SP-17 Plug In for Browser

Final Report

CS 4850 - Section 03 –Spring 2025

Nathan Lathrop, Rachel Anderson, Hunter Lee, Austin Wood

Sharon Perry

April 27, 2025

Website: https://sp-17-plugin-project.github.io/BrowserPlugIn/

GitHub: https://github.com/SP-17-PlugIn-Project/BrowserPlugIn/tree/main

STATS AND STATUS	
Lines of Code	422
Components/Tools	7
Hours Estimate	274
Hours Actual	300

Contents

Introduction

Browser Extensions like Honey that provide coupons for the user are widely used, but many times are secretly predatory in nature. The purpose of this project is to create a browser that provides the best available coupons for some of the largest online shopping domains without employing any predatory practices against affiliate marketers or customers.

The scope of this project includes planning, designing, developing, testing, and implementing a chrome-based browser extension that can store coupons available for some of the most popular online shopping domains, retrieving said coupons for the user when requested, and removing any nonfunctional coupons form the service, where users feedback will determine whether code is still viable. The coupons will be gathered both via web scraping done via the team, as well as through an interface allowing users to input new coupons.

Requirements

Project Scope

The finalized browser extension will automatically search for available coupons at checkout pages, apply the best possible coupon to maximize potential savings, provide an interface within a Chrome sidebar, and allow users to rate the effectiveness of the applied coupons. To ensure proper functionality, a self-created website will serve as a prototype environment for testing coupon detection and retrieval before applying the extension to real retail websites.

Initially, coupon data will be stored in a CSV file for early development and testing. The extension will migrate to Firebase Firestore. Firestore will enforce access control, ensuring that only authenticated users can submit and modify coupon data.

The extension will also implement rate-limiting to prevent excessive automated requests when retrieving coupon data to ensure compliance with website policies. The extension will follow ethical web scraping practices and respect robots.txt, where applicable.

Definitions and Acronyms

- **Chrome Extension**: A small software program that enhances the Chrome browser's functionality.
- **Coupon**: A digital code that provides discounts on online purchases.
- **Database**: A system used to store and retrieve coupon information.
- **UI**: User Interface.

Assumptions

- The extension will work on a website where we have some API access.

- The extension will not use or contain any personal information from the user, which means there is less need for security.
- The extension will receive reliable feedback from users to maintain the coupon database.
- The extension will have access to a database created and maintained by our team for both coupons and user login credentials.
- The extension will follow Chrome Web Store guidelines.

Design Constraints

Environment

- Developed using JavaScript, HTML, and CSS.
- Runs on Google Chrome (latest versions supported).
- Interfaces with e-commerce websites and checkout pages.

User Characteristics

- **General Shoppers**: Users looking for discounts while shopping online.
- **Tech-Savvy Users**: Users that are comfortable with browser extensions and online savings tools.
- **Retailers**: Indirect users who may provide exclusive coupons for marketing.

System

- It will retrieve coupons from a proprietary database containing:
    - Coupon codes.
    - Associated retailer information.
    - Effectiveness ratings.
- A secondary table in the database will store user login information.
- It will interact with e-commerce checkout pages to apply coupons.

Functional Requirements

Login

- Users can create an account.
- Login using a username and password.
- Password recovery option via email.
- User login data will be stored in the database.

Authentication (Optional for Users)

- Email-based authentication.

Display Sidebar Menu

- **Check for Coupons**: Scan for available discounts.
- **Use Coupon**: Automatically apply the best coupon.
- **Rate Coupon**: Allow users to rate applied coupons.
- **Exit Sidebar**: Close the extension interface.

Navigation for Database Access

- Fetch coupons and user credentials from database.

Non-Functional Requirements

Security

- Basic encryption for login credentials.
- Secure HTTPS communication for data retrieval.

Capacity

- Database storage capacity: 5GB (scalable as needed).

Usability

- User-friendly Chrome sidebar UI (<25% of screen space).
- Keyboard shortcuts for quick access.

