**DOCTOR-PATIENT APPOINTMENT SYSTEM**

*A dissertation submitted in partial fulfilment of the requirements for the degree of*

**Master of Computer Application**

*Submitted by*

**VIVEK KUMAR VERMA (CSM21040)**

Guided by:

|  |  |  |
| --- | --- | --- |
| **Dr. Rosy Sarmah**  Associate Professor  Tezpur University |  | Mr. Nimish Bhardwaj Programmer Analyst  Tech Variable |



School of Engineering

Department of Computer Science and Engineering

Tezpur University

Napaam - 784028, Assam, India

June 2022



Department of Computer Science and Engineering

Tezpur University

**CERTIFICATE BY SUPERVISOR**

This is to certify that the report entitled “**Doctor–Patient Appointment System**” submitted to the Department of Computer Science and Engineering, Tezpur University in partial fulfilment for the award of the degree of Master of Computer Application, is a record of project work carried out by **Vivek Kumar Verma (CSM21040)** under my supervision during the period from January, 2023 to June, 2023. All support received by him from various sources have been duly acknowledged. No part of this report has been submitted elsewhere for the award of any other degree or diploma.

|  |  |  |
| --- | --- | --- |
| Date:  Place**:** |  | Prof. Rosy Sarmah  Department of CSE  Tezpur University |



Department of Computer Science and Engineering

Tezpur University

**CERTIFICATE BY THE EXAMINER**

This dissertation titled “**Doctor-Patient appointment System**” submitted by **Vivek Kumar Verma (CSM21040)** in partial fulfilment of the requirements for the major project of Master of Computer Applications has been examined.

Examiner:

Date:

Place:



Department of Computer Science and Engineering

Tezpur University

**DECLARATION**

I hereby declare that the dissertation work titled **Doctor-Patient Appointment System** submitted to the Department of Computer Science & Engineering, Tezpur University is prepared by me and was not submitted to any other institution for award of any other degree.

Place: Vivek Kumar Verma

Date: CSM21040

Department of Computer Science

and Engineering

**ACKNOWLEDGEMENT**

I would like to extend my heartful gratitude to my project guide Dr. Rosy Sarmah, Associate Professor, Dept. Of CSE, Tezpur University, for giving me the opportunity to work under her and providing me ample guidance and support through the course of the project.

I would also like to thank Mr. Nimish Bardwaj ( Programmer Analyst ), Mr. Sivaranjan Goswami

( Senior Programmer Analyst ), Tech Variable for guiding me throughout the project.

I would also like to thank Head of the department of Computer Science and Engineering department and all the Faculty members of Dept. of CSE, Tezpur University for the valuable guidance and co-operation throughout the project.

I would also like to extend my gratitude to the God, family and friends who has been helpful in keeping me in a good state of mind throughout this project.

My thanks and appreciations also go to all other people who have directly or indirectly helped me out with their abilities.

Vivek Kumar Verma

CSM21040

**Abstract**

The objective of the project is to create an “Online Doctor-Patient Appointment System” web app as a part of my internship at “TechVariable Pvt Ltd.”.

The Doctor-Patient Appointment System is a digital platform designed to streamline and simplify the process of scheduling and managing appointments between doctors and patients. This system aims to enhance communication and convenience for doctors and patients, ultimately improving the healthcare experience. This project is an EHR project being developed at TechVariable.

The Doctor-Patient Appointment System allows patients to easily get register, create a profile and add their medical records. They can easily search for doctors based on their names specialty, location, and availability. They can see the details of the doctors. Once a suitable doctor is selected, patients can request an appointment at their preferred date and visiting location of the doctor. The patient can easily see the history of the appointment and their status.

Doctors, on the other hand, benefit from an organized and automated appointment system. They can update their profile and update their availability status. They are also given the facility of adding multiple visiting locations and visiting hours for those locations. Upon receiving the appointment, the doctor can confirm, or cancel it. After checking the patient on the day of the appointment doctor can mark the appointment as complete. Each update on the status of the appointment is sent to the respective patient through email. The doctor is provided with an easy-to-use interface and a short summary of current-day appointment statistics.

This system segregates the current-day appointments and arranges them in different segments making the work of doctors more convenient.

By implementing the Doctor-Patient Appointment System, healthcare institutions can optimize their appointment scheduling processes, reduce administrative burdens, and enhance patient satisfaction. Patients can easily access quality healthcare services, while doctors can efficiently manage their schedules and provide personalized care. Overall, this system aims to foster a seamless doctor-patient relationship and improve healthcare accessibility for all.

KEYWORD: Python, Django, Postgresql, SMTP, Tailwind CSS, HTML, Multi-threading.

**About the Company**

TechVariable, is an Enterprise IT services and consultancy company have been working on enterprise-grade application development for the last 4 years. We are headquartered at Guwahati and also operate from Bangalore.

Here are a few of our projects that can be highlighted to showcase our capabilities (additionally you may also visit <https://www.techvariable.com/case-study/):>

1) Technology partner for a Product Life-cycle Management vendor:

We are the Technology Partner for a Chicago based client who is currently at a product development process to develop a next-gen adaptive PLM solution which has the intelligence to  
help enterprises grow their business, enhance their visibility, integrate their data, processes, people, and business systems all within the system. We are involved in end to end development  
and maintenance of the product as well as providing support services to the end clients.  
The underlying technologies are Angularjs in the front-end with Nodejs as our backend and Neo4j as our database. This also includes interface to Elastic Search and also interaction to BPMN engine, CAD viewer and other document viewers as well. Also, we are working on an interface where we are to read large CSV files using Python, clean the data and upload it to this PLM system.

2) Data lake platform for a Malaysian news agency:

This platform is a combination of many features packed into a centralized data store for analysis and dissemination. It includes solutions like data exploration/aggregation, data governance, and data visualization. This has social media aggregation and analytics, use of web scrapers and  
crawlers, audio/video transcription, content-language translation, sentiment analysis and graphical visualization using Tableau, Power BI, etc.

3) Balanced performance Dashboard for a Food & Beverage company:

A project which involves creating a graphical dashboard by integrating data silos residing in a separate application for finance and engineering. This project involved data extraction from SAP and Oracle Agile and injecting the data to the dashboard to generate actionable business KPIs.

Our other services include Microservices, AWStack, DevOps, DataLake, Business Intelligence, etc.

We are currently working on a Dedicated Development Centre methodology where a team of dedicated resources headed by a project manager works for a particular client. This ensures a smooth and seamless process. We follow the agile process of development and our team is very comfortable with daily stand up & bi-weekly sprint execution for accounting 100% deliverables on  
time. We have good experience in working at Offshore Software Development projects and have worked with clients from Chicago and New York in the USA, Denmark, Sydney, Dubai, and Qatar.

**Contents**

1. Introduction --------------------------------------------------------------------- 9
   1. Problem Statement ------------------------------------------------------------------- 9
   2. Objectives ----------------------------------------------------------------------------- 10
2. Initial System Study ------------------------------------------------------ 11-12
3. Feasibility Analysis --------------------------------------------------------------- 13-14
4. System Analysis ------------------------------------------------------------------- 15
5. System Requirement Specification --------------------------------------------- 16-17
6. Language and Tools being used ------------------------------------------------- 18
7. System Design ------------------------------------------------------------------ 19
   1. System Architecture -------------------------------------------------------­- 19-20
   2. Database Schema ------------------------------------------------------------- 21
   3. Data Flow Diagram ---------------------------------------------------------- 22-27
8. Testing ------------------------------------------------------------------------------- 28-30
9. Results and Discussion --------------------------------------------------------- 31-39
10. Conclusion & Future work ---------------------------------------------------------- 40
11. Bibliography ------------------------------------------------------------------------- 41

**Chapter 1**

**Introduction**

The Doctor-Patient Appointment System is a modern solution that simplifies the healthcare experience for doctors and patients. Traditional appointment booking methods can be frustrating and time-consuming, causing delays and inconvenience. But with this innovative system, everything becomes hassle-free.

For patients, the system offers a user-friendly web portal. It allows them to quickly search for doctors based on their needs. No more waiting on hold or struggling to find the right date and time. Just a few clicks and the appointment is booked.

Healthcare providers benefit from the system too. They can easily manage their schedules, access patient records, and receive reminders for upcoming new appointments. The system also helps optimize scheduling, ensuring that resources are utilized efficiently and reducing the chances of double-booking.

Implementing the Doctor-Patient Appointment System improves operational efficiency, reduces administrative burdens, and enhances patient satisfaction. It's a modern way to revolutionize healthcare by making it easier and more convenient for everyone involved.

**1.1 Overview**

This is a Django based Doctor-Patient Appointment System Web App offers an efficient and convenient platform for patients to book appointments with their preferred doctors. It empowers healthcare providers to optimize their schedules, manage patient records, and enhance communication with their patients via email. By leveraging Django's robustness and flexibility, this web app contributes to an improved healthcare experience for both doctors and patients.

