2021

“Clientio”

Development of a Client Management portal for small to medium scale law firms in Maldives

FINAL YEAR PROJECT | CYRYX COLLEGE

AHMED FAARIH(BIT-SD-D6)

“Clientio”

Abstract

Clientio a client management system designed for small to medium scale law firms in Maldives. This paper will focus on the development of a web application, including the front end and backend design techniques and tools used in the development of this system. This paper will also have a look at the testing done before the deployment of this system, and setup and user manual of this application. This project was designed so that law firms and lawyers operating in Maldives can have a better communication and management with their clients.

The project can be downloaded from:

https://github.com/faarih01/clientio

or

https://drive.google.com/file/d/1j0ssRwmqBRPqVyPOFvd4JPR6Z3-BAO2J/view?usp=sharing

Table of Contents

Chapter 1: Introduction to the Problem

1.1 Introduction

The judiciary sector of the Maldives is one of the outdated / underdeveloped sectors in Maldives. Even though the year is 2021, the justice system of Maldives is far behind in terms of technology and providing awareness to people on how their rights can be guaranteed. Normally it takes about 2 to 3 years for an individual / organization to finish a court case. This is where law firms come in.

A good business attorney will provide vital assistance in almost every aspect of your business, from basic zoning compliance and copyright and trademark advice to formal business incorporation and lawsuits and liability. (Ennico, 2021) Law firms have been the link between public / organization in this day to day running in between courts and judicial system. As lawyers get caught up within courts and day to day tasks, it often hard and difficult to give proper updates to their clients as it requires loads phone calls and mostly time.

This report is based on the development of a client management portal which will be used to give updates to clients regarding the cases. Via which clients will be able to get their case details, updates, request for documents, upload documents from user (client) and administrator (Law firm) side. This report also outlines the system requirements, system analysis, their level of contribution, system testing procedures.

1.2 Purpose

The purpose of this project is to help local law firms and the local communities, who are battling for their rights which has been illegally taken away from them. As mentioned, the legal sector of the Maldives is very far behind in terms of technology and many other resources compared to developed countries. So, the introduction of this system is mainly

targeted to help local start-up law firms to manage their clients and overall business documentations more efficiently and in a modern way.

1.3 Objective

- ✓ To increase and strengthen the relationship between lawyers and clients.
- ✓ To make day to day tasks easier and efficient for startup law firms
- ✓ Deliver cases information to clients.
- ✓ Manage clients via a proper system and reduce miscommunication
- ✓ To provide steady and constant updates of cases to clients

1.4 Existing Solution (Gap analysis current state)

Currently, there are very few or no solutions provided for managing clients specifically targeted for law firms and their clients in the Maldives. Currently well-established firms are using third party apps which are expensive and start-ups/medium scale firms would not be able to use this third-party apps as their subscriptions are not affordable. Hence it could be said that the are no existing solutions designed for start-ups and medium scale law firms in the Maldives. Main issues with the current state:

- Clients are unable to get proper updates of their cases on time.
- No quick way of requesting documents from clients
- No quick way of sending important document to lawyers.
- Unable to provide case details quickly
- Cases of fraud

1.5 Proposed Solution (Gap analysis desired state)

The introduction of such system would enable start-ups and medium scale businesses to properly manage their clients, cases and documents related to cases. This would enable law firms to get to their desired state which are as follows:

- Build a great relationship with clients and law firms.
- Increase overall value of firm and profitability
- Make a platform to share documents easily
- Provide regular updates of cases to firms
- Build a proper IT infrastructure for legal stake
- Easily to refer to previous cases and their documents if needed

Chapter 2: Software Requirement Specification (SRS)

Requirement Analysis, also known as Requirement Engineering, is the process of defining user expectations for a new software being built or modified. In software engineering, it is sometimes referred to loosely by names such as requirements gathering or requirements capturing. Requirement's analysis encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product or project, taking account of the possibly conflicting requirements of the various stakeholders, analysing, documenting, validating, and managing software or system requirements. (Paradigm, n.d.)

I will be looking and the functional, non-functional and usability requirements of this Clientio.

2.1 Functional Requirements

In software engineering, a functional requirement defines a system or its component. It describes the functions a software must perform. A function is nothing but inputs, its behavior, and outputs. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform.

As there are two main users of the application, I have decided to group the functional requirements in to two:

Admin/Law firms' side functional requirements.

- ✓ Authenticate Users
- ✓ Reset password upon requests
- ✓ Add & remove and Update users from Application
- ✓ CRUD of projects
- ✓ Assign users to projects
- ✓ Send updates to users according to their projects
- ✓ Request for documents, view user uploaded documents

Clients side functional requirements:

- ✓ Authenticate users
- ✓ Change password
- ✓ View client details
- ✓ View case details and case proceedings
- ✓ Able to upload the requested documents

2.2 Non-Functional Requirements

A non-functional requirement defines the quality attribute of a software system. They represent a set of standards used to judge the specific operation of a system. Example, how fast does the website load?

A non-functional requirement is essential to ensure the usability and effectiveness of the entire software system. Failing to meet non-functional requirements can result in systems that fail to satisfy user needs. (Martin, 2021)

Here are some of the non-functional requirements of the client management system:

- ✓ A client must only see the details / updates of his case no other.
- ✓ The website must be able to handle multiple clients together.
- ✓ A client must not be able to update his or other clients case details or updates.

2.3 Usability Requirements

I have tried to make my systems usability requirements in line with the Jakob Nielsen's 10 general principles for interaction design. They are called "heuristics" because they are broad rules of thumb and not specific usability guidelines. (Nielsen, 1994)

1. Visibility of system status

The design should always keep users informed about what is going on, through appropriate feedback within a reasonable amount of time. For example, in my application when a user is registered to the system an email will be sent to her email entered, notifying that she has been added to the system.

2. Match between system and the real world

The design should speak the users' language. Use words, phrases, and concepts familiar to the user, rather than internal jargon. Follow real-world conventions, making information appear in a natural and logical order.

3. User control and freedom

Users often perform actions by mistake. They need a clearly marked "emergency exit" to leave the unwanted action without having to go through an extended process. For example, users can always click the sidebar or back button of the browser to abandon a task or to move to a new task.

4. Consistency and standards

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform and industry conventions. For example, in my application all delete buttons will be in red color. By seeing a red button user will know that it’s a delete button.

5. Error prevention

Good error messages are important, but the best designs carefully prevent problems from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action. For example, in my application, if a user has cases the delete button of that user will be disabled from html. This is to prevent errors from database.

6. Recognition rather than recall

Minimize the user's memory load by making elements, actions, and options visible. The user should not have to remember information from one part of the interface to another. Information required to use the design (e.g., field labels or menu items) should be visible or easily retrievable when needed.

7. Flexibility and efficiency of use

Shortcuts — hidden from novice users — may speed up the interaction for the expert user such that the design can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

8. Aesthetic and minimalist design

Interfaces should not contain information which is irrelevant or rarely needed. Every extra unit of information in an interface competes with the relevant units of information and diminishes their relative visibility. I have tried to keep the design very simple and the application very minimal and only to point.

9. Help users recognize, diagnose, and recover from errors.

Error messages should be expressed in plain language (no error codes), precisely indicate the problem, and constructively suggest a solution. I have planned to use sweet alert JavaScript library to notify users upon errors.

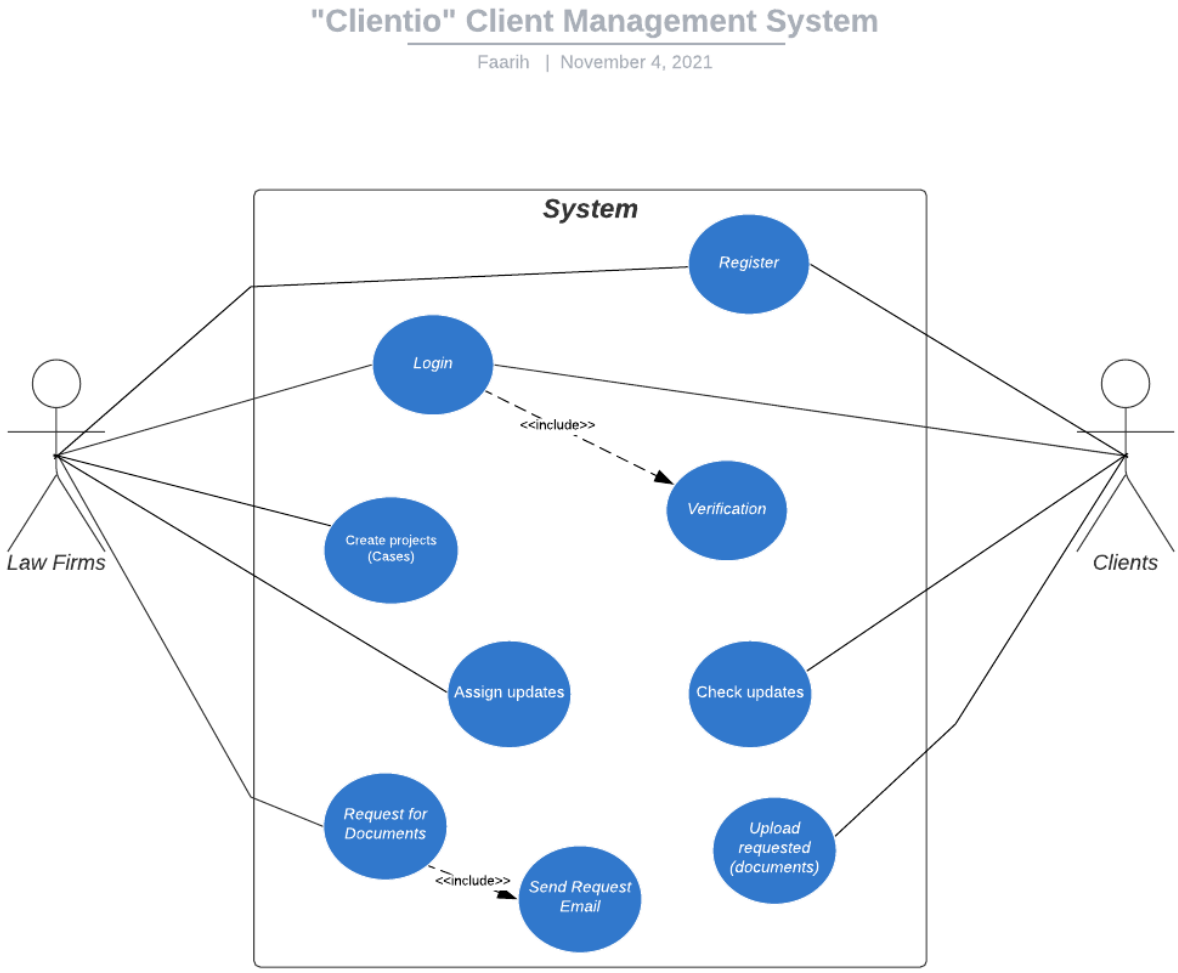
10. Help and documentation.

It’s best if the system doesn’t need any additional explanation. However, it may be necessary to provide documentation to help users understand how to complete their tasks. (Nielsen, 1994)

Chapter 3: Use Case Analysis

Use cases helps us focus on what is essential and ultimately create a system that dose something useful. The description of what system dose is principally captured as text; the use-case diagram serves as an overview or summary of the systems behaviour. (Kurt Bittner, 2003) The flowing will be a use case analysis with diagram and written description of the main use cases of the system.

The following is the use case diagram for “cinentio” with the main functions of the system explained in detail



Description of each use case.

