Design Document

Design Document: Version 1

Table of Contents

1. Introduction

The project aims to create client management software for legal firms, optimizing communication and efficiency for improved client satisfaction and operational transparency.

2. Architecture Constraints and Design Decisions

Why Spring Boot, React, and MySQL?

Spring Boot:

- Simplifies coding process.
- Vast ecosystem and community support.
- Convention-over-configuration approach.

React:

- Fast, responsive, and component-based.
- Dynamic user interfaces.
- Extensive ecosystem with tools and libraries.

MySQL:

- Reliable and scalable.
- Robust features for managing structured data.
- Support for transactions and integrity constraints.

3. SOLID Principles

In my project, the implementation of SOLID principles is fundamental to ensuring a robust and maintainable software architecture.

Single Responsibility Principle :

Backend Layers:

- **Business Layer**: Responsible for encapsulating the business logic of the application, ensuring that it remains focused on core functionality.
- **Controller Layer**: Handles incoming requests, routing them to appropriate business logic, and preparing responses.
- **Domain Layer**: Defines the core domain models and entities, maintaining the integrity and consistency of the data.

- **Persistence Layer**: Manages the storage and retrieval of data, ensuring efficient data access and management operations.

Open/Closed Principle:

Dependency Injection:

- Utilizing dependency injection allows for the extension of functionalities without modifying existing code. This design choice promotes code reusability and scalability by decoupling components and facilitating easier testing and maintenance.

Liskov Substitution Principle :

Interface Implementation:

- Interfaces are designed to adhere to the LSP, ensuring that derived classes can be substituted without altering the correctness of the system. This approach enables seamless swapping of components and promotes flexibility in design.

Interface Segregation Principle:

Tailored Interfaces:

- Interfaces are tailored to specific client needs, adhering to the ISP to prevent clients from depending on unnecessary functionalities. Each interface exposes only the methods required by its clients, promoting modularity and reducing dependencies.

Dependency Inversion Principle:

Inversion of Control (IoC):

- Implementing IoC containers facilitates dependency inversion, decoupling high-level modules from low-level implementations. Controllers depend on abstractions (interfaces) rather than concrete implementations, promoting flexibility and testability.

Backend Classes and Interfaces:

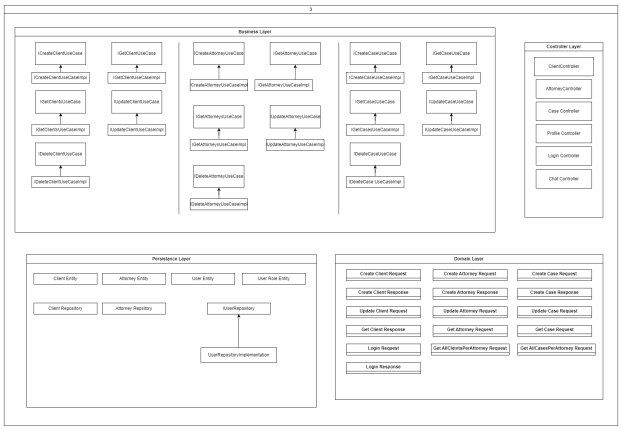
- Each class and function in the backend is meticulously designed to adhere to the SRP, with a clear focus on a single responsibility. Additionally, the use of interfaces ensures loose coupling between components, allowing for easier maintenance and testing.

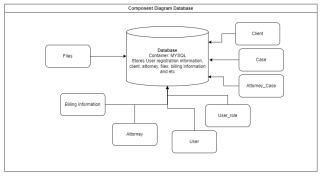
Frontend Layering:

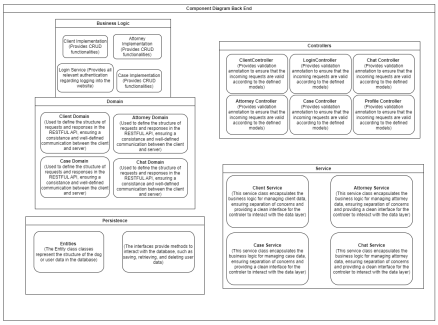
Components, Pages, and APIs:

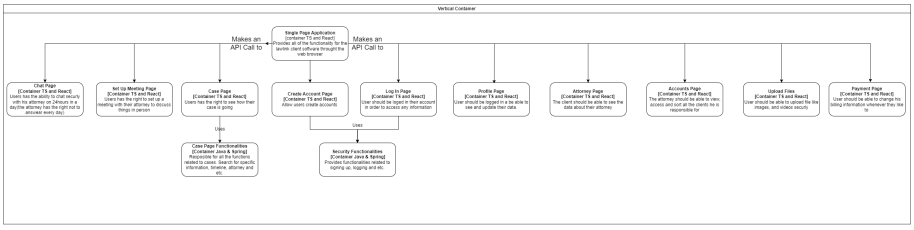
- The frontend architecture is organized into layers for components, pages, and APIs. For example, a user page consists of three components - single account, list of accounts, and account creation. This structured approach facilitates easier development, maintenance, and scalability of the frontend."

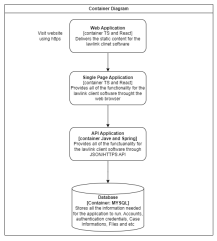
4. C4 Diagrams

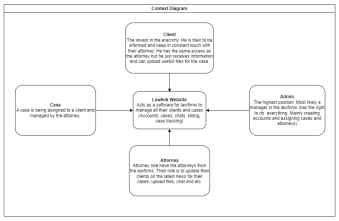

3
Business Layer
ICreateClientUseCase
ICreateClientUseCaseImpl
IGetClientUseCase
IGetClientUseCaseImpl
IGetClientsUseCase
IGetClientsUseCaseImpl
IUpdateClientUseCase
IUpdateClientUseCaseImpl
IDeleteClientUseCase
IDeleteClientUseCaseImpl
ICreateAttorneyUseCase
ICreateAttorneyUseCaseImpl
IGetAttorneyUseCase
IGetAttorneyUseCaseImpl
IGetAttorneysUseCase
IGetAttorneysUseCaseImpl
IUpdateAttorneyUseCase
IUpdateAttorneyUseCaseImpl
IDeleteAttorneyUseCase
IDeleteAttorneyUseCaseImpl
ICreateCaseUseCase
ICreateCaseUseCaseImpl
IGetCaseUseCase
IGetCaseUseCaseImpl
IGetCasesUseCase
IGetCasesUseCaseImpl
IUpdateCaseUseCase
IUpdateCaseUseCaseImpl
IDeleteCaseUseCase
IDeleteCase UseCaseImpl
Controller Layer
ClientController
AttorneyController
Case Controller
Profile Controller
Login Controller
Chat Controller
Persistance Layer
Client Entity
Attorney Entity
User Entity
User Role Entity
Client Repository
Attorney Repository
UserRepository
UserRepositoryImplementation
Domain Layer
Create Client Request
Create Client Response
Update Client Request
Get Client Response
Login Request
Login Response
Create Attorney Request
Create Attorney Response
Update Attorney Request
Get Attorney Request
Get AllClientsPerAttorney Request
Create Case Request
Create Case Response
Update Case Request
Get Case Request
Get AllCasesPerAttorney Request


Component Diagram Database
Database
Container: MYSQL
Stores all the information needed, client, attorney, files, billing information and etc
Files
Billing information
Client
Case
Attorney_Case
User_role
Attorney
User

