icons and arrows indicate the messages being passed between them. The numbers next to the messages are called sequence numbers. As the name suggests, they show the sequence of the messages as they are passed between the objects. There are many acceptable sequence numbering schemes in UML. A simple 1, 2, 3... format can be used.

Example:



## **3.6 Interaction Diagrams for PRICE WATCH**

### **3.6.1 Signup**

In the Signup process for PRICE WATCH, users begin by creating an account, which is illustrated through various diagrams. Figure 3.2 depicts the sequence diagram, outlining the step-by-step interactions between the user and the system during the account setup. This diagram highlights the flow of messages and responses, such as user input, validation checks, and confirmation prompts. Meanwhile, Figure 3.3 presents the collaboration diagram, which focuses on the relationships and interactions among different components involved in setting a password. This diagram illustrates how the user, password management system, and validation services collaborate to complete the password setup efficiently. Together, these diagrams provide a comprehensive view of the Signup process, emphasizing the sequence of operations and the interaction between various elements to ensure a smooth user experience.

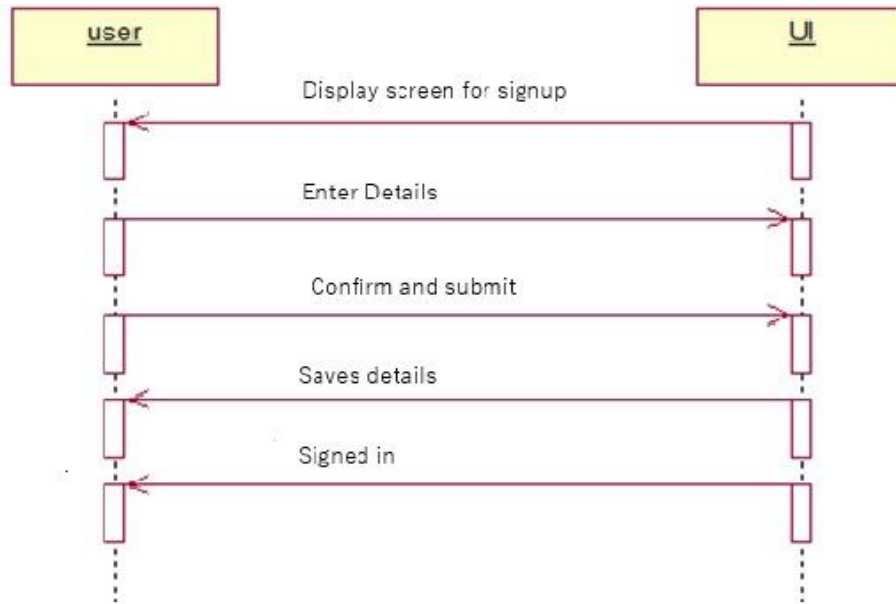


Figure 3.2 Signup Sequence Diagram



Figure 3.3 Signup Collaboration Diagram

### 3.6.2 Login

In the Login process for PRICE WATCH, users access their accounts by providing their credentials, which is depicted through detailed interaction diagrams. The sequence diagram, shown in Figure 3.4, captures the step-by-step communication between the user, the login interface, and the authentication system. This diagram outlines how user inputs, such as email and password, are transmitted to the system,

which then performs authentication checks and returns a response, either granting or denying access based on the validity of the credentials. Figure 3.5 presents the collaboration diagram, focusing on the interactions between different components, including the user interface, authentication server, and session management system. This diagram illustrates how these components work together to verify user credentials and establish a secure session. The combination of these diagrams provides a detailed understanding of the Login process, highlighting the flow of information and the collaborative effort required to ensure secure and

#### Efficient-access application

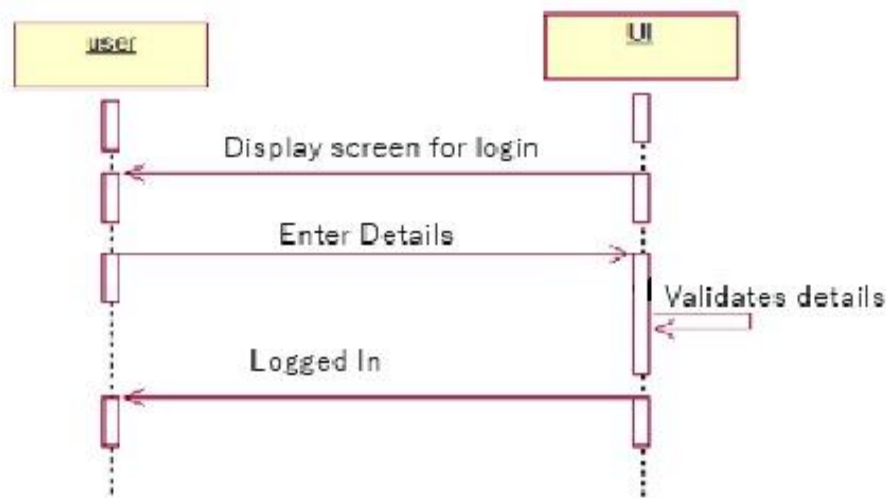


Figure 3.4 Login Sequence Diagram



Figure 3.5 Login Collaboration Diagram

### 3.6.3 Entering Product Details

Entering product details in PRICE WATCH is a crucial step for users to effectively monitor and manage their desired products. Figure 3.6 presents the sequence diagram, Initially, users need to provide essential information about the product they are interested in, including the product name, url. This allows the system to track and identify the product correctly.

Once the basic details are entered, users also need to set the desired price threshold for notifications. This setup enables PRICE WATCH to track the product's price across various retailers and alert users when the price meets their criteria. Figure 3.7 presents the collaboration diagram.

The system processes and stores these details, integrating them into the user's personalized watchlist. This functionality ensures that users receive timely updates on price changes, helping them make informed decisions and potentially save money by purchasing the product when it falls within their preferred price range.

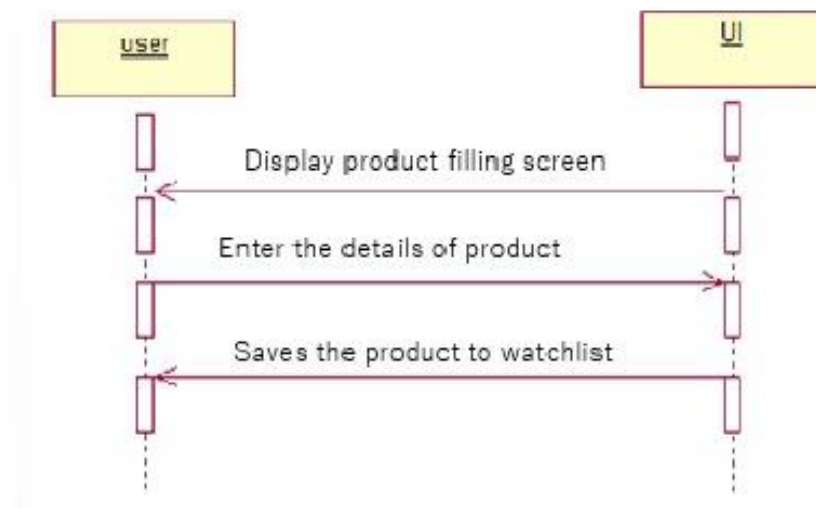


Figure 3.6 Entering Product Details Sequence Diagram

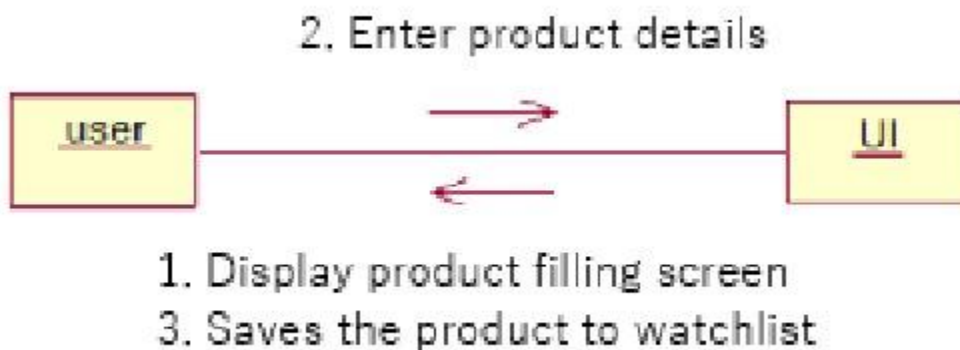


Figure 3.7 Entering Product Details Collaboration Diagram

#### 3.6.4 Products in Watchlist

After filling in the product details, the product is added to the user's watchlist in PRICE WATCH. Figure 3.8 presents the sequence diagram, The watchlist acts as a personalized tracking tool where users can easily monitor the status of all their selected products. Each item on the watchlist is displayed with its key details, including current price, delete option, view product option, the product name. Figure 3.9 presents the collaboration diagram.

The watchlist also allows users to set preferences for notifications. Users can choose to receive alerts when the product's price drops to a certain level. This feature ensures users are kept informed about the best times to purchase based on their specified criteria, enhancing their ability to make cost-effective decisions.



Figure 3.8 Watchlist Screen Sequence Diagram





Figure 3.9 Watchlist Screen Collaboration Diagram

### 3.6.5 View product

When a user clicks on a "view Product" option in their watchlist on PRICE WATCH, they are prompted to page of the product. This action guides the user by taking him/her to the original product page for easy access.

This describes how use can visit the product. The Figure3.10 shows the sequence diagram and Figure3.11 shows the collaboration diagram to view product.



Figure 3.10 View Product Sequence Diagram

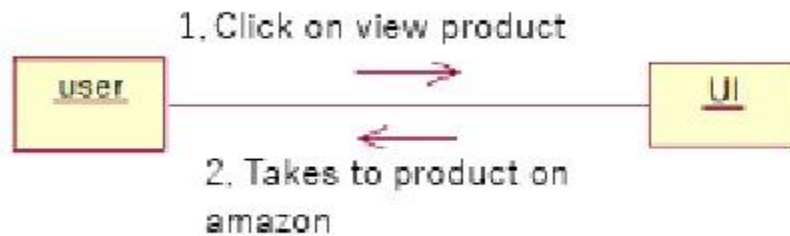


Figure 3.11 View Product Collaboration Diagram

### 3.6.6 Logging out

When a user selects the "Logout" option in PRICE WATCH, the system securely terminates the user session. This action ensures that all personal and session-specific data is cleared from the device, safeguarding user privacy.

