***MyForce***

***Global***

**Software Design Specification**

[Complete the below table only if this document will be routed on paper or signed digitally. If routing through D2 delete the table]

|  |  |  |  |
| --- | --- | --- | --- |
| Prepared by: | Title | Signature | Date |
| Shridhar Kakkeri | Présentation Manager and RTE Architecture |  | 31 Aug 2023 |
| Reviewed by: | Title | Signature | Date |
|  |  |  |  |
| Approved by: | Title | Signature | Date |
|  |  |  |  |

**Table of Contents**

**[1.0](#_heading=h.gjdgxs)****INTRODUCTION 4**

[1.1](#_heading=h.30j0zll) Purpose 4

[1.2](#_heading=h.1fob9te) Scope 4

[1.3](#_heading=h.3znysh7) Assumptions 4

[1.4](#_heading=h.2et92p0) Limitations 4

[1.5](#_heading=h.tyjcwt) Exclusions 4

**[2.0](#_heading=h.3dy6vkm)****DEFINITIONS & ACRONYMS 4**

**[3.0](#_heading=h.4d34og8)****DATA AND PROCESS FLOWS 5**

**[4.0](#_heading=h.2s8eyo1)****DETAILED SOFTWARE DESIGN 5**

[4.1](#_heading=h.17dp8vu) Development Method 5

[4.2](#_heading=h.26in1rg) Data Design 5

[4.3](#_heading=h.2xcytpi) Architectural and Component Level Design 6

[4.4](#_heading=h.1ci93xb) Software Components 6

[4.5](#_heading=h.3as4poj) User Interface Design 6

[4.6](#_heading=h.1pxezwc) Screen Images 6

[4.7](#_heading=h.147n2zr) Interface Design Rules 6

[4.8](#_heading=h.3o7alnk) GUI Components available 7

[4.9](#_heading=h.23ckvvd) Constraints 7

**[5.0](#_heading=h.1hmsyys)****REFERENCES 7**

**[6.0](#_heading=h.2grqrue)****ADDENDUMS, SUPPLEMENTS, REVISIONS 7**

**[7.0](#_heading=h.vx1227)****REVISION HISTORY 8**

**[8.0](#_heading=h.3fwokq0)****APPENDIX(ES) 8**

**List of Tables**

[Table 1](#_heading=h.1t3h5sf) Definitions & Acronyms 4

[Table 2](#_heading=h.41mghml) References 7

1. 1.0 **INTRODUCTION**
2. **Purpose**

The primary purpose of this Software Design Specification (SDS) is to provide a detailed roadmap for the design and architecture of the MyForce application and to provide a clear blueprint of how the application is structured and designed. It serves as a reference guide for developers, ensuring that the application is constructed effectively and consistently.

By having this document, you can quickly reference and understand how different parts of the application interact, which technologies are employed, and how data is managed. This helps in troubleshooting, future enhancements, and ensuring consistency in the development process. The document also acts as a reference point for making informed decisions during the application's lifecycle.

In essence, this architecture document saves time, enhances collaboration, and contributes to the successful development and maintenance of the MyForce application.

1. **Scope**
2. This document applies to requirements forPresentation Managerto support the following business functions:
   1. Used to upload the zip file containing ppt, thumbnails of slides, pdf and mp4 of the presentation against the Presentation record created in the SDK.
   2. The uploaded file to be available in Media section for the MR to download.
3. **Assumptions**

This document applies to requirements forPresentation Managerto support the following business functions:

* **User Access :** It's assumed that users of the application will have the necessary access credentials to log in securely. This may involve integrating with an authentication system or utilizing Salesforce's user management features.
* **Web Application :**
* The **Presentation Manager application** assumes user will be listed will all the presentation names with its record ID.
* The user will be aware of the record ID and the region to which it belongs to, to upload the presentation.

* The **RTE application** assumes retrieves the email to be sent from the salesforce records, updates the sender address, and sends the email(s) using AWS SES service.
* **Data Integrity:** The assumption is that data updated and retrieved from Salesforce databases through the application will be accurate, consistent, and up-to-date. Proper error handling and validation mechanisms are in place to maintain data integrity.

