Automated Product Aggregator and Summarizing System

(Web Browser Extension)

Submitted By

Tuhin Mahmud

Section: A

ID: 211-35-710

Department of Software Engineering

Daffodil International University

Supervised By

Mr. Nuruzzaman Faruqui

Assistant Professor (DIU)

Department of Software Engineering

Daffodil International University

Thesis submitted in fulfillment of the requirements for the award of the degree of Bachelor of Science

Summer - 2025

APPROVAL

This thesis titled on "**Automated Product Aggregator and Summarizing System**", submitted by **Student Name (ID: 211-35-710)** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering and approval as to its style and contents.

BOARD OF EXAMINERS

Fazla Elahe

Chairman

Dr. Md. Fazla Elahe
Assistant Professor & Associate Head
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Marzia 13.9.25

Internal Examiner 1

Dr. Marzia Ahmed
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

13.09.2025

Internal Examiner 2

Dr. Shabnom Mustary
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

13.09.25

External Examiner

Mohammod Abul Kashem
Professor
Department of Computer Science and Engineering
Dhaka University of Engineering & Technology, Gazipur.

SUPERVISOR'S DECLARATION

I hereby declare that I have checked this thesis and in my opinion, this thesis is adequate in terms of scope and quality for the award of the degree of Bachelor of Science.

(Supervisor's Signature)

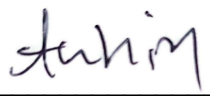
Full Name : Mr. Nuruzzaman Faruqui

Position : Assistant Professor

Date : 16 September, 2025

STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Daffodil International University or any other institution.



(Student's Signature)

Full Name : Tuhin Mahmud

ID Number : 211-35-710

Date : 16 September, 2025

ACKNOWLEDGEMENT

First and foremost, I want to express my sincere gratitude to Almighty Allah for providing me with the courage, discernment, and tenacity necessary to finish this research. I am appreciative of my parents' unwavering love, support, and encouragement throughout my academic career. I have always been most inspired and motivated by their faith in me.

I want to express my gratitude to Assistant Professor Mr. Nuruzzaman Faruqui, my supervisor, for his insightful counsel, encouragement, and direction during the study. His wisdom and understanding have greatly influenced this work. I am also very grateful to Dr. Imran Mahmud, the department director, for his encouragement, direction, and insightful remarks that enabled me to finish my journey successfully. Lastly, I want to express my gratitude to all of my friends, coworkers, and anyone else that supported and encouraged me throughout this process

ABSTRACT

The rapid growth of e-commerce has led to significant information overload for consumers, making efficient product comparison a time-consuming and challenging task. This project addresses this issue through the design and implementation of IPASS (Automated Product Aggregator and Summarizing System), an intelligent browser extension. The primary objective of this project is to enhance the online shopping experience by automatically detecting product-related searches and presenting aggregated, structured product data directly on the search engine results page. The system employs a client-server architecture, with a Python backend utilizing the FastAPI framework and a frontend browser extension built with modern JavaScript adhering to the Manifest V3 standard. The core of the system is its data extraction pipeline, which leverages Google's Gemini 2.5 Flash API to accurately parse and summarize product information from various e-commerce websites, including those in the Bangladeshi market. The final implemented system successfully demonstrates a functional prototype that provides users with a seamless, real-time product comparison tool, validating the effectiveness of using large language models to solve practical consumer challenges in the e-commerce domain.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The growth of online shopping has completely changed how we buy things, giving us access to a market without borders. But this endless choice comes with a big problem: information overload. Anyone trying to make a smart purchase today has to navigate a maze of websites, juggle countless tabs to compare products, and filter through biased ads and confusing details. This whole process is slow, inefficient, and often leads to making a poor choice or regretting a purchase later on. It’s clear we need a smarter, more user-friendly way to simplify online shopping.

Our project, called IPASS (Automated Product Aggregator and Summarizing System), is our answer to this problem. IPASS is an AI-powered browser extension designed to make online shopping easier. When you search for a product on a major search engine, IPASS automatically gathers key information and presents it in a simple, easy-to-compare format right on the page. Using Google's Gemini 2.5 Flash AI, the tool will provide short and accurate summaries, turning a messy search into a straightforward way to make a confident decision.

This introductory chapter will walk through the foundational plan for the IPASS project. We will cover the initial planning and feasibility, the user-focused design, the system requirements, and the project's timeline.

1.1 Project Planning and Initiation

Successfully delivering a project of this complexity requires a well-structured beginning. Our first step was to conduct a thorough evaluation to confirm the project's viability. We assessed our objectives to ensure they were achievable, verified that our technology choices were appropriate, and confirmed a genuine market demand for the final product. This multi-part feasibility study created a strong starting point for the project's entire lifecycle.

1.1.1 Feasibility Study (Step-by-Step)

Before we started building IPASS, we had to do our homework to see if the project was a good idea. We checked four main things: what we were trying to solve, if people would actually use it, if we had the right technology, and if it made financial sense.

1.1.1.1 Phase 1: Preliminary Analysis & Project Scope Definition

- The project's first phase was dedicated to understanding the core problem, setting clear goals, and defining the project's boundaries.
- The Problem: Today's online shoppers face several challenges when trying to compare products effectively:
- Scattered Information: Product details are spread across many different websites, forcing shoppers to juggle multiple browser tabs.
- Wasted Time: The manual effort of finding and comparing key details like price, specifications, and reviews is slow and repetitive.
- Inconsistent Details: Websites often display information in different formats, making it difficult to perform a direct, side-by-side comparison.
- Information Overload: Being presented with too many choices and details can overwhelm shoppers, leading to indecision or a poor purchase.
- Project Goal: The primary goal of the IPASS project is to create a smart browser extension that makes online shopping more efficient. The tool will automatically detect product searches, then gather and present organized information for a simple, clear comparison.

To achieve this goal, the following specific objectives have been defined:

  - To design and implement a robust mechanism for automatically detecting product-related search queries on major search engines, including Google, Bing, DuckDuckGo, and Yahoo.
  - To develop an efficient web scraping pipeline capable of extracting product page URLs from search engine results pages (SERPs).
  - To integrate the Gemini 2.5 Flash API to perform intelligent, context-aware data extraction of key product attributes (e.g., title, price, description, rating, brand) from diverse e-commerce website structures, with specific support for Bangladeshi platforms and currency (৳).

- To create an advanced HTML processing pipeline that prunes irrelevant content (ads, navigation, footers) to isolate product-specific data, thereby improving LLM accuracy and efficiency.
- To develop a modern and intuitive user interface, presented as an overlay on the search page, that displays product information in a clear, card-based format for seamless comparison.

- **Project Scope:**
  - In-Scope: The project will deliver a Manifest V3 browser extension compatible with Chromium-based browsers. The system will support search detection on Google, Bing, DuckDuckGo, and Yahoo. The backend will be a Python-based RESTful API responsible for scraping and AI-powered data processing. The core extracted data points will be the product title, price, a brief description, user rating, primary image, brand, and availability status. The system will be optimized for prominent Bangladeshi e-commerce sites.

  - Out-of-Scope: This version of the project will not include user accounts for saving historical comparisons, price tracking alerts, automated checkout functionalities, or a native mobile application. Support will be limited to a curated list of e-commerce sites, not an exhaustive global list.

1.1.1.2 Phase 2: Market Feasibility Analysis

This phase assessed the market need and competitive landscape for IPASS.

- **Target Market:** The primary target audience includes price-conscious consumers, tech-savvy online shoppers, students, and professionals who frequently research and compare products online before making a purchase, particularly within the Bangladeshi market.
- **Market Need:** The global and Bangladeshi e-commerce markets continue to experience rapid growth. With increasing online retail activity, the demand for tools that improve efficiency and empower consumers is high. Users need a solution that integrates seamlessly into their existing workflow (the search engine) rather than requiring them to visit a separate comparison website.