Use case:	**Register**
Use case description:	This use case allows user to login into the system to access the relevant functions according to the user’s role. There are two main roles (Admin role & Client Role). The user can reset password with the used email of this use case.
Primary Actor:	User / Law firms (User)
Secondary Actor:	None
Include use case	-
Preconditions:	-
Postconditions:	The system displays the relevant dashboard.
Main flows:	1. The user enters his name, email, password (twice for conformation). 2. The user submits the entered information. 3. The system validates the form information. 4. The system registers the information into the database. 5. The system displays users home dashboard. 6. The use case ends.
Alternative flows:	**3A Missing information in the form.** 1. The system asks the user for the missing filed 2. Use case resumes at main flow step 01

	3B Invalid email address 1. The system displays that the user should include "@" in their email 2. Use case resumes at main flow step 01 **3B Password entered is less than 8 characters** 1. The system displays that the user should enter "The password must be at least 8 characters. 2. The case resumes at main flow step 01

Use case:	**login**
Use case description:	This use case allows users to register to the system. The user should be able to register with name, email and a password. The users must enter a valid email address as this email will be used to reset password and many other activities. Upon successful registration the user will be taken to the dashboard page.
Primary Actor:	User / Law firms (User)
Secondary Actor:	None
Include use cases	Verification of users
Preconditions:	The user has a valid email and password
Postconditions:	The system displays the relevant dashboard.
Main flows:	1. The user enters his email, password 2. The user submits the entered information 3. The system validates the form information 4. The system triggers verify user Use case 5. The system displays users home dashboard 6. The use case ends
Alternative flows:	**3A Missing information in the form (Email/password)** 1. The system asks the user for the missing filed 2. Use case resumes at main flow step 01 **3B Invalid email address /Password** 1. The user is asked to enter a valid email & password 2. Use case resumes at main flow step 01

Use case:	**Create projects / Cases**
Use case description:	This use case allows law firms to create a case that they have taken to represent. After providing the case details the case is assigned to a user.
Primary Actor:	Law firms (User)
Secondary Actor:	None
Include use cases	-
Preconditions:	The user logged with the login use case as ADMIN and is in the projects-Create page
Postconditions:	The system displays the Projects (Index) page.
Main flows:	1. The user enters project title, chooses project type from "drop-down" of types and description. 2. The user chooses the Projects/Cases owner that is the client of the case from drop-down of clients. 3. The system validates the form information. 4. The system displays the projects index page, with the newly created project. 5. The use case ends
Alternative flows:	**3A Missing information in the form (Required fields such as title, details...)** 1. The system asks the user for the missing filed 2. Use case resumes at main flow step 01

	3B The user dose not assign the case to a client 1. The user is asked to choose a client 2. Use case resumes at main flow step 01

Use case:	**Create cases updates**
Use case description:	This use case enables law firms to update cases as they proceed. The law firm will provide information regarding to how a case is going on as time passes by. After creating the updates, the law firm would also see how the cases/project has proceeded from the projects (show) page.
Primary Actor:	Law firms (User)
Secondary Actor:	Clients (Law firm perspective)
Include use cases	-
Preconditions:	The user logged with the login use case as ADMIN and is in projects updates create page
Postconditions:	The system displays the Projects index page.
Main flows:	1. The user enters update date, description of the update, and assigns that update to the relative case from the drop down. 2. The system validates the form information such as date. 3. The system displays the projects index page with a success alert. 4. The use case ends
Alternative flows:	2**A Missing information in the form (Update: date, description)** 1. The system asks the user for the missing filed 2. Use case resumes at main flow step 01

Use case:	**Check case updates**
Use case description:	This use case enables clients to check and view the update of their respective cases.
Primary Actor:	Clients
Secondary Actor:	Law firms
Include use cases	-
Preconditions:	The user logged with the login use case as GO USER
Postconditions:	-
Main flows:	1. The user uses selects the "Case updates" page from the sidebar. 2. The system shows the user his cases updates with their respective dates and descriptions. 3. The use case ends
Alternative flows:	-

Use case:	**Create document requests.**
Use case description:	This use case enables law firms to request clients to upload needed documents related to cases.
Primary Actor:	Law firms
Secondary Actor:	-
Include use cases	Send request emails
Preconditions:	The user logged with the login use case as ADMIN and is in the document request create page.

Postconditions:	Update the client's dashboard notifications icon and notifications.
Main flows:	1. The user enters the details of the document he wants and selects the client he wants to send this request to. 2. Triggers "send request email" and sends an email to requested user's email. 3. The system shows a success alert saying that the request has been sent to the selected user. 4. The use case ends
Alternative flows:	**1A Missing information (details of request)** 1. The system asks the user for the missing filed 2. Use case resumes at main flow step 01

Use case:	**Upload requested documents**
Use case description:	This use case enables the users to upload documents requested by the law firms
Primary Actor:	Clients
Secondary Actor:	-
Include use cases	-
Preconditions:	The user logged with the login use case as GO USER and is in his requests page.
Postconditions:	The requests index page is updated with the newly uploaded file
Main flows:	1. The user selects the document he wants to upload from his PC /Mobile 2. The system validates the uploaded file type. 3. The system shows a success alert thanking the client for his time. 4. The use case ends
Alternative flows:	2**A Missing information / wrong file type** 1. The system asks the user for the missing filed or the user uploads the file with the correct type 2. Use case resumes at main flow step 01

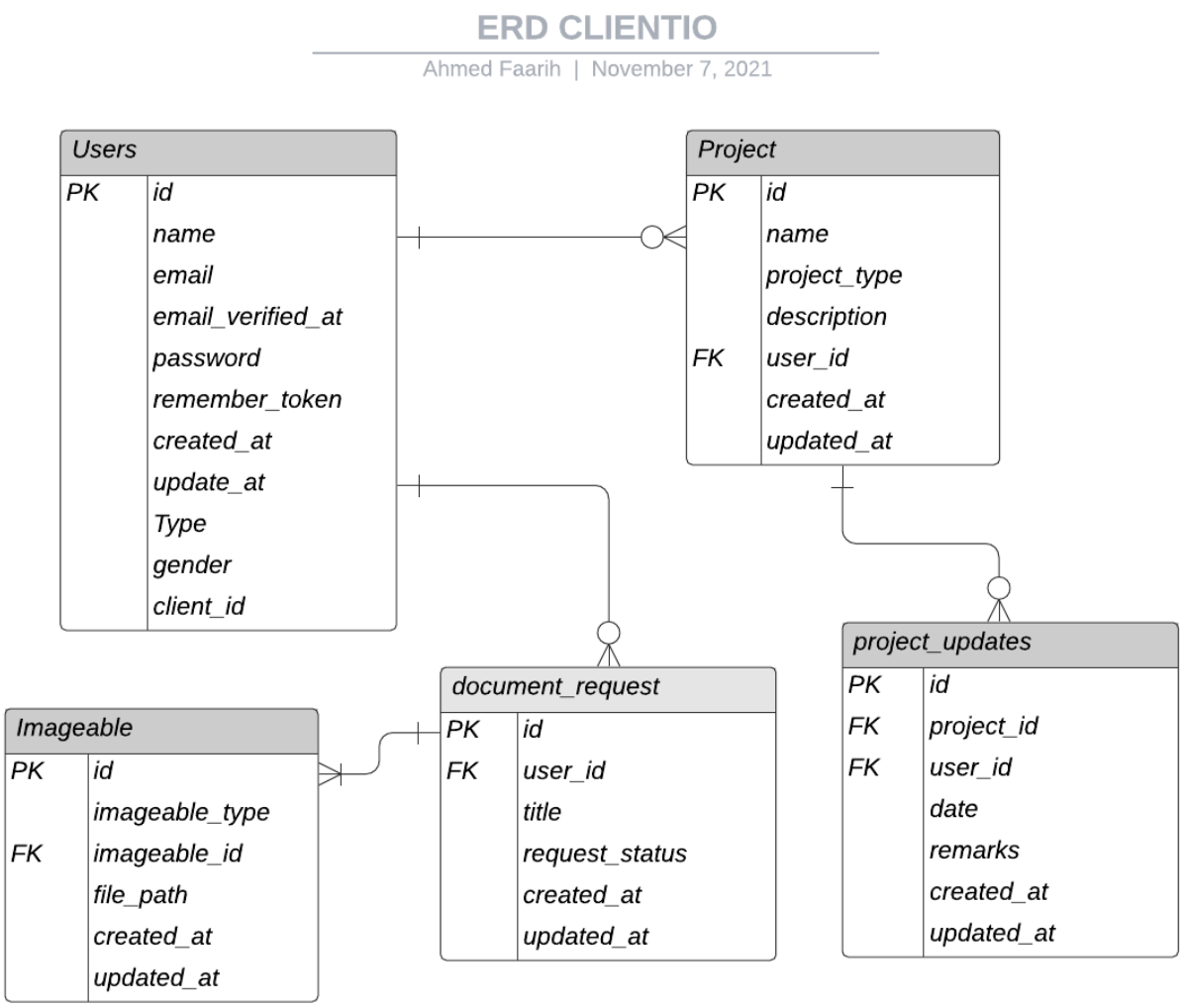
Chapter 4: Design

In this chapter I will be looking at the database design concepts and the overview architecture of my system. I will be describing the details of the, ERD, Class diagram, Activity Diagram,

4.1. ERD with data dictionary

Database system is the base of a management information system. Relational databases are widely used in various areas. Entity-Relationship Diagram (ERD) is a common technique for data structures and database systems design. (Chen, 2009) I have designed the ERD of the system in my initial project proposal, but a few fields have been added to the ERD due to some requirements of the system.

The following is the ERD of client management system along with data directory and a brief explanation of the relationships between entities. The following ERD has been designed to support Laravel 8 so that the development could proceed in a speedy way.



While looking at the above ERD the Users can have zero or many projects(cases). But a project should belong to one and only one user(client). A project can have zero or many updates, but an update should belong to a project. A user (client) can have many document requests. But each request should belong to one user. A document request should have one or multiple imageables (files), but a file should belong to one request. The following is a detailed data directory of the above ERD diagram.

Entity Name: Users

Entity description: All types of users that are using the client management system.

Colum Name	Description	Data type	Length	PK	Nullable	Unique
id	The unique id of each user.	bigInt	20	**True**	False	True
name	Name of the user using the system	varchar	255	False	False	False
email	Email of the user using the system	varchar	255	False	False	False

email_verified_at	Time stamp of the time users' email was verified	timestamp	-	-	True	-
password	The password of the user using the system	varchar	255	False	False	False
Remember_token	The token used by the user to make the system remember him.	varchar	100	False	False	
created_at	The time the entity as created at	timestamp	-	-	True	False
updated_at	The time the entity was last updated at.	timestamp	-	-	True	False
type	The user type ie a client or an administrator	enum	-	False	False	False
gender	The gender of the user	enum	-	False	False	False
Client_id	The client number or identifier for the client	varchar	255	False	True	Ture

Entity Name: Projects (cases)

Entity description: Types and description of a case that the firm has taken over.

Colum Name	Description	Data type	Length	PK	FK	Nullable	Unique
id	The unique id of each case.	bigInt	20	**True**	-	False	True
User_id	Id of the client that case belongs to	varchar	255	False	**True**	False	False
title	Title of the case	varchar	255	False		False	False
type	The type of case that is being taken	Enum	-	-	-	Fase	-
description	The password of the user using the system	longtext	-	-	-	False	False
created_at	The time the entity as created at	timestamp	-	-		True	False
updated_at	The time the entity was last updated at.	timestamp	-	-		True	False

Entity Name: Projects updates

Entity description: The updates of the cases that are being given to clients

Colum Name	Description	Data type	Length	PK	FK	Nullable	Unique
id	The ID of each update that is being given	bigInt	20	**Ture**	False	False	True
project_id	The relevant project that the update is being given to.	bigInt	20	False	**Ture**	False	False
User_id	The id of the client of the project	bigInt	20	False	True	False	False
Date	The date of the update	date	-	False	False	False	False
remarks	The remarks or description of the update	longtext					
created_at	The time the entity as created at	timestamp	-	-		True	False
updated_at	The time the entity was last updated at.	timestamp	-	-		True	False

Entity Name: Document requests

Entity description: Document requests are request sent to the clients via the portal by Law firm requesting certain information / documents.

Colum Name	Description	Data type	Length	PK	FK	Nullable	Unique
id	The ID of each request sent by the lawfirm	bigInt	20	**Ture**	False	False	True
User_id	The client ID of the user that the request was sent to.	bigInt	20	False	**True**	False	False
Title	The title of the request asking for the document.	varchar	255	False			
Request_status	This trigger checks if a request is complete or not.	tinyInt	1	False	False	True	False

created_at	The time the entity as created at	timestamp	-	-		True	False
updated_at	The time the entity was last updated at.	timestamp	-	-		True	False

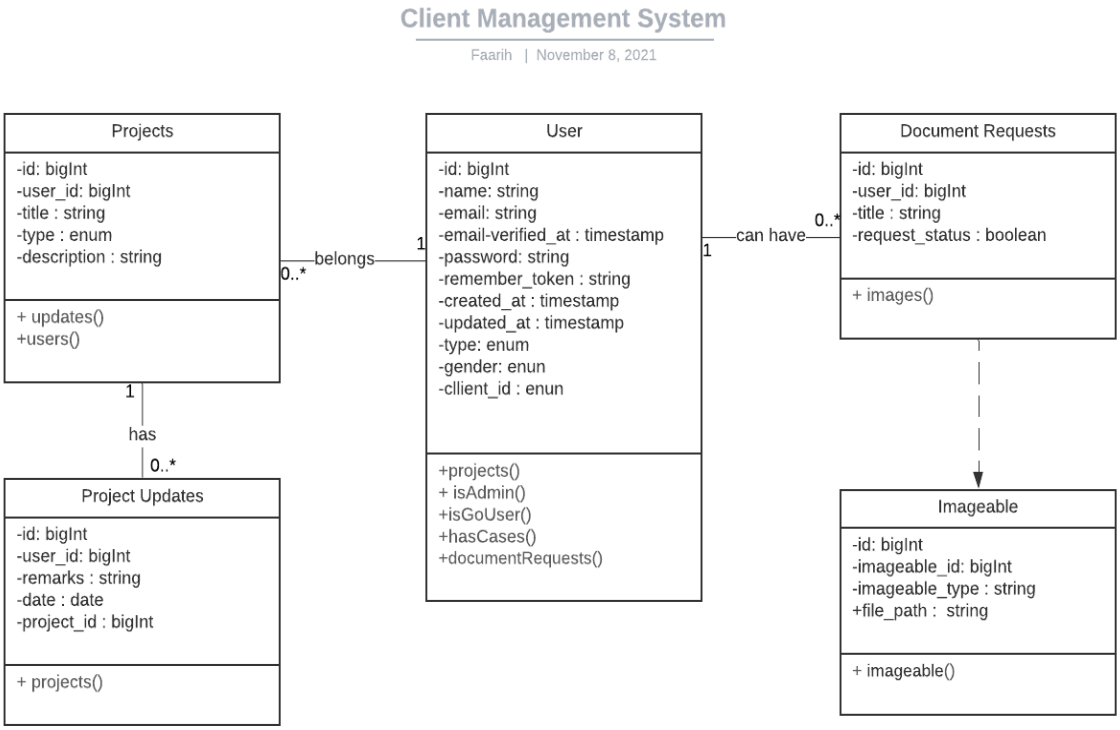
Entity Name: Document requests

Entity description: Document requests are request sent to the clients via the portal by Law firm requesting certain information / documents. Please look at (https://laravel.com/docs/8.x/eloquent-relationships) for a better understanding of this entity.

