

# **BHAO.PK PRICE COMPARISON ENGINE V1.1**

MUHAMMAD AHAD ALI KHAN

SHERYAR KARIM

MUHAMMAD HARIS JAMALI



**Fall-2025**

**Department of Computer Science**

**Capital University of Science & Technology, Islamabad**

Submission Form for Final-Year

# PROJECT REPORT

<b>Version</b>	V 1.0
----------------	-------

<b>TITLE</b>	Bhao.pk Price Comparison Engine
--------------	---------------------------------

<b>SUPERVISOR NAME</b>	DR. Najam Aziz
------------------------	----------------

MEMBER NAME	REG. NO.	EMAIL ADDRESS
Muhammad Ahad Ali Khan	BCS213129	alik123kh@gmail.com
Sheryar Karim	BCS221074	sheryarkarim65@gmail.com
Muhammad Haris Jamali	BCS221027	harisshaheen09@gmail.com

---

---

---

**Supervisor's Signature**

# APPROVAL CERTIFICATE

This project, titled as “Bhao.pk Price Comparison Engine” has been approved  
for the award of

## Bachelors of Science in Computer Science

### Committee Signatures:

Supervisor:

---

(Dr. Najam Aziz)

Project Coordinator:

---

(Mr. Ibtisam Zia)

Head of Department:

---

(Dr. Masroor Ahmed)

## DECLARATION

I/We, hereby, declare that “No portion of the work referred to, in this project has been submitted in support of an application for another degree or qualification of this or any other university/institute or other institution of learning”. It is further declared that this undergraduate project, neither as a whole nor as a part thereof has been copied out from any sources, wherever references have been provided.

### MEMBERS' SIGNATURES

---

---

---

# ACKNOWLEDGEMENTS

# Executive Summary

## Table of Contents

Declaration.....	4
Acknowledgements.....	5
Executive Summary.....	6
 <b>Chapter 1.....</b>	<b>10</b>
Introduction.....	10
1.1. Project Introduction.....	10
1.2..... Existing Examples / Solutions .....	11
1.3 ..... Business Scope .....	13
1.3.1 The Intended Audience.....	13
1.3.2 Use in Commercial Settings.....	13
1.3.3 Models of Revenue.....	13
1.4. Useful Tools and Technologies.....	14
1.5..... Project Work Break Down .....	15
1.6. Project Time Line.....	15
 <b>Chapter 2.....</b>	<b>17</b>
2.1..... Functional Requirements .....	17
2.2..... Non-Functional Requirements .....	19
2.3. Selected Functional Requirements.....	20
2.4. System Use Case Modeling.....	22
2.6. Domain Model.....	42
 <b>Chapter 3.....</b>	<b>43</b>
System Design.....	43

3.1. Layer Definition.....	<b>43</b>
3.1.1. Presentation Layer: .....	43
3.1.2. Business Logic Layer: .....	44
3.1.3. Database Layer: .....	44
3.2. Software Architecture.....	<b>44</b>
3.3. Class Diagram.....	<b>45</b>
3.4. Sequence Diagram .....	<b>48</b>
3.4.1. User .....	48

## List of Tables

i: Table 1.1: Comparison with Existing Examples .....	12
ii: Table 2.1: Functional Requirements.....	17
iii: Table 2.2: Non-Functional Requirement.....	19
iv: Table 2.3: Selected Functional Requirement.....	20
iii: Figure 2.1: Use case Diagram    v: Table 2.4: Search Product .....	22
vi: Table 2.5: Set Price Alert.....	23
vii: Table 2.5: Set Price Alert.....	24
viii: Table 2.7: View Product Details.....	25
ix: Table 2.8: Filter & Sort Results.....	27
x: Table 2.9: View Price History .....	28
xi: Table 2.10: Sign Up.....	29
xii: Table 2.11: Login.....	30
xiii: Table 2.12: View Recommendations .....	31
xiv: Table 2.13: Manage Wishlist .....	32
xv: Table 2.14: View Dashboard Stats.....	34

## List of Figures

i: Figure 1.1: Project Work Break down.....	17
ii: Figure 1.2: Project Timeline.....	18
iii: Figure 2.1: Use case Diagram    v: Table 2.4: Search Product.....	23
iv: Figure 2.2: SSD Search Products SSD.....	37
v: Figure 2.3: SSD Filter and Sort SSD.....	38
vi: Figure 2.4: View Details SSD.....	38
vii: Figure 2.5: SSD View Price History SSD.....	39
viii: Figure 2.6: Set Price Alert SSD.....	39
ix: Figure 2.7: Product Recommendation SSD.....	40
x: Figure2.8: Manage Wishlist SSD.....	40
xi: Figure 2.9: Sign Up SSD.....	41
xii: Figure 2.10: Login SSD.....	41
xiii: Figure 2.11: Admin Stats SSD.....	42
xiv: Figure 2.19: Domain Model.....	43
xv: Table 3.1: Layers Definition.....	44
xvi: Figure 3.1: Software Architecture Diagram.....	46
xvii: Figure 3.2: Class Diagram.....	47
xviii: Figure 3.2: Search Products Sequence Diagram.....	49
xix: Figure 3.3: Filter & Sort Sequence Diagram.....	50
xx: Figure 3.4: View Details Sequence Diagram.....	51
xxi: Figure 3.5: Price History Sequence Diagram.....	52
xxii: Figure 3.6: Set Price Alert Sequence Diagram.....	53
xxiii: Figure 3.7: Show Recommendations Sequence Diagram.....	54
xxiv: Figure 3.8: Manage Wishlist Sequence Diagram.....	55
xxv: Figure 3.9: Sign Up Sequence Diagram.....	56
xxvi: Figure 3.10: Login Sequence Diagram.....	56
xxvii: Figure 3.11: Sign Up Sequence Diagram.....	57
xxviii: Figure 3.12: Entity Relations Diagram.....	58
xxix: Figure 3.13: Entity Relations Diagram.....	59
xxx: Figure 3.13: Web Homepage UI.....	60
xxxi: Figure3.14: Mobile Homepage UI.....	60
xxxii: Figure 3.15: Mobile Search UI.....	61

xxxiii: Figure 3.16: Mobile Search UI.....	62
xxxiv: Figure 3.17: Web Product Page UI.....	63
xxxv: Figure 3.18: Mobile Product Page UI.....	63
xxxvi: Figure 3.19: Web Profile Page UI.....	64
xxxvii: Figure 3.20: Mobile Profile Page UI.....	65
xxxviii: Figure 3.21: Web Login Page UI.....	66
xxxix: Figure 3.22: Mobile Login Page UI.....	66
xl: Figure 3.23: Web Sign Up Page UI.....	67
xli: Figure 3.24: Mobile Sign Up Page UI.....	68
xlii: Figure 3.25: Web Admin Login Page UI.....	69
xliii: Figure 3.26: Mobile Admin Login Page UI.....	69
xliv: Figure 3.27: Web Admin Login Page UI.....	70
xliv: Figure 3.28: Mobile Admin Login Page UI.....	71

# Chapter 1

## Introduction

This chapter introduces the Bhao.pk project, describing its problem statement, objectives, and the technologies used to implement it. It also presents the scope, significance, and development plan of the proposed system. By the end of this chapter, the reader will understand how the project aims to simplify online shopping decisions in Pakistan through intelligent price comparison and alert mechanisms.

### 1.1. Project Introduction

**Bhao.pk** is an intelligent web and mobile platform that aggregates prices of products listed across multiple Pakistani e-commerce sites. It enables users to quickly identify the lowest price for a specific product and receive instant alerts when prices drop below a set threshold. The system collects product and pricing data through automated web scrapers, cleans and normalizes the data, and then displays results on an interactive dashboard.

The project's core purpose is to give users transparent access to price information that is otherwise scattered across different retailers. A single search query such as “70W USB-C Charger” will return aggregated listings from major online stores like **Daraz**, **Shophive**, **Mega.pk**, **PriceOye**, and **BTech**. The backend clusters identical or similar products using fuzzy text matching to eliminate duplicates and inconsistencies.

The platform will also maintain a historical record of price changes, allowing users to visualize price trends over time and set up alerts when prices fall below their target value. The entire system is designed to be cross-platform: the same codebase will generate both a responsive web application and a mobile app.

## 1.2. Existing Examples / Solutions

Existing global price comparison systems show that price aggregation is highly valuable for consumers, but none are specialized for Pakistan's fragmented e-commerce ecosystem.



**Google Shopping** aggregates product listings from thousands of retailers worldwide, offering side-by-side comparisons and filtering by category or brand. However, it depends on merchant data feeds and lacks real-time integration with local Pakistani stores. Its algorithms are optimized for structured APIs, which limits adaptability to dynamic regional websites.



**Idealo**, one of Germany's leading price comparison engines, provides detailed price histories, vendor ratings, and automated alerts when prices change. It uses a mix of API and web-scraping methods but operates exclusively in European markets. While technically advanced, it cannot process PKR pricing or local vendor structures.



**Local Attempts in Pakistan** include smaller startups and online deal-sharing communities that post offers manually on social media platforms like Facebook or WhatsApp. These efforts lack automation, consistent data collection, and historical tracking. As a result, users must still visit multiple sites individually to verify prices and authenticity.

*i: Table 1.1: Comparison with Existing Examples*

<b>System/ Platform</b>	<b>Core Functionality</b>	<b>Data Source</b>	<b>Limitations</b>	<b>Unique Value of Bhao.pk</b>
Google Shopping	Aggregates global product listings, enabling real-time price and availability comparison across thousands of retailers.	Merchant-supplied APIs and global data feeds.	No integration with Pakistani vendors; does not support PKR-based pricing.	Serves as a global benchmark for automated product aggregation and intelligent ranking.
Idealo (Germany)	Provides price tracking, vendor ratings, and discount alerts across European retailers.	Retailer APIs and automated web crawlers across EU marketplaces.	Limited to European markets; cannot process PKR or local Pakistani vendors.	Demonstrates how historical price tracking builds trust and empowers consumers.
Local Attempts (Pakistan)	Manual deal sharing through Facebook, WhatsApp, and Telegram communities.	User-posted, unstructured community data.	No automation, centralized dashboard, or consistent price tracking.	Highlights the need for a unified, automated, Pakistan-specific price comparison ecosystem like Bhao.pk.

## 1.3 Business Scope

**Bhao.pk** has strong market potential in Pakistan, where online shopping continues to grow but lacks transparent price comparison tools:

### 1.3.1 The Intended Audience

- **Everyday Online Shoppers:** Looking for lowest prices without visiting multiple websites.
- **Students and Budget Buyers:** Seeking affordable tech, accessories, or daily items.
- **E-commerce Analysts:** Can study price trends and market volatility.
- **Retailers:** Can analyze competitor pricing dynamically.

### 1.3.2 Use in Commercial Settings

- Integration into affiliate marketing networks.
- Tools for businesses to track competitor prices and adjust strategies.
- Integration with fintech cashback or coupon systems.
- Academic use in data analytics and information retrieval projects.

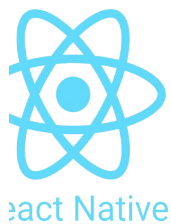
### 1.3.3 Models of Revenue

- **Affiliate Commissions:** Redirect users to retailers through tracked links.
- **Ad-Based Revenue:** Sponsored listings or priority placement for vendors.
- **Subscription Plans:** Users pay for extended alerts and historical data exports.

- **API Access:** Businesses pay to access aggregated pricing data.

## 1.4. Useful Tools and Technologies

The development of **Bhao.pk** relies on a combination of web technologies and data-processing frameworks to ensure smooth cross-platform performance and efficient information retrieval.



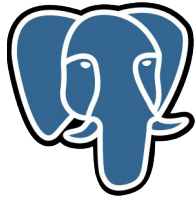
The system is built using **React with React Native (Expo)**, which allows both the web dashboard and the mobile application to share the same codebase. This framework ensures responsive interfaces, real-time updates, and reduced development effort compared to maintaining separate apps.



For backend services, **Node.js with Express** provides a lightweight, scalable environment capable of handling multiple concurrent API requests efficiently. It manages data flow between the scrapers, database, and user interface, forming the communication core of the system.



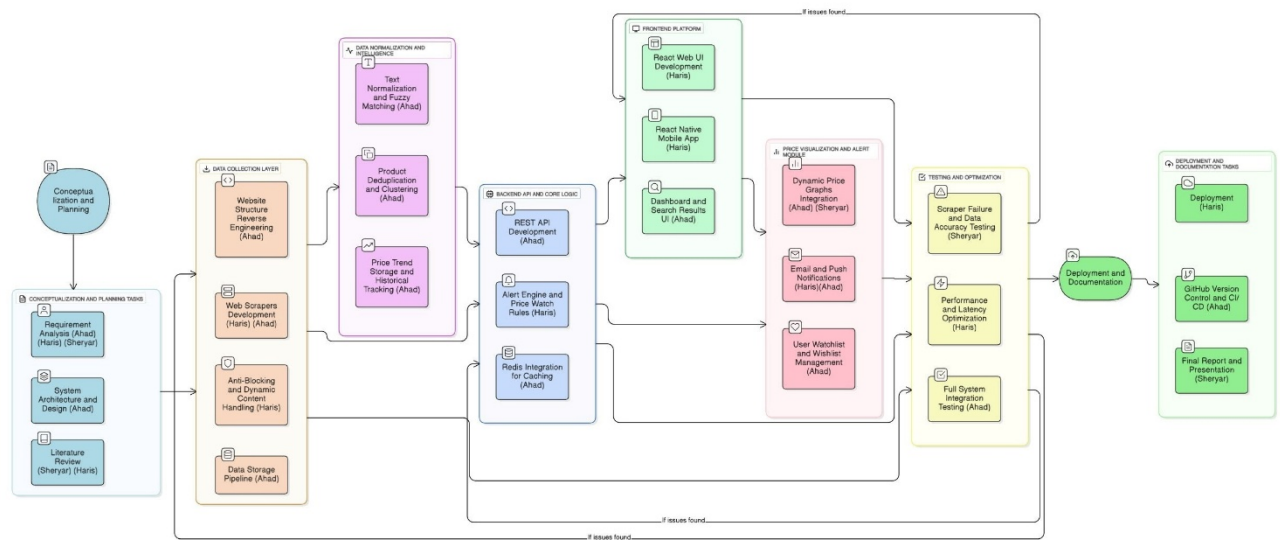
Automated data collection is powered by **Scrapy and BeautifulSoup**, which extract structured product and pricing data from e-commerce websites. These libraries handle HTML parsing, data cleaning, and link traversal, enabling continuous updates without manual input.



The project stores and manages information using **PostgreSQL**, an open-source relational database that supports large-scale data queries, indexing, and time-series analysis. It ensures reliability and fast retrieval for price tracking, user alerts, and historical analytics.

## 1.5. Project Work Break Down

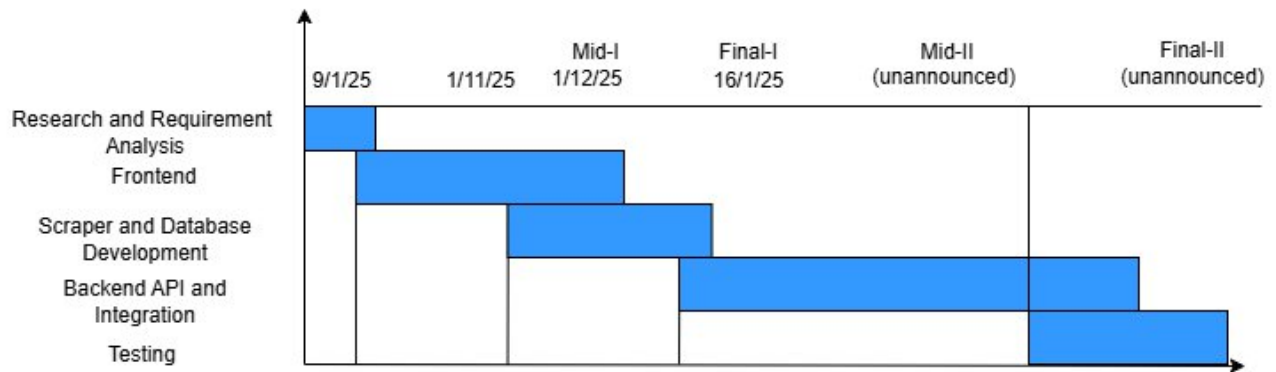
The project breakdown structure is shown in Figure 4



i: Figure 1.1: Project Work Break down

## 1.6. Project Time Line

The proposed project timeline is shown in the following the figure.



ii: Figure 1.2: Project Timeline

## Chapter 2

### Requirement Specification and Analysis

Requirements analysis is a process of determining user expectations for a new or modified product. These features, called requirements, must be quantifiable, relevant and detailed. In software engineering, such requirements are often called functional specifications. In Chapter 2 we will enlist the functional and non-functional requirements and model functional requirements in the form of use case model.