**1.2 Objective**

Some of the main key objectives of the project are as follows:

* Simplify Appointment Booking:

The primary objective of the doctor-patient appointment system is to simplify the process of booking appointments. It should provide a user-friendly interface that allows patients to easily search for doctors, view their availability, and book appointments with just a few clicks.

* Enhance User Experience:

This System should enhance and modernise the way the appointments are made in the earlier times. It gives users an easy to understand system where they can easily get registered, get securely login, set profile, upload medical records and book appointments.

* Develop Robust and Scalable System:

Robustness and Scalability has been the need of the hour for any of the modern day system. This system is highly robust and easy to scale.

* Enhance Data Security and Privacy:

One of the critical objectives is to ensure the security and privacy of patient data. The app should implement robust security measures to protect sensitive information, comply with data protection regulations, and ensure that patient records are kept confidential.

* Make Doctors Work Easy:

Doctor can easily make decision about the appointment by studying the medical records of the patients. This makes the doctor’s job easier in treating a patient with personalized medical care. Appointments are organized and shown in a very nice interface which helps the doctor to keep record of the appointment statistics.

* Save Time and Effort:

This system proves the facility of searching the doctor and provides all relevant information about the doctor so that patients can easily choose the doctor of their preference and book appointment in just a few clicks to their convenient visiting location of the doctor. This saves their time and effort to a great extend.

**Chapter 2**

**Initial System Study**

**2.1 Drawback Current System:**

A large healthcare institution was facing challenges with its appointment booking process. Patients had to endure long waiting times, struggled to find available slots, and experienced frequent miscommunications. This resulted in frustrated patients, reduced efficiency, and increased administrative burden on healthcare providers.

**2.2 Proposed System:**

The Doctor-Patient Appointment System, a modern web-based solution. The system was built using Django, a robust framework for web development. It offered a comprehensive set of features to simplify and streamline the appointment booking process.

Key features:

* + - Easy appointment procedure
    - Flexibility of time to book appointment
    - Secure storage of medical records
    - Notification via email
    - Easy interface for both doctor and patient
    - Robust and scalable system

**2.3 Scope of System:**

The proposed Doctor-Patient Appointment system is designed keeping in mind the point to make the hectic appointment process easy and convenient for both doctor and patients. This project has the strength to revolutionize the medical system by providing the services like electronic health record (E.H.R.). It saves the valuable amount of time and resources of the users and so it has great scope of success in the near future.

**2.4 Scope of this project**

This project has been done in the fourth semester of the Master of Computer Application (MCA) course of Tezpur University. The period of work is roughly four months. Within this project we accomplish the following part of the system proposed above -

1. Learning the Django system
2. Working on requirement specification
3. Database selection and models design
4. Creating a good user interface design
5. Coding the entire project in MVT architecture of the Django.

After the accomplishment of the project evaluation of its performance and optimization of the system has been a great challenge.

This project comes with various features as mentioned in sub-point 2.2. With the good scope of automating the appointment process and providing the safety, security to confidential data of the users, this project has an excellent chance of succeeding.

**Chapter 3**

**Feasibility Analysis**

Two kinds of feasibility were analyzed for the project

**3.1 Economical Feasibility**

This product can be served in the cloud which can provide a flexible storage and computation power as it is required. Since Django provides highly scalable and robust architecture, this system can easily be loaded with the tones of features as needed in the future. This product has a good chance to be converted into a subscription based application which can easily generate more revenue than the storage and server costs.

The Doctor-Patient Appointment System is economically feasible because it helps save money and generate benefits for healthcare institutions. By automating appointment booking, reduces administrative costs and improves efficiency, allowing staff to focus on other important tasks. The system also reduces no-shows through reminders, which saves money by maximizing the use of doctors' time. Additionally, it enhances patient satisfaction, leading to loyal patients who are more likely to return for future appointments. Although there are upfront costs for development and implementation, the long-term benefits outweigh these expenses, making the system a financially viable and beneficial solution for healthcare providers.

**3.2 Technical Feasibility**

Django [1], a powerful Python framework, provides a solid foundation for building web applications. It offers a wide range of features, including database management, URL routing, and user authentication, which are crucial for developing a robust appointment system.

SMTP (Simple Mail Transfer Protocol) [2] integration allows the system to send email notifications, such as appointment reminders and updates, to patients and doctors. This ensures efficient communication and enhances the overall user experience.

PostgreSQL [3], a reliable and scalable open-source database, is a suitable choice for storing and managing appointment-related data. It offers advanced features, such as transactional support and data integrity, to ensure data consistency and reliability.

Tailwind CSS [4], a utility-first CSS framework, allows for rapid development and easy customization of the application's user interface. It provides a set of ready-to-use components and utility classes that streamline the styling process, resulting in a visually appealing and responsive design.

HTML [5] is the fundamental mark-up language for building web pages, while Sweet Alert is a JavaScript library that provides attractive and user-friendly pop-up alerts. These technologies enhance the user interface and provide a seamless experience for both patients and doctors.

The combination of these technologies ensures that the Doctor-Patient Appointment System is technically feasible, scalable, and efficient in managing appointments, handling data securely, and providing a user-friendly interface. The integration of Django, SMTP, PostgreSQL, Tailwind CSS, HTML, and Sweet Alert creates a solid foundation for a successful and reliable web application.

**Chapter 4**

**System Analysis**

The Doctor-Patient Appointment System is carefully analyzed to ensure it meets the needs of both doctors and patients. System analysis involves studying how the system works, identifying requirements, and understanding how it will be developed.

In simple terms, system analysis for this appointment system means thoroughly understanding the problems faced by patients and healthcare providers, such as long waiting times and miscommunications. The analysis considers how the system can solve these issues by providing a user-friendly interface for patients to search and book appointments easily.

The analysis also focuses on the needs of doctors, ensuring they have a convenient way to manage their schedules and access patient records. It examines how the system can improve communication between doctors and patients by providing secure messaging.

Throughout the analysis, the goal is to create a system that is efficient, saves time and money, and enhances the overall experience for both patients and doctors. By understanding the requirements and challenges, the system can be designed and developed to address these needs effectively.

**Chapter 5**

**System Requirement Specification**

1. User Registration and Authentication:

The system should allow patients and doctors to create accounts and authenticate themselves securely.

User registration should collect necessary information such as name, contact details, and email address.

1. Appointment Booking:

Patients should be able to search for doctors based on specialization, location, and availability. The system should provide an intuitive interface for selecting preferred dates and time slots. Patients should receive confirmation notifications once appointments are successfully booked.

1. Doctor Profiles:

Doctors should have the ability to create and manage their profiles.

Profile information should include qualifications, specialties, working hours, and clinic locations.

1. Scheduling and Calendar Management:

Doctors should have a dynamic calendar interface to manage their availability and schedule appointments. Doctors should be able to block off specific time slots for personal time or unavailability.

1. Communication and Notifications:

The system should facilitate secure messaging between doctors and patients.

Patients should receive automated reminders for upcoming appointments through email or in-app notifications.

1. Electronic Medical Records (EMR):

Doctors should have access to patient medical histories and records within the system.

The system should ensure the security and confidentiality of patient medical information.

1. Administrative Dashboard:

The system should provide an administrative panel to manage user accounts, view analytics, and generate reports. The dashboard should allow administrators to perform system maintenance tasks.

1. Security and Privacy:

The system should implement robust security measures to protect user data.

User data, including personal and medical information, should be encrypted and stored securely.

1. Performance and Scalability:

The system should be able to handle a large number of users and appointments without compromising performance. It should be scalable to accommodate future growth and increased user demand.

1. Compliance:

The system should adhere to data protection regulations and ensure compliance with privacy laws.

The System Requirement Specification outlines the key features and functionalities required for the Doctor-Patient Appointment System. These requirements form the basis for the design and development of the system, ensuring it meets the needs of both patients and healthcare providers.

**Chapter 6**

**Language and Tools Being Used**

1. **Django[1]:** Django is a Python web framework used for developing robust and scalable web applications. It provides a high-level, model-view-controller (MVC) architecture that simplifies the development process.
2. **SMTP (Simple Mail Transfer Protocol)[2]:** SMTP is a communication protocol used for sending emails. It is integrated into the appointment system to facilitate email notifications, such as appointment reminders and updates, to patients and doctors.
3. **PostgreSQL [3]:** PostgreSQL is an open-source relational database management system. It is used to store and manage the appointment-related data securely. PostgreSQL offers advanced features like transaction support and data integrity.
4. **Tailwind CSS [4]:** Tailwind CSS is a utility-first CSS framework that simplifies the styling and layout of web pages. It provides a set of pre-defined CSS classes and components, making it easier to create responsive and visually appealing user interfaces.
5. **HTML (Hypertext Markup Language) [5]:** HTML is the standard markup language for creating web pages. It is used to structure and present the content of the appointment system.
6. **Sweet Alert [6]:** Sweet Alert is a JavaScript library that enhances the appearance and functionality of popup alerts. It provides attractive and user-friendly notifications to improve the user experience.