Colum Name	Description	Data type	Length	PK	FK	Nullable	Unique
id	The ID of each imageable(file) uploaded by clients	bigInt	20	**Ture**	False	False	True
Imageable_type	The model that the file is related to.	varchar	255	False	**True**	False	False
Imageable_id	The id of the entity that the (file) belongs to.	bigInt	20	False	**True**	False	False
File_path	The file name or location it is uploaded to.	varchar	255	False	False	True	False
created_at	The time the entity as created at	timestamp	-	-		True	False
updated_at	The time the entity was last updated at.	timestamp	-	-		True	False

4.2. Class Diagram

One of the most important components of UML are class diagrams, which model the information on the domain of interest in terms of objects organized in classes and relationships between them. (Daniela Berardi, 2005). The following is the class diagram of “clientio”: draw in an **implementation perspective** relating to **Laravel 8** framework.



While looking at the client management system all classes except document request has associate relationships ealier explained in the ERD. However, the class document request could be said as a dependency relation with Imageable class.

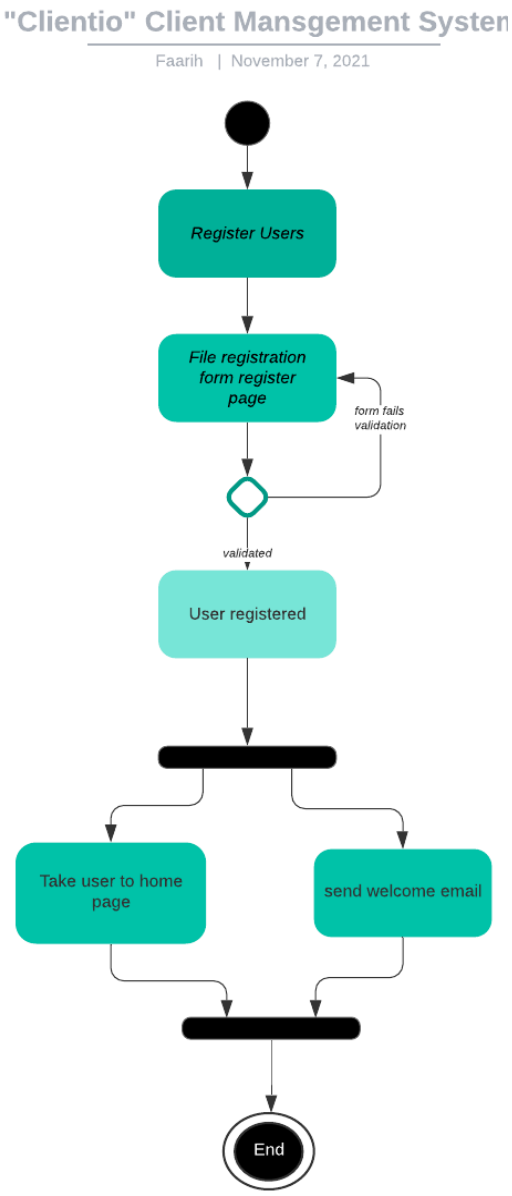
4.3 Activity Diagram

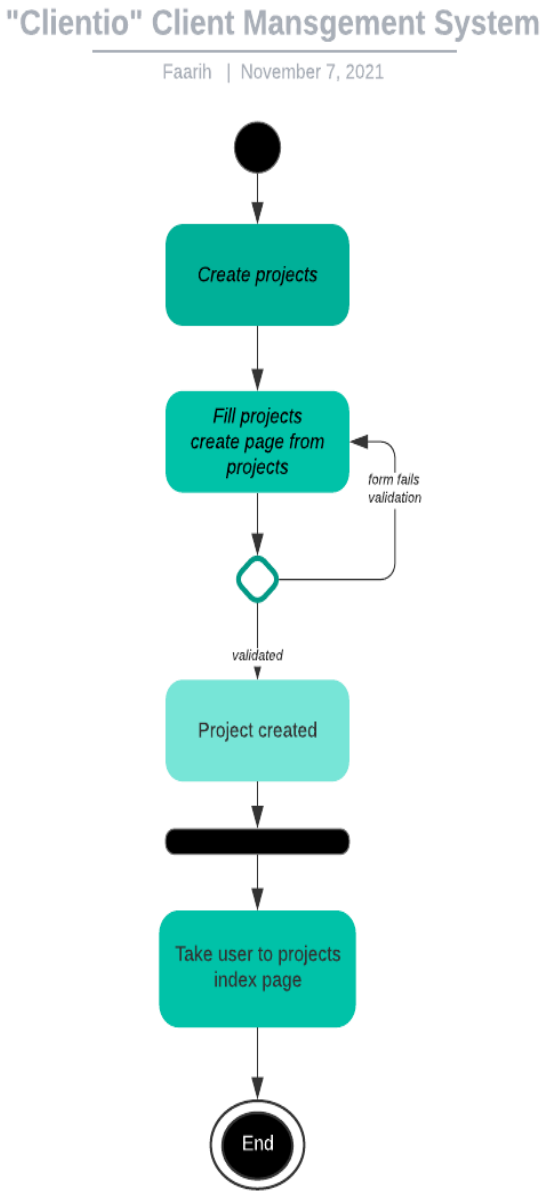
An activity diagram is a behavioural diagram in UML to describe how a user will be interacting with an application. Activity Diagrams describe how activities are coordinated to provide a service which can be at different levels of abstraction. Typically, an event needs to be achieved by some operations, particularly where the operation is intended to achieve several different things that require coordination, or how the events in a single use case relate to one another. (www.visual-paradigm.com, n.d.).

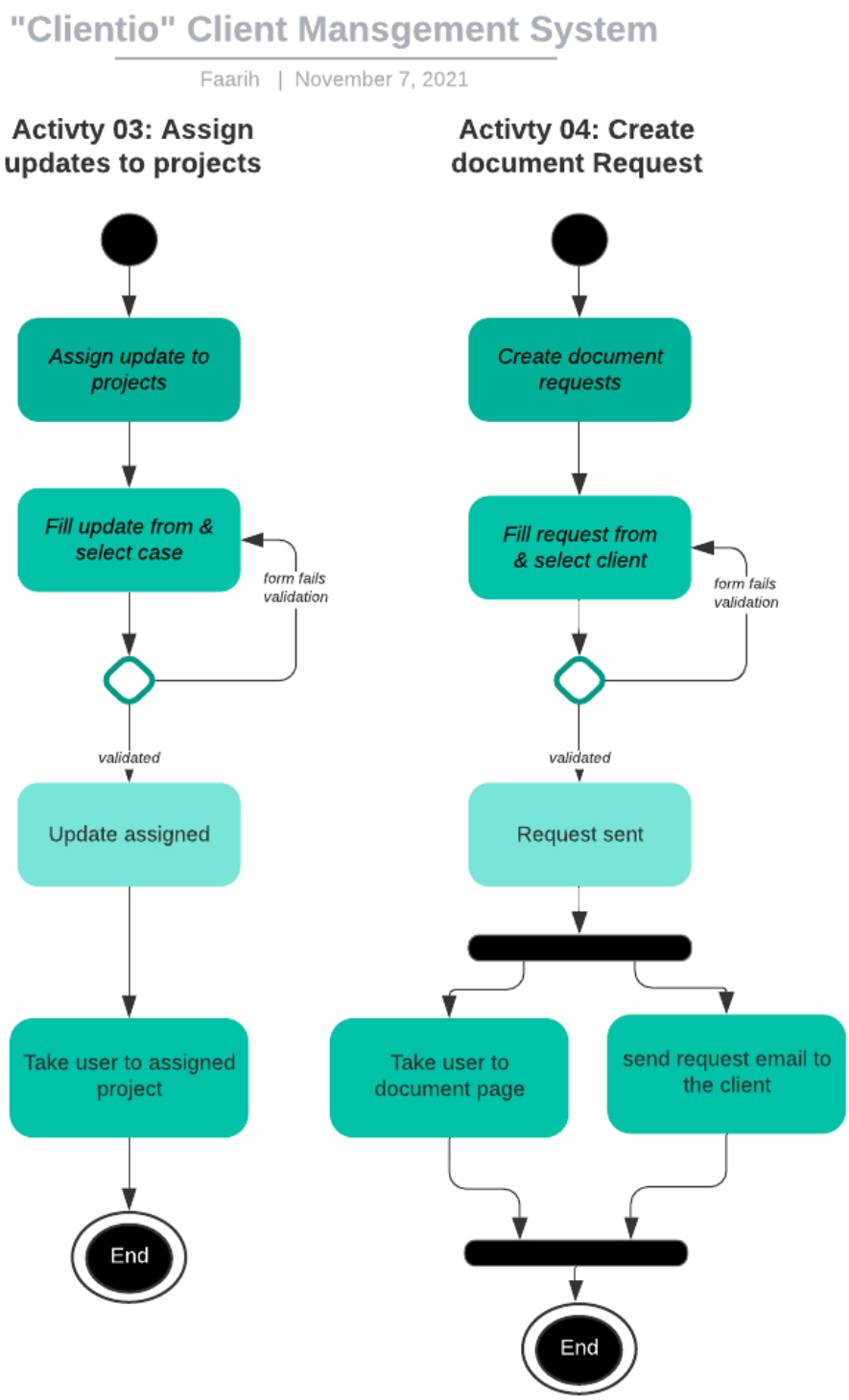
In this section we are going to look at the activity diagram of some of the important features of client Management system. Please note that this activity diagram is created assuming the user is already logged in as their respective role.

Activity 01: Registering Users

Activity 02: Creating projects


"Clientio" Client Mansgement Systen
Faarih | November 7, 2021
Register Users
File registration form register page
form fails validation
validated
User registered
Take user to home page
send welcome email
End


"Clientio" Client Mansgement System
Faarih | November 7, 2021
Create projects
Fill projects create page from projects
form fails validation
validated
Project created
Take user to projects index page
End



There is other process inside the application but the above are some one of the important functions of the application detailed by activity diagram.

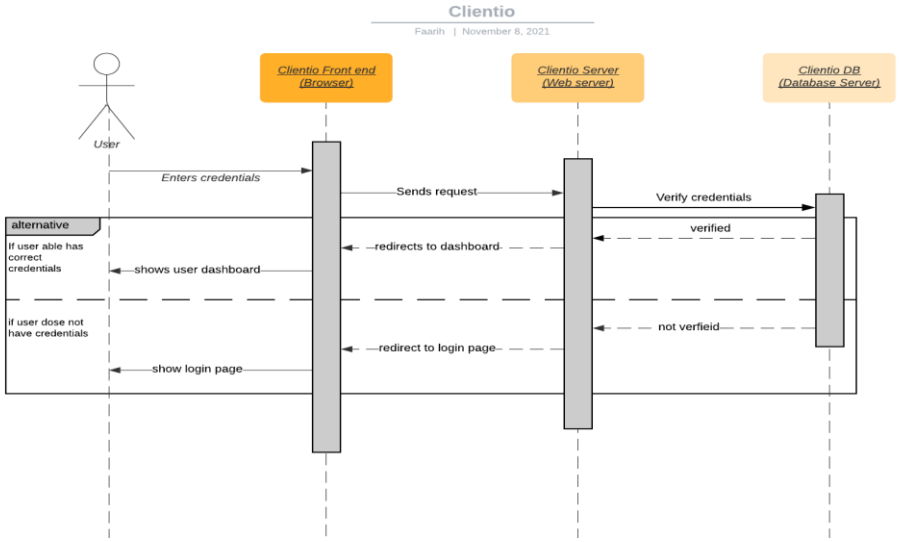
4.3. Sequence Diagram

A sequence diagram is a type of interaction diagram because it describes how—and in what order—a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. (www.lucidchart.com/, n.d.)

