**Product Requirements and Specifications Document**

**TERRALOGIC Learning Management System**

|  |  |
| --- | --- |
| **Project/ System** | Learning Management System |
| **Sub-Area/Module** | - |
| **JIRA/SNOW #** | - |

**Document Outline**

[1.](#_heading=h.2et92p0) Overview 5

[1.1.](#_heading=h.tyjcwt) Reference Documents for PRD 8

[2.](#_heading=h.1t3h5sf) Scope 8

[2.1 In Scope Systems/Functionality 8](#_heading=h.4d34og8)

[1.2.](#_heading=h.2s8eyo1) Assumptions and Out of Scope 9

[High level Work breakdown 9](#_heading=h.17dp8vu)

[System Requirements and Specifications 10](#_heading=h.3rdcrjn)

[1.3.](#_heading=h.26in1rg) Requirements – SW Applications (Mobile and Web Apps) 10

[1.4.](#_heading=h.35nkun2) Requirements – Data Visualization and Analytics 11

[1.5.](#_heading=h.1ksv4uv) Requirements – Server & Interfaces 12

[1.6.](#_heading=h.44sinio) Dependencies and ownership(Terralogic and Terralogic) 12

[Key Phases for implementation 14](#_heading=h.2jxsxqh)

[Design Overview 14](#_heading=h.z337ya)

[1.7.](#_heading=h.3j2qqm3) The use case sequence diagram 14

[1.8.](#_heading=h.1y810tw) The high level Architecture & Geo deployment 15

[1.9.](#_heading=h.4i7ojhp) Use-Cases 16

[1.10.](#_heading=h.2xcytpi) Schema 18

[1.11.](#_heading=h.1ci93xb) API 19

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date Modified** | **Created/**  **Modified By** | **Action** |
| 0.1 | 25-Jul-2023 | Laiju V G | Initial Draft release |
| 0.2 | 26-Jul-2023 | Laiju V G | Use-Cases updated |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**APPROVED BY**

1. Terralogic SW Solutions \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date:

(NAME)

2. Terralogic \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date:

(NAME)

**Acronyms**

|  |  |  |
| --- | --- | --- |
| **Sl. No** | **Short name** | **Description** |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 8 |  |  |
| 9 |  |  |
| 10 |  |  |
| 11 |  |  |
| 12 |  |  |
| 13 |  |  |
| 14 |  |  |
| 15 |  |  |
| 16 |  |  |
| 17 |  |  |
| 18 |  |  |
| 19 |  |  |

# 

# Overview

The Learning Management System (LMS) is a comprehensive platform designed to enhance employee learning and knowledge about their assigned projects. It offers a range of content formats, including e-documents, video learning, and hard copy books, accessible based on user roles and project affiliations. The LMS facilitates content access, tracks user activity, enables content requests, and supports performance improvement programs for a more efficient and engaging learning experience.

The Terralogic Learning Management System is proposed to be implemented on a portal for reserving content available in BLR2-2F Rack & digital content on Terralogic servers/linked servers which is available for the employees of ODC system defined in BLR2-2F.

The system at this point is solely restricted to ODC employees of BLR2-2F only.

The complete solution will have Front-End web-portal that is SSO authorized using the Terralogic email.

Services in the backend will allow import of inventory, reserving the books in the inventory, getting notifications of due dates or new books added in the inventory.

**Key Features:**

* **Role-Based Access Control:**
  + Users can access content relevant to their roles and projects, ensuring targeted learning experiences.
  + Restricted access to content ensures sensitive information is only available to authorized personnel.

* **Library Book Availability:**
  + Users can check the availability of hard copy books in their location.
  + The system will display the book's location if it is not available at the user's current site, promoting efficient resource utilization.

* **Video Learning Management:**
  + Administrators can add and categorize video content based on projects and domains.
  + Employees can easily access videos relevant to their work, contributing to project-specific skill development.

* **Content Request System:**
  + Employees and managers can request specific learning materials not currently available in the LMS.
  + Administrators review and process these requests, ensuring a dynamic and up-to-date content repository.

