Rimsys	# Software Design Specification
Project Title:	Document and Task Manager
Date Created:	Aug 02, 2022
Author:	Lemar Gray
Intended Audience:	Project Team

Revision History

Version	Description	Author	Date
1.0	Initial Design	Lemar Gray	Aug 02, 2022

Table of Contents

1. Overview

This document outlines the approach to implementing a document and task manager system. The resources needed to successfully implement this system includes a Backend Developer, Frontend Developer, Quality Assurance Tester, and DevOps Engineer. The project should take approximately two months to complete. The tech stack was chosen according to the current skillsets of the team. The tech stack includes: Laravel on the backend, VueJs version 2 on the frontend, Amazon AWS for the cloud and MySQL for the database.

2. Requirement Summary

The business is requesting that a document and task manager system be implemented with the following features:

- The system should allow users to add documents
- The system should version documents
- The system should allow users to add tasks to a document
- The system should allow users to download documents

3. Clarifications Needed

- Is the client document management system the same as document and task manager?
- When a user add documents to the system is this the same as uploading a document?
- What file types should the system accept?
- Could you elaborate on how a user will add tasks to a document?
- Should documents be automatically versioned when a user uploads new version of the same document?

- How much users are expected to use the system within the first three months after its launch?
- How much users can the client document management system handle currently?
- What is the maximum size of a document a user should be allowed to upload?

4. Assumptions

- The document and task manager will be developed from scratch using the tech stack that was outlined.
- The document created by the clients using the document manager will be stored in the client document management system.
- The document and task manager system will communicate with the client document management system via a restful API.
- A user will add documents by uploading the document using the document and task manager system.
- A user will add tasks to a document in a to-do list style using the document and task manager system. This means that the tasks will not be added in the document itself but will be associated with the document at the database level.
- A document will be versioned every time the user uploads a new version of the same document.
- The client document management system was already architected to manage the load.
- A user should only be allowed to add tasks to a document created by that user.
- A user should only be allowed to version a document created by that user.
- A user should only be allowed to download a document created by that user.
- A user should be able to download a versioned copy of a document.

- A user should only be able to download a version copy of a document created by that user.
- A user should only be allowed to upload a word or pdf document.
- A user should not be allowed to upload a document that is more than 50MB.