After logging out, users are typically redirected to the login page or homepage. This allows them to sign in again when needed, ensuring that only authenticated users can access their watchlist and other personalized features. The Figure3.12 shows the sequence diagram and Figure3.13 shows the collaboration diagram to log out.

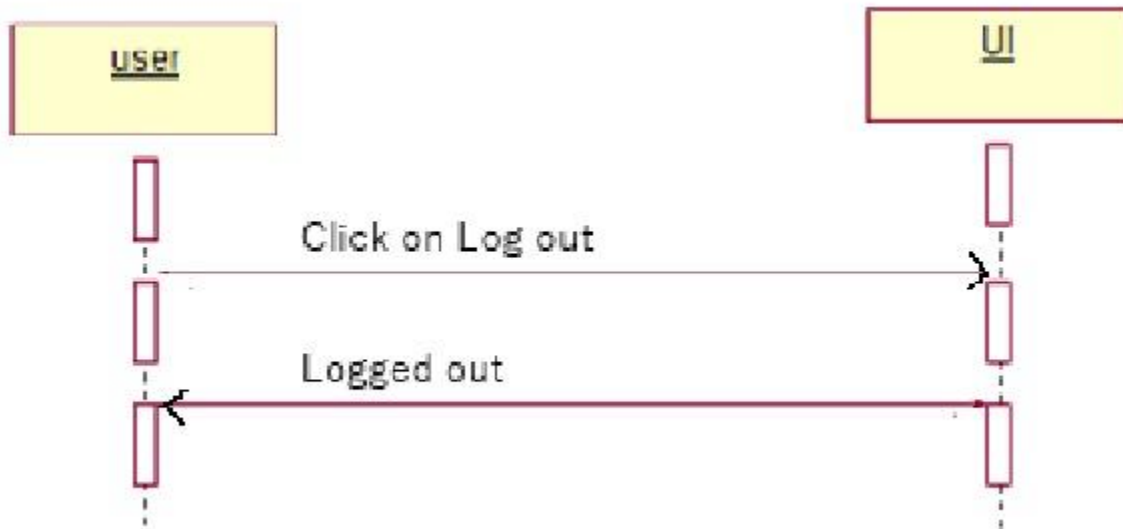


Figure 3.12 Logout Sequence Diagram



Figure 3. 13 Logout Collaboration Diagram

### 3.6.7 Deleting Product

Deleting a product from the watchlist in PRICE WATCH removes it from tracking and stops any associated notifications. The watchlist is updated to exclude the deleted item, reflecting the change immediately. The Figure3.14 shows the sequence diagram and Figure3.15 shows the collaboration diagram to delete product.



Figure 3.14 Delete Product Sequence Diagram



Figure 3.15 Delete Product Collaboration Diagram

## **CHAPTER 4**

### **INSTALLATIONS**

## 4. INSTALLATIONS

### 4.1 Installation of Visual Studio Code

To install Visual Studio Code (VS Code), follow these steps for your operating system:

#### Windows

1. **Download:** Visit [Visual Studio Code](#) and download the Windows installer.
2. **Run Installer:** Double-click the .exe file and follow the setup wizard.
3. **Launch:** Choose to launch VS Code after installation or find it in the Start Menu.

### 4.2 Installation of MongoDB Compass

To install MongoDB Compass, follow these steps:

#### Windows and macOS

1. **Download:** Go to the [MongoDB Compass download page](#) and select the appropriate version for your operating system.
2. **Run Installer:**
  - **Windows:** Double-click the .msi installer file and follow the setup wizard.
  - **macOS:** Open the .dmg file and drag MongoDB Compass to the Applications folder.
3. **Launch:** Find MongoDB Compass in your applications or programs list and open it.

### 4.3 Installation of Python

To install Python, follow these steps:

## Windows

1. **Download:** Go to the [Python website](#) and download the Windows installer (.exe file).
2. **Run Installer:** Double-click the downloaded file. Check the box "Add Python to PATH" and click "Install Now."
3. **Verify:** Open Command Prompt and run `python --version` to ensure it is installed correctly.

## 4.4 Installation of NodeJS

To install Node.js, follow these steps:

### Windows and macOS

1. **Download:** Go to the Node.js website and download the installer for your operating system.
2. **Run Installer:**
  - **Windows:** Double-click the .msi file and follow the setup wizard.
  - **macOS:** Open the .pkg file and follow the installation instructions.
3. **Verify:** Open a terminal or command prompt and run:

```
node --version  
npm --version
```

## 4.5 Installation of ReactJS

To install React.js on Windows, follow these steps:

### 1. Install Node.js

1. **Download Node.js:**
  - Visit the Node.js website and download the Windows installer (.msi file).
2. **Run the Installer:**
  - Double-click the downloaded .msi file and follow the setup wizard. Ensure "Add to PATH" is checked.
3. **Verify Installation:**

- Open Command Prompt and run:
- `node --version`
- `npm --version`

## **2. Create a React Application**

- 1. Open Command Prompt.**
- 2. Use Create React App:**

```
npx create-react-app my-app
```

Replace my-app with your preferred project name.

- 3. Navigate to Your Project Directory:**

```
cd my-app
```

- 4. Start the Development Server:**

```
npm start
```

Your React application will open in your default web browser at <http://localhost:3000>.

## **4.6 Installation of Axios**

To install Axios:

- 1. Open Terminal/Command Prompt.**
- 2. Run:**

```
npm install axios
```

## **4.7 Installation of React**

To install react:

- 1. Open Terminal/Command Prompt.**
- 2. Run:**

```
npm install react
```



## 4.8 Installation of React-Dom

To install react-dom:

1. **Open Terminal/Command Prompt.**
2. **Run:**

```
npm install react-dom
```

## 4.9 Installation of React-Hot-Toast

To install react-hot-toast:

1. **Open Terminal/Command Prompt.**
2. **Run:**

```
npm install react-hot-toast
```

## 4.10 Installation of React-Icons

To install react-icons:

1. **Open Terminal/Command Prompt.**
2. **Run:**

```
npm install react-icons
```

## 4.11 Installation of React-Router-Dom

To install react-router-dom:

1. **Open Terminal/Command Prompt.**
2. **Run:**

```
npm install react-router-dom
```

## 4.12 Installation of React-Script:

To install react-script:

1. **Open Terminal/Command Prompt.**

**2. Run:**

```
npm install react-script
```

#### **4.13 Installation of Web-Vitals**

To install web-vitals:

- 1. Open Terminal/Command Prompt.**
- 2. Run:**

```
npm install web-vital
```

#### **4.14 Installation of Express**

To install express:

- 1. Open Terminal/Command Prompt.**
- 2. Run:**

```
npm install express
```

#### **4.15 Installation of Mongoose**

To install mongoose:

- 1. Open Terminal/Command Prompt.**
- 2. Run:**

```
npm install mongoose
```

#### **4.16 Installation of Nodemon**

To install nodemon:

- 1. Open Terminal/Command Prompt.**
- 2. Run:**

```
npm install nodemon
```

#### **4.17 Installation of Pymongo**

To install pymongo:

1. **Open Terminal/Command Prompt.**
2. **Run:**

```
npm install pymongo
```

#### **4.18 Installation of Selenium**

```
pip install selenium
```

#### **4.19 Installation of ChromeDriver**

```
npm install chromedriver
```

## **Chapter 5**

# **PROCEDURE**

## 5. PROCEDURE

### 5.1 Working of Frontend

#### Home.js:

The `Home` component in this React application manages the user interface for adding products to a watchlist. It starts by importing necessary libraries and setting up state variables using the `useState` hook to handle form inputs: `url`, `name`, and `maxPrice`. The `handleSubmit` function is triggered when the form is submitted; it sends a POST request with the product details to the server endpoint `/add-product` using Axios. Depending on the response, it either shows a success message or an error message using `react-hot-toast`, and clears the form fields on success.

The component also includes navigation functionalities. It utilizes `useNavigate` from `react-router-dom` to manage page transitions. The `handleLogout` function logs the user out by navigating to the home page and displaying a logout message, while `handleWatchList` redirects the user to the watchlist page. The UI features a navigation bar with buttons for accessing the watchlist and logging out, and a form for inputting product details. Styling is applied through the `Home.css` file to ensure a visually appealing layout.

#### LandingPage.js

The `LandingPage` component displays a welcome message and toggles between "Log In" and "Sign Up" forms based on the `isSignIn` state. It uses the `loginHandler` function to switch views. The component includes conditional buttons and text to allow users to switch between signing in and signing up. Styling is managed through `LandingPage.css`.

## Login.js

The **LogIn** component handles user authentication in the Price Watch application. It uses `useRef` to manage references to the email and password input fields. On form submission, it retrieves the values, checks for empty fields, and sends a POST request to the `/login` endpoint using Axios. If the login is successful (status 200), it stores the email in `localStorage`, displays a success message, and navigates to the Home page. Otherwise, it shows an error message. The form's layout and styling are handled through `LogIn.css`.

## Singup.js

The **SignUp** component manages user registration with password visibility toggles. It uses `useRef` to access email, password, and confirm password inputs, and `useState` to control password visibility. On form submission, it checks if all fields are filled and if passwords match. It sends a POST request to the `/signup` endpoint using Axios. On successful registration, it stores the email in `localStorage`, displays a success message, and navigates to the Home page. If there are errors, appropriate messages are shown. Styling is applied through `SignUp.css`.

## WatchList.js

The **Watchlist** component retrieves and displays a list of products that the logged-in user is monitoring. Using `useEffect`, it fetches products from the backend based on the email stored in `localStorage` when the component mounts. The fetched products are stored in the `products` state and displayed using the `Card` component. Users can delete products from the list, which updates the state to remove the item. The component also includes navigation buttons for returning to the home page or logging out, with corresponding handlers for navigation and user logout. Styling is managed through `WatchList.css`.

## 5.2 Working of Backend

### Index.js

The provided Express server code handles user authentication and product management for a Price Watch application. It includes endpoints for signing up and logging in users, which interact with a MongoDB database to check for existing users or validate credentials. Users can add products to a watchlist, and the server checks for duplicates before saving new entries. Products are fetched based on the logged-in user's email and can be deleted by product ID. The server responds with appropriate success or error messages and logs any issues encountered during database operations. The server runs on a specified port and uses middleware to parse JSON and URL-encoded data.

### MongoDB.js

The backend code connects to a local MongoDB database using Mongoose and defines a schema for user login data, including required fields for `email` and `password`. It creates a Mongoose model `LogInCollection` based on this schema to manage user records in the `LogInCollection` collection. Upon successful connection to MongoDB, a confirmation message is logged; otherwise, an error message is displayed. The `LogIn Collection` model is exported for use in handling user authentication operations in the application.