* **User Activity Tracking:**
  + The system tracks user interactions, providing administrators with insights into content consumption patterns.
  + Administrators can identify popular content and understand user preferences.
* **Performance Improvement Program (PiP):**
  + Managers can assign tasks and projects to employees for skill enhancement.
  + Completion of assigned tasks can lead to performance reviews and potential rewards, fostering a culture of continuous learning.
* **Gamified Learning Experience:**
  + Assignments with quizzes can be gamified to motivate employees and increase engagement.
  + Achievements and rewards create a sense of accomplishment and encourage active participation.
* **Manager-Exclusive Content:**
  + Monthly Business Reviews (MBRs) and Quarterly Business Reviews (QBRs) are accessible only to managers.
  + Non-managerial users can view the presence of these reviews without accessing their content.
* **Admin Privileges:**
  + System admins can change roles of the users and manage who can access what content, also they can share these roles with managers so that managers can also take quick actions for ease of use

**Additional Features:**

* **Peer-to-Peer Learning:**
  + Enable employees to share their expertise by creating user-generated content, such as blogs or knowledge-sharing forums.
* **Mobile Learning App:**
  + Develop a mobile app for on-the-go access to learning materials, promoting continuous learning outside the workplace.
* **Skill Badges and Certifications:**
  + Award digital badges and certifications upon completing specific courses or achieving certain skill levels

### Reference Documents for PRD

|  |  |  |
| --- | --- | --- |
| **#** | **Document Name & Why Relevant** | **Link to the document** |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |

# 

# Scope

### 2.1 In Scope Systems/Functionality

1. Design, Develop and Integrate LMS- Learning Management System to offer a platform for digitial (eBooks/curated videos) and physical content(books) that can be reserved through a centralized portal and tracked for completion.
   1. Identify a solution to build the components of the LMS
   2. Develop SW application for Web, Mobile - Android and IOS operating systems and implement the features as provided in the requirements in line with Terralogic Brand UI/UX
   3. Develop User Interface to enrich the User Experience.
2. The proposed system initially is planned to be deployed stand-alone but proposed to be integrated with HR-OS subject to business decisions agreed by Terralogic management.
3. Deliver and deploy the Solution SW across Dynaquest / appropriate cloud.
4. Updated SW release fixing all the bugs after initial deployment.
5. Provide all the documentation for the HA system includes
   1. Design specifications Documents
   2. SW Applications or source code
   3. User guide
   4. Installation document
   5. Maintenance guide of the system.
6. The Language for all the Apps (mobile & Web) shall be English.
7. After Ver.1 of the system released, Next version (Ver. 2) will be taken up based on User feedback/experience like feature, Visual UI/UX enhancement, MobileApps etc. will be taken up as extension to the project with an approval of extended SOW from Terralogic.

### Assumptions and Out of Scope

1. The project Scope is defined and as per the agreed Product requirements, if any changes to the same will be notified to Terralogic for approval both time schedule and cost.
2. The Infrastructure SW any specific tools shall be provided by DQ IT team
3. The software initial version is solely for internal consumption of employees in BLR2-2F & not any commercial expansion at the time of preparing this document.

# High level Work breakdown

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID# | Category | Item Description |  | Remarks |
| W000 | Initial integration-Web Deployment | Phase1 |  |  |
| M000 | Mobile Applications- Android , IOS | Phase2 |  |  |
| D000 | Data Visualisation and Analytics | Phase 2 |  |  |
| C000 | Cloud interface and Server Integration | Phase1 |  |  |
|  |  |  |  |  |

