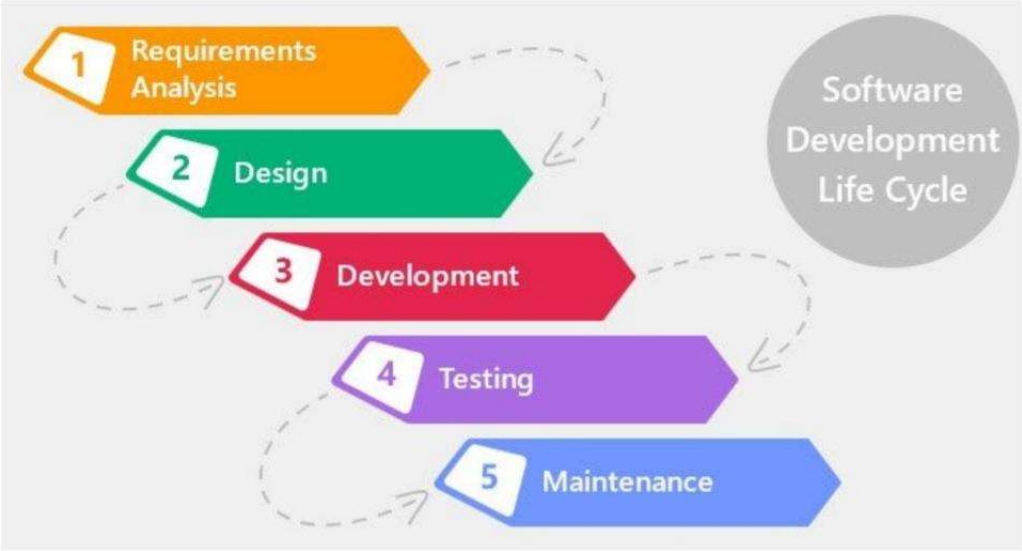
POLITECNICO
MILANO 1863

2020



Design Document

VERSION 2.0

June 08, 2020

Deliverable: DesignDoc

Title: Design Document

Authors:

Wafi Mahdi Eldoud Mahdi

Effort: %25 (Application Functions, HTML Templates, Modules)

MohamedElmustafa Omer

Effort: %25 (Application Functions, HTML Templates, Modules)

Abubakr Albashir

Effort: %25 (Database Design, HTML Templates, Aim, Scope)

Faris Elsmani

Effort: 25% (Database Design, HTML Templates, Aim, Scope)

Date: 08 June 2020

Table of contents:

4. Application Functions:

5. Database Design:

1.Introduction:

1.1 Aim:

The design document aims to guide through the various functions carried out in the web application created to access information from the HATUA research.

HATUA research is introduced and well-defined in the RASD, which will be delivered alongside the Design and Test Plan Document, and Software Release Document, and it can be reached from this link: Github

This document (Design and Test Plan Document) will provide details regarding the web application's modules, templates, functions, database and testing plan.

1.2. Scope:

This document is concerned with showcasing technical details regarding the web application.
The Application will allow casual users to search for information from a database derived from the HATUA research, which will include the following patients' information:

1. Location.
2. Interview date.
3. Gender.
4. Marital status.
5. Religion.
6. Education level.
7. Ethnic group.
8. Preconceived notions about a certain disease.
9. Time from first symptoms.
10. First doctor visit.

The inquired information will be displayed in several visualization methods such as pie charts and bar charts. The search function in the application will allow users to draw comparisons between the research subjects' reaction to symptoms with regards to their level of education, tribe and creed. Moreover, it will also allow users who register as members to interact with each other via a built-in message board.

2. Modules:

2.1 Flask is a microframework that allows us to create servers in Python. From flask we import: Flask, render_template, redirect, request, url_for, session, g., ...etc.These libraries/functions help us:

- Render_template: Render our html pages.
- Request: is a python HTTP library. That is useful on many fronts in our work.
- Session: Helps us track the session data, comes handy when the member wants to: Logout or Add a comment.
- Redirect: Helps us redirect users to other pages (mainly the homepage), usually used with (url_for).
- Flash(): it is used to generate informative messages and keep the user updated of every development, for example error messages, as can be seen in the web-application
- g: a global variable that allows us to interact with the html templates.
- Bokeh: is an interactive data visualization library to visualize data and create interactive plots and applications running on a web browser.

