# PROFESSIONAL TRAINING REPORT

**At**

**Sathyabama Institute of Science and Technology (Deemed to be University)**

Submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering

By

## Bibiana Magdelene

**REG. NO. 39110164**

****

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**SCHOOL OF COMPUTING**

**SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY**

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,**

**CHENNAI – 600119, TAMILNADU**

**NOVEMBER 2021**

|  |  |  |
| --- | --- | --- |
|  | **SATHYABAMA**  **INSTITUTE OF SCIENCE AND TECHNOLOGY** (DEEMED TO BE UNIVERSITY) **Accredited with Grade “A” by NAAC**  (Established under Section 3 of UGC Act, 1956)  JEPPIAAR NAGAR, RAJIV GANDHI SALAI  CHENNAI– 600119  [**www.sathyabama.ac.in**](http://www.sathyabama.ac.in) |  |

# DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**BONAFIDE CERTIFICATE**

This is to certify that this Project Report is the bonafide work of **D. BIBIANA MAGDELENE (Reg. No: 3911064)** who carried out the project entitled “**Medical equipment supply system**” under my supervision from March 2022 to April 2022.

## Internal Guide

## Mrs.Nafees Muneera M.E.,M.B.A.,PH.D

**Head of the Department**

**Lakshmanan Sir**

**Vigneshwari Mam**



## Submitted for Viva voce Examination held on

**InternalExaminer ExternalExaminer**

**DECLARATION**

I, **BIBIANA MAGDELENE.D** hereby declare that the project report entitled **Medical equipment supply system** done by me under the guidance of **Mrs. M. Nafees Muneera, M.E.,M.B.A.,PH.D** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering.

## DATE: Bibiana Magdelene.D

**PLACE: SIGNATURE OF THECANDIDATE**

**ACKNOWLEDGEMENT**

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D**, **Dean**, School of Computing, **Dr. S. Vigneshwari, M.E., Ph.D. and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department** of **Computer Science and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Mrs.Nafees Muneera,** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project

**TRAINING CERTIFICATE**

# ABSTRACT

# Medical Devices are considered a fundamental component of Health Systems; the benefits they can provide continue to increase as they're essential to prevent, diagnose, treat and rehabilitate illnesses and diseases in a safe and effective way.

A medical device can be any instrument, apparatus, implement, machine, appliance, implant, reagent for in vitro use, software, material or other similar or related article, intended by the manufacturer to be used, alone or in combination for a medical purpose.

The Medical equipment supply management system lets people to sell and donate equipments through the website. It will give the details of equipments to those who need it. The hospitals can take the number from the website and contact them directly to get all their equipments delivered. Those who wish to donate or sell used medical equipments can create posts and will be contacted from buyers through this website. People who cannot afford these equipments can look for used equipments.

# From needles and syringes to gloves, masks and flu vaccine, people can sell or buy the medical supplies they need to help keep your patients healthy. This website will find the equipments you need in the right time and also helps to reduce cost for those who need it.

|  |  |  |
| --- | --- | --- |
|  | **TABLE OF CONTENTS** |  |
| **CHAPTER No.** | **TITLE** | **PAGE.NO** |
|  | ABSTRACT | i |
|  | LIST OF FIGURES | iv |
|  | LIST OF TABLES | iv |
| **1** | **INTRODUCTION** | **1** |
|  | 1.1 What is the Website used for | 1 |
|  | 1.2 How does the Website Work | 1 |
| **2.** | **AIM AND SCOPE OF THE PRESENT INVESTIGATION** | **2** |
|  | 2.1 Aim of the Project | 2 |
|  | 2.2 Scope | 2 |
|  | 2.3 Development Enviroment Software | 3 |

2.3.1 Atom Editor 3

2.3.2 Python 3

2.3.3 Hardware Requirements 4

2.4 Operational Environment 4

**3 EXPERIMENTAL OR METERIALS AND**

**METHODS,ALGORITHMS USED. 5**

3.1 Web Framework 5

3.2 Framework-Flask 5

3.3 Flask-SQLAlchemy 6

3.4 Critical Elements 6

3.5 Routes and View functions 7

3.6 HTTP Methods 8

3.7 Deploying Python Flask Framework

3.8 Importing Modules 10

3.8.1 OS 10

3.8.2 Flask 11

3.8.3 SQLAlchemy 11

3.8.4 Migrate 11

3.8.5 Flask-Login 12

3.8.6 Werkzeug-Security 12

3.8.7 UserMixin 13

3.8.8 Date and Time 13

**4 RESULTS AND DISCUSSION ,**

**PERFORMANCE ANALYSIS. 14**

4.1 Outcomes 14

**5 SUMMARY AND CONCLSUIONS 20**

5.1 Sample Code 21

5.2 Reference 26**LIST OF FIGURES**

## FIGURENO. FIGURENAME PAGENO.

3.1 Basic Flask App 7

3.2 Main Functions 7

3.3 Modules 10

**LIST OF TABLES**

**FIGURENO. FIGURENAME PAGENO**

2.1 **Table for operation environment 4**

**Chapter 1**

1. **INTRODUCTION**

The Medical Device market has grown fast; following technological trends in healthcare to improve people's welfare, there are currently more than 10,000 different kinds of medical devices around the world. Health Authorities face an overwhelming variety of options with regard to increasingly sophisticated, technologically complex and new medical devices. In addition to having more options, authorities must also consider the technological complexity of the devices; more global and competitive markets; increases in the marketing of used and refurbished equipment; donation of devices; reuse of single-use devices; ever-increasing use of devices in physician's offices and at the household level; patients with greater access to information. Access to good quality, affordable, and appropriate health products is indispensable to advance universal health coverage, address health emergencies, and promote healthier populations.