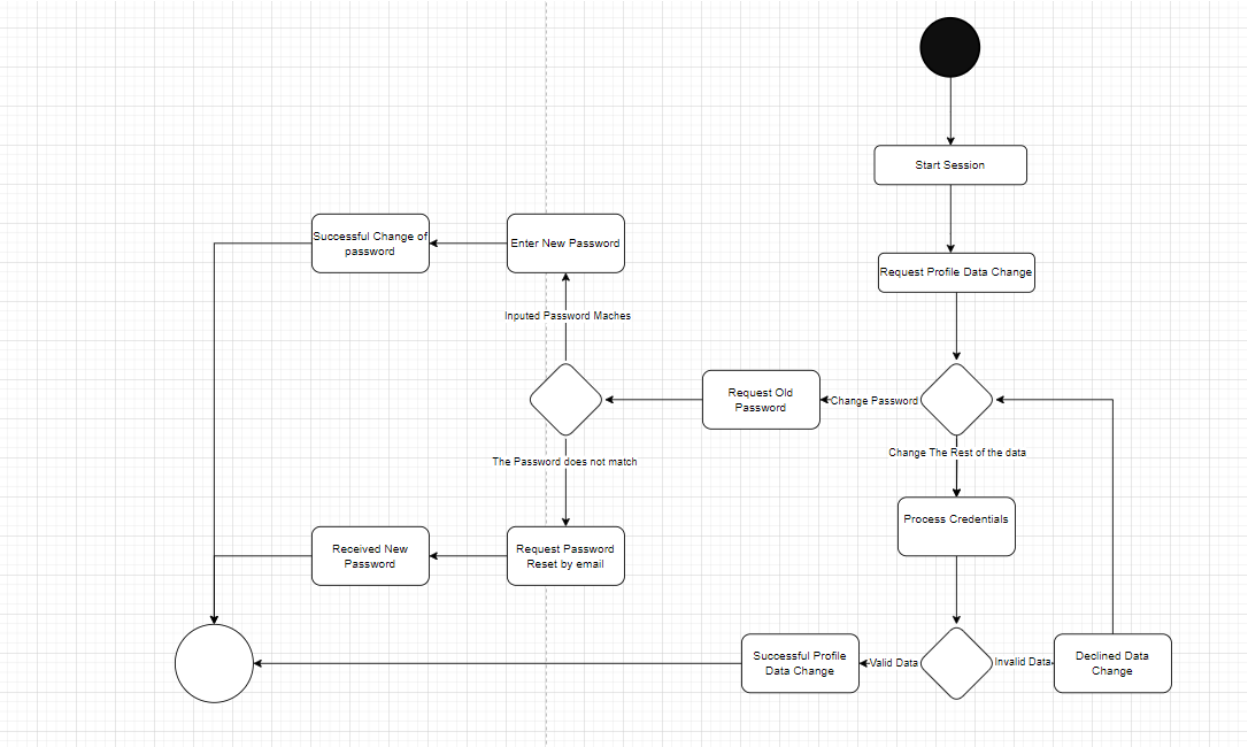

Component Diagram Back End
Business Logic
Client Implementation (Provides CRUD functionalities)
Attorney Implementation (Provides CRUD functionalities)
Login Service (Provides all relevant authentication regarding logging into the website)
Case Implementation (Provides CRUD functionalities)
Domain
Client Domain (Used to define the structure of requests and responses in the RESTFUL API, ensuring a consistance and well-defined communication between the client and server)
Attorney Domain (Used to define the structure of requests and responses in the RESTFUL API, ensuring a consistance and well-defined communication between the client and server)
Case Domain (Used to define the structure of requests and responses in the RESTFUL API, ensuring a consistance and well-defined communication between the client and server)
Chat Domain (Used to define the structure of requests and responses in the RESTFUL API, ensuring a consistance and well-defined communication between the client and server)
Persistence
Entities (The Entity class classes represent the tables in the db) or user data in the database)
(The interfaces provide methods to interact with the database, such as saving, retrieving, and deleting user data)
Controllers
ClientController (Provides validation annotation to ensure that the incoming requests are valid according to the defined models)
LogicController (Provides validation annotation to ensure that the incoming requests are valid according to the defined models)
Chat Controller (Provides validation annotation to ensure that the incoming requests are valid according to the defined models)
Attorney Controller (Provides validation annotation to ensure that the incoming requests are valid according to the defined models)
Case Controller (Provides validation annotation to ensure that the incoming requests are valid according to the defined models)
Profile Controller (Provides validation annotation to ensure that the incoming requests are valid according to the defined models)
Service
Client Service (This service class encapsulates the business logic related to client data, ensuring separation of concerns and providing a clean interface for the controller to interact with the data layer)
Attorney Service (This service class encapsulates the business logic related to attorney data, ensuring separation of concerns and providing a clean interface for the controller to interact with the data layer)
Case Service (This service class encapsulates the business logic related to case data, ensuring separation of concerns and providing a clean interface for the controller to interact with the data layer)
Chat Service (This service class encapsulates the business logic related to chat data, ensuring separation of concerns and providing a clean interface for the controller to interact with the data layer)