External Interface Requirements

User Interface Requirements

- Chrome sidebar that occupies less than 25% of screen space.
- Intuitive and minimalistic design.
- Clear buttons for coupon functions.
- Keyboard shortcuts for quick access.

Software Interface Requirements

- Connects with Chrome's API for extension functionality.
- Interfaces with online retailer checkout pages.

- Communicates with the coupon database.
- The extension will first interact with a self-created prototype website for initial testing and later extend functionality to live retail websites.
- Coupon data will first be stored in a CSV file before migrating to Firestore, where user authentication will be applied.

Communication Interface Requirements

- Email support for password recovery.
- Browser-based communication between extension and online shopping websites.

Analysis/Design

Design Considerations

Assumptions and Dependencies

- The extension will run on the Google Chrome browser.
- Users will have an active internet connection.
- The backend database will be implemented and maintained for storing coupon data and user login credentials.
- The extension will interact with e-commerce websites to apply coupons during checkout.
- Basic security measures will be implemented to protect user data.

General Constraints

- The extension must comply with Google Chrome Web Store policies.
- It should not significantly impact browser performance.
- Must be compatible with major online retailers.
- The system must first store coupon data in CSV and later migrate to Firestore.
- It should have an optional authentication method for user accounts.
- Data privacy and security regulations must be followed.
- Limits will be placed on API calls to prevent excessive data requests.
- Performance must be optimized to deliver real-time coupon application.

Development Methods

The project will follow the Feature Driven Development method, which works by building out a list of features to be designed for the product and then completing each feature 1 by 1 in order of necessity for the final product.

Architectural Strategies

- **Programming Languages and Frameworks**: The frontend will be developed using JavaScript, HTML, and CSS, while the backend will use the Node.js library for handling API requests. The database will use CSV.
- **Client-Server Architecture**: The system will follow a client-server architecture, where the Chrome extension serves as the client, interacting with the backend API to retrieve and apply coupons.
- **Database Management**: A relational database will store coupons, their effectiveness, associated websites, and user login information. This ensures structured data organization and efficient retrieval.
- **User Interface**: The extension will provide a user-friendly interface with minimal disruption to the browsing experience. It will notify users about available coupons through a pop-up interface.
- **Error Handling and Recovery**: The system will implement logging mechanisms to track failures, ensuring robust error handling for API failures, database connection issues, and invalid coupon applications.
- **Security and Data Protection**: The database will store user credentials securely with encryption. Secure HTTPS connections will be enforced for all client-server communications.
- **Concurrency and Performance**: The backend will support multiple concurrent API requests using Node.js.

System Architecture

The system is structured into three main subsystems, each responsible for a specific set of functionalities. These components work together to deliver seamless coupon retrieval and application:

1. **Chrome Extension Frontend**: This component is responsible for interacting with the user and scanning e-commerce websites for applicable coupons. It sends API requests to the backend to retrieve available coupons and displays the best options to the user. A popup UI will be displayed where users can submit and retrieve coupons. This component will also detect promotional codes on checkout pages.
2. **Backend Server**: This subsystem handles business logic, processes API requests, and communicates with the database. It retrieves and evaluates coupon data, manages authentication, and ensures secure communication with the frontend.
3. **Database**: The database stores critical application data, including available coupons, their effectiveness ratings, user login information, and historical coupon usage. This ensures persistent and structured storage for optimized retrieval. A CSV file will be used in early stages of development and testing.

The components interact as follows:

- The **frontend** detects when a user is on a checkout page and sends a request to the **backend** for relevant coupons.
- The **backend** queries the **database** to retrieve the most effective coupons for the given retailer and sends the response back to the **frontend**.

- The **frontend** displays the available coupons, allowing the user to select one to apply.
- If the user is logged in, their coupon usage data is updated in the **database** to improve future recommendations.

The system decomposition ensures that each component is responsible for a distinct function, promoting modularity and maintainability. The frontend focuses on user interaction, the backend manages logic and communication, and the database provides persistent storage. This separation of concerns enhances system scalability and allows for independent updates and improvements to each component.