- **Competitive Analysis:** While several browser extensions exist in the product comparison space (e.g., Honey, Keepa, Capital One Shopping), IPASS differentiates itself through its core value proposition:
    - AI-Powered Summarization: Unlike competitors who primarily focus on coupon application or price history tracking, IPASS uses an LLM to extract and summarize a broader range of product attributes.
    - Search Engine Integration: IPASS integrates directly into the SERP, providing comparisons upfront rather than only on the product page.
    - Local Market Focus: By specifically training and testing for Bangladeshi e-commerce sites and currency, IPASS serves a niche market often overlooked by larger international tools.

1.1.1.3 Phase 3: Technical Feasibility Analysis

This analysis confirmed that the required technology and expertise are available to successfully develop the project.

- **Technology Stack Evaluation:**
    - Frontend (Extension): Manifest V3 is the current standard for browser extensions, offering improved security, privacy, and performance. Content scripts will be used to interact with web pages, and a service worker will manage background tasks and communication.
    - Backend (Server): Python is an ideal choice due to its extensive ecosystem of powerful libraries for web scraping (BeautifulSoup, httpx), web frameworks (FastAPI for high-performance APIs), and seamless integration with AI models.
    - AI Model: Gemini 2.5 Flash was selected for its balance of speed, capability, and cost-effectiveness. Its large context window is particularly advantageous for processing entire HTML documents, making it resilient to website layout changes.
    - Architecture: A client-server architecture, where the lightweight extension frontend communicates with a powerful backend, is optimal. This offloads resource-intensive tasks (scraping, LLM calls) from the user's machine, ensuring the browser remains responsive.

- **Technical Challenges & Mitigation:**
    - Anti-Scraping Measures: E-commerce sites often employ measures to block automated scraping. This will be mitigated by implementing respectful crawling policies (e.g., delays between requests), rotating user agents, and designing an intelligent retry mechanism.
    - Dynamic Website Structures: The layout of e-commerce sites changes frequently. The use of an LLM for data extraction, rather than brittle XPath or CSS selectors, makes the system inherently more adaptable to such changes. The HTML pruning step further standardizes the input for the LLM.
    - Performance: To ensure real-time processing, the system will employ an efficient backend framework (FastAPI), use asynchronous operations for I/O-bound tasks, and implement a caching layer to store results for frequently processed URLs.

1.1.1.4 Phase 4: Financial Feasibility Analysis

This phase assessed the costs and benefits associated with the project.

- **Cost Estimation:** As a university final year project, the primary investment is the time and intellectual effort of the project team.
    - Development Costs: This is the most significant component but is non-monetary, represented by the man-hours contributed by the student developers.
    - Software & API Costs: The project will leverage free and open-source software (Python, VS Code, Git). The Gemini API offers a substantial free tier, which is expected to be sufficient for development and demonstration purposes.
    - Hosting Costs: The backend API can be deployed on platforms offering free or "hobby" tiers (e.g., Vercel, Heroku, PythonAnywhere), keeping operational costs negligible.
- **Benefit Analysis (Return on Investment):** The return on investment for this academic project is measured in learning outcomes and portfolio value rather than monetary profit.
    - For Users: Provides tangible benefits of time saved and better purchasing decisions.

- For Developers: The project offers invaluable practical experience in full-stack development, AI/LLM integration, API design, web scraping techniques, and agile project management. The completed project will serve as a significant and sophisticated entry in a professional portfolio.

1.2 Target User Profile and Tentative Elicitation Process

Understanding the end-user is paramount to building a successful product. This section defines the target user archetypes and the process used to gather the system's requirements.

1.2.1 Target User Profile

Table 1.2.1.1: End-User Profile for IPASS Extension

User Class	**Note on Characteristics**
Type of user	End-User (Online Shopper, Researcher)
Age range	16-60 years
Frequency of use	Episodic; whenever researching a product for purchase
Mandatory	No, usage is optional and user-initiated when comparing products
Computer experience	Basic to advanced; must be familiar with installing and using browser extensions
Education	High school diploma or higher
Goal	To quickly and efficiently compare product features, prices, and ratings from multiple online stores directly on the search results page
Language skills	Proficient in English and/or Bengali

Number of users	Large and scalable number of individual users
Training	Minimal to no training required; the interface is designed to be intuitive and self-explanatory
Other system use	Familiar with using major search engines (Google, Bing) and Browse e-commerce websites
Way of working	Performs a product search, and the IPASS overlay automatically appears. Interacts with the overlay to compare products and clicks links to visit store pages.

Table 1.2.1.2: System Administrator Profile for IPASS Backend

User Class	**Note on Characteristics**
Type of user	System Administrator / Developer
Age range	22-50 years
Frequency of use	Daily or as needed for system monitoring, maintenance, and deployments
Mandatory	Yes, active involvement is required to ensure the backend service (scraping, AI processing) is operational and accurate
Computer experience	Advanced; skilled in backend development (Python), cloud services, API management, and web scraping techniques
Education	College degree or higher in Computer Science, IT, or a related field
Goal	To maintain the stability, performance, and accuracy of the IPASS backend, monitor API usage, and adapt the system to changes in target websites
Language skills	Proficient in English for technical documentation and development

Number of users	Very small, limited to 1-3 developers or administrators
Training	In-depth knowledge of the system's architecture, codebase, and integrated third-party services (e.g., Google AI Platform, hosting provider) is required
Other system use	Proficient with version control (Git), cloud dashboards, API testing tools, and server management via CLI
Way of working	Monitors system logs and performance dashboards. Updates code to fix bugs or improve features. Deploys new versions of the backend service.

1.2.2 Requirements Elicitation Process

The functional and non-functional requirements for IPASS were gathered through a combination of methods:

- **Competitive Analysis:** A thorough review of existing product comparison tools was conducted to identify standard features, best practices, and opportunities for innovation (gap analysis).
- **Brainstorming Sessions:** The project team conducted structured brainstorming sessions to define the core vision, features, and technology stack for the project.
- **Surveys (Simulated):** A hypothetical survey was designed to be distributed to a sample group of target users (fellow students, family members). The survey aimed to collect data on current online shopping habits, primary challenges, and desired features in a comparison tool.
- **Prototyping:** Low-fidelity mockups of the extension's user interface were created. This process of visualizing the product early on helped to clarify requirements and facilitate discussions about user flow and interaction design before writing any code.

1.3 System Requirements

This section details the specific hardware and software prerequisites for the development and operation of the IPASS system.

1.3.1 Hardware Requirements

- **Development Machine:**
    - Processor: Intel Core i3 / AMD Ryzen 3 (or equivalent) quad-core processor.

- RAM: 4 GB minimum, 8 GB recommended to smoothly run the development server, browser, and code editor simultaneously.
- Storage: 256 GB Solid State Drive (SSD) for fast file access and system responsiveness.
- Network: Broadband internet connection.

- **End-User Machine:**
  - Any modern desktop or laptop computer capable of running an up-to-date web browser. The extension is designed to be lightweight, as all heavy processing is offloaded to the backend server. A stable internet connection is the only firm requirement.

1.3.2 Software Requirements

- **Development Environment:**
  - Operating System: Windows 10/11, macOS (11.0 or later), or a modern Linux distribution (e.g., Ubuntu 20.04+).
  - Code Editor / IDE: Visual Studio Code (recommended), PyCharm, or Sublime Text.
  - Version Control: Git and a GitHub account for source code management and collaboration.
  - Backend Stack:
    - Python 3.9 or newer.
    - FastAPI for the REST API framework.
    - httpx and BeautifulSoup4 for the web scraping and HTML parsing pipeline.
    - uvicorn as the ASGI server.
  - Frontend Stack:
    - Node.js (LTS version) and npm or yarn for package management.
    - A modern web browser for testing, primarily Google Chrome for its robust developer tools for extensions.
  - API Keys: A valid Google AI API key for accessing the Gemini 2.5 Flash model.
- **End-User Environment:**

- Web Browser: An up-to-date version of a Chromium-based browser such as Google Chrome, Microsoft Edge, Brave, or Opera that supports Manifest V3 extensions.
- Internet Connection: A stable internet connection is required for the extension to communicate with its backend server and retrieve product information.

