CLAWS

Bobcat C.l.a.w.s

College **L**ifestyle **A**ffordable **W**ares **S**earch

An Online Price Grabber for Students

TABLE OF CONTENTS

DATE!

Group Members: Abigail De Rousselle, Dillon Hughes, Dustin Bruce, Garrett Dipalma, Jawad Abu-Gabal, Jess Stevenson, Kevin Garcia, Rayyan Khan, & Rebekah Hardsand

Background:

BETTER NARRATIVE

A website that allows users to search for and compare products sold at local businesses. The intended user base is the students attending Texas State University in San Marcos, Texas. The website will initially only contain data on the products of local big-box stores such as Walmart, Target, Best Buy, and Office Depot, but it is a goal to add small businesses later. Product data will include things such as prices, sales, specifications, reviews, and modes by which to purchase the product.

Rising prices are the main reason for the creation of the website. Since 2021 the entire world has seen an increase in inflation, the effects are especially apparent when shopping. Between paying for food, rent, utilities, tuition, and expensive equipment and books for school, many college students are feeling the strain on their wallets. The goal of the website is to help users make informed decisions about the purchases they make, by providing as much information as possible. So not only can users find the best deals and save their money, but they can be assured of the quality of the product they are buying.

The second reason for the website's creation is to help small local businesses compete against large businesses. Many people choose to go to big-box stores instead of small locally owned businesses because of their presence and prices. As a result, small businesses are forced out of business, and consumers may suffer. A lack of competition can lead to a decrease in innovation, product variety, competitive prices, and jobs, and an increase in income inequality. By eventually working in accounts for business owners, local businesses can have their name and product as present in the public eye as those of big-box stores. And that visibility may increase income whether due to better prices, better products, convenience, or even a consumer's desire to help locally owned businesses survive.

Definition:

The website, built with Angular, will have two ways to search for a product, by category or through a search bar. All results can be filtered or sorted to make finding the right product easier for the user. When displaying products that may match the product the user is interested in, an overview of product details will be available for each product. Upon choosing a particular product a new page will open giving a more detailed description of the product; as well as stores it may be purchased at, their corresponding prices and sales, and potentially any relevant coupons that may be applied to the purchase. Other product details include name, UPC, a product description, product specific specifications, reviews, images, and manufacturer details.

INITIALLY

~~Currently~~ the website will only be offering products sold by select big-box stores in the San Marcos, Texas area. The product data of these stores is more easily attainable than small local businesses. Parsehub, a third-party web scraping tool, will be used to gather most product data from these store's websites. All information gathered will be stored in a MySQL database which will interact with a Node.js server to disperse information to the user.

Scope:

The scope of this project encompasses the development and deployment of a targeted product search and information website. The target of this product search application is college students in San Marcos, TX. The primary objective is to provide these students with an efficient and user-friendly platform for finding products they will need for participation in college studies, including detailed product descriptions, pricing, and relevant discounts. The website will feature two main search methods: category-based navigation and a versatile search bar, ensuring that users can access the desired information seamlessly. The products we have defined as "core" are as follows: Electronics, Appliances, Furniture. Furthermore, users will have the ability to filter and sort search results to refine their product selection. The project will involve the creation of a user interface, backend functionality for data retrieval and storage, integration with external data sources, and the implementation of a secure and responsive website.

Proposed Scope Expansion: ← LOCAL STORES?

While the initial project scope focuses on delivering core functionalities for product search and information, we recognize the potential for future enhancements. Specifically, features such as real-time availability tracking and geographical expansion to serve regions beyond the San Marcos area are currently considered outside the primary project scope. We would also consider implementing coupon aggregation as a function, including possible API integration with a 3rd party coupon aggregation site. However, as the project progresses and user feedback is gathered, we remain open to the possibility of integrating these features in subsequent phases. This flexibility allows us to prioritize delivering a working and user-friendly platform initially and then evaluate the feasibility and demand for additional functionalities, ensuring that the project can evolve to meet evolving user needs and market demands. We also recognize that some functionality currently inside the project scope might not be feasible depending on data availability.

Coupons: / SALES

Later, coupons will be added to the app because it will help students by giving them the best deal to use so they can buy technology at the best prices. Depending on the application, they will be given a coupon discount depending on the device they want to get and buy, and here are two applications that we are using as inspiration.