#### 2.1. Functional Requirements

A functional requirement defines a function of a system or its component. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish.

ii: Table 2.1: Functional Requirements

S. No.	Functional Requirement	Type	Status
--------	------------------------	------	--------

1	The user shall view product listings retrieved from multiple supported e-commerce platforms.	Core	Pending
2	The user shall view a section of "Trending Products" highlighted on the homepage.	Core	Pending
3	The user shall view real-time search suggestions while typing in the search bar.	Core	Pending
4	The user shall view a list of their recently viewed products for quick access.	Core	Pending
5	The user shall view a single unified product page containing prices from multiple vendors.	Intermediate	Pending
6	The user shall search for a product using keywords in the search bar.	Core	pending
7	The user shall view aggregated search results from multiple stores.	Core	Pending
8	The user shall sort search results by Price (Low to High).	Core	Pending

9	The user shall filter results by specific Store.	Core	Pending
10	The user shall filter results by Price Range (Min/Max).	Core	Pending
11	The user shall view detailed product information on a dedicated page.	Core	Pending
12	The user shall click "Open in Vendor Site" to navigate to the original vendor's website.	Core	Pending
13	The user shall create a new account (Sign up) using email and password.	Core	Pending
14	The user shall log into the system using valid credentials.	Core	Pending
15	The user shall log out of the system.	Core	Pending
16	The user shall recover a forgotten password via email.	Core	Pending

17	The user shall set a specific target price alert for a product.	Core	Pending
18	The user shall receive an email notification when a price drops to or below the target.	Core	Pending
19	The user shall receive a push notification on the mobile app for price drops.	Core	Pending
20	The user shall view a graphical history of price changes over time.	Core	Pending
21	The user shall add products to a personal "Wishlist".	Intermediate	Pending
22	The user shall remove items from their "Wishlist".	Intermediate	Pending
23	The user shall view updated prices refreshed at scheduled intervals.	Advanced	Pending
24	Admin shall log into the backend dashboard.	Backend	Pending
25	Admin shall view statistics on total products scraped and active users.	Intermediate	Pending
26	The user shall identify the best deals via automatically assigned "Best Value" badges.	Intermediate	Pending
27	The user shall view personalized product recommendations based on their browsing history.	Advanced	Pending

## 2.2. Non-Functional Requirements

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. They are contrasted with functional requirements that define specific behaviors or functions.

*iii: Table 2.2: Non-Functional Requirement*

S. No.	Non-Functional Requirements	Category
1	The system interface shall be user-friendly and easy to navigate for new users.	Usability

2	The system shall ensure that user passwords and personal data are stored securely.	Security
3	The system shall provide helpful messages and feedback to user enters input.	Usability
4	The mobile application shall run smoothly on standard mobile devices.	Performance
5	The design shall use consistent fonts, colors, and icons throughout the application.	Usability
6	The website shall be responsive.	Usability

## 2.3. Selected Functional Requirements

Following is the list of the requirements selected for the current iteration.

*iv: Table 2.3: Selected Functional Requirement*

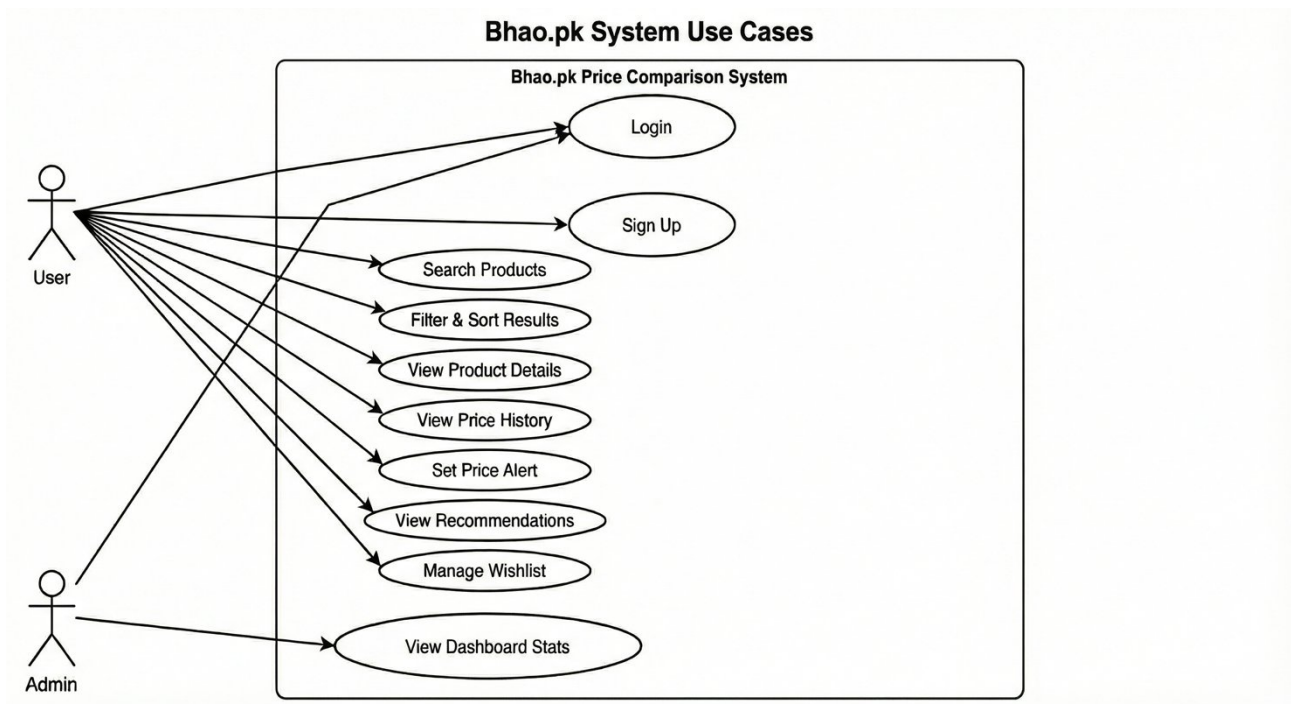
S. No.	Functional Requirement	Type
1	The user shall view product listings retrieved from multiple supported e-commerce platforms.	New
2	The user shall view a section of "Trending Products" highlighted on the homepage.	New
3	The user shall view real-time search suggestions while typing in the search bar.	New
4	The user shall view a list of their recently viewed products for quick access.	New
5	The user shall sort search results by Price (Low to High).	New

6	The user shall filter the results by specific Store.	New
7	The user shall filter the results by Price Range (Min/Max).	New
8	The user shall create a new account (Sign up) using email and password.	New
9	The user shall set a specific target price alert for a product.	New
10	The user shall log into the system using valid credentials.	New
11	The user shall log out of the system.	New
12	The user shall recover a forgotten password via email.	New
13	The user shall receive an email notification when a price drops to or below the target.	New
14	The user shall receive a push notification on the mobile app for price drops.	New
15	The user shall view a graphical history of price changes over time.	New
16	The user shall add products to a personal "Wishlist".	New
17	The user shall remove products from their "Wishlist".	
18	The user shall identify the best deals via automatically assigned "Best Value" badges.	New
19	The user shall view personalized product recommendations based on their browsing history.	New

20	Admin shall log into the backend dashboard.	New
21	Admin shall view statistics on total products scraped and active users.	New

## 2.4. System Use Case Modeling

A use case is a list of actions or event steps, typically defining the interactions between a role (known in the Unified Modeling Language as an actor) and a system, to achieve a goal. The actor can be a human or other external system. Customer's use cases are shown in the following the figure.



iii: Figure 2.1: Use case Diagram

v: Table 2.4: Search Product

<b>Use Case ID:</b>	Uc1
---------------------	-----

Use Case Name:	Search Product		
Created By:	Muhammad Ahad Ali Khan	Last Updated By:	Muhammad Haris Jamali
Date Created:	15/11/2025	Last Revision Date:	16/11/2025
Actors:	User (Guest or Registered)		
Description:	The user searches for a specific item to compare prices across stores.		
Trigger:	Search Button		
Preconditions:	The scrapper must be functional or the system must have scraped data available in the database.		
Postconditions:	A list of relevant products with prices is displayed.		
Normal Flow:	Customer	System	
	1: User enters keywords in search bar.	System queries the database using fuzzy matching.	
	2: User clicks search.	System displays aggregated results.	
Alternative Flows:	User applies filters (price/store) to the results.		
Exceptions:	1. No products found matching the query.		
	2. Database connection error.		

**vi: Table 2.5: Set Price Alert**

Use Case ID:	Uc2		
Use Case Name:	Set Price Alert		
Created By:	Muhammad Ahad Ali Khan	Last Updated By:	Muhammad Haris Jamali
Date Created:	15/11/2025	Last Revision Date:	16/11/2025
Actors:	Registered User		
Description:	User sets a target price to receive notifications when the product becomes cheaper.		
Trigger:	"Set Alert" Button		
Preconditions:	User must be logged in.		
Postconditions:	An alert record is created in the database.		
Normal Flow:	Customer	System	
	1: User views product details.	System verifies value is lower than current price.	
	2: User clicks "Set Alert".	System saves the alert preference.	
	3: User enters target price.		
Alternative Flows:	User hits back or cancel.		

<b>Exceptions:</b>	1. Target price is higher than the current price (System displays warning).
--------------------	---

**vii: Table 2.5: Set Price Alert**

<b>Use Case ID:</b>	Uc3		
<b>Use Case Name:</b>	View Search Results		
<b>Created By:</b>	Muhammad Ahad Ali Khan	<b>Last Updated By:</b>	Muhammad Haris Jamali
<b>Date Created:</b>	15/11/2025	<b>Last Revision Date:</b>	16/11/2025
<b>Actors:</b>	User (Guest), Registered User		
<b>Description:</b>	The system presents a list of products that match the user's search query, showing key details like title, image, and price range.		
<b>Trigger:</b>	Successful completion of the "Search Products" use case.		
<b>Preconditions:</b>	A search query has been executed.		
<b>Postconditions:</b>	A curated list of product cards is presented to the user.		
<b>Normal Flow:</b>	<b>Customer</b>	<b>System</b>	
	1: User views search page.	System receives the list of matching products from the database.	
		System aggregates listings from different stores for the same product.	

		System displays products in a grid layout, showing image, title, and "Starts from Rs. X".
<b>Alternative Flows:</b>	1. The number of results exceeds one page limit (e.g., >20 items). 2. System displays pagination controls (e.g., "Next", "Page 2"). 3. User clicks "Next". 4. System fetches and displays the next set of results.	
<b>Exceptions:</b>	1. No results: System displays "No products found."	

*viii: Table 2.7: View Product Details*

<b>Use Case ID:</b>	Uc4		
<b>Use Case Name:</b>	View Product Details		
<b>Created By:</b>	Muhammad Ahad Ali Khan	<b>Last Updated By:</b>	Sheryar Karim
<b>Date Created:</b>	15/11/2025	<b>Last Revision Date:</b>	16/11/2025
<b>Actors:</b>	User (Guest), Registered User		
<b>Description:</b>	The user views full details and a price comparison table for a product.		
<b>Trigger:</b>	User clicks a product card.		
<b>Preconditions:</b>	Product ID exists.		
<b>Postconditions:</b>	Product Detail Page is displayed.		
<b>Normal Flow:</b>	Customer	System	

	1: User clicks a product.	System receives the list of matching products from the database.
	2: User sees product details, store listings, and history.	System aggregates listings from different stores for the same product.
	3: User can proceed to other actions.	System displays products in a grid layout, showing image, title, and "Starts from Rs. X".
<b>Alternative Flows:</b>	User decides to go to some other page from navigation.	
<b>Exceptions:</b>	1. Product not found: Redirect to 404 page.	

**ix: Table 2.8: Filter & Sort Results**

<b>Use Case ID:</b>	Uc5		
<b>Use Case Name:</b>	Filter & Sort Results		
<b>Created By:</b>	Muhammad Ahad Ali Khan	<b>Last Updated By:</b>	Muhammad Haris Jamali
<b>Date Created:</b>	15/11/2025	<b>Last Revision Date:</b>	16/11/2025
<b>Actors:</b>	User (Guest), Registered User		
<b>Description:</b>	The user applies criteria (price range, store) or changes the order (low-to-high) to narrow down search results.		
<b>Trigger:</b>	User interacts with filter sidebar or sort dropdown.		

<b>Preconditions:</b>	Search results must be currently displayed.	
<b>Postconditions:</b>	The product list is updated to reflect specific criteria.	
<b>Normal Flow:</b>	Customer	System
	1. User selects a filter (e.g., Store: Daraz) or sort order.	System filters the current list.
		System filters the current list.
		System refreshes the grid with updated results.
		System refreshes the grid with updated results.
<b>Alternative Flows:</b>	User clicks "Clear All" to revert to the original search results.	
<b>Exceptions:</b>	System displays "No products match your filters."	

**x: Table 2.9: View Price History**

Use Case ID:	Uc6		
Use Case Name:	View Price History		
Created By:	Muhammad Ahad Ali Khan	Last Updated By:	Muhammad Haris Jamali
Date Created:	15/11/2025	Last Revision Date:	16/11/2025
Actors:	User (Guest), Registered User		
Description:	User views a graph of price changes over time.		
Trigger:	User views Product Detail Page.		
Preconditions:	Historical price data exists.		
Postconditions:	Price graph is displayed.		
Normal Flow:	Customer	System	
	1: User loads product page.	System fetches historical data.	
		System renders graph.	
Alternative Flows:	1. User changes graph view from default "30 Days" to "6 Months" via a control. 2. System fetches additional historical data from the database. 3. System re-renders the graph with the new time scale.		
Exceptions:	Insufficient data: System doesn't show the graph and instead says it doesn't have enough historical data to construct a graph.		

*xi: Table 2.10: Sign Up*

Use Case ID:	Uc7		
Use Case Name:	Sign Up		
Created By:	Muhammad Ahad Ali Khan	Last Updated By:	Muhammad Ahad Ali Khan
Date Created:	15/11/2025	Last Revision Date:	21/11/2025
Actors:	User (Guest)		
Description:	New user creates an account.		
Trigger:	User clicks "Sign Up".		
Preconditions:	User email is not already registered.		
Postconditions:	New user account created and logged in.		
Normal Flow:	Customer	System	
	1: User provides email and password.	System validates and checks uniqueness.	
	2:: User clicks create account.	System creates user record.	
		System logs user in.	
Alternative Flows:	User clicks login link from the signup page.		

<b>Exceptions:</b>	Email exists: Show error message.
--------------------	-----------------------------------

**xii: Table 2.11: Login**

Use Case ID:	Uc8		
Use Case Name:	Login		
Created By:	Muhammad Haris Jamali	Last Updated By:	Sheryar Karim
Date Created:	18/11/2025	Last Revision Date:	21/11/2025
Actors:	User (Guest), Admin		
Description:	Existing user authenticates.		
Trigger:	User clicks "Login".		
Preconditions:	User/Admin has an account.		
Postconditions:	User is authenticated as Registered User. Admin is authenticated as Admin		
Normal Flow:	Customer	System	
	1: User/Admin enters email and password.	System verifies credentials.	
	2: User/Admin clicks create account.	System creates session.	
Alternative Flows:	User clicks "Forgot Password?" link on login form.		

<b>Exceptions:</b>	Invalid credentials: Show error message.
--------------------	--

*xiii: Table 2.12: View Recommendations*

<b>Use Case ID:</b>	Uc9		
<b>Use Case Name:</b>	View Recommendations		
<b>Created By:</b>	Muhammad Haris Jamali	<b>Last Updated By:</b>	Muhammad Ahad Ali Khan
<b>Date Created:</b>	18/11/2025	<b>Last Revision Date:</b>	23/11/2025
<b>Actors:</b>	Registered User		
<b>Description:</b>	User sees personalized product suggestions.		
<b>Trigger:</b>	User visits homepage.		
<b>Preconditions:</b>	User is logged in with some activity history.		
<b>Postconditions:</b>	Recommended items are displayed.		
<b>Normal Flow:</b>	Customer	System	
		System identifies user.	
		System retrieves user persona and finds matching products.	

		System displays recommendations.
<b>Alternative Flows:</b>	User clicks a "Show More" button in the recommendations section.	
<b>Exceptions:</b>	No history: System shows popular items.	