1. **Limitations**

User can upload on only **Zip** file with max of **100MB** size limit.

1. **Exclusions**

* This document does not cover the deployment process of the application.

1. **DEFINITIONS & ACRONYMS**

The definitions and acronyms defined in the ITQMS Glossary Standard (SOP-000477729) are the Viatris standard definitions to be used.

*[Additions may be made with unique definitions to this document. Include definitions for all terms which may be unfamiliar to the anticipated audience for this document.]*

*Table 1*  *Definitions & Acronyms*

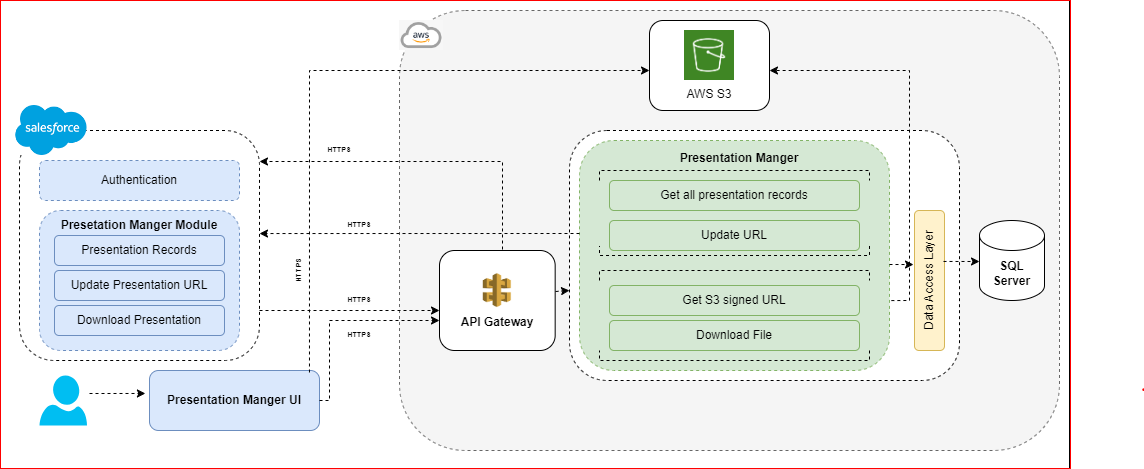
|  |  |
| --- | --- |
| **Abbreviations, Acronyms and Terms** | **Definition** |
| MR | Medical representatives |
| Rest API | Representational State Transfer API |
| AWS | Amazon Web Services |
| SSO | Single Sign On |
| URL | Uniform Resource Locator |
| CI/CD | Continuous integration and continuous delivery |

**3 DATA AND PROCESS FLOWS**

The data and process flow in a Presentation Manager and RTE application developed using Rest API are illustrated below.

**3.1 Ecosystem Architecture of Présentation Manager**

Presentation Manager is an application to be used by Salesforce admin to upload the presentation containing ppt, pdf, thumbnails, mp4, etc. These presentations are available for the medical representative in Media module for download.



* **Figure 1: High Level Architecture**

**User Authentication and Login:**

* The user logs in using ~~their~~ his/her credentials and can use the SSO login.
  + - To access salesforce login page, need to pass request parameters such as client secret, token endpoint, instance URL, ClientId, Callback URL to salesforce

**Presentation Manager Module:**

* User will be listed with all the presentation records using Rest API, on the Presentation manager user interface.
* Admin must be able to upload the file via presentation manager UI so that f iles are saved against the respective records.
* User will be able to filter the records by selecting the currency from the

dropdown available.