Detailed System Design

Classification

- The system is divided into three primary components: **Chrome Extension Frontend**, **Backend API**, and **Database**.
- The frontend consists of JavaScript modules and UI components.
- The backend comprises API routes and business logic implemented in Node.js.
- The database is a structured relational database.

Definition

- **Chrome Extension Frontend**: Responsible for detecting applicable coupons, interacting with the user, and communicating with the backend API.
- **Backend API**: Handles authentication, processes coupon-related logic, and provides data to the frontend.
- **Database**: Maintains persistent storage of coupon and user-related data.

Constraints

- **Chrome Extension Frontend**:
  - Must comply with Chrome Web Store policies and browser security standards.
  - Should not cause performance degradation or interfere with the user's browsing experience.
- **Backend API**:
  - Must efficiently handle multiple concurrent API requests.
  - Needs to enforce strict security measures, including HTTPS and authentication mechanisms.
- **Database**:
  - Must support fast read/write operations to ensure real-time coupon retrieval.
  - Should be optimized for scalability and future data expansion.

Resources

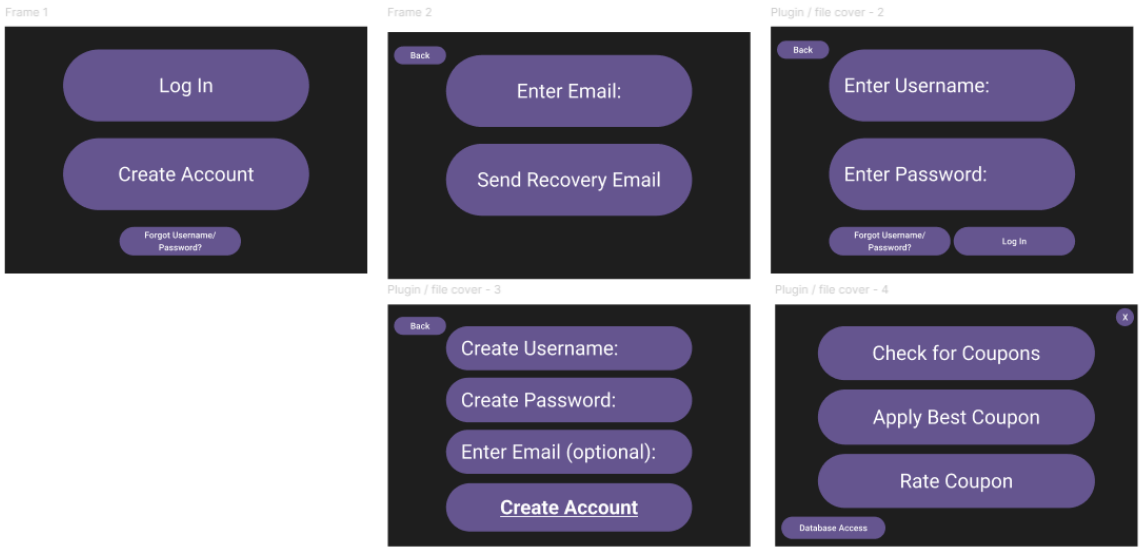
- **Frontend**: Uses browser resources such as DOM manipulation and local storage for temporary data caching.