2.2 JSON is an open standard file format, and data interchange format, that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and array data types (or any other serializable value).
From JSON we import:response, raw-data, json.loads and pdf.json_normalize.

2.3 Psycopg2: Is a PostgreSQL database adapter for the Python programming language, it is used to connect python with the database using the connect() function. Before making a local connection with the database, a cleanup procedure is carried to delete any tables with similar names from the database.

2.4 werkzeug.security: Is a module used to store and secure passwords while keeping them encrypted, from this model we import:

- Generate_password_hash: This allows us to store hashed values of passwords in the database, instead of storing them in clear.
- Check_password_hash (stored_password,password): This allows us to check whether the password provided matched the hashed_value that is stored in the database.

2.5 werkzeug.exceptions: Is a module that adapts several Python exceptions which can be implemented from inside the interface. from werkzeug.exceptions we import:

- Abort(): This function is used to terminate the process if a certain condition is met, it returns the chosen HTTP error.

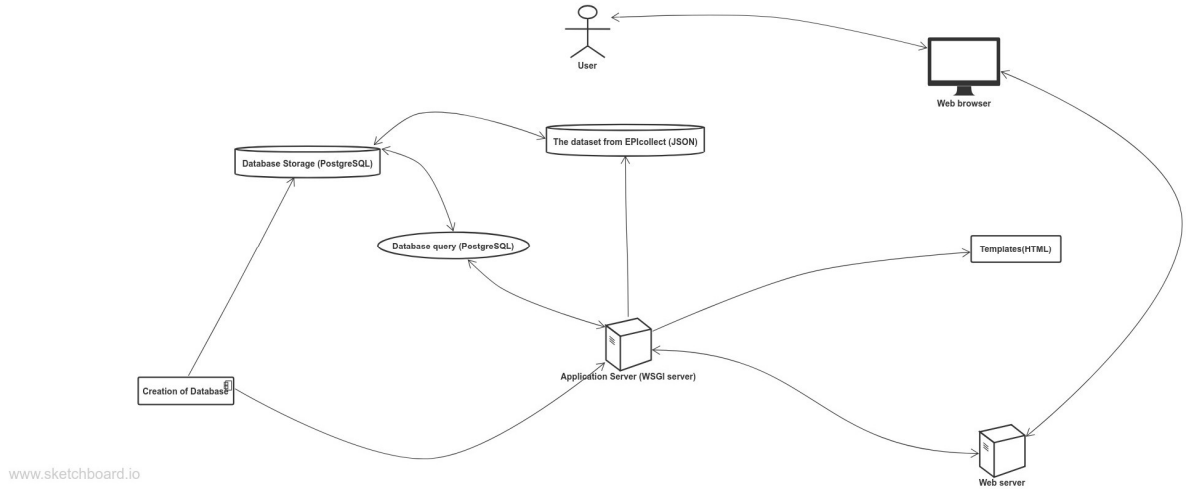
2.6 The Dataset in UG HATUA PHASE 2 OUTPATIENTS will be retrieved from the internet using JSON, we will use the request library to send requests to the REST API service of the

EPICOLLECT platform. The JSON library will be used to encode the raw response text into a JSON variable and then into a Pandas DataFrame.
Finally, the Pandas DataFrame will be converted into a GeoDataframe. Our dataset will be stored as a table in the database (data_table).

The data_table is created according to the specifications explained in the Database section alongside two other tables.

This Dataset will be available to all users of the Software.

2.7: Web Application Diagram:



2.8 CSS:
The main parts of the CSS are built on a template called Medisen by Colorlib, June 2019, Licence: CC BY 3.0.

3. HTML Templates:

3.1 Index.html: This page is accessed from the ('/Home') function and contains the unified background and design for the web application pages. It also contains the application's Name (should always be visible in the application). With links to a number of pages, such as Register, Login and Search pages. If the user is already logged in it shows a message that says: Logged in as username and provides a link to logout.