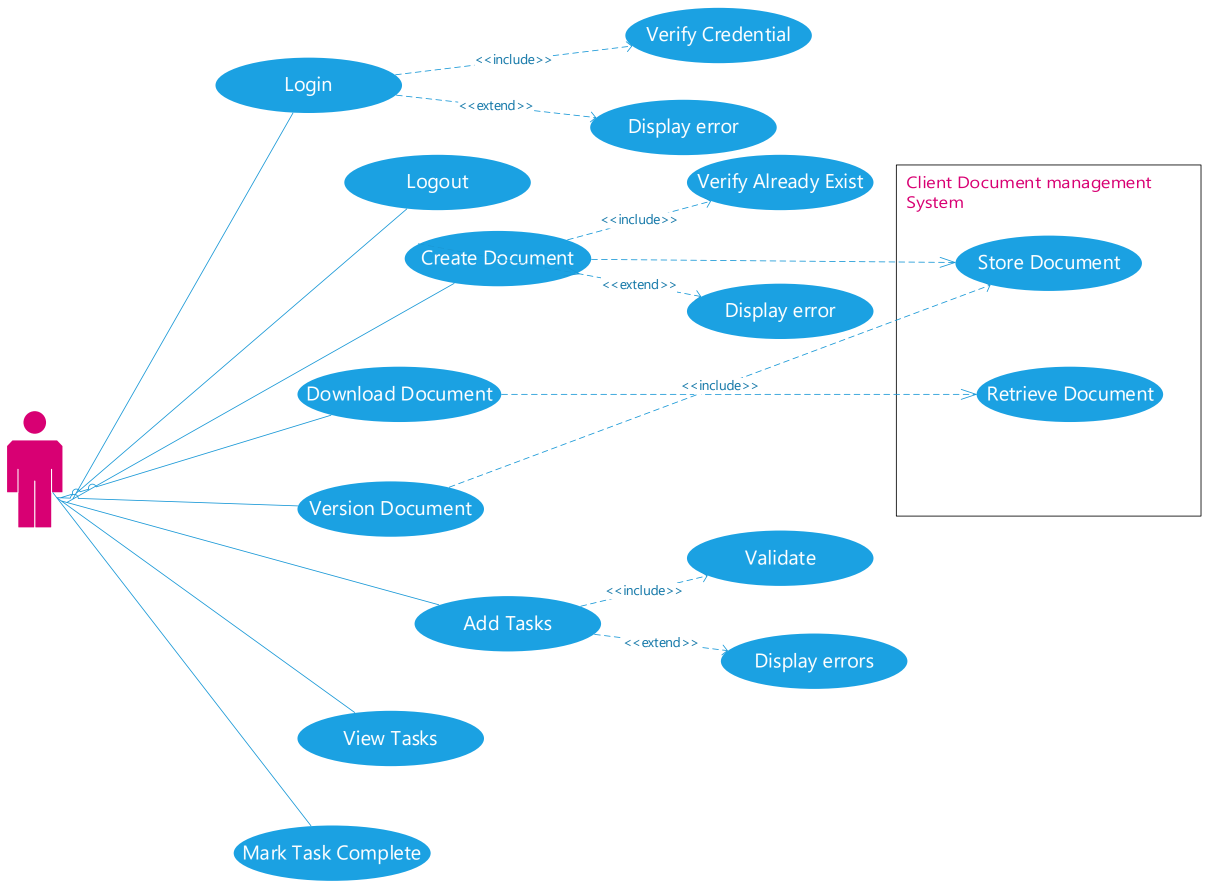
5. Dependencies

The following are the dependencies for the implementation of this system:

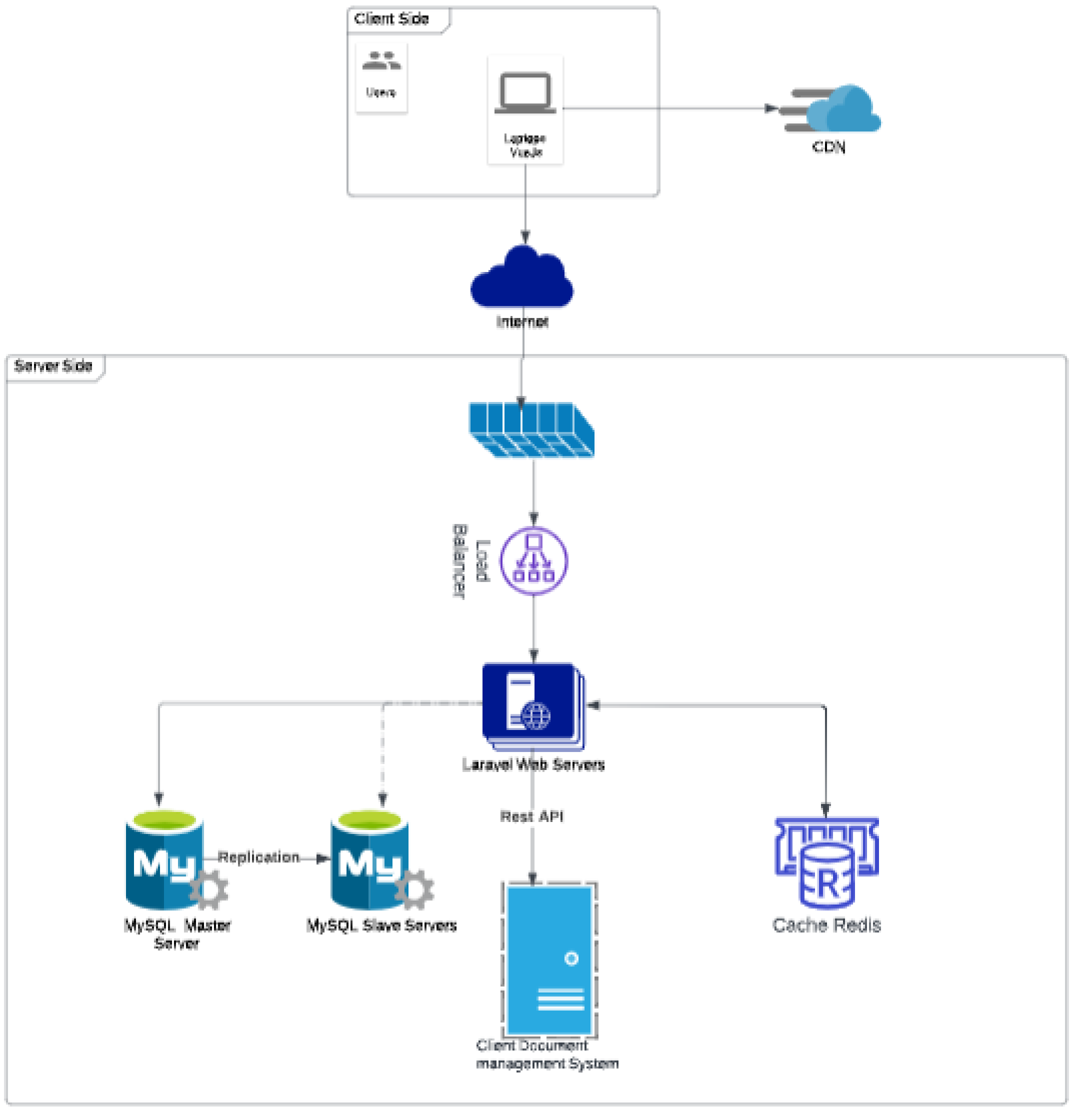
- The client document management system
- Completion of the UI design