Shopkick

When you purchase often at your favorite retailers, you can use Shopkick to earn rewards and receive free gift cards, and you can then scan receipts to receive prizes. Simply save your points and use them at the store of your choice to find special offers, receive shopping incentives (kicks), and convert your points into gift cards. Launch the app while you are in the store at Target, Walmart, Amazon, or Best Buy to earn points without making a purchase. To get even more points, you can also use a connected card to make a purchase. Alternatively, you can watch movies in the app to find exclusive deals and promo codes, as well as receive discounts and prizes online.

Coupons.com

Features of Coupons.com: You can access hundreds of paperless coupons with the Coupons.com app while you shop (activate your location for in-store deals) or at home before you leave for the store. A wide range of grocery stores as well as big-box retailers like Target and Walmart all have coupons available. In order to have coupons available when you check out, you can also add them to shop loyalty cards. Additionally, the app includes a cash back incentive where users can activate deals at specific retailers and then present proof of purchase to receive cash back via PayPal.

Boundaries:

While the project aims to provide a robust product information platform, certain boundaries and limitations must be acknowledged. This project will not include the development of a dedicated mobile application but will ensure that the website is responsive and accessible on various devices. The integration of third-party data sources and real-time pricing information may be subject to data availability and API constraints. Additionally, the scope does not encompass the development of an e-commerce transaction system; users will be redirected to external retailers' websites for making purchases.

Requirements:

Front End

User Requirements:

1. Target Audience is College Students in San Marcos, TX
2. Allow User to browse through product categories and subcategories
3. Display list of products that match users search criteria or selected category
4. Provide detailed view of selected product (specifications, features, description, user reviews)
5. Allow users to add product to a comparison list for evaluation
6. Present a user-friendly interface for comparing products
7. Enable users to sort and filter products based on brand, price, rating, or availability
8. Display user ratings and reviews for product
9. Provide a list of related products based on user browsing

10. Implement accessibility features such as screen readers and keyboard navigation
11. Implement a feedback mechanism for users to report issues or to seek assistance

User functionality requirements:

1. find products to purchase based on price, reviews, and specifications
2. Use a search bar to find results
3. Filter search results
4. View a many to many comparison
5. View a one-to-one comparison
6. Find coupons for products

Non-Functional Requirements:

1. should be viewable on laptops, desktops, and mobile (desktop view)
2. interface should be easy to navigate for users without having to spend time looking around the page

Technical Stack:

1. Angular
2. Bootstrap (Framework)
3. Angular Material (Framework)
4. Mobile Angular UI (Framework)

Dependencies:

Will rely upon MySQL database for data to front end which gathers data from parsehub API call.

Core Framework:
- Angular: Since the project is built with Angular, it is the primary dependency.

User Interface:
- Angular Material or Bootstrap: For UI components like buttons, cards, modals, etc., we may use Angular Material or Bootstrap, which are commonly used with Angular projects.
- FontAwesome or Material Icons: For icons in the UI.

State Management:
- NgRx: For complex state management needs, we might consider using NgRx (Angular-specific)

Routing:
- Angular Router: For handling in-app navigation, Angular's own Router package could be used.

HTTP Client:
- Angular HttpClient: To handle HTTP requests to the backend server or any other APIs.

Real-Time Updates:
- Socket.io Client: If we include real time features, we will need Socket.io client

Boundaries:

Data Presentation:

- Real-time Updates: Prices and product availability can change rapidly. Front-end must be designed to handle real-time updates efficiently, without significantly affecting user experience.
- Standardization: Since you are pulling data from different retailers, the information might not be standardized. Front-end will need to present this data coherently and consistently.

Data Integrity:

- Accuracy: If the back-end data is not accurate or up to date, this will affect the front-end display. Users must be able to trust the prices we show.
- Data Consistency: Making sure that the displayed data is always up to date with the database can be a challenge.
- Fallbacks and Errors: The front-end should gracefully handle cases where data is not available, whether that is due to an issue with data scraping, an API being down, or some other problem.

Technical:

- Browser Compatibility: Application might start with specific browser like google chrome. Later we could make it to where all browsers are usable.
- Rate Limiting: If your application makes any client-side API calls, you may have to deal with rate limits on those APIs.