3.2 Register.html: This page is accessed from the ('/Register') function. It asks the user to provide personal information such as username, password, email and age. If the user doesn't provide a username, password, email or age the system returns "please fill this field".
If the username provided by the user doesn't match any other username in the database then the system allows the registration. The input will be stored in the (sys_table) in the database, a hashed value of the password will be stored instead of the real password. Also, it redirects the user to the Login page (Login.html). If the username provided by the user matches another one in the (sys_table) the system returns an error message that says "Username already used! try another one!"
In order to direct users to choose diverse usernames; the system does not accept usernames that consist only of digits (e.g. 123456), and redirects the user to (Not acceptable) HTTP error page.

3.3 Login.html: This page is accessed from the ('/Login') function. It asks the user to input login information (username and password) and provides submission confirmation. If the user's login information matches the ones in the (sys_table) then the user is allowed to log into the system, it redirects the user to the home page (Home.html), the Add Comment (AddComment.html), Comments (Comments.html), Visualizing page (ViewMap.html), CustomizePlot (CustomizePlot.html) and search (Search.html) pages deactivated.activated.
if the username provided by the user doesn't match the ones in the (sys_table) in the database then it returns an error message that says "Login failed! Wrong username" and leaves the user on the same page, and if the provided username is correct, but the password entered does not match the username it returns an error message that says "Login failed! Wrong password". If the user provides the correct username and password but doesn't check the check button, the system shows a message "please check this box if you want to proceed".

3.4 Search.html: This page is accessed from the ('/Search) function. It asks the user to input an inquiry regarding the related attribute field from the attribute data in the dataset. If the inquiry is compatible with the database, it retrieves all the records related to the inquiry from the data_table in the database and shows in the page. If the user enters incompatible with the

data set, the system will return the Search.html page blank. If the user requests to search with empty search, the system will return a message " please fill the search". If the user entered “all” as input for the search, then the system will show all of the dataset records in the Search.html page.

3.5 ViewMap.html: This page is accessed from the (‘/ViewMap) function. It takes the member to an imported tiled map from OpenStreetMaps which includes the patients’ locations database information stored in the (data_table). The user then can access information displayed in the map based on location as a pop-up with two attributes of patients (Gender and Religion).

3.6 AddComment.html: This page is accessed from the (‘/Addcomment’) function, it asks the member to add a comment if he wishes. The input will be stored in the comment_table in the database. The comment (comment) will be added in the comment_table in the database with the username (userid) and also every comment will be assigned a unique identifier (comment_id). The user after adding a new comment will be directed to the Comments.html page.

3.7 Comments.html: This page is accessed only if the member is logged in (userid in session). It is accessed from the (‘/Comments’) function, it retrieves from the comment_table in the database and shows all of the comments done by all members. The comment (comment) will be shown along with the username (userid), and the unique identifier (comment_id).

3.8 CustomizePlot.html: This page is accessed from the (‘/CustomizePlot’) function, which the member can reach by link from the Index.html page. It asks the user to choose the way the user wants to visualize the plot of a certain type of plot(line graph,circles, bar chart) for the selected field from the dataset. The visualizing ways are requested from the provided buttons and then the chosen visualizing method will be shown in the page.

3.9 Backend Developers.html: This page is accessed from the (‘/Backend_Developers) function. It allows the user to render the template of Backend Developers that contains information about the developers of the backend part of this system (python functions and modules). This template is accessible for both users and members through the main page of the web system.

3.10 Frontend Developers.html: This page is accessed from the (‘/Frontend_Developers) function. It allows the user to render the template of Frontend Developers that contains information about the developers of the frontend part of this system (HTML, CSS). This template is accessible for both users and members through the main page of the web system.