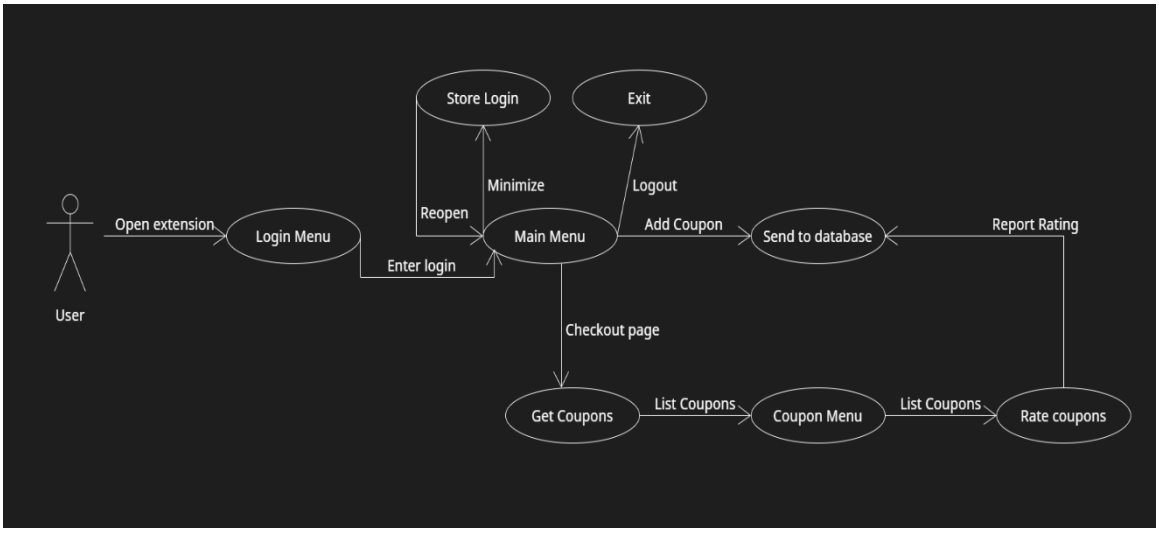
- **Backend**: Requires server hosting, database connections, and secure API endpoints.
- **Database**: Utilizes structured storage with indexing for fast query execution.
- **Security Measures**: Implements encryption for sensitive user data and access control mechanisms.