6. Detail Design

6.1. Use Cases



6.2. Architecture Design



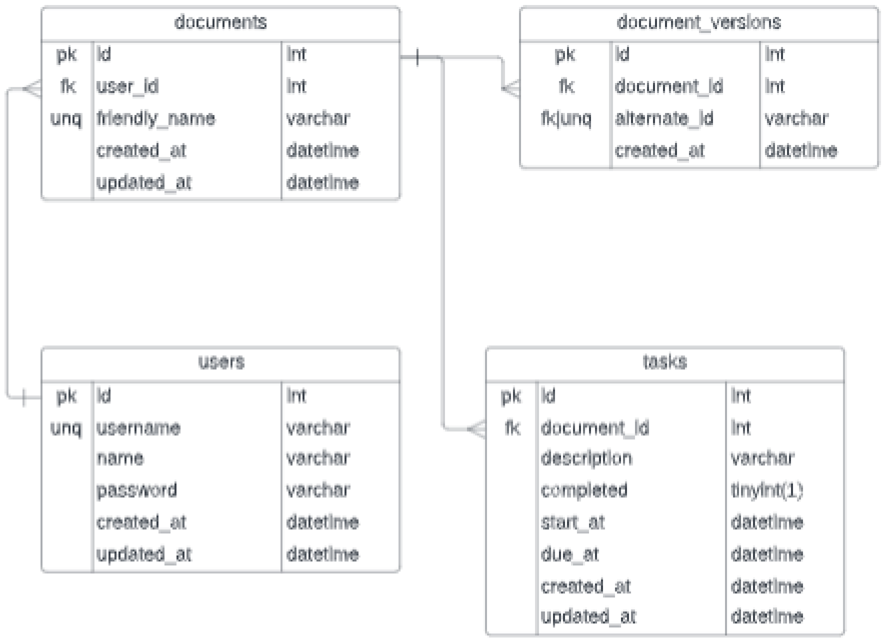
6.3. Database Tables

Below is a list of all the database tables that should be implement for this system.

Table	**Description**
users	Store users and their credentials
documents	Store the documents created by users
document_versions	Store each document upload by a user has a separate document version
Tasks	Store tasks related to a document version and tracked if completed

6.4. Table Relationship

Constraint: The friendly_name and user_id combination on the documents table must be unique.



documents

pk	id	int
fk	user_id	int
unq	friendly_name	varchar
	created_at	datetime
	updated_at	datetime