Design:

- UI/UX Consistency: Maintaining a consistent look and feel across various pages and elements.

Third-Party Integrations:

- Parsehub Limitations: If Parsehub cannot scrape certain types of data, the front-end will need to account for this by either not offering that option or finding a workaround.

Testing Requirements:

1. Verify that webpage works consistently across popular browsers
2. Test the responsiveness on different devices and screen sizes
3. Check for rendering issues
4. Verify that all UI components and features work as expected
5. Evaluate the user friendliness of the UI
6. Address any usability issues such as confusing navigation
7. Verify error messages are clear, informative, and user friendly
8. Review and update test cases and test data to ensure thorough coverage of scenarios

UNIT TESTING & PROCESS
DATA VALIDATION

Database

Functional Requirements:

1. The database will store the information obtained from the pulled/scrapped data.
2. The database will contain tables such as category, sub-category, product, store, store location, coupon, and junction tables for coupon and product, and stores.
3. The database will interact with the server side to provide data to display on the web app.

Technology Stack:

1. MySQL: Database management system to build the database.

2. Texas State Provided Server: Server provided by Texas State University to host the database.
3. Interact with the database using SQL queries and using server-side language and libraries from Node.js.

Database Structure:

- Category:
  - Used to state the different categories, the category ID is a foreign key in the product table.
  - Used for filtering and labeling.
  - Ex: Electronics, Appliances, Etc.
- Sub_Category:
  - Used to narrow the categories into smaller subcategories, it is also used as a foreign key in the product table.
  - Used for filtering and labeling.
  - Ex: Electronics: {cellphones, tablets, laptops}
- Product:
  - Contains the details of products that can be used to display on front end, as well as filters.
    - UPC, Name, Price, image URLs, store, categories.
- Store:
  - Contains store details, such as address, name, and ID.
  - Used as a foreign key in product to tie a product to a store.
- Store_Location:
  - Contains the address for a given store
  - Gets the foreign store id from Store table
  - Online stores will be null
- Coupon:
  - Contains sales details such as ID, description, dates, and URL
  - Coupons can be tied to stores or products, and products and stores can have multiple coupons.
- Coupon_Store:
  - Junction table to relate a coupon to a store.
  - Contains store_id and coupon_id, from Store and Coupon tables as foreign keys.
- Product_Coupon:
  - Junction table to relate a coupon to a product.
  - Contains product_id and coupon_id from Product and Coupon tables as foreign keys.

Data Sources

Functional Requirements:

1. The program will pull product information from various store websites.
2. The program will match a product from one store website to another.
3. The program will collate the product data pulled from each store into a final JSON.

4. The program will send the final JSON for a product to the database.

Non-Functional Requirements:

1. The program should be almost completely automated, so that little programmer intervention is needed.
2. The program should only pull data that was previously identified by a programmer.
3. The program should be able to accurately match identical products across stores, despite differences in available data.
4. The program should use Chrome as its browser when searching other stores for identical products.

Technical Stack:

1. Python: Programming language. ← NODE.JS? } NEEDS FOR PARSE HUB?
2. Selenium: Python module for automating browser use.
3. Chrome: Browser used to find URLs to be passed to ParseHub.
4. ParseHub: Web Scraping program with a GUI interface.
5. ParseHub API Wrapper: API allowing for programmatic use of ParseHub.

Server-Side

1. We will be using Node.js as the language and framework due to the frontend being coded with Angular and both are JavaScript based.
2. We will need to download NPM (Node package manager), which is free, incredibly easy and will give us access to many useful libraries we will be using.
    2.1. Express is one package that is popular due to its ease of access and many tools for server-side web development.
    2.2. There is also a package specifically made to interface and interact with a MySQL database, making that side of the server work much simpler.