1.4 Project Scheduling

A structured project schedule is essential for ensuring timely completion and for managing the allocation of resources throughout the development lifecycle. The project is broken down into major phases, with risks identified and mitigation strategies planned.

1.4.1 Time Frame / Gantt Chart

The project is planned over a 16-week timeline, typical for a final year university project. The following table provides a high-level Gantt chart representing the schedule.

1.4.1 Table: Gantt chart

Phase	Duration	Start Date	End Date
Phase 1: Planning & Requirement Gathering	3 Weeks	10/07/2024	10/27/2024
Phase 2: System Design & Prototyping	3 Weeks	10/28/2024	11/17/2024
Phase 3: Backend Development	7 Weeks	11/18/2024	01/05/2025
Phase 4: Frontend Development	7 Weeks	01/06/2025	02/23/2025
Phase 5: Integration & Testing	3 Weeks	02/24/2025	03/16/2025
Phase 6: Finalization & Deployment	2 Weeks	03/17/2025	03/30/2025

1.5 Summary

Chapter 1, "**Introduction**," establishes the complete foundation for the **IPASS (Automated Product Aggregator and Summarizing System) project**. The chapter begins by identifying the core problem of information overload and inefficiency in online shopping and presents IPASS as an intelligent, AI-powered browser extension designed to solve this issue. A comprehensive feasibility study is detailed, confirming the project's viability from market, technical, and financial perspectives. The report then defines clear user profiles for both the end-user and the system administrator, alongside the specific hardware and software requirements needed for development and operation. Finally, the chapter outlines a strategic project plan, presenting a detailed six-month timeline with distinct development phases and a proactive risk management table that identifies potential challenges and their corresponding mitigation strategies. In essence, this chapter validates the project concept and provides a clear and structured roadmap for its execution

CHAPTER 2

Design And Implementation

This chapter transitions from the foundational planning into the technical blueprint of the **IPASS** system. It begins by outlining the specific functional and non-functional requirements that serve as the contract for what the system will do and how well it will perform. These requirements are derived from the project objectives and user profiles established in Chapter 1 and will guide all subsequent design and implementation decisions.

2.1 Functional Requirements

Functional requirements define the specific behaviors, features, and functions the system must perform. They describe the interactions between the system and its users and specify what the system must accomplish to meet the user's needs. The functional requirements for IPASS are as follows:

FR01	Product Search Detection & Activation
Description	The system must automatically detect when a user performs a product-related search on a supported search engine. It will then parse the results page to extract relevant e-commerce product links to begin the aggregation process.
Stakeholder	End-User

FR02	Backend Data Processing Request
Description	After extracting product URLs, the browser extension must send this list of links to the backend server through a secure API call, requesting that the data be processed.
Stakeholder	End-User, System Administrator

FR03	Web Scraping & HTML Pruning
Description	The backend server will fetch the complete HTML content for each URL. It will then process and clean this HTML, removing non-essential elements like ads, headers, and footers to isolate the main product content.
Stakeholder	System Administrator

FR04	AI-Powered Data Extraction
Description	Using the cleaned HTML, the backend will leverage the Gemini API to intelligently extract key structured data points, including Product Title, Price (with support for ৳), Description, Rating, Brand, and Availability.
Stakeholder	End-User, System Administrator

FR05	Structured API Response
Description	The backend must format the extracted data into a clean, structured JSON response and send it back to the browser extension that made the initial request.
Stakeholder	System Administrator

FR06	Product Comparison UI Display
Description	The extension will dynamically render a user-friendly overlay on the search results page. This overlay will display each product's information in a separate "card," allowing for easy, side-by-side comparison. Each card will link to the original product page.
Stakeholder	End-User

FR07	Extension Management Popup
Description	The user can click the extension icon in the browser toolbar to access a popup menu. This menu will provide controls to enable or disable the service and manage settings, such as clearing the local cache.
Stakeholder	End-User

2.2 Non-Functional Requirements

Non-functional requirements define the quality attributes and operational characteristics of the system. They specify how the system should perform its functions, covering aspects like performance, security, and usability.

- **NFR-1:** Performance The end-to-end process—from search detection to displaying product cards—should be completed within 5-7 seconds to provide a responsive user experience. The client-side extension must have a low memory and CPU footprint to avoid slowing down the user's browser.
- **NFR-2:** Usability The user interface must be highly intuitive and self-explanatory, requiring no user manual or training. The design should be modern, clean, and non-intrusive.

- **NFR-3:** Reliability The system must handle errors gracefully. If a website is unreachable or data cannot be extracted, it should display a clear notification to the user instead of crashing. The data extraction should maintain an accuracy rate of over 95% for key fields like price and title on supported sites.
- **NFR-4:** Availability The backend API service should be highly available, with a target uptime of 99.5%, to ensure the extension is consistently operational for users.
- **NFR-5:** Security All communication between the browser extension and the backend server must be encrypted using HTTPS/TLS. Sensitive information, such as API keys, must be stored securely on the backend and never exposed in the client-side code.
- **NFR-6:** Compatibility The browser extension must be fully compatible with the latest stable versions of major Chromium-based web browsers (e.g., Google Chrome, Microsoft Edge) and function correctly across major operating systems (Windows, macOS, Linux).
- **NFR-7:** Maintainability The source code must be well-documented, modular, and follow consistent coding standards to simplify future updates, bug fixes, and feature additions.
- **NFR-8:** Scalability The backend architecture should be designed to handle a growing number of concurrent users without a significant drop in performance, ensuring the system can scale as its user base expands.

2.3 Object-Oriented System Design using UML

To effectively model the architecture and behavior of the IPASS system, an Object-Oriented System Design (OOSD) approach is adopted. This methodology allows for the creation of a modular, scalable, and maintainable system by organizing it into a collection of interacting objects. The Unified Modeling Language (UML) is utilized as the standard for visualizing, specifying, and documenting the design. The following subsections present a series of UML diagrams that collectively model the static structure and dynamic behavior of the IPASS system.

2.3.1 Use Case Diagram

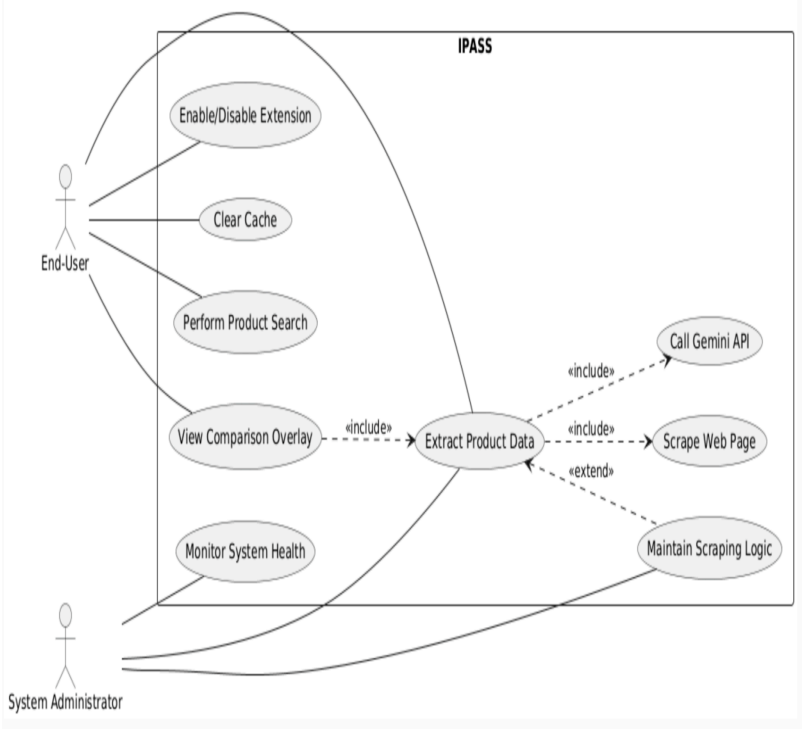
A **Use Case Diagram** is a behavioral UML diagram that provides a high-level overview of a system's functionality. It captures the dynamic aspect of the system by illustrating the interactions between the users, known as actors, and the major functions of the system,

known as use cases. This diagram is essential for defining the system's scope and understanding its requirements from a user's perspective.