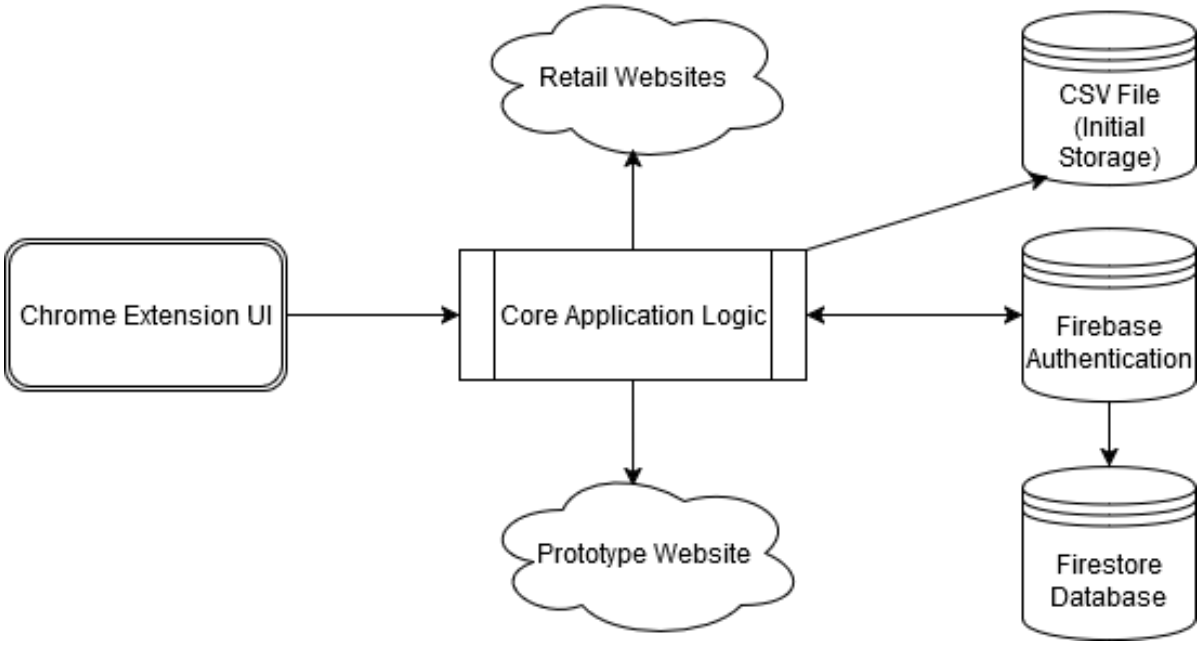
Interface/Exports

- **API Endpoints**:
  - /login: Authenticates users and establishes a session.
  - /getCoupons: Retrieves relevant coupons for a given website.
  - /applyCoupon: Applies the most effective coupon during checkout.
  - /rateCoupon: Allows users to rate the effectiveness of applied coupons.
- **Frontend Components**:
  - Popup UI for displaying available coupons.
  - Background script for detecting checkout pages.
  - Content script for interacting with e-commerce websites.
- **Database Tables**:
  - users: Stores user login credentials and session data.
  - coupons: Contains coupon details, including expiration and retailer information. Also stores frequency of coupon usage and whether or not the coupon was able to be applied.

Mockup/Diagrams







Glossary

- **Chrome Extension**: A software module that enhances browser functionality.
- **API**: Application Programming Interface, a set of rules for software interaction.
- **Backend**: The server-side logic handling data processing and storage.
- **Database**: A structured system for storing and retrieving data efficiently.

Development

Outline

Plugin

Account Creation

- User creates an account with a username, password, and an optional email to link to the account. Users are also able to use the extension under a ‘Guest’ profile, however doing so will restrict their ability to use certain features.

Login

- The user can sign into the extension using their account username and password. If the user has forgotten this information, and supplied an email address connected to their account, they can use this to receive an email containing their account username and password. Logging in gives the user access to all subcategories in the coupons category.

Coupons

Check for Coupon

- The user clicks on the search function button, which will then search the list of stored coupons we have to find any that are related to the website they are using, then presents the user with the coupon that has the highest discount available for that store. If the coupon does not work, the user will have the option to report it as nonfunctional, at which point the extension will make a note of this and present the user with the next best coupon, repeating until one works. Optionally, we considered adding in the ability for the extension to attempt automatically entering in the coupon for the user, allowing for the entire process to be automated.
- Currently, the user can enter in the name of the website that they are using, and the extension will provide the user with all coupons for that store for the user to enter themselves. Other features listed above are in the process of being implemented.

Add Coupon to Database

- The user has the option to provide the extension with a coupon in case it is not in the system already. Once the user has provided the coupon, the extension will store it as a possible coupon for the future. This coupon will be added to the database of other coupons later depending on certain factors, such as being provided by multiple different users, or approved by a member of the team.
- Currently, coupons are being stored in a CSV file that is taking the position of a database temporarily. Due to built-in restrictions, the program is not able to automatically update the CSV file. As such, if a user adds in coupon codes, they are temporarily saved for that specific instance on their device and will not be saved within the extension files. Once a database has been established, we will revisit this function to determine if it is still feasible and something we wish to implement.

Input Coupon

- The user, upon receiving a code from the program for the online retailing website, will be able to copy the code from the extension, and manually place the code into the coupon section of the retail website, manually checking if the code worked. Optionally, we considered having the extension automatically enter the code for the user and decided it would be extra to be implemented only if everything else was complete first.
- Currently, the extension, once provided with a website name, provides the user with the codes available for said website. Certain functionality, such as providing a copy to clipboard function, are not yet implemented.

Save Coupon to Account

- The user can select certain coupons and save them to a personalized section full of their saved coupons. This section allows for the user to be granted quick access to their most used coupons without having to go through the normal coupon process. These coupons can be copied by the user via clipboard.
- As we have yet to implement user accounts, this function is not yet functional. The personal coupon page exists within the extension and will be updated once user accounts have been implemented.

Delete Coupon

- If a user finds that a coupon does not work, as stated above, they can provide this information to the extension. Once they have done so, the extension will mark that coupon as having not worked for a user. If a certain criterion is met, such as enough accounts have stated that coupon as nonfunctional, then the program will remove the coupon from the database.
- Since we are temporarily still using a CSV file to store coupons, this function has yet to be implemented. Once a database has been established, we will revisit this function to determine if it is still feasible and something we wish to implement.