document_versions

pk	id	int
fk	document_id	int
fk|unq	alternate_id	varchar
	created_at	datetime

users

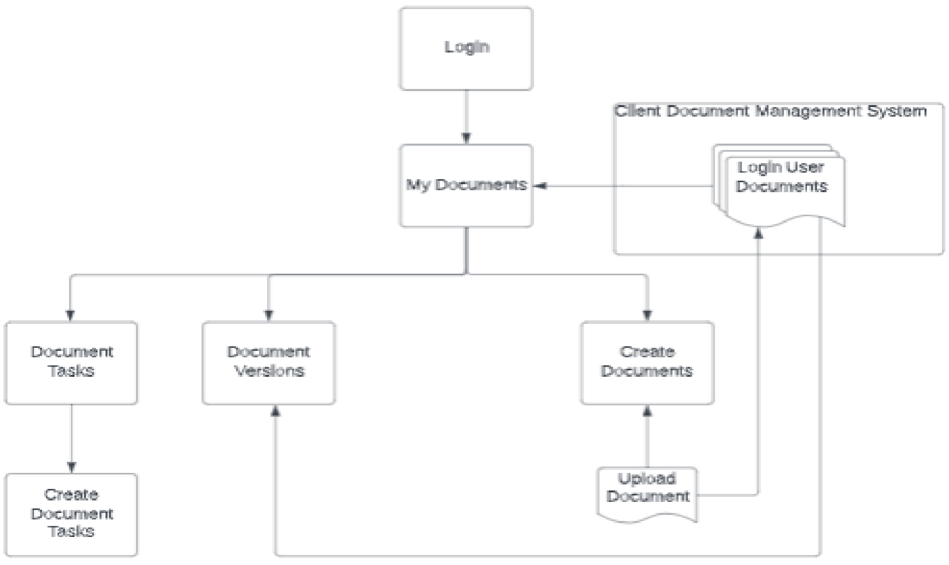
pk	id	int
unq	username	varchar
	name	varchar
	password	varchar
	created_at	datetime
	updated_at	datetime

tasks

pk	id	int
fk	document_id	int
	description	varchar
	completed	tinyint(1)
	start_at	datetime
	due_at	datetime
	created_at	datetime
	updated_at	datetime

6.5. Screens/Pages

6.5.1. Screen Flows



6.5.2. Screens

6.5.2.1. Endpoints

Http Method	Request Path	Description
GET	/login	Get the login page
POST	/login	Login an user
GET	/documents	Get a paginated list of the login user's documents
POST	/documents	Create a new document
GET	/documents/{id}/tasks	Get the specified document for the login user
POST	/documents/{id}/tasks	Add a task to the specified document belonging to the login user
GET	/documents/{id}download	Allow the login user to download the specified document
GET	/documents/{id}/versions	
GET	/documents/{id}/versions/{id}/download	Get a paginated list of the login user's document versions base on the specified document
POST	/logout	Logout a user

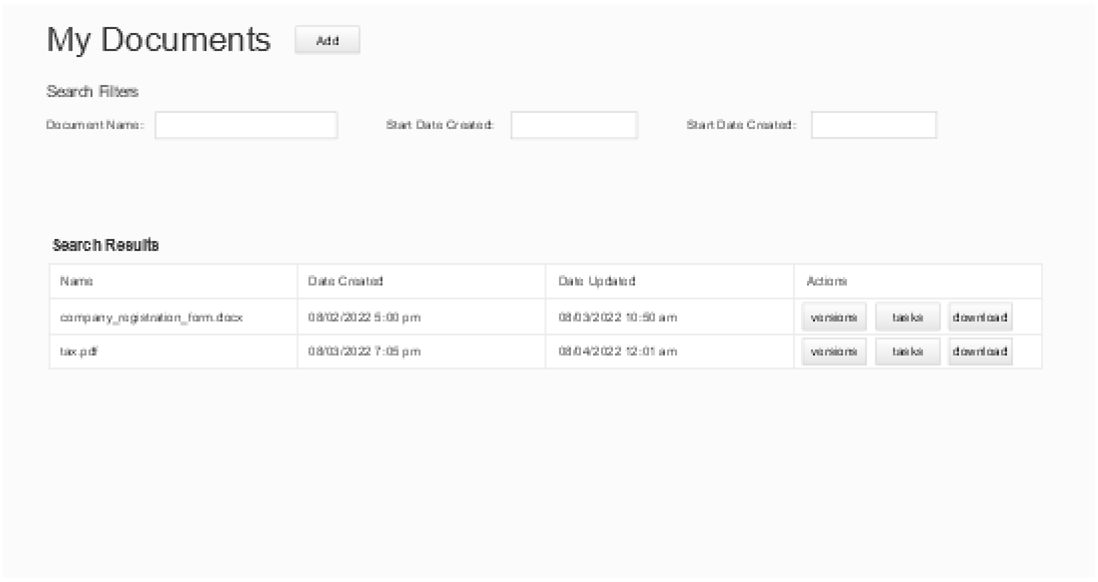
6.5.2.2. My Document page

The screen (my documents) below should be the landing Page after successful login. This screen should only show the documents created by the login user.