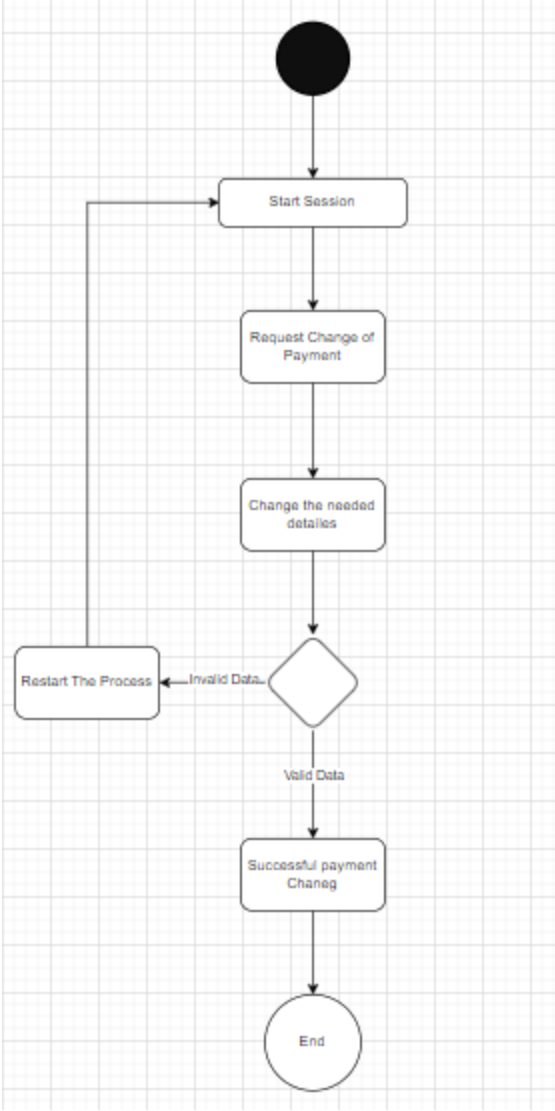

Vertical Container
Makes an API Call to
Single Page Application [container TS and React] Provides all of the functionality for the lawfirm client software through the web browser
Makes an API Call to
Chat Page [Container TS and React] Users has the ability to chat securly with his attorney on 24hours in a day(the attorney has the right not to answer every day)
Set Up Meeting Page [Container TS and React] Users has the right to set up a meeting with their attorneys to discuss things in person
Case Page [Container TS and React] Users has the right to see how their case is going
Create Account Page [Container TS and React] Allow users create accounts
Log In Page [Container TS and React] User should be logged in their account in order to access any information
Profile Page [Container TS and React] User should be logged in a be able to see and update their data
Attorney Page [Container TS and React] The client should be able to see the data about their attorney
Accounts Page [Container TS and React] The attorney should be able to view, access and update details he is responsible for
Upload Files [Container TS and React] User should be able to upload file like images, and videos securly
Payment Page [Container TS and React] User should be able to changes his billing information whenever they like to
Uses
Case Page Functionalities [Container Java & Spring] Responsible for all the functions related to cases, data:str for specific information, lawface, attorney and etc
Uses
Security Functionalities [Container Java & Spring] Provides functionalities related to signing up, logging and etc