Logout

- The user can press a log-out button in the extension. Doing so closes their current session of the extension and additionally signs the user out of their account within the extension, requiring them to sign in again the next time they utilize the extension.
- Currently, the log out button is nonfunctional for both intended features. Once a database has been established and user accounts have been implemented, we will work on updating this feature.

Scrape Coupons

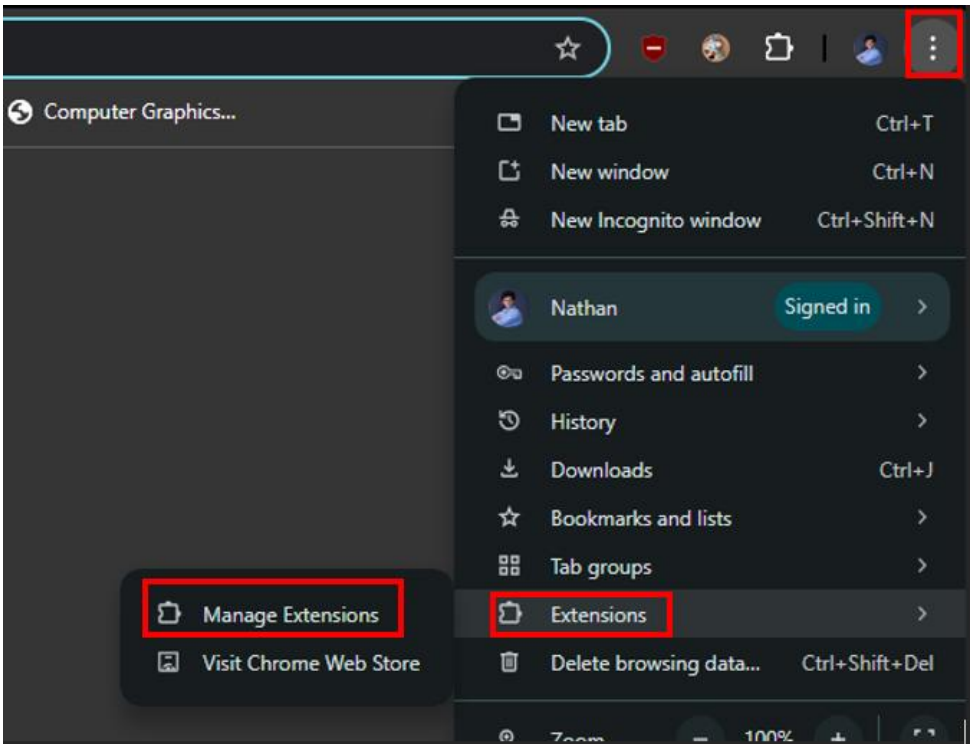
- To provide the extension with a large starting set of coupons, as well as to allow the extension to update with new coupons, we will implement a scrape function that allows for the extension to scrape coupons from multiple websites to then add to the database of coupons. This scraping function can be run periodically to keep the extension updated.
- Currently, as we are unable to automatically update the CSV file using the program, the extension has a temporary button that runs the scrape command. This downloads a new CSV file containing the information scraped by the function. The scrape function is also not fully implemented, as it is likely that each website stores their coupons using different methods and file names, meaning that the scraping function must be fine-tuned for each website it scrapes.