3.11 Contact.html: This page is accessed from the (‘/contact) function. It allows the user to render the template of contact that contains information about the contact details of the HATUA organization that represents the stakeholder of this system. This template is accessible for both users and members through the main page of the web system.

3.12 About.html: This page is accessed from the (‘/about) function. It allows the user to render the template that contains information and more details of the HATUA organization that represents the stakeholder of this system. This template is accessible for both users and members through the main page of the web system.

4. Application functions:

4.1 User opens the webpage:

@app.route (‘/ ’)
@app.route (‘/Home’)
@app.route (‘/home’)
This function renders the Homepage of the web application (Index.html)

4.2 User/Member searches for data:

@app.route('/search', methods=('GET','POST'))
@app.route('/Search', methods=('GET','POST'))
*The function must answer to two HTTP requests:
I) If request is POST:
1. Acquire the search item from the member/user.
2. Cross reference search item with items in database.
3. Retrieve database attributes relevant to a search item.
4. If a search item is not compatible with the dataset, return message "Not Found".
II) On GET, it redirects the user to the Search page.

4.3 Member registration:

@app.route(‘/Register’, methods=(‘GET’,’POST’)).
@app.route(‘/register’, methods=(‘GET’,’POST’)).
*The function must answer to two HTTP requests:
I) If request is POST:
1. Get the information sent by the Member-to-be (username and password)
2. Check if they have been correctly inserted (Username and password filled and follow predefined syntax)
3. Check if the username exists in the database
4. If yes, send an error message to the user. “Username already exists, login if it’s your account”
5. If not, store the information into the database and return a message that says: “Successful registration!”
6. Redirect the user to the login page
II) On GET, it redirects the user on the registration page.

4.4 Member Login:

@app.route(‘/Login’, methods=(‘GET’,’POST’))
@app.route(‘/login’, methods=(‘GET’,’POST’))
*The function must answer to two HTTP requests:
I) If request is POST:
1. Get the information sent by the Member (username and password)
2. Check if the username exists in the database.
3. If not, send an error message to the member. “Error! wrong username”
4. If yes, check if the password exists in the database
5. If not, send an error message to the member.“Error! wrong password”

6. If yes, store the Member_id in the session variable.

II) On GET, it redirects the user on the login page.

4.5 Member Adds a Comment:

@app.route('/AddComment', methods=('GET','POST'))

@app.route('/addcomment', methods=('GET','POST'))

*The function must answer to two HTTP requests:

I) If request is POST :

1. Check if the user is already logged in.
2. If not, show an error message "only logged in users can add comment"
3. If yes, acquire the comment from the member.
4. Store the comment in the database.
5. Visualize the comment alongside the user_id
6. Redirect the member to the comment page..

II) On GET, it redirects the user to the add comment page.

4.6 Member logout:

@app.route('/Logout')

@app.route('/logout')

The function must:

1. Clean the session variables.
2. Redirect the user on the application homepage.

4.7 Member/User conducts map-based search:

@app.route('/ViewMap')

@app.route('/viewmap')

The function must:

- Redirect the user/member to the imported Open Street Maps with the contribution of CartoDB.
- Render the basemap from Open Street Maps with the contribution of CartoDB.
- Shows the locations of dataset entries.

4.8 Member/User renders Backend Developers:

@app.route('/Backend_Developers')

The function must:

- Redirect the user/member to the Backend Developers page.
- Renders the Backend Developers.html.

4.9 Member/User renders Frontend Developers:

@app.route('/Frontend_Developers')

The function must:

- Redirect the user/member to the Frontend Developers page.
- Renders the Frontend Developers.html.

4.10 Member/User renders Contact:
@app.route('/contact')
@app.route('/Contact')
The function must:

- Redirect the user/member to the contact page.
- Renders the contact.html.

4.11 Member/User renders about:
@app.route('/about')
@app.route('/About')
The function must:

- Redirect the user/member to the about page.
- Renders the about2.html.

4.12 conn_db():
This function checks whether there's a Database connection, if not it creates a Database connection, and establishes the connection in any case (Returns g.db).

4.13 enddb_conn():
This function ends the database connection if it exits.