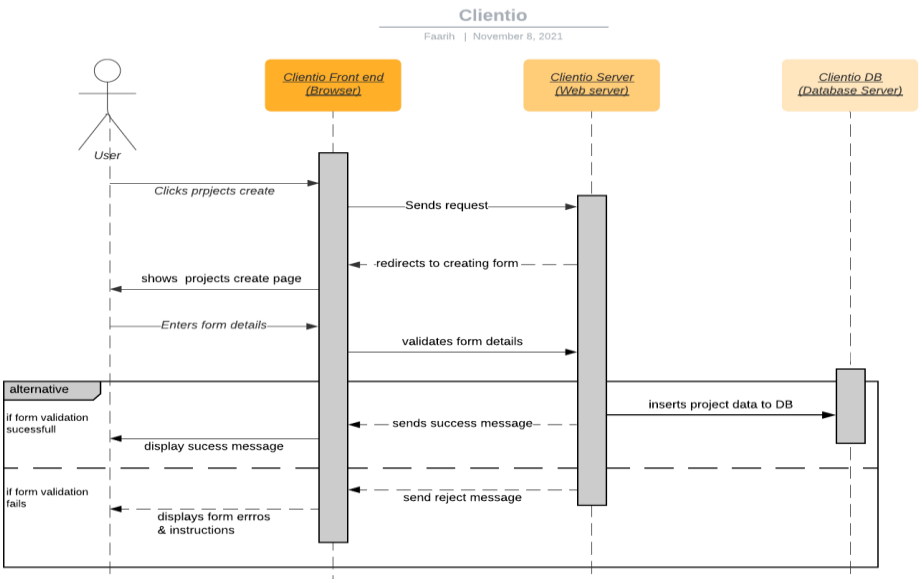
The following will be sequence of diagrams of the important process of the client management system.

Note: Similar activities with repetitions have been ignored to minimize.

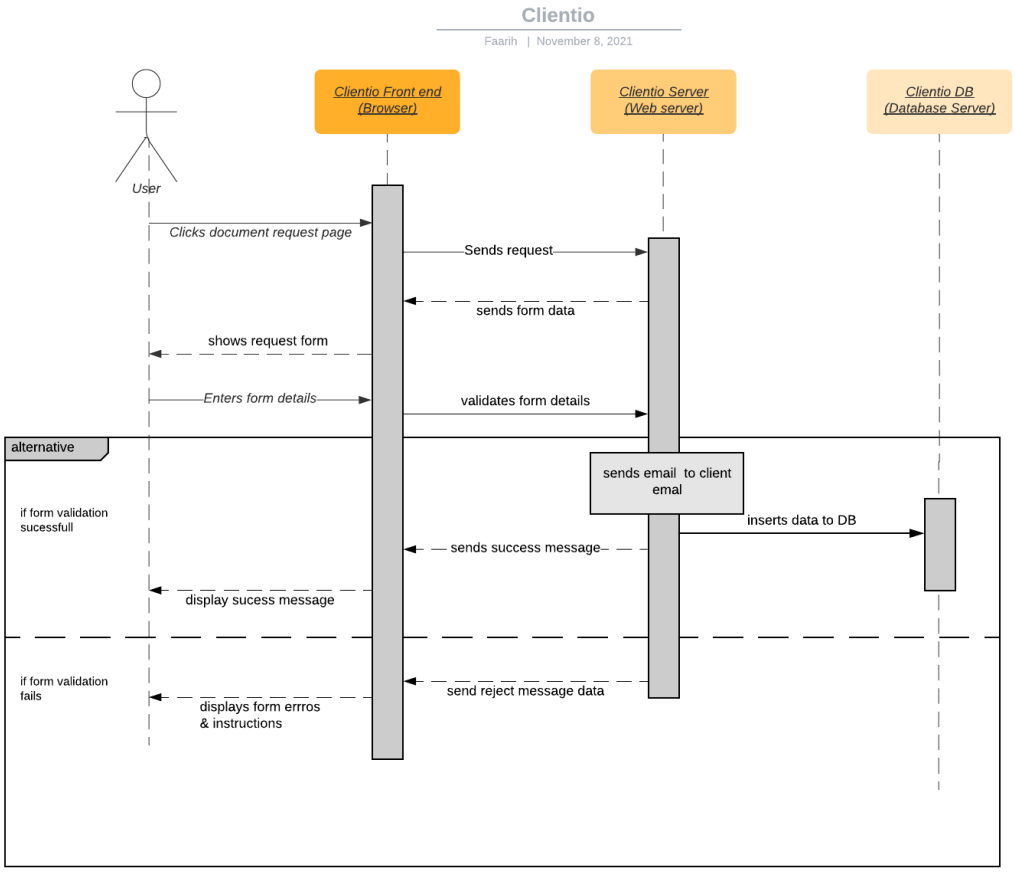
01. Logging in to the application (This step will be skipped here after assuming user is already logged)



02. Creating a project (case)



03. Sending request for document submission



Chapter 5: Testing

Software testing is a process, to evaluate the functionality of a software application with an intent to find whether the developed software met the specified requirements or not and to identify the defects to ensure that the product is defect-free in order to produce a quality product. (Rajkumar, 2021) There are various types of testing involved when it comes to software testing and in this section I am going to explaining the testing of “Clientio” with these techniques and tools.

5.1. Test Case Specifications

A TEST CASE is a set of actions executed to verify a particular feature or functionality of a software application. A Test Case contains test steps, test data, precondition, postcondition developed for specific test scenario to verify any requirement. The test case includes specific variables or conditions, using which a testing engineer can compare expected and actual results to determine whether a software product is functioning as per the requirements of the customer. (Hamilton, test-case, 2021)

Two basic approaches to software testing are black box testing and white box testing. White box testing based on an analysis of internal working and structure of a piece of software. It only checks how the system processes the input to generate required output. On the other hand, black box testing focuses on the functional requirement of the software. (Khan, 2011) The following are some of the test cases I have written to test "Clientio" before deployment. The following is going to be the test case template I will be using in to write test cases.

Positive Test Cases		
ID		
Priority		
Description		
Reference		
Users		
Pre-requisites		
Steps		
Inputs		
Expected results		
Status		

5.2. Black Box Test Cases

Black box testing is also known as behaviour testing or eye to eye testing which checks the functionality of a software/application without going deeper into the internal design/ architecture of the system. The following are some of the important black box test cases written for the testing of this system.

01. Test case for registration

Positive Test Cases		
ID	TCP-01	
Priority	High	
Description	To verify user is able to register to the portal	
Reference	Functional Requirement reference	
Users	Type: Go user (Clients)	
Pre-requisites	A	System is online.
	B	User is in the register page
Steps	A	Enter Name
	B	Enter email adress
	C	Enter password
	D	Enter confirmation of password
	E	Press register button

Inputs		Name, Email, Password
Expected results		Successfully takes user to dashboard page. Sends registration email to user email.
Status		Tested, Passed.
Negative Test Cases		
ID	TCN-01	
Priority	High	
Description	To verify invalid users are able to register	
Reference	Functional Requirement reference	
Users	Type: Go user (Clients)	
Pre-requisites	A	System is online.
	B	User is in the register page
Steps	A	Enter Name
	B	Enter email adress
	C	Enter password
	D	Enter confirmation of password
	E	Press register button
Inputs		Name, Email without @ sign, Password without 8 characters
Expected results		Dose not enter the records to DB and displays error message
Status		Tested, Passed.

02. Test case for Logging in to the system.

Positive Test Cases		
ID	TCP-02	
Priority	High	
Description	To verify user authentication to the web application	
Reference	Functional Requirement reference	
Users	Type: Go user (Clients)	
Pre-requisites	A	System is online.
	B	User is in the login page
Steps	A	Enter email
	B	Enter password
	C	Press login button
Inputs		Email, Password
Expected results		Successfully takes user to dashboard page.
Status		Tested, Passed.

Negative Test Cases	
ID	TCN-02
Priority	High
Description	To verify invalid users are able to register
Reference	Functional Requirement reference

Users	Type: Go user (Clients)	
Pre-requisites	A	System is online.
	B	User is in the login page
Steps	A	Enter email
	B	Enter password
	C	Press login button
Inputs		Incorrect email / password / both / empty
Expected results		Dose not enter the records to DB and displays error message
Status		Tested, Passed.

03. Creating a project & Assigning a User

Positive Test Cases		
ID	TCP-03	
Priority	High	
Description	To verify weather lawfirms are able to create projects (Cases)	
Reference	Functional Requirement reference	
Users	Type: ADMIN (Law firms)	
Pre-requisites	A	System is online.
	B	User is logged in as Admin
	C	User is in projects create Page.
Steps	A	Enter project title
	B	Select project type from drop down
	C	Enter description of project
	D	Select client from drop down of clients
	E	Press submit button
Inputs		Tittle, Type, Description, Client
Expected results		Shows sweet alert message with success dialog box & Enters details to DB
Status		Tested, Passed.

Negative Test Cases		
ID	TCN-03	
Priority	High	
Description	To verify weather lawfirms are able to create projects (Cases) without proper content	
Reference	Functional Requirement reference	
Users	Type: ADMIN (Law firms)	
Pre-requisites	A	System is online.
	B	User is logged in as Admin
		User is in projects create Page.
Steps	A	Leaves project title blank / Or fills it
	B	Select project type from drop down
	C	Leaves description blank / Fills it

	D	Select client from drop down of clients
	E	Press submit button
Inputs		Empty title / description or both or empty form
Expected results		Dose not enter the records to DB and displays error with missing fields
Status		Tested, Passed.

04. Editing a created project

Positive Test Cases		
ID	TCP-04	
Priority	High	
Description	To verify weather users are able to edit details of existing projects	
Reference	Functional Requirement reference	
Users	Type: ADMIN (Lawfirms)	
Pre-requisites	A	System is online.
	B	Logged in as Admin
	C	Is in projects Index Page
Steps	A	Press edit button of the project from index page
	B	Edit title / description / type / client
	C	Press submit buttton
Inputs		Edit title or description or type or client or empty
Expected results		Old form data when project edit page loads, Sweet alert message saying project updated. Data in database updated.
Status		Tested, Passed.

Negetive Test Cases		
ID	TCN-04	
Priority	High	
Description	To make sure that users are unable to edit and submit with missing fields of the project	
Reference	Functional Requirement reference	
Users	Type: ADMIN (Lawfirms)	
Pre-requisites	A	System is online.
	B	Logged in as Admin
	C	Is in projects Index Page
Steps	A	Press edit button of the project from index page
	B	Edit title / description / type / client = > remove
	C	Press submit buttton
Inputs		Edit title or description or type or client or empty with one or may fields missing
Expected results		Displays error message asking the user to include the missing fields of the project.
Status		Tested, Passed.

05. Creating project updates

Positive Test Cases		
ID	TCP-05	
Priority	High	
Description	To verify weather lawfirms are able to create updates for cases	
Reference	Functional Requirement reference	
Users	Type: ADMIN (Lawfirms)	
Pre-requisites	A	System is online.
	B	Logged in as Admin
	C	Is in projects show page Page
Steps	A	Press create button under updates
	B	Enter update date
	C	Enter remarks of the update
	D	Select Case the update belongs to from drop down.
Inputs		Date, Remarks, Case
Expected results		Shows sweet alert message saying the update was created
Status		Tested, Passed

Negetive Test Cases		
ID	TCN-05	
Priority	High	
Description	To verify the form validations of update form	
Reference	Functional Requirement reference	
Users	Type: ADMIN (Lawfirms)	
Pre-requisites	A	System is online.
	B	Logged in as Admin
	C	Is in projects show page
Steps	A	Press create button under updates
	B	Enter update date
	C	Enter remarks of the update
	D	Select Case the update belongs to from drop down.
Inputs		Date, Remarks or both missing.
Expected results		Displays message asking for the missing fields.
Status		Tested, Passed

06. Editing Projects update

Positive Test Cases	
ID	TCP-06

Priority	High	
Description	To verify weather entered updates of project are editable	
Reference	Functional Requirement reference	
Users	Type: ADMIN (Lawfirms)	
Pre-requisites	A	System is online.
	B	Logged in as Admin
	C	Is in projects show page
Steps	A	Press edit button of the update that needs to be edited
	B	Enter new update date
	C	Enter new remarks of the update
	D	Select Case the update belongs to from drop down.
Inputs		Date, Remarks or both with new values or with one new value.
Expected results		Shows sweet alert message saying the update was edited successfully
Status		Tested, Passed

Negetive Test Cases		
ID	TCN-06	
Priority	High	
Description	To verify that users are unable to send update data with missing fields	
Reference	Functional Requirement reference	
Users	Type: ADMIN (Lawfirms)	
Pre-requisites	A	System is online.
	B	Logged in as Admin
	C	Is in projects show page
Steps	A	Press edit button of the update that needs to be edited
	B	Enter new update date / missing
	C	Enter new remarks of the update / missing
	D	Select Case the update belongs to from drop down.
Inputs		No date/ remarks / No remarks/date or both missinhg
Expected results		Displys message showing the missing fields /filed
Status		Tested, Passed

07. Deleting projects

Positive Test Cases		
ID	TCN-07	
Priority	High	
Description	To verify weather users are able to delete projects	
Reference	Functional Requirement reference	
Users	Type: ADMIN (Lawfirms)	
Pre-requisites	A	System is online.