Database Connection

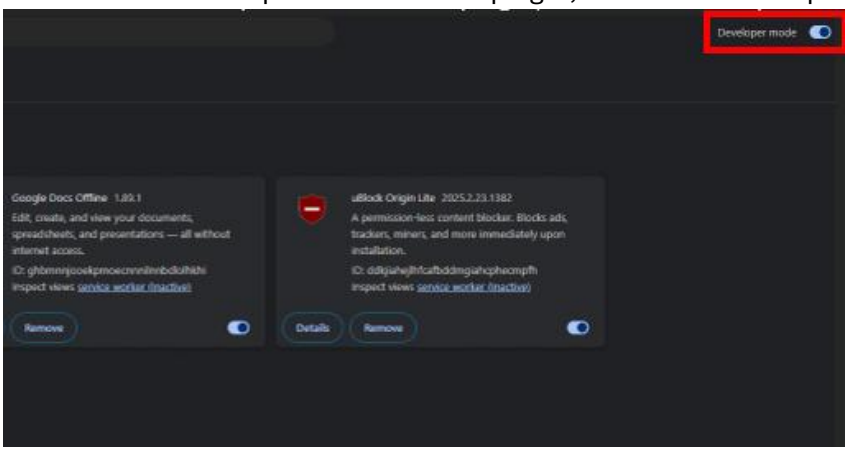
This project uses a Firebase Firestore database. This database stores both the coupons and the user login credentials. Once the user has connected their extension to the database, they are able to register an email and password, which will be automatically saved in the database and can then be used on any device that has the extension. The coupons are manually inputted by the team into the database. Currently, only members of the development team have permission to add or remove coupons from the database. The database branch is currently only usable for developers, as we have been unable to implement the module required for automatic connection to the database.

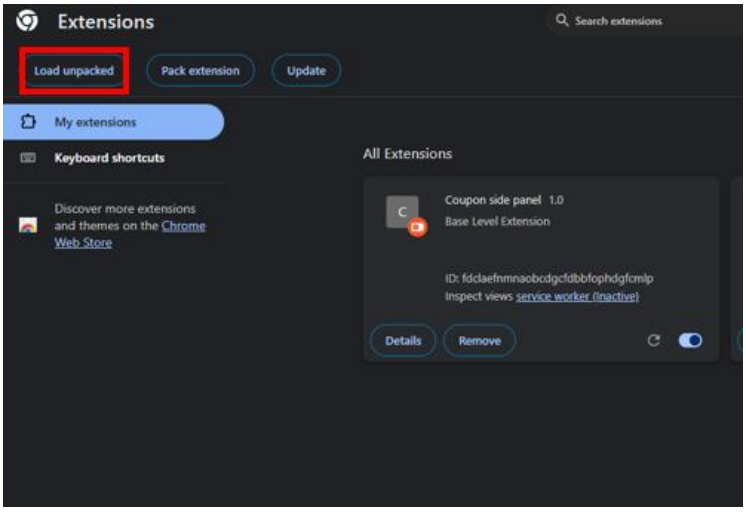
Set Up Steps (Main Branch)

- Download the project from the GitHub page: https://github.com/SP-17-PlugIn-Project/BrowserPlugIn
- Open google chrome, click on the menu, hover over extensions, and click on manage extensions

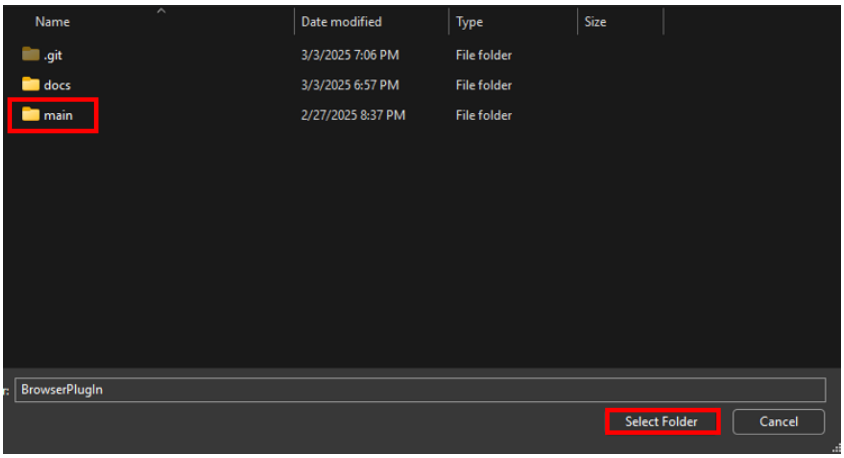


- Click to enter developer mode in the top right, then click load unpacked in the top left