4.14 mysession():
This function gets the userid from the database and saves it in the session.
Determining whether a user is logged in or not defines different routes for the web application to go in, and it's key in the logout function.

4.15 Comments:
@app.route('/Comments')
@app.route('/comments')
The function must:

- Create a Database connection.
- Retrieve all the comments from the database and show them on the web-page alongside the username of the members that added the comments and the comment_id.

5. Database Design:

The database management system: The database management system of the web application is PostgreSQL 10 DBMS that works on the local machine.

5.1 Database Schema:

The database schema consists of three tables:

1. The sys_table: the table contains the system users' information needed for registration and login:(user ID, usernames, passwords and age).

2. The comment_table: the table contains the user ID, the users' comments and the time of writing or editing the comments.
3. The Data_table: the table contains the dataset of the patients which are the scope of the system including their locations and other attributes related to them.

The connection between the first two tables is done by the foreign key field: userid.

5.2 Tables Design:

5.2.1 The sys_table:

Field Name	Datatype	Mandatory/ Optional	Note
userid	Serial		Primary key
username	VARCHAR(255)	Mandatory	NOT NULL, unique
password	VARCHAR(255)	Mandatory	NOT NULL
email	VARCHAR(255)	Optional	
age	INTEGER	Optional	

5.2.2 The comment_table:

Field Name	Datatype	Mandatory/ Optional	Note
comment_id	INTEGER		Serial primary key
userid	INTEGER	Mandatory	NOT NULL,UNIQUE
timestamp	FLOAT	Mandatory	
comment	VARCHAR(500)	Mandatory	NOT NULL

The userid field is used as a foreign key to connect the first two tables.

5.2.3The data_table:

Field Name	Datatype	Optional/Mandatory	Notes
patient_id	INTEGER		NOT NULL,UNIQUE
long	POINT	Mandatory	
lat	POINT	Mandatory	
Interview_date	DATE	Optional	
Gender	VARCHAR(255)	Optional	
Martial_Status	VARCHAR(255)	Optional	
Religion	VARCHAR(255)	Optional	
Education_level	VARCHAR(255)	Optional	
time_from_first_symptom	DATE	Optional	
disease_stigma	VARCHAR(255)	Optional	

6.Test Plan:

6.1 Overview:

This chapter aims to subject a number of the web application's functions to testing in order to oversee possible missteps and incompatibilities. It also aims to determine whether the functions are running as intended in the design plan.

6.2 Approach:

The testing process will implement the manual method of the Unit Testing approach, which aims to identify possible actions from the user and the corresponding expectation for each of these actions.

6.3 Scope:

The web application function that will undergo the testing process are:

i) Register.
ii) Login.
iii) Search.
iv) Add Comments.

6.4 Test cases:

6.4.1 Register test cases

i) Register 1.0:

This test case will assure that all users of the software will be able to register successfully to the software database by importing all required data.

The user reach the register page in the software and then do the following steps:

1. The user enters a new username and password in the username and password fields.
2. The user enters his email address and his age in the email and age field.
3. The user clicks the register button.

The expected result : the software saves the registration data in the database and redirects the user to the login page.

ii) Register 2.0:

This test case will assure that all users of the software will not be able to register to the software database by importing only the username and password.

The user reach the register page in the software and then do the following steps:

1. The user enters a new username and password in the username and password fields.
2. The user clicks the register button.

The expected result : the software returns an error message that says “Please fill out this field.” and points to the Email field.

iii) Register 3.0:

This test case will assure that the user will not be able to register without entering a username.

The user reach the register page in the software and then do the following steps:
1. The user enters a new password in the password field.
2. The user enters his email address and his age in the email and age field.
3. The user clicks the register button.

The expected result : the software shows an error message to the user “Username is required”.

iv) Register 4.0:

This test case will assure that the user will not be able to register without entering a password.

The user reach the register page in the software and then do the following steps:
1. The user enters a new username in the username field.
2. The user enters his email address and his age in the email and age field.
3. The user clicks the register button.

