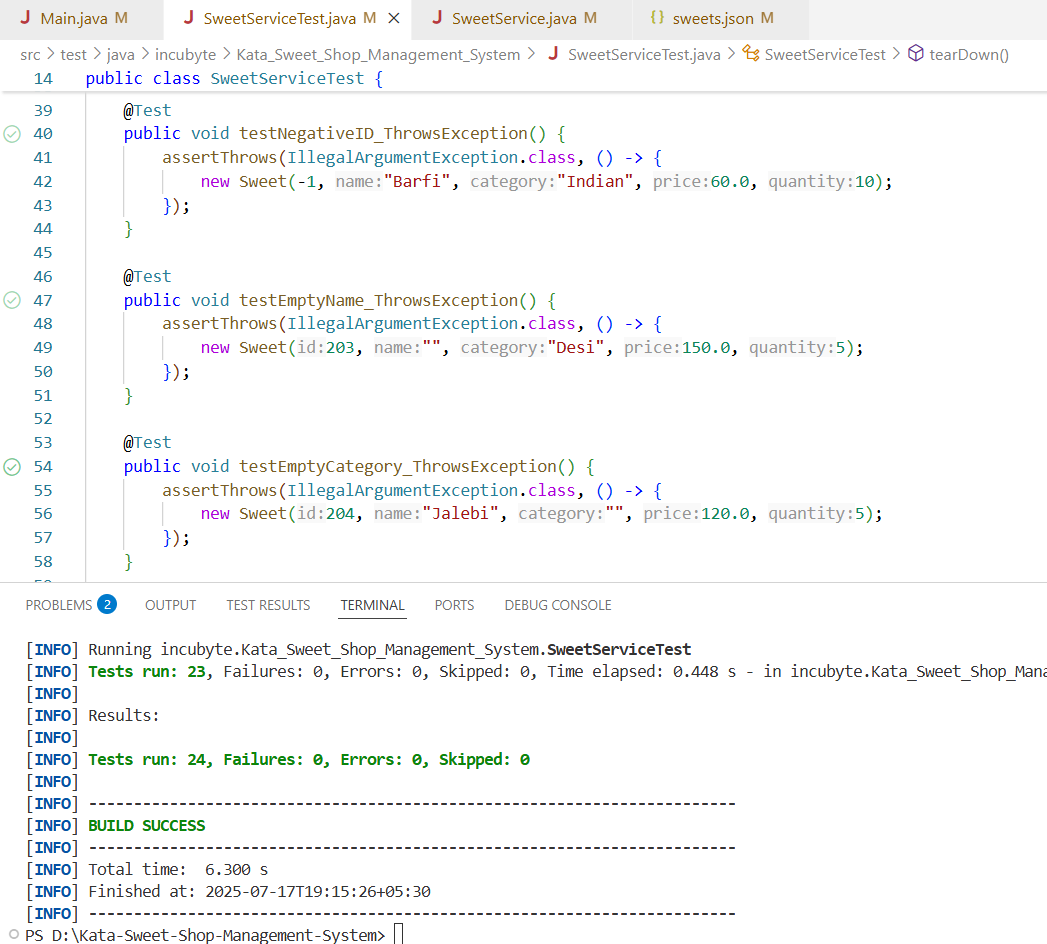
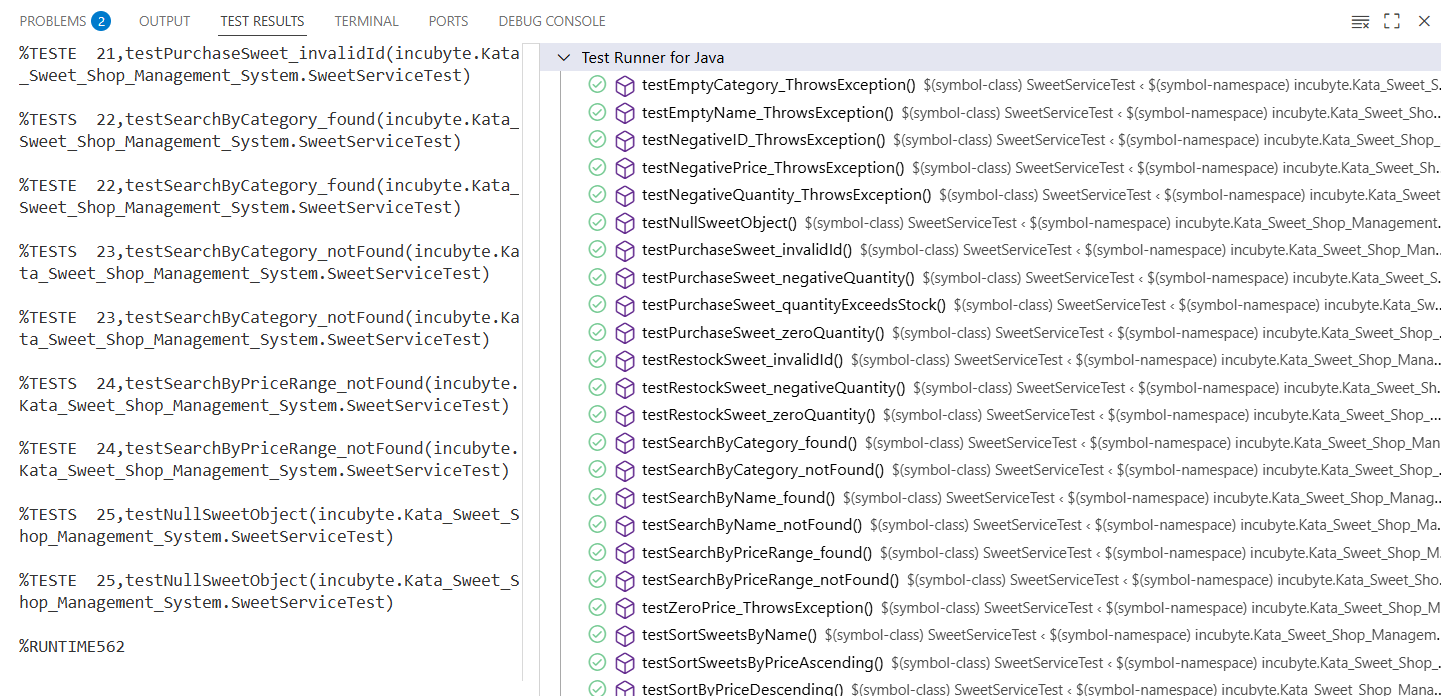
*The screenshot* displays the successful execution of *unit tests* for the Sweet Shop Management System. These tests were written using *JUnit* as part of a *Test-Driven Development (TDD)* approach to validate key functionalities such as input validation, purchasing sweets, restocking inventory, searching by category or price range, and sorting operations. All test cases passed without any errors, demonstrating that the system reliably handles both *valid inputs and edge cases*.