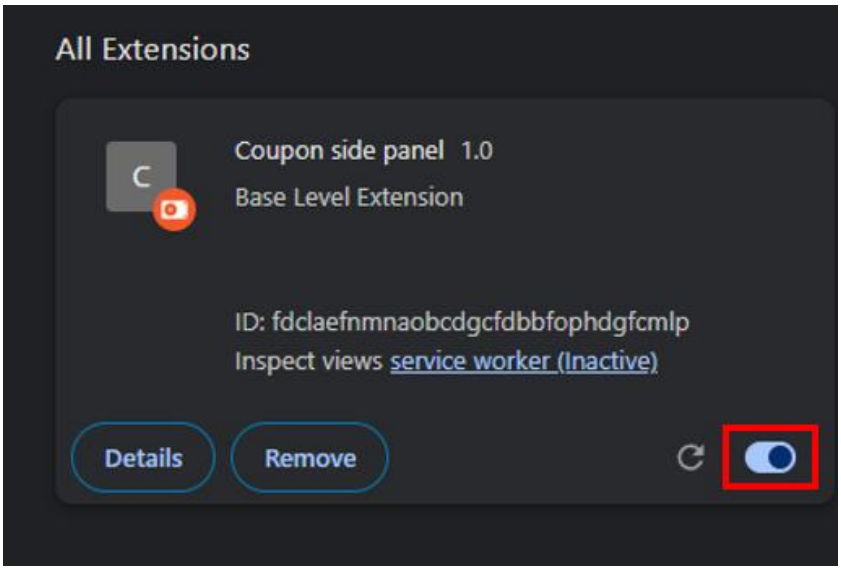




- Select the main folder from the downloaded file



- Make sure the extension is turned on



- Click the extension icon in the top right to use

Test (Plan and Report)

Software Test Plan

Our extension will be tested by having team members Rachel and Nathan downloading the most recent version of the extension and then loading it into google chrome in developer mode as a normal extension. Next, these members will use the browser as if they were a customer- they will go through the multiple pages of the coupon and will determine whether each section of the extension is working properly as defined in the project scope. These Results will be logged in the Software test report, and the team members will log their findings into a separate document to share with the other team members in the next meeting. This will test account creation, account login, the 'My Coupons' sidebar page, the coupon search page, the database capabilities, and user capabilities. The extension will be testing in the google chrome browser as a sidebar. The failed requirements in testing were all expected to occur. We have yet to implement a password recovery email, and the ability to scrape coupons from websites has not yet been properly implemented. We are working on implementing a way to scrape coupons from websites, but the process is not something easily automated, and as such might not be possible to implement within the time frame. For the user's ability to rate and suggest coupons, we are considering if it is viable to work on implementing this, as doing so opens potential vulnerabilities in our database.

Software Test Report

Requirement	Pass	Fail	Severity
Create account	x		Low
Login	x		Low
Password recovery		x	High
Logout	x		Low
My Coupons			
Save Coupons	x		Low
Copy Saved Coupons	x		Low
Delete Saved Coupons	x		Low
Search Coupons			
Can Search for Websites	x		Low
Can Scrape Website		x	High
Can Add Coupons	x		Medium
Will Display Added Coupons	x		Medium
Database			
Shows coupons in DB	x		Low
Connects to DB	x		Low
User can rate coupons		x	High
User can suggest coupons for DB		x	High

Version Control

This project was created using GitHub and its built in version control features.

Conclusion

This project was created for the purpose of making a coupon browser extension that focuses on ethical coupon aggregation and applications for online retailers. Over the semester, we have created an extension prototype that allows users to search for, apply, and save coupons for future use. The current system connects to a firebase database that stores both user ID information and coupon information.

Certain features, such as user input coupons, are not implemented, however, the foundational system and functionality has been implemented and is able to be improved upon at a future date. For this project, the team has put forward roughly 300 hours of collective effort and have reached roughly 80% completion of all planned features. The remaining work includes

allowing users to input coupons, scraping more coupons to add to the database, and finalizing the database connection to the extension.

Appendix

All team members studied and read the information for developing a web browser extension for chrome. This information was provided from the following website: https://developer.chrome.com/docs/extensions/get-started/tutorial/hello-world