# System Requirements and Specifications

# Requirements – SW Applications (Mobile and Web Apps)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID# | Category | Business Requirements– Purchase Requisition System | Basic/  Phase 1,2,3  (B/P1/P2/P3) | Premium  (B+) | Importance  (H, M, L ,I) |
| MW001 | Mobile/Web App | A Web & Mobile application software shall allow to view, control, and monitor and organize the inventory of learning content The Web application will be limited to Browsers Chrome & Mozilla Firefox While the Mobile Apps will be in Andriod & iOS latest versions. | B,P1 |  | H |
| MW002 | Mobile/ Web App | The application software shall offer User Management, Dashboards and booking interfaces. | B, P2 |  | H |
| MW003 | Mobile/ Web App | (User registration )  The User Management shall allow user to register the TBD | B, P1 |  | H |
| MW004 | Web App | The Dashboard shall allow access to features such as Search Inventory, New content uploads, Reservation, Administration(Admin RBAC),Analytics | B, P2 |  | H |
| MW009 | Mobile/ Web App | (Configure and view alerts)  The User able to configure for Alerts & messages or Schedules via SMS & Email alerts required to notify users for important notifications | B,P1 |  | M |
| MW012 | Mobile/ Web App | The content presented on the Web and Mobile versions shall offer consistent UI on all platforms in order for customer to have near same experience on using the interface on Desktop/Smartphones/Tablets. | B |  | H |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# Requirements – Data Visualization and Analytics

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID# | Category | Business Requirements– Purchase Requisition System | Basic  (B) | Premium  (B+) | Importance  (H, M, L, I) |
| D002 | Data visualization and Analytics | This App shall provide Reports/Analytics of time bound filters of utilisation of resources. | B |  | H |
| D003 | Data visualization and Analytics | The User able to view data statistics for LMS content consumption over a period of time/day/week/month /year etc. provided a condition the interface API available to integrate to Terralogic Learning Management System |  | B+ | M |
| D004 | Data visualization and Analytics | User shall able to estimate the future consumption and able to plan their budget effectively. |  | B+ | M |
| D005 | Data visualization and Analytics | Data visualization shall be in form of pictorial representation (graphs/charts) based on UX design |  | B+ | M |

# Requirements – Server & Interfaces

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID# | Category | Business Requirements– Interfaces | Basic/  Phase 1  (B/P1) | Premium  (B+) | Importance  (H, M, L, I) |
| S001 | Server | The Terralogic Cloud shall have Application Server running Web service and Backends that collect and offer data to the Web & Mobile Apps.  Initial deployment will only target US geography. | B, P1 |  | H |
| S002 | Server | The Terralogic Cloud shall have a scalable infrastructure to support dynamic vertical/horizational scaling | B |  | H |
|  |  |  |  |  |  |

# Dependencies and ownership(Terralogic and Terralogic)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SL.No.** | **Item** | **Terralogic** | **Terralogic** | **Remarks** |
| 1 | **Product Design development** | Define Scope | Review and Approve the Scope | PRD |
| 2 |  | Prepare Product requirements and Specifications document (PRD) | Review and Approve the PRD | Any changes from based document will need to be reviewed and approved |
| 3 |  | Develop Hardware, Mobile Apps(Android,IOS), Web app design development | Approve any design changes |  |
| 4 |  | Any scope changes will be Discussed through CCB(Change control board)/or any other authorized i.e.Product mgmt team | Change Approval (which includes Cost/Schedule) | CCB includes both TL and KT members |
| 5 | **Project Data Sharing/Repository** | Customer Specific | Terralogic to provide any sharepoint/share drives for secure data access and recovery |  |
| 6 | **Bug Tracking/Agile PM tool** | Customer Specific | **GitHub IssueTracking System JIRA access provided** |  |
| 7 | **Project Management** | PMs, SM part of Project Governance Model (Laiju ) | Terralogic to share the Governance model if any from their side for interactions and updates/reviews |  |
| 8 | **Status Update/Meetings** | Weekly (PM along with leads) |  | **Done** -Using Excel sheet for tracking MOM |
| 9 |  | Escalation mechanism as informed in SoW | Terralogic to provide SPoC |  |
| 13 | **UI/UX and ID** | UI/UX specification can be provided by Design Team. | UI/UX Design approval for SW apps, Web App | Terralogic Design team if available to leverage |
| 14 | **Validation Testing** | TL set up all the required SW/HW in the real environment to test the system | Customers IT Team to facilitate an infrastructure for integration testing. |  |
| 16 | **Product Documentation** | Deliver installation guide, User guide | e-Documentation approval and inform any specific format |  |

# Key Phases for implementation

The table below lists the key phases the project will be implemented and delivered.