	B	Logged in as Admin
	C	Is in projects index page
Steps	A	Press Delete button from
	B	Enter new update date
	C	Enter new remarks of the update
	D	Select Case the update belongs to from drop down.
Inputs		Date, Remarks or both with new values or with one new value.
Expected results		Shows sweet alert message saying the update was edited successfully
Status		Tested, Passed

08. Deleting users from system

Positive Test Cases		
ID	TCN-08	
Priority	High	
Description	To verify weather lawfrims are able to delete old clients	
Reference	Functional Requirement reference	
Users	Type: ADMIN (Lawfirms)	
Pre-requisites	A	System is online.
	B	Logged in as Admin
	C	Is in users index page
Steps	A	Press Delete button from the user that needs to be deleted
Inputs		-
Expected results		Shows sweet alert message saying the user waa deleted successfully
Status		Tested, Passed

09. Sending document requests

Positive Test Cases		
ID	TCN-09	
Priority	High	
Description	To verify weather law firms are able to request clients for document	
Reference	Functional Requirement reference	
Users	Type: ADMIN (Lawfirms)	
Pre-requisites	A	System is online.
	B	Logged in as Admin
	C	Is in document request create page
Steps	A	Enter request titile
	B	Select client to be requested from drop-down
	C	Press submit buttom

Inputs		Title, Client
Expected results		Shows sweet alert message saying request was sent to the requested client, the requested client receives an email
Status		Tested, Passed

Negative Test Cases		
ID	TCN-07	
Priority	High	
Description	To make sure that document requests cannot be proceeded with missing details	
Reference	Functional Requirement reference	
Users	Type: ADMIN (Lawfirms)	
Pre-requisites	A	System is online.
	B	Logged in as Admin
	C	Is in document request create page
Steps	A	Enter request titile
	B	Select client to be requested from drop-down
	C	Press submit buttom
Inputs		Missing client/description or both
Expected results		Shows error message showing the missing fileds
Status		Tested, Passed

10. Viewing profile as a client

Positive Test Cases		
ID	TCN-10	
Priority	High	
Description	To verify that clinents are able to view their profiles	
Reference	Functional Requirement reference	
Users	Type: Go user (Clients)	
Pre-requisites	A	System is online.
	B	Logged in as Go user
Steps	A	Click profile from dashboard
Inputs		-
Expected results		Shows client user details along with project details
Status		Tested, Passed

11. Viewing project updates as a client

Positive Test Cases	
ID	TCN-11
Priority	High
Description	To verify that clients are able to view case updates

Reference	Functional Requirement reference	
Users	Type: Go user (Clients)	
Pre-requisites	A	System is online.
	B	Logged in as Go user
Steps	A	Click project updates
Inputs		-
Expected results		Shows table of updates from latest first date wise.
Status		Tested, Passed

12. Uploading documents when requested.

Positive Test Cases		
ID	TCP-12	
Priority	High	
Description	To verify that clients are able to upload documents.	
Reference	Functional Requirement reference	
Users	Type: Go user (Clients)	
Pre-requisites	A	System is online.
	B	Logged in as Go user
Steps	A	Click document requests
	B	Select file to upload
	C	Press submit button
Inputs		File type : csv,txt,xlx,xls,pdf not greater than 02 MB
Expected results		Display sweet alert message thanking the user, Notification from dashboard gone, Document request submission gone from dashboard.
Status		Tested, Passed

Negative Test Cases		
ID	TCN-08	
Priority	High	
Description	To verify are unable to submit wrong type of files. Or empty	
Reference	Functional Requirement reference	
Users	Type: Go user (Clients)	
Pre-requisites	A	System is online.
	B	Logged in as Go user
Steps	A	Click document requests
	B	Select file to upload
	C	Press submit button
Inputs		File type: other types such as GPEJ, PNG, **other than** (csv, txt, xlx, xls, pdf) **greater than** 02 MB

Expected results		Shows error message asking to upload correct file, asking to reduce file sise.
Status		Tested, Passed

5.2.1. Equivalence Partitions (EP)

Equivalence partitioning is a black box testing method that divides the input data of a software unit into partitions of data from which test cases can be derived. It reduces no. of test cases. Inequivalence class partitioning an equivalence class is formed of the inputs for which the behaviour of the system is specified or expected to be similar. An equivalence class represents a set of valid or invalid states for input conditions. (Khan, 2011)

While looking at the Equivalence class partitioning, I have used this technique in many ways to reduce the number of test cases and to increase the efficiency of the tests carried out.

For example in test case TCP-05 I have entered two invalid inputs / one valid input and have carried the test case with empty fields. This technique has been used in carrying out other test cases aswell.

5.2.2. Boundary Value Analysis

Boundary value analysis is a testing technique that focuses more on testing at boundaries or where the extreme boundary values are chosen. Boundary value include maximum, minimum, just inside/outside boundaries, typical values and error values. (Khan, 2011) This technique also have been used in my test cases to increase the accuracy of the test cases.

For Example in **TCN-01** the system only takes passwords that are strings, minimum of 8 charaters, and after being written 02 times. Hence to make this 100 % sure I have tested the closest to it I have entered 7 characters.

Future more in the same test case emails with similar names have been check to make sure that the system only enter unique emails.

5.2.3. Decision Table Testing

Decision table testing is a software testing technique used to test system behaviour for different input combinations. This is a systematic approach where the different input combinations and their corresponding system behaviour (Output) are captured in a tabular form. That is why it is also called as a Cause-Effect table where Cause and effects are captured for better test coverage. (Hamilton, decision-table-testing, 2021).

I have used decision table testing in testing certain functionalities such as logging to the System as GO user (Client) and As Admin (Lawfirm). (T- Correct username/ Password | F- Incorrect Username/Passowrd)

Inputs	Rule 1	Rule 2	Rule 3	Rule 4
Email (Go user)	T	T	F	F
Password (GO user)	T	F	T	F
Expected results	Client Dashboard	Display Incorrect Password	Display unregistered email	Display no registration
Email (Admin)	T	T	F	F
Password (Admin)	T	F	T	F
Expected results	Law firm Dashboard	Display Incorrect Password	Display unregistered email	Display no registration

5.2.4. State transition Testing

This is a testing technique used to check the change in the state of the system depending on the inputs given. With this technique the systems response to a given change in input is analysed. In above test case TCP-12 this technique can be analysed when the user uploads the correct files for document requests the system removes the notification button from the respective user. However if the file type is different the system sends incorrect message and keeps the notification unresolved. This test technique has been used in other areas as well.

5.2.5. Use Case Testing

Use Case Testing is a software testing technique that helps to identify test cases that cover entire system on a transaction-by-transaction basis from start to end. Test cases are the interactions between users and software application. Use case testing helps to identify gaps in software application that might not be found by testing individual software components. (Hamilton, use-case-testing, 2021)

Please note that this test cases was focused and carried out in details with the use case analysis.

5.3. White Box Test Cases

White-box testing refers to test methods that rely on the internal structure of the software. White-box methods are based on executing or “covering” specific elements of the code. The essential rationale for these methods is that it is impossible to detect a fault in some piece of code by testing if that code is never executed. (Ostrand, 2002). I have used white box testing in number of ways to verify that the systems code execution and other database queries has worked perfectly.

5.3.1 Cyclometric complexity

Cyclomatic complexity is software metric that delivers a quantitative degree of the logical difficulty of a program. Cyclomatic Complexity (CYC) is derived as the number of edges of the program’s control-flow graph minus the number of its nodes plus two times the number of its linked components. (Srinivas Nidhra, 2012). I have used this technique in testing of the system to identify the number of test cases to be written for certain process. For example if a particular process involves a if condition the cyclometric complexity of the system will be 02 by default.

5.3.2 Statement coverage

Statement coverage technique is used to design white box test cases. This technique involves execution of all statements of the source code at least once. It is used to calculate the total number of executed statements in the source code out of total statements present in the source code. (javatpoint.com, n.d.) Test cases written above has used this technique, for example when a user submits the form, the validations have been tested to verify that all the validation of the form was tested during the testing.

5.4 Performance testing

Performance testing is the practice of evaluating how a system performs in terms of responsiveness and stability under a particular workload. Performance tests are typically executed to examine speed, robustness, reliability, and application size. To test the performance of the client management system for the targeted users I have tested the system on MN Lawyers provided hosting with Ooredoo (5MPS) internet connection to check the speed of systems response.

Chapter 6: Tools and Techniques

There has been plenty of tools used in the development of client Management system Clientio. In this section I will be highlighting the numerous tools that I have used in the development of Clientio. Firstly, I would like to highlight the tools used for development such as Editors and other platforms used in development.

01. Php storm

https://www.jetbrains.com/phpstorm/

Php storm by jetbrains was the IDE I have used as main language used in this application was PHP powered by Laravel.

02. **XAMPP**

https://www.apachefriends.org/index.html

XAMPP was the local server I used in the development of the application , XAMPP tools such as PHP myadmin and other resources have come in plenty in this development of my final project.

03. Visual studio Code

https://code.visualstudio.com/

Although most of the backend development was done with PHP storm, the frot end implementations and CSS files has been edited with Visual Studio Code as I prefer it.

04. Mail trap

https://mailtrap.io/

Mailtrap is **a test mail server solution** that allows testing email notifications without sending them to the real users of your application. I have used this service to test my email services and email templates while developing.

05. Git & GitHub

https://git-scm.com/

https://github.com/faarih01

I have used both git & GitHub for version controlling services and to share my development with other testers.

Secondly, I would like to highlight Languages, Frameworks & Libraires used in development of this application.

01. Php Laravel (Version 8)

https://laravel.com/docs/8.x

Laravel is a web application framework with expressive, elegant syntax. I have chosen this framework because the idea of the system was a web application and Laravel 8 had the capabilities to easily build this system.

02. Sweet alert JS library

https://sweetalert2.github.io/

This JavaScript library has been used in application during a number of process to show notification and alert messages.

03. JavaScript

Although my application is not JS application, I have used JavaScript for front end development, specially Menus and drag & drop buttons.

04. Bootstrap 05

https://getbootstrap.com/docs/5.0/getting-started/introduction/

Bootstrap 5 is a very common front-end framework used in web development widely these days. I have also used this in my front-end development. About 80 % of the frond end code will belong to bootstrap classes.

Chapter 7: Summary and Conclusion

To Conclude, this project will be useful software/tool for small to middle scale law firms who are looking for a modern way to manage their clients. If this project is used in it most efficient way by law firms looking to make a change, public would get many eases as well. As this is a multi-authenticated application powered by Laravel 8 this project can handle multiple users with ease.

Chapter 8: User Manual

The following is the user manual for using the client management system. Firstly I would like to continue with a manual for setting this application. I have this application files at https://github.com/faarih01/clientio free to use for anyone who wants to try this out.

Steps to install the application

01. Clone the repository from github.com

```
git clone https://github.com/faarih01/clientio.git
```

02. Install composer

```
composer install
```

03. Setup environment file by putting the following details in the application.

    A new user can copy the .env.example file and save it as .env. Please note in this demo I’m going to be setting up the application for local host and will be using the mailtrap.io for email server. Save the .env file as bellows.

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:ak/yknj11d9qDGtXRHPPsesA4iw1Rs4tuENN0490TWM=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=test
DB_USERNAME=root
DB_PASSWORD=

BROADCAST_DRIVER=log
CACHE_DRIVER=file
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

MEMCACHED_HOST=127.0.0.1

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_MAILER=smtp
MAIL_HOST=smtp.mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=79ab0ab0e6b9ed
MAIL_PASSWORD=caab525f9646af
MAIL_ENCRYPTION=tls
MAIL_FROM_ADDRESS=mnlawyers@legal.com
MAIL_FROM_NAME="${APP_NAME}"

AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=

PUSHER_APP_ID=
PUSHER_APP_KEY=
PUSHER_APP_SECRET=
PUSHER_APP_CLUSTER=mt1

MIX_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
MIX_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
```

04. If the application asks for APP key

    ```
    php artisan key:generate
    ```

05. Create empty database same as written in environment file with access to it in phpMyAdmin or in your local database manager.
06. Run migration command
    `php artisan migrate`

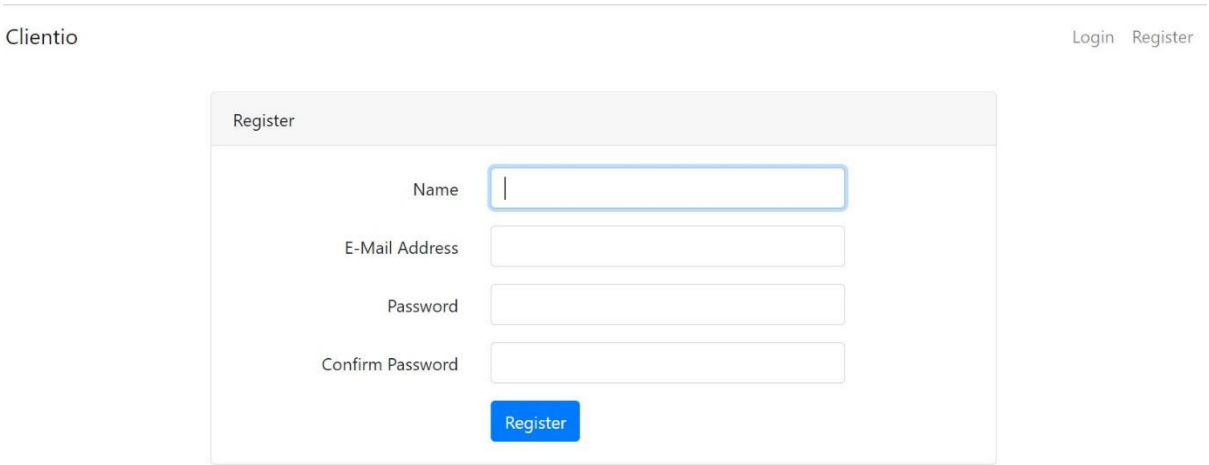
07. Run the application
    `php artisan serve`

The application will now show default landing page for unregistered users when served.

Note: This user manual will be available to download from the link mentioned in the above image.

Managing the application

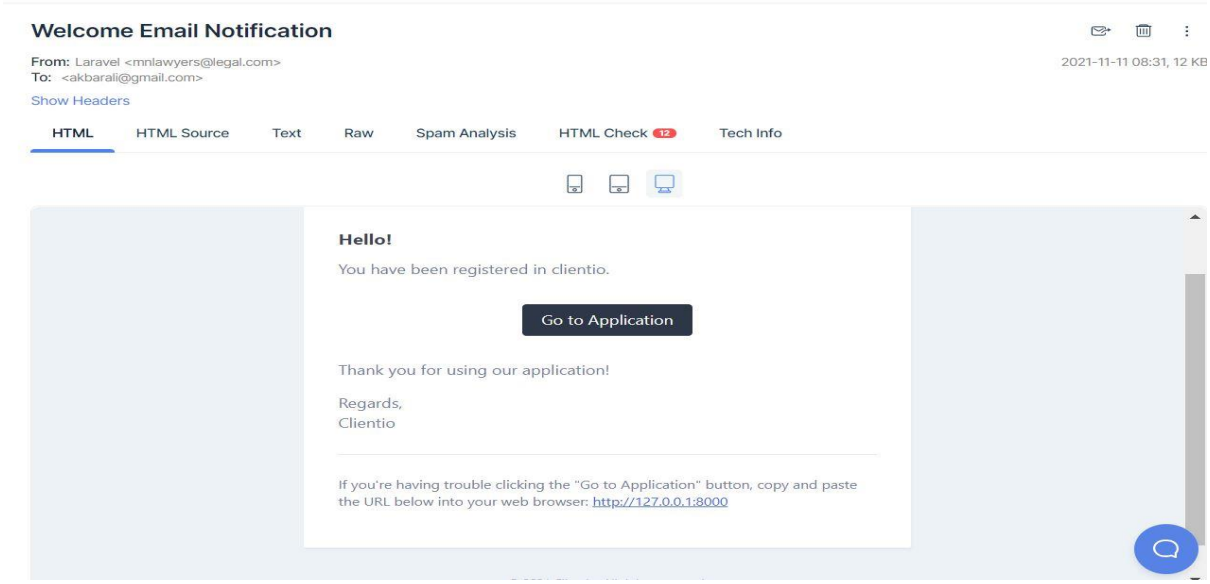
01. Registering the first administrator account.



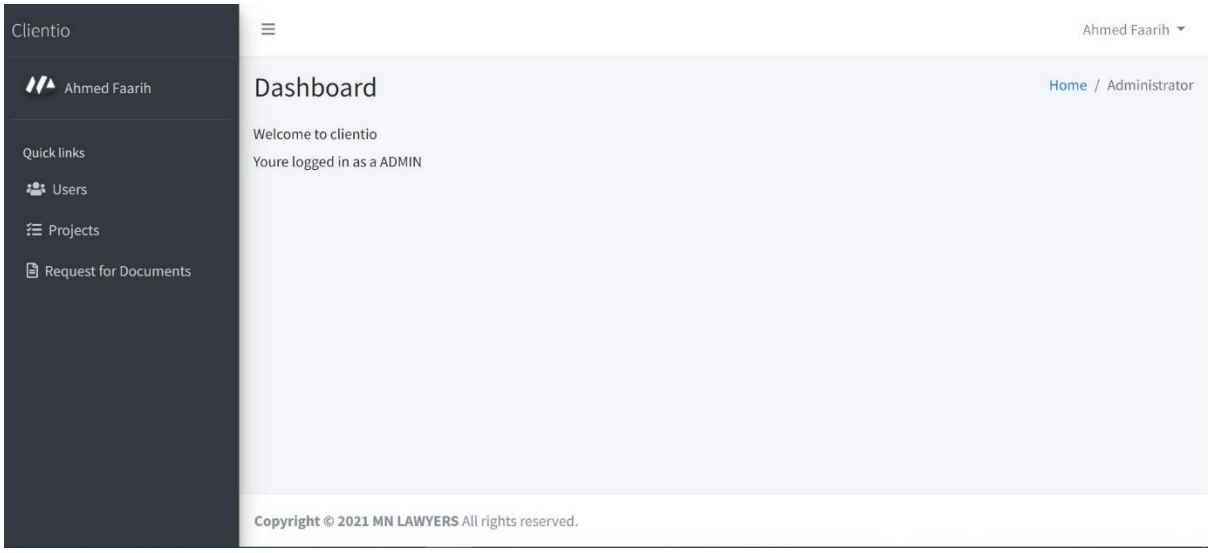
The above is the default registration page. No that all users must register themselves for the application via which the registration. Note all users will be first registered as Clients. The first user to register to application should change their user type to ADMIN from php myadmin or their prefered database manager. See below.

Save and reload the application and the user will be reloaded with administrator rights.

02. Upon successful registration the user will receive an email saying...

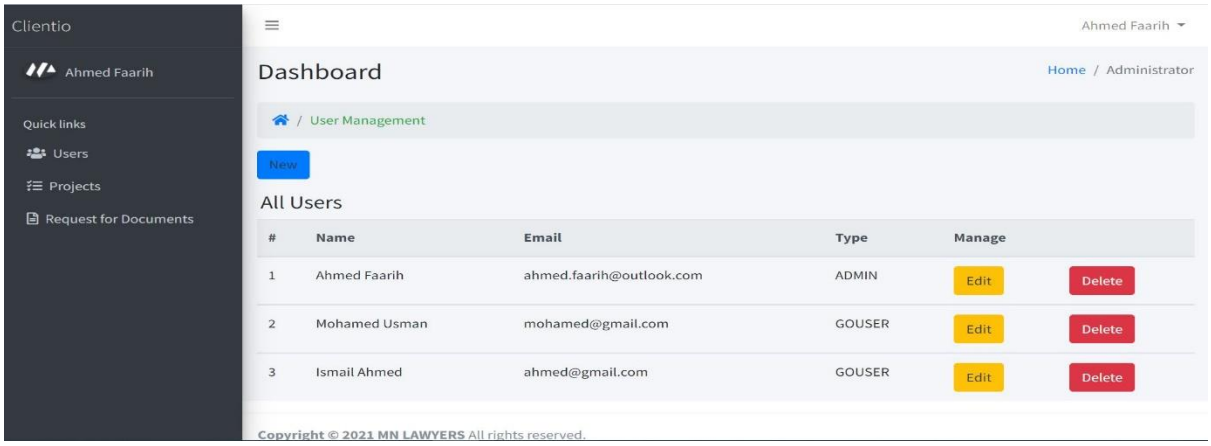


03. The administrator dashboard

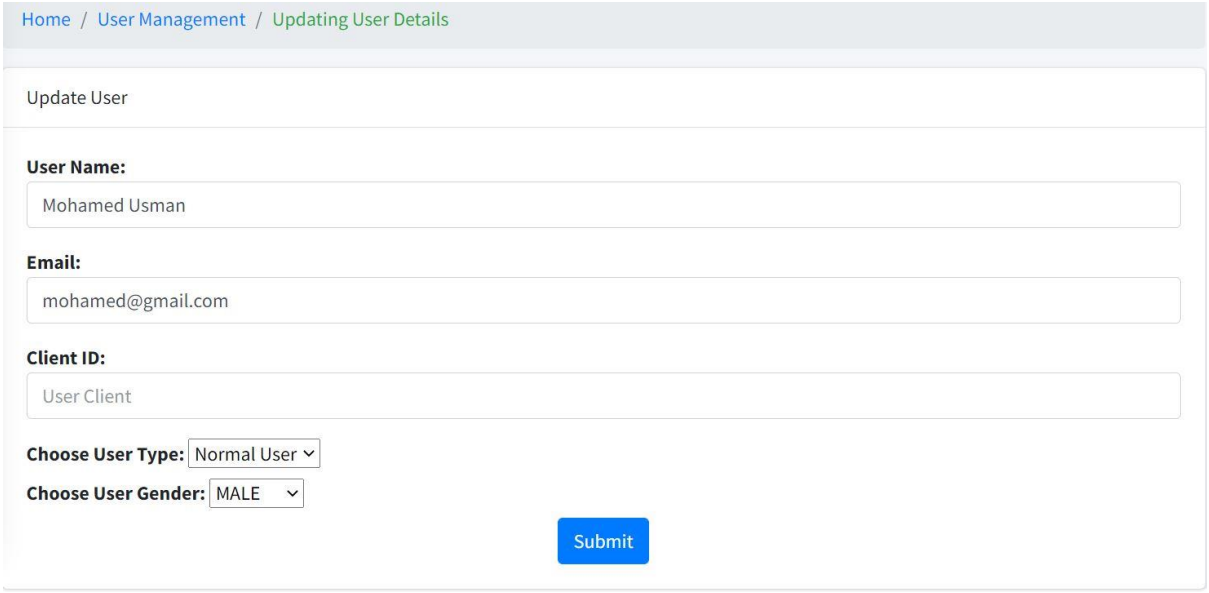


Note the dashboard should say the user is logged in as administrator.

04. Managing users. When the admin ie lawfirm users click users page it should load users index page. With registered users. As below.



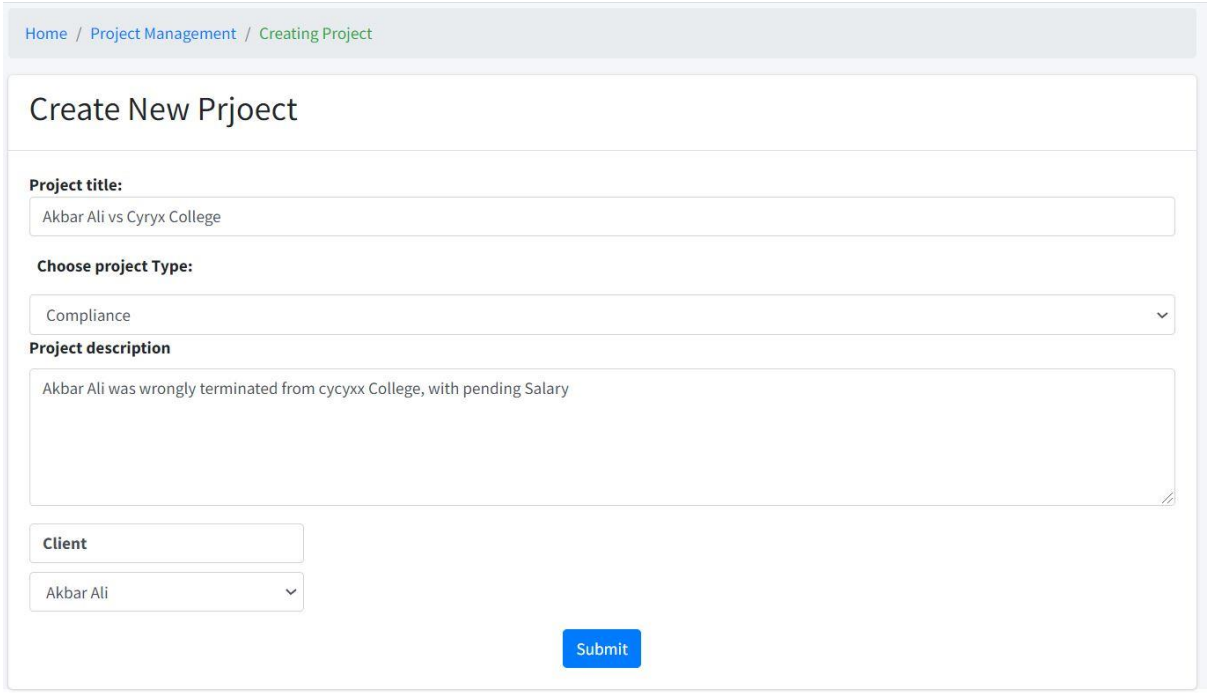
05. Editing existing users.



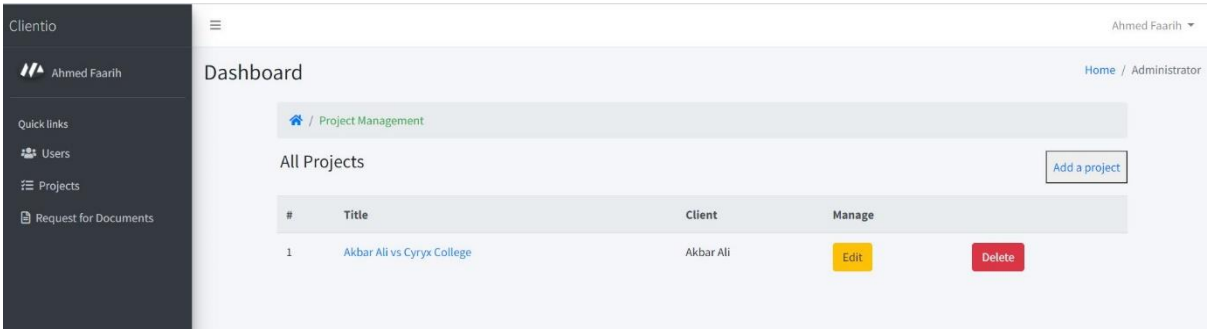
From above the administrator can change user type from Go user/ Admin, so that after creating the first administrator account the user can skip the php myadmin and change users roles directly from the dashboard. The administrator can also give Client ID if the user is a client and change his gender as well.

06. Creating projects.

The law firm should enter the case details and take the case user that is client. For example in the below case Akbar Ali.



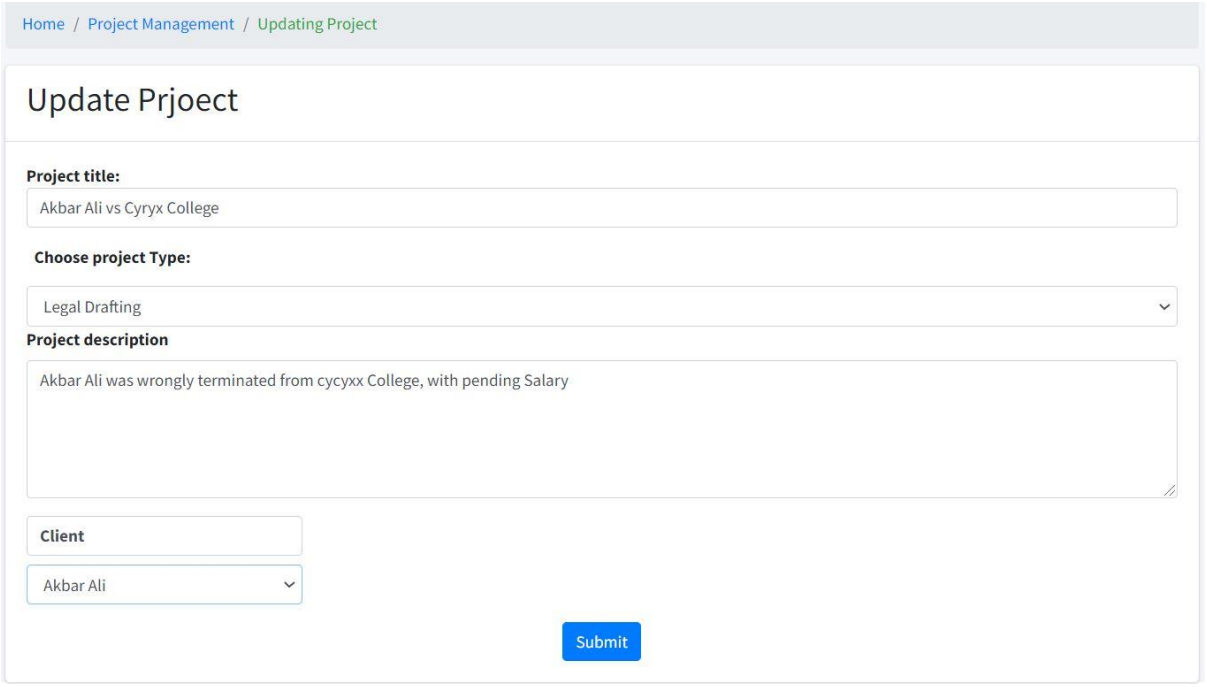
07. Projects page after creation of projects.