The expected result : the software shows an error message to the user “Password is required”.

v) Register 5.0:

This test case will assure that the user will not be able to register without entering a unique username.

The user reach the register page in the software and then do the following steps:

1. The user enters a username already in the database and a password in the username and password fields.
2. The user enters his email address and his age in the email and age field.
3. The user clicks the register button.

The expected result : the software shows an error message to the user “Username already used! try another one”.

6.4.2 Login test cases:

i) Login 1.0:

This test case will assure that all Members of the software who are already registered to the database will be able to login successfully by importing his/her username and password.

The user reach the login page in the software and then do the following steps:

1. The user enters his/her username and passwords in the required field.
2. The user clicks the login button.

The expected result : the software redirects the user to the home page and shows a message the “Logged in as <username>, Welcome!’.

ii) Login 2.0:

This test case will assure that all the Members of the software will be alerted when each of them adds in a username not compatible with the one stored in the database.

The user reach the login page in the software and then do the following steps:

1. The user enters his/her wrong username in the required field.
2. The user enters the correct password in the required field.
3. The user clicks the login button.

The expected result : the software alerts the user that he/she has entered the wrong username.

iii) Login 3.0:

This test case will assure that all the Members of the software will be alerted when each of them adds in a password not compatible with the one stored in the database.

The user reach the login page in the software and then do the following steps:

1. The user enters his/her the correct username in the required field.
2. The user enters the wrong password in the required field.
3. The user clicks the login button.

The expected result : the software alerts the user that he/she has entered the wrong password.

iv) Login 4.0:

This test case will assure that all the users of the software will be alerted when each of them clicks the login button while leaving one of the required fields unfilled.

The user reach the login page in the software and then do the following steps:

1. The user clicks the login button without filling the username or password fields.

The expected result : the software alerts the user that he/she has must fill the required fields.

6.4.3 Search test cases:

i) Search 1.0:

The test case will assure the Users of the software will be able to visualize search elements based on a submitted inquiry .

The user reach the search page in the software and then do the following steps:

1. The user will submit an inquiry compatible with database elements in the search bar.
2. The user confirms the inquiry by clicking in the submit button.

Expected result: The software will display database elements compatible with the search inquiry.

ii) Search 2.0:

The test case will assure the Users of the software will receive a message informing them that their inquiry is not compatible with any database elements.

The user reach the search page in the software and then do the following steps:

1. The user will submit an inquiry incompatible with database elements in the search bar.
2. The user confirms the inquiry by clicking in the submit button.

Expected result: The software will take no action and will remain on the search page.

iii) Search 3.0:

This test case will assure that all the users of the software will be alerted when each of them clicks the search button while leaving the search bar unfilled.

The user reach the search page in the software and then do the following steps:

1. The user clicks the search button with the search bar unfilled.

Expected results: The Software alerts the user that the search bar is empty.

6.4.4. Add Comment test cases:

i) Add Comment 1.0:

This test case will assure that all the Members of the software will be able to leave a comment on the application webpage.

The user reach the AddComment page in the software and then do the following steps:

1. The Member inputs a comment in the designated input box.
2. The Member confirms his/her entry by clicking the post button.

Expected results: The software saves the comment in the database and redirects the Member to the comments page.

ii) Add Comment 2.0:

This test case will assure that all the users of the software will be alerted if he leaves the input box empty.

The user reach the AddComment page in the software and then do the following steps:

1. The Member does not input a comment in the designated input box.
2. The Member clicks on the post button.

Expected results: The Software alerts the Member that the input box is empty.

iii) Add comment 3.0:

This test case will assure that only the members of the software who are already registered to the database can contribute to the website by adding comments.