### Product.js

This backend code sets up a Mongoose connection to a local MongoDB database and defines a schema for product data using Mongoose. The `productSchema` includes fields for `email`, `name`, `url`, `highprice`, `price`, and `registration\_email\_sent`, with appropriate types and default values. A Mongoose model `ProductCollection` is created from this schema to manage products in the `ProductCollection` collection. This model is then exported for use in handling product-related operations in the application.

### **5.3 Working of Price Tracking.**

#### **Pythonselenium.py**

The backend script connects to a MongoDB database to fetch product details and uses Selenium to scrape current prices from product URLs. It runs a loop to periodically check these prices, comparing them against user-defined thresholds. If a price drops below the threshold, it sends an email notification to the user. Additionally, the script sends an initial registration email when a product is first tracked and updates the product's price in the database. It operates continuously, handling exceptions and retrying after delays.

### **5.4 How To Run The Frontend, Backend, Python**

**"Install the packages specified in Chapter 3, including Express, Mongoose, Nodemon, and any others mentioned there."**

#### **1,Frontend:**

Open Terminal

Run the Commands:

- cd frontend
- npm start
- As shown in Figure 5.1



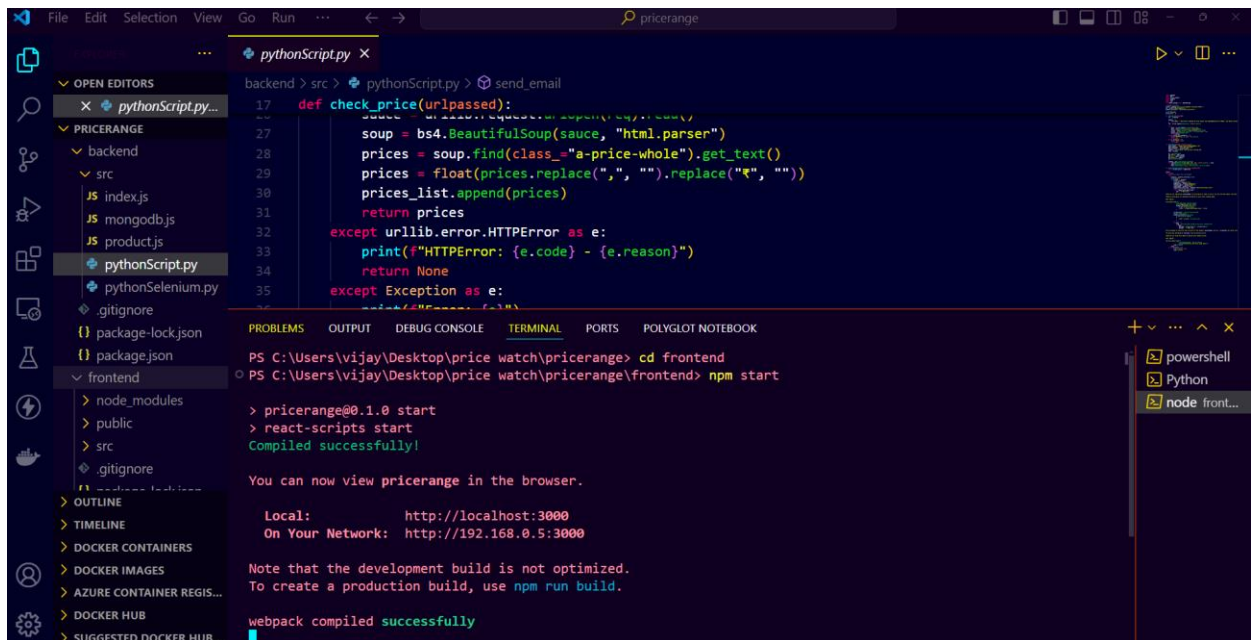


Figure 5.1 starting Frontend

- **2.Bakend:**

Open new terminal by clicking + symbol

Run the Commands:

- cd backend
- nodemon src/index.js

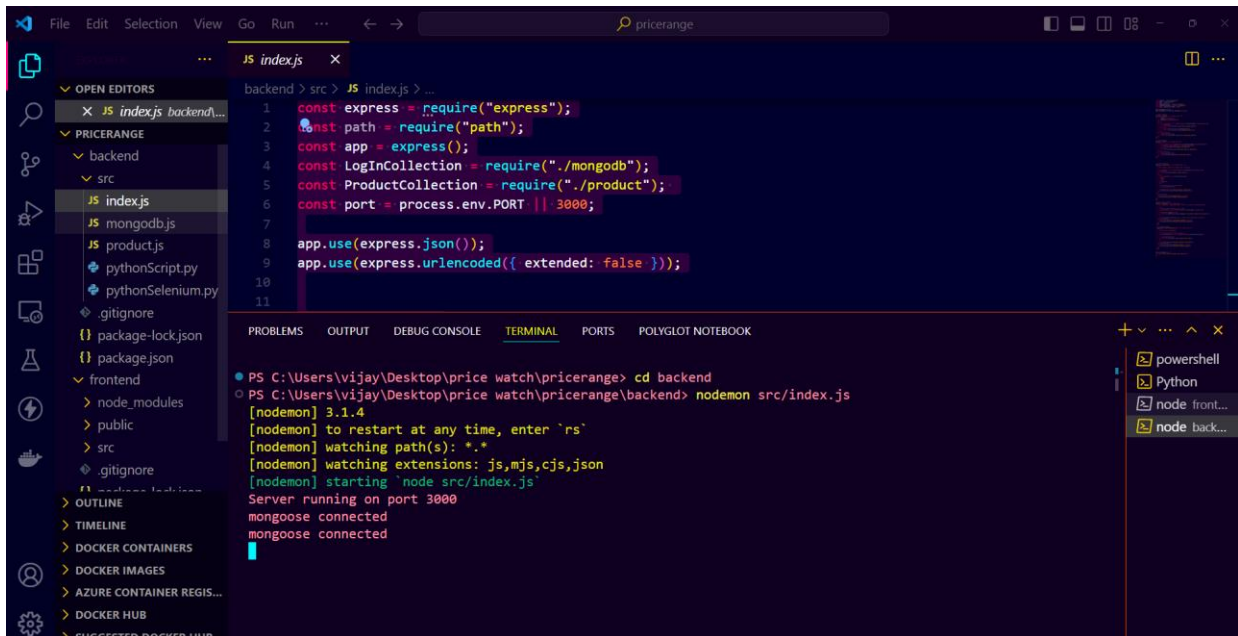


Figure 5.2 starting Backend

3.Run the python by clicking the Run Button.

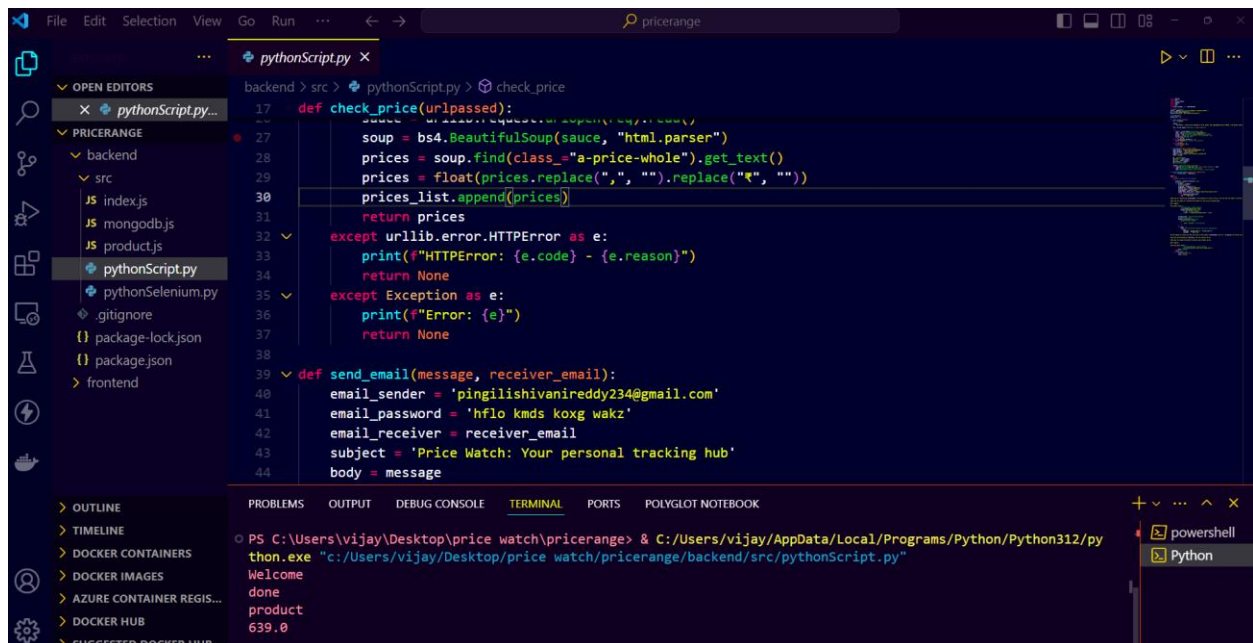


Figure 5.3 starting Python

## **Chapter 6**

# **TESTING AND RESULTS**

## 6. Testing and Results

### 6.1 Initial Screen

The following screen appears as shown in Figure 6.1, as soon as the user opens the application.