* 1. ***What is the website used for?***

The website is an intermediate for buyers who wish to buy medical equipments newly or used ones and for sellers who want to sell these equipments. Many people who can’t afford to buy new equipments are struggling without proper health care. The website is most useful to hospitals, health care centers, and even patients who stay at home.

* 1. **How does the website work?**

People who want to sell equipments can register in the website and log in to their accounts. They can then create posts about their equipments and give description about it. All the posts will be available in the home page. The buyer can choose it and get the information about the equipment they need and they can get it delivered.

**Chapter 2**

**2.AIM AND SCOPE OF THE PRESENT INVESTIGATION:**

***2.1 Aim of the project***

The project aims to help people get their equipments right on time. Many people who suffered covid died without proper equipments. These lives can be saved if everyone can get the equipments on time.

***2.2 Scope***

Today, there are an estimated 2 million different kinds of medical devices on the world market, categorized into more than 7000 generic devices groups.

Without medical devices, common medical procedures – from bandaging a sprained ankle, to diagnosing HIV/AIDS, implanting an artificial hip or any surgical intervention – would not be possible. Medical devices are used in many diverse settings, for example, by laypersons at home, by paramedical staff and clinicians in remote clinics, by opticians and dentists and by health-care professionals in advanced medical facilities, for prevention and screening and in palliative care. Such health technologies are used to diagnose illness, to monitor treatments, to assist disabled people and to intervene and treat illnesses, both acute and chronic.

The demand for equipments will keep increasing in the future and this web application will help balance and make the equipments be available for all hospitals and patients.

***2.3 Development Environment Software***

**2.3.1 Atom Editor**

Atom is a [free and open-source](https://en.wikipedia.org/wiki/Free_and_open-source_software) [text](https://en.wikipedia.org/wiki/Text_editor) and [source code editor](https://en.wikipedia.org/wiki/Source_code_editor) for [macOS](https://en.wikipedia.org/wiki/MacOS), [Linux](https://en.wikipedia.org/wiki/Linux), and [Microsoft Windows](https://en.wikipedia.org/wiki/Microsoft_Windows) with support for [plug-ins](https://en.wikipedia.org/wiki/Plug-in_(computing)) written in [JavaScript](https://en.wikipedia.org/wiki/JavaScript), and embedded [Git Control](https://en.wikipedia.org/wiki/Git). Developed by [GitHub](https://en.wikipedia.org/wiki/GitHub), Atom is a [desktop application](https://en.wikipedia.org/wiki/Application_software) built using [web technologies](https://en.wikipedia.org/wiki/World_Wide_Web). Most of the extending packages have free software licenses and are community-built and maintained. Atom is based on [Electron](https://en.wikipedia.org/wiki/Electron_(software_framework)) (formerly known as Atom Shell), a [framework](https://en.wikipedia.org/wiki/Software_framework) that enables cross-platform desktop applications using [Chromium](https://en.wikipedia.org/wiki/Chromium_(web_browser)) and [Node.js](https://en.wikipedia.org/wiki/Node.js).

**Features:**

Atom is a "hackable" text editor, which means it is customizable. There is an [init script](https://en.wikipedia.org/wiki/Init) one can customize using [CoffeeScript](https://en.wikipedia.org/wiki/CoffeeScript), a style sheet to customize the looks of Atom, and a keymap to map or re-map [key combinations](https://en.wikipedia.org/wiki/Key_combination) to commands. One can even make a package to wrap all of this functionality into a single package, written in their choice of CoffeeScript or JavaScript.

***2.3.2 Python***

Python is an object-oriented high-level programming language. Python can be used to build server-side web applications. While a [web framework](https://www.fullstackpython.com/web-frameworks.html) is not required to build web apps, it's rare that developers would not use existing open source libraries to speed up their progress in getting their application working. Python offers concise and readable code. Python is easy to use and has a simple syntax.

Python is relatively easy to read and understand, as its syntax is more like English. Its straightforward layout means that you can work out what each line of code is doing. As it’s an open-source language, anyone can use Python to code. What’s more, there is a community that supports and develops the ecosystem, adding their own contributions and libraries. Because Python is such a stable, flexible, and simple programming language. As you can see, there are many applications for this popular language, with a wide support network and a diverse range of libraries that can help.

**2.3.3 Hardware requirements:**

* 100GB Free Space
* idle cut-off 90 minutes
* RAM: ~12.6 GB Available
* Disk: ~33 GB Available

***2.4 Operation Environment***

The table shown below is the minimum requirement:

**Table** **2.1 Table for operation environment**

|  |  |
| --- | --- |
| Processor | Intel Pentium 233Ghz or better performance |
| Operating System | Microsoft Window XP, Vista or Window 7 |
| Memory | 2GB RAM |
| Screen Resolution | Monitor with screen resolution minimum 1024 x 768 |

**CHAPTER-3**

**3.EXPERIMENTAL OR MATERIALS AND METHODS, ALGORITHMS USED**

**3.1 Web Framework**

A web framework is an architecture containing tools, libraries, and functionalities suitable to build and maintain massive web projects using a fast and efficient approach. They are designed to streamline programs and promote code reuse. To create the server-side of the web application, you need to use a server-side language. Python is home to numerous such frameworks, famous among which are Django and Flask.