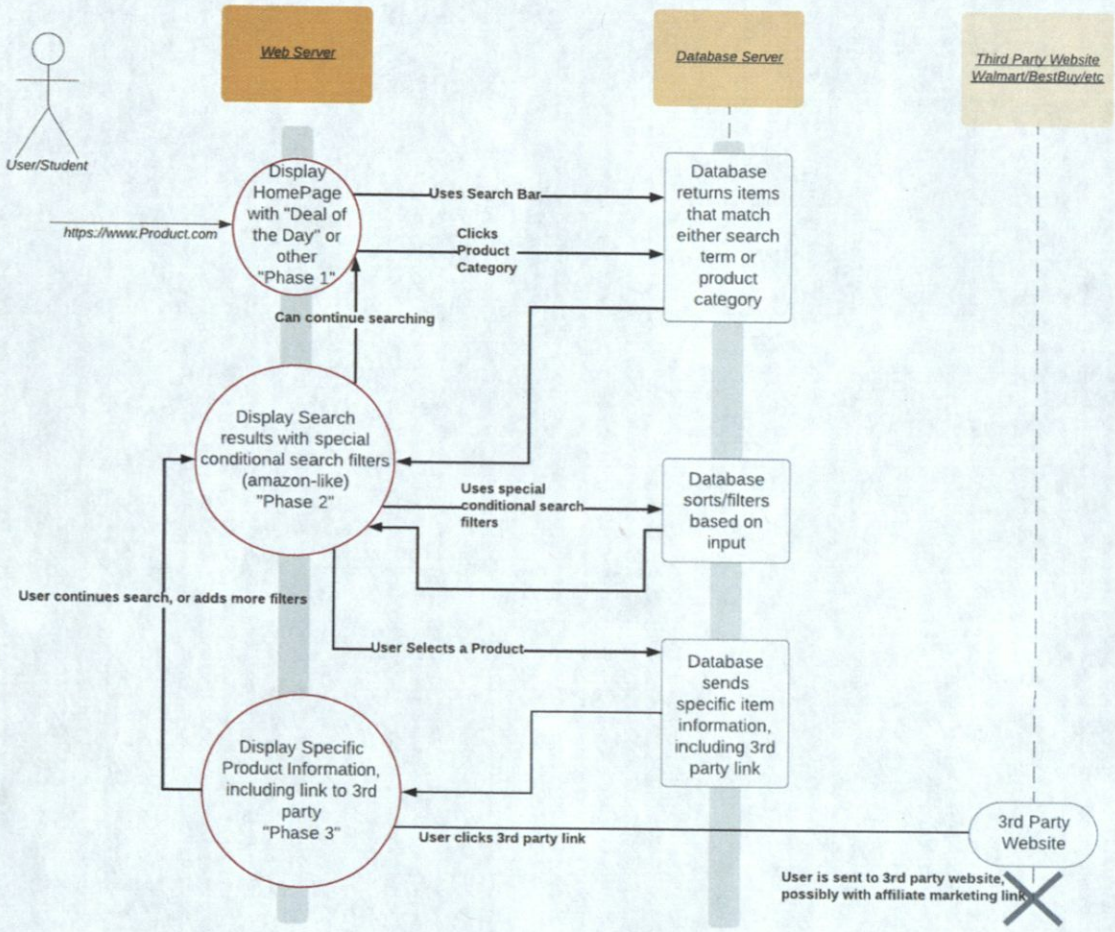
Schedule:

- 6 weeks before status report, utilizing 1-week sprints; Due: 10/12
    - Sprint 1 tasks
        - Server is setup and running basic webpage
        - Server has all dependencies installed
        - DB is setup
    - Sprint 2 tasks

        - Angular UI setup for webpage
        - Dummy data is inserted in DB
        - Webpage can pull from DB and display info
        - Scraper or API pulls from at least one website
            - Able to run script and save to DB or a file on the server
    - Sprint 3 tasks
        - UI fixes
        - API integration
        - Work on scraper/Api scripts to save data in the correct format
    - Sprint 4 tasks
        - Daemons set for schedule time to run API and Scraper scripts
        - Work on back scripts to save DB info
        - Work on scripts to track changes to DB for every scraper/Api call
    - Sprint 5 tasks
        - OpenAI API integration for filter and sorting
        - Bug fixes in Parsehub scripts
    - Sprint 6 tasks
        - Stress Test webpage
        - Daemon Script Fixes
        - Test backups
- 4 weeks for prototype, utilizing 2-week sprints; Due: 11/9
    - Scope extension implementation
- 3 weeks for final; Due 11/30
    - Deployment
    - Bux fixes
    - Stress Tests

High Level Designs:

