5.1: System Components Diagram

$$\begin{tikzcd}
\text{[User Browser]} \arrow[r, leftrightarrow] & \text{[Next.js Frontend]} \arrow[r, leftrightarrow] \arrow[d, leftrightarrow] & \text{[Node.js API Server]} \arrow[r, leftrightarrow] & \text{[PostgreSQL Database (Supabase)]} \\
 & \text{[Python Web Scraper]} \arrow[r, leftrightarrow] \arrow[d, leftrightarrow] & \text{[External Brand Websites]} & \\
 & \text{[Google Sheets Integration]} & &
\end{tikzcd}$$

5.2: CSCI Component Breakdown

5.2.1 Frontend User Interface CSC -- React-based web interface for user interaction

- 5.2.1.1 Homepage Display CSU -- Main landing page with product grid
- 5.2.1.2 Product Browser CSU -- Individual product viewing and details
- 5.2.1.3 Outfit Creator CSU -- Interface for creating and managing outfits
- 5.2.1.4 Closet Manager CSU -- Interface for organizing outfits into closets
- 5.2.1.5 User Authentication CSU -- Login and user account management

5.2.2 API Server CSC -- Node.js backend for data processing and API endpoints

- 5.2.2.1 Product Management CSU -- CRUD operations for products
- 5.2.2.2 User Management CSU -- User authentication and profile management
- 5.2.2.3 Outfit Management CSU -- Outfit creation and modification
- 5.2.2.4 Closet Management CSU -- Closet organization and sharing

5.2.3 Database Management CSC -- PostgreSQL database for data persistence

- 5.2.3.1 Product Storage CSU -- Product information and metadata storage
- 5.2.3.2 User Data CSU -- User profiles and preferences storage
- 5.2.3.3 Relationship Management CSU -- User-product-outfit-closet relationships

5.2.4 Web Scraping CSC -- Python-based product discovery system

- 5.2.4.1 Universal Crawler CSU -- Generic website scraping functionality
- 5.2.4.2 Data Processing CSU -- Product data extraction and normalization
- 5.2.4.3 Integration CSU -- Google Sheets and database integration

5.3: Functional Requirements by CSC

5.3.1 Frontend User Interface

5.3.1.1 The frontend shall display a responsive homepage with a grid layout of fashion products.
5.3.1.2 The frontend shall provide a search bar for users to find specific products by title, brand, or category.
5.3.1.3 The frontend shall display product cards showing product image, title, price, and brand information.
5.3.1.4 The frontend shall provide a "Save to Outfit" button for each product that allows users to add items to their collections.
5.3.1.5 The frontend shall display a user dashboard showing saved outfits and closets.

5.3.1.6 The frontend shall provide an outfit creation interface where users can drag and drop products to create outfits.
5.3.1.7 The frontend shall allow users to name and categorize their outfits (e.g., "Casual Friday", "Date Night").
5.3.1.8 The frontend shall provide a closet management interface where users can organize outfits into themed collections.
5.3.1.9 The frontend shall support user authentication with login and registration forms.
5.3.1.10 The frontend shall display error messages for failed operations and provide user feedback for successful actions.

5.3.2 API Server

5.3.2.1 The API server shall provide REST endpoints for product CRUD operations (Create, Read, Update, Delete).
5.3.2.2 The API server shall authenticate users using secure token-based authentication.
5.3.2.3 The API server shall validate all incoming data before processing requests.
5.3.2.4 The API server shall return appropriate HTTP status codes for all API requests.
5.3.2.5 The API server shall support pagination for product listings to handle large datasets.
5.3.2.6 The API server shall provide endpoints for outfit creation, modification, and deletion.
5.3.2.7 The API server shall provide endpoints for closet management operations.
5.3.2.8 The API server shall implement rate limiting to prevent API abuse.

5.3.3 Database Management

5.3.3.1 The database shall store product information including title, description, price, brand, category, and image URLs.
5.3.3.2 The database shall maintain user account information including authentication credentials and preferences.
5.3.3.3 The database shall store outfit configurations linking users to their saved product combinations.
5.3.3.4 The database shall maintain closet organization data linking users to their outfit collections.
5.3.3.5 The database shall implement proper indexing for efficient query performance.
5.3.3.6 The database shall enforce referential integrity between related data tables.

5.3.4 Web Scraping System

5.3.4.1 The scraper shall be able to extract product information from various e-commerce websites using generic CSS selectors.
5.3.4.2 The scraper shall identify and extract product titles, prices, images, and URLs from target websites.
5.3.4.3 The scraper shall handle different website structures and adapt to various HTML layouts.

5.3.4.4 The scraper shall avoid duplicate product entries by implementing URL-based deduplication.
5.3.4.5 The scraper shall integrate with Google Sheets to store scraped product data.
5.3.4.6 The scraper shall post extracted products to the database via API endpoints.
5.3.4.7 The scraper shall implement respectful crawling practices with appropriate delays between requests.

5.4 Performance Requirements by CSC

5.4.1 Response Time Requirements
5.4.1.1 The homepage shall load within 3 seconds of user request.
5.4.1.2 Product search results shall be returned within 2 seconds of query submission.
5.4.1.3 API endpoints shall respond within 1 second for standard CRUD operations.
5.4.1.4 Database queries shall execute within 500 milliseconds for typical operations.

5.4.2 Scalability Requirements
5.4.2.1 The system shall support up to 100 concurrent users without performance degradation.
5.4.2.2 The database shall handle up to 10,000 products without significant performance impact.
5.4.2.3 The web scraper shall process up to 200 products per crawling session.

5.4.3 Data Quality Requirements
5.4.3.1 Product images shall be displayed with a minimum resolution of 300x300 pixels.
5.4.3.2 The scraper shall achieve at least 80% accuracy in product data extraction.
5.4.3.3 Duplicate product detection shall maintain less than 5% false positive rate.
5.5 Project Environment Requirements
5.5.1 Development Environment Requirements

Software Requirements:
Node.js 18.0.0 or higher
Python 3.8 or higher
PostgreSQL 13 or higher
Git for version control
VS Code or similar IDE
Google Cloud Platform account for Sheets API
Hardware Requirements:
8GB RAM minimum
50GB available disk space
Internet connection for API calls and web scraping

5.5.2 Execution Environment Requirements

Production Deployment:

Vercel for Next.js frontend deployment
Render or Fly.io for Node.js API server
Supabase for PostgreSQL database hosting
Google Cloud for Sheets API integration

Security Requirements:

HTTPS encryption for all data transmission
Environment variable management for API keys
Database connection security with SSL