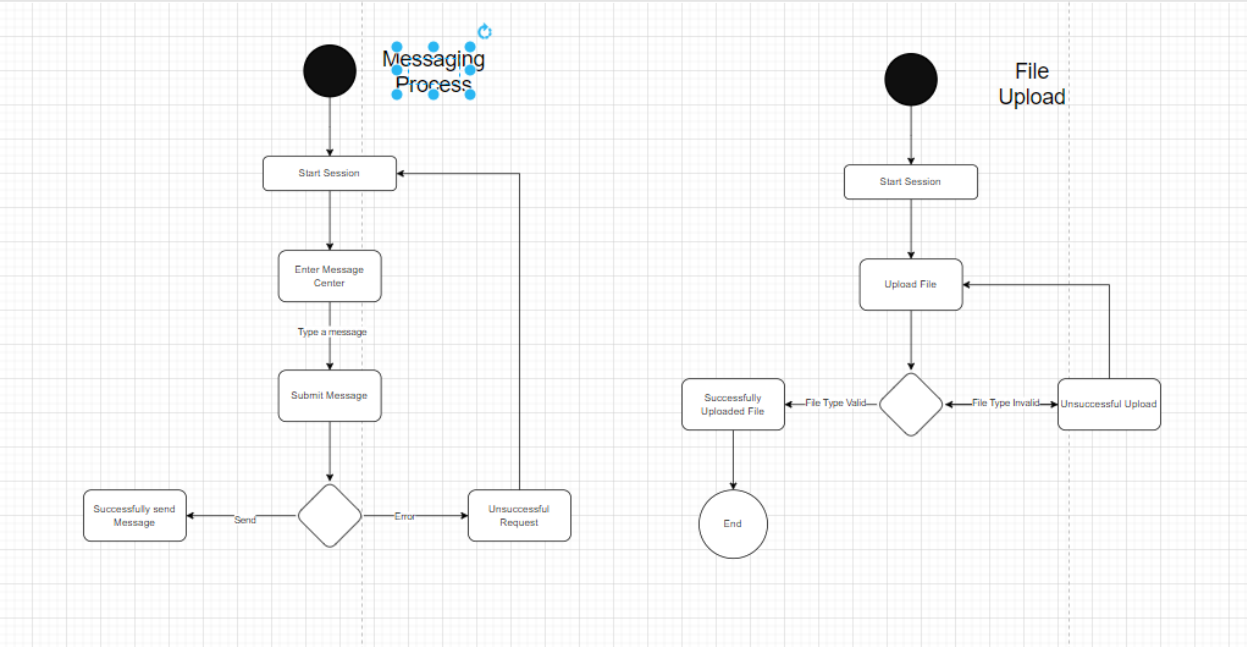

Container Diagram
Visit website using https
Web Application [container TS and React] Delivers the static content for the lawfirm clinet software
Single Page Application [container TS and React] Provides all of the functionality for the lawfirm client software through the web browser
API Application [container Java and Spring] Provides the business logic for the lawfirm client software through JSON/HTTPS API
Database [Container: MYSQL] Stores all the information needed for the lawfirm (Client data, Users authentication credentials, Cases informations, Files and etc)