*xiv: Table 2.13: Manage Wishlist*

<b>Use Case ID:</b>	Uc10		
<b>Use Case Name:</b>	Manage Wishlist		
<b>Created By:</b>	Sheryar Karim	<b>Last Updated By:</b>	Muhammad Ahad Ali Khan
<b>Date Created:</b>	20/11/2025	<b>Last Revision Date:</b>	24/11/2025
<b>Actors:</b>	Registered User		
<b>Description:</b>	The user saves products to a personal list for future reference or removes them when no longer needed.		
<b>Trigger:</b>	User clicks the "Heart" icon or "Add to Wishlist" button.		
<b>Preconditions:</b>	User is logged in.		
<b>Postconditions:</b>	Product is added to or removed from the user's saved list in the database		
<b>Normal Flow:</b>	Customer	System	

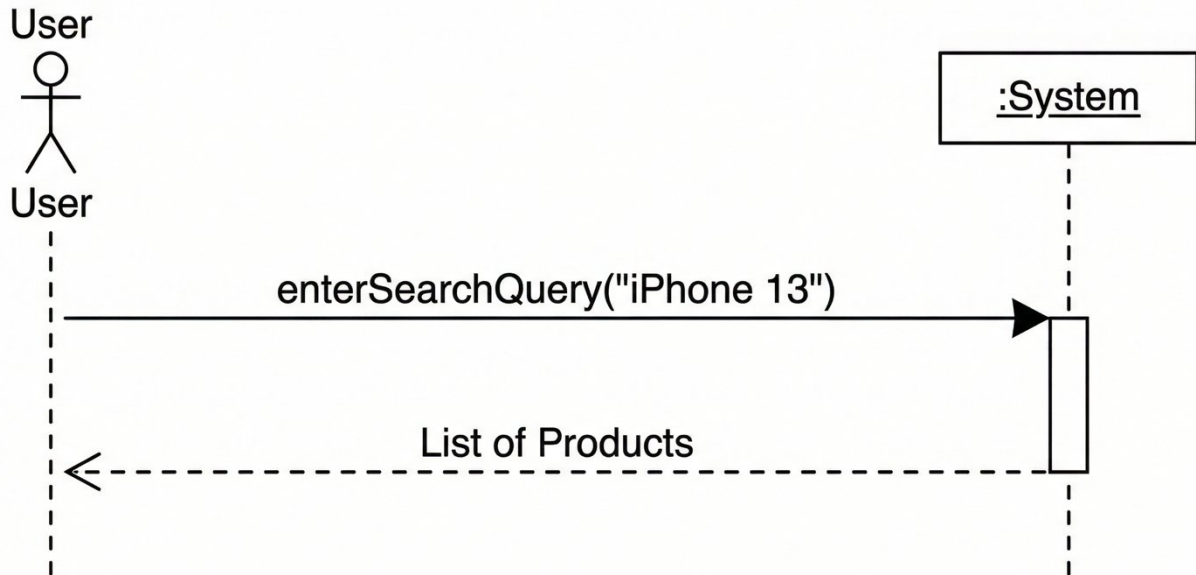
	1: User clicks the "Heart" icon on a product card.	System checks if the item is already in the wishlist.
		System adds the item ID to the user's wishlist record.
		System updates the icon to appear "filled".
<b>Alternative Flows:</b>	If the item is already in the wishlist, clicking the icon removes it.	
<b>Exceptions:</b>	System fails to save the item and displays an error "Try again later".	

**xv: Table 2.14: View Dashboard Stats**

<b>Use Case ID:</b>	Uc11		
<b>Use Case Name:</b>	View Dashboard Stats		
<b>Created By:</b>	Sheryar Karim	<b>Last Updated By:</b>	Muhammad Haris Jamali
<b>Date Created:</b>	20/11/2025	<b>Last Revision Date:</b>	24/11/2025
<b>Actors:</b>	Admin		
<b>Description:</b>	The admin views system health metrics, including total products scraped and active user counts.		
<b>Trigger:</b>	Admin logs into the backend portal.		

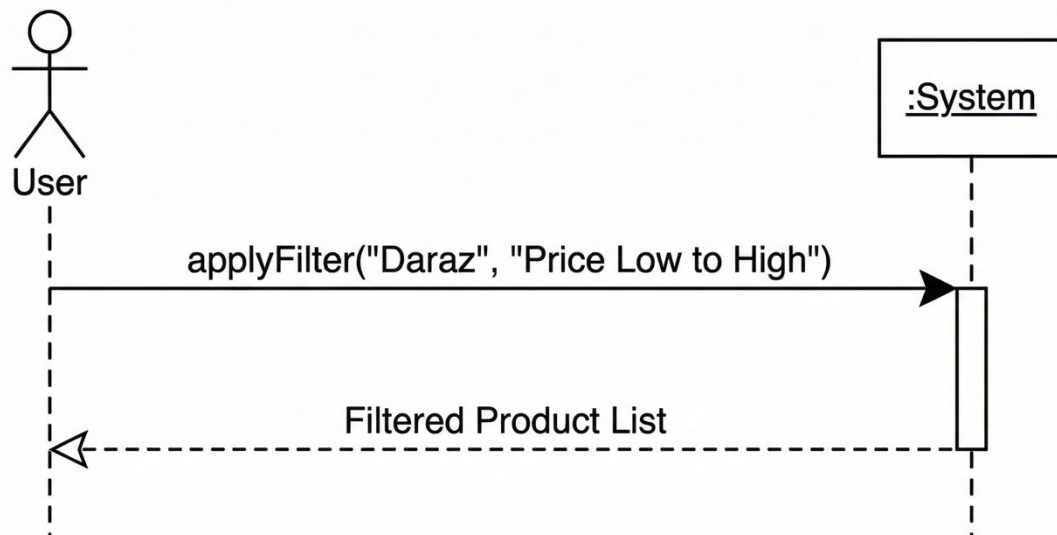
<b>Preconditions:</b>	Admin must be authenticated.	
<b>Postconditions:</b>	Statistical graphs and counters are displayed.	
<b>Normal Flow:</b>	Admin	System
	1: Admin navigates to the main Dashboard tab.	System queries the database for total product count and user logs.
		System renders charts (e.g., Scraper Status, New Signups).
<b>Alternative Flows:</b>	None.	
<b>Exceptions:</b>	System displays 0 values if the database is empty.	