**Chapter 7**

**System Design**

* 1. System Architecture
     1. Presentation Layer:

User Interface (UI): The UI layer handles the presentation of the system to users. It consists of HTML templates using the Jinja templating engine to generate dynamic content and integrate data from the backend.

User Interface Logic: This layer contains the logic responsible for handling user interactions and rendering the UI components. It utilizes JavaScript libraries, such as Sweet Alert, to enhance the user experience with interactive pop-up alerts.

* + 1. Application Layer:
* **Django Framework**: Django serves as the application layer, handling the request-response cycle, routing, and processing user inputs. It coordinates the flow of data between the UI layer and the backend components.

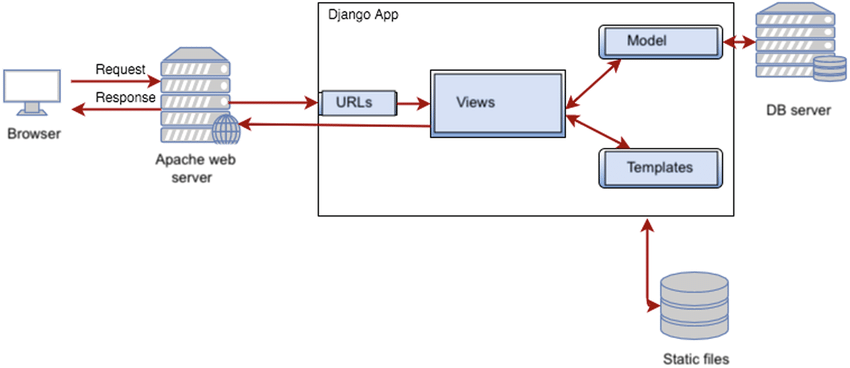


Fig 1: Django Architecture.

* **Business Logic**: The business logic layer manages the core functionalities of the appointment system, including appointment booking, scheduling, and communication between doctors and patients. It handles the logic for managing appointment availability, sending email notifications using SMTP mail, and processing user requests.
* **Multithreading:** Multithreading is utilized to handle concurrent requests and improve system performance. It allows the system to handle multiple user interactions simultaneously, ensuring efficient utilization of system resources.
  + 1. Data Access Layer:

PostgreSQL Database: PostgreSQL serves as the backend database management system, storing and managing the system's data securely. It is accessed through Django's built-in Object-Relational Mapping (ORM) capabilities, providing an interface for querying and manipulating data.

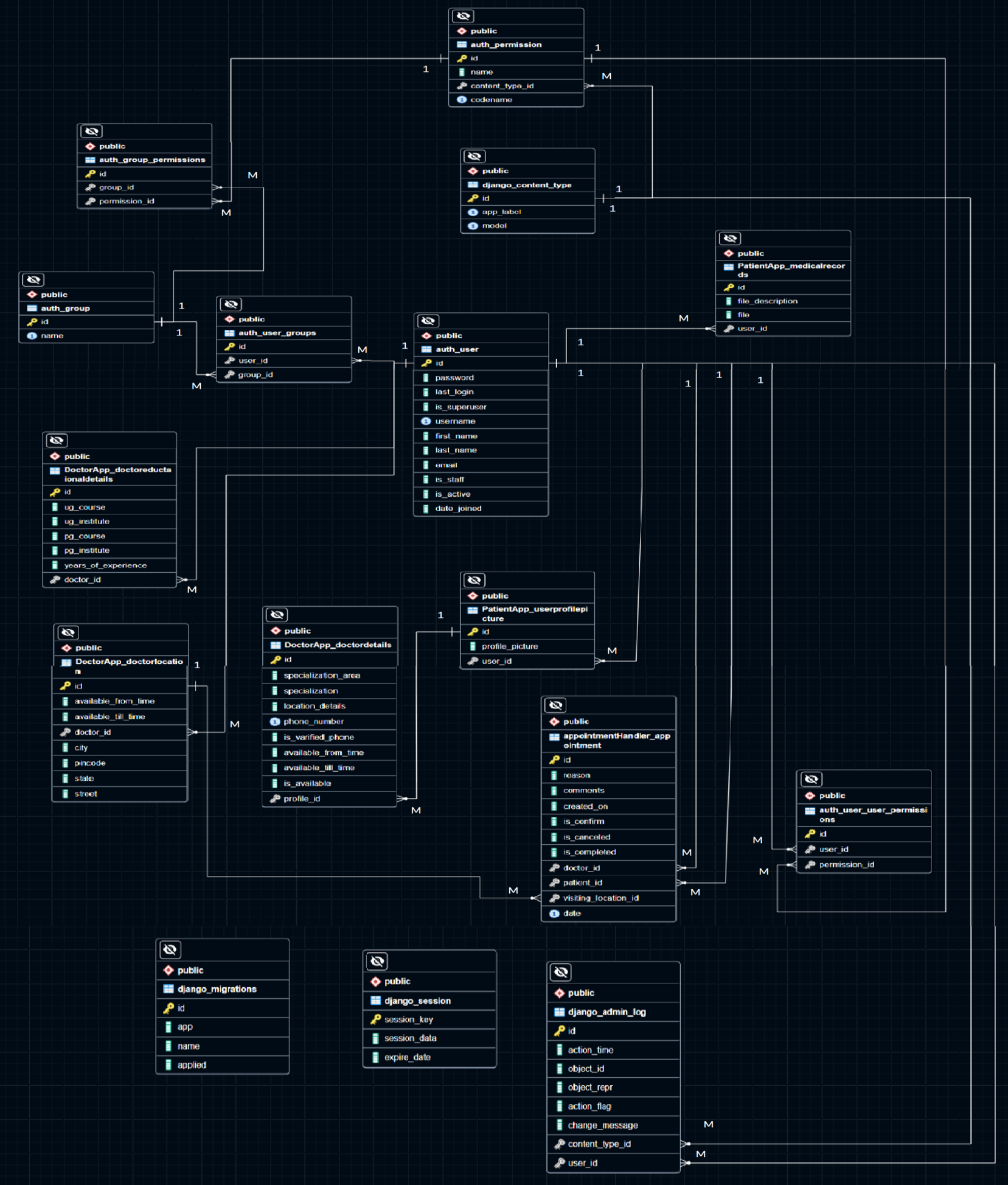
* + 1. External Services:

SMTP Mail Service: An external SMTP mail service is integrated into the system to handle the sending of email notifications, such as appointment reminders and updates. It ensures reliable and efficient email communication between the system and users.

The system architecture follows the Model-View-Controller (MVC) pattern, with Django providing the MVC framework. The Presentation Layer consists of the UI components and UI logic, the Application Layer handles the business logic and request processing, the Data Access Layer manages data storage and retrieval using PostgreSQL, and External Services integrates SMTP mail for communication. Multithreading is employed to enhance system performance by allowing concurrent request handling.

This architecture ensures a scalable, modular, and efficient Doctor-Patient Appointment System, providing seamless user interactions, secure data management, and reliable email communication.