* When uploading files, it should fulfill some criteria: File type should be .zip and file size should not be more than 100MB.
* Zip file can contain PDF file, Text file, video and image.
* The uploaded file should be saved in AWS S3 bucket, so that S3 URL is

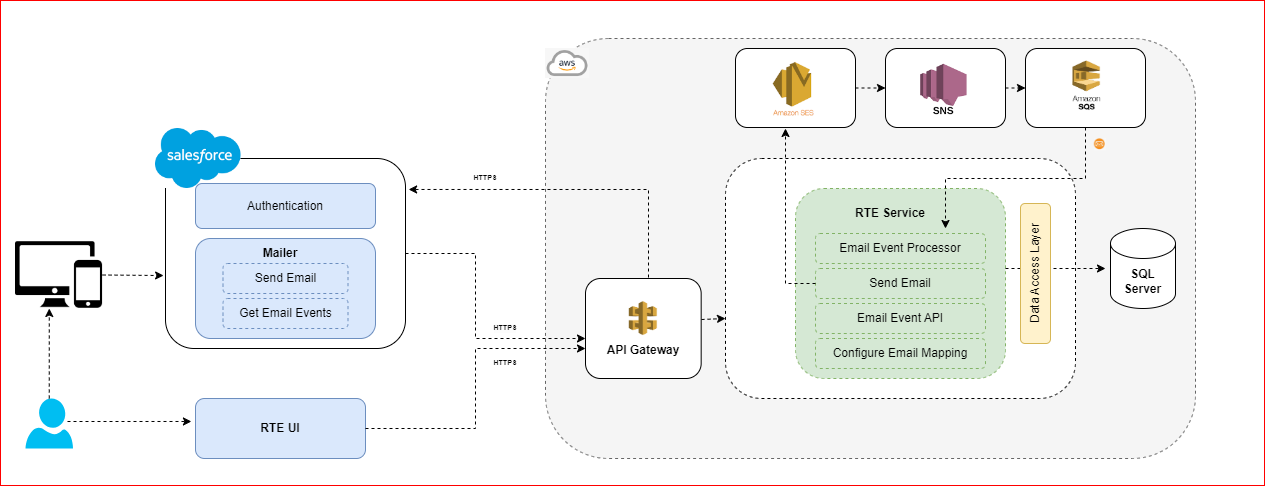
generated.

* As a medical representative, I must be able to download the presentation in My Force's media section so that it can be used for presenting to HCP (Health care pharma).
* The presentation manager uses Db to store metadata.

**Log Out:**

* The user can log out of the application, ending the current session.

**3.2 Ecosystem Architecture of RTE**



**Figure 2: High Level Architecture**

1. **User Authentication and Login:**
2. The user logs in using ~~their~~ his/her credentials and can use the SSO login.
3. To successfully send requests, REST API requires an access token obtained by authentication.
4. To access salesforce login page, need to pass request parameters such as  client secret, token endpoint, instance URL, ClientId, Callback URL to salesforce
5. User authentication will be handled in the API Gateway
6. **RTE Module**
7. Web application and backend server will communicate over HTTPS using REST APIs for exchanging data.
8. RTE UI to configure the email mapping of the user to send the email from salesforce
   * + - 1. **send email :**
     + The user can trigger email from the salesforce UI as well as the mobile application. The user can select the template, define the content of the mail, and trigger the email from salesforce or mobile application. It will make the API call to RTE service to send the email.

**RTE service will take the following actions.**

* Validate the user and check the license of the user which is already

configured in the RTE service.

* + - RTE service will interact with AWS SES to send the email.
    - Store the reference id and the necessary details in the RTE database.
    - We have predefined topic configured in AWS SNS to track the email events. System will capture the following email events only (***sends, failure, rejects, delivered, bounce, spam, delivery delays, subscriptions, open, click***).
    - Every event will be pushed to AWS SQS, this will be consumed by the event processor. RTE service will store the event logs as well as the latest state (Delivered, Bounced, etc.) and user actions (Open, click the link, etc.) of the email.

**(b)    Email Event Processor**

This module will have the logic of processing the events which are there on AWS SQS. All the events of the email will be pushed to AWS SQS and the event processor will process the events and store it in DB.

**(C) Email Event API**

RTE service will provide the API which will pull the event details for the mail which are stored in the DB. Salesforce will consume the API and store it in salesforce.