Sections:

- **Search Filters** – constraint: All queries must contain the user_id. If a value was not entered in any of the search fields, then that field should not be apart of the query.
  - Fields
    - **Document Name** – Should check against the friendly name column on the documents table for all documents that contain the value entered by the login user.
    - **Start Date Created –** Should check against the date created column on the documents table for all documents greater than or equal to the value entered by the login user.

        - The start date created must be before the end date created if both dates are entered
      - **End Date Created -** Should check against the date created column on the documents table for all documents less than or equal to the value entered by the login user.
        - The end date created must be after the start date created if both dates are entered
- **Search results**
  - Action buttons
    - Download – should retrieve the latest version of the document and download the same unto the user's computer using the friendly name of the document as the file name.
    - Versions – When clicked the user should be redirected to the document versions page for the associated document.
    - Tasks – When clicked the user should be redirected to the document tasks page for the associated document.


My Documents

Search Filters

Document Name:

Start Date Created:

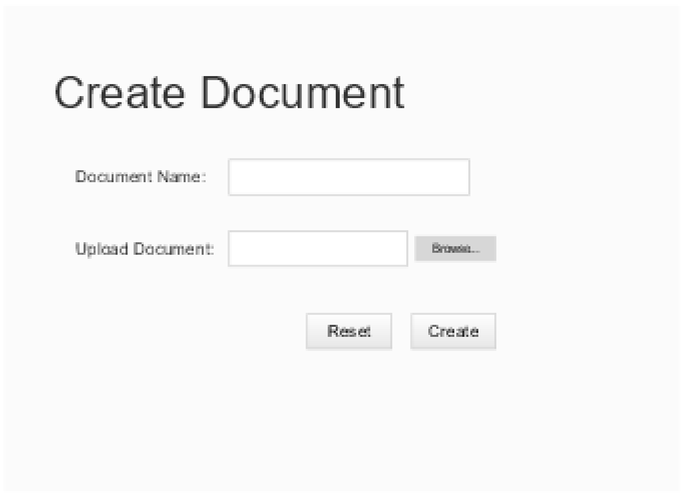
Start Date Created:

Search Results

Name	Date Created	Date Updated	Actions
company_registration_form.docx	08/02/2022 5:00 pm	08/03/2022 10:50 am	versions tasks download
tax.pdf	08/03/2022 7:05 pm	08/04/2022 12:01 am	versions tasks download

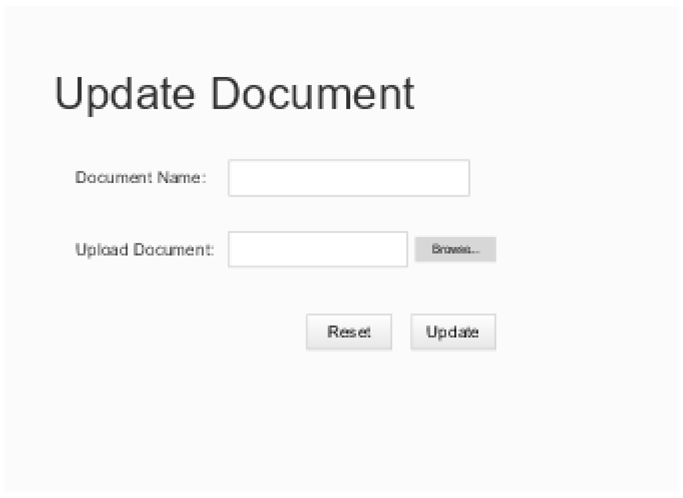
6.5.2.3. Create Document page

The below screen (create document) should be reachable from the landing page (my documents). The screen should allow the user to enter a friendly name for the document and select the document to be uploaded. The system should validate that the document being uploaded is either a word or pdf document. Both the document name and upload document fields are required. The system should perform validation on both the backend and the frontend. Upon validation being passed, then the system should base64 encrypt the document and send the same to the client document management system. The login user's unique identifier should also be sent to ensure that the document is place in a folder name using this identifier.