**Implementation Approach will be carried out in 3 Phases-Phase 0, Phase 1, 2.**

|  |  |  |
| --- | --- | --- |
| **#** | **Key phases/Milestones** | **Remarks** |
| 1 | **Phase 0 -- Primary Backend Services to be built** | Input for Phase 1 |
| 2 | **Phase 1 – UI-Services and new modules**   1. **Integrated UI-Services per UI/UX design teams proposal** 2. **Integrate to HR-OS** 3. **Testing the same in the real time environment** | To be planned |
| 3 | **Phase 2 – Data Analytics on usage** | To be planned |
| 4 | **Phase 3 -- Final Deployment of the solution to HR-OS for external customer deployments.** | TBD |

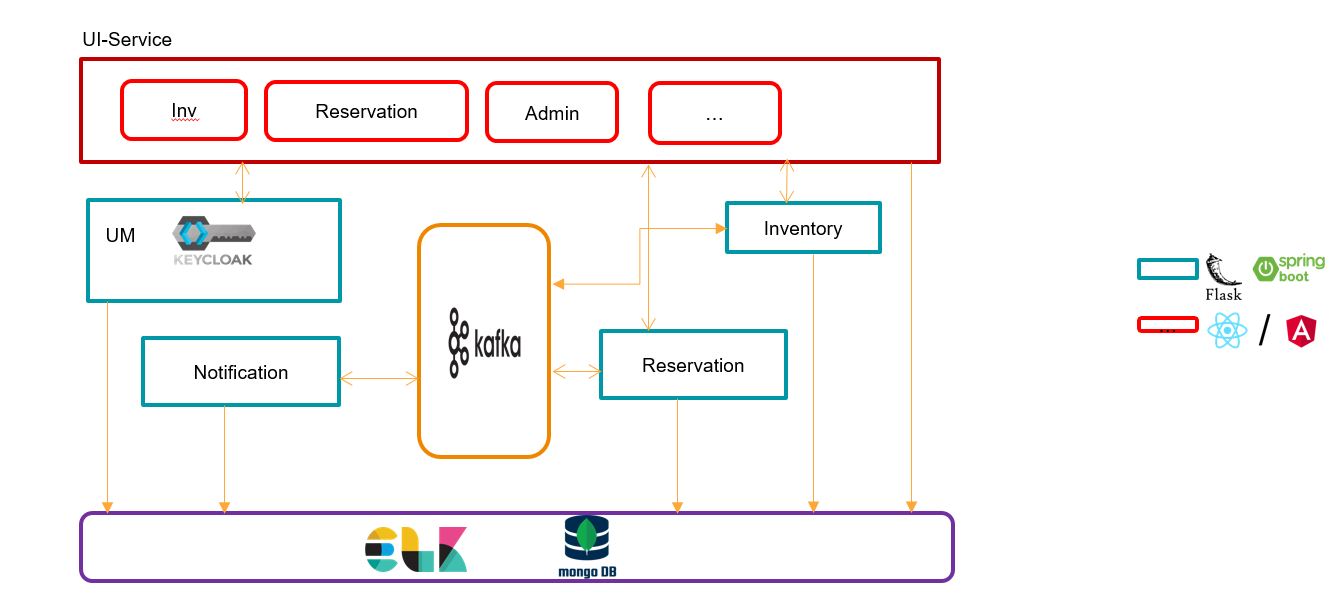
# Design Overview

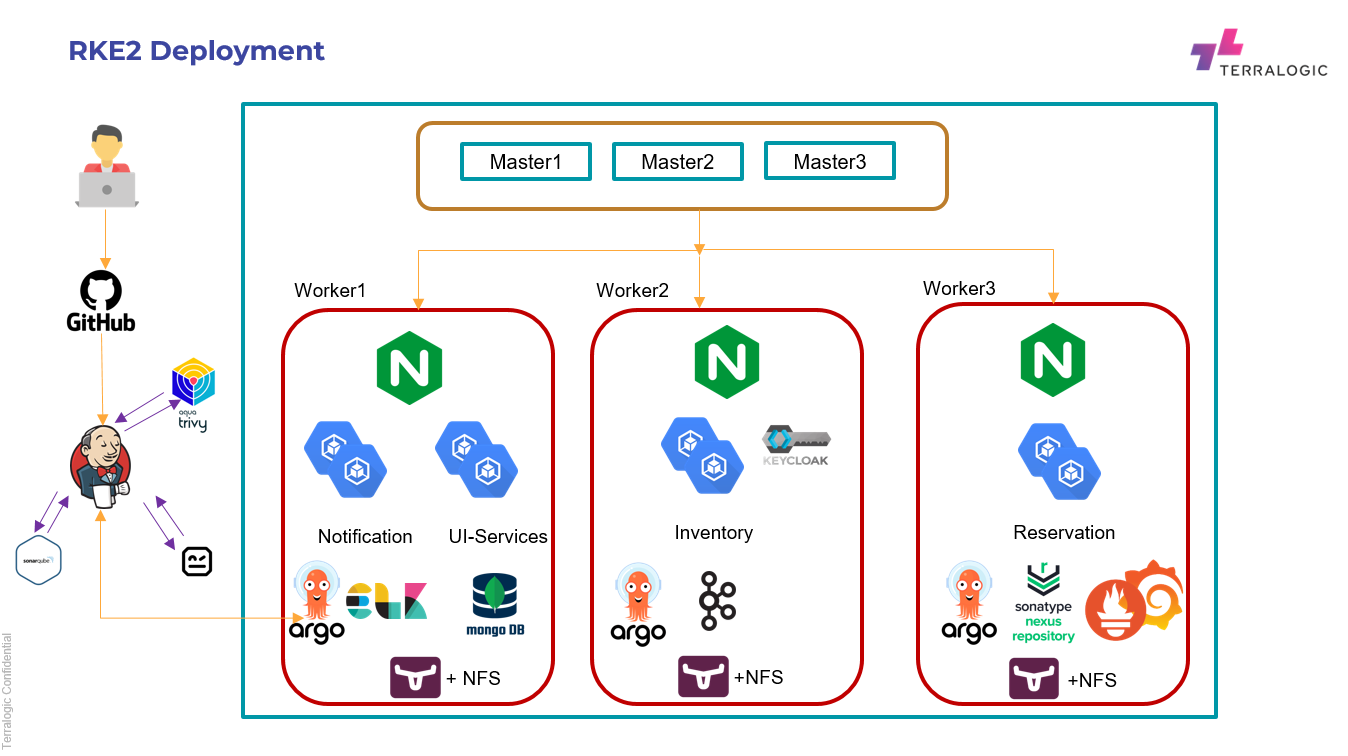
### The use case sequence diagram

The Use case flow diagram representing the map of functions offered by the Web & Mobile Applications is as embedded below.

To be Designed

### The high level Architecture & Geo deployment





### Use-Cases

Following table covers the list of scenarios that will be covered in the initial implementation across all micro-services.

**Inventory**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.no.** | **Use-case** | **How this is executed** | **What is needed in schema** | **Remarks** |
|  | Creation of new books/E-books/Videos | * The admin will be creating the essential details, i.e. **book id,book name,description,type** and a **logo** representing whether it's a **Books/E-books/videos**. * The inventory service generates a unique **inv\_id** for the book and adds it to the database. * If the book is available online,the user will be provided with the **url reference** to access the book. It may be a **PDF** or a **video** based on user preference. | It should show -  **inv\_logo,inv\_id,inv\_name,inv\_desc,inv\_type,inv\_blob,inv\_archieve\_status** |  |