**D)  Configure Email Mapping**

RTE service will validate the access rights of the user before sending the email. This module will hold the logic of storing the email mapping which will be used to validate the user access.

* This service retrieves the email to be sent from the salesforce records, updates the sender address, and sends the email(s) using AWS SES service.

1. **DETAILED SOFTWARE DESIGN**

This section provides a comprehensive description of how the software will be structured and organized. This section should cover both high-level architectural decisions and low-level design details. 

1. **Development Methods**

The following development methods will be followed in MyForce Application:

* + **Agile Development:**
* Adopt an Agile methodology to allow iterative development, frequent feedback, and continuous improvements. This approach is well-suited for adapting to changing requirements and ensuring that the application aligns with user needs.
* **Scrum Framework:**
* Implement the Scrum framework within Agile. Divide the development process into time-bound iterations called sprints. Each sprint results in a potentially shippable product increment, allowing for regular assessment and adjustments.
* **User-Centered Design:**
* Prioritize user needs and feedback. Involve testing team in the design process through user research, prototyping, and usability testing. This ensures the application meets all the expectations.
* **Cross-Functional Teams:**
* Form cross-functional teams comprising developers, designers, t~~T~~esters and product managers. This diversity of skills fosters collaboration and better decision-making.
* **Continuous Integration and Continuous Deployment (CI/CD):**
* Implement CI/CD pipelines to automate testing, integration, and deployment processes. This ensures that changes are thoroughly tested and can be quickly deployed to production   
  .
* **Version Control:**
* Utilize version control systems (e.g., TFS) to manage code changes, collaborate effectively, and maintain a history of development iterations.
* **Design Patterns:**
* MVC design patterns that are appropriate for Web application to ensure a well-structured and maintainable codebase. Rest API  also provides guidelines for building scalable applications.
* **Modular Architecture:**
* Divide the application into modular components, promoting code reusability, maintainability, and easier testing. This also allows different teams to work on different parts of the application simultaneously.
* **Testing Strategies:**
* Implement a combination of unit testing, integration testing, and user acceptance testing to ensure the application's reliability, functionality, and performance.
* **Documentation:**

Maintain comprehensive documentation, including architecture diagrams, user guides, and development guidelines. This aids in knowledge transfer and future maintenance.

* **Component Level  Explanation**

**Presentation Layer**

Responsible for rendering the user interface, capturing user input, and providing a seamless user experience.

* **View:**
* **Responsibility**: The View is responsible for presenting the user interface (UI) to the user. It displays data and captures user input. View should be lightweight and should mainly handle UI layout and user interaction.
  + - **Tasks**:
* Renders UI elements, such as buttons, labels, forms, and lists, to display information to the user.
* Captures user interactions, such as button clicks, text input, and gestures.
* Receives data from the View Model to display in the UI.
* Forwards user input to the View Model for processing.
* **Example**: In this app, the View would be the screens or pages that users interact with, such as Presentation Page
* **Module View**: It is responsible to presenting the Reusable user interface to the View.

**Business Logic Layer:**

 Manages application workflows, business rules, and data processing, ensuring efficient data flow between components.

* **ViewModel:**
* **Responsibility**: It acts as a mediator between the Data ~~Acess~~Access Layer and the View. It encapsulates the presentation logic and transforms the data from the Model into a format that the View can easily render. It also handles user input and interactions, updating the Model as needed. It prepares and provides data to the View for display.
* **Tasks**:
* Stores and manages the presentation logic for the View. It determines what data to display and how to format it.
* Requests data from the Model and transforms it into a format suitable for the View.
* Contains state and behavior related to the View, such as handling UI events or user input.
* Does not have direct knowledge of the UI's layout or appearance.
* **Example**: In This app, the ViewModel might retrieve data from the Model, format it, and provide it to the View for display.
* **Model:**
* **Responsibility**: The Model represents the application's data and business logic. It defines how data is structured and manipulated. It encapsulates Salesforce SDK structures. It is responsible for managing data retrieval, storage, and manipulation.
* **Tasks**:
* Encapsulates data structures, such as objects or database tables, that represent the core data of the application.
* Contains business logic for data validation, calculations, and operations.
* May interact with external data sources, like databases or web services, to fetch or update data.
* Typically, it does not have knowledge of the UI or user interactions.
* **Example**: In This app, the Model might include classes for Contact, Organization, and Email Activity etc.