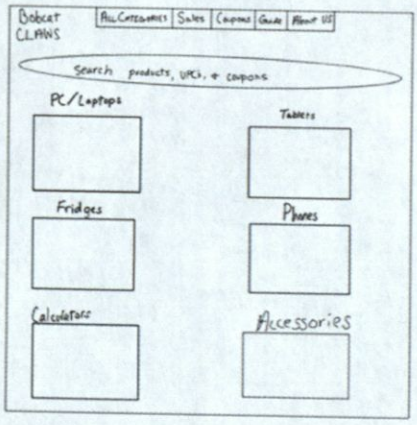
Sequence Diagram



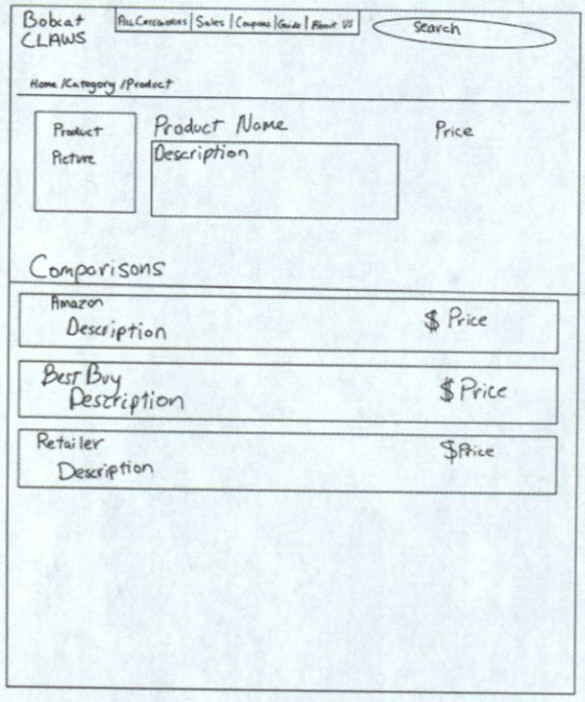
DATA SOURCE FLOW

Front End Wireframes

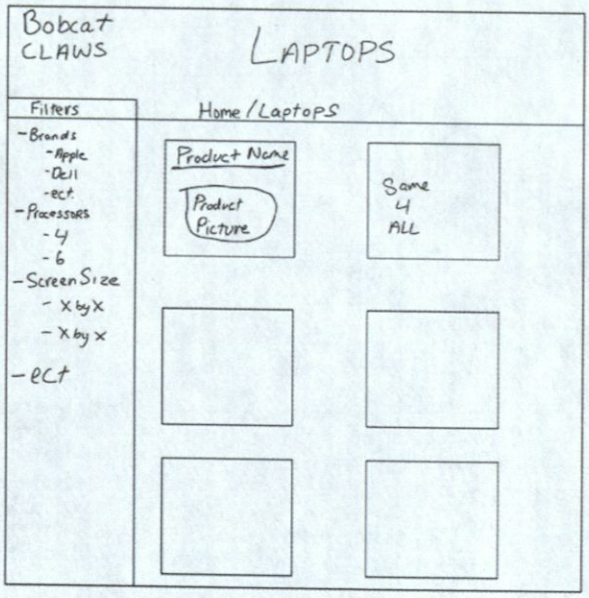
The home page shows all of our categories. With a search bar with some features at the top