**3.2 Framework-Flask**

[Flask](http://flask.pocoo.org/) is a lightweight Python web framework that provides useful tools and features for creating web applications in the Python Language. It gives developers flexibility and is an accessible framework for new developers because you can build a web application quickly using only a single Python file. Flask is also extensible and doesn’t force a particular directory structure or require complicated boilerplate code before getting started.

Flask Framework depends on two external libraries: The Jinja2 template, Werkzeug WSGI toolkit. Even though we have a plethora of web apps at our disposal, Flask tends to be better suited due to -

* Built-in development server, fast debugger.
* Integrated support for [unit testing](https://www.xenonstack.com/insights/what-is-unit-testing).
* RESTful request dispatching.
* Jinja2 Templating.
* Support for secure cookies.
* Lightweight and modular design allows for a flexible framework.

**3.3 Flask –SQLAlchemy**

Flask-SQLAlchemy is an extension for [Flask](https://palletsprojects.com/p/flask/) that adds support for [SQLAlchemy](https://www.sqlalchemy.org/) to your application. It aims to simplify using SQLAlchemy with Flask by providing useful defaults and extra helpers that make it easier to accomplish common tasks

SQLAlchemy is the Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL.

It provides a full suite of well known enterprise-level persistence patterns, designed for efficient and high-performing database access, adapted into a simple and Pythonic domain language.

SQL databases behave less like object collections the more size and performance start to matter; object collections behave less like tables and rows the more abstraction starts to matter. SQLAlchemy is most famous for its object-relational mapper (ORM), an optional component that provides the **data mapper pattern**, where classes can be mapped to the database in open ended, multiple ways - allowing the object model and database schema to develop in a cleanly decoupled way from the beginning.

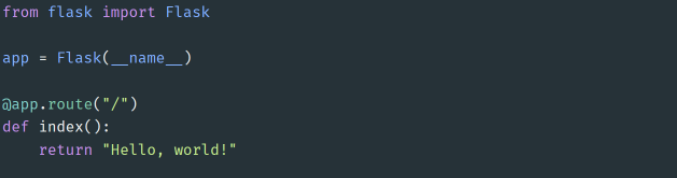
**3.4 Critical Elements**

* Module Init - (project\_root/app\_name/admin/\_\_init\_\_.py) - required to enable the app
* Module URL - (project\_root/app\_name/admin/url.py) - Url definitions of each module
* App root Init - (project\_root/app\_name/\_\_init\_\_.py) - Not necessary to define the entire app within \_\_init\_\_.py
* Module Views - (project\_root/app\_ame/admin/views.py) - Defines views for each module. Separate ‘.py.’ Files as the project scale to ensure they are accessible to URLs.
* Module Templates - (project\_root/app\_name/admin/templates/admin/main.html) - Normal template folder.

**3.5 Routes and View functions**

Clients send requests to the webserver, in turn, sends them to the Flask application instance. The instance needs to know what code needs to run for each URL requested and map URLs to Python functions. The association between a URL and the function that handles it is called a route. The most convenient way to define a route in a Flask application is through the (app.route). Decorator exposed by the application instance, which registers the ‘decorated function,’ decorators are python feature that modifies the behavior of a function.

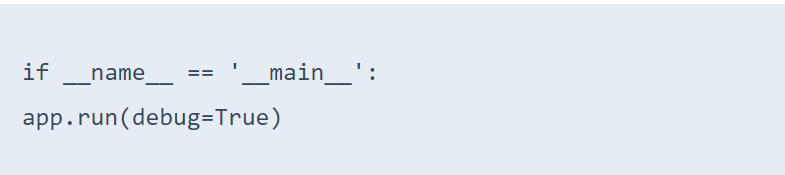
Fig 3.1 Basic Flask App



The index is a view function, and the response can even be a string format HTML.

* Server Startup - The application instance has a ‘run’ method that launches flask’s integrated development webserver –

Fig 3.2 Main function



Once the script starts, it waits for requests and services in a loop.

* Local-Host - Run a python script in a virtual environment. Flask starts the server listening on 127.0.0.1 and port 5000 by default. To accept connection from any remote address, use host = ‘0.0.0.0.’

**3.6 HTTP Methods**

**Request**

To process incoming data in Flask, you need to use the request object, including mime-type, IP address, and data. HEAD: Un-encrypted data sent to server w/o response.

**GET**

Sends data to the server requesting a response body.

**POST**

Read form inputs and register a user, send HTML data to the server are methods handled by the route. Flask attaches methods to each route so that different view functions can handle different request methods to the same URL.

**Response**

Flask invokes a view function. It has to return a response value to the client. HTTP requires it to be more than a string response, a status code.

**Templates**

To maintain the site. Flask uses a powerful template engine, ‘Jinja2’, in its simplest form. A Jinja2 template is a file that contains the text of a response, returned by a view function that has a dynamic component represented by a variable.

**Linking**

Dynamic url routing support is included using ‘url\_for()’ helper function. For example, url\_for('sagar', name='project\_file', \_external=True) would return http://localhost:5000/sagar/project\_file.

#### Security

CSRF(Cross-Site-Request-Forgery) occurs when a malicious website sends requests to a different website on which the victim logs in. Flask-WTF protects against all such attacks. Apart from that, Flask also implements some common security mechanisms like session-based management, role mgmt, password hashing, basic HTTP and token-based authentication, optional log-in tracking.

