**ACS 560 Software Engineering**

Homework 4

Name: Vijayagiridharan Subramanian

**Task 1: Research Formats for Writing Requirements**

When developing software systems, choosing the appropriate format for documenting requirements is crucial for ensuring clarity and effective communication among stakeholders. In this task, I will compare two common formats: the IEEE Standard 830-1998 format and the User Story format.

### **Format 1: IEEE Standard 830-1998**

The IEEE Standard 830-1998 provides a structured framework for developing Software Requirements Specifications (SRS). This format includes the following components:

1. **Introduction**:
   * **Purpose**: Defines the document’s intent and objectives, setting the stage for what is to follow.
   * **Definitions, Acronyms, and Abbreviations**: Clarifies key terminology, ensuring that all stakeholders share a common understanding.
   * **References**: Lists related documents and resources, allowing readers to explore foundational material.
   * **Overview**: Summarizes the document’s contents, providing a roadmap for the reader.
2. **Overall Description**:
   * **Product Perspective**: Describes how the software fits within the existing ecosystem and its interactions with other systems.
   * **Product Functions**: Specifies the high-level features and functions that the software will provide.
   * **User Characteristics**: Identifies the target users, their skills, and any special requirements.
   * **Constraints**: Highlights limitations, including legal, regulatory, or technical constraints that impact design and functionality.
   * **Assumptions and Dependencies**: Lists external factors that could impact the success of the project.
3. **Specific Requirements**:
   * **External Interfaces**: Describes how the system will interact with other systems or components.
   * **Functional Requirements**: Details specific actions the system must perform to meet user needs.
   * **Performance Requirements**: Establishes metrics such as response times and resource utilization.
   * **Design Constraints**: Defines limitations related to architecture, technologies, or methodologies.
   * **Attributes**: Specifies non-functional qualities such as usability, reliability, and maintainability.
   * **Other Requirements**: Captures any additional critical requirements that do not fit into the above categories.

#### **Advantages:**

* **Thoroughness**: Provides a detailed and comprehensive framework, ideal for large-scale or regulated projects.
* **Standardization**: Establishes a common format that aids collaboration and understanding among diverse stakeholders.

#### **Disadvantages:**

* **Complexity**: Can be overly complex and time-consuming to produce, particularly for smaller projects.
* **Detail Level**: May include excessive detail that could overwhelm users in simpler scenarios.

### **Format 2: User Story Format**

The User Story format is a simpler, more flexible approach to capturing requirements, frequently used in agile methodologies. Each user story typically follows this structure:

* **As a** [user role], **I want** [goal] **so that** [reason].

For example:

* As a customer, I want to filter menu items by category so that I can quickly find what I’m looking for.

#### **Advantages:**

* **Simplicity**: Easy to understand and accessible to all stakeholders, including non-technical members.
* **User-Centric**: Focuses on user needs and goals, promoting a design approach that prioritizes user experience.
* **Collaboration**: Encourages team discussions and iterations, aligning development with user feedback.

#### **Disadvantages:**

* **Detail Limitations**: May lack sufficient detail for complex systems without additional documentation or refinement.
* **Ambiguity Risks**: Can lead to misunderstandings if user stories are not clearly defined or elaborated.

### **Comparison of Formats:**

The IEEE Standard 830-1998 format is tailored for environments where comprehensive documentation is essential, such as in government or safety-critical projects. It offers detailed specifications that help manage complexity and ensure compliance with regulations. Conversely, the User Story format is ideal for agile development settings, where adaptability and user focus are prioritized. It enables rapid iteration and emphasizes direct communication among team members, making it particularly suitable for projects that evolve based on user feedback.

**Chosen Format**: For this assignment, I will use the **User Story format**. This format’s emphasis on simplicity and user perspectives is crucial for developing features that effectively meet user needs within the REST API project.

**References**:

* IEEE. (1998). **IEEE Recommended Practice for Software Requirements Specifications (IEEE Std 830-1998)**.
* Cohn, M. (2004). **User Stories Applied: For Agile Software Development**. Addison-Wesley.

**Task 2: Functional and Non-Functional Requirements for REST API**

### **1. Functional Requirements for Existing Functionality:**

1. **Retrieve Menu Items**:
   * The system shall allow users to obtain a complete list of all available menu items. This functionality ensures that users can access the full inventory of offerings.
2. **Retrieve Menu Item by ID**:
   * The system shall enable users to access detailed information about a specific menu item using its unique identifier. This supports users in finding precise details quickly.
3. **Create Menu Item**:
   * The system shall support users in creating a new menu item, including details like name, description, price, and category. This allows for dynamic updates to the menu based on user input.

