**Mail Order Pharmacy Application - Requirement Document**

**Important Instructions:**

1. You must adhere to the Design Considerations specific to each Technology Track.
2. You must not submit projects with compile-time or build-time errors.
3. Being a Full-Stack Developer Project, you must focus on ALL layers of the application development.
4. Unit Testing is Mandatory, and we expect a code coverage of 100%. Use Mocking Frameworks wherever applicable.
5. A READ ME has to be provided with steps to execute the submitted solution, the Launch URLs of the Microservices in cloud must be specified. (Importantly, the READ ME should contain the steps to execute DB scripts, the LAUNCH URL of the application.
6. Follow coding best practices while implementing the solution. Use appropriate design patterns wherever applicable.
7. You are supposed to use an In-memory database or sessions as specified, for the Microservices that will be deployed in the cloud. No Physical database is suggested.

**1. Introduction**

**1.1 Purpose of this Document -** The purpose of the software requirement document is to systematically capture requirements for the project and the system “Mail Order Pharmacy” that has to be developed. Both functional and non-functional requirements are captured in this document. It also serves as the input for the project scoping.

The scope of this document is limited to addressing the requirements from a user, quality, and non-functional perspective.

High Level Design considerations are also specified wherever applicable, however the detailed design considerations have to be strictly adhered to during implementation.

**1.2 Project Overview -** A leading Pharmaceutical Benefits Management (PBM) Organization wants to develop a Mail-Order Pharmacy module using Microservices.

PBM is a middle-man between an Insurance Company and its members. Because of its size, it can negotiate large customer contracts and negotiate with both pharmacies and pharmaceutical companies to get the best rates for the drugs.

Mail-order pharmacies operate through your health insurance plan and can be cheaper and more convenient than getting your medications from a local pharmacy. A “mail-order” prescription means that you don’t have to pick up your medication from a pharmacy; instead, it is delivered to your doorstep.

There will also be a Member Portal to be developed part of this scope that consumes these Microservices. The Portal allows its members to subscribe for drug refill, allows them to pay bills etc.

**2. Scope**

The scope of the project includes the development of the following modules:

| **Req. No.** | **Module Name** | **Description** |
| --- | --- | --- |
| REQ\_01 | **Drugs Module** | Drugs Module is a Middleware Microservice that performs following operations:   * Search Drugs by Name / ID * Get Dispatchable Stock of a Drug by Location |
| REQ\_02 | **Subscription Module** | Subscription Module is a Middleware Microservice that performs the following operations:   * Subscribe/Un-subscribe for Mail-Order Pharmacy   **Note**: Based on subscription, there will be a Mail-Order Delivery Module that will deliver the drugs to the members. Delivery is out of scope of the system. The Refill Details, Payment Status will be available in the Refill Module. |
| REQ\_03 | **Refill Module** | Refill Module is a Middleware Microservice that performs the following operations:   * View Last Refill for the Subscriptions * Get Prescriptions that are Due For Refills * Take Adhoc Refill Order for a subscription   **Note:** Online Payment is currently not integrated in this Microservice responsibility. Assume that Payments will be done outside the scope of this system and this Service will get the payment status along with refill details as a data feed. |
| REQ\_04 | **Member Portal** | An Web Portal that allows a member to Login and allows to do following operations:   * Login * Subscribe / Un-Subscribe for the Mail-Order * View the last refill status * View Prescriptions that are due for Refills * Take an Adhoc Refill Order * View Drugs |

**3. Functional Requirements**

**3.1 Drugs Module**

* **Description:** This core service reflects the Drugs supported by the Pharmacy, and the stock of drugs as of date, and the ability to dispatch to the current member location. Every request has to be authorized and only then a response will be sent back.
* **Functionalities:**

Post Authorization, the Drugs Microservice will perform the following functionalities:

* Search Drug based on Name / ID (the response will provide the complete drug details like manufacturer, medical composition, manufactured date, expiry date, units in a package, cost per package, quantities available across locations)
* View Dispatchable Stock of the Drug (Given the Drug ID or Name and Location, the service will provide the drug details and the quantity available to dispatch in the given location
* **Entities:**
  + Drug (ID, Name, Manufacturer, Expiry Date, Quantity).
  + Drug Location (Location ID, Drug ID, Quantity).
* **Endpoints:**
  + GET: /searchDrugsByID (Input: Drug\_ID | Output: Drug Details like ID, Name, Manufacturer, Manufactured Date, Expiry Date, Quantity available across loc. etc..)
  + GET: /searchDrugsByName (Input: Drug\_Name | Output: Drug Details like ID, Name, Manufacturer, Manufactured Date, Expiry Date, Quantity available across loc. etc..)
  + POST:Me (Input: Drug\_ID, Location) | Output: Drug ID, Name, Expiry Date, Available Stock in the given location)
* **Trigger:** Can be invoked from Refill / Subscription Service or from the MVC Member Portal App.
* **Steps and Actions:**

1. This is a basic GET service of the Drug Details in real time.
2. Authorization has to be performed
3. Basic Drug Input Validations have to be performed like ID / Name / Location etc.

* **Non-Functional Requirements:**
* Only Authorized Member can access these REST End Points
* At any point in time, these REST End Points can be invoked by multiple clients, hence all the requests have to be processed in parallel

**3.2 Subscription Module**

* **Description:** Subscription Microservice interacts with Drugs Microservice, but it gets invoked from Member Portal (MVC app).
* **Functionalities:**

Subscribe for Mail-Order Pharmacy:

* Get the Prescription Details and Insurance Policy Details from Member Portal
* Scan the Drug Details and check its availability by interacting with Drugs Microservice. If drugs are available then accept the subscription. If not, convey the message that the subscription cannot be accepted due to drug unavailability.

**Note:** The Drug stock may not be available during subscription that is still fine, however the drug must be present in the list of the Drugs Service. Assume that Insurance Provider/Policy Details will not be verified in the scope of this system.

Unsubscribe for Mail Order Pharmacy:

* View If there are any Re-fills with pending payment status. If none, then unsubscribe. Else, display that Un-subscribe is not successful due to pending payment:

1. Interact with Refill Service to understand the refill status, pending status

* **Entities:**
  + Member Prescription (Member ID, Prescription Details).
  + Member Subscription (Subscription ID, Status, Refill Frequency).
* **Endpoints:**
  + POST: /subscribe (Input: prescription details, Policy Details, Member\_ID | Output: Subscription Status, Subscription Status Description)
  + POST: /unsubscribe (Input: Member\_ID, Subscription ID) | Output: Subscription Status, Subscription Status Description)
* **Trigger:** Can be invoked from Member MVC Portal. However this will interact with both Drugs Service and Refill Service for Subscribe and Un-subscribe operations respectively.
* **Steps and Actions:**

1. Subscription Microservice will have 2 End Points.
2. If  /subscribe end point is invoked by Member Portal, the Microservice will check the drugs list by interacting with Drugs Microservice and then will accept the subscription. The response will be sent back to the Member Portal.
3. If  /unsubscribe end point is invoked, then the Unsubscribe Microservice will invoke the Refill Microservice for retrieving the latest refill status. If there are no payment dues for refill, then unsubscription will be performed. The appropriate status will be returned back.