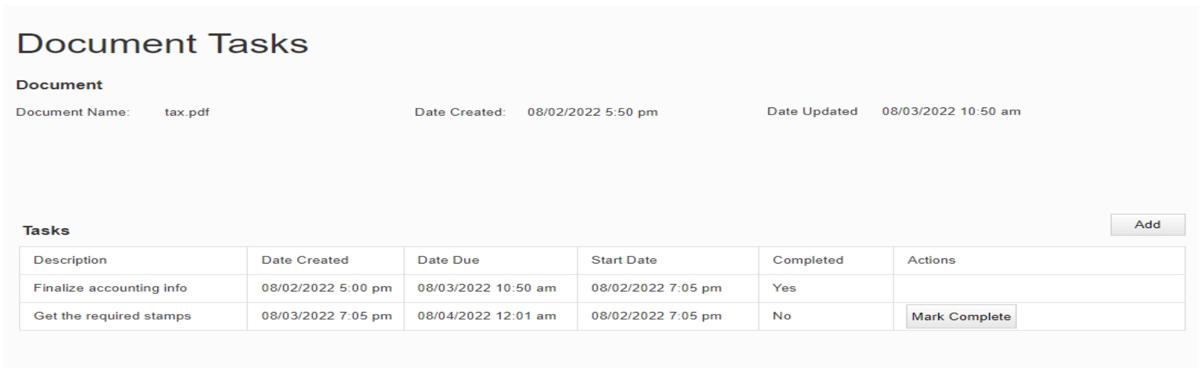
6.5.2.4. Upload Document page

The screen (upload document) below should allow a user to upload a new version of a file that the user uploaded previously. The document name field should be a select2 or similar field to allow the user to pick the document name from an existing list of documents created by the login user. The system should validate that both the document name and upload document fields, which are required, have a value when the login user clicks the update both. The system should validate that the filetype of the document being uploaded is either word or pdf.



6.5.2.5. Document Tasks page

The screen (document tasks) below should display all the tasks associated with the selected document. The system should validate that the document selected belongs to the login user. The system should not display the "mark complete" button if the "completed" column associated with the task is Yes (1). When the user clicks the mark complete button the system should validate that the tasks was not already completed. Once all validations are passed then the system should up the task to completed equal Yes (1).



Document Tasks

Document

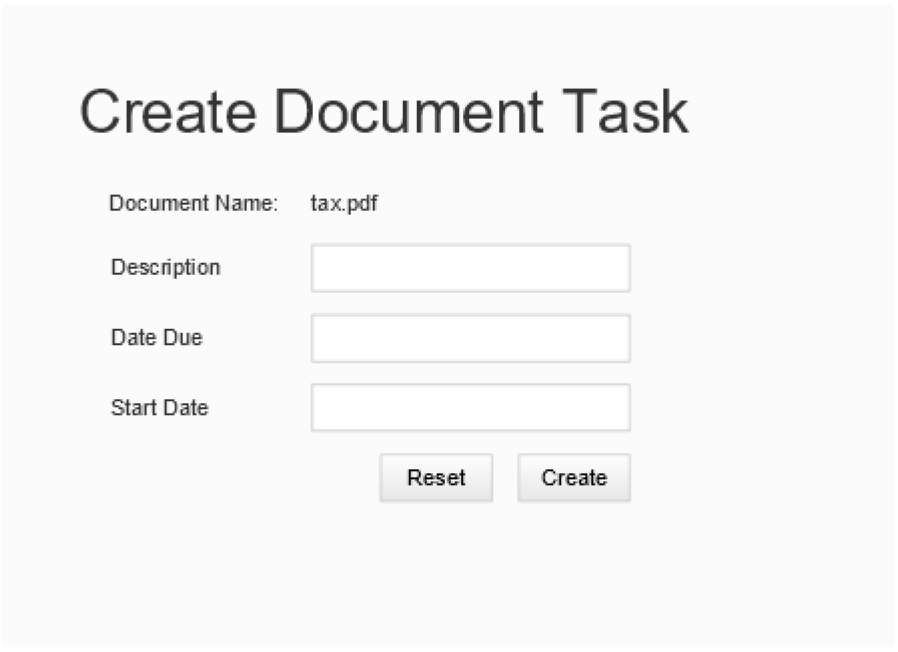
Document Name:	tax.pdf	Date Created:	08/02/2022 5:50 pm	Date Updated	08/03/2022 10:50 am