## *3.7 Importing Modules*

## *Fig 3.1 Modules*

import os

from flask import Flask

from flask\_sqlalchemy import SQLAlchemy

from flask\_migrate import Migrate

from flask\_login import LoginManager

from werkzeug.security import generate\_password\_hash,check\_password\_hash

from flask\_login import UserMixin

from datetime import datetime

## *3.7.1 OS-*

It is possible to automatically perform many operating system tasks. The OS module in Python provides functions for creating and removing a directory (folder), fetching its contents, changing and identifying the current directory, etc.

You first need to import the os module to interact with the underlying operating system. So, import it using the import os statement before using its functions.

## The OS module in Python provides functions for interacting with the operating system. OS comes under Python’s standard utility modules. This module provides a portable way of using operating system-dependent functionality. The \*os\* and \*os.path\* modules include many functions to interact with the file system.

## 3.7.2 Flask-

## To use the flask web framework we need to install it and then import it. Importing flask module in the project is mandatory.

## 3.8.3 SQLAlchemy-

## To use the SQLAlchemy database we need to import it.

## 3.7.4 Migrate-

## Flask migrate is defined as an extension that is used in the Flask application for handling database migrations for SQLAlchemy using Alembic. This module enables developers to quickly set up and starts the database schema migrations.

## The reason we require database migrations can be explained as follows. Suppose we build a database, and then require it to be modified by adding an extra column. Post addition we feel that the schema now present doesn’t fit well into the full application architecture and would like to return back to the original one. In a normal case it is difficult to do so, but with flask migrate the tasks are much smoother! In this article, we will look at ways on how we perform migration in Flask.

## Database migrations are the art of managing changes that are incremental, reversible, and controlled by version in the relational database schema. In the definition, we saw that we perform any migration when we feel the necessity to update into a newer version or even revert back to the older version.

## 3.7.5 Flask-Login-

## Flask-Login provides user session management for Flask. It handles the common tasks of logging in, logging out, and remembering your users’ sessions over extended periods of time.

## It will:

## Store the active user’s ID in the session, and let you log them in and out easily.

## Let you restrict views to logged-in (or logged-out) users.

## Handle the normally-tricky “remember me” functionality.

## Help protect your users’ sessions from being stolen by cookie thieves.

## Possibly integrate with Flask-Principal or other authorization extensions later on.

## 3.7.6 Werkzeug.security-

## *Werkzeug* is python library which contains lot of development and debugging tools for implementation of web application gateway interface(WSGI) applications. The good part is you can use this system not only for your web applications but also for standalone python applications like desktop apps, scripts, mobile apps and so on.

## Secure password storage:

## Passwords will not be stored as plaintext in database.

## Password will be stored as hash which is irreversible to plaintext. (one way hash).

## With given hash attacker cannot guess plaintext.

## Each user password will be hashed with salt to mitigate rainbow table attacks; Just in case if database got compromised.

## 3.7.7 UserMixin-

Flask-login requires a User model with the following properties:

* has an is\_authenticated() method that returns True if the user has provided valid credentials
* has an is\_active() method that returns True if the user’s account is active
* has an is\_anonymous() method that returns True if the current user is an anonymous user
* has a get\_id() method which, given a User instance, returns the unique ID for that object

UserMixin class provides the implementation of this properties. Its the reason you can call for example is\_authenticated to check if login credentials provide is correct or not instead of having to write a method to do that yourself.

**3.7.8 Date and Time-**

In Python, date and time are not a data type of their own, but a module named datetime can be imported to work with the date as well as time. Python Datetime module comes built into Python, so there is no need to install it externally. Python Datetime module supplies classes to work with date and time

**CHAPTER 4**

**4.RESULTS AND DISCUSSION****, PERFORMANCE ANALYSIS**

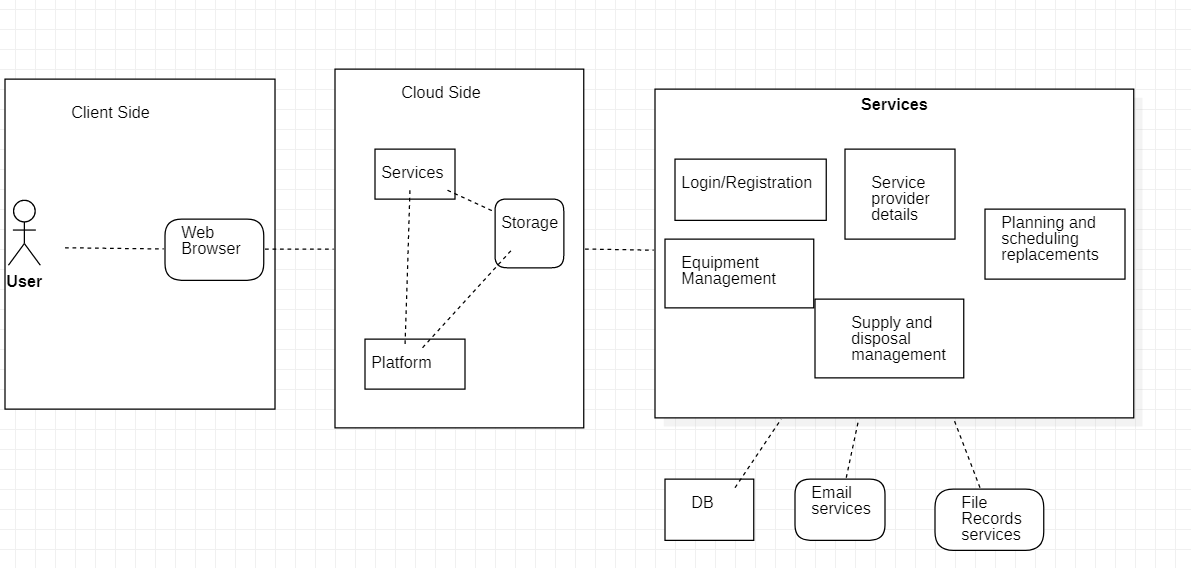
***4.1 Outcomes:***

A web based application which is a platform between sellers and buyers has been developed.