### **2. Functional Requirements for New Functionality:**

1. **Update Menu Item**:
   * The system shall allow users to modify the details of an existing menu item by referencing its unique ID. This enables accurate and timely updates.
2. **Delete Menu Item**:
   * The system shall allow users to remove a menu item by providing its unique identifier. This feature ensures that the menu remains current and relevant.
3. **Retrieve Menu Items by Category**:
   * The system shall allow users to filter menu items by a specified category, enhancing the user experience by making item discovery more efficient.

### **3. Non-Functional Requirements:**

1. **Performance**:
   * The system shall respond to API requests within two seconds under normal operating conditions, ensuring a smooth user experience.
2. **Scalability**:
   * The system shall support up to 10,000 concurrent users without performance degradation, accommodating growth and peak usage effectively.
3. **Security**:
   * The system shall encrypt all data transmitted via the API using HTTPS to protect user information, complying with industry standards for security.

**Task 3:**

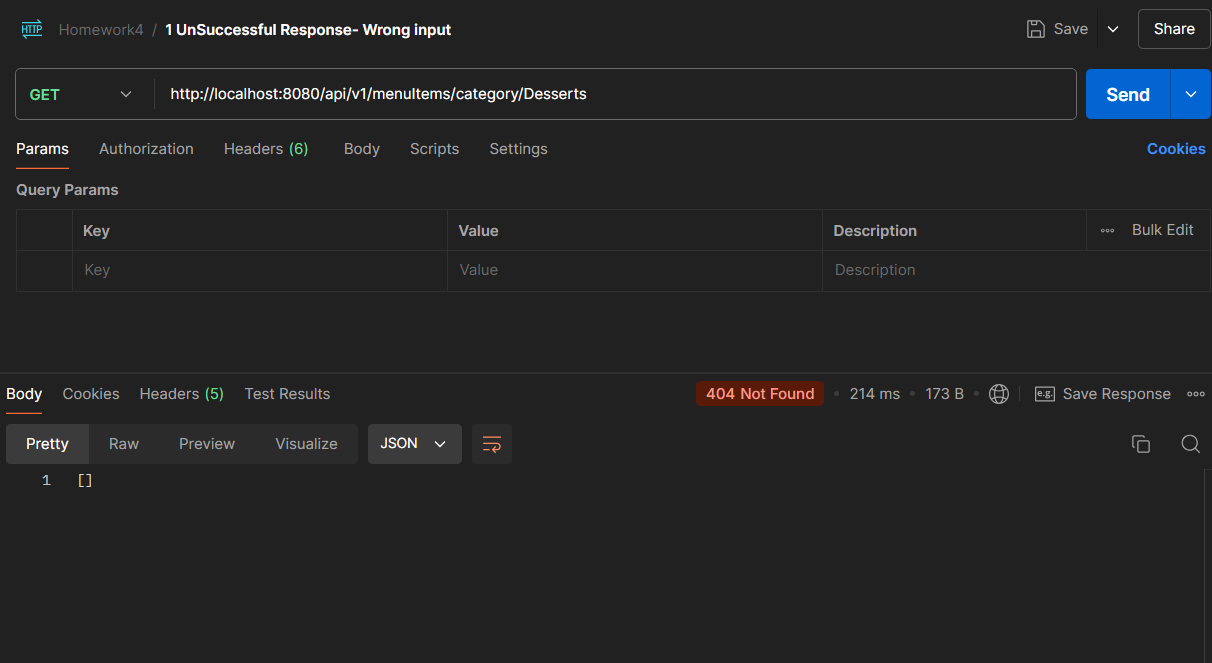
**Functional Requirement:**

**Retrieve Menu Items by Category:**

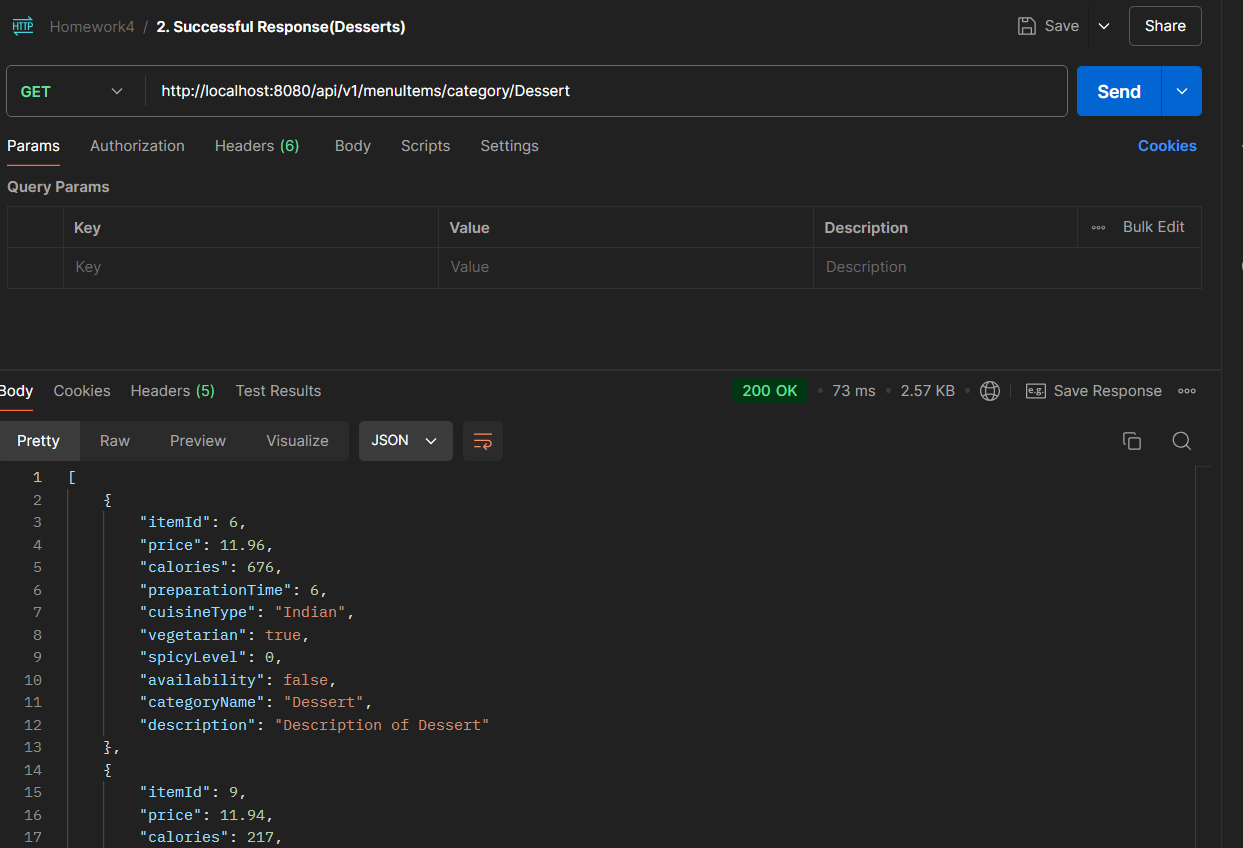
The system shall allow users to fetch a list of menu items filtered by a specific category.

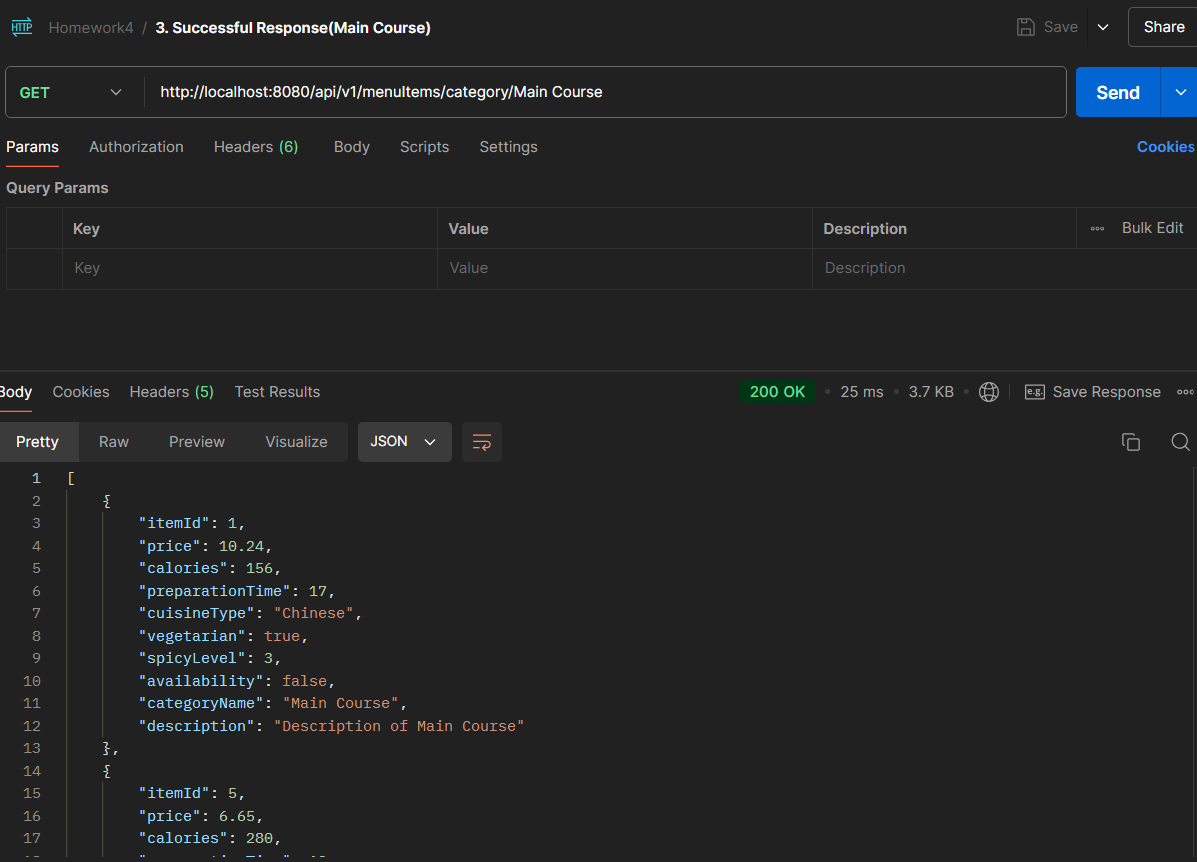
I provided postman responses in json for this functionality.