|  | View Book Details | * When the user gets the access to inventory service-they can view the following -   **inv-logo,inv-id,inv-name,inv-desc,inv-type,inv-blob,inv\_archie\_status**.   * The service displays the book/e-book and video information to the user. | It should show -  **inv\_logo,inv\_id,inv\_name,inv\_desc,inv\_type,inv\_blob,inv\_archieve\_status** | Pagination is required. |
|  | Update Book Details | * Admin have the privilege to update certain details,such as,**inv\_type.** Suppose, a new type of option is available for the certain book that can be updated. * Updated changes will be saved in the database. | It should show -  **inv\_logo,inv\_id,inv\_name,inv\_desc,inv\_type,inv\_blob,inv\_archieve\_status** |  |
|  | Deletion of a Book/E-book/Video | * Admin can remove certain books from the database if it is not available. Ex, Suppose the link given for a particular e-book reaches the expiration of that link,then the admin has to remove it from the database. * Deleted changes will be saved in the database. | It should show -  **inv\_logo,inv\_id,inv\_name,inv\_desc,inv\_type,inv\_blob,inv\_archieve\_status** |  |
| 5. | Searching for a Book/E-book/Video | * When there will be n-number of books/e-books/videos,the user will face issues in searching for a particular.So, in order to make it easy,searching options plays an important role. * **This User case is for future discussion.** | For future discussion. |  |