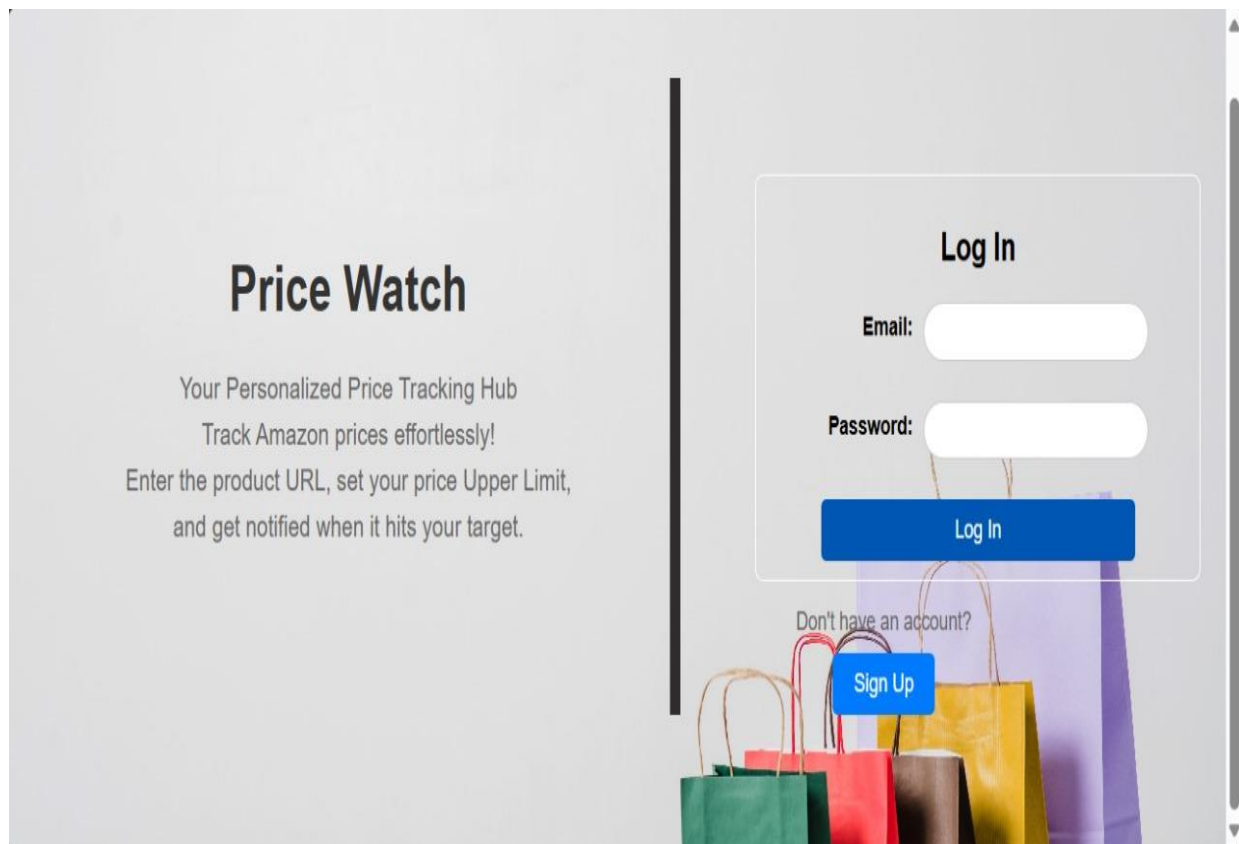
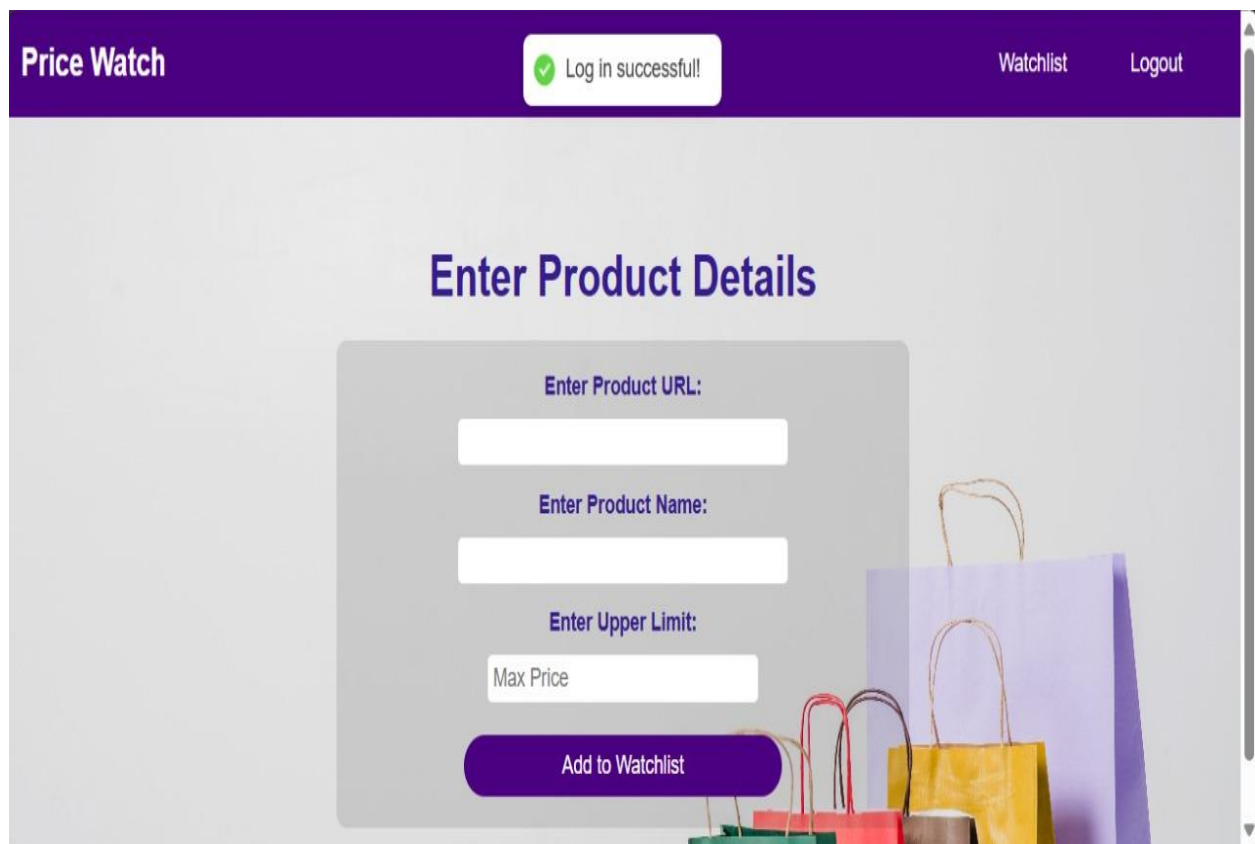


Figure 6.1 Initial Screen

## 6.2 Login Successful Toast

A screen as shown in figure 6.2 is displayed when user logs in successfully.



The screenshot displays the 'Price Watch' application interface. At the top, a purple header bar contains the text 'Price Watch' on the left, a green toast notification 'Log in successfull!' in the center, and 'Watchlist' and 'Logout' links on the right. The main content area has a light gray background with the heading 'Enter Product Details' in a large, bold, dark blue font. Below this heading is a form with three input fields: 'Enter Product URL:', 'Enter Product Name:', and 'Enter Upper Limit:'. The 'Enter Upper Limit:' field has a placeholder text 'Max Price'. A purple button labeled 'Add to Watchlist' is positioned below the form fields. In the bottom right corner, there is a decorative image of several colorful shopping bags (purple, yellow, red, green).

Figure 6.2 Login Successful Toast

## 6.3 Unknown User Login Fail

shown in Figure 6.3 is displayed when user failed to login by entering wrong email or password.

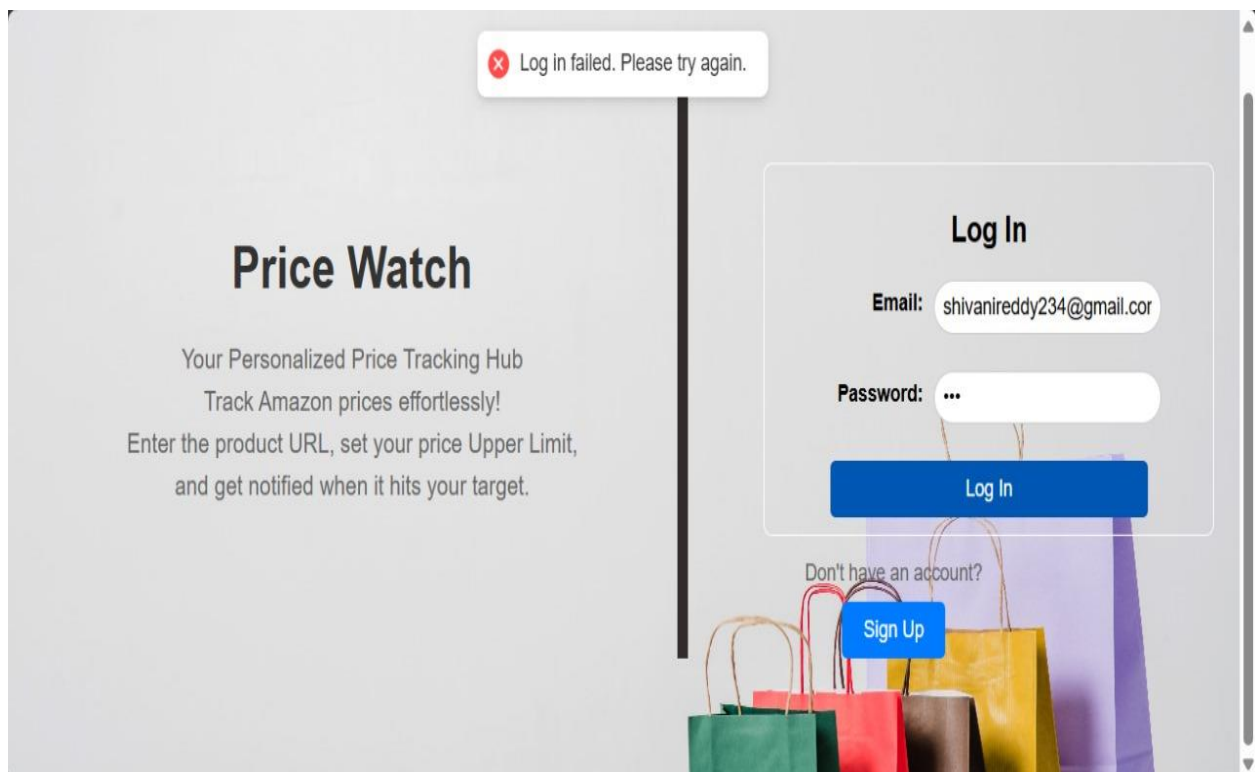
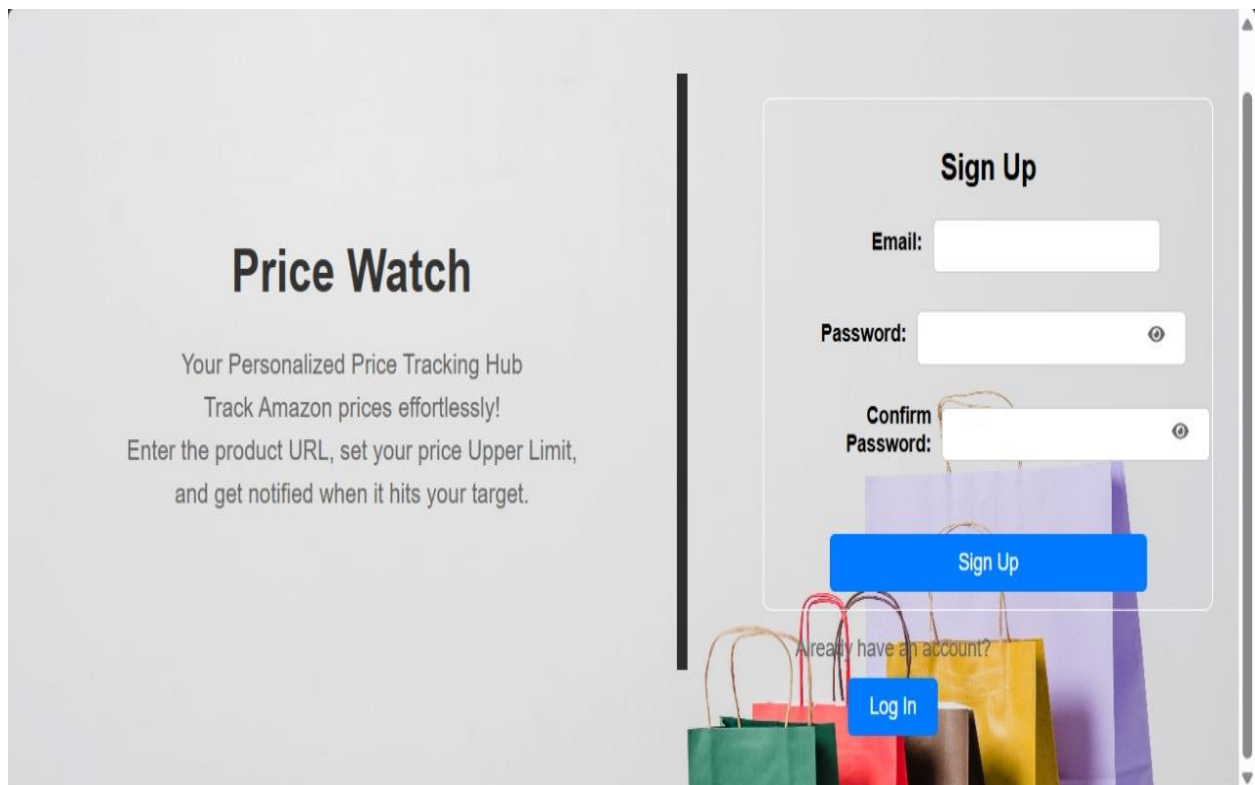