Tasks

Description	Date Created	Date Due	Start Date	Completed	Actions
Finalize accounting info	08/02/2022 5:00 pm	08/03/2022 10:50 am	08/02/2022 7:05 pm	Yes	
Get the required stamps	08/03/2022 7:05 pm	08/04/2022 12:01 am	08/02/2022 7:05 pm	No	Mark Complete

6.5.2.6. Create Document Task page

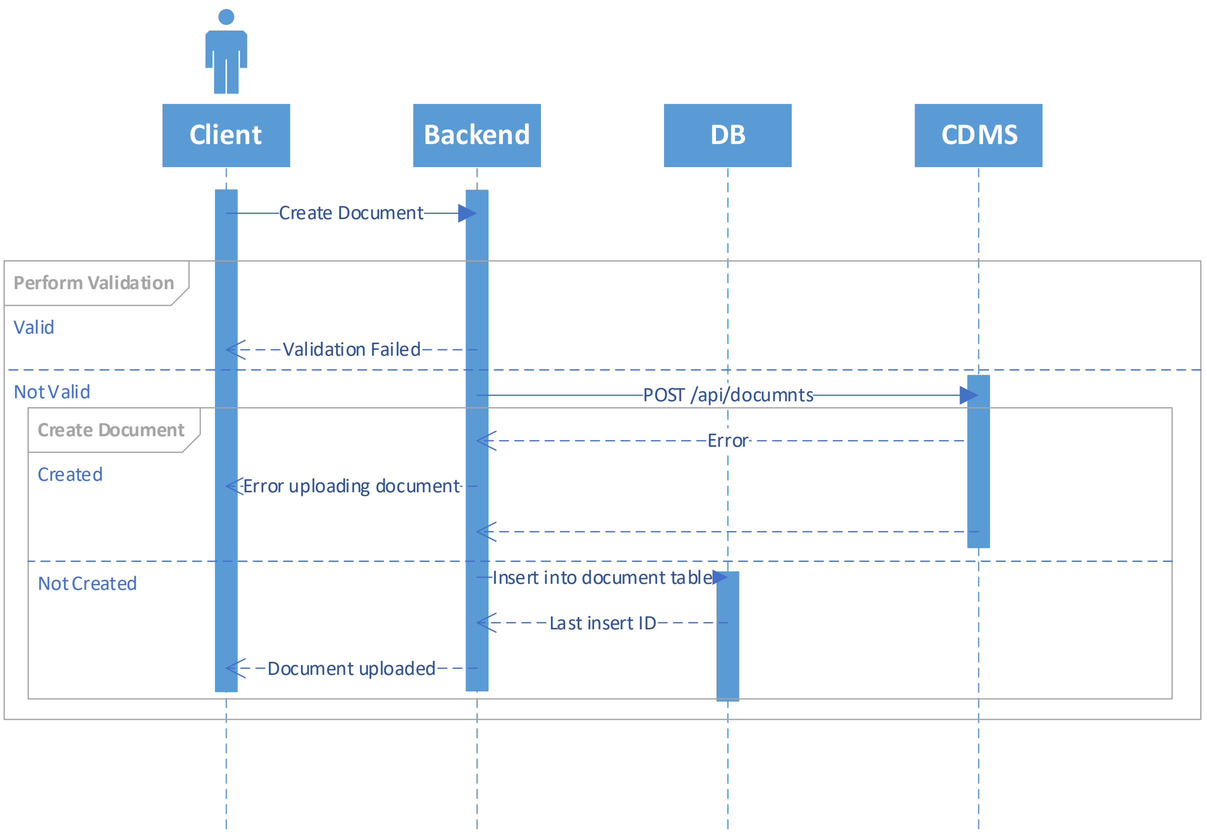
The below screen (create document task) should allow a login user to add a task to the selected if the document belongs to that user. The system should validate that a value was entered for all fields. The system should also validate that the value entered by the user for the fields: date due and start date is actual a date and that the start date is on or before the date due.



Create Document Task

Document Name:	tax.pdf
Description	
Date Due	
Start Date	

6.6. Sequence Diagram


Client
Backend
DB
CDMS
Create Document
Perform Validation
Valid
Validation Failed
Not Valid
POST /api/documnts
Create Document
Error
Created
Error uploading document
Not Created
Insert into document table
Last insert ID
Document uploaded