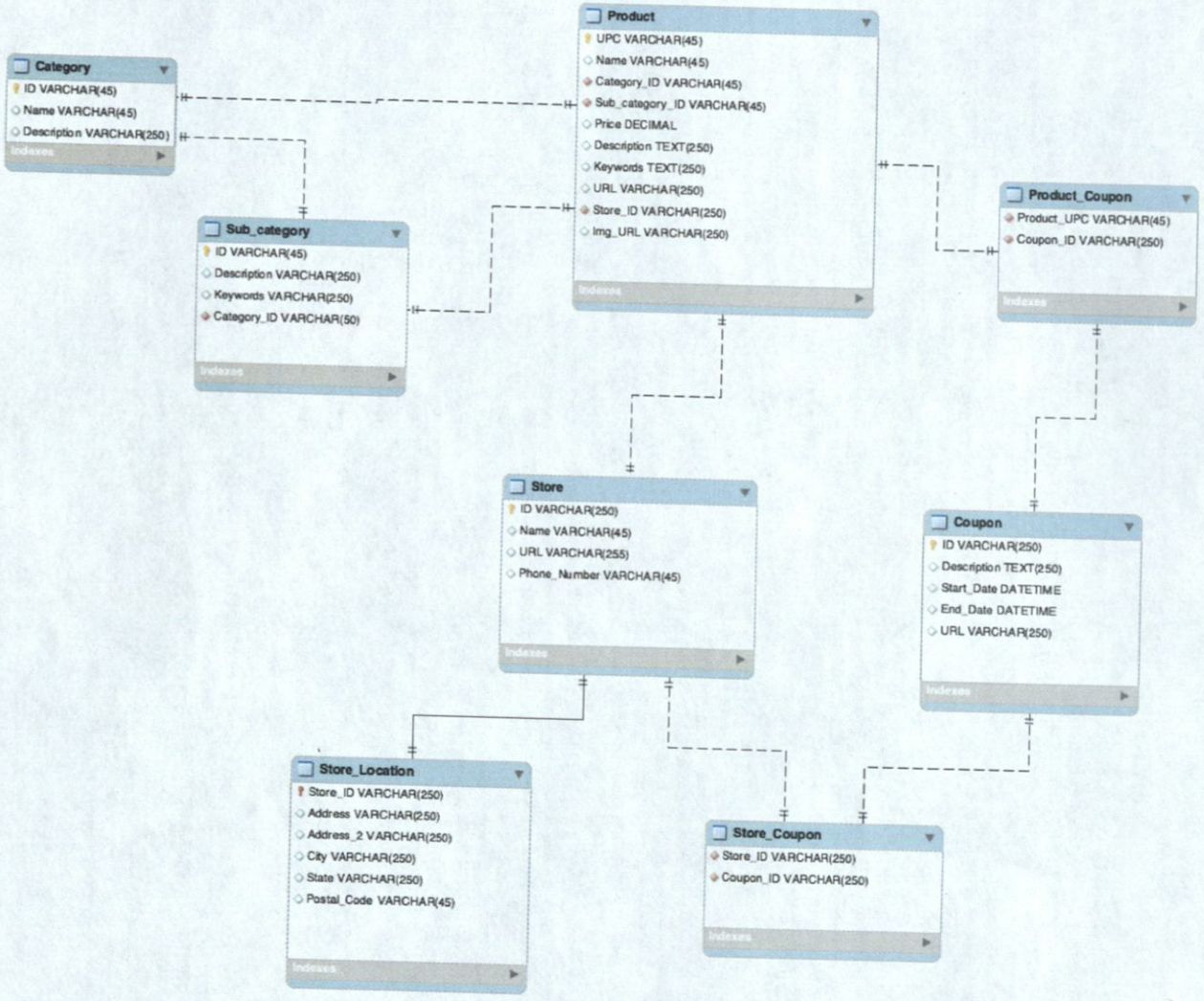
The product page is after the user has found a specific model. Me they find the model they like they can view all of the price options from the different retailers.



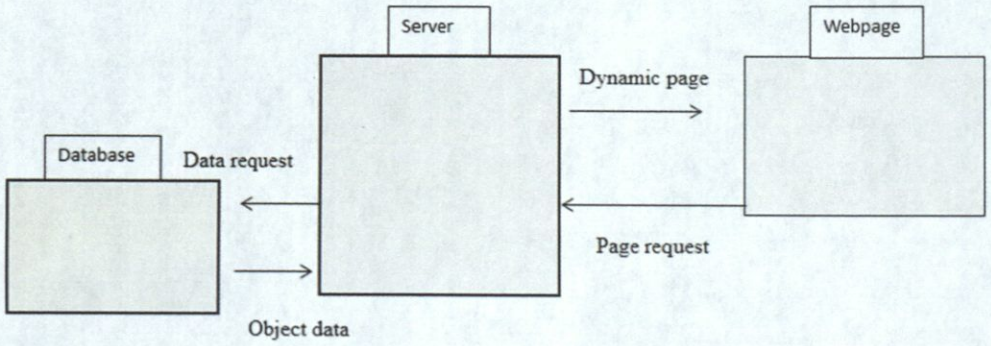
The category page should allow you to go back to home. When you click the category from the home page you should be able to see a variety of items and be able to filter them to find and meet needs. In the given example the category is laptops.



Database Design



Server Design



Web Scraping Flow Chart