Figure 6.3 Unknown User Login fail

## 6.4 Empty Signup



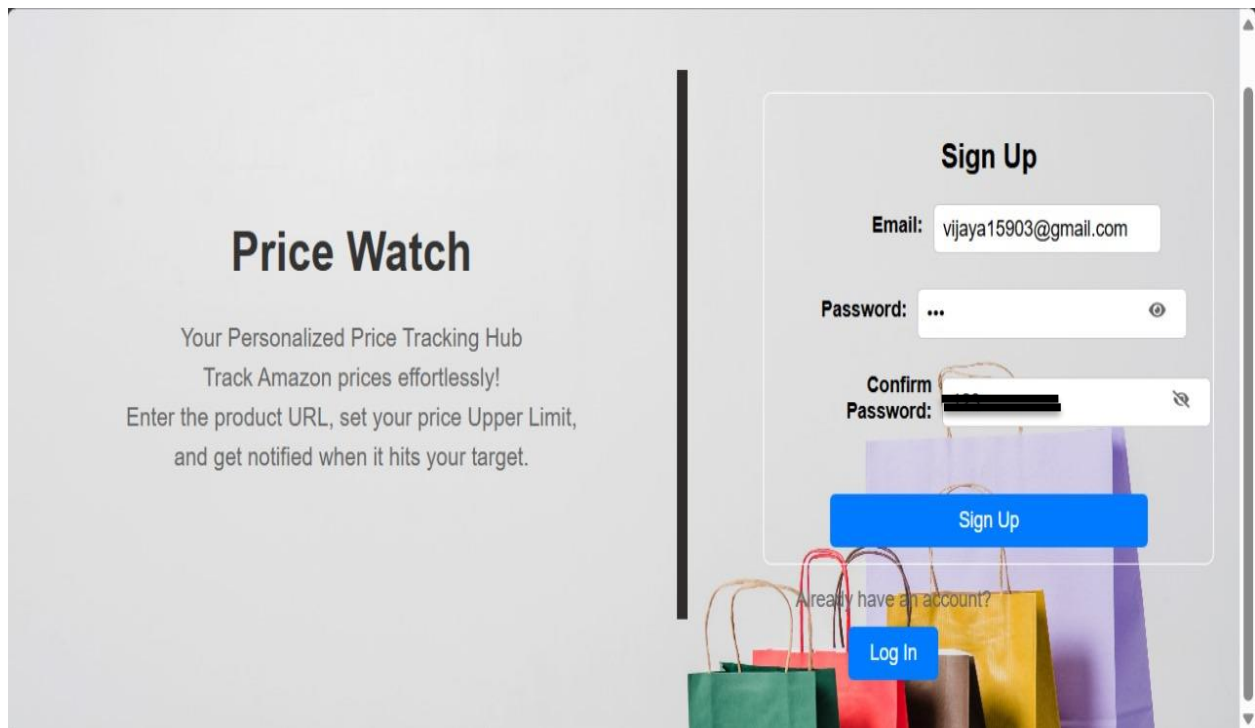
The screenshot displays the 'Price Watch' application interface. On the left, the title 'Price Watch' is prominently displayed in a large, bold, black font. Below it, the text 'Your Personalized Price Tracking Hub' and 'Track Amazon prices effortlessly!' are shown in a smaller font. Further down, a descriptive sentence reads: 'Enter the product URL, set your price Upper Limit, and get notified when it hits your target.' On the right side of the screen, there is a white rectangular box containing the 'Sign Up' form. The form has a title 'Sign Up' at the top. It includes three input fields: 'Email:', 'Password:', and 'Confirm Password:'. Each field has a corresponding label to its left. The 'Password:' and 'Confirm Password:' fields have a small eye icon to their right, indicating a toggle for password visibility. Below the input fields is a blue button labeled 'Sign Up'. At the bottom of the form box, there is a link that says 'Already have an account?'. Below the form box, there is a blue button labeled 'Log In'. The background of the app is a light gray, and at the bottom, there is a decorative image of several colorful shopping bags (green, red, yellow, and purple).

Figure 6.4 Empty Signup

A Screen as shown in the Figure 6.4 is displayed when user clicks sign up.

## 6.5 Signup Filling

A Screen as shown in Figure 6.5 is displayed when the user signing up.



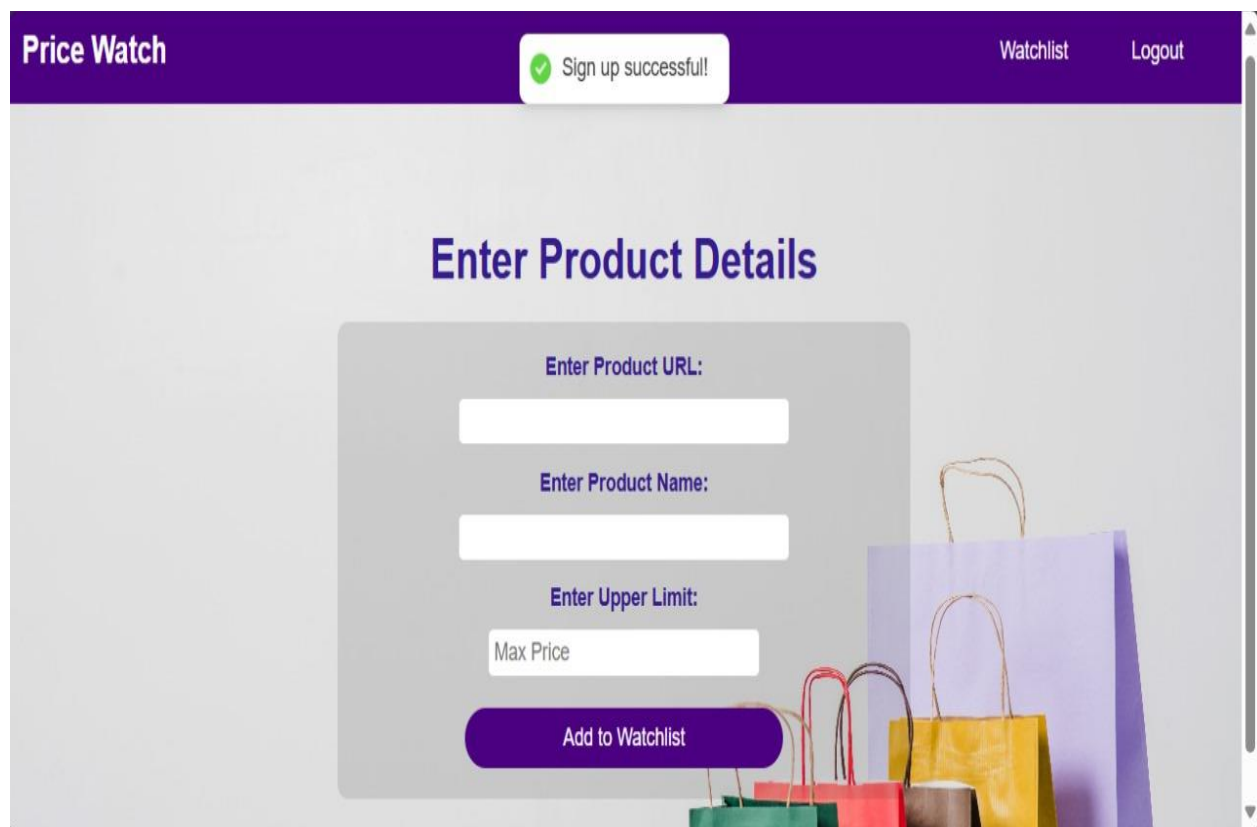
The screenshot displays a web interface for a 'Price Watch' application. On the left, the heading 'Price Watch' is prominently displayed, followed by the subtext 'Your Personalized Price Tracking Hub' and 'Track Amazon prices effortlessly!'. Below this, a brief description states: 'Enter the product URL, set your price Upper Limit, and get notified when it hits your target.' On the right side, a 'Sign Up' form is presented within a white-bordered box. The form contains three input fields: 'Email' (filled with 'vijaya15903@gmail.com'), 'Password' (filled with three dots and featuring an eye icon for toggling visibility), and 'Confirm Password' (filled with black dots and featuring a checkmark icon for verification). A blue 'Sign Up' button is positioned below the 'Confirm Password' field. At the bottom of the form box, the text 'Already have an account?' is visible, accompanied by a blue 'Log In' button. The background of the entire page is a light gray, and the bottom edge features a decorative graphic of several colorful shopping bags in green, red, yellow, and purple.