**Reservation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.no.** | **Use-case** | **How this is executed** | **What is needed in schema** | **Remarks** |
| 1 | **View Reservations**  **(My-Reservations)** | The user wants to view their reservations/bookings by navigating to the “My  Reservations” screen.  This screen should display a list of their reservations along with relevant details. | It should show  records by  reservation\_ID  reservation\_created\_date  Inventory\_logo Inventory\_name Inventory\_description reservation\_status (Requested/Issued/Cancelled)  reservation\_status\_comments  reservation\_expiry | Must be paginated |
| 2 | **Create Reservation** | The user wants to create a reservation by selecting one or more contents from the inventory listing screen. The user can book a maximum of three contents in a single reservation. When the user clicks the "Book" or "Reserve" button, the reservation service checks the number of reservations made by the user in the current calendar month through the RBAC API of the Admin.  If the user has made three reservations already in the current month, no further reservations are allowed. | It should push the above reservation records with relevant data. |  |
| 3 | **Update Reservation** | The "Update Reservation" use case allows users to modify their existing reservations, primarily for cancellation purposes. However, as the cancellation functionality is already covered under the "Delete Reservation" use case, the need for a separate update operation is not explicitly required at this point. | NA |  |
| 4 | **Delete the reservation** | The "Delete Reservation" use case allows users to cancel their existing reservations when they no longer need the booked item or when it is waiting for allocation by the Admin.  For inventory items that have already been received or allocated, canceling the reservation is equivalent to returning the physical book or canceling access to the e-content (e-Book/Video). The Admin service plays a role in updating the system with relevant cancellation information. | It should push the reservation records with relevant cancellation information.  reservation\_ID  reservation\_created\_date  Inventory\_logo Inventory\_name Inventory\_description Reservation\_status (Requested/Issued/Cancelled)  reservation\_status\_comments  reservation\_expiry |  |
| 5 | Search API | For Future TBD | For Future TBD |  |

**Notification**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SL .NO** | **Use-case** | **How this is executed** | **What is needed in schema** | **Remarks** |
| **1** | **Book Due Date Reminder** | Send an email or notification to users a few days before their borrowed books are due to be returned. Include details of the books borrowed and the due date to avoid late returns. | It should be able to push the data like  **user\_id,**  **user\_email\_address,**  **inv\_id,**  **inv\_name,**  **description,**  **due\_date,**  **remainder\_date,**  **reservation\_expiry** |  |
| **2** | **Overdue Book Notification** | For users who have not returned books on time, send a reminder email or notification informing them about the overdue status. | It should be able to push the data like  **user\_id,**  **user\_email\_address,**  **inv\_id,**  **inv\_name,**  **due\_date,**  **overdue \_days,** |  |

### Schema

This section captures anticipated schema requirements.

**Inventory**

| inv\_logo | inv\_id | inv\_name | inv\_desc | inv\_type | inv\_blob | inv\_archieve\_status |
| --- | --- | --- | --- | --- | --- | --- |
| image | 47586767383838 | “Learning Python” | A book to bootstrap python deployment | BOOK | NA | Active/Inactive/  Deleted |

|  |  |  |  |
| --- | --- | --- | --- |
| Field Name | Description | Usage | Remarks |
| inv\_logo | (Image)  Logo of the inventory | Exposed | Logo depicting the type of inventory from the list  inv\_type |
| inv\_id | Numeric [32 digits] | Internal & not exposed |  |
| inv\_name | String [define max length] | Exposed |  |
| inv\_description | String [define max length] | Exposed |  |
| inv\_type | Custom Keyword  [BOOK/e-BOOK/VIDEO] | Exposed |  |
| inv\_blob | Blob | Exposed | In case of Book (Image URL reference of Book, URL of e-BOOK, URL of video) |
| inv\_archieve\_status | String [define max length]  Boolean | Exposed | Book/e-book or video inactive | delete |