* 1. ER Diagram



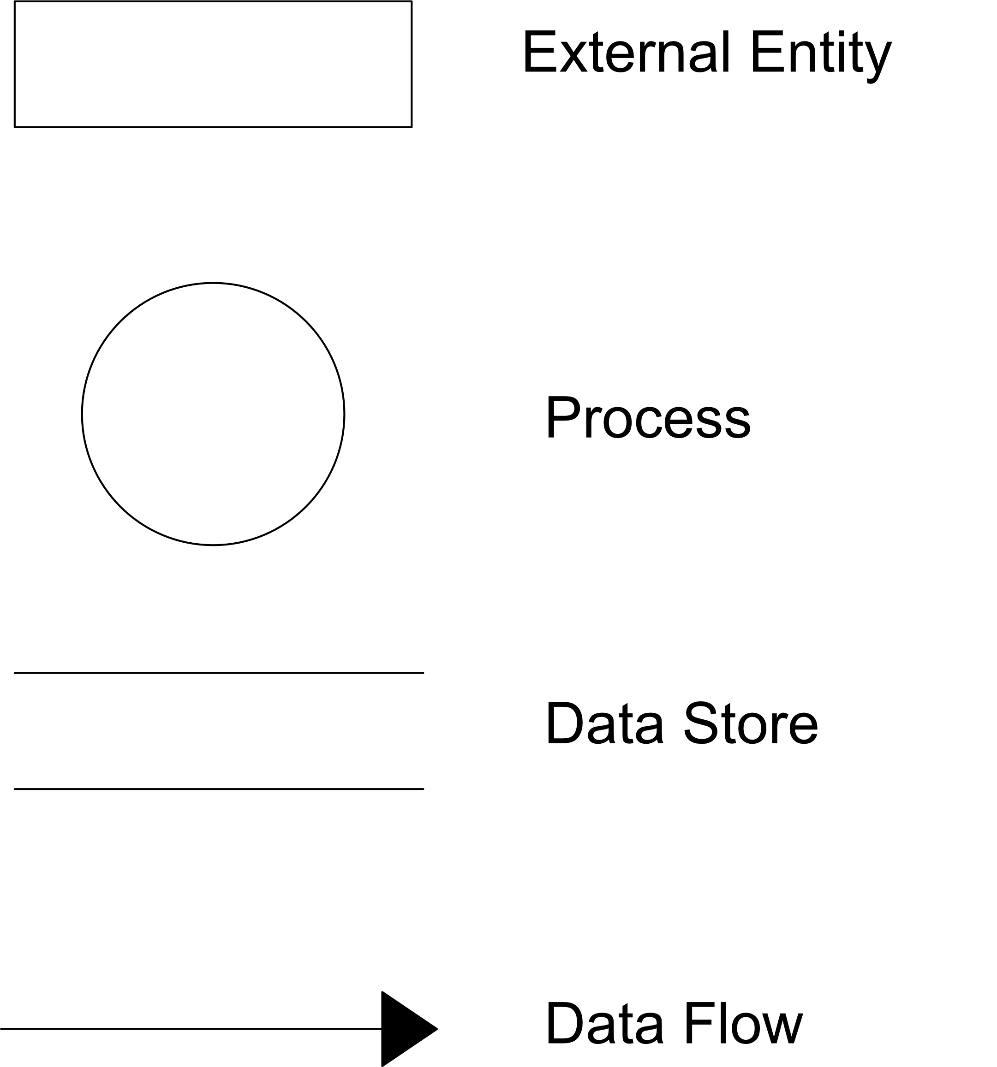
* 1. Data Flow Diagram

**Circle:** A circle (bubble) shows a process that transforms data inputs into data outputs.

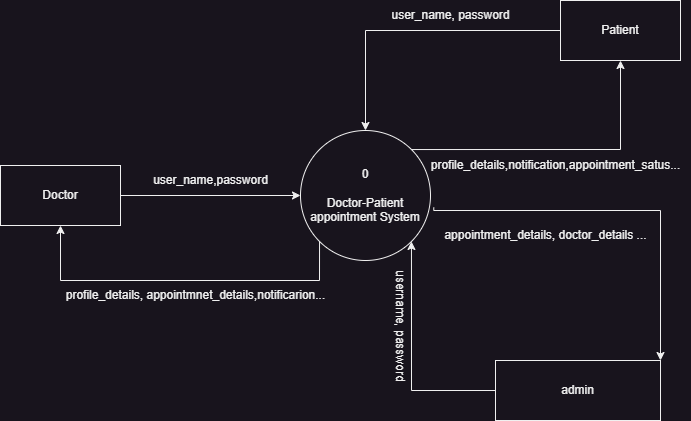
**Data Flow:** A curved line shows the flow of data into or out of a process or data store.

**Data Store:** A set of parallel lines shows a place for the collection of data items. A data store indicates that the data is stored which can be used at a later stage or by the other processes in a different order. The data store can have an element or group of elements.

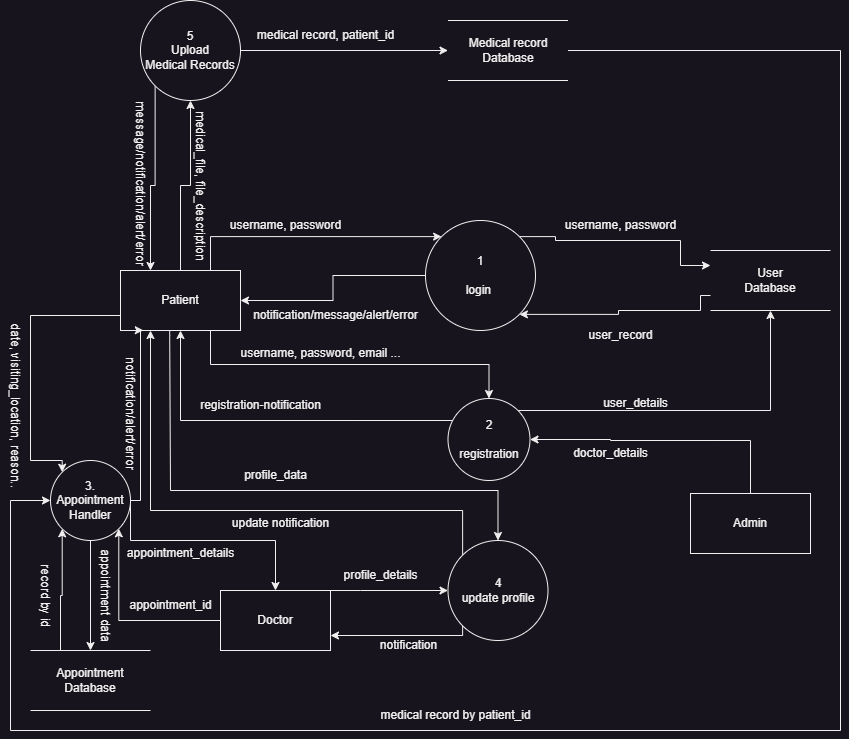
**Source or Sink:** Source or Sink is an external entity and acts as a source of system inputs or sink of system outputs.

Symbols used in DFD representation:

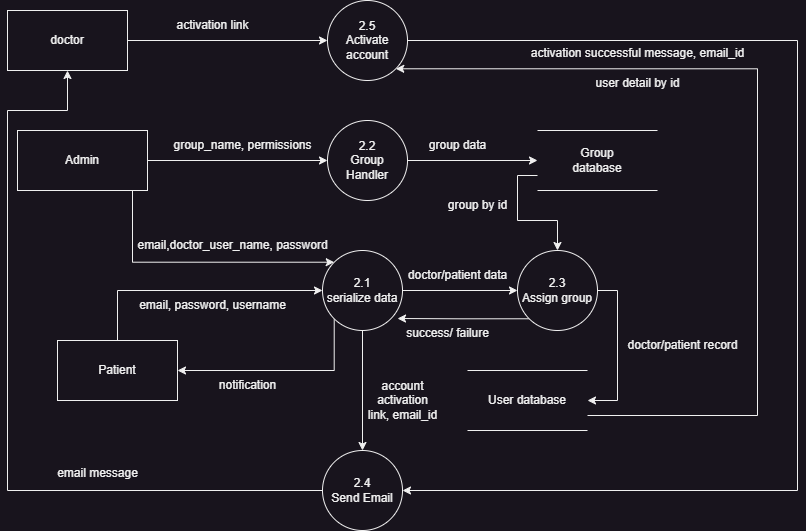
* + 1. Context Diagram/ DFD Level 0:



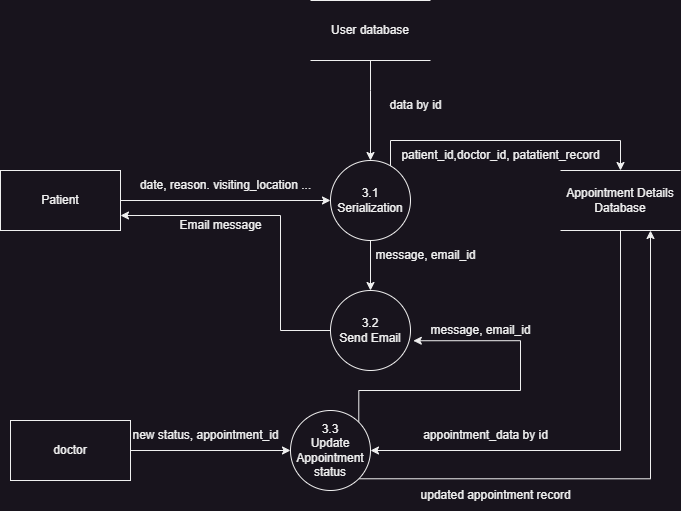
* + 1. DFD Level 1



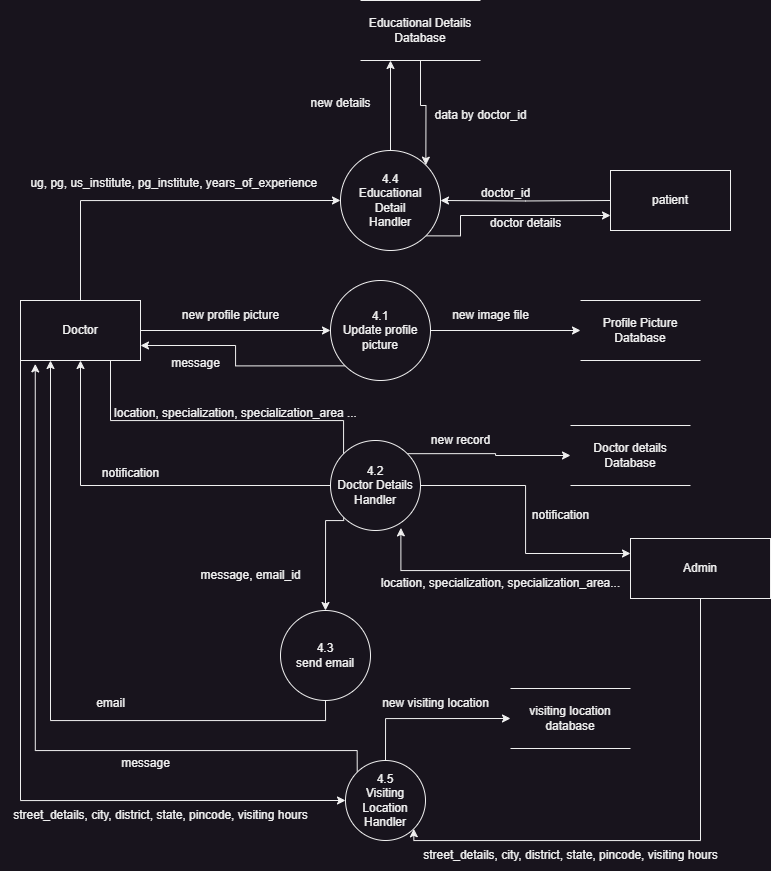
* + 1. DFD level 2
       1. registration module
          1. Registration