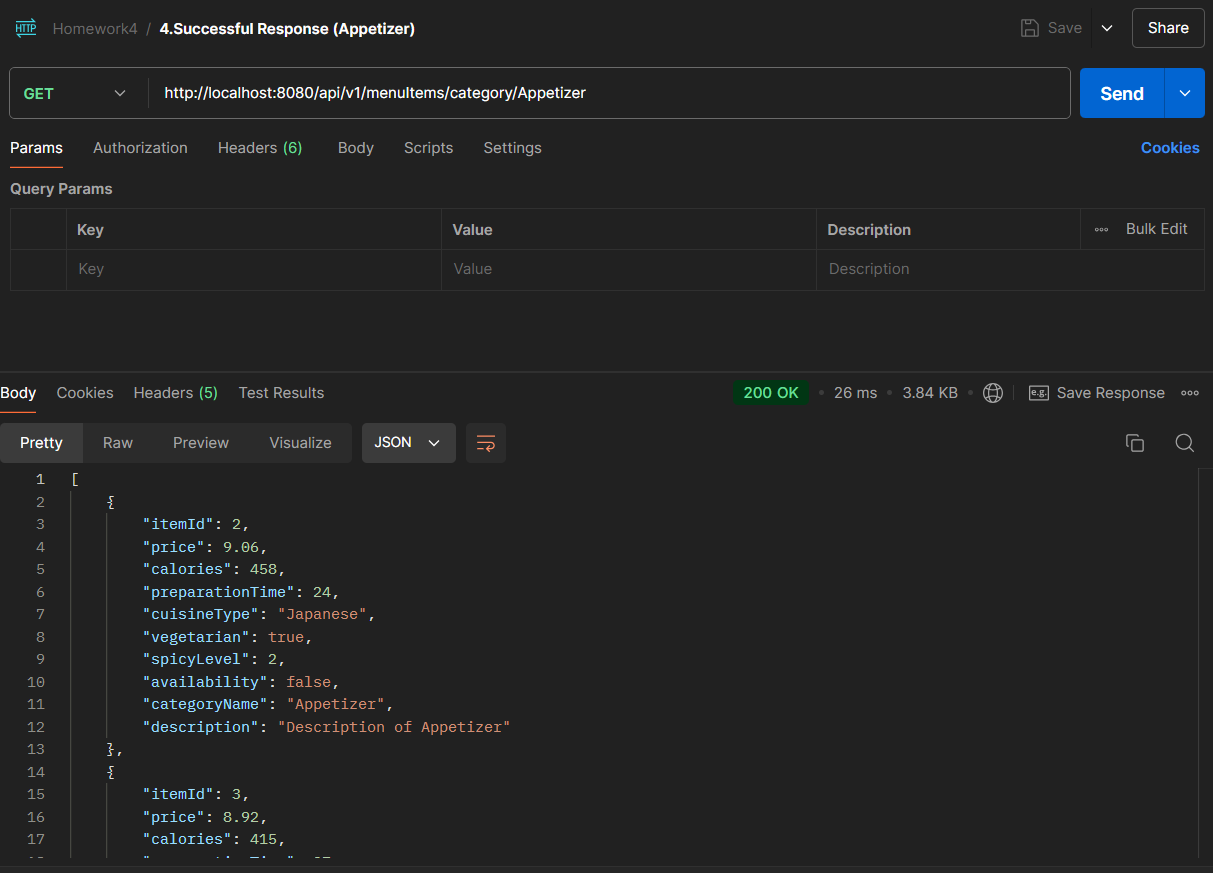
1. Unsuccessful Response because of the Wrong category input.



1. Successful Responses:



****

****