Figure 6.5 Displaying Signup Filling



## 6.6 Signup Successful Toast

A Screen as shown in the Figure 6.6appears, when user signup successfully.

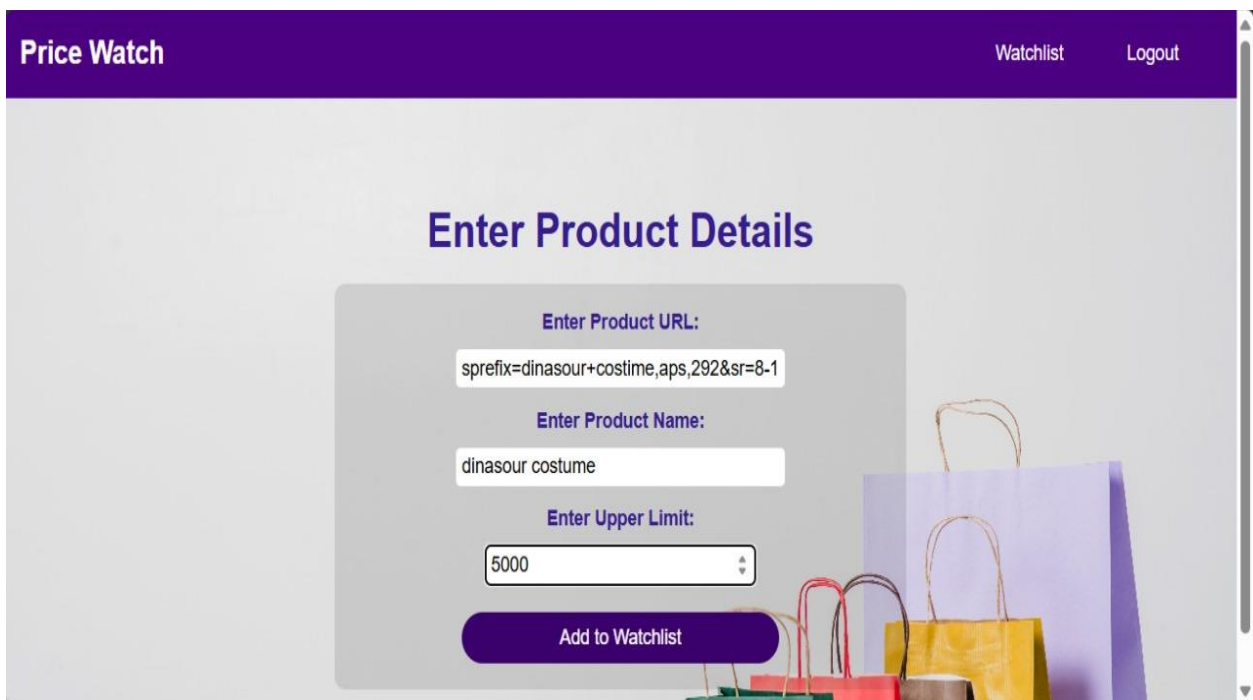


The screenshot displays the 'Price Watch' application interface. At the top, a purple header bar contains the text 'Price Watch' on the left, a green toast notification in the center that reads 'Sign up successful!' with a green checkmark icon, and the links 'Watchlist' and 'Logout' on the right. Below the header, the main content area has a light gray background. Centered in this area is the heading 'Enter Product Details' in a dark blue font. Underneath this heading is a gray rounded rectangle containing a form. The form has three input fields: 'Enter Product URL:' with a white input box, 'Enter Product Name:' with a white input box, and 'Enter Upper Limit:' with a white input box containing the text 'Max Price'. Below these fields is a purple rounded button labeled 'Add to Watchlist'. In the bottom right corner of the form area, there is a decorative image of several colorful shopping bags (purple, yellow, red, green, and brown).

Figure 6.6 Signup Successful Toast

## 6.7 Entered Product Details

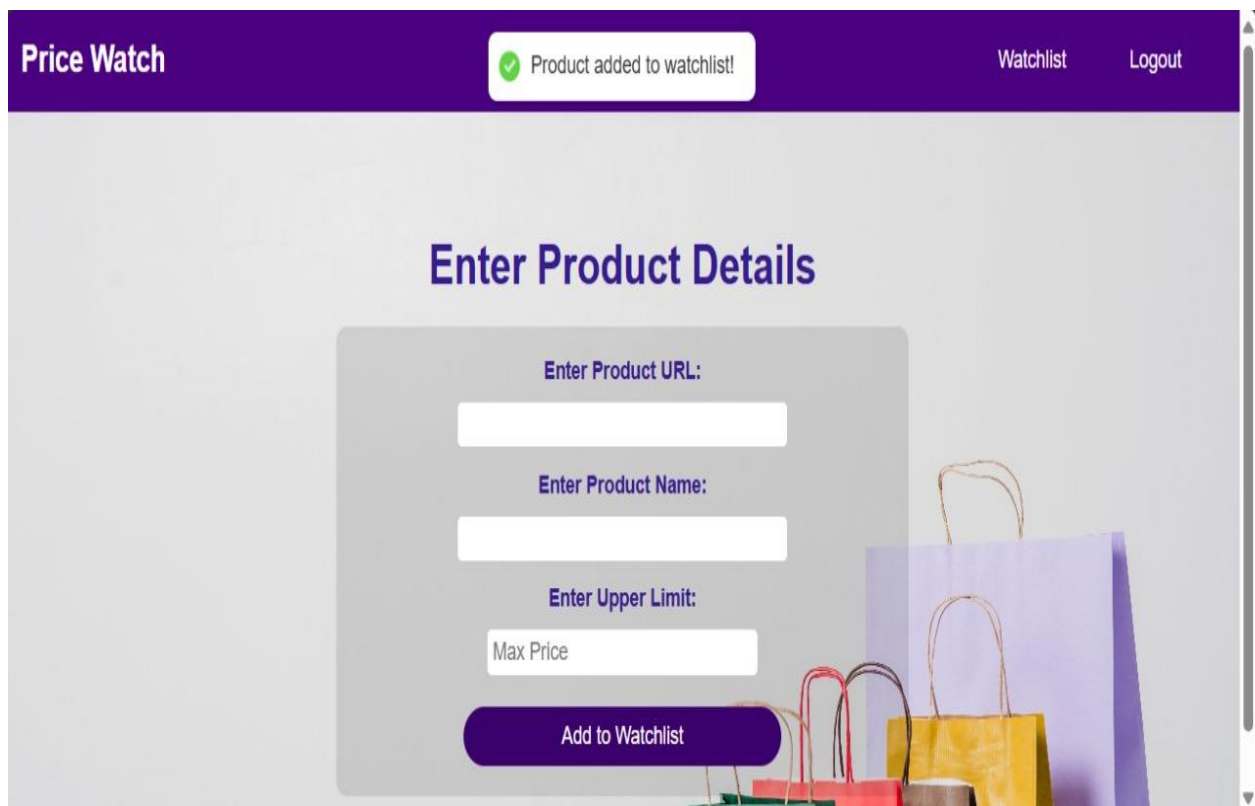
Screen as shown in Figure 6.7 is displayed when user entered the product details.



The screenshot displays a web application titled "Price Watch" with a purple header bar. In the top right corner of the header, there are links for "Watchlist" and "Logout". The main content area has a light gray background and features a central form titled "Enter Product Details" in a bold, dark blue font. The form is contained within a light gray rounded rectangle and includes three input fields: "Enter Product URL:" with the value "sprefix=dinasour+costime,aps,292&sr=8-1", "Enter Product Name:" with the value "dinasour costume", and "Enter Upper Limit:" with the value "5000". Below these fields is a purple button labeled "Add to Watchlist". To the right of the form, there is a decorative image of several colorful shopping bags (purple, yellow, red, and green) stacked together.