## Bhao.pk - Search Products SSD



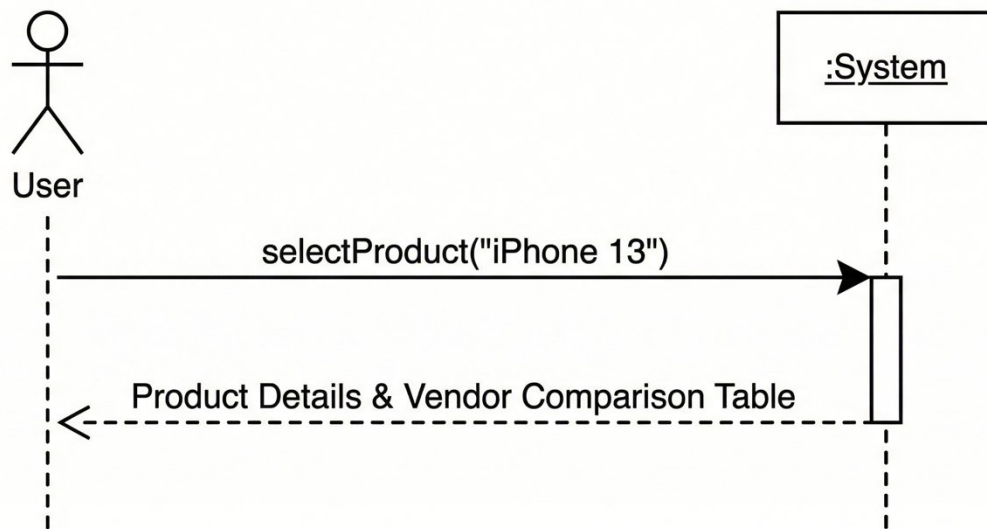
iv: Figure 2.2: SSD Search Products SSD

## Bhao.pk - Filter & Sort SSD



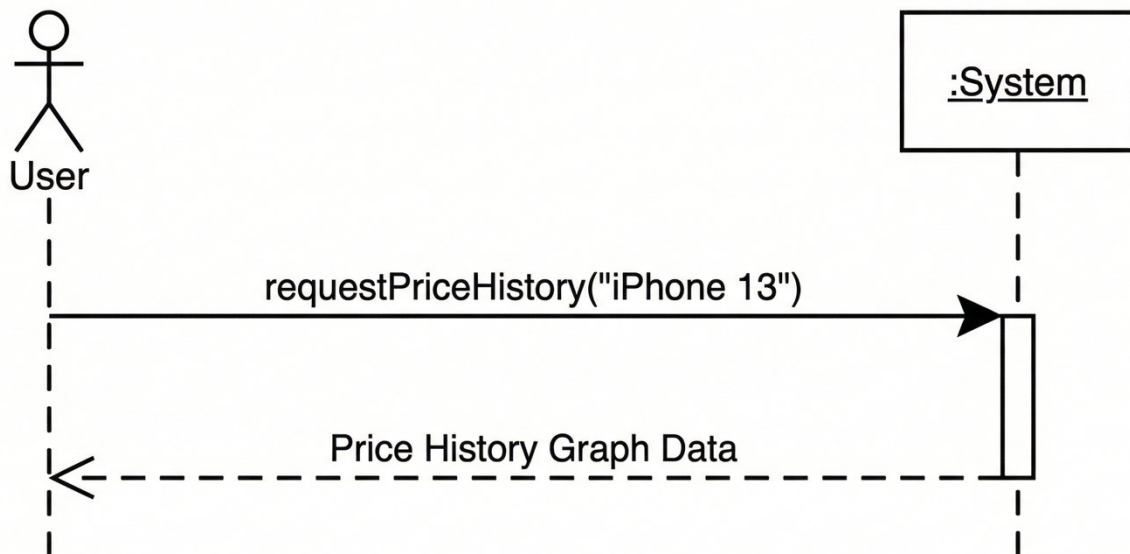
v: Figure 2.3: SSD Filter and Sort SSD

## Bhao.pk - View Details SSD



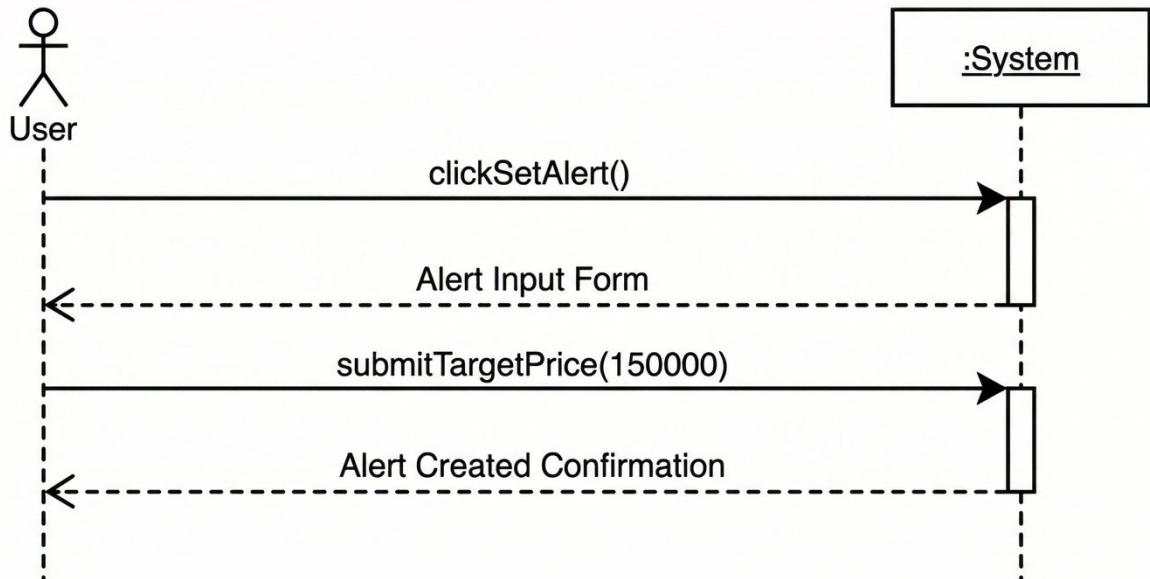
vi: Figure 2.4: View Details SSD

## Bhao.pk - View Price History SSD



vii: Figure 2.5: SSD View Price History SSD

## Bhao.pk - Set Alert SSD



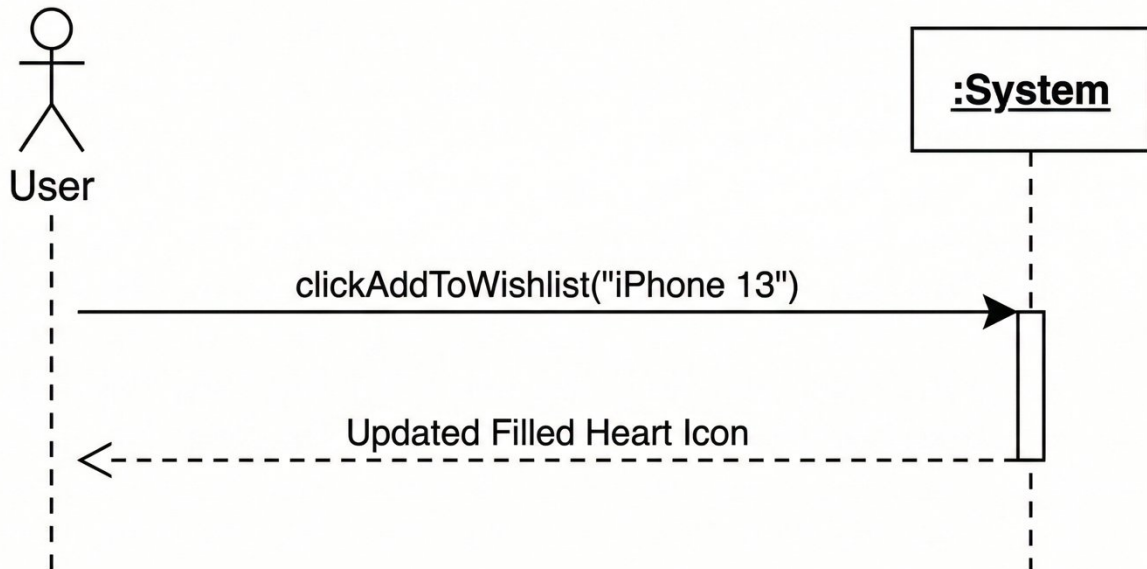
viii: Figure 2.6: Set Price Alert SSD

## Bhao.pk - Recommendations SSD



ix: Figure 2.7: Product Recommendation SSD

## Bhao.pk - Manage Wishlist SSD

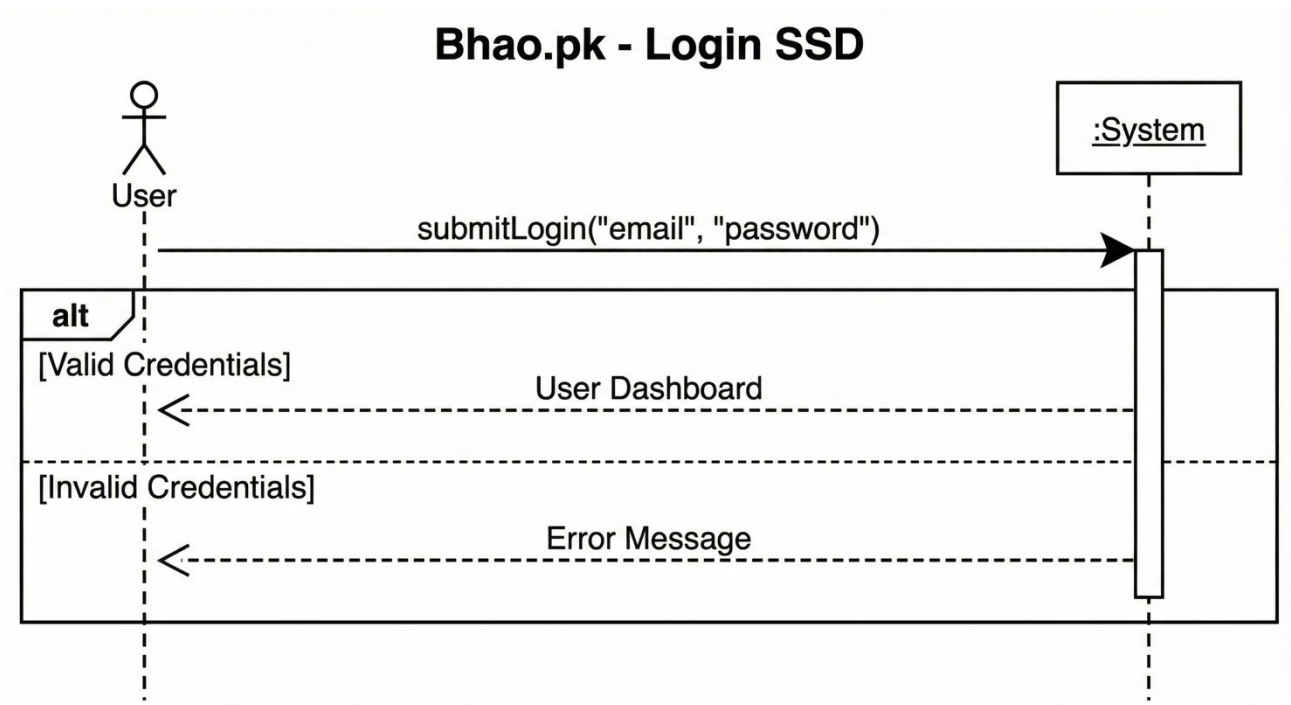


x: Figure2.8: Manage Wishlist SSD

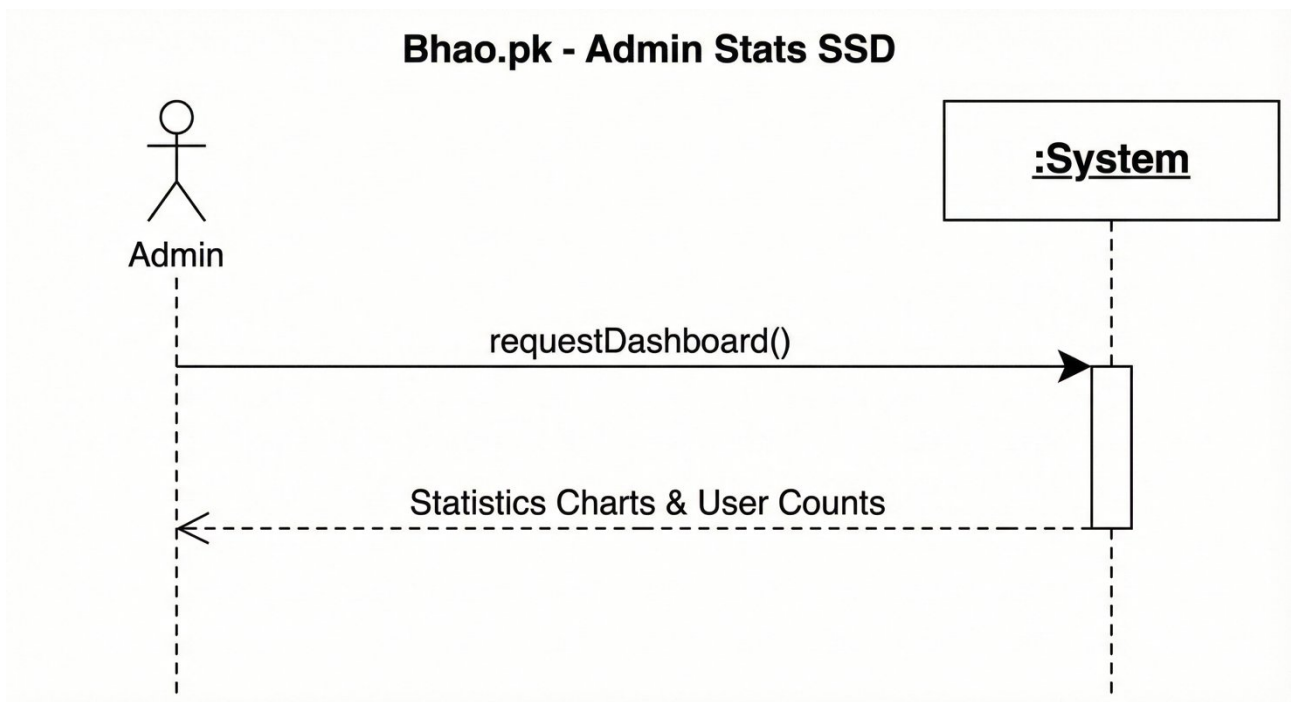
## Bhao.pk - Sign Up SSD



xi: Figure 2.9: Sign Up SSD



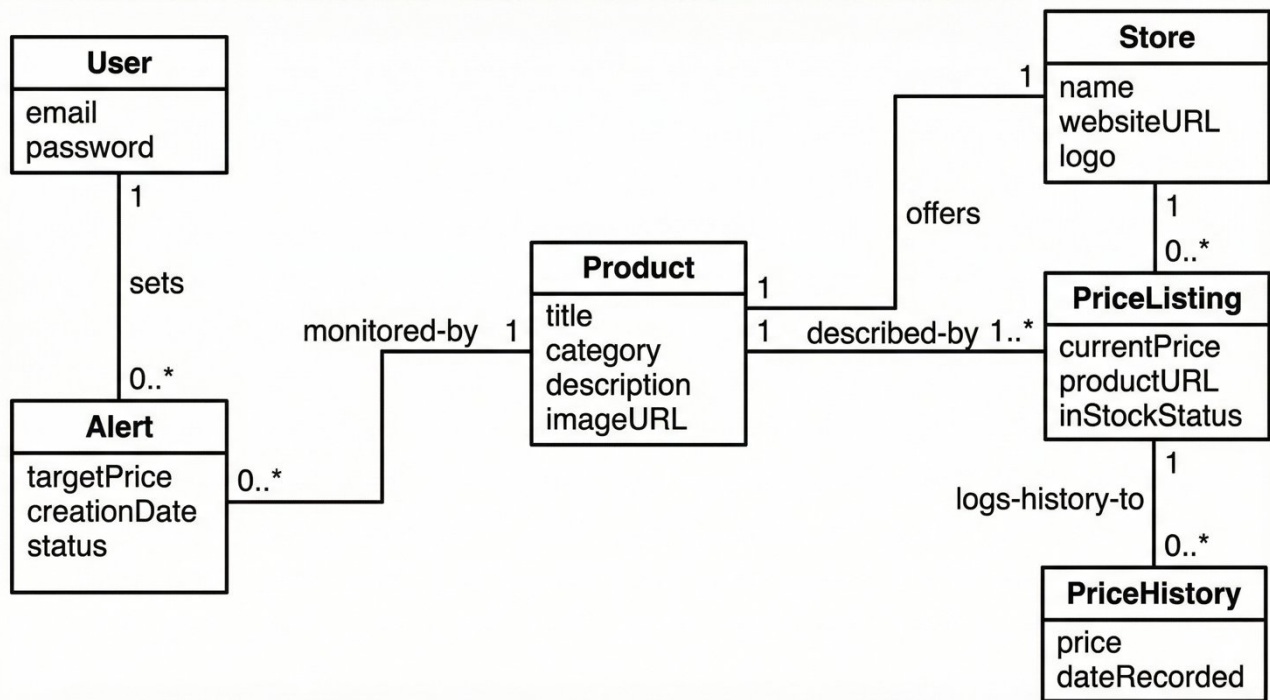
*xii: Figure 2.10: Login SSD*



*xiii: Figure 2.11: Admin Stats SSD*

## 2.6. Domain Model

The Domain Model for Bhao.pk illustrates the conceptual structure of the price comparison ecosystem, centering on the **Product** entity as the core item being tracked. Each generic product is linked to multiple **StoreListings**, representing specific offers from external **Stores** like Daraz or Telemart. **Users** interact with the system by setting **Alerts** on these products to monitor price changes. To maintain up-to-date information, a backend **ScraperBot** creates and updates listings, while a **PriceHistory** entity logs every price fluctuation over time to support historical trend analysis.



xiv: Figure 2.19: Domain Model

## Chapter 3

### System Design

The system design of Bhao.pk is built upon a robust, multi-tiered **Software Architecture** that ensures scalability and clear separation of concerns. It is organized into three primary layers: the **Presentation Layer**, which handles user interactions via a React-based frontend; the **Business Logic Layer**, which serves as the core engine managing APIs, the automated **Scraping Manager**, and the Alert System; and the **Database Layer**, which ensures persistent data storage using PostgreSQL.

### 3.1. Layer Definition

*xv: Table 3.1: Layers Definition*

Layers	Description
Presentation Layer	This layer will be used for the interaction with the user through a graphical user interface.
Business Logic Layer	This layer contains the business logic. All the constraints and majority of the functions reside under this layer.
Database Layer	This layer contains the database of the application being developed.

#### 3.1.1. Presentation Layer:

This layer is responsible for the user interface and user experience. It handles user inputs and displays data retrieved from the server. It includes the web frontend (React.js) and potential future mobile interfaces.

#### 3.1.2. Business Logic Layer:

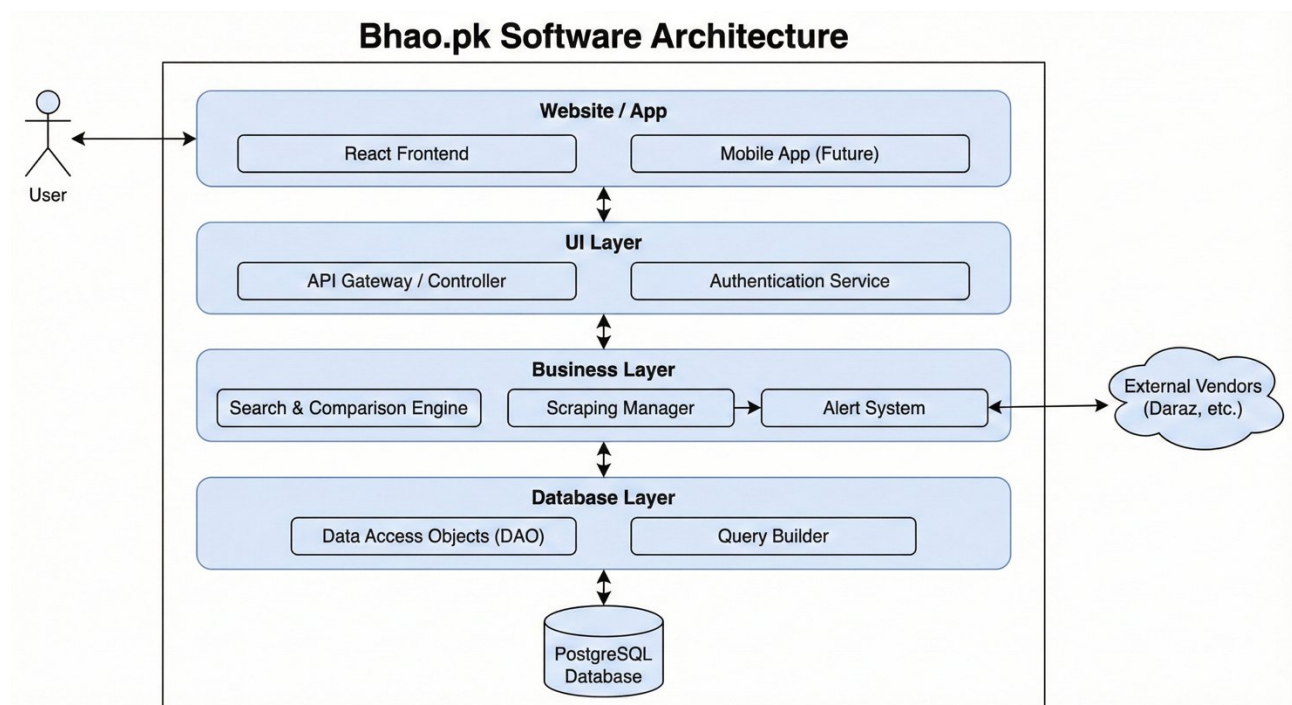
This layer contains the core functionality. It manages data processing, executes search algorithms, handles scraping scheduling, and processes alerts. It serves as the bridge between the UI and the database.

#### 3.1.3. Database Layer:

Database layer includes database servers where information is stored and retrieved. Data in this tier is kept independent of application servers or business logic.

### 3.2. Software Architecture

This layer is responsible for persistent data storage. It manages the relational database (PostgreSQL) holding information on users, products, stores, and historical price records. Below is the architecture diagram of the system:

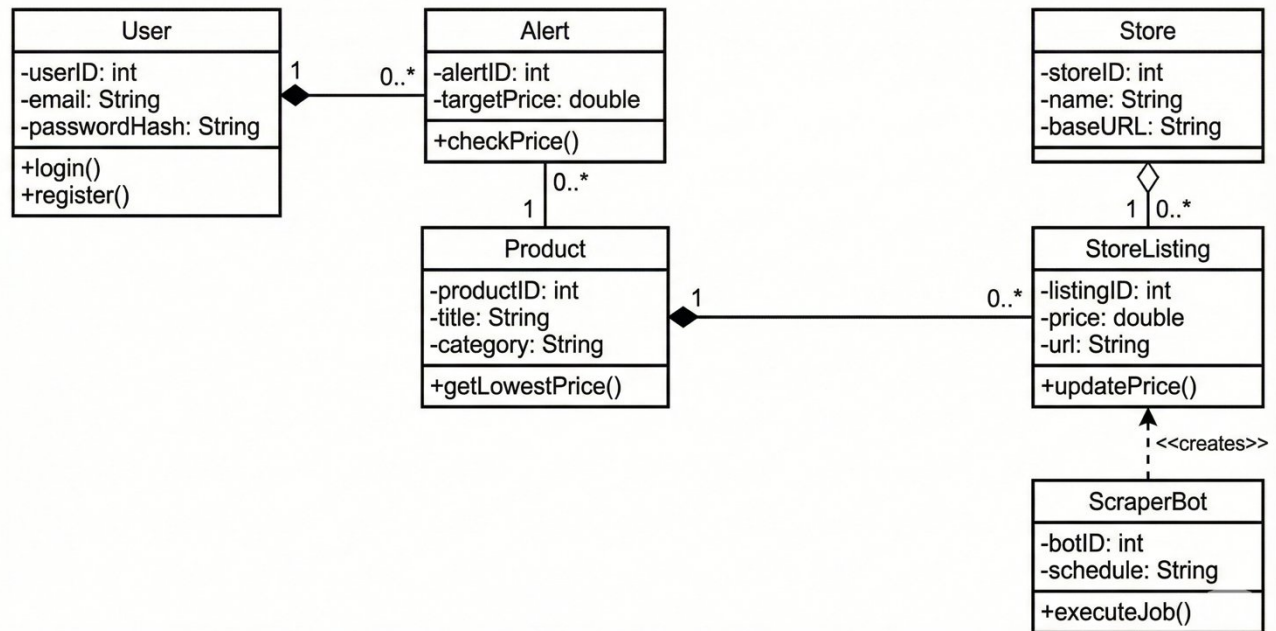


*xvi: Figure 3.1: Software Architecture Diagram*

### 3.3. Class Diagram

The Class Diagram for Bhao.pk defines the system's static structure, centering on the **Product** class which is composed of multiple specific **StoreListings** to represent offers from different vendors. **Stores** act as aggregators for these listings, while **Users** maintain a strict composition relationship with **Alerts**, allowing for personalized price monitoring that depends on the user account's existence. Additionally, the diagram illustrates backend automation through the **ScraperBot** class, which demonstrates a dependency on **StoreListings** to dynamically create and update price data from external sources.

**UML Class Diagram for Class Bhao.pk**



*xvii: Figure 3.2: Class Diagram*

### **User:**

This class represents a user of the Bhao.pk system. It stores essential account information such as `userId`, `email`, `passwordHash`, and their role (e.g., guest or registered). The main responsibilities of this class include managing user authentication via `login()` and creating new accounts via `register()`.

### **Alert:**

This class represents a price notification set by a registered user. It holds data including its unique `alertId`, the user's desired `targetPrice`, the `creationDate`, and a boolean `isActive` status. Its primary operation, `checkPrice()`, compares a product's current price against the target price to determine if a notification should be triggered.

### **Product:**

This is the central entity of the system, representing a generic item that can be sold by multiple vendors. It contains general attributes like productId, title, category, and a baseImage URL. Its key operations include getLowestPrice() to calculate the best available deal and getListings() to retrieve all associated vendor offers.

### **StoreListing:**

This class represents a specific offer for a Product from a single Store. It links the general product to a concrete vendor and contains listing-specific details like listingId, current price, the direct vendorUrl to the purchase page, and a lastUpdated timestamp. It includes an updatePrice() operation to modify its price based on new data.

### **Store:**

This class represents an external vendor or e-commerce website (e.g., Daraz, PriceOye). It stores identity information such as the storeId, the store's name, its baseUrl, and a logoUrl.

### **ScraperBot:**

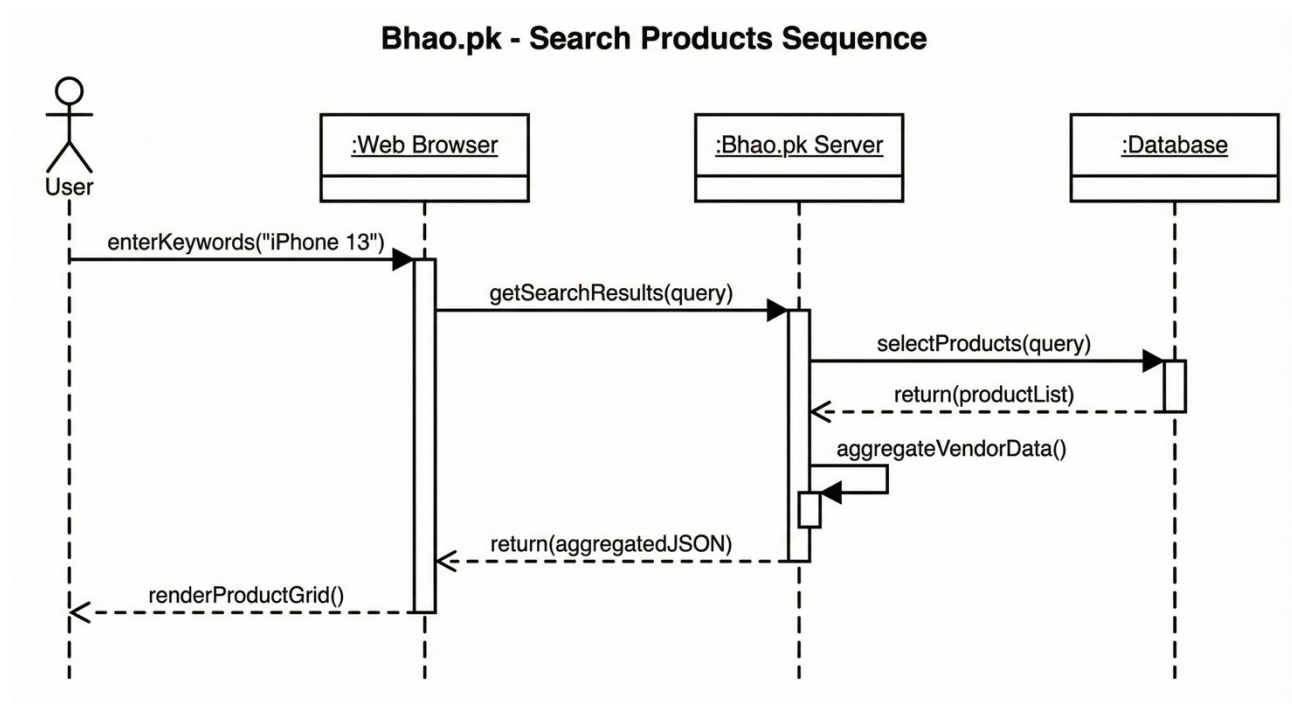
This class is part of the backend system responsible for automated data extraction. It includes attributes for its botId, the targetDomain it is assigned to crawl, and its execution schedule. Its operations include executeJob() to start the crawling process and extractData() to parse product information from raw HTML.