* + - 1. Appointment Handler Module



* + - 1. Profile handler Module

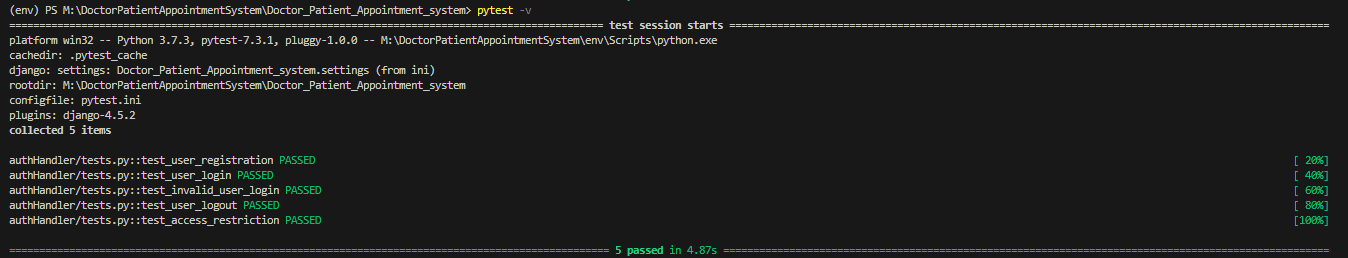


**Chapter 8**

**Testing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case id | Description | Actual result | Expected Result | Status |
| 1 | Test User Registration:  This test case verifies that a new user can successfully register an account.  It checks if the user is able to provide the required information, such as username, password, confirm\_password and email, and register successfully.  Assertions are made to ensure the user is added to the database and redirected to the appropriate page after registration. | Registrations successful,  User added | Successful Registration,  User record to be added | PASS |
| 2 | Test User Login:  This test case ensures that registered users can log in to their accounts.  It checks if the login functionality works correctly by providing valid credentials and verifying that the user is authenticated.  Assertions are made to confirm that the user is redirected to the correct page after login. | Successful login  And redirect to appropriate page | A successful login  And redirect to appropriate page | PASS |
| 3 | Test Invalid User Login:  This test case validates the handling of invalid login attempts.  It provides incorrect credentials and verifies that the system rejects the login attempt and displays the appropriate error message.  Assertions are made to ensure that the user is not authenticated and that the correct error message is displayed. | Unsuccessful login and successful redirect to login page | An unsuccessful login and redirect to login page again | PASS |
| 4 | Test User Logout:  This test case confirms that logged-in users can successfully log out from their accounts.  It simulates a user session and then triggers the logout functionality.  Assertions are made to ensure that the user is logged out and redirected to the appropriate page. | Successful logout and redirection to system’s home page | A successful logout and a redirect system’s home page | PASS |
| 5 | Test Access Restriction:  This test case checks that certain pages or functionalities are accessible only to authorized users.  It attempts to access restricted pages without logging in and verifies that the system redirects the user to the login page.  Assertions are made to ensure that unauthorized users are not granted access to restricted areas. | Un-successful page access without login and a redirection to login page | An denied access of restricted page without login and a redirection to login page | PASS |