**Reservation**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| reservation\_ID | Reserved\_user | reservation\_created\_date | Inventory\_id | Inventory\_name | Inventory\_logo | Reservation\_status | Reservation\_status\_comments | Reservation\_expiry\_date |
| 47586767383594 | “sihi” | 01-01-2023 | 47586767383838 | “Learning python” | image | “Issued” | “please return the book by feb 1st” | 01-02-2023 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field\_Name** | **Type** | **Description** | **Useage** | **Remarks** |
| Reservation\_id | Integer (Primary Key) | Unique identifier for each reservation | Exposed | Primary key for quick and efficient lookup. |
| Reserved\_user | String | User who made the reservation | Exposed |  |
| Reservation\_created\_date | Date/Time | Date and time when the reservation wascreated | Exposed |  |
| Inventory\_logo | BLOB/Binary | Binary data representing the inventory logo | Exposed |  |
| Inventory\_name | String(URL) | Name of the reserved inventory item | Exposed |  |
| Inventory\_description | String | Description of reserved inventory item | Exposed |  |
| Reservation\_status | String | Status of the reservation | Exposed |  |
| Reservation\_status\_comments | String | Additional comments or notes related to the reservation status | Exposed |  |
| Reservation\_expiry\_date | Date/Time | Date and time when the reservation expires | Exposed |  |

**Notification**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **user\_id** | **user\_email\_adresss** | **inv\_id** | **inv\_name** | **description** | **due\_date** | **remainder\_date** | **reservation\_expiry** | **notification\_archieve** | **overdue\_days** |
| (Value to be filled later when adding the module) | (Value to be filled later when adding the module) | 4758676738383 | “Learning python” | This message is notify that book/video/ebook will expire in the particular time | 09-09-2023 | 09-09-2023 | 09-09-2023 | active/deleted/inactive | 4 |

|  |  |  |  |
| --- | --- | --- | --- |
| Field Name | Description | Usage | Remarks |
| user\_id | Numeric [32 digits] | Internal & not exposed |  |
| user\_email\_address | String [define max length] | Exposed |  |
| inv\_id | Numeric [32 digits] | Internal & not exposed |  |
| inv\_name | String [define max length] | Exposed |  |
| due\_date | Date | Exposed |  |
| remainder\_date | Date | Exposed |  |
| reservation\_expiry | Date /Time | Exposed | The number of days the item is overdue. Calculated as (current date - due date) if overdue, else 0. |
| overdue\_days | Numeric (Integer) | Exposed |  |

**Administration**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |

**User-Management**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |

### API

The deployment below is a draft design on a Kubernetes orchestration platform.

**Inventory**

As per considering the use cases and schema following are the requirements for the number of API.

1. **Creation of the inventory -** Here, the **create operation(POST method)** takes place. Admin needs to create new books and add the details.

Two API are required for this-

* If Admin records a book/e-book/video individually, in that case we require one API.
* If Admin records a book/e-book/video in bulk, in that case it requires another API.

Upon considering, the use-cases two API are required for creating operation.

1. **Viewing of the inventory -** Here, **read operation(GET method)** takes place. Users will get the access to view the book/e-book/video.

Returning API is required -

* One API is required for viewing the book and its details.Here, we need to add **pagination** for the same API. When there will be n-number of books, need to make sure that **Users can view upto 10 books in one page**.According, to total number of books pages can be decided.

1. **Updating the inventory -**  Here, the **update operation (PUT method)** takes place. Admin needs to update the details of a particular book.

Two API are required for this -

* If Admin chooses to update the book details individually, in that case we require one API.
* If Admin chooses to update the book details bulk, in that case it requires another API.

1. **Deletion -** Here, the **delete operation (DELETE method)** takes place.