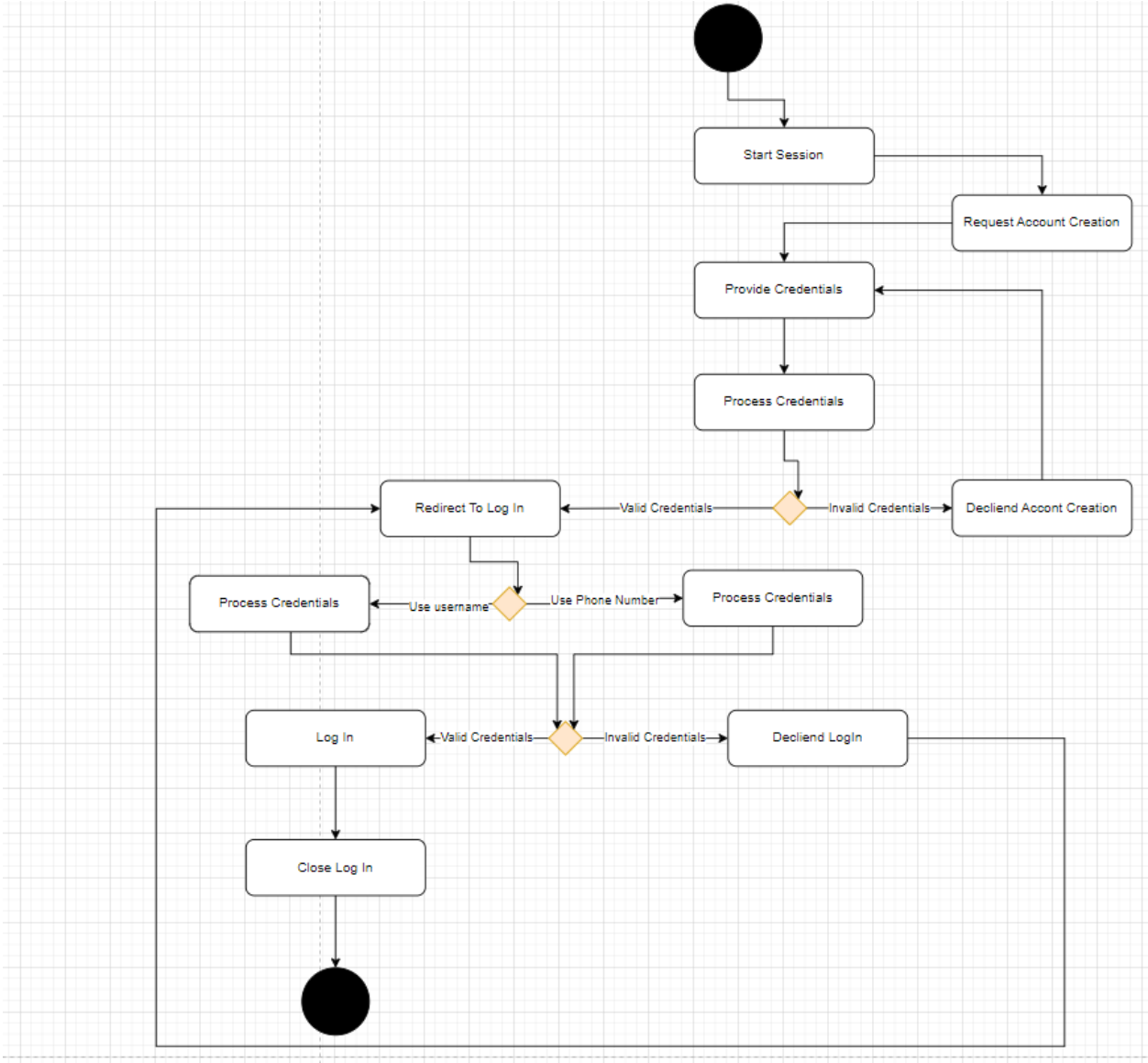

Context Diagram
Client
The lowest in the hierarchy. He is then to be informed and keep in constant touch with their attorney. He has the same access as the attorney but he just receives information and can upload useful files for the case
Case
A case is being assigned to a client and managed by the attorney
Lawbrik Website
Acts as a software for lawfirms to manage their daily activities, access (Accounts), cases, chats, doing case tracking)
Admin
The highest position. Most likely a manager in the lawfirms (has the right to do everything. Mainly creating accounts and assigning cases and attorneys)
Attorney
Attorney role has the attorneys from the lawfirms. Their role is to update their clients on the latest news for their cases, upload files, chat and etc.