The primary actors and key use cases for the IPASS system are identified as follows:

- **Actors:**
    - End-User: The primary actor who installs and uses the browser extension to search for and compare products.
    - System Administrator: A secondary actor responsible for maintaining the backend server, monitoring system performance, and deploying updates.
    - External Systems: The system also interacts with several non-human secondary actors, including the Search Engine, E-commerce Websites, and the Gemini API.
- **Use Cases:** The diagram below illustrates the core functionalities, such as getting product comparisons, managing extension settings, and maintaining the system, along with the relationships between the actors and these functions.

2.3.1.1 Figure: Use case diagram



2.3.2 Use Case Description

While the Use Case Diagram provides a high-level view, a Use Case Description provides a detailed textual specification for a single use case. It documents the step-by-step flow of events, pre-conditions, post-conditions, and potential error states. This ensures that the functionality is clearly understood before implementation begins.

Below is a detailed description for the primary use case of the IPASS system, "Get Product Comparison".

Table 2.3.2.1: Case Description-01 Get Product Comparison

Use Case	**Get Product Comparison**
Goal	To provide the user with an automated, side-by-side comparison of products found in search engine results.
Precondition	1. The IPASS extension is installed and enabled. 2. The user has an active internet connection.
Success End Condition	The product comparison overlay is successfully displayed on the page with accurate data.
Failed End Condition	The system fails to display the overlay or shows an error message indicating failure.
Primary Actors	End-User
Secondary Actors	System Administrator, Search Engine, E-commerce Website, Gemini API
Trigger	The user executes a product-related search query on a supported search engine.
Description / Main Success Scenario	1. User executes a product search on a supported search engine. 2. The extension identifies the action and extracts e-commerce product URLs from the search results. 3. The extension sends the list of URLs to the IPASS backend API. 4. The backend scrapes, prunes, and uses the Gemini API to extract structured data for each URL. 5. The backend returns a single JSON response with all product data to the extension. 6. The extension renders the Product Comparison Overlay on the user's screen with the received data.
Alternative Flows	2a. No valid e-commerce URLs are found in the search results. 2a1. The system displays a "No products found to compare" message. 3a. The IPASS backend API is unreachable or returns an error. 3a1. The extension displays a "Service currently unavailable" error message. 4a. A specific e-commerce website cannot be scraped or the Gemini API fails to extract its data. 4a1. The product for that specific URL is omitted from the final results.

Quality Requirements	1. The end-to-end process should complete in under 7 seconds. 2. The extension must not noticeably degrade the browser's performance.

Table 2.3.2.2: Case Description-02 Manage Extension Settings

Use Case	**Manage Extension Settings**
Goal	To allow the user to control the extension's behavior, such as enabling/disabling the service or clearing locally cached data.
Precondition	The IPASS extension is installed in the user's browser.
Success End Condition	The user's desired setting change is successfully applied and persisted.
Failed End Condition	The setting change is not applied or saved.
Primary Actors	End-User
Secondary Actors	None
Trigger	The user clicks on the IPASS extension icon in the browser's toolbar.
Description / Main Success Scenario	1. The user clicks the IPASS extension icon. 2. The extension's popup UI is displayed. 3. The user clicks the "Enable/Disable" toggle switch. 4. The extension's state is changed (e.g., from enabled to disabled). 5. The UI provides immediate visual feedback confirming the change.
Alternative Flows	3a. The user clicks the "Clear Cache" button instead of the toggle. 3a1. The extension clears any locally stored data. 3a2. The UI displays a temporary "Cache cleared" confirmation message.
Quality Requirements	1. The popup UI must open in less than 1 second. 2. All actions within the popup must provide immediate visual confirmation.

2.3.3 Activity Diagram

An **Activity Diagram** is a behavioral UML diagram that models the flow of control and the sequence of activities in a process. It is used to represent workflows of step-by-step operational activities and choices. For the IPASS project, the activity diagram is ideal for visualizing the entire end-to-end workflow of the core product comparison feature.

The diagram below will illustrate the flow of actions across different components of the system, represented **by swimlanes** for the End-User, Browser Extension, and Backend Server. It clearly shows the sequence of operations, decision points, and parallel activities involved in fetching and presenting product data.

Figure 2.3.3.1: Main Product Comparison Workflow

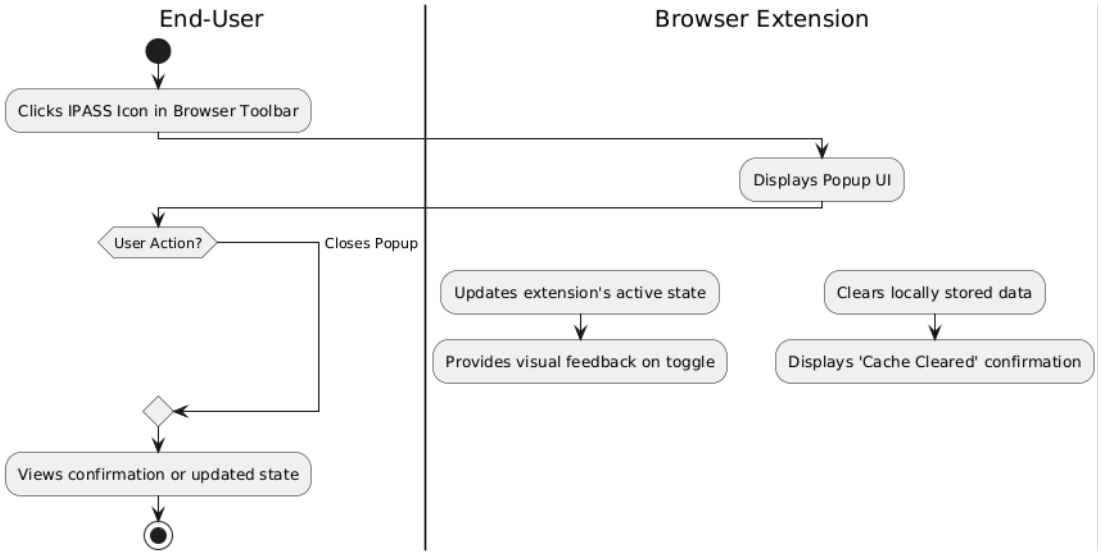
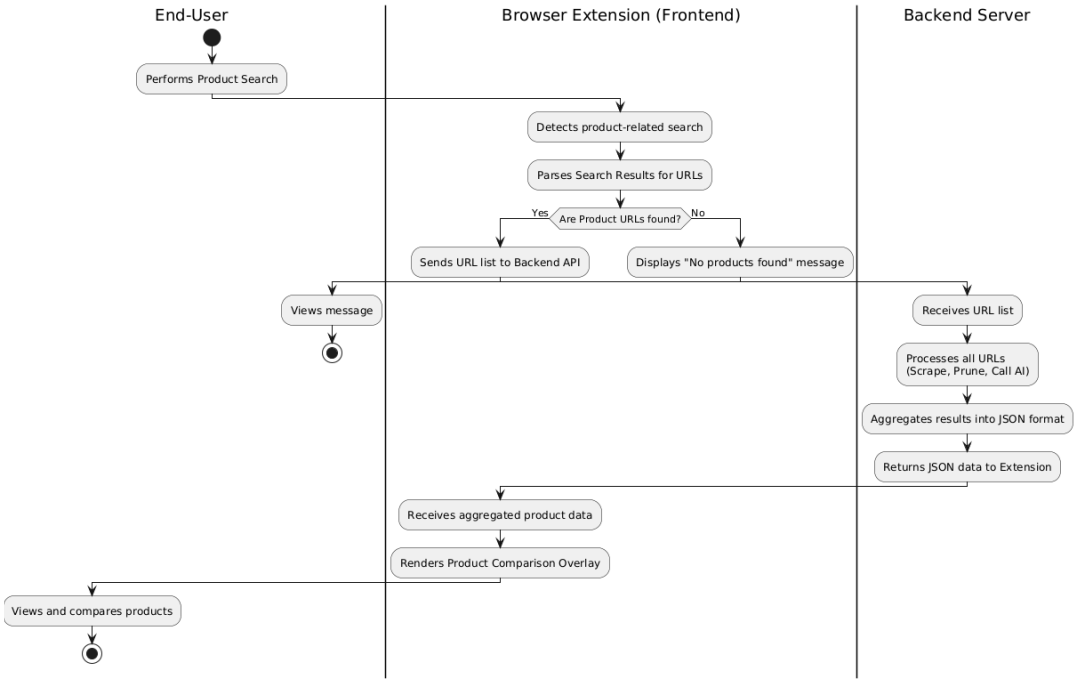


Figure 2.3.3.2 : Managing Extension Settings



As shown in the diagram, the process begins with the user's search action. The control flow moves from the browser extension, which handles search detection, to the backend server, which performs the heavy lifting of scraping and AI-based processing. The diagram highlights the hand-off of responsibilities between components and visualizes the main success pathway as well as potential branches for error handling.