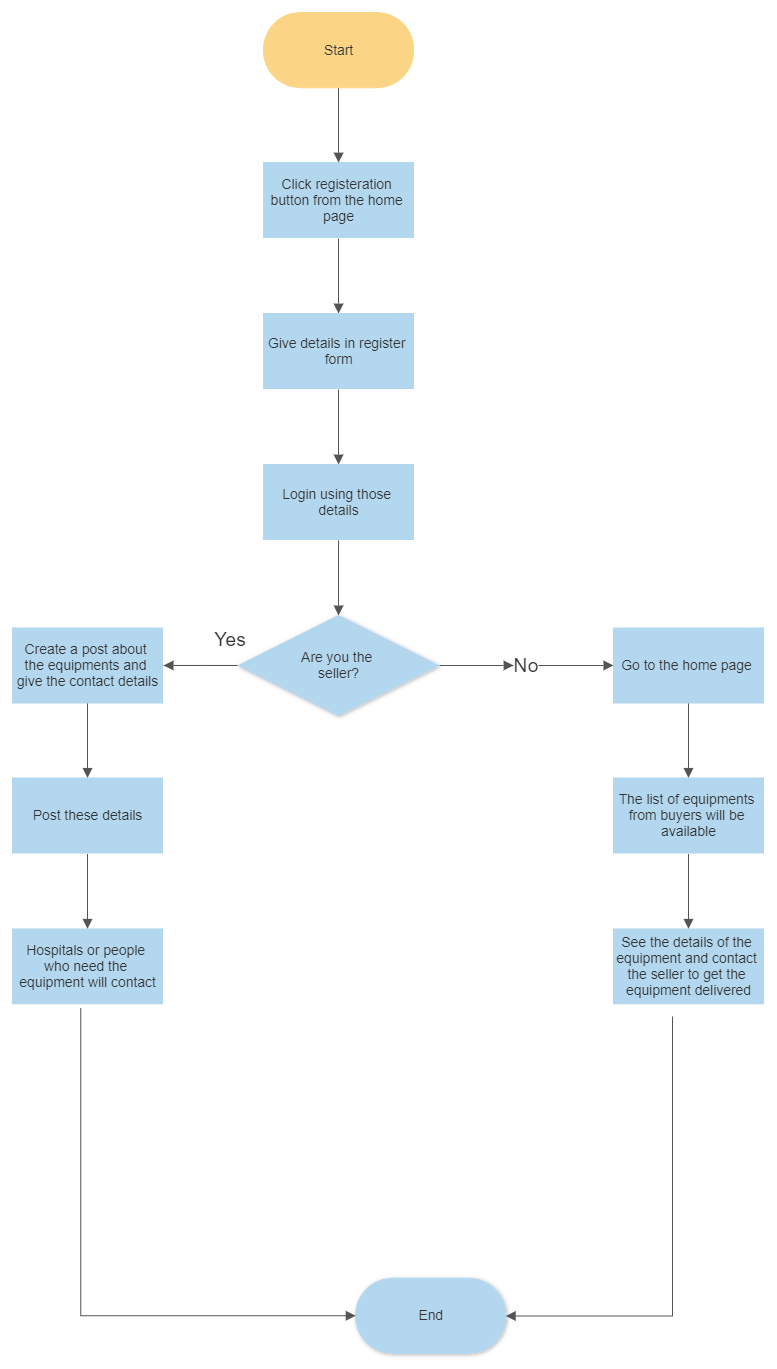
We would be able to:

* Sell medical equipments
* Donate medical equipments to those in need
* Get contact details of those who wish to sell the equipments