The user open the home page of the software and then do the following steps:

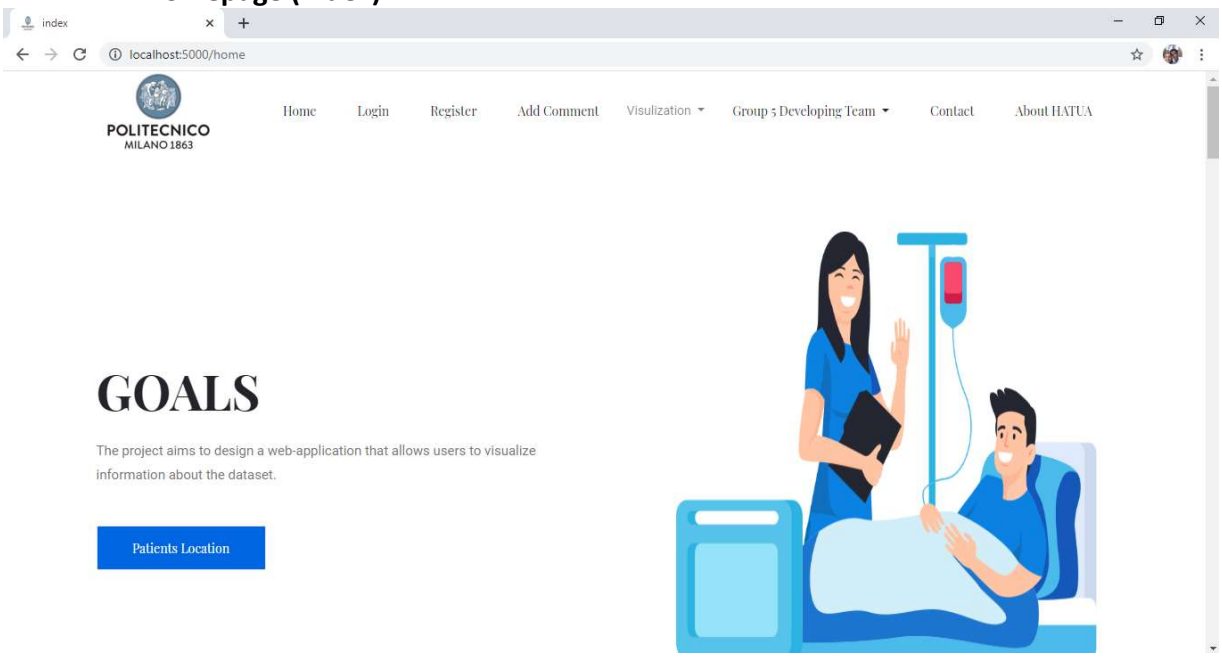
1. The user clicks on the Add Comment button.

Expected results: The Software shows a message "Only logged in users can add comments!''.

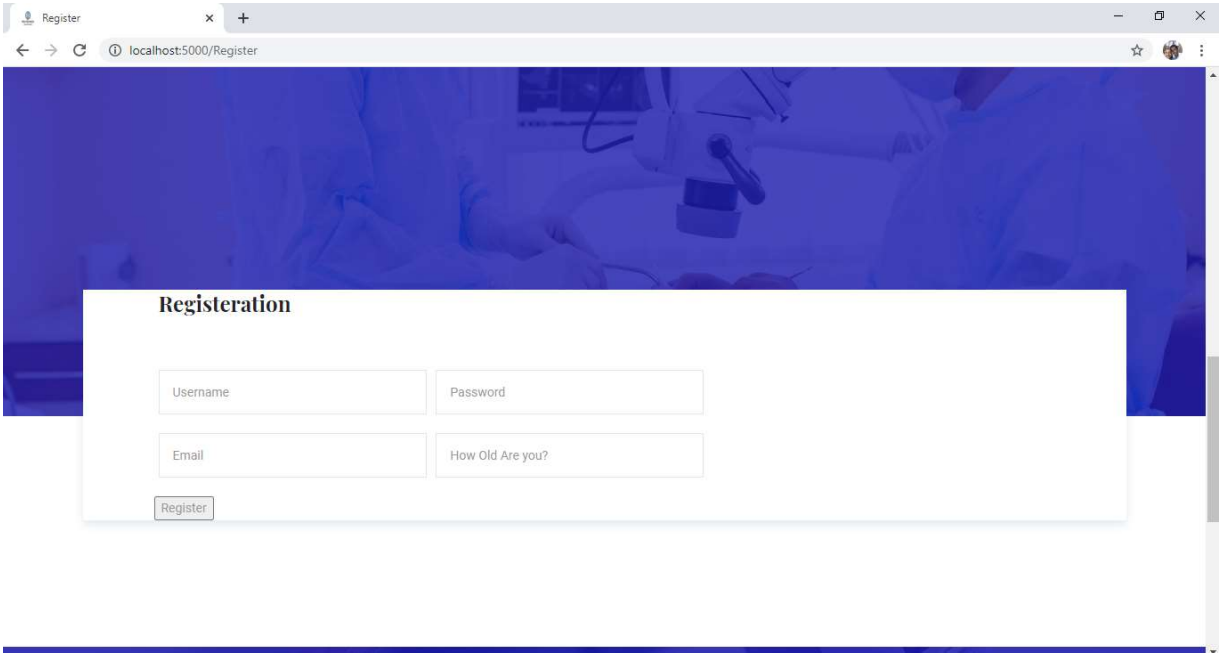
7. Software Interfaces:

The following is a representation of the the web application's pages layout

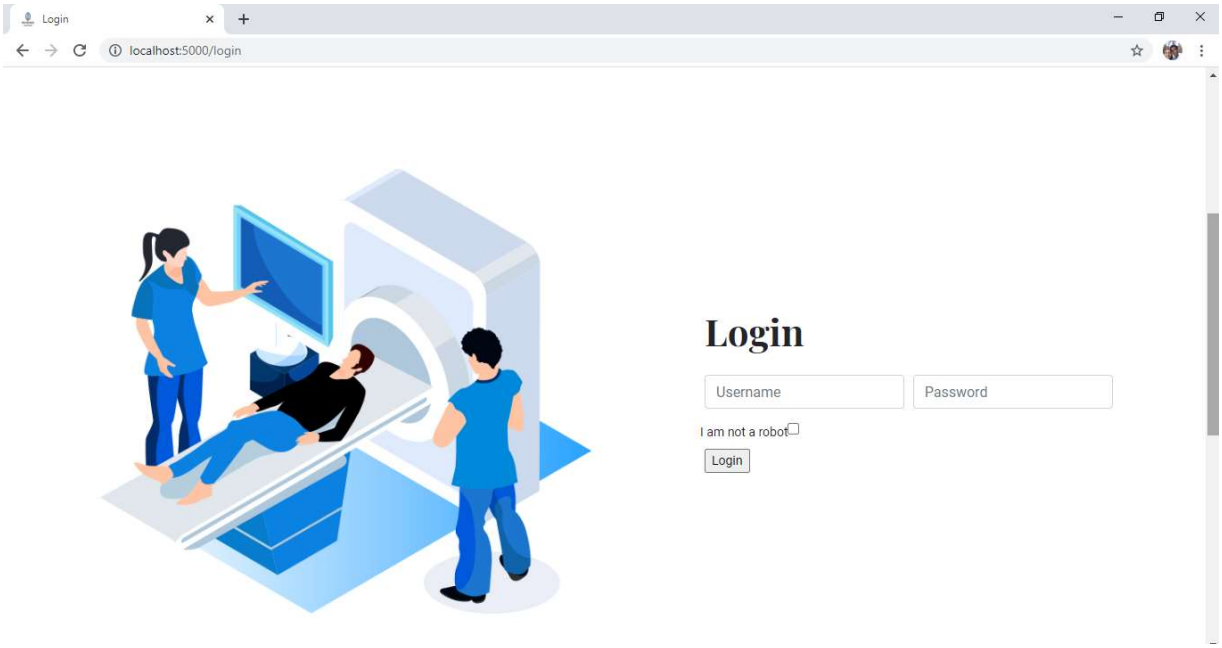
7.1 Homepage (Index):



7.2 Register:



7.3 Login:


Login
localhost:5000/login
Login
Username
Password
I am not a robot
Login