**Data Access Layer**

·  Facilitates communication with the SQL server using the Salesforce instance, ensuring data integrity. The Data Access/Manager layer is responsible for coordinating and managing specific tasks that involve multiple components. It helps in separating concerns related to higher-level operations    
   
Service manager could be anything from below, it depends on our requirement and depends on modules. Use Cases for Service Manager here:

* **Data Manager:** Manages Rest APIs requests and responses with the Model layer to fetch data and communicates with the View Model layer to update UI states.

1. **Data Design**

* **Description**: The MyForce Application is built on the Salesforce platform, which utilizes a predefined data model. Leverage Rest API is  interact with Salesforce data, supporting CRUD(create, read, update, delete) operations on Salesforce records from within the MyForce app.

**Features**:

* View and search Salesforce records.
* Edit and update record details.
* Create new records and associate them with relevant Salesforce objects.
* Delete records while maintaining data integrity.

1. **Internal Software Data Structure**

* **Description**: Implement offline data access and synchronization to ensure users can work with Salesforce data even when offline. Also the data structure will be the same as a Salesforce server data structure.

**Features**:

* Cache Salesforce data locally for offline access.
* Synchronize changes made offline with the Salesforce server when online connectivity is available.

1. **Temporary Data Structure**

MyForce app is going to use the Salesforce platform, which utilizes a predefined data model and data structure. So we are not going to use any temporary data structure for this.

1. **Database Description**

We are also understanding the data structure and schema which is used inside the app. We will provide all the details in later versions.

*[Provide the data dictionary giving complete design of the database schema i.e. details of the table with column properties & relationships etc.]*

**Technology Stack:**

* **Development Environment:**
* Visual Studio: Version 17 or higher.
* Framework: .Net7
* Base Windows Version: 10.0

* **Authentication and User Management:**
* Salesforce Cloud: Utilizing Salesforce as the authentication and identity management system.
* OAuth 2.0: Implementing OAuth 2.0 authentication flow for secure user access.
* User Profiles: Managing user roles and permissions using Salesforce user profiles.
* **Application Target:**
* Application: Developed exclusively for downloading files from S3 bucket and update email template and fetch all the emails from salesforce, providing an optimized experience on larger screens.

1. **Narrative for the component**

**User Interface (UI) Component:**

* **Responsibilities:** This component handles the presentation layer of the application, rendering screens, capturing user input, and providing a user-friendly interface.

**Data Management Component:**

* **Responsibilities:** This component deals with data synchronization between the Web app and the Salesforce backend. It manages local data storage, retrieval, and updates.
* **Processing Detail:** Handling data fetching, managing local databases (SQL Server), and ensuring data consistency.

1. **User Interface Design**

**Screen Images:** Wireframes and mockups of key screens, including the customer profile view, appointment scheduler, and product catalog, will be developed. 

**Objects and Actions:** All screen objects, properties, user actions, and validation rules will be identified, ensuring a consistent user experience. 

**Interface Design Rules:** Adherence to Salesforce design guidelines will maintain a unified and intuitive user interface. 

**GUI Components Available:** Standard GUI components such as buttons, text fields, drop-down lists

1. **Interface Design Rules**

Here are interface design rules those will be consider in MyForce application:

**Consistency:**

* Ensure a consistent look and feel throughout the application, including fonts, colors, buttons, and navigation.
* Use standard UI elements for common actions (e.g., buttons, field, list etc) to make the app intuitive.

**Navigation:**

* Implement clear and intuitive navigation, including a home screen or dashboard for easy access to key features.
* Use breadcrumb navigation or a clear hierarchy to help users understand their location within the app.