### 3.4. Sequence Diagram

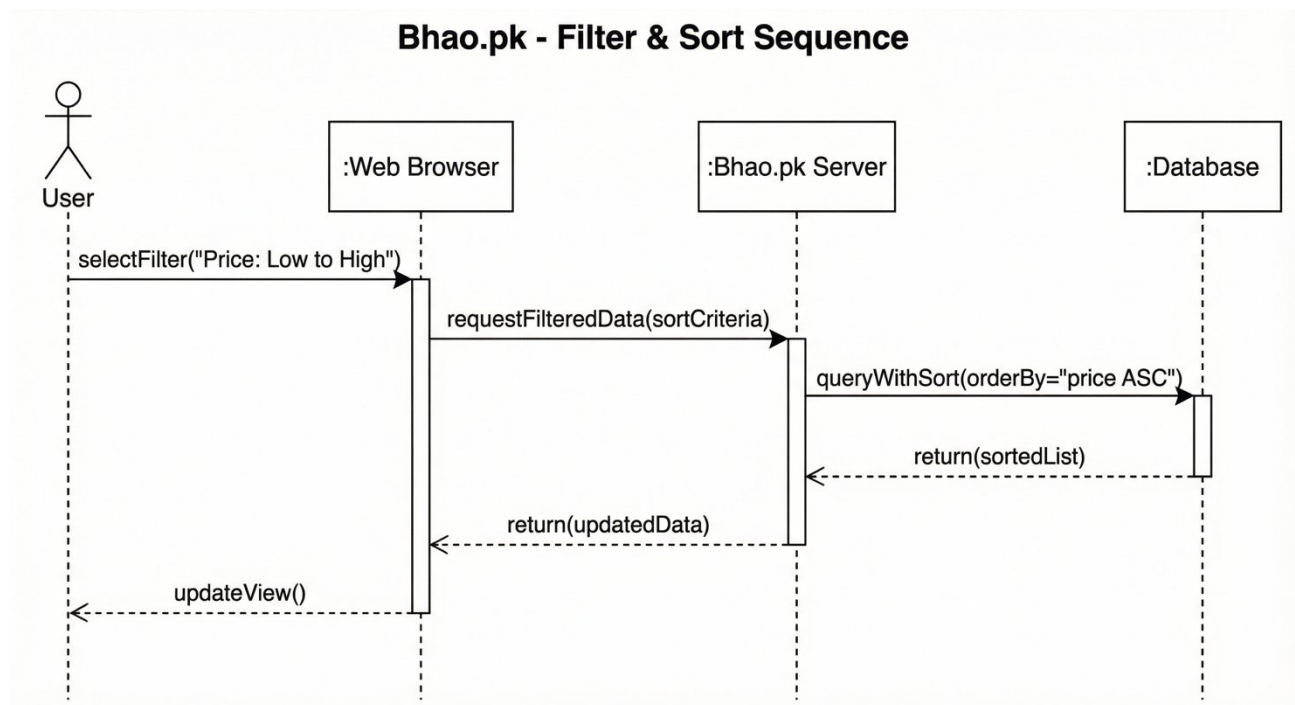
The Sequence Diagrams for Bhao.pk model the dynamic behavior of the system, illustrating the precise step-by-step message flows between the User, Web Browser, Server, and Database. They provide a detailed blueprint of the logic required to execute core functionalities, such as Searching for Products, Viewing Price History, and Setting Alerts, by visualizing how frontend API requests are processed, validated, and responded to by the backend architecture.

#### 3.4.1. User

This Sequence diagram shows the flow of User actions how he interacts with the system and how he performs general tasks like sign up and login.

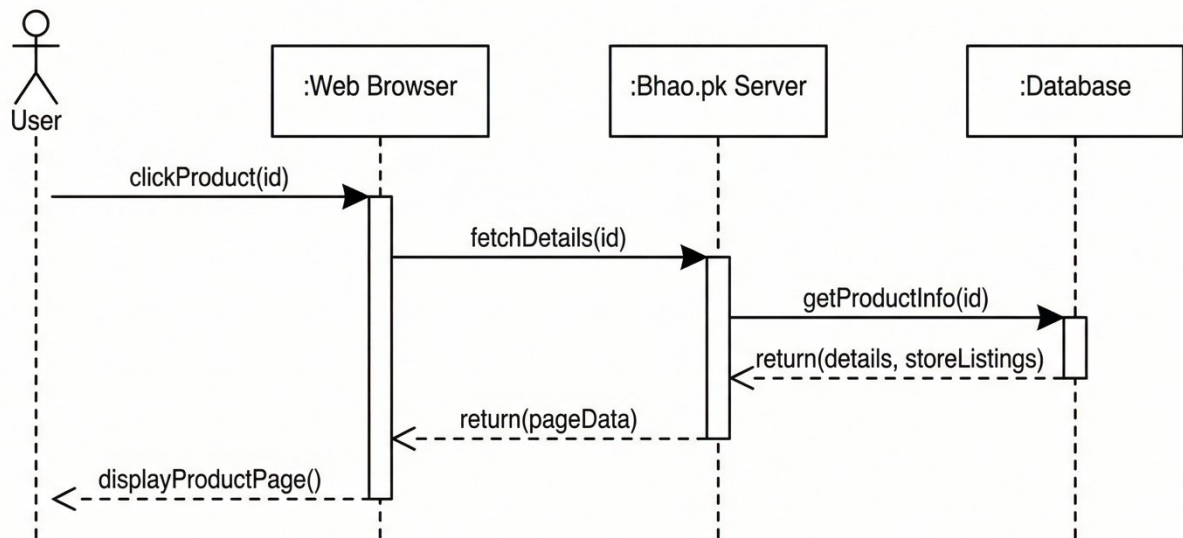


*xviii: Figure 3.2: Search Products Sequence Diagram*

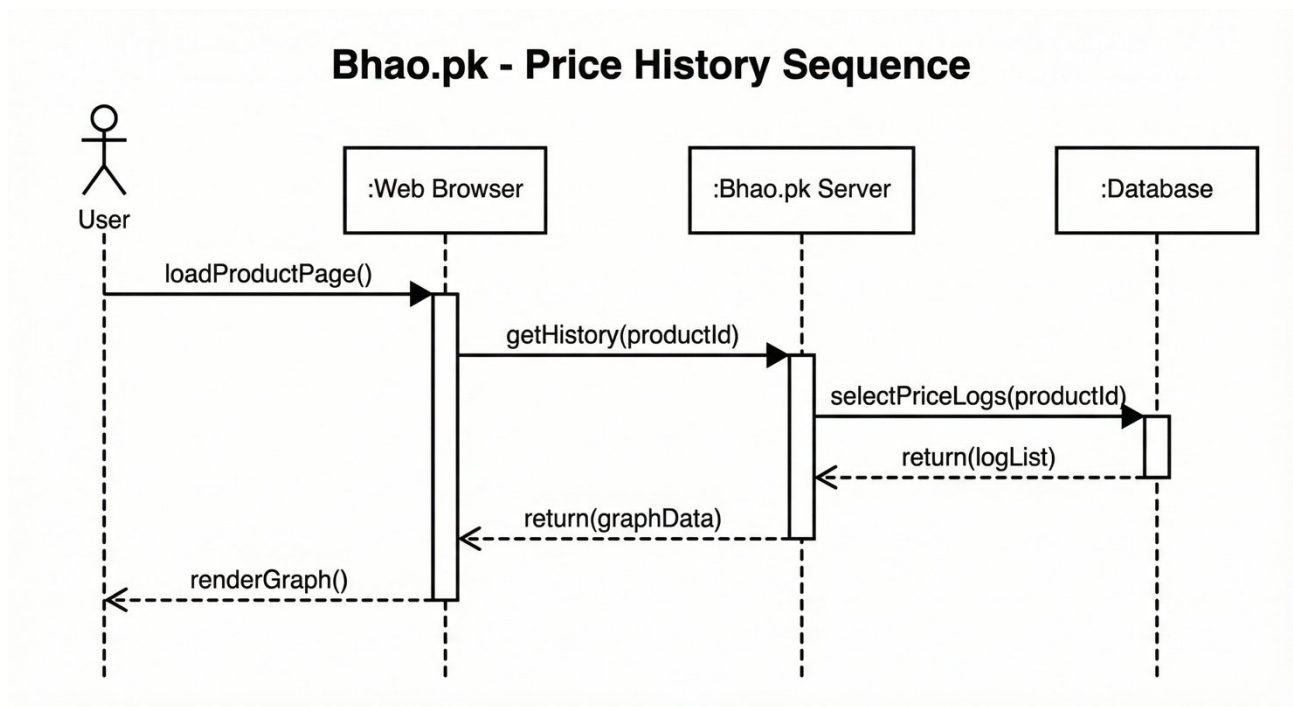


*xix: Figure 3.3: Filter & Sort Sequence Diagram*

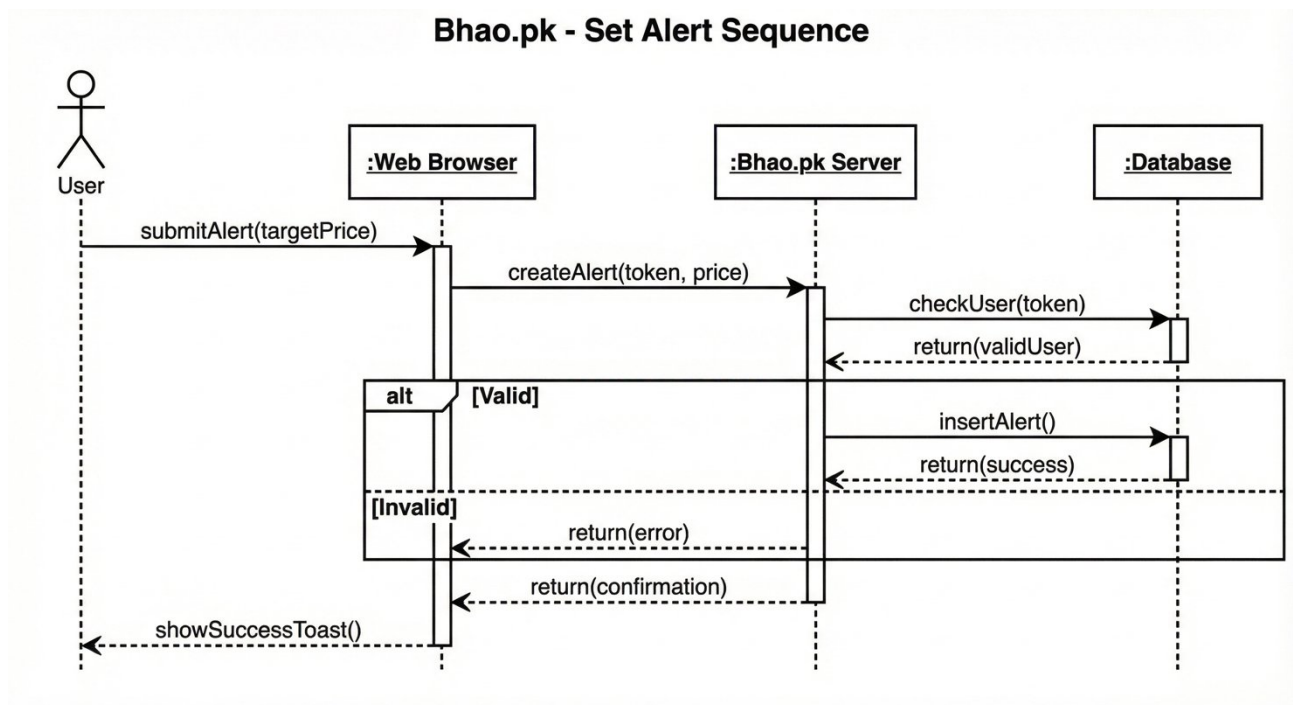
## Bhao.pk - View Details Sequence



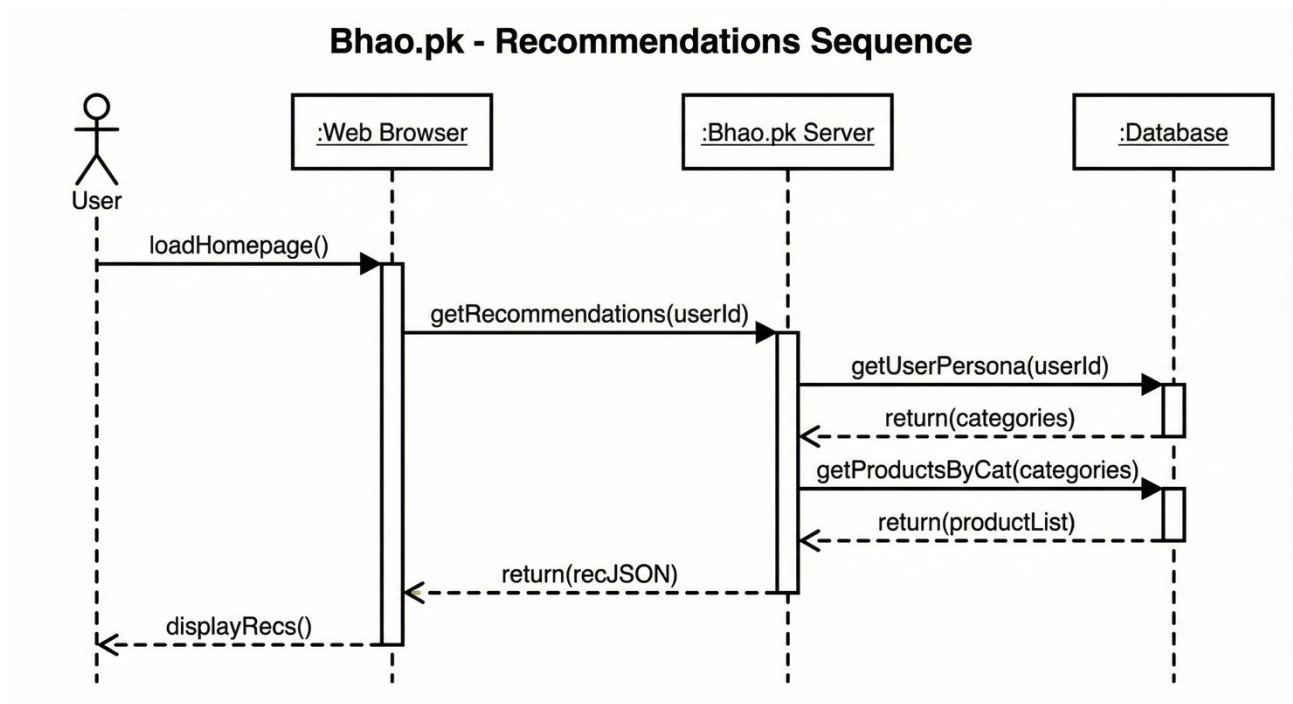
**xx: Figure 3.4: View Details Sequence Diagram**



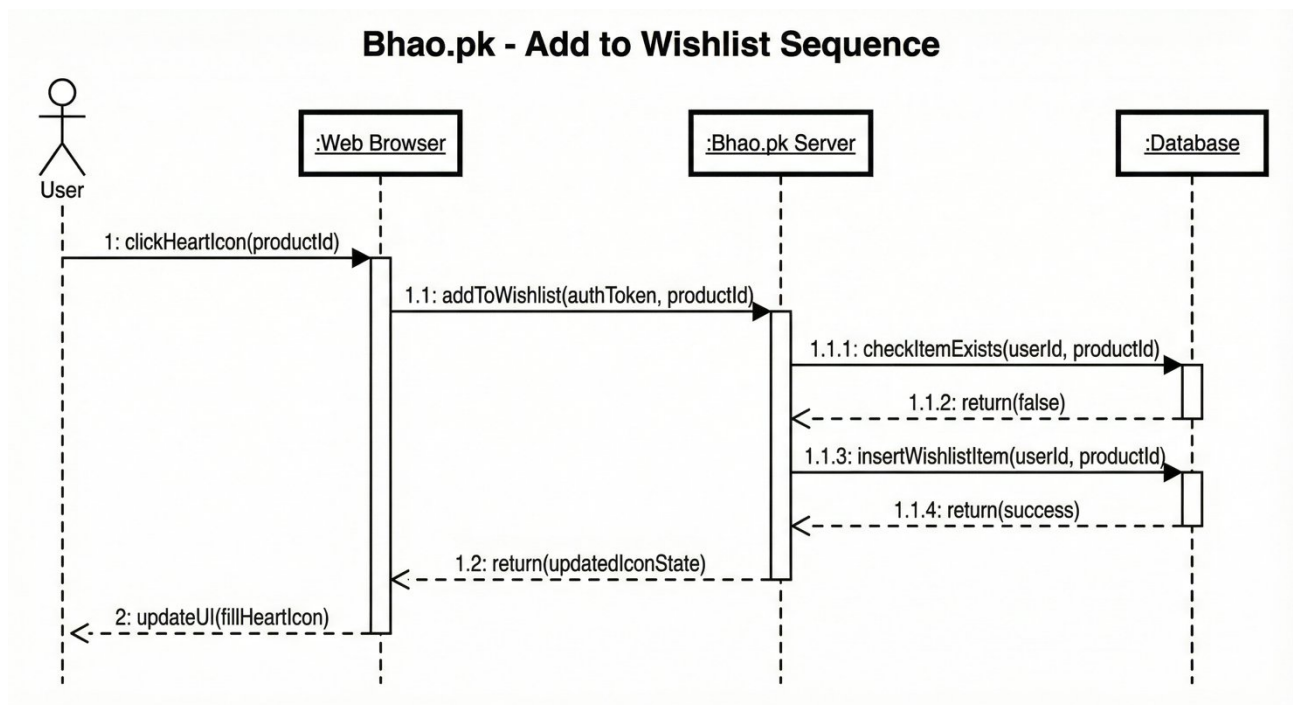
*xxi: Figure 3.5: Price History Sequence Diagram*