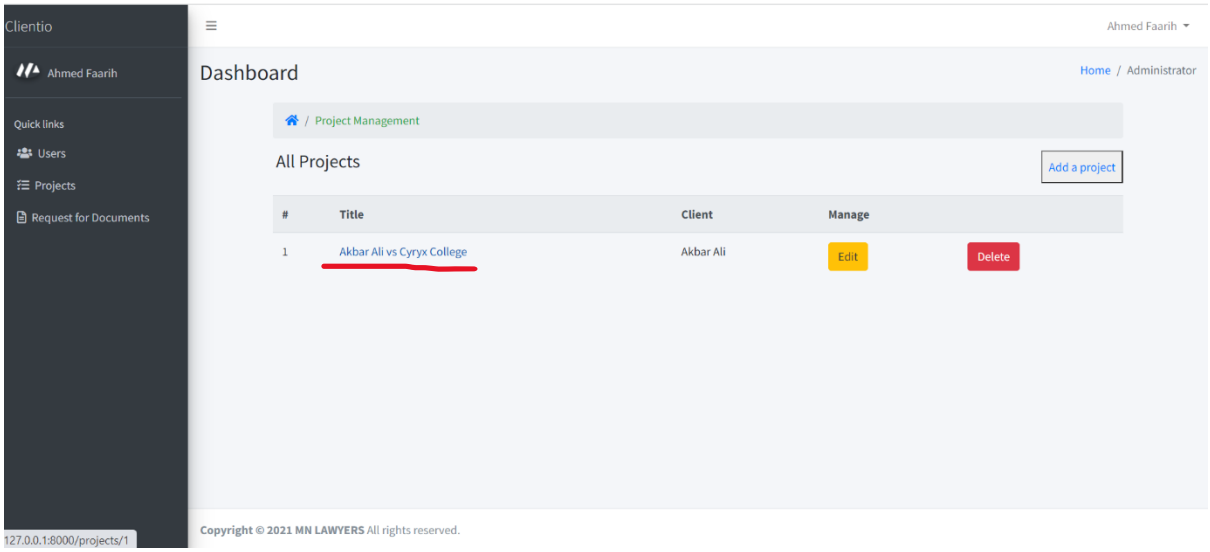
As mentioned above users should be able to delete the project from delete button on the projects index page.

08. Editing the project

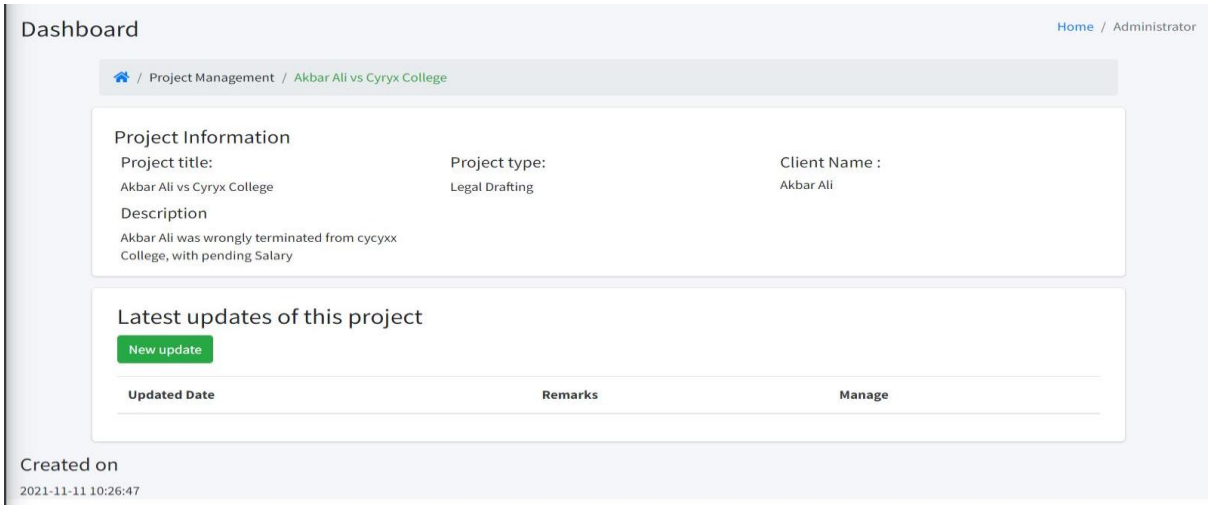
The edit button should take the user to the edit page with the existing data of the project page.

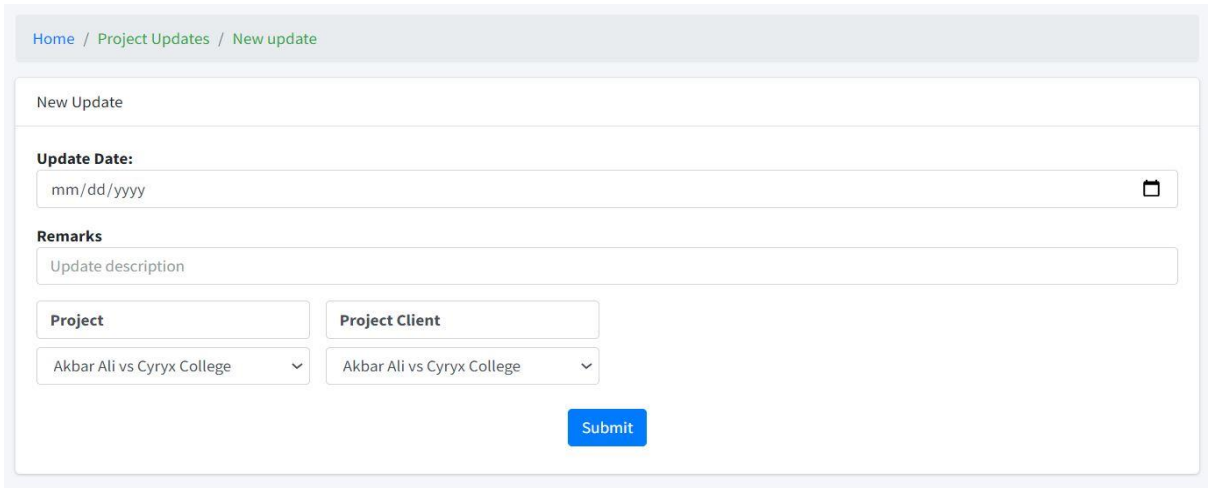


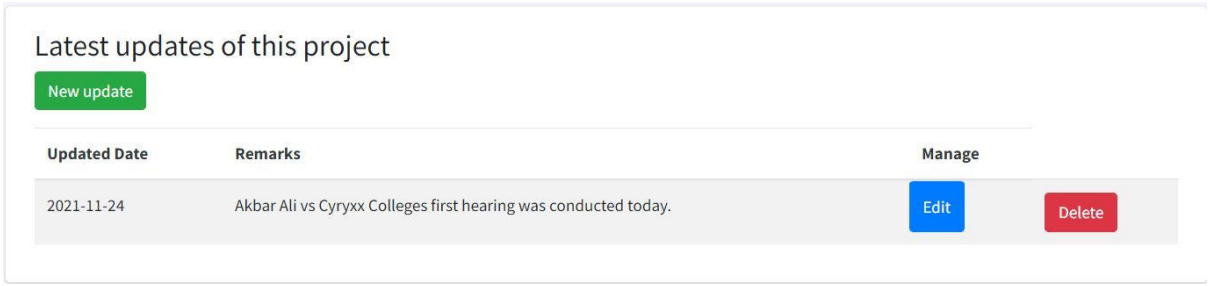
09. Adding updates to created project. Note to view the details of the project or to create updates to the project the project title should be clicked.



When the user clicks the project title the user should be proceeded to the details of the project. From the below updates section the user should proceed to new update.

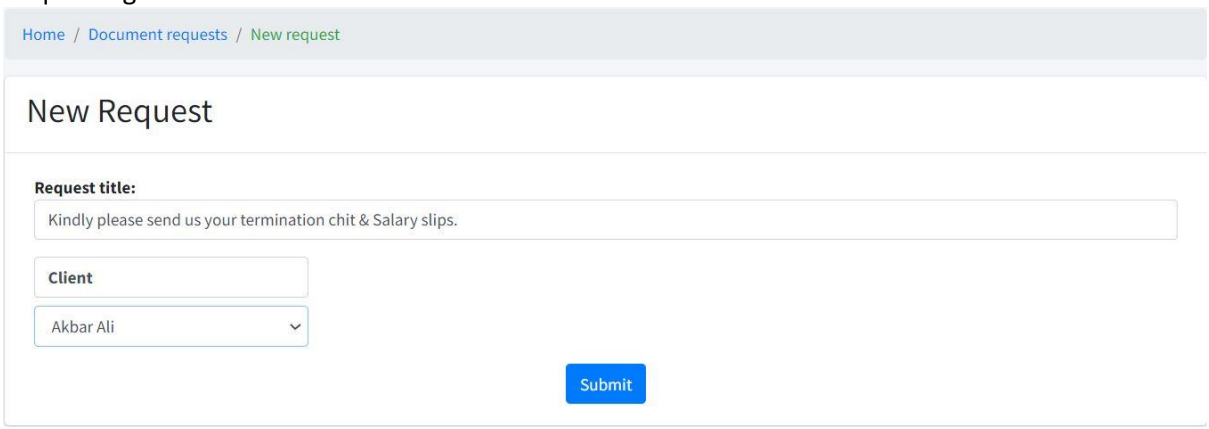




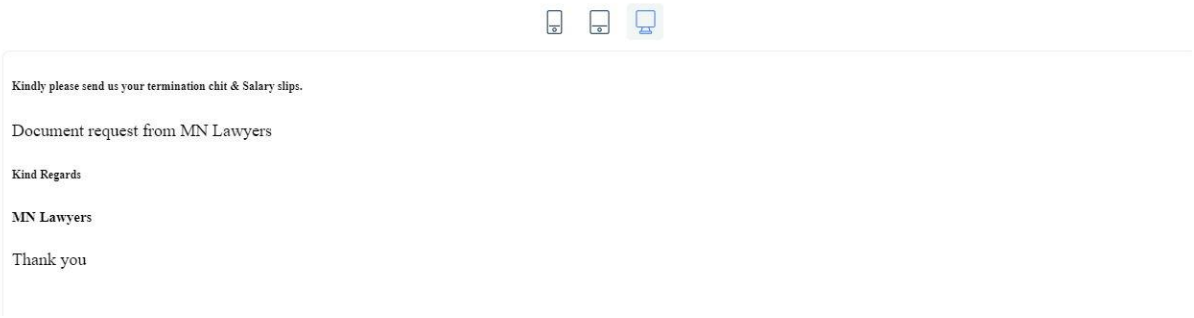


The new update should be shown as below.

10. Requesting for documents

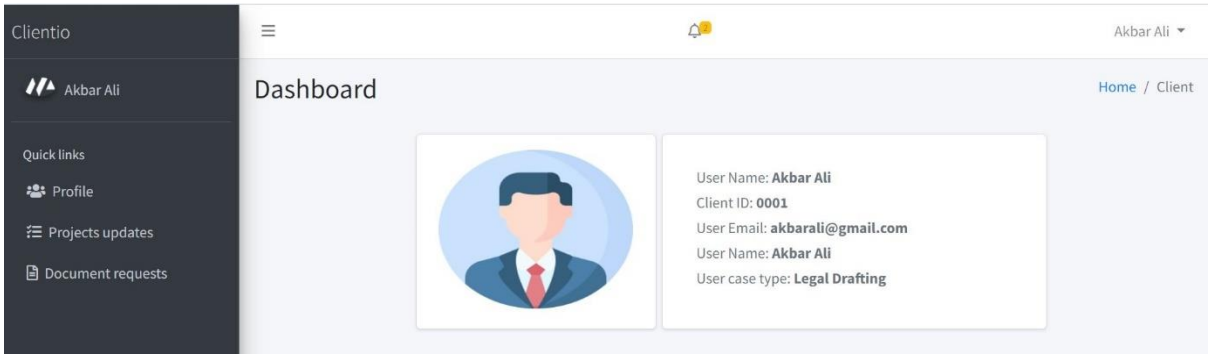


11. Upon successful request, the client should receive the below email

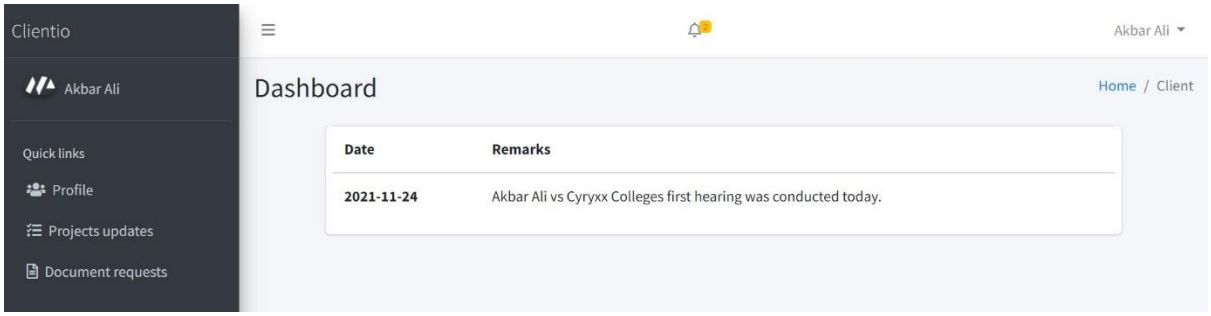


Logging in as Go user.

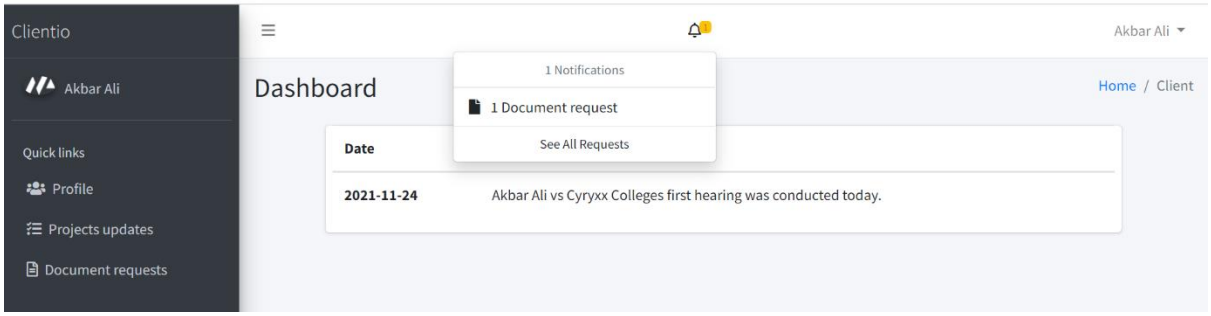
The profile button will show this page with the details of the user.



The update button will show the updates of the logged in user’s case.

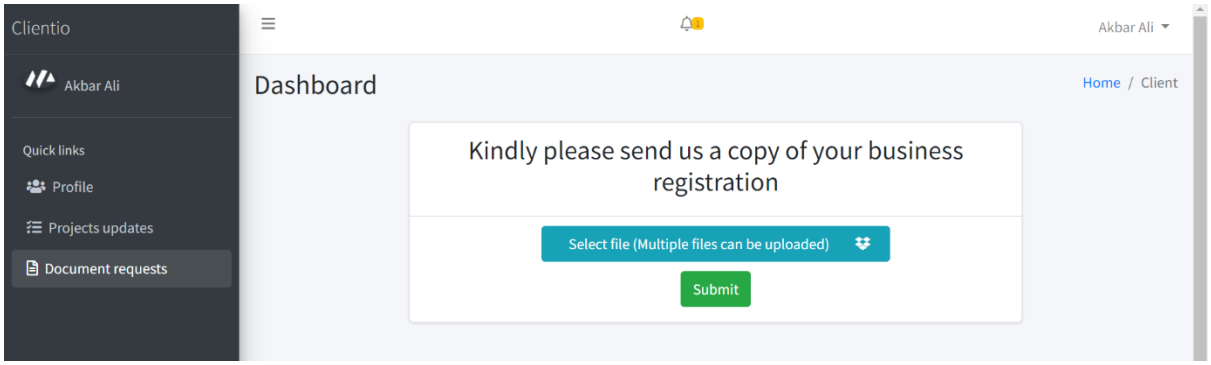


When the user clicks the notification button…

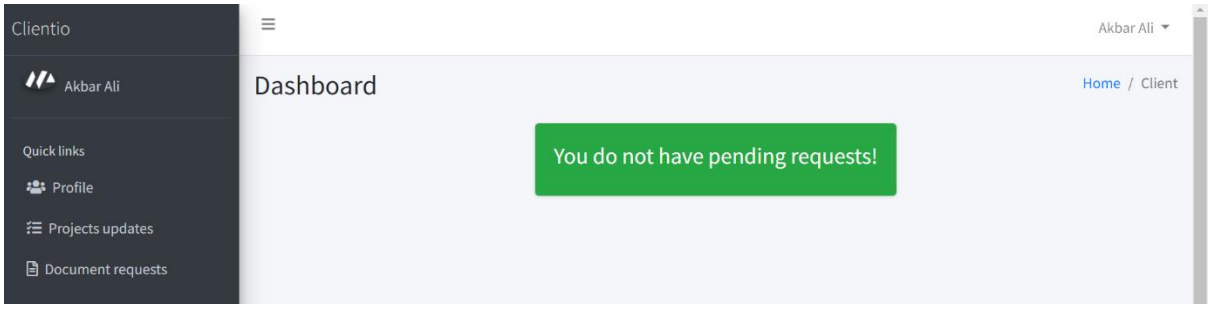


This should take the user to document requests page or the user can also use the side menu if preferred.

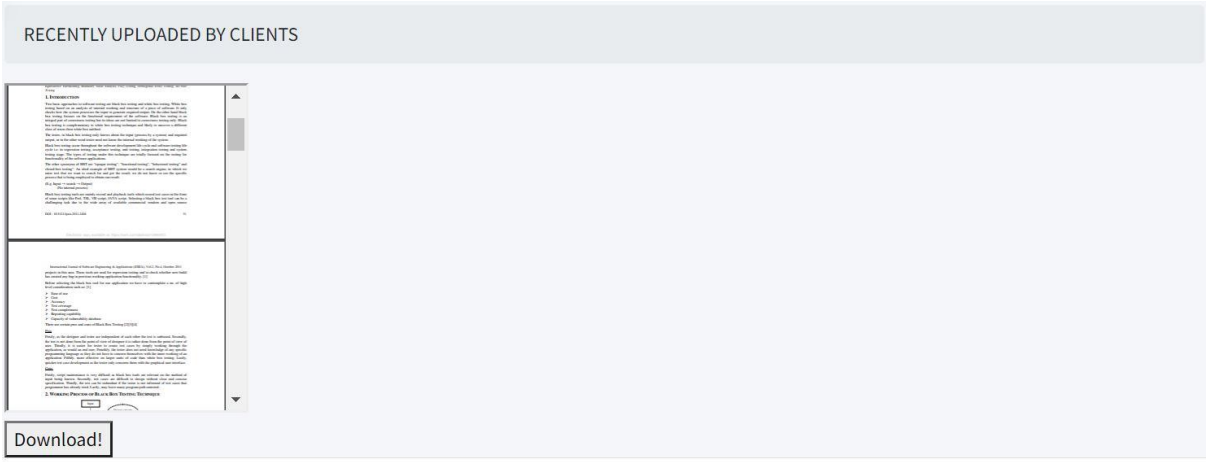
This would be the document requests upload page. The user should upload the requested file /files.



Upon submission of the request the notification from the users dashboard should be gone.



When logged in as an administrator, the uploaded file should be seen as document requests. As below ordered from last to first.



The above is a simple user manual for the client management system with important functions demonstrated.

Chapter 9: Lessons learnt & Future Enhancements

This project has thought me a lot about backend web development and Laravel 8 applications. This project has also helped me learn about Software testing and to implement tests, write test cases for an application in ways I have not tried before. While looking back at the project, this has helped me develop my skill set as a Developer especially back end.