5. Activity Diagrams

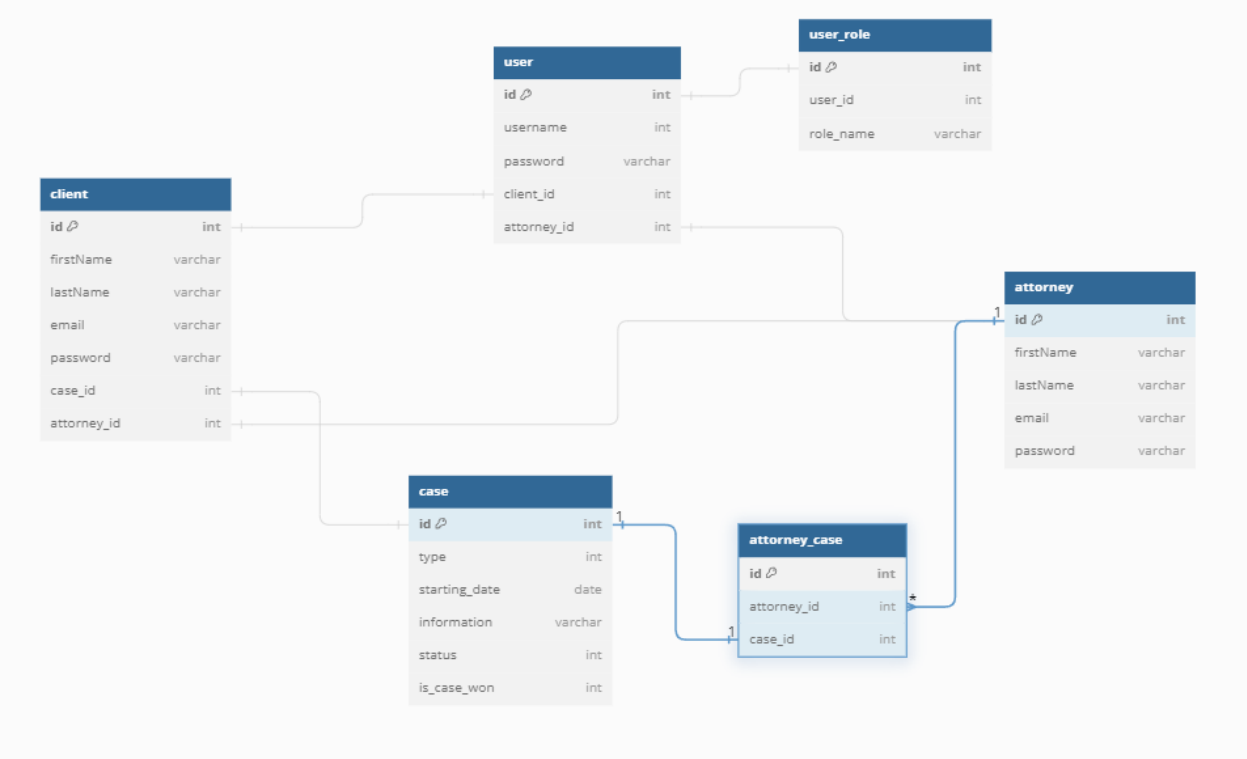



Start Session
Request Change of Payment
Change the needed detailes
Restart The Process
Invalid Data
Valid Data
Successful payment Chaneg
End


Messaging Process
Start Session
Enter Message Center
Type a message
Submit Message
Successfully send Message
Send
Error
Unsuccessful Request
File Upload
Start Session
Upload File
Successfully Uploaded File
File Type Valid
File Type Invalid
Unsuccessful Upload
End



6.Database Diagram



7. User Stories

1. As a legal practitioner, I want to have a dashboard where I can view all active cases, upcoming appointments, and important deadlines, so I can efficiently manage my workload.
    - Exception: In case of technical issues or server downtime, users should have access to a backup system or offline mode to view essential case information.
2. As a client, I want to be able to update my contact information and billing details securely through the software, to ensure accurate communication and billing.
    - Exception: If there's a failure in updating information due to system errors, users should receive a clear error message and alternative instructions for updating their details.
3. As a legal administrator, I want to generate customizable legal reports for clients in PDF format, including case summaries, billing statements, and legal documents, to provide transparent and detailed information.

- Exception: If there are any discrepancies or errors in generating reports, there should be an option to manually review and edit the report before finalizing and sending it to the client.

4. As a legal consultant, I want to record video updates for clients summarizing case progress and key developments, to maintain regular communication and provide personalized service.
    - Exception: If there are technical issues during the recording process, users should have the option to save drafts or re-record without losing the content already recorded.
5. As a client, I want to communicate securely with my legal team through a messaging center within the software, to discuss case details, ask questions, and receive updates.
    - Exception: If there's a breach in security or unauthorized access to the messaging center, users should be alerted immediately and provided with instructions on how to secure their communication.
6. As a legal practitioner, I want to schedule appointments with clients directly through the software, to streamline appointment management and ensure efficient use of time.
    - Exception: In case of scheduling conflicts or overlapping appointments, users should have the ability to reschedule or notify clients promptly.
7. As a client, I want access to a library of legal resources within the software, including articles, guides, and legal documents, to educate myself on relevant legal matters.
    - Exception: If there's any outdated or incorrect information in the resources, users should be able to report it for review and receive updates accordingly.
8. As a legal administrator, I want to track case progress and updates in real-time, to ensure timely communication and efficient resolution of legal matters.
    - Exception: If there's a delay in updating case progress due to technical issues or data synchronization problems, users should receive notifications about the delay and steps being taken to resolve it.

9. As a client, I want to receive notifications and reminders for upcoming appointments and deadlines, to stay informed and organized.
    - Exception: If there's a failure in sending notifications or reminders, users should have access to an alternative method of checking their appointments and deadlines, such as a calendar view within the software.
10. As a legal practitioner, I want to gather feedback from clients through the software, to assess client satisfaction and identify areas for improvement in service delivery.
    - Exception: If there's a technical issue preventing clients from providing feedback, there should be an alternative method provided for them to share their feedback, such as email or phone survey.