**Visual Hierarchy:**

* Highlight important elements, such as primary actions or critical information, using size, color, or placement.
* Group related elements together visually to convey their relationship.

**Whitespace:**

* Use whitespace effectively to create a clean and uncluttered interface.
* Space out elements to prevent overcrowding and make the interface more readable.

**Typography:**

* Choose readable fonts and font sizes for both headings and body text.
* Ensure appropriate line spacing (leading) for improved readability.

**Touch-Friendly Elements:**

* Design buttons and interactive elements with a size and spacing that is touch-friendly, especially for mobile devices.

**Error Handling:**

* Clearly communicate errors to users with descriptive error messages.
* Suggest solutions or provide guidance on resolving errors.

**Progress Indicators:**

* Use progress bars to indicate ongoing processes, such as data synchronization.
* Inform users of the progress.

**Responsive Design:**

* Ensure the app's interface adapts to different screen sizes and orientations, especially for iPad devices.

**Offline Experience:**

* Design for an offline experience where possible, allowing users to perform critical tasks even without an internet connection.

1. **GUI Components available**

GUI(Graphical User Interface) components serve different purposes within your application and contribute to its usability and functionality. The selection of components should align with the specific needs of each screen and user interaction, providing an intuitive and efficient user experience.    
Below are some of the GUI components which is going to be used in MyForce application:

**Text Fields:**

* Used for inputting text data, such as names, addresses, or notes.
* May include options for single-line or multi-line input.

**Buttons:**

* Various types of buttons for actions like "Save," "Submit," "Delete," and "Cancel."
* Icon buttons for quick actions.

**Dropdown Menus/Select Boxes:**

* Allow users to select from a list of options, such as currency
* **Lists and Tables:**
* Display lists of items, such as ID, Name,Version,Currency, Upload icon or notifications.
* Tables for structured data presentation with columns and rows.

**Alerts and Modals:**

* Display important messages, confirmations, or error notifications in pop-up dialogs.

**Icons, Images :**

* Use icons for visual cues and branding.

**Text Labels and Headings:**

* Display static text or headings to provide context and information.

**Notifications and Error Handling**

**Notifications:  N/A**

Here's how you can structure documentation for these components:

**Network Connection Errors:**

* **Description:** Handle errors related to unstable or lost network connections.
* **Response:** Display an appropriate error message and provide guidance on re-establishing a connection.

**Data Sync Failures:**

* **Description:** Address errors occurring during data synchronization with Salesforce.
* **Response:** Log the error, attempt retries, and notify the user of the issue. //

**Invalid User Input:**

* **Description:** Handle user input errors, such as missing or incorrect data.
* **Response:** Highlight the erroneous input fields and provide guidance on correction. /

**User-Friendly Messages:**

* Ensure that error messages are user-friendly, concise, and provide actionable guidance to help users understand and resolve issues.

**Documentation:**

Maintain a log of common errors, their causes, and resolutions for reference by support and development teams.

1. **Reports**

N/A 

1. **REFERENCES**

*Table 7  References*

|  |  |
| --- | --- |
| **Document Name** | **Title** |
| SOP-000477729 | ITQMS Glossary Standard |
| SOP-000482185 | Preparation of Specifications |

*Table 2* *References*

1. **ADDENDUMS, SUPPLEMENTS, REVISIONS**

Addendums, supplements, and revisions to this document are permitted. A description of the change must be entered into the Revision History at the end of this document.

During the course of this endeavor, if it becomes necessary to amend, supplement, or revise this document, these changes will be prepared in the format of this document and will be reviewed and approved by those individuals specified in the last revision of the original plan or an appointed designee.

1. **REVISION HISTORY**

*[Note: In the revision history, list the reasons for change.*

*The revision history should be entered in descending order.*

*Keep only the last five versions of the document in the revision history table.]*

|  |  |  |
| --- | --- | --- |
| **Version** | **Description of Change** | **Revised By** |
| 1.0 | Original | *Shridhar Kakkeri* |