2.3.4 Sequence Diagram

A Sequence Diagram is a UML interaction diagram that details how different objects and components of a system interact with each other over time. It focuses on the time-ordering of messages exchanged between objects, represented as vertical lifelines. This diagram is crucial for understanding the communication protocol between the frontend extension and the backend API.

The sequence diagram for IPASS will model the "Get Product Comparison" scenario. It will visualize the sequence of messages, such as API calls and data returns, between the End-User, the :BrowserExtension, the :BackendAPI, and the external :GeminiAPI.

Figure 2.3.4.1: Successful Product Comparison Sequence Diagram

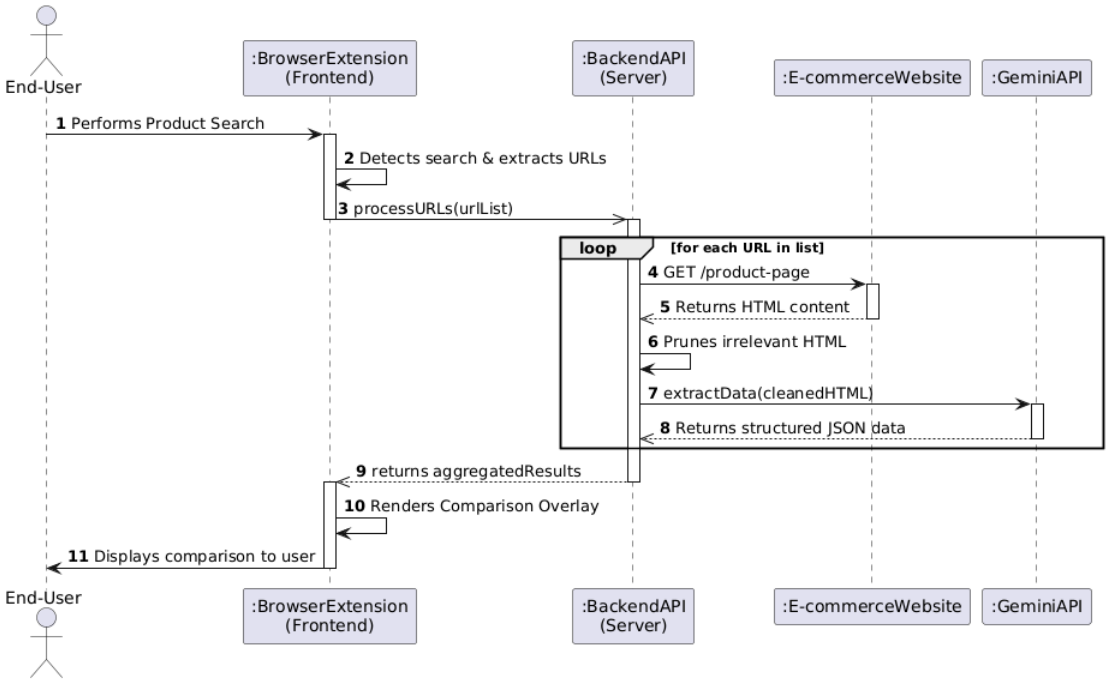
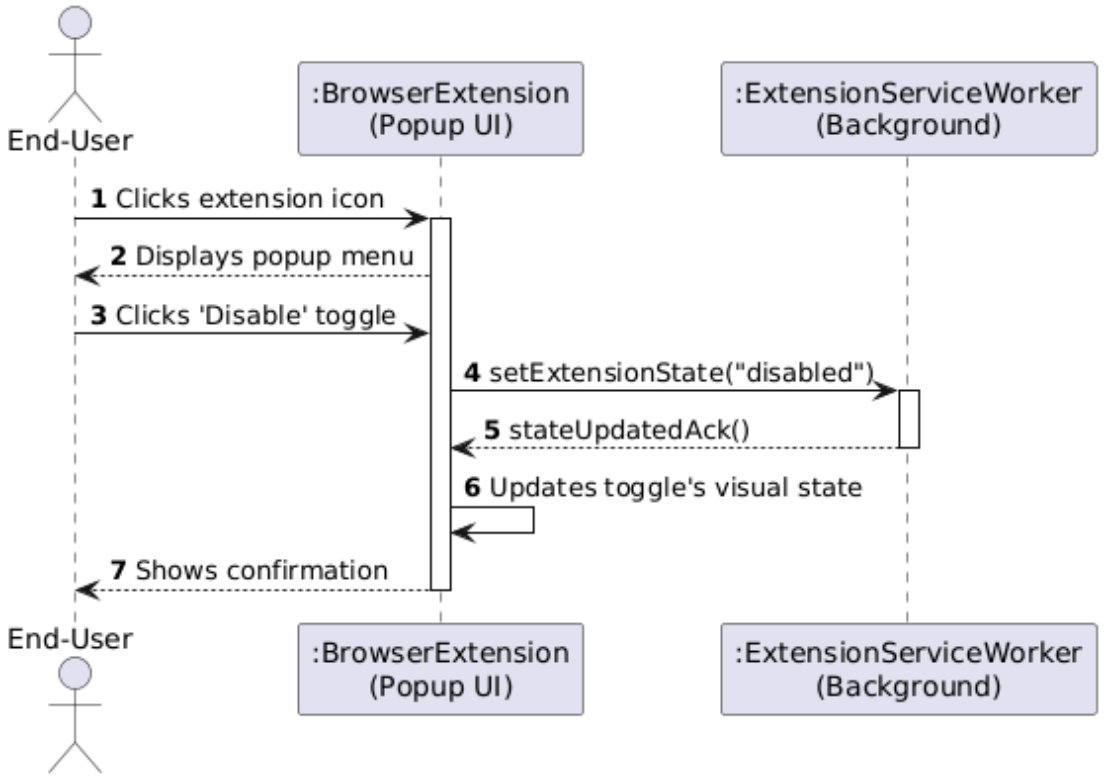


Figure 2.3.4.2 : Managing Extension Settings Sequence Diagram



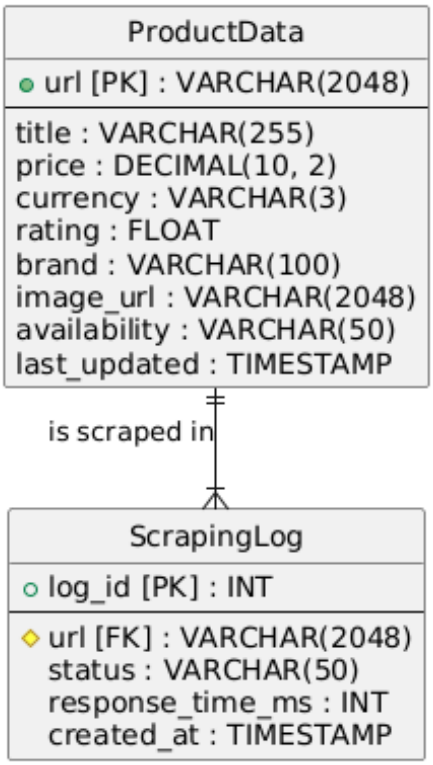
The diagram illustrates the chronological sequence of interactions. It begins with an event triggered by the user, which leads to a series of synchronous and asynchronous messages between the lifelines. This visualization clarifies the dependencies, timings, and the request-response nature of the client-server architecture, making it an invaluable tool for developers during implementation.