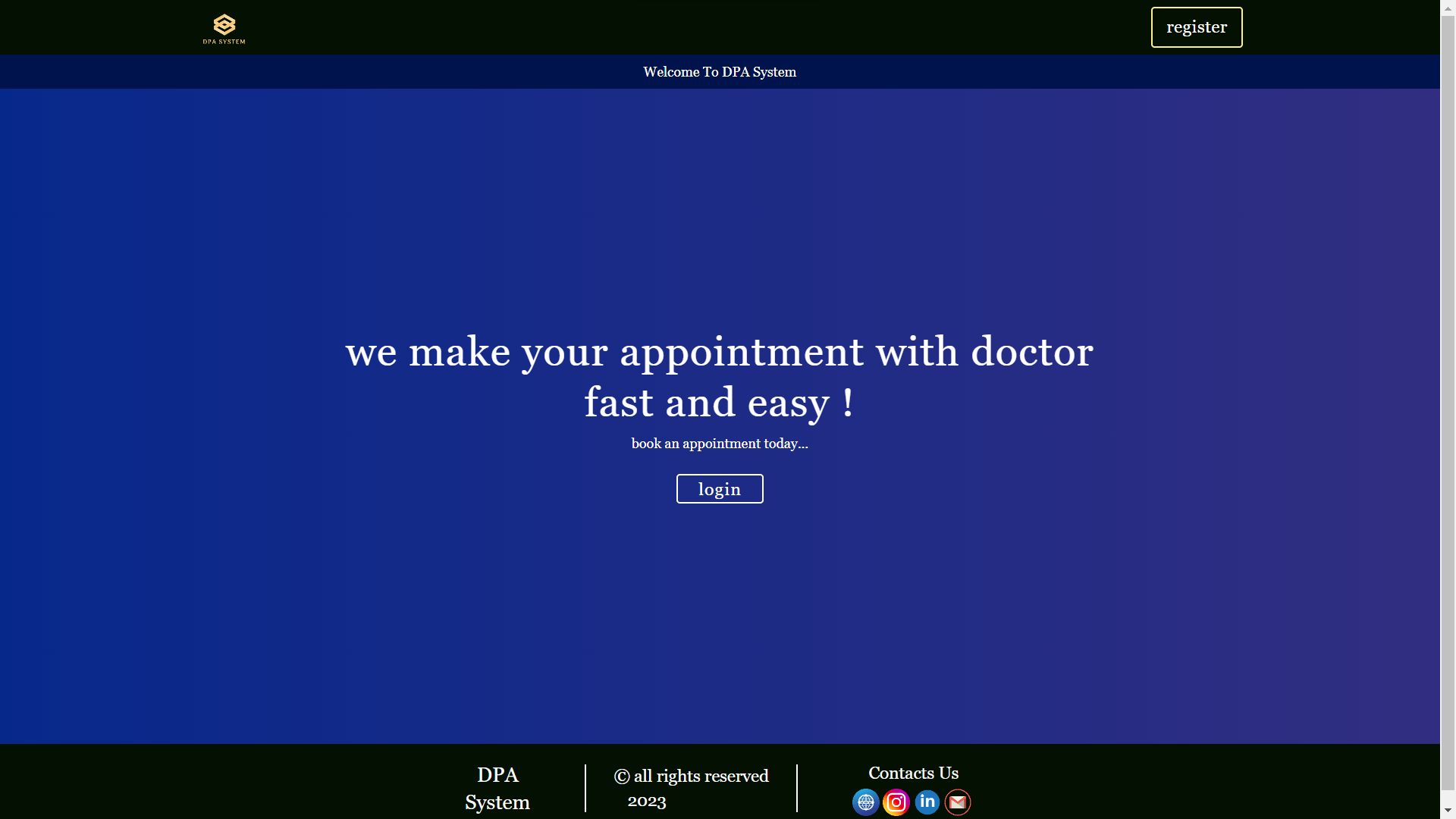
Test Output Screenshot



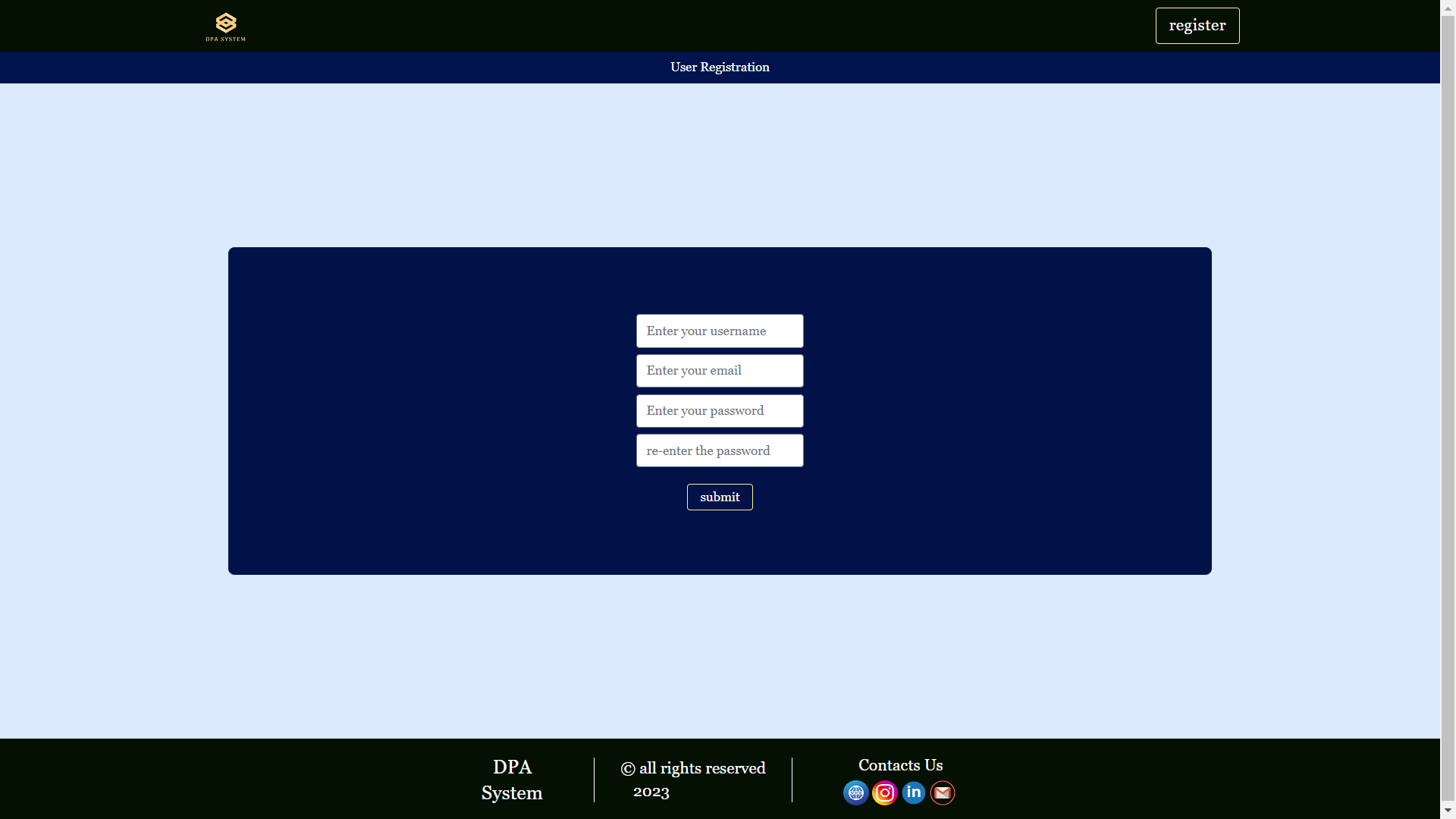
**Chapter 9**

**Results and Discussion**

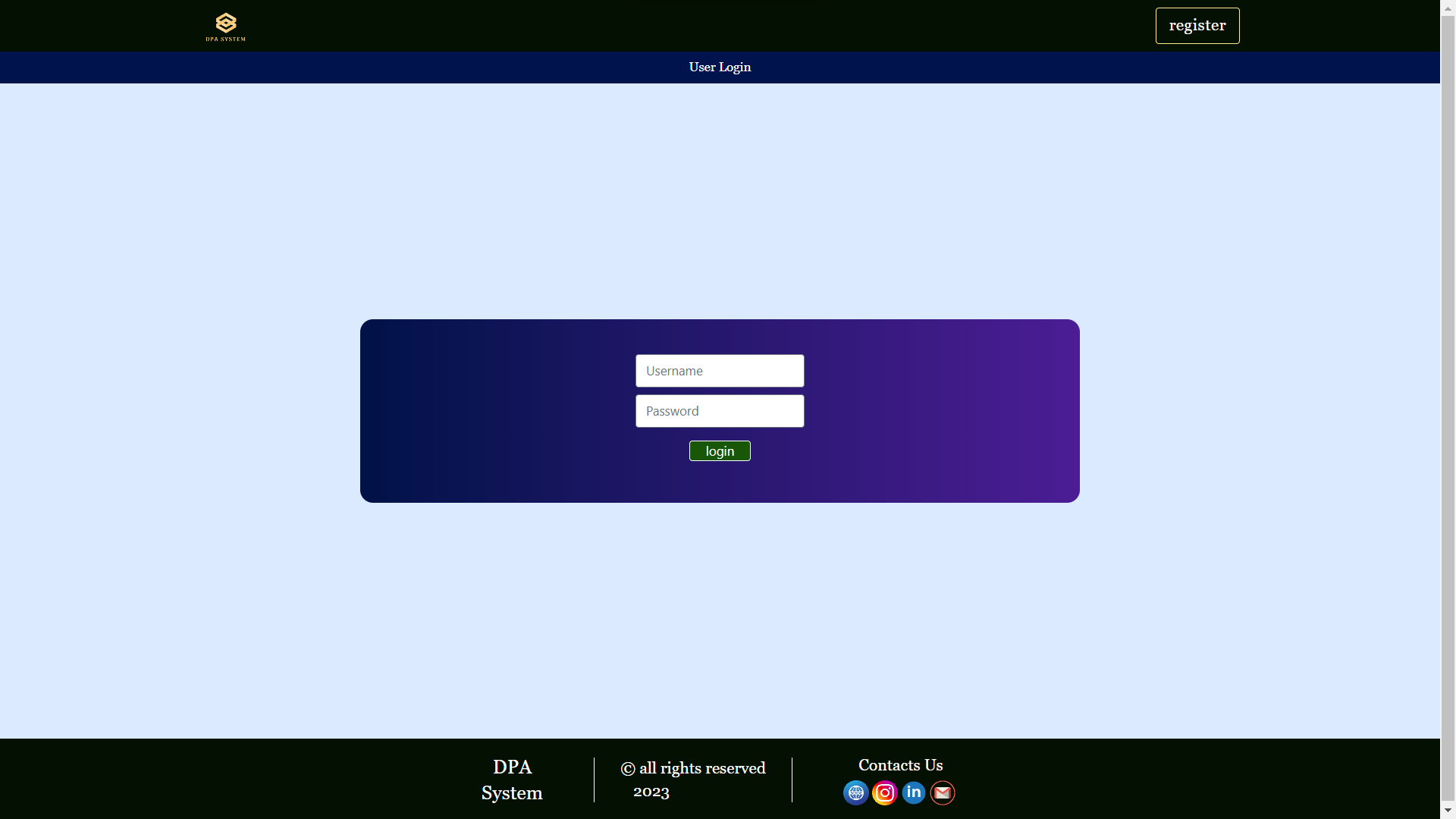
* DPAS(doctor patient appointment system) landing page