Figure 6.7 Entered Product Details

## 6.8 Product Added to Watchlist

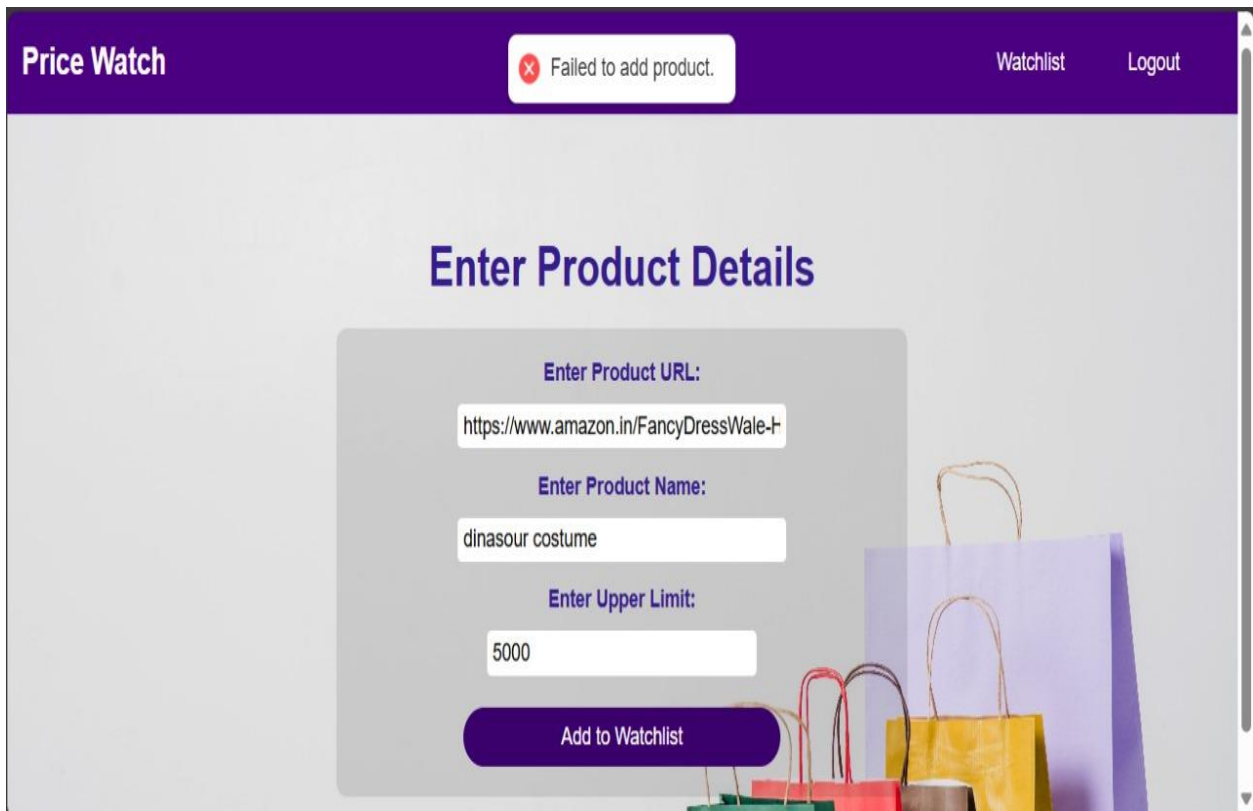


The screenshot displays the 'Price Watch' web application interface. At the top, a purple header bar contains the text 'Price Watch' on the left, a white notification box with a green checkmark and the text 'Product added to watchlist!' in the center, and the links 'Watchlist' and 'Logout' on the right. Below the header, the main content area has a light gray background. Centered in this area is the heading 'Enter Product Details' in a bold, dark blue font. Underneath this heading is a gray rounded rectangle containing four input fields and a button. The first field is labeled 'Enter Product URL:' and is empty. The second field is labeled 'Enter Product Name:' and is empty. The third field is labeled 'Enter Upper Limit:' and contains the text 'Max Price'. Below these fields is a purple rounded button with the text 'Add to Watchlist' in white. To the right of the input fields, there is a decorative image of several colorful shopping bags (purple, yellow, red, and green) stacked together.

Figure 6.8 Product Added to Watchlist

## 6.9 Existing URL Failed

A screen as shown in figure 6.9 is displayed when user tries adding existing product.



The screenshot displays the 'Price Watch' application interface. At the top, a purple header bar contains the text 'Price Watch' on the left, a white notification box with a red 'x' icon and the message 'Failed to add product.' in the center, and the links 'Watchlist' and 'Logout' on the right. Below the header, the main content area has a light gray background. Centered in this area is the heading 'Enter Product Details' in a bold, dark blue font. Underneath this heading is a gray rounded rectangle containing a form. The form has three input fields: 'Enter Product URL:' with the text 'https://www.amazon.in/FancyDressWale-+', 'Enter Product Name:' with the text 'dinasour costume', and 'Enter Upper Limit:' with the text '5000'. Below these fields is a purple button with the text 'Add to Watchlist'. To the right of the form, there is a decorative image of several colorful shopping bags (purple, yellow, red, green) stacked together.

Figure 6.9 Existing URL Failed

## 6.10 Watchlist

The screen is displayed as shown in Figure 6.10, when user retrieved products.

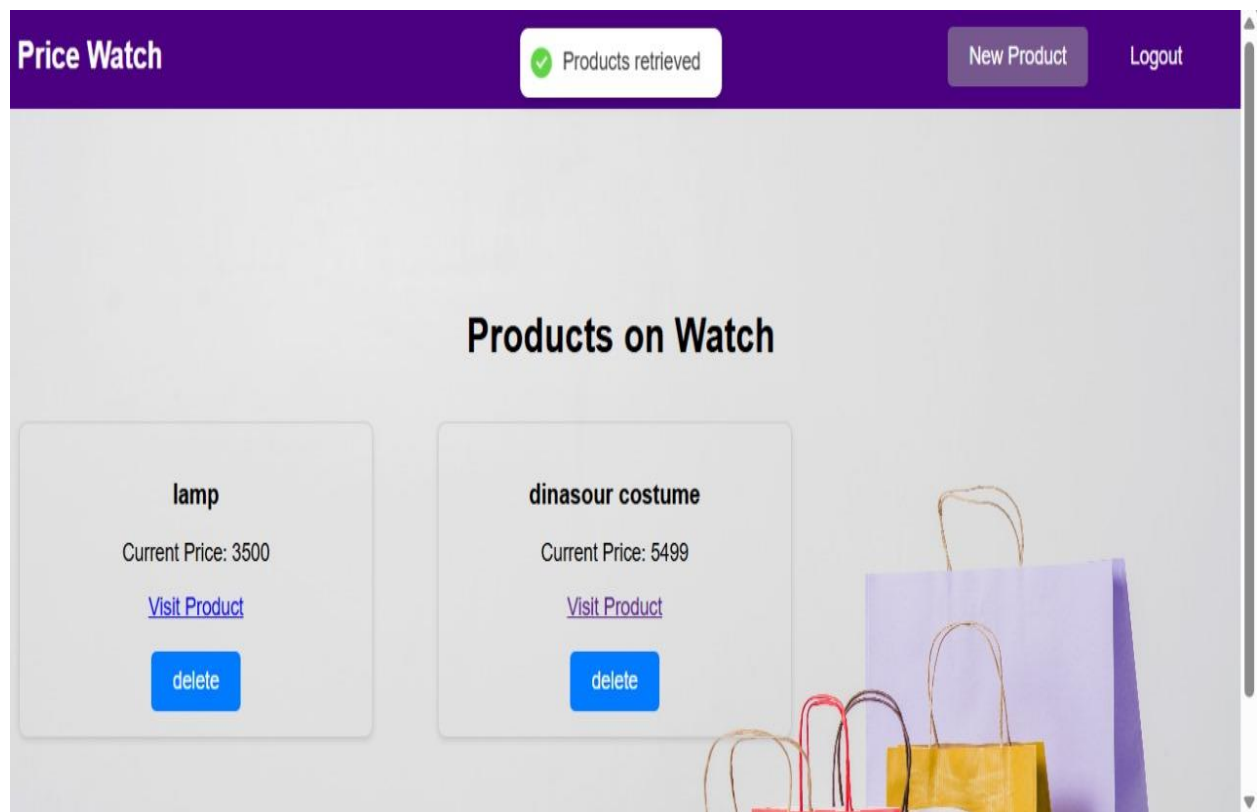


Figure 6.10 Watchlist

## 6.11 No Products in Watchlist

The Screen in Fig: 6.11 is displayed when user navigates to Watchlist

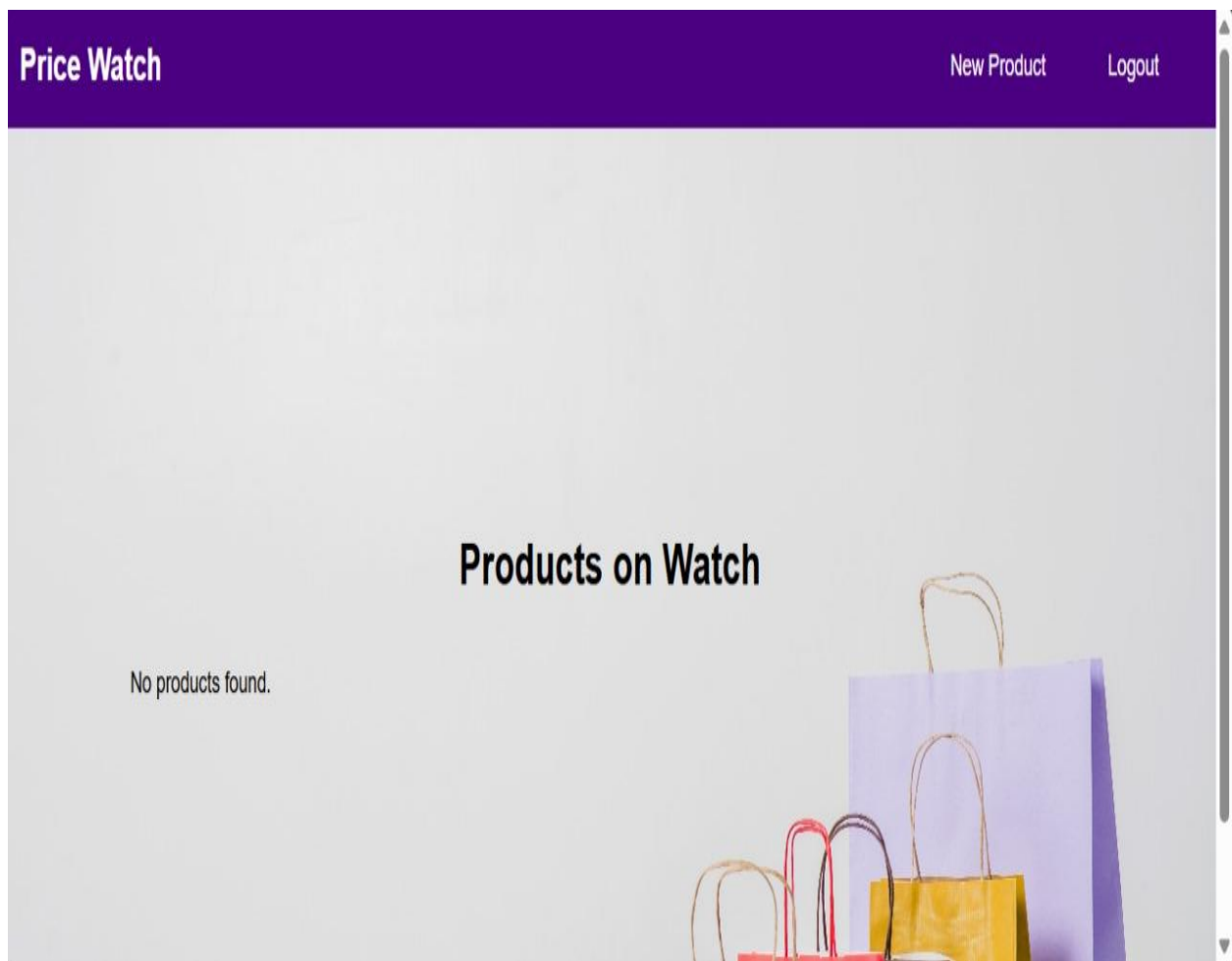


Figure 6.11 No Products in Watchlist

## 6.12 After Delete Product

The screen as shown in the Figure 6.12 is displayed when user deleted the product from watchlist.