2.3.6 ER Diagram

An **Entity-Relationship (ER) Diagram** is a data model used to visualize the logical structure of a database. It shows the entities (tables), their attributes (columns), and the relationships that exist between them. While IPASS is primarily a real-time system, a database can be used for caching to enhance performance and reduce costs.

The ER diagram below proposes a simple schema for a backend caching mechanism. The primary goal of this database is to store the results of recent product data extractions, preventing the system from repeatedly scraping and processing the same URL in a short period.

Figure 2.3.6.1: ER Diagram for the Backend Cache



As illustrated, the central entity is ProductData, which stores the key information extracted from an e-commerce page. The product's URL serves as the primary key, ensuring uniqueness. A last_updated timestamp attribute is crucial for managing the cache's validity, allowing the system to periodically refresh stale data. This simple schema provides a significant performance and efficiency boost to the system.

2.4 Coding

The implementation phase translated the designs and requirements into functional code. The complete source code for the project, including the Python backend, the JavaScript-based browser extension, and all related configuration files, is managed with version control and is available for review in the project's code repository.

https://github.com/tuhin-cmd/IPASS

2.5 Chapter 2 Summary

This chapter provided the complete technical blueprint for the design and implementation of the IPASS system. It began by formally defining the detailed functional and non-functional

requirements, which serve as the criteria for the system's success. These requirements established what the system must do and the quality standards it must meet.

Following the requirements, the chapter detailed the Object-Oriented System Design using UML. A series of standard diagrams—including the Use Case, Activity, Sequence, Class, and ER diagrams—were presented to model the system from various perspectives, illustrating its architecture, behavior, and data structure. Finally, the chapter provided a reference to the coding appendix where the implementation of this design can be found. In essence, this chapter translates the project's initial concept into a structured and well-defined technical design.

Chapter 3

Software Testing

Software testing is a critical process to ensure the quality, reliability, and correctness of the IPASS system. The primary objective of this phase is to identify and rectify defects, validate that the software meets the specified requirements, and verify that the final product is stable and performs as expected. This chapter outlines the key features to be tested, the strategies employed, and the results of the system testing process.

3.1 Testing Features

The testing process focuses on validating the core features of the IPASS system. The key functional areas that require thorough testing are:

- **Search Detection and Activation:** Verifying that the extension correctly identifies product-related searches on all supported search engines (Google, Bing, DuckDuckGo, Yahoo) and activates the overlay appropriately.
- **Data Extraction Accuracy:** Ensuring the backend system, powered by the Gemini API, extracts product information (title, price, rating, brand) with a high degree of accuracy from various Bangladeshi and international e-commerce sites.
- **Backend API Performance:** Testing the responsiveness and reliability of the Python FastAPI server, including its ability to handle multiple concurrent requests.
- **User Interface (UI) Rendering:** Confirming that the comparison overlay and extension popup render correctly across different screen resolutions and browsers without visual glitches.
- Extension Controls: Testing the functionality of the popup menu, including the ability to enable/disable the service and clear the cache.

3.2 Testing Strategies

A multi-level testing strategy was adopted to ensure comprehensive coverage and quality. This approach involves different levels of testing, each with a specific focus.

- **Unit Testing:** This strategy involves testing the smallest individual parts of the codebase in isolation. In the backend, this means testing individual Python functions for data processing and API endpoint logic. In the frontend, it involves testing specific JavaScript utility functions. The goal is to verify that each unit of code works as intended.

- **Integration Testing:** This strategy focuses on testing the interaction and data flow between different integrated components. The most critical integration point for IPASS is the communication between the frontend browser extension and the backend API. Tests are designed to ensure that API requests from the extension are correctly handled by the server and that the responses are properly parsed and displayed by the frontend.
- **System Testing:** This involves testing the complete and fully integrated IPASS system as a whole. End-to-end tests are performed to validate that the entire workflow—from a user's search query to the display of the comparison overlay—functions correctly and meets all the functional and non-functional requirements defined in Chapter 2.

3.3 System Testing

This section documents the specific test cases executed during the System Testing phase. Each test case includes the steps to reproduce the test, the expected outcome, and the actual result observed, leading to a Pass/Fail status.

Table 3.3.1: Test Case Report for Product Comparison Workflow

Test Case: TC-001		Test Case Name: Product Comparison Workflow			
System: IPASS (Automated Product Aggregator and Summarizing System)		Subsystem: Core Feature - SERP Overlay			
Designed by: QA Team		Design Date: 04/08/2025			
Executed by: QA Team		Execution Date: 04/08/2025			
Description: To verify that the IPASS extension correctly detects product searches, fetches data, and displays the comparison overlay.					
Pre-condition: The user has the IPASS extension installed and enabled in Google Chrome with an active internet connection.					
Step	Action	Input Data (Search Query)	Response	Pass / Fail	Comment

1	Perform a valid product search.	"samsung galaxy s25 price in bd"	Overlay appears with product cards from sites like Daraz, StarTech, etc. Prices are in BDT (৳).	Pass	Core "happy path" functionality is successful.
2	Perform a non-product search.	"dhaka university"	IPASS overlay does not appear.	Pass	System correctly ignores non-relevant searches.
3	Perform a product search that yields no e-commerce results.	"buy adamantium shield"	Overlay appears with a "No products found to compare" message.	Pass	System gracefully handles cases with no valid product URLs.
4	Perform a search where some sites are unsupported.	"logitech mx master 3s"	Overlay appears with cards for supported sites. Unsupported sites are skipped without crashing the extension.	Pass	System shows resilience when encountering websites it cannot parse.
Post-condition: The browser remains stable after all tests. For successful cases, the overlay provides accurate and actionable information.					

Table 3.3.2: Test Case Report for Extension Management

Test Case: TC-002		Test Case Name: Extension Management & Controls			
System: IPASS (Automated Product Aggregator and Summarizing System)		Subsystem: Extension Popup UI & Controls			
Designed by: QA Team		Design Date: 04/08/2025			
Executed by: QA Team		Execution Date: 04/08/2025			
Description: To verify that the user can control the extension's behavior through the toolbar popup menu.					
Pre-condition: The IPASS extension is installed in Google Chrome.					
Step	Action	Input Data	Response	Pass / Fail	Comment

1	Click the extension icon, toggle service to 'Off', then perform a product search.	Search for "lapto p price in bd"	The toggle switch updates to 'Off'. The comparison overlay does NOT appear on the search results page.	Pass	Disabling the extension correctly prevents the core feature from running.
2	From a disabled state, click the extension icon and toggle service to 'On'. Perform product search.	Search for "lapto p price in bd"	The toggle switch updates to 'On'. The comparison overlay appears as expected on the search page.	Pass	Re-enabling the extension successfully restores its functionality.
3	Click the extension icon, then click the 'Clear Cache' button.	N/A	A temporary confirmation message like "Cache Cleared!" appears in the popup UI.	Pass	The cache clearing function is accessible and provides necessary user feedback.
Post-condition: The extension's state (enabled/disabled) persists correctly after changes. The popup UI is responsive.					

Test Case: TC-003	Test Case Name: Data Extraction Edge Cases
System: IPASS (Automated Product Aggregator and Summarizing System)	Subsystem: Backend Data Extraction & AI Processing
Designed by: QA Team	Design Date: 04/08/2025
Executed by: QA Team	Execution Date: 04/08/2025
Description: To verify the accuracy and correct formatting of extracted data, especially in non-standard or edge cases.	
Pre-condition: The IPASS system is fully operational. Tester can inspect the JSON response from the backend for detailed verification.	