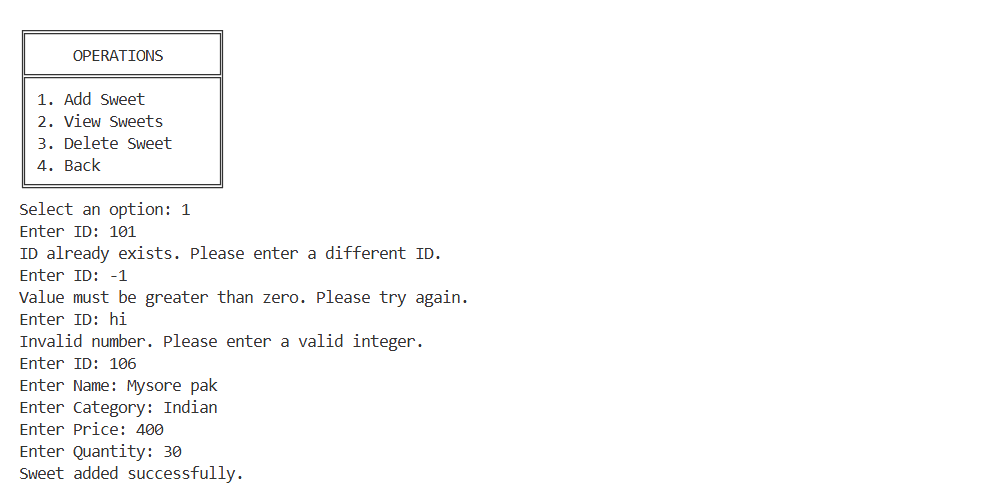




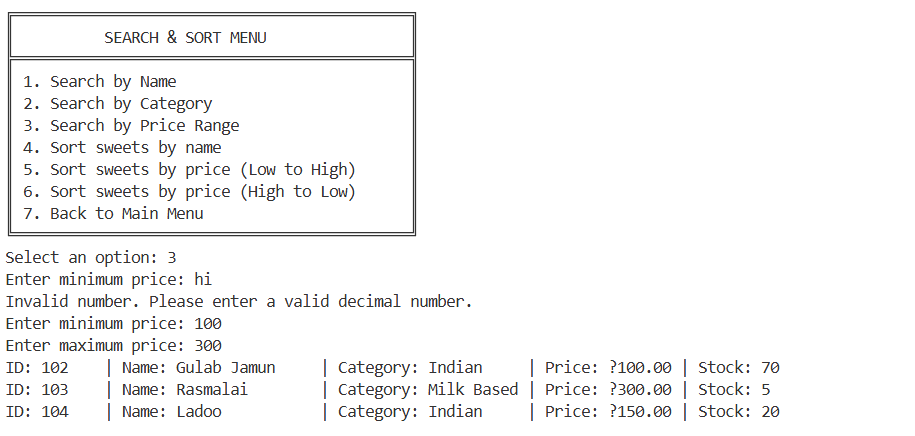
This screenshot shows the sweets.json file, which is used to store sweet data in a structured JSON format. Each sweet entry includes fields like id, name, category, price, and quantity. This file acts as a simple persistent data store for the *Sweet Shop Management System*, enabling operations like add, update, delete, and search to function correctly during runtime.



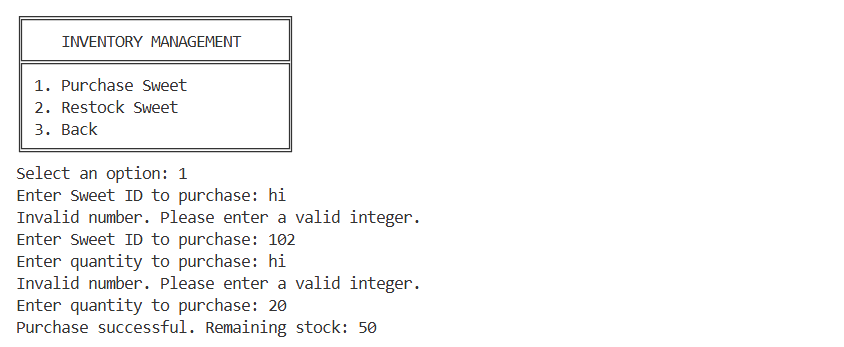
The Add Sweet functionality allows users to input a new sweet's details such as ID, name, category, price, and quantity. The system validates all entries by checking for duplicate IDs, ensuring positive values, and confirming proper data types. If the user enters an existing ID, a negative number, or non-integer text, the application prompts them with relevant error messages until valid input is received. Once all values are correct, the sweet is added successfully to the inventory, confirming the addition with a success message.



In the Search & Sort Menu, users can search for sweets within a specific price range. The system first asks for minimum and maximum price values and ensures they are valid decimal numbers. If incorrect data like text input is given, the user is prompted with an appropriate error. Once valid input is entered, the application filters the sweets whose prices fall within the provided range and displays them in a structured format showing ID, name, category, price, and available stock.



The Purchase Sweet option under Inventory Management allows users to buy sweets by providing a sweet ID and purchase quantity. The application validates these inputs to ensure they are positive integers and not invalid characters or negative numbers. If validation fails, the system provides clear messages and re-prompts the user. Once valid details are entered, the system deducts the purchased quantity from the existing stock and confirms the successful transaction along with the remaining stock count.



The Restock Sweet feature enables users to add more stock to an existing sweet. The user is required to input the sweet ID and the quantity to restock. Similar to other modules, validations are in place to reject invalid data types or negative values. If incorrect input is provided, the system displays helpful error messages. Upon receiving valid data, the system adds the specified quantity to the sweet’s stock and displays the new total quantity, confirming that restocking was successful.

