**Medica software documentation**

**The technology**

System compatibility: windows, mac os, linux

**Software development tools**

For the realization of this project, the medica team to increase productivity and boost output certain development tools were used in the following process;

**System analysis**

Software used: star UML

**Design**

For wireframes: ADOBE XD

User Interface/User Experience: figma

**DEVELOPMENT:**

A series of different software were used by the different developers, including visual studio code, NetBeans ide

**Front end**: the user interface is built using java swing on the NetBeans ide

**Backend**: the server side of the application is built with java. The architecture of this software is being built based on a REST API client – server architecture, an architecture conceived and built and maintained by the medica team.

**THE ARCHITECTURE**

**database**

**Middleware**

**Api interface**

**Front end device**

*Fig 1. A simple REST API architecture*

the architecture follows a request/response relationship where the server-side receives a request for a resource from the client-side through an API (application programming interface) call through a URL where by the server checks the resource from the database and returns a response. Our teams choose a JSON format for data transmission due to its light weight increasing the efficiency of our application.

**The different segments of the architecture**

**Database**: using enterprise level software best for a young startup base on the research of the project manager, medica uses PostgreSQL and MySQL for database management software(DBMS) during development

**Framework and functions**

Database management:

* Import settings
* Create a connect function: connecting to the DBMS.
* Object queries general
* Queries with filters.

Tables:

* Import functions from database management.
* Create table: for creating different tables in our database.
* Hash password: for security reasons for users of the system.

**Middle ware**: the middleware in other words what we call resource manager. Basically receive and respond to requests from the api interface.

**API gateway**: The API gateway or interface built in rest API form where each resource is handled as a single entity. Receives and sends data through http POST(create), GET(read), PUT(update), DELETE(delete) by creating URL paths. Basic CRUD operations

Overview of functions

Views:

* Route REST api request and response.
* Get parameters from JSON and pass it to queries or sql statements.
* Create, read, update and delete from tables.
* Create notification
* Authenticate
* Jwt tokens (identify users).
* Cookies (unique user data).
* Log in.
* Log out by deleting cookies.
* Update account by passing cookies.
* Creating a payload for a user: bundles user data and encode into a unique token.

Settings:

* Set paths for routing.
* Settings for hashing algorithms
* Connection to database
* Connection to resources(assets)
* Import urls.

Urls(links):

* Route to settings
* Import views

Test:

* Pen testing (for security) or unit test algorithms.

Communication: for our efficient work flow we use GitHub, emails and WhatsApp