**3.3 Refill Module**

* **Description:** Refill Microservice interacts with Drugs Microservice before taking an Adhoc Refill Order. Post authorization of request; interacts with Subscription Service to get the refill frequency subscribed to calculate the outstanding refill dues.
* **Functionalities:**
  + View the latest Refill Status of a subscription
  + Get Subscriptions that are due for Re-fill from a given date
  + For the given subscription ID, get the refill frequency chosen by interacting with Subscription Service
  + Take an Adhoc Refill Order based on availability of drugs based on a given location.
* **Entities:**
  + Refill Order (Subscription ID, Refill Date, Refill Order ID).
  + Refill Order Line Item (Subscription ID, Drug ID, Quantity).
* **Endpoints:**
  + GET: /viewRefillStatus (Input: Subscription\_ID | Output (Refill Order Details)
  + GET: /getRefillDuesAsOfDate (Input: Subscription\_ID, From Date | Output: Subscription IDs that are Due for Refill).
  + POST: /requestAdhocRefill (Input: Policy\_ID, Member\_ID, Subscription\_ID, Location | Output: Refill Details)
* **Trigger:** Can be invoked from Member Portal.
* **Steps and Actions:** Refill Microservice will have 3 End Points:
* View Refill Status end point will just respond back with the last refill status for the given Subscription ID
* Get Refill Dues will have to interact with the Subscription Service to understand the frequency of subscription and  based on that, the service has to calculate the following:
* If the frequency of subscription is weekly or monthly, it has to return the list of subscriptions that are due to refill from the last refill date
* Request Adhoc Refill has to reach out to Drugs service to see if the requested drugs are available as per member Location and if only available, the request has to be accepted. If drugs are not available, return a response that the “refill request failed” with appropriate reasons.

**3.4 Authorization Microservice**

* **Description:** Provides Authentication and Authorization for the portal.
* **Functionalities:**
  + Service to Service communication has to happen using JWT
  + Pass End User Context across Microservices.
  + Have the token expired after a specific amount of time, say 15 minutes.
  + Have this service configured in the cloud along with other services

**3.5 Member Portal**

* **Description:** Member Portal  must allow a member to Login.
* **Functionalities:**
  + User login.
  + Subscribe / Unsubscribe a Prescription for Refill Orders
  + View Drugs that are supported
  + View Refill Status
  + View Refills that are due for the member
  + Request an Adhoc Refill

**3.6 Swagger**

* **Functionalities:**
  + All the Microservices must be configured with Swagger for documentation
  + Register the swagger resources in the Swagger Microservice and enable them as REST end points
  + Configure this service along with other services in the cloud

**4. Non-Functional Requirements**

* **Performance:** The application should handle multiple requests simultaneously without performance degradation.
* **Security:** All sensitive data should be encrypted, and user authentication should be implemented.
* **Scalability:** The application should be designed to scale easily to accommodate an increasing number of users.
* **Usability:** The user interface should be intuitive and easy to navigate.
* **Reliability:** The application should have a high uptime, with minimal downtime for maintenance.

**5. Software Requirements**

* **Backend:**
  + Java 17.
  + Spring Boot.
  + Maven for build management.
  + H2 Database (in-memory for testing).
  + JUnit, Mockito (for testing)
  + Docker (Optional).
* **Frontend:**
  + NodeJS 18+
  + React 18+
  + TypeScript
  + React Router v6
  + Axios for API calls
  + Material - UI for UI components
  + React Hook Form for form management
  + React Query for server state management
  + Jest & React Testing Library for testing
* **Additional Tools:**
  + Postman for API testing.
  + Git for version control.

**6. System Architecture**

The application will follow a monolithic architecture where all modules are part of a single codebase. This will include layers for:

* **Presentation Layer:** The Member Portal (MVC-based).
* **Business Logic Layer:** The core logic for managing drugs, subscriptions, and refills.
* **Data Access Layer:** Interactions with the database.

**7. Development and Deployment**

* **Development Environment:** Local development using the specified hardware and software requirements.
* **Deployment Environment:** The application will be deployed on a single server in a production environment.
* **Database:** The application will use an SQL Server for persistent storage.

**8. Testing Requirements**

* **Unit Testing:** Unit tests should cover at least 80% of the codebase. Mocking frameworks should be used where applicable.
* **Integration Testing:** Ensure all modules interact correctly.
* **User Acceptance Testing (UAT):** The application should be tested with real-world scenarios by a sample of end-users.

**9. Documentation**

* **Code Documentation:** Follow industry best practices for documenting code.
* **User Guide:** A comprehensive guide for end-users on how to use the Member Portal.
* **Developer Guide:** Detailed instructions for setting up the development environment, building, and deploying the application.

**10. Conclusion**

This document outlines the requirements for developing a Mail Order Pharmacy Application. The project will be built as a monolithic application, encompassing all necessary functionalities for managing drugs, subscriptions, and refills, with a focus on delivering a robust and user-friendly system.