Step	Action	Input Data (Search Query)	Response	Pass / Fail	Comment
1	Search for a product that is currently on sale (with a discount).	Find a specific product on Daraz listed with a discount.	The overlay correctly extracts the final discounted price, not the original crossed-out price. Price is formatted as ৳X,XXX.	Pass	AI correctly identifies the most relevant current price.
2	Search for a product listed as "Out of Stock".	Find a specific, known out-of-stock item on a supported site.	The product card in the overlay correctly displays "Out of Stock" as the availability status.	Pass	Availability status extraction is functional for non-standard cases.
3	Search for a newly listed product that has no user ratings.	Find a new product on a local e-commerce site.	The rating field on the product card is left blank or displays "No ratings" instead of an error or "0".	Pass	System gracefully handles missing optional data points like ratings.
4	Search for a product with a very long, complex title.	"Apple iPhone 15 Pro Max 256GB Natural Titanium"	The overlay displays the full, accurate product title without incorrect truncation or modification.	Pass	Title extraction is robust and captures detailed product names.
Post-condition: The backend consistently provides well-formatted JSON data, even when the source HTML from e-commerce sites is complex or missing information.					

Chapter 4

Deployment and Maintenance

In this chapter, we'll cover our plan for getting the IPASS system up and running for users. We'll also go over how we intend to keep it working well and make it better over time. Our whole project was guided by an Agile mindset—building in small steps and constantly listening to feedback. That same thinking carries over into this next phase, giving us the flexibility to respond to user suggestions and any technical hurdles we encounter.

4.1 Software Release Life Cycle (SRLC)

To get from a finished product to a live service, we broke the process down into a few clear stages.

1. Building and Testing First, we built the application's features in short, focused sprints. As we mentioned back in Chapter 3, every new piece of code went through rigorous testing to make sure it was reliable, worked as expected, and met our quality goals.

2. Going Live Getting the system out to the public meant tackling two different pieces at once: the server that powers everything and the browser extension that users interact with.

- **The Backend:** Our Python server is bundled inside a Docker container, which is like a self-contained package that guarantees it runs the same way no matter where it's hosted. We host this container on a cloud service like Heroku or Vercel because they can handle growth and manage the technical side for us. We've also set up an automated workflow with GitHub Actions, so whenever we have an update, it gets tested and deployed automatically without manual fuss.
- **The Frontend:** The browser extension is much simpler. We package all the files into a .zip archive and send it to the Chrome Web Store. They have a review process, and once they give the green light, it’s live for anyone to download and use.

3. Life After Launch Once IPASS is out in the wild, our focus shifts to making sure it stays healthy and useful for the long haul. This involves a few key activities:

- **Keeping Watch:** We'll keep a close eye on the server's health—checking its uptime, speed, and any errors that pop up. Tools like Sentry or UptimeRobot give us a heads-up on potential trouble before it becomes a real problem for users.
- **Fixes and Updates:** When we find bugs—either from our monitoring or from user reports we'll prioritize them and get fixes out the door. Server updates are pushed through our automated system, while changes to the extension go through the Chrome Web Store.
- **Future Improvements:** Most importantly, we're sticking with our Agile approach. We plan to keep developing new features in cycles, letting the IPASS system grow and improve based on what our users are telling us and where the technology takes us.

Chapter 5

User Manual

A Guide to Using the IPASS Extension

This guide will walk you through setting up and using the IPASS browser extension to make your online shopping simpler and more informed.

5.1 Adding IPASS to Your Chrome Browser

Getting the IPASS extension is quick and easy. Just follow these steps:

1. Open your Google Chrome browser.
2. Go to the Chrome Web Store and search for **"IPASS: AI Product Summarizer,"** or use the direct link to our page.
3. Click the **"Add to Chrome"** button.
4. A small window will ask for your confirmation. Click **"Add extension"** to finish.
5. You will now see the IPASS icon in your browser's toolbar, ready to use.

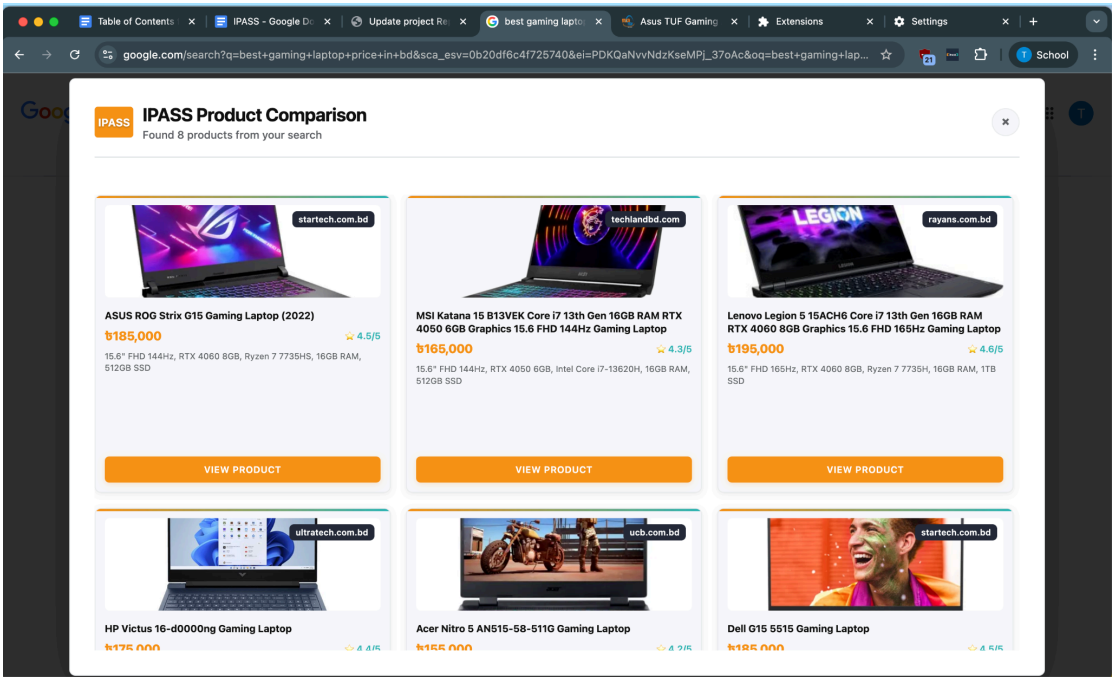
5.2 How to Use IPASS When Shopping

IPASS works automatically in the background to help you. There are no extra steps to turn it on.

1. **Search for a Product:** Go to a search engine like google.com and type in what you are looking for (for example, "wireless headphones" or "Samsung TV price").
2. **See Instant Comparisons:** When the search results appear, the IPASS window will slide out on the side of the page. It automatically collects and displays information about the products you searched for.
3. **Compare Products Easily:** In the IPASS window, you can instantly compare prices, user ratings, and brands from different online stores, all in one clean view.

4. **Visit the Product Page:** If you see something you like, simply click on its card in the IPASS window. This will open the product's original page on the e-commerce site in a new tab for you to see more details or make a purchase.