***4.2* Architecture Diagram:**

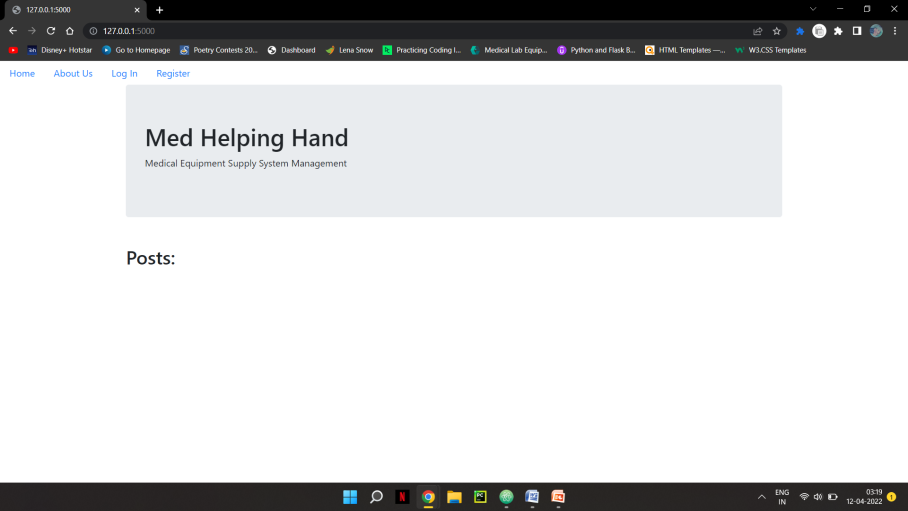
****

***4.3 Flowchart***

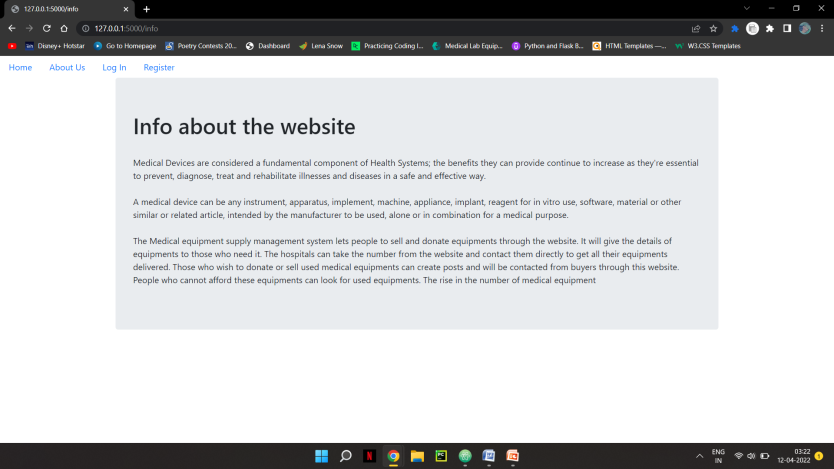
******

**4.4 Application Snapshots**

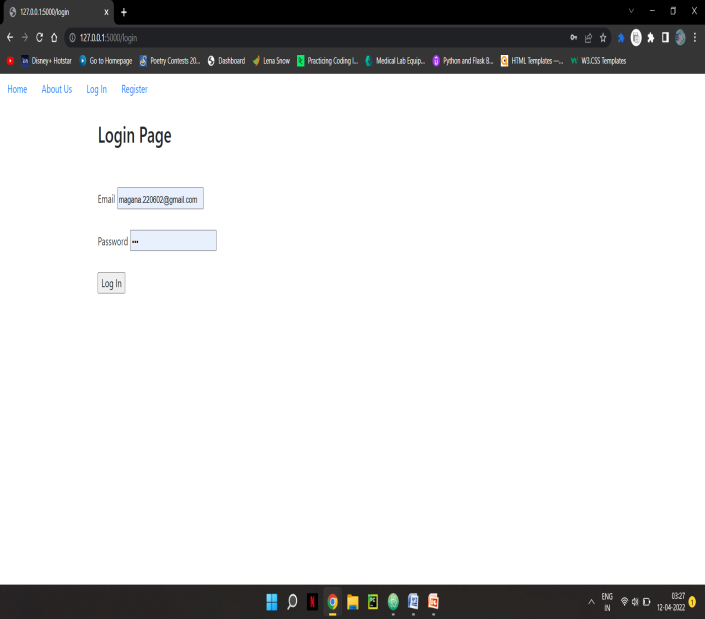
**Home Page**

****

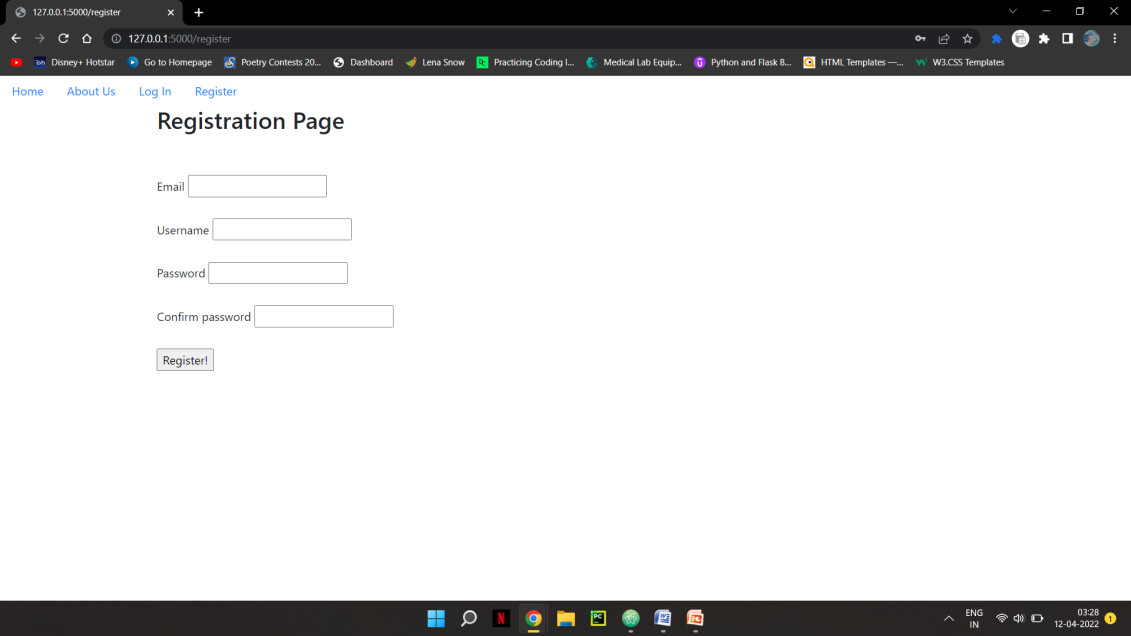