* Registration page



* Login page



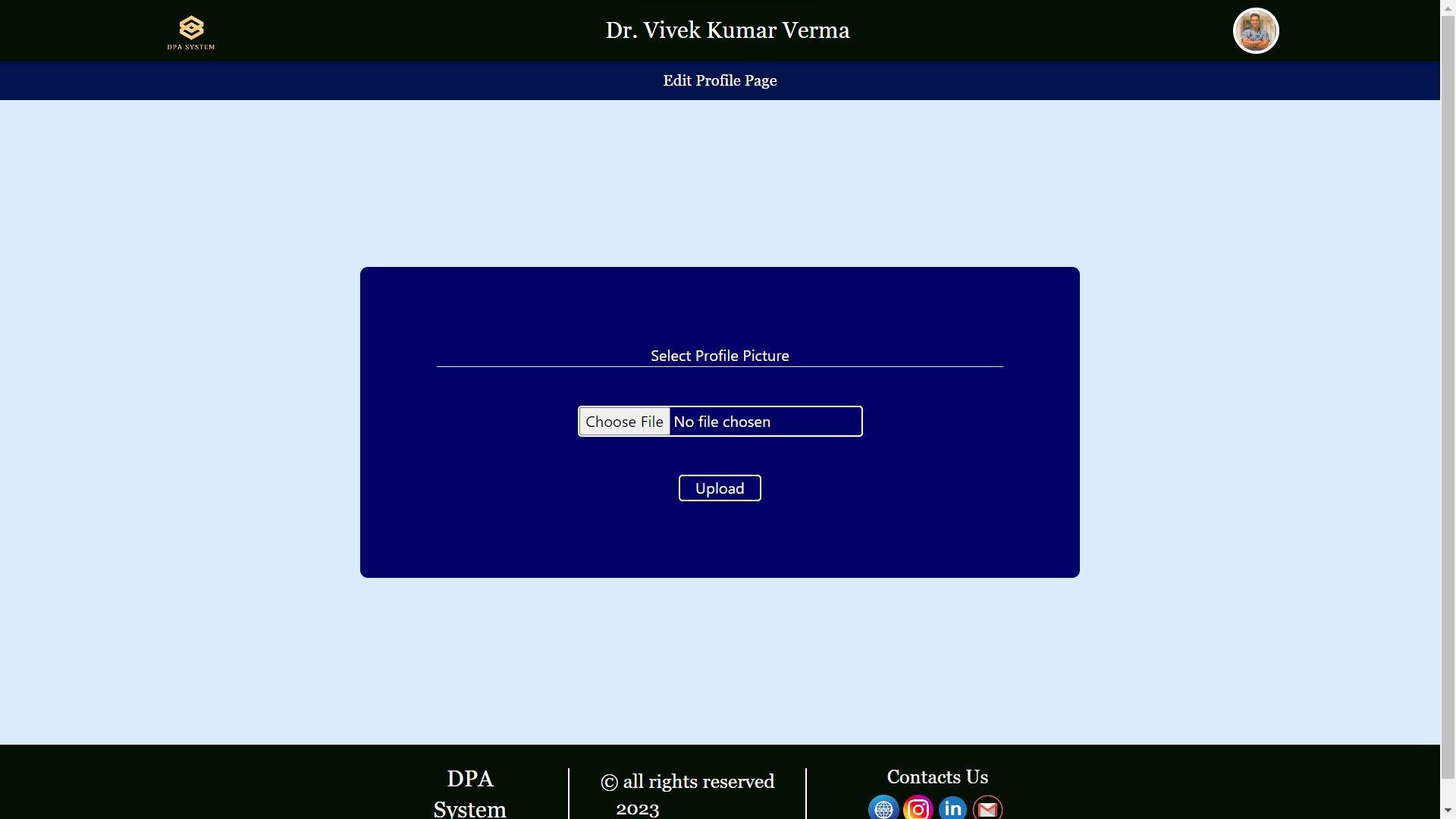
* Doctor home page



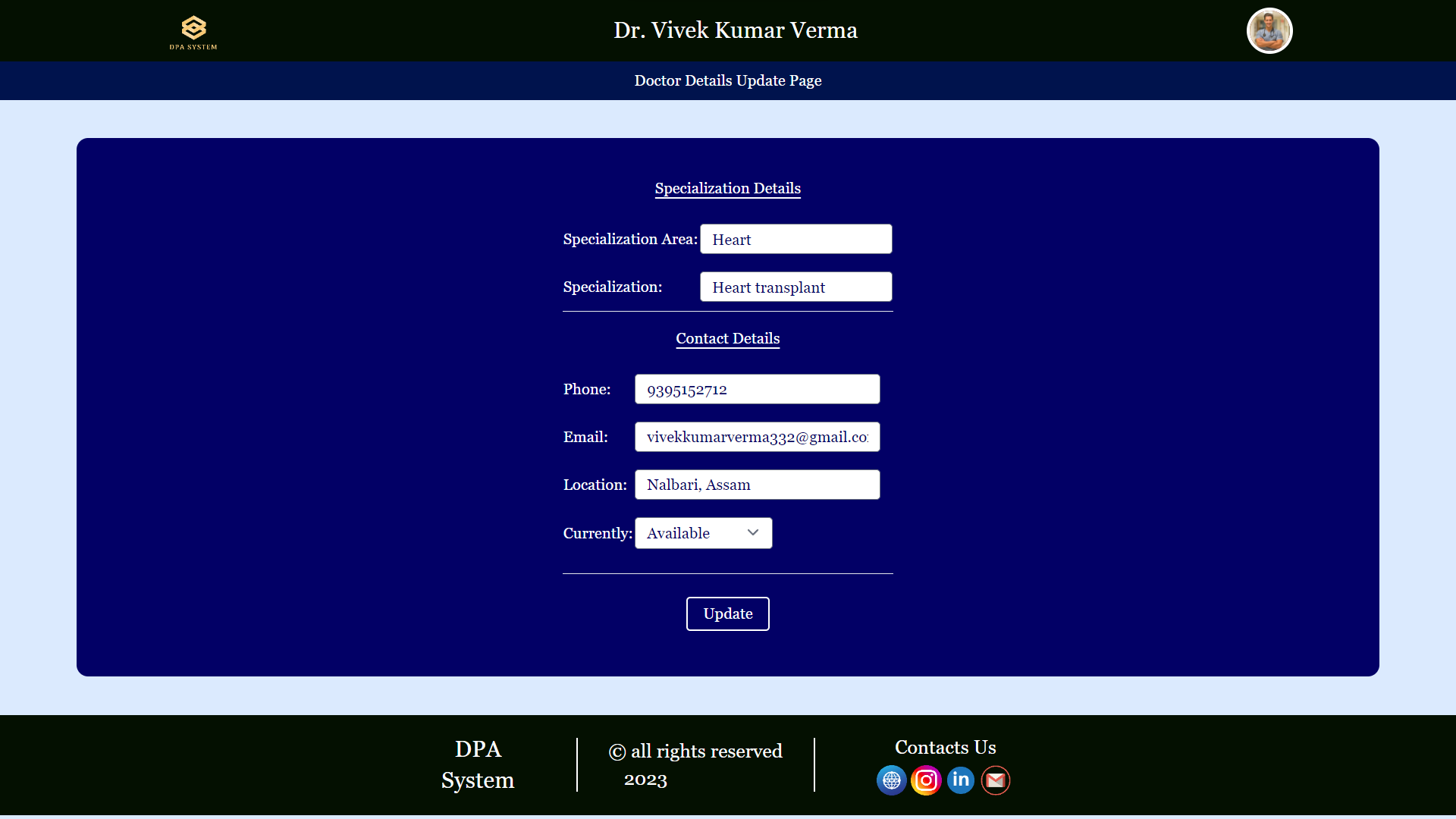
* Doctor profile page



* Doctor profile picture upload page



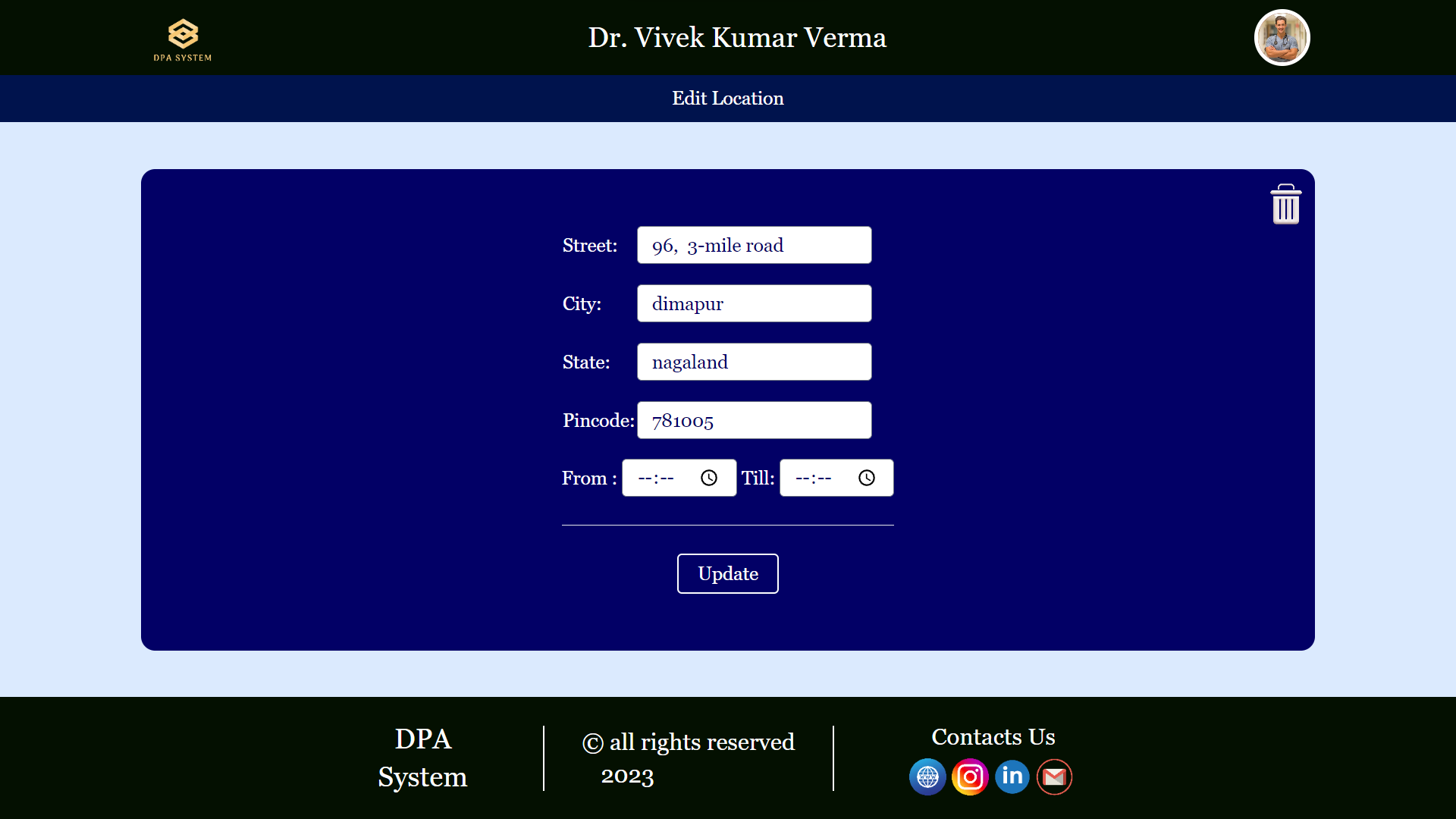