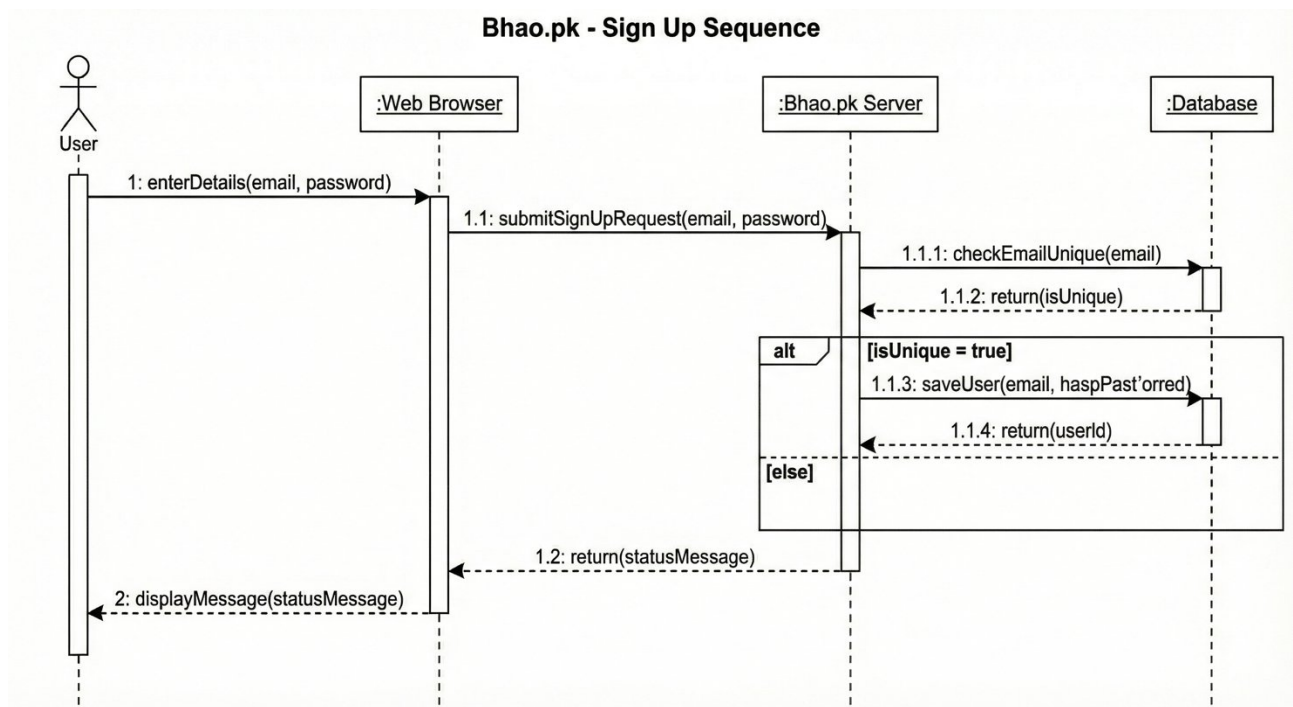
xxii: Figure 3.6: Set Price Alert Sequence Diagram



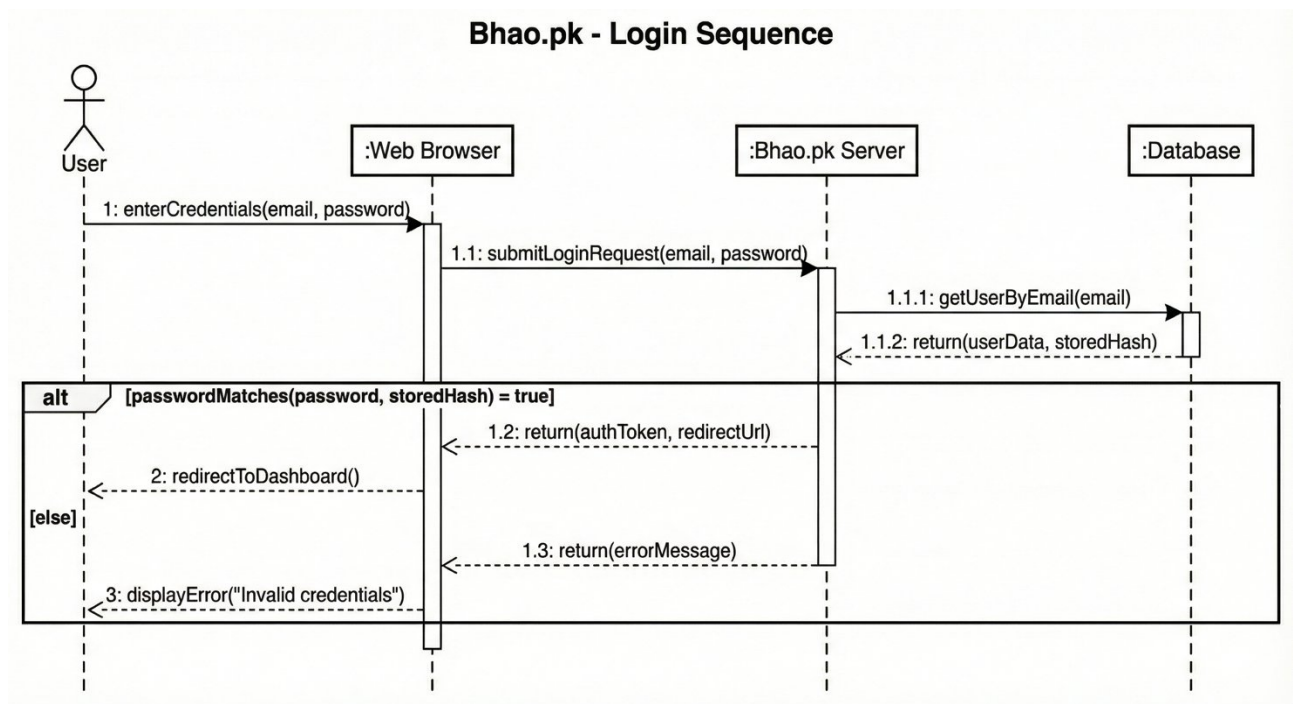
xxiii: Figure 3.7: Show Recommendations Sequence Diagram



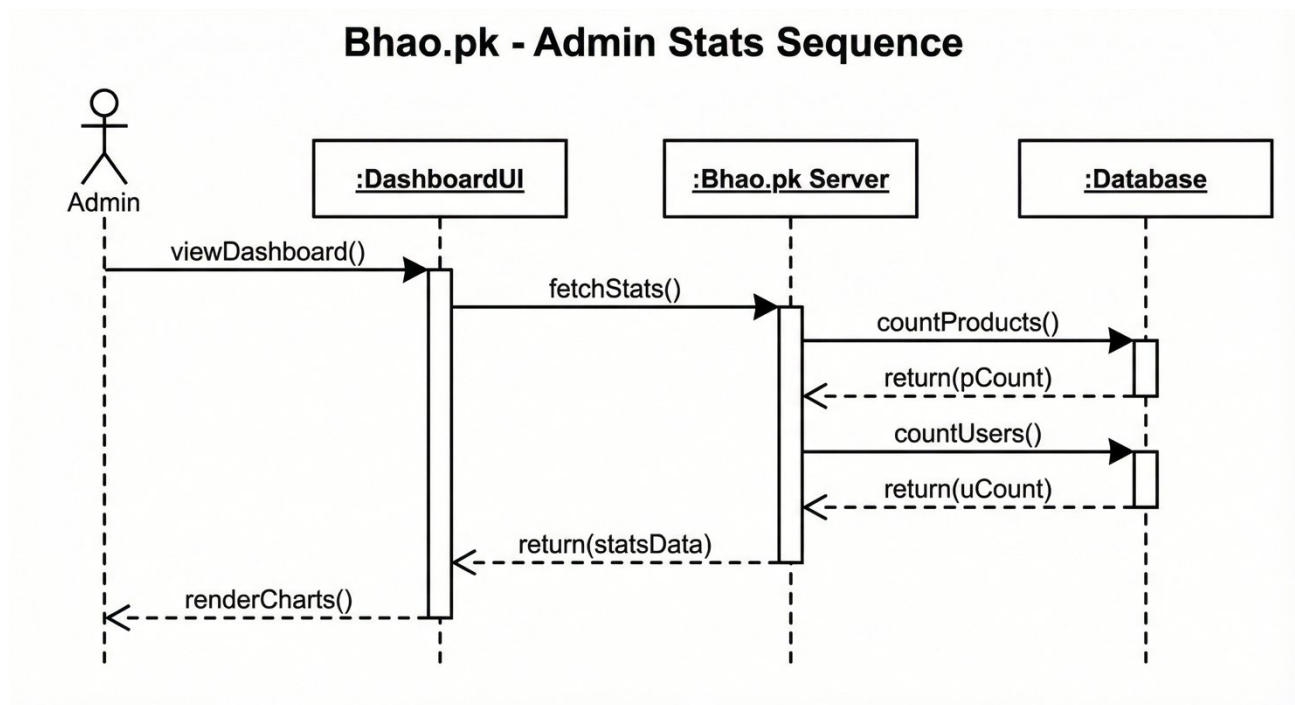
**xxiv: Figure 3.8: Manage Wishlist Sequence Diagram**



xxv: Figure 3.9: Sign Up Sequence Diagram



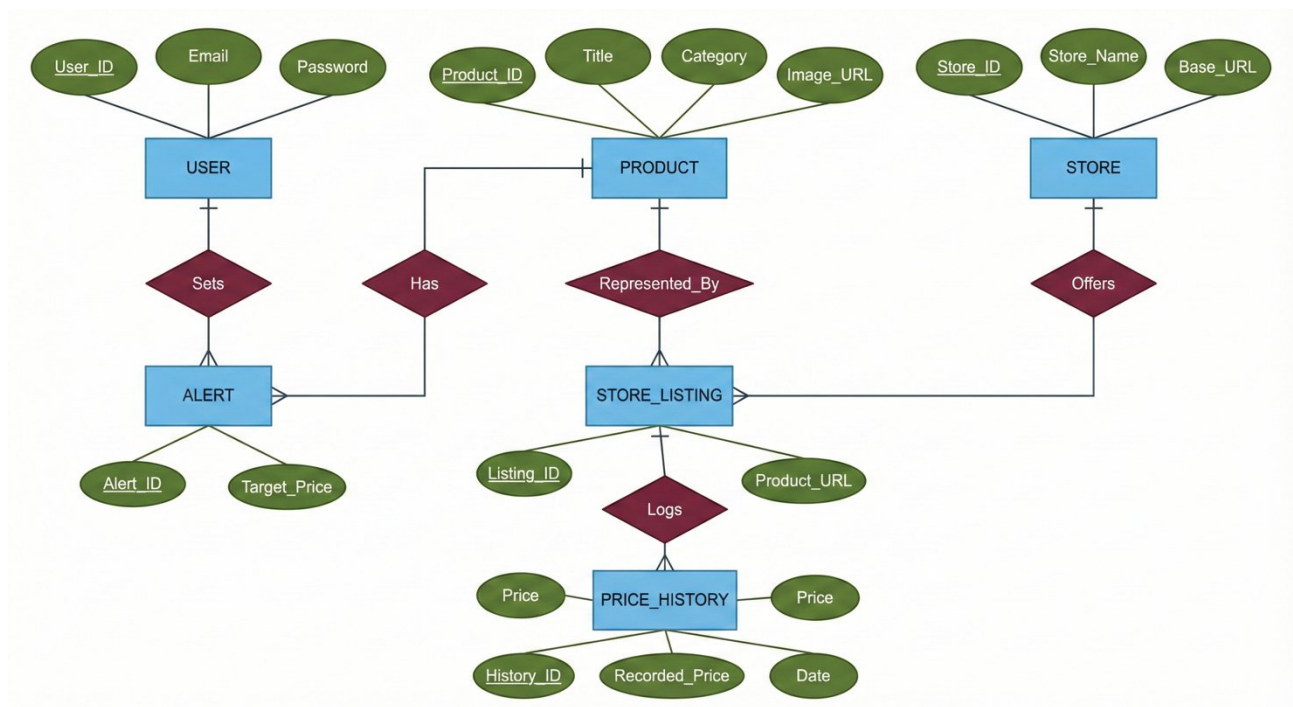
xxvi: Figure 3.10: Login Sequence Diagram



xxvii: Figure 3.11: Sign Up Sequence Diagram

### 3.5. Entity Relationship Diagram

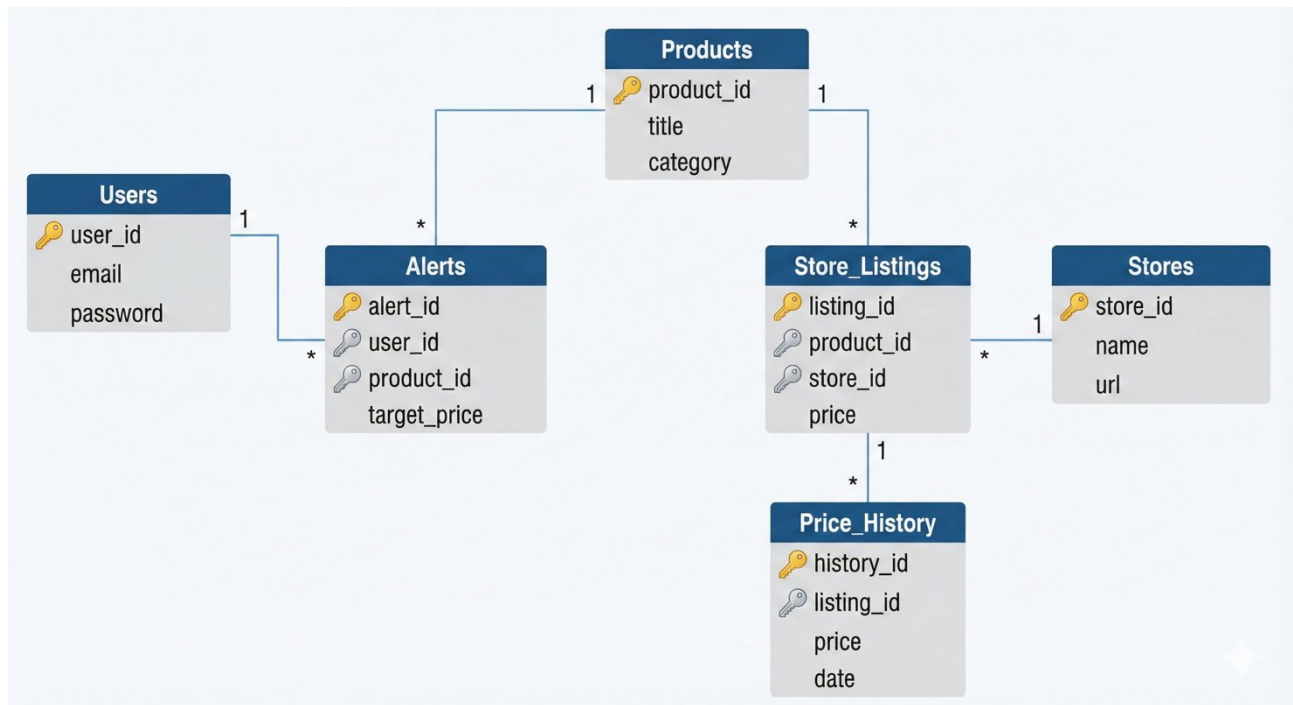
The Entity Relationship Diagram (ERD) for Bhao.pk visualizes the system's logical data structure, anchored by the **Product** entity which connects to multiple **StoreListings** to represent unique offers from various **Stores**. **Users** interact with the system by establishing **Alerts** on these products, creating a direct link between user preferences and market data. Additionally, the model incorporates a **PriceHistory** entity related to each listing, ensuring that every price fluctuation is recorded to support historical trend analysis.