**Info Page:**

****

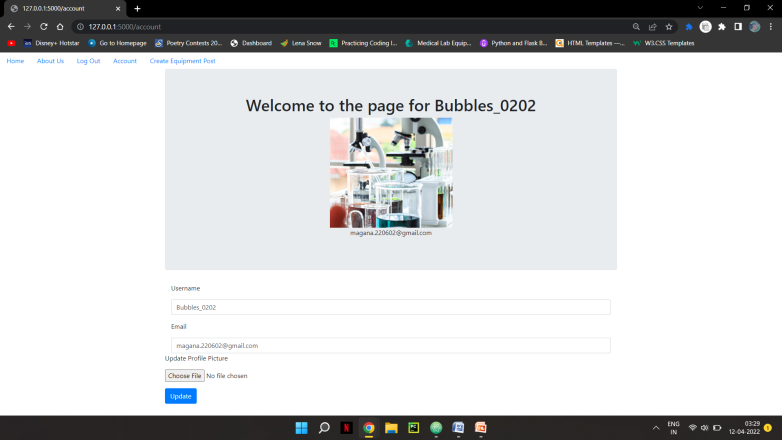
Login Page:



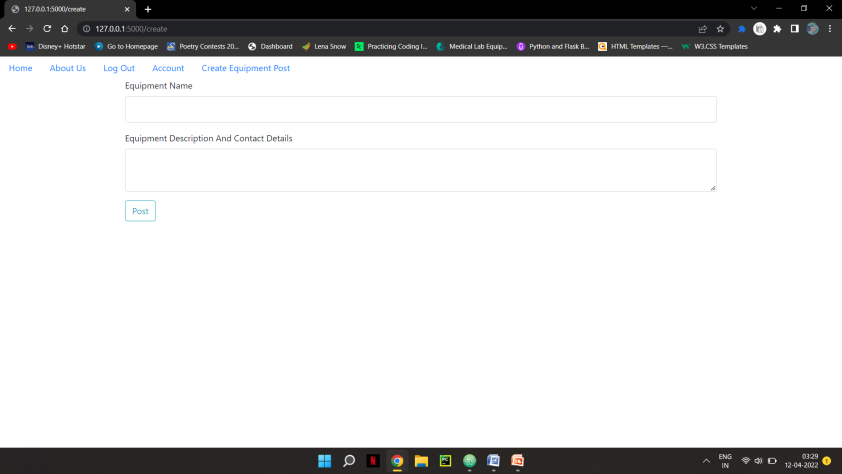
Registration Page:



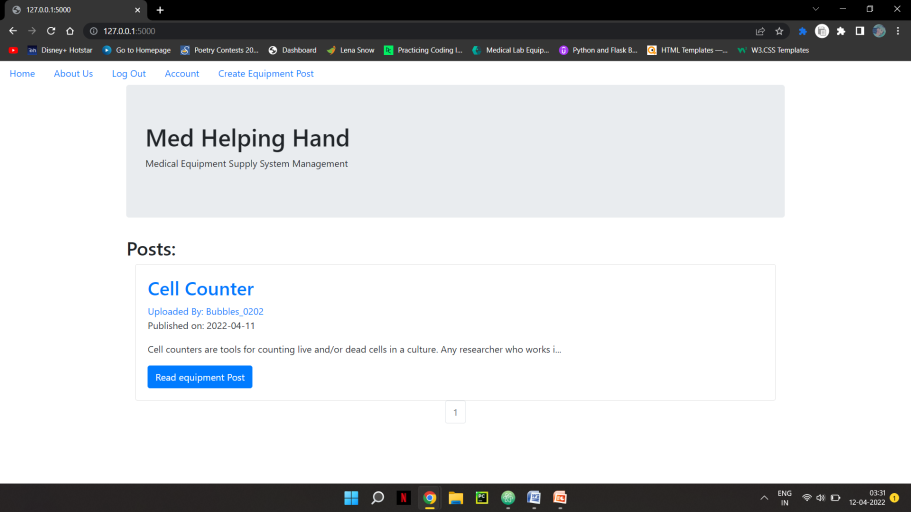
Account page:



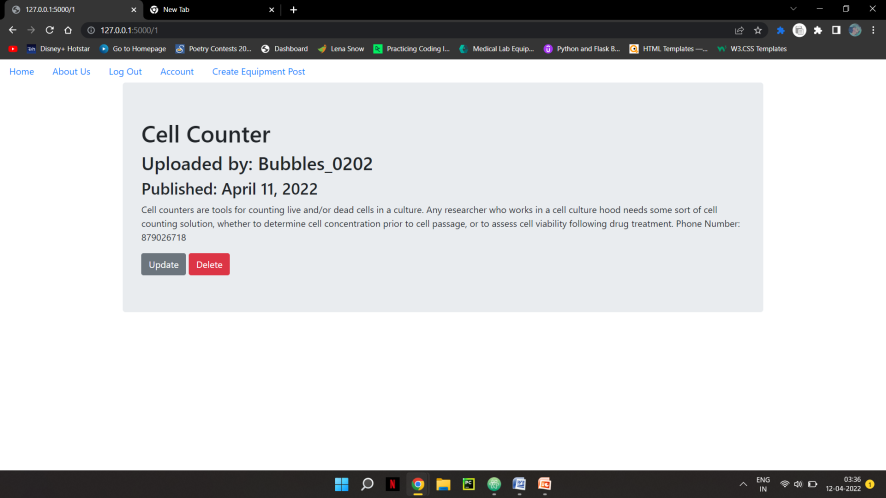
Create post page:



Posts Page:

****

Inside each post:

****

**CHAPTER 5**

**SUMMARY AND CONCLUSIONS**

# The Medical equipment supply management system shows all types of equipments from various sellers at all price range in one place. The buyers can search for their needed equipments and get the contact details. The Sellers of medical equipment can post the details about the equipment and give their contact details. Hence, this web application acts as an intermediate between the buyers and the sellers. With just one click people can get the equipment details and the contact details.

# The people who wish to donate the used equipments can also sell in this web application. Those who cannot afford to buy new medical equipments can still buy these used equipments at a cheaper rate. The people no longer need to worry about taking care of their health.

# The buyers don’t need to go to various websites and see all the businesses selling equipments. They can see all types of equipment posts in one website.

# During the emergency times like Covid-19 many people were in the hospitals without proper beds and adequate medical equipments. So many lost their lives without proper medical attention. This web application helps the medical staff to easily know where they can buy their equipments and keep everything stocked even during these times. They don’t need to worry and can search for sellers easily.

# *5.1 Sample Code:*

# *Models.py*

# from equipmentsupply import db,login\_manager

# from datetime import datetime

# from werkzeug.security import generate\_password\_hash,check\_password\_hash

# from flask\_login import UserMixin

# @login\_manager.user\_loader

# def load\_user(user\_id):

# return User.query.get(user\_id)

# class User(db.Model, UserMixin):

# # Create a table in the db

# \_\_tablename\_\_ = 'users'

# id = db.Column(db.Integer, primary\_key = True)

# profile\_image = db.Column(db.String(20), nullable=False, default='default\_profile.png')

# email = db.Column(db.String(64), unique=True, index=True)

# username = db.Column(db.String(64), unique=True, index=True)

# password\_hash = db.Column(db.String(128))

# posts = db.relationship('EquipmentPost', backref='author', lazy=True)

# def \_\_init\_\_(self, email, username, password):

# self.email = email

# self.username = username

# self.password\_hash = generate\_password\_hash(password)

# def check\_password(self,password):

# return check\_password\_hash(self.password\_hash,password)

# def \_\_repr\_\_(self):

# return f"UserName: {self.username}"

# class EquipmentPost(db.Model):

# # Setup the relationship to the User table

# users = db.relationship(User)

# id = db.Column(db.Integer, primary\_key=True)

# user\_id = db.Column(db.Integer, db.ForeignKey('users.id'), nullable=False)

# date = db.Column(db.DateTime, nullable=False, default=datetime.utcnow)

# title = db.Column(db.String(140), nullable=False)

# text = db.Column(db.Text, nullable=False)

# def \_\_init\_\_(self, title, text, user\_id):

# self.title = title

# self.text = text

# self.user\_id =user\_id

# def \_\_repr\_\_(self):

# return f"Post Id: {self.id} --- Date: {self.date} --- Title: {self.title}"

# \_\_init\_\_.py

# import os

# from flask import Flask

# from flask\_sqlalchemy import SQLAlchemy

# from flask\_migrate import Migrate

# from flask\_login import LoginManager

# app = Flask(\_\_name\_\_)

# ##Configurations

# app.config['SECRET\_KEY'] = 'mysecret'

# #################################

# ### DATABASE SETUPS

# ###############################

# basedir = os.path.abspath(os.path.dirname(\_\_file\_\_))

# app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///' + os.path.join(basedir, 'data.sqlite')

# app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = False

# db = SQLAlchemy(app)

# Migrate(app,db)

# ##########################

# #### LOGIN CONFIGS

# #########################

# login\_manager = LoginManager()

# login\_manager.init\_app(app)

# login\_manager.login\_view = "users.login"

# ###Blue Print Configurationss

# from equipmentsupply.core.views import core

# from equipmentsupply.users.views import users

# from equipmentsupply.equipment\_posts.views import equipment\_posts

# from equipmentsupply.error\_pages.handlers import error\_pages

# # Register the apps

# app.register\_blueprint(users)

# app.register\_blueprint(equipment\_posts)

# app.register\_blueprint(core)

# app.register\_blueprint(error\_pages)

# *5.2 References:*

1. [**https://www.fingent.com/blog/top-5-open-source-erp-systems-for-medical-equipment-suppliers/**](https://www.fingent.com/blog/top-5-open-source-erp-systems-for-medical-equipment-suppliers/)
2. [**https://www.hindawi.com/journals/jhe/2021/6685456/**](https://www.hindawi.com/journals/jhe/2021/6685456/)
3. [**https://flask.palletsprojects.com/en/2.1.x/**](https://flask.palletsprojects.com/en/2.1.x/)
4. [**https://www.digitalocean.com/community/tutorials/how-to-make-a-web-application-using-flask-in-python-3**](https://www.digitalocean.com/community/tutorials/how-to-make-a-web-application-using-flask-in-python-3)
5. [**https://www.sqlalchemy.org/**](https://www.sqlalchemy.org/)