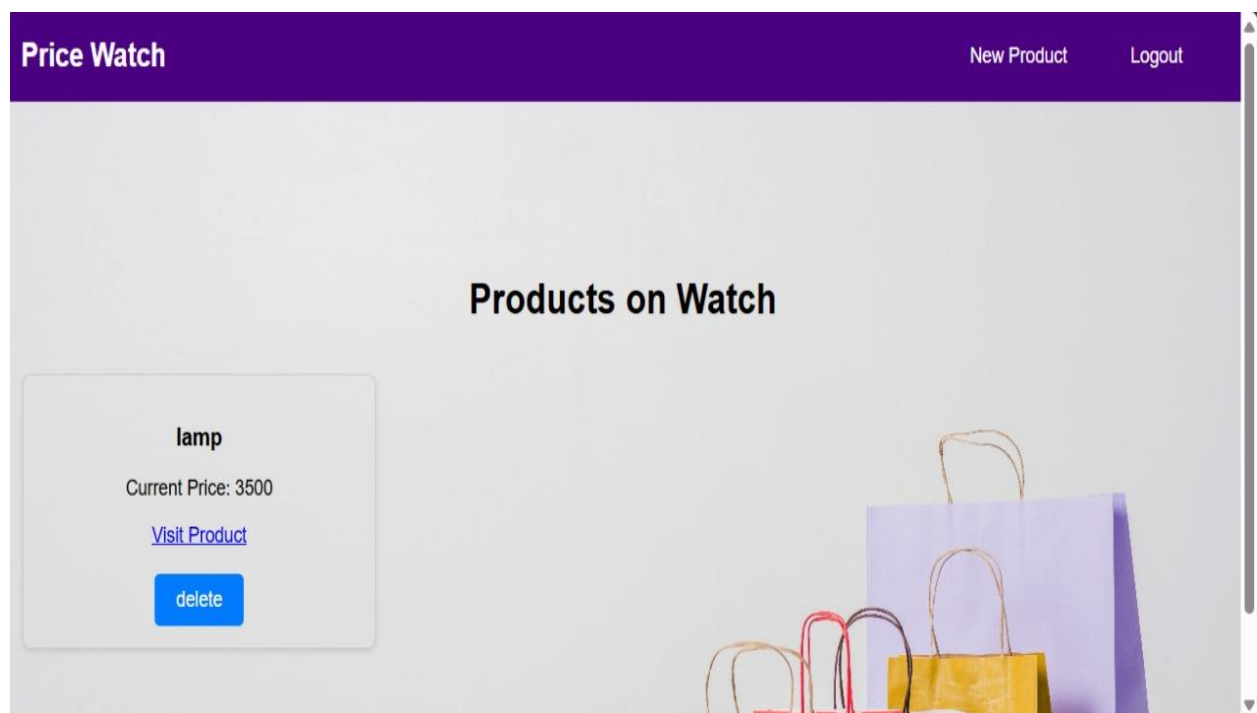


Figure 6.12 After Deleted Product

## 6.13 Registration mail

The mail as shown in the figure 6.13 is the registration mail sent to user for a product.

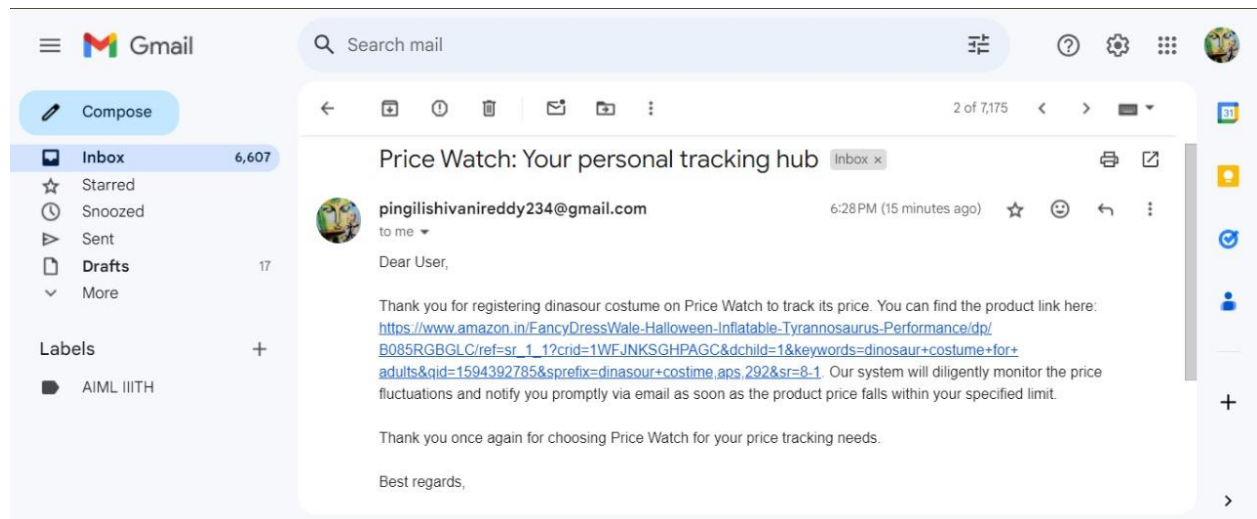


Figure 6.13 Registration mail



## 6.14 Price fall alert

The mail as shown in the figure 6.14 is the price fall alert sent to user for a product.

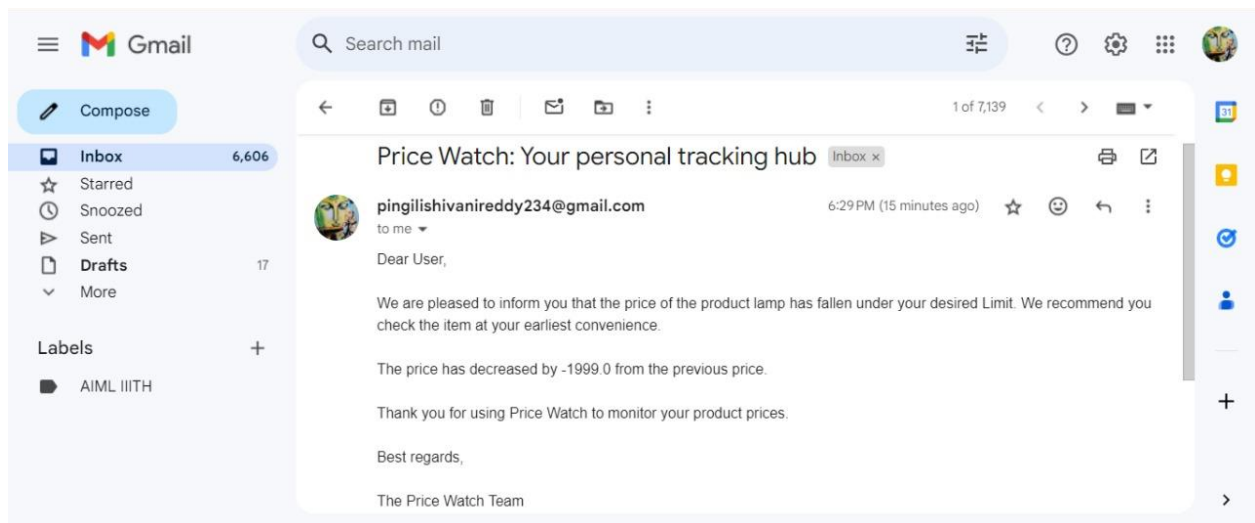


Figure 6.14 price fall alert

## Product data

The Table 6.1 is the product details from the database.

Table 6.1

_id	email	name	url	highprice	price	registration_email_sent	_v
669bb9bfbc5ed245bf9ec08	pingilishivanireddy234@gmail.com	dinasour coust	https://www.amazon.in/FancyDres	5000	5499	TRUE	0
669bba1cbca5ed245bf9ec0b	pingilishivanireddy234@gmail.com	lamp	https://www.amazon.in/sspa/click'	3000	3500	TRUE	0
669bba5ebca5ed245bf9ec0f	vijaya15903@gmail.com	desk	https://www.amazon.in/sspa/click'	3000	6229	TRUE	0
669bba7dbca5ed245bf9ec13	varunn4503@gmail.com	study desk	https://www.amazon.in/sspa/click'	4000	4349	TRUE	0

## Pre registration mail sent

The Table 6.2 is the product details from the database immediately after registering a product and before registration mail being sent.

Table 6.2

_id	email	name	url	highprice	price	registratic	_v
669bb9bfbc5ed245bf9ec08	pingilishivanireddy234@gmail.com	dinasour coustume	https://www.amazon.in/FancyC	5000	5499	TRUE	0
669bba1cbca5ed245bf9ec0b	pingilishivanireddy234@gmail.com	lamp	https://www.amazon.in/sspa/c	3000	3500	TRUE	0
669bba5ebca5ed245bf9ec0f	vijaya15903@gmail.com	desk	https://www.amazon.in/sspa/c	3000	6229	TRUE	0
669bba7dbca5ed245bf9ec13	varunn4503@gmail.com	study desk	https://www.amazon.in/sspa/c	4000	0	FALSE	0

## Login data

The Table 6.3 is the login details from the database.

Table 6.3

_id	email	password	_v
669777ef1bb3cb43b79b0277	pingilishivanireddy234@gmail.com	██████████	0
6697b8388ffe30e61d69cd61	varunn4503@gmail.com	██████████	0
669bb2528797d4b5f50f0bec	vijaya15903@gmail.com	██████████	0

## **Chapter 7**

# **CONCLUSION**

## **7. Conclusion**

### **7.1 Work Carried Out**

Price Watch is an efficient and user-friendly application designed to help users monitor and track price changes for their desired products on E-commerce websites. By allowing users to register products and set target prices, the system periodically checks current prices using web scraping and alerts users via email when the price drops below their specified threshold. This automated price tracking ensures users never miss out on the best deals, making their shopping experience more convenient and cost-effective.

### **7.2 Scope for Future Work**

The future scope for Price Watch includes expanding support to track prices from more retailers, developing a mobile app, incorporating advanced analytics and trend predictions, enhancing notification options with SMS and push notifications, enabling direct purchases through the platform, adding social sharing features, improving web scraping techniques, adding more attractive styling to the pages and leveraging AI and machine learning for personalized recommendations and price trend predictions. These enhancements will make Price Watch more versatile, user-friendly, and indispensable for savvy shoppers.

## REFERENCES

### [1] Quick Start-React

[Quick Start – React](#)

### [2] Installations

<https://www.mongodb.com/try/download/compass>

<https://code.visualstudio.com/download>

### [3] Web Scrapping

Bar-Ilan, J. (2001). Data collection methods on the web for infometric purposes – A review and analysis. *Scientometrics*, 50(1), 7–32.

### [4] UML Diagrams

<https://www.uml-diagrams.org>