xxviii: Figure 3.12: Entity Relations Diagram

### 3.6. Database Schema

The Database Schema for Bhao.pk represents the logical configuration of the relational database, structured around the central **Product** table which maintains a universal catalog of items. It utilizes a **Store\_Listings** table to bridge these generic products with specific external **Stores**, allowing the system to manage multiple vendors and prices for a single item. **Users** are connected to specific products through an **Alerts** table to manage notifications, while a dedicated **Price\_History** table archives temporal data for every listing to support historical trend analysis



xxix: Figure 3.13: Entity Relations Diagram

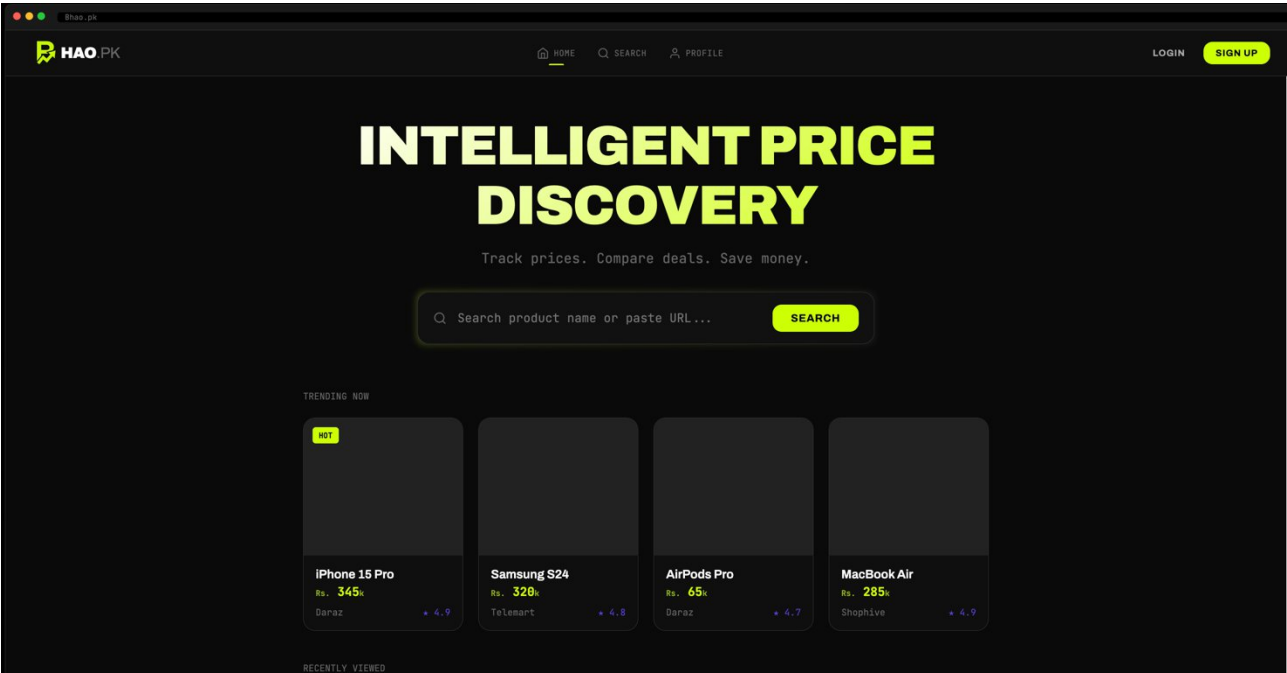
### 3.7. User Interface Design

The Bhao.pk user interface employs a sleek, dark-mode aesthetic with high-contrast neon accents to ensure readability and reduce eye strain. Its minimalist design prioritizes the search function and aggregates complex price data into clean, intuitive dashboards and graphs, enabling users to compare vendors and set alerts effortlessly.

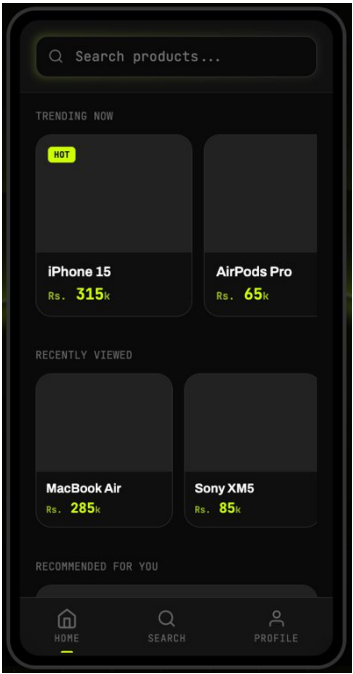
#### Homepage:

The Home Page serves as the primary entry point for Bhao.pk, featuring a minimalist, search-centric layout designed for immediate user engagement. Adopting a cohesive dark-mode aesthetic with neon accents, the web version centers around a prominent search bar and a "Trending Now" grid to encourage exploration. The mobile interface adapts this structure for smaller screens, placing the search bar at the top and utilizing a vertical scroll

layout, accompanied by a fixed bottom navigation bar for seamless accessibility across devices.



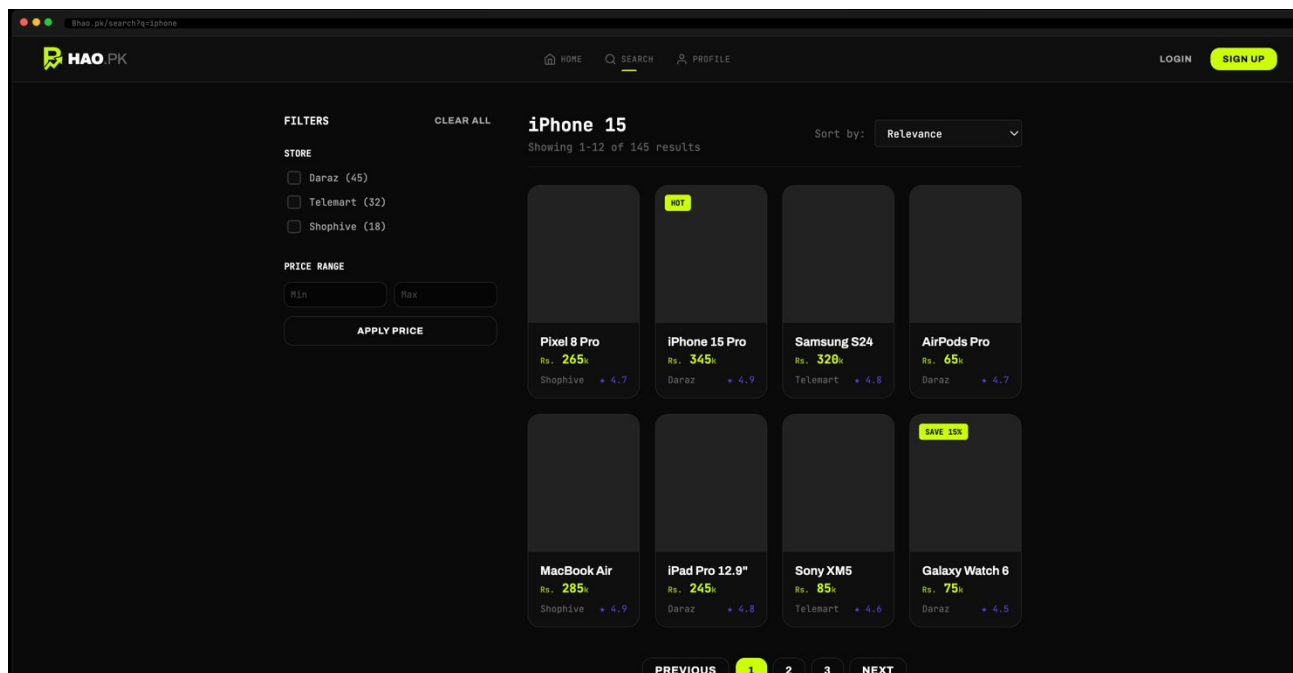
xxx: Figure 3.13: Web Homepage UI



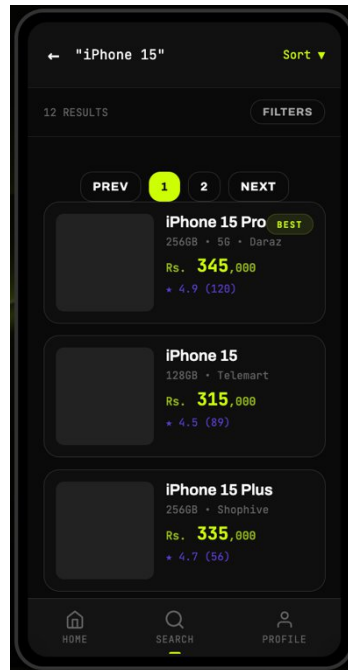
xxxi: Figure3.14: Mobile Homepage UI

## Search:

The Search Page provides a unified product discovery experience, tailored for both web and mobile platforms. On the web interface, the page features a robust sidebar with filters for price, store, and category, alongside a sorting dropdown, allowing users to efficiently refine their search results within a spacious grid layout. The mobile version optimizes this experience for smaller screens by condensing filters into a collapsible menu or modal, stacking product cards vertically for easy scrolling, and ensuring key actions like "Sort" and "Filter" are accessible via touch-friendly buttons at the top of the viewport.



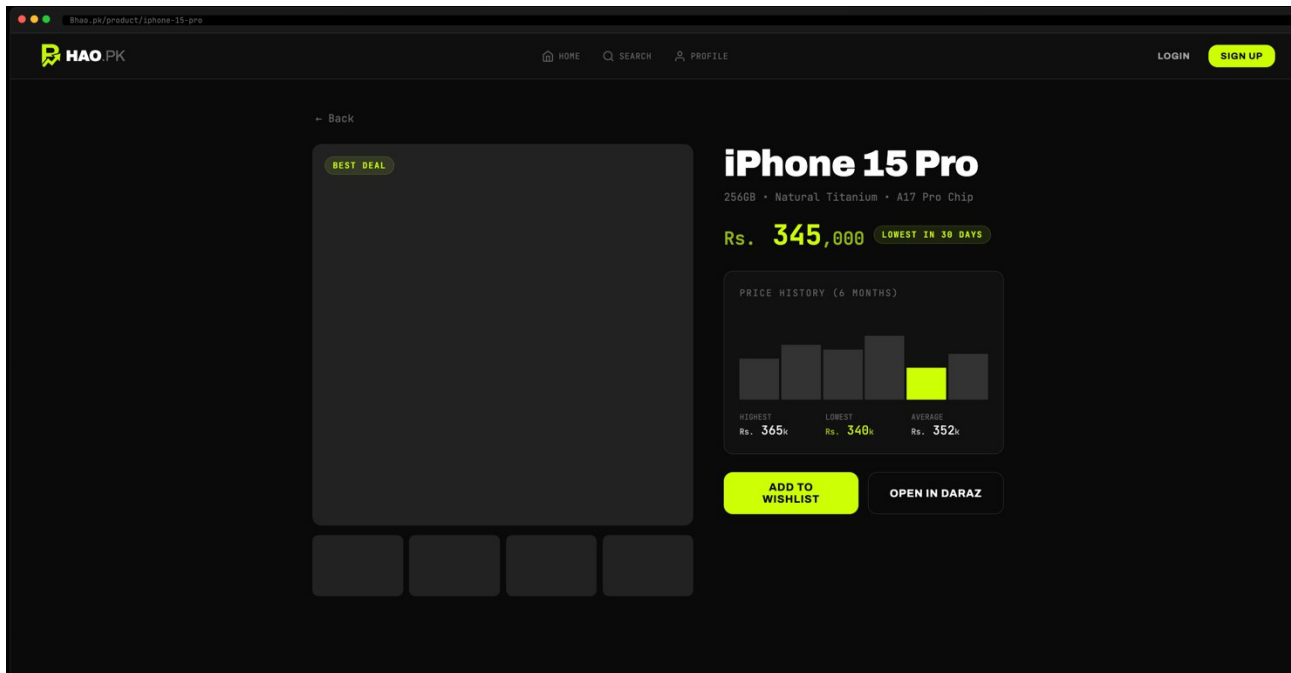
xxxii: Figure 3.15: Mobile Search UI



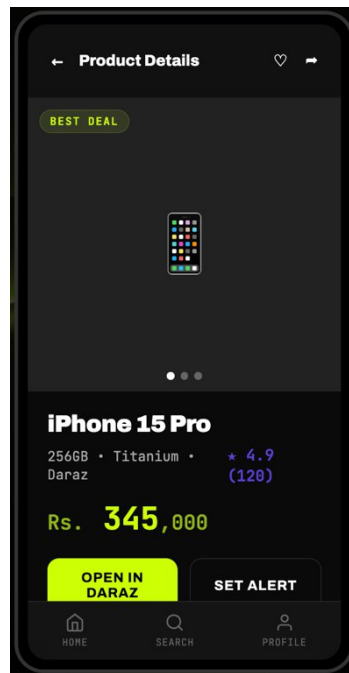
xxxiii: Figure 3.16: Mobile Search UI

## **Product:**

The Product Detail Page serves as the primary decision-making hub, presenting comprehensive product specifications and historical price trends in a unified, dark-mode interface. The web layout utilizes a split-screen design to showcase high-resolution imagery alongside an interactive price history graph and vendor comparison links. On mobile, this content is optimized into a vertical scroll, prioritizing price visibility and anchoring key actions like "Open in Store" and "Set Alert" at the bottom for seamless one-handed interaction.



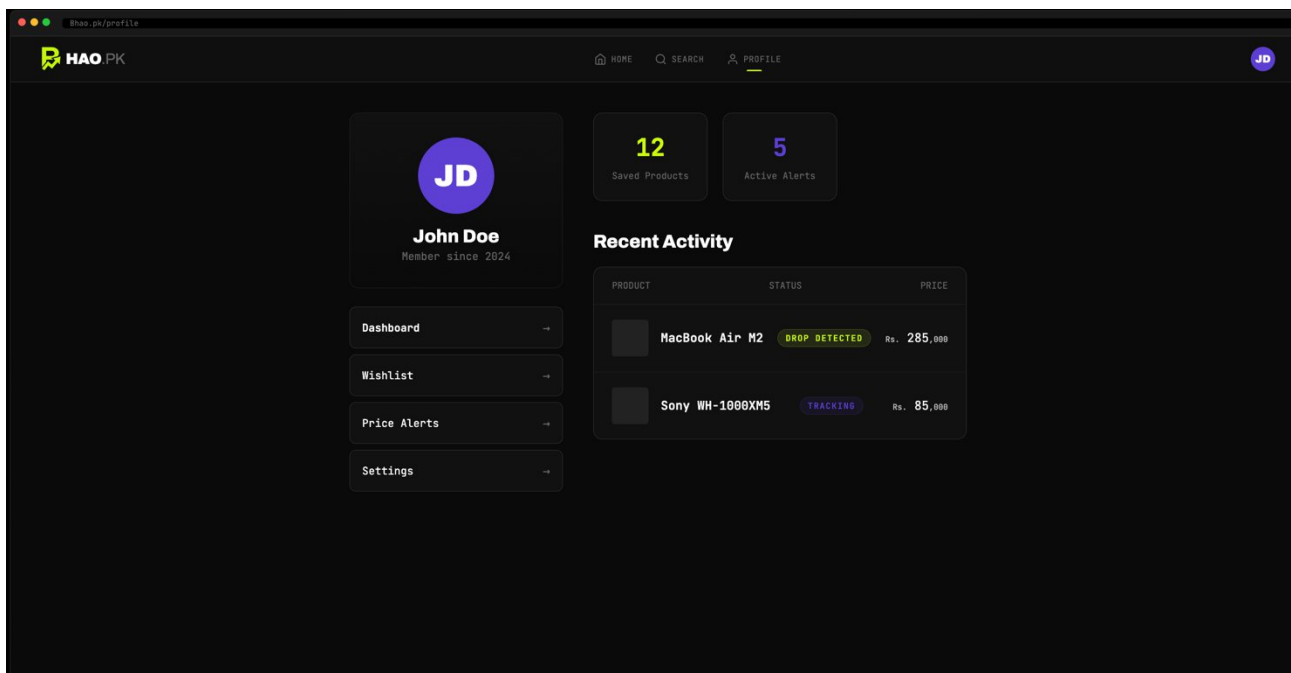
xxxiv: Figure 3.17: Web Product Page UI



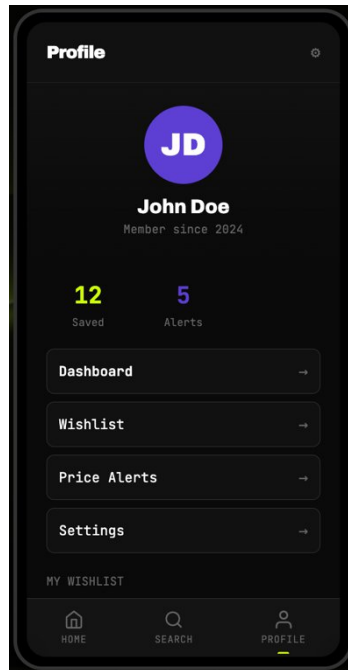
xxxv: Figure 3.18: Mobile Product Page UI

## Profile:

The User Profile Page acts as a personalized command center, featuring a consistent dark-mode design across both platforms. On the web, it presents a dashboard-style layout with key metrics like "Active Alerts" and "Saved Items" displayed as prominent cards alongside a "Recent Activity" table. The mobile version streamlines this into a vertical menu with quick-access buttons for "Dashboard," "Wishlist," and "Price Alerts," ensuring users can manage their account settings and preferences efficiently on smaller screens.