Figure 5.2.1 : Performed product comparison



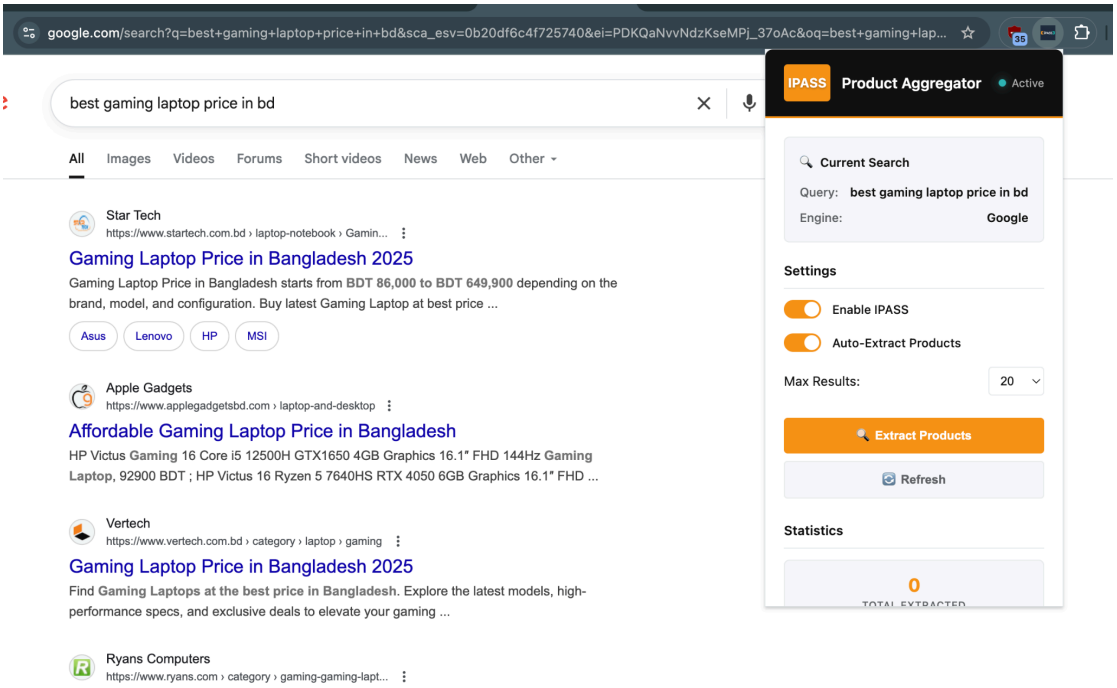
5.3 Managing Settings

You can easily manage the extension's settings from the popup menu.

1. Click the IPASS icon in the Chrome toolbar to open the popup.

Use the main toggle switch to enable or disable the service at any time.

Figure 5.3.1 : Settings popup of the extension



Chapter 6

Project Summary

This final chapter provides an overview of the IPASS project. We will review its goals, successes, and the difficulties encountered along the way. We will also discuss the system's current shortcomings and suggest directions for future improvements.

6.1 Project Summary

The main purpose of the IPASS project was to solve a common problem for online shoppers: feeling overwhelmed and wasting time. To do this, we built a smart browser extension that automatically finds and compares products. The tool works with major search engines, pulls key product information from shopping websites with the help of the Gemini API, and organizes it for easy side-by-side comparison.

Among our main successes, we built a reliable technical foundation, successfully used a powerful AI to pull information accurately, and designed a simple interface that makes the shopping process much smoother. Our biggest hurdle was dealing with the constantly changing layouts of different e-commerce sites. We overcame this by using a flexible AI, which worked much better than older, more rigid methods that can break easily.

6.2 Limitations

While the project met its main goals, it does have some weaknesses:

- **Limited Website Support:** The extension is designed to work with a select group of popular Bangladeshi and international shopping sites. It might not work properly on others.
- **Reliance on Other Services:** The tool depends on services like Google Search and the Gemini API to function. Any changes they make to their systems could cause our extension to stop working correctly.
- **No Saved History:** The current version does not save your product comparisons. Once you close the browser, the information is gone.

6.3 Future Scope

The IPASS system provides a solid base for many new features. Looking ahead, we could:

- **Add Price Watches and Alerts:** We could add a feature that tracks product prices and lets you know when something you want goes on sale.
- **Create Personal Accounts:** Adding user accounts would let people save their comparison lists and access them from different computers.
- **Support More Browsers:** We could create versions for other popular browsers, like Mozilla Firefox and Microsoft Edge.
- **Use AI for Smarter Review Analysis:** The AI could be taught to read through customer reviews and give a quick summary of what people generally think about a product.
- **Develop a Mobile App:** A dedicated mobile app would let users compare products while they are on the go.

6.4 Conclusion

In summary, this project successfully developed IPASS to solve the common problem of inefficient online shopping comparisons. We created an AI-powered browser extension that demonstrates a working solution for gathering product data automatically and improving the user's experience.

The system's foundation proved to be strong, and using the Gemini 2.5 Flash API was a highly effective way to get accurate information from many different sites. The final product works as intended and provides a smooth and intuitive experience for the user.

Ultimately, this project is more than an academic exercise; it's a meaningful tool with real-world value. It highlights the huge potential of modern AI models to create smarter and more user-friendly web tools. The IPASS system is a solid foundation for future work and more advanced features in e-commerce technology.

References

Fowler, M. (2003). *UML distilled: A brief guide to the standard object modeling language* (3rd ed.). Addison-Wesley Professional.

Google. (2025). *Gemini API documentation*. https://ai.google.dev/docs

Google. (2025). *Manifest V3*. Chrome for Developers. https://developer.chrome.com/docs/extensions/mv3/intro/

Harris, T. (2023, May 15). *RESTful API design*. Microsoft Azure Architecture Center. https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design

Mitchell, R. (2018). *Web scraping with Python: Collecting more data from the modern web* (2nd ed.). O'Reilly Media.

Python Software Foundation. (2025). *The Python language reference* (Version 3.9). https://docs.python.org/3.9/reference/index.html

Richardson, L. (2025). *Beautiful Soup documentation*. https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Tiangolo, S. (2025). *FastAPI*. https://fastapi.tiangolo.com/

211-35-710

ORIGINALITY REPORT

11%	9%	3%	10%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Daffodil International University Student Paper	3%
2	dspace.daffodilvarsity.edu.bd:8080 Internet Source	1%
3	Submitted to National School of Business Management NSBM, Sri Lanka Student Paper	1%
4	Submitted to Capella University Student Paper	<1%
5	Submitted to University of Birmingham Student Paper	<1%
6	Submitted to University of New South Wales Student Paper	<1%
7	Submitted to Vidyalankar Institute of Technology Student Paper	<1%
8	uh-ir.tdl.org Internet Source	<1%
9	Submitted to CSU, San Jose State University Student Paper	<1%
10	Submitted to Institute of Space Technology Student Paper	<1%
11	dspace.bracu.ac.bd Internet Source	<1%

12	arxiv.org Internet Source	<1%
13	Submitted to kkwagh Student Paper	<1%
14	www.igi-global.com Internet Source	<1%
15	qwirey.com Internet Source	<1%
16	www.cs.vu.nl Internet Source	<1%
17	www.diva-portal.org Internet Source	<1%
18	Pushpa Choudhary, Sambit Satpathy, Arvind Dagur, Dhirendra Kumar Shukla. "Recent Trends in Intelligent Computing and Communication", CRC Press, 2025 Publication	<1%
19	Submitted to Universidad TecMilenio Student Paper	<1%
20	pure.uva.nl Internet Source	<1%
21	Submitted to Blue Mountain Hotel School Student Paper	<1%
22	Submitted to Southern New Hampshire University - Continuing Education Student Paper	<1%
23	Submitted to University of Hertfordshire Student Paper	<1%
24	Submitted to University of Wales Institute, Cardiff	<1%

	Student Paper	
25	horace-rexus.de Internet Source	<1 %
26	Bui, Hoang Duc. "Assessment of Privacy Risks in Mobile and Web Applications/Services", University of Michigan, 2022 Publication	<1 %
27	Submitted to Universiti Malaysia Pahang Student Paper	<1 %
28	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	<1 %
29	docs.geolitix.com Internet Source	<1 %
30	etd.aau.edu.et Internet Source	<1 %
31	cashgenerator.co.uk Internet Source	<1 %
32	de Almeida Mourinho, Nuno Gaspar Afonso. "ForensicVM: Developing a Virtualisation Plugin for Autopsy Software: Challenges and Solutions in Acquiring Digital Evidence from Virtualised Forensic Images", Instituto Politecnico de Beja (Portugal) Publication	<1 %
33	demo.plastproinc.com Internet Source	<1 %
34	gecgudlavalleru.ac.in Internet Source	<1 %
35	manual.calibre-ebook.com Internet Source	<1 %

36	ntnuopen.ntnu.no Internet Source	<1%
37	Paul Fischer, Lutz Schweikhard. "Multiple active voltage stabilizations for multi-reflection time-of-flight mass spectrometry", Review of Scientific Instruments, 2021 Publication	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		