Admin needs to delete certain book/e-book/video can be done through this method.

Two API are required for this -

* If Admin chooses to delete book/e-book/video individually, in that case we require one API.
* If Admin chooses to delete book/e-book/video bulk, in that case we require another API.

After reviewing the use-cases and noting the required number of fields, I have made an approximate count of the APIs needed for the inventory service.

**Reservation**

**1.View Reservations API(Read):**

* **One API IS used to view the reservation details**

This API allows users to view their reservations by navigating to the "My Reservations" screen. It retrieves and returns a list of reservation records along with relevant details, such as reservation ID, creation date, inventory details, status, comments, and expiry date. The API should support pagination to handle large numbers of reservations efficiently.

**2.Create Reservation API(Post):**

* Single API-a single reservation API that allows the user to create one reservation at a time, selecting one or more contents within that reservation, would be suitable for implementing the "Create Reservation" functionality. The API would handle the necessary checks, validation, and data processing to ensure that the user can book a maximum of three contents in a single reservation and that the user's eligibility is verified before proceeding with the reservation creation.

The "Create Reservation" API enables users to create a new reservation by selecting one or more contents from the inventory listing screen. When the user clicks the "Book" or "Reserve" button, this API is called to handle the reservation creation process. It should validate the user's eligibility based on the maximum allowed reservations per month and interact with the RBAC API of the Admin for this purpose. Upon successful validation, the reservation records should be pushed to the database.

**3.Update Reservation API(PUT):**

The "Update Reservation" API would handle the modification of existing reservations, primarily for cancellation purposes, as mentioned in the use cases. It would allow users to update the status of their reservations, such as changing the reservation status from "Requested" to "Cancelled."

**4.Delete Reservation API(DELETE):**

The "Delete Reservation" API allows users to cancel their existing reservations. When a user no longer needs the reserved item or the item is waiting for allocation by the Admin, the user can use this API to cancel the reservation. The API updates the reservation status to "Cancelled" and may involve interactions with the Admin service to handle cancellations for received/allocated inventory items.

**Notification**

1. **Book Due Date Reminder API:**

**Endpoint**: **POST** /api/book-due-reminder

Description: Create a notification for sending a reminder to users before their borrowed books are due to be returned.

The "POST /api/book-due-reminder" API endpoint is used to create a notification for sending a reminder to users before their borrowed books are due to be returned. It enables the library management system to proactively notify users about the upcoming due dates of their borrowed books, ensuring that they return the items on time and avoid late returns.

2. **Overdue Book Notification API:**

**Endpoint**: **POST** /api/overdue-book-notification

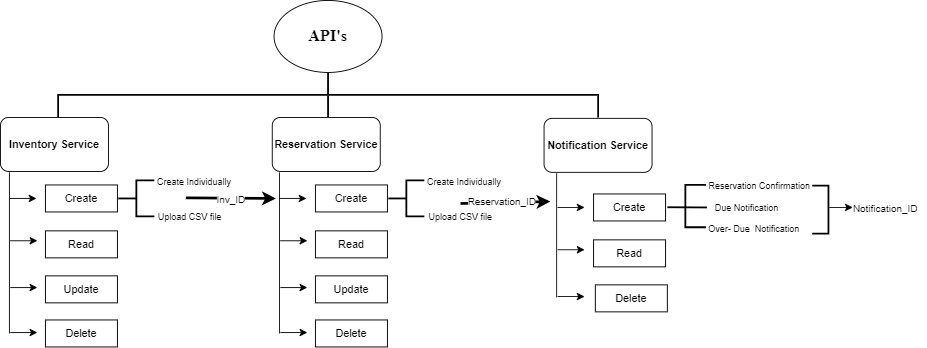
**Description**: Create a notification for sending a reminder to users who have not returned books on time.

The "POST /api/overdue-book-notification" API endpoint is used to create a notification for sending a reminder to users who have not returned books on time. It allows the library management system to notify users about the overdue status of their borrowed books, reminding them to return the items as soon as possible to avoid further delays .

**Administration**

**User-Management**

**1.12. Flow-Diagram**



\*\*\*End of document\*\*\*