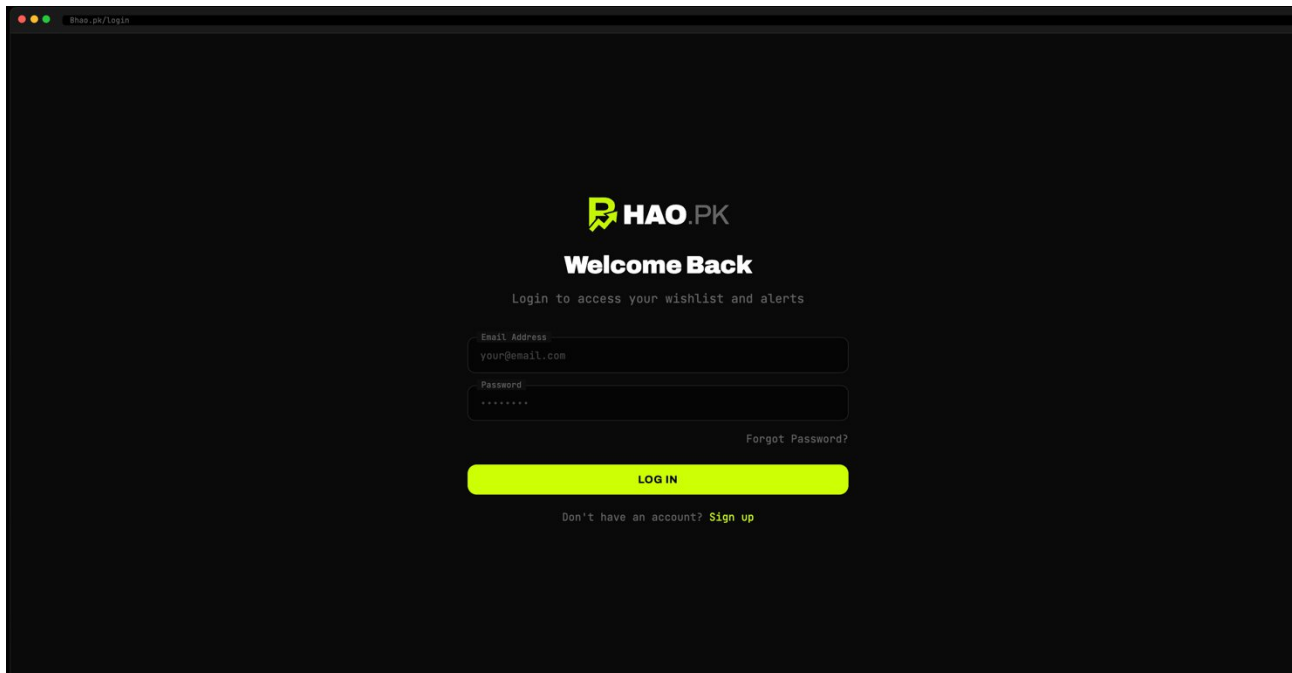
xxxvi: Figure 3.19: Web Profile Page UI



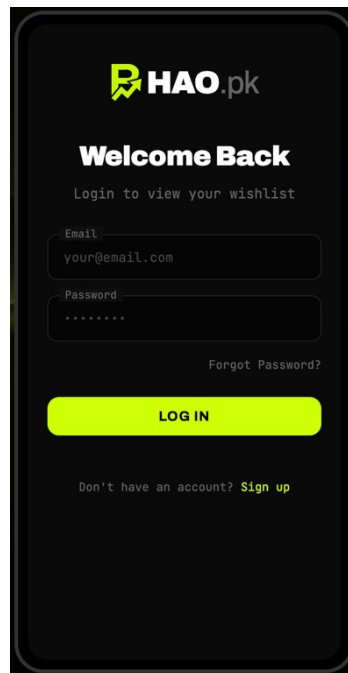
*xxxvii: Figure 3.20: Mobile Profile Page UI*

## **Login:**

The Login Page provides a secure and intuitive entry point for registered users across both platforms. Adhering to the app's dark-mode aesthetic, it features a clean form for email and password input, prominent "Login" and "Forgot Password" actions, and clearly visible links for new user registration. On mobile, the layout is vertically centered with large, touch-friendly buttons, while the web version centers the form within a spacious, branded container to maintain focus.



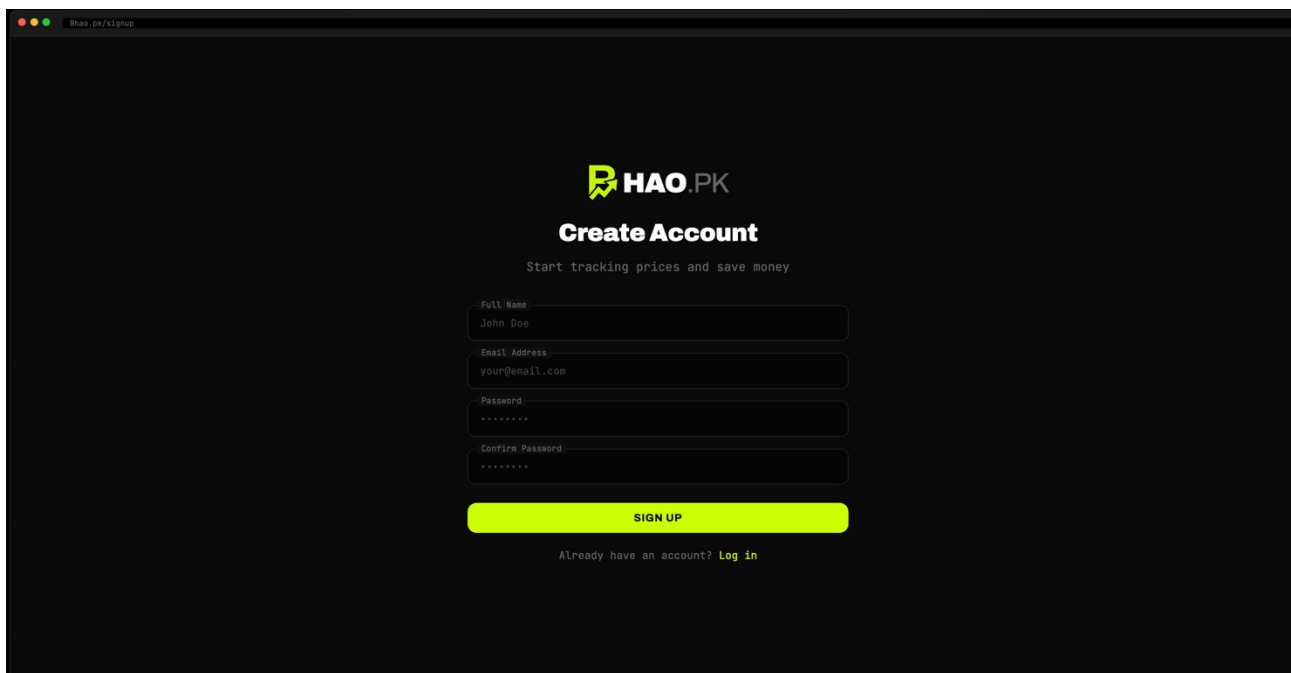
xxxviii: Figure 3.21: Web Login Page UI



xxxix: Figure 3.22: Mobile Login Page UI

## Sign Up:

The Signup Page offers a streamlined registration process for new users, maintaining a consistent dark-mode aesthetic across platforms. Both the web and mobile interfaces feature a simple form for "Full Name," "Email," and "Password" (with confirmation), clearly labeled input fields, and a prominent neon "Sign Up" button. The design minimizes friction by including clear error validation and a direct link to the "Login" page for existing users, ensuring a smooth onboarding experience.



HAO.PK

### Create Account

Start tracking prices and save money

Full Name  
John Doe

Email Address  
your@email.com

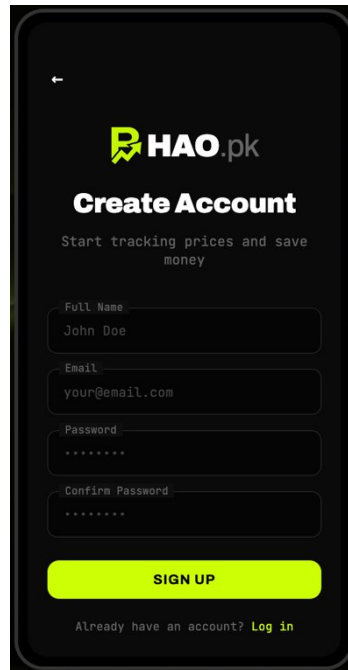
Password  
\*\*\*\*\*

Confirm Password  
\*\*\*\*\*

**SIGN UP**

Already have an account? [Log in](#)

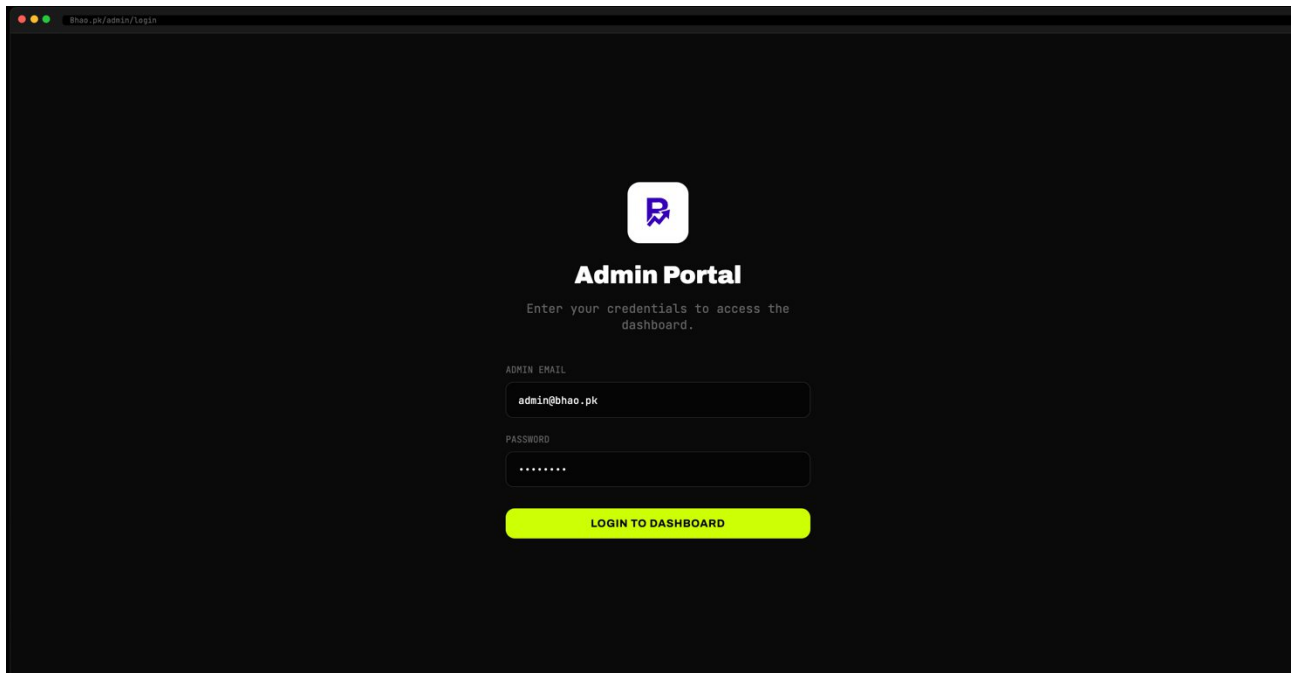
*xl: Figure 3.23: Web Sign Up Page UI*



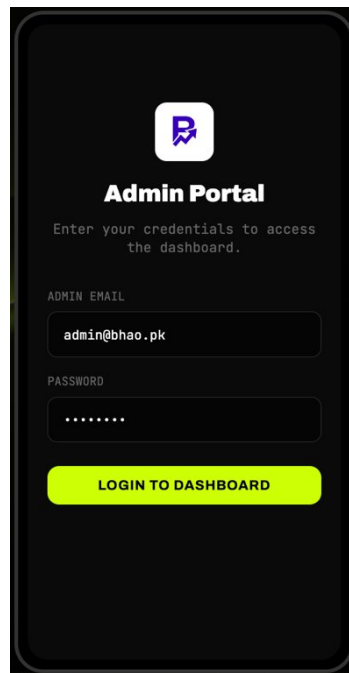
*xli: Figure 3.24: Mobile Sign Up Page UI*

## **Admin Login:**

The Admin Login Page is a secure, restricted-access portal designed with the same cohesive dark-mode aesthetic as the user-facing application. It features a simplified login form requiring specific admin credentials, stripped of social login options or registration links to ensure security. The interface is optimized for both desktop and mobile, providing administrators with a direct gateway to the backend dashboard for monitoring system health and statistics.



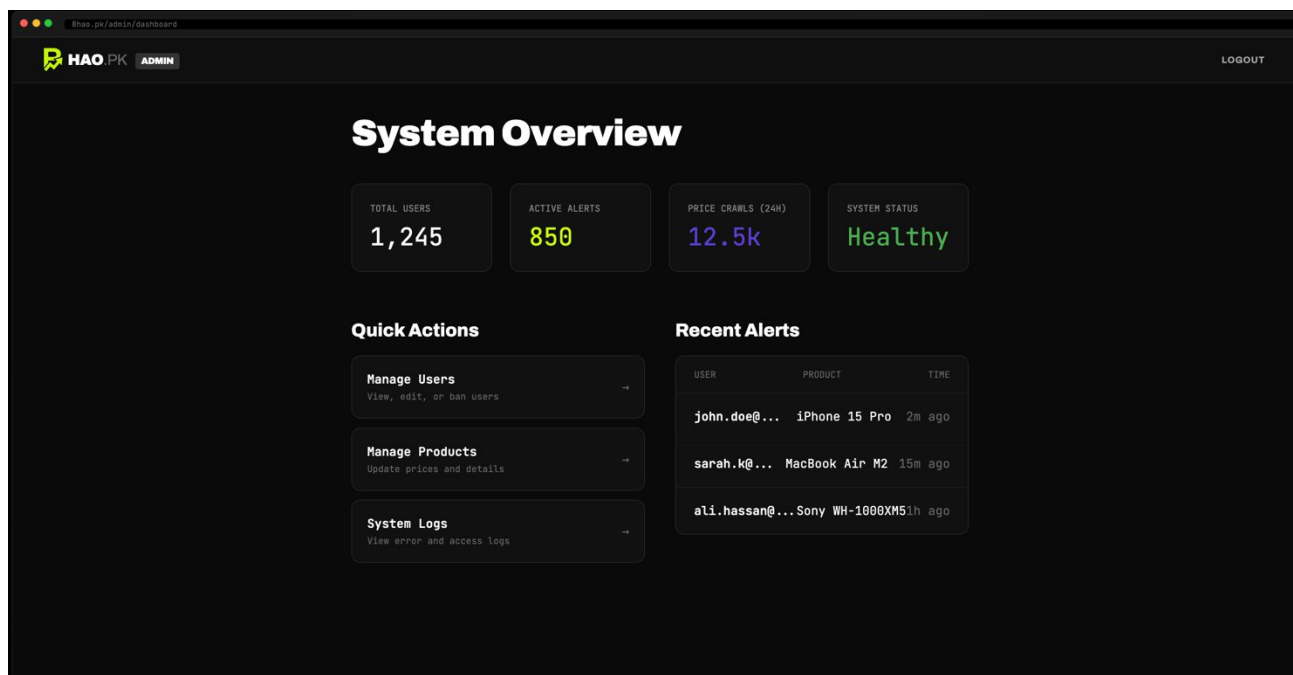
*xlii: Figure 3.25: Web Admin Login Page UI*



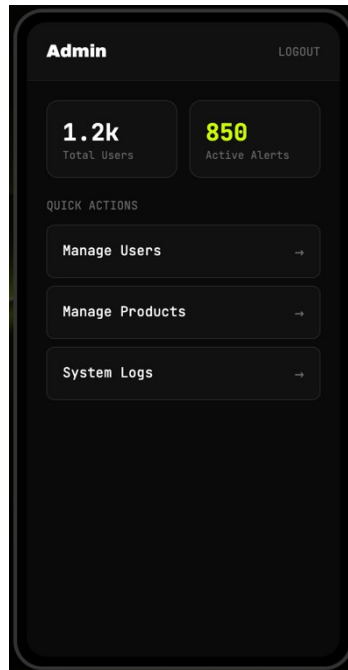
*xliii: Figure 3.26: Mobile Admin Login Page UI*

## Admin Dashboard:

The Admin Dashboard Page acts as a centralized command center, accessible securely on both web and mobile platforms. The interface provides real-time oversight of system health, featuring prominent metrics for total active users, scraped products, and system uptime. The design is optimized for rapid assessment, with interactive charts visualizing user growth and scraper performance, alongside quick-action tools for managing user accounts and system configurations.



*xliv: Figure 3.27: Web Admin Login Page UI*



***xlv: Figure 3.28: Mobile Admin Login Page UI***