8. RESTful API Documentation

1. Endpoint: `/accounts`

- **Description**: Endpoint to manage clients.

1.1 `GET` - Retrieve Clients

- **Description**: Retrieves a list of clients.
- **Request**:
    - Method: `GET`
    - URL: `/accounts`

```
"clients": [
        {
            "id": 1,
            "firstName": "John",
            "lastName": "Doe",
            "email": "john.doe@example.com",
            "password": "password123",
            "caseId": null,
```

```
            "attorney": {
                "id": 1,
                "firstName": "Alex",
                "lastName": "A",
                "email": "agg@s",
                "password": "a",
```

    - Headers:
        - `Authorization`: Bearer <token>

- **Response**:
    - Status Code: `200 OK`
    - Body:

```
"clients": [
        {
            "id": 1,
            "firstName": "John",
            "lastName": "Doe",
            "email": "john.doe@example.com",
            "password": "password123",
            "caseId": null,
            "attorney": {
                "id": 1,
                "firstName": "Alex",
                "lastName": "A",
                "email": "agg@s",
                "password": "a",
```

1.2 `POST` - Create Resource

- **Description**: Creates a new resource.
- **Request**:
    - Method: `POST`

- URL: `/accounts`
- Headers:
  - `Content-Type`: `application/json`
  - `Authorization`: Bearer <token>
- Body:

```
{
  "firstName": "",
  "lastName": "",
  "email": "",
  "password": "",
  "caseId": 0,
  "attorneyId": 0,
  "role": ""
}
```

- **Response**:
  - Status Code: `201 Created`
  - Body:

```
{
  "id": "2",
  "firstName": "Stanislav",
  "lastName": "Nikolov",
  "email": "stasnikolov03@gmail.com",
  "password": "stas",
  "caseId": 0,
  "attorneyId": 2,
  "role": "CLIENT"
}
```

1.3 `GET` - Retrieve Resource by ID

- **Description**: Retrieves a single resource by its ID.
- **Request**:
  - Method: `GET`
  - URL: `/accounts/{id}`
  - Headers:
    - `Authorization`: Bearer
- **Response**:
  - Status Code: `200 OK`
  - Body:

```
{
        "id": 15,
    "firstName": "Alexa",
    "lastName": "A",
    "email": "adgg@s",
    "password": "a",
    "caseId": null,
 }
```

1.4 `PUT` - Update Resource

- **Description**: Updates an existing resource.
- **Request**:
  - Method: `PUT`
  - URL: `/accounts/{id}`
  - Headers:
    - `Content-Type`: `application/json`

        - `Authorization`: Bearer <token>
    - Body:

```
jsonCopy code
{
                        "id": 1,
            "firstName": "UpdateStas",
            "lastName": "Updated",
            "email": "john.doe@stas.com",
            "password": "password123",
            "caseId": null,
            "attorney": 2
}
```

- **Response**:
    - Status Code: `200 OK`
    - Body:

```
{
                        "id": 1,
            "firstName": "UpdateStas",
            "lastName": "Updated",
            "email": "john.doe@stas.com",
            "password": "password123",
            "caseId": null,
            "attorney": 2
}
```

1.5 `DELETE` - Delete Resource

- **Description**: Deletes a resource by its ID.
- **Request**:

- Method: `DELETE`
- URL: `/accounts{id}`
- Headers:
  - `Authorization`: Bearer <token>

- **Response**:
  - Status Code: `204 No Content`

2. Endpoint: `/login`

- **Description**: Endpoint to manage clients.

1.1 `POST` - Login

- **Description**: lLog in the application
- **Request**:
  - Method: `POST`
  - URL: `/login`
  - Headers:
    - `Content-Type`: `application/json`
    - `Authorization`: Bearer <token>
  - Body:

```
{
    "username": "adgg@s@lalwink.com",
      "password": "a"
}
```

- **Response**:
  - Status Code: `201 Created`
  - Body:

```
{
    "accessToken": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhZGdnQHNAbGI
    "userId": 15,
    "userRoles": "ATTORNEY"
}
```