While looking at future enhancements, there would always be room for improvement and to increase the scope of the application. For example, a payment gateway such as BML gateway or other activities of law firms based on their requirements.

References

Chen, Q. L.-L. (2009). Entity-Relationship Diagram. In Q. L.-L. Chen, *Modeling and Analysis of Enterprise and Information Systems. .* Berlin: Springer.

Daniela Berardi, D. C. (2005). *Reasoning on UML class diagrams.* Rome: Faculty of Computer Science, Free University of Bolzano/Bozen.

Ennico, C. (2021, June 25). *https://www.entrepreneur.com/article/58326*. Retrieved from https://www.entrepreneur.com: https://www.entrepreneur.com/article/58326

Hamilton, T. (2021, October 08). *decision-table-testing*. Retrieved from guru99.com: https://www.guru99.com/decision-table-testing.html

Hamilton, T. (2021, October 19). *test-case*. Retrieved from guru99.com: https://www.guru99.com/test-case.html

Hamilton, T. (2021, October 21). *use-case-testing*. Retrieved from https://www.guru99.com/: https://www.guru99.com/use-case-testing.html

javatpoint.com. (n.d.). Retrieved from https://www.javatpoint.com: https://www.javatpoint.com/statement-coverage-testing-in-white-box-testing

Khan, M. E. (2011). DIFFERENT APPROACHES TO BLACK BOX. *International Journal of Software Engineering & Applications (IJSEA) Vol 2*, 32-33.

Kurt Bittner, I. S. (2003). *Use Case Modelling.* Boston: Pearson Education Inc.

Martin, M. (2021, October 06). *functional-vs-non-functional-requirements*. Retrieved from www.guru99.com: https://www.guru99.com/functional-vs-non-functional-requirements.html

Nielsen, J. (1994). *Usability Engineering.* San Francisco: Morgan Kaufmann Publishers Inc.

Ostrand, T. (2002, January 15). White-Box Testing. *Encyclopedia of Software Engineering*.

Paradigm, V. (n.d.). *Guide*. Retrieved from www.visual-paradigm.com: https://www.visual-paradigm.com/guide/requirements-gathering/requirement-analysis-techniques/

Rajkumar. (2021, 06 28). *software-testing*. Retrieved from www.softwaretestingmaterial.com: https://www.softwaretestingmaterial.com/software-testing/

Srinivas Nidhra, J. D. (2012). BLACK BOX AND WHITE BOX TESTING TESTING TECHNIQUES. *International Journal of Embedded Systems and Applications*, 44.

www.lucidchart.com/. (n.d.). *uml-sequence-diagram*. Retrieved from www.lucidchart.com: https://www.lucidchart.com/pages/uml-sequence-diagram

www.visual-paradigm.com. (n.d.). *uml-unified-modeling-language/what-is-activity-diagram/*. Retrieved from visual-paradigm.com.