* Doctor detail update page



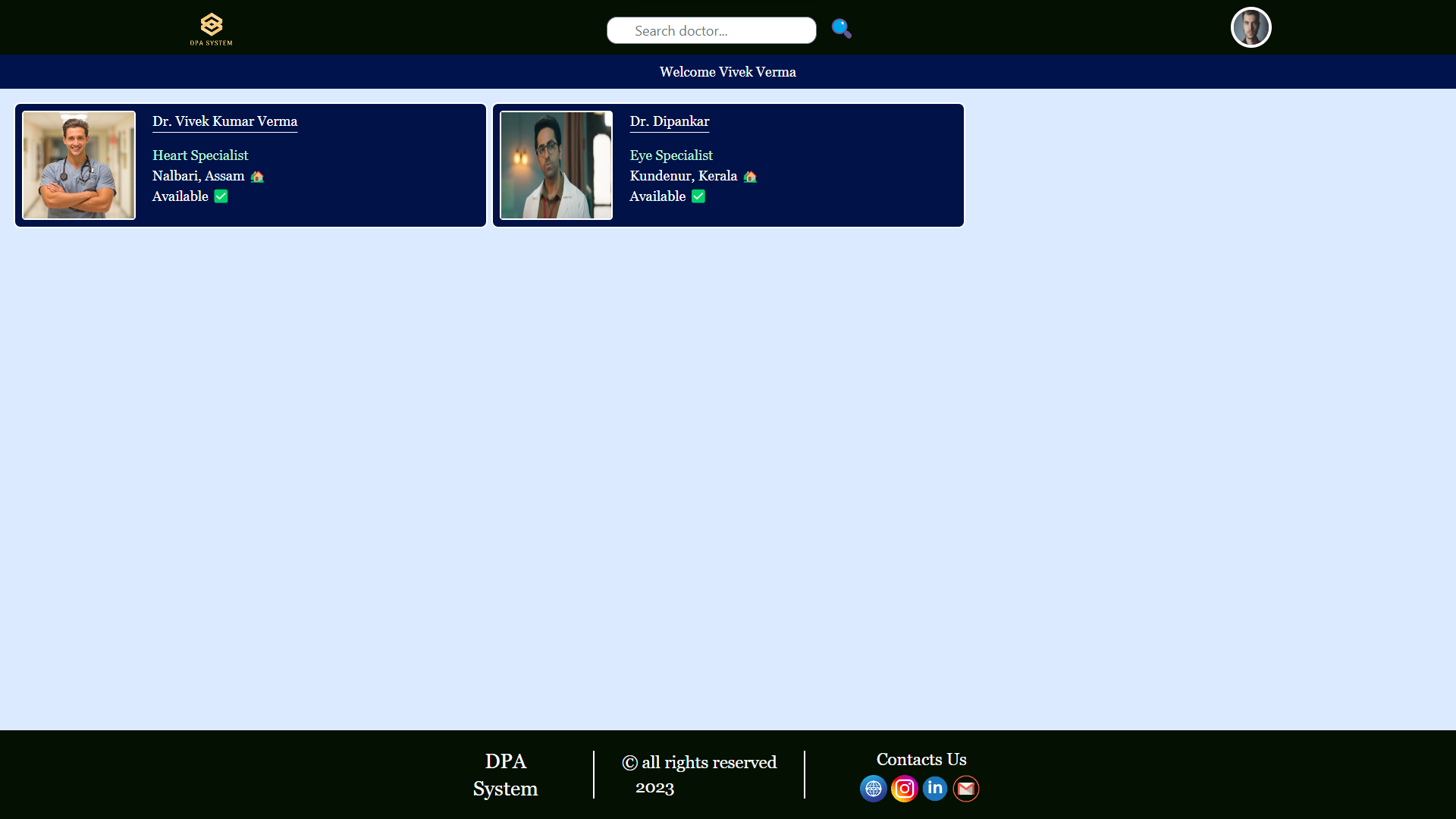
* Doctor educational detail update page



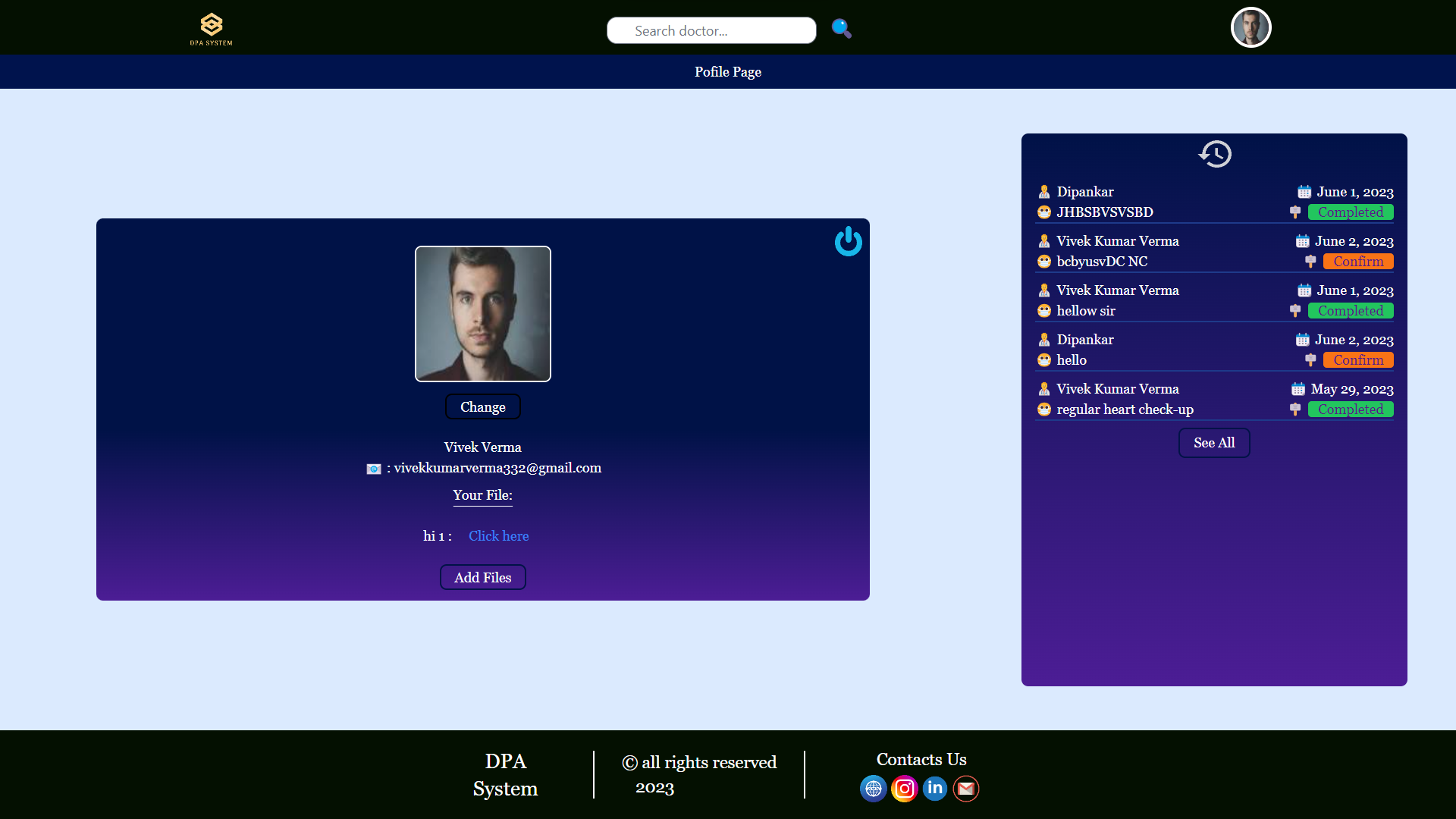
* Doctor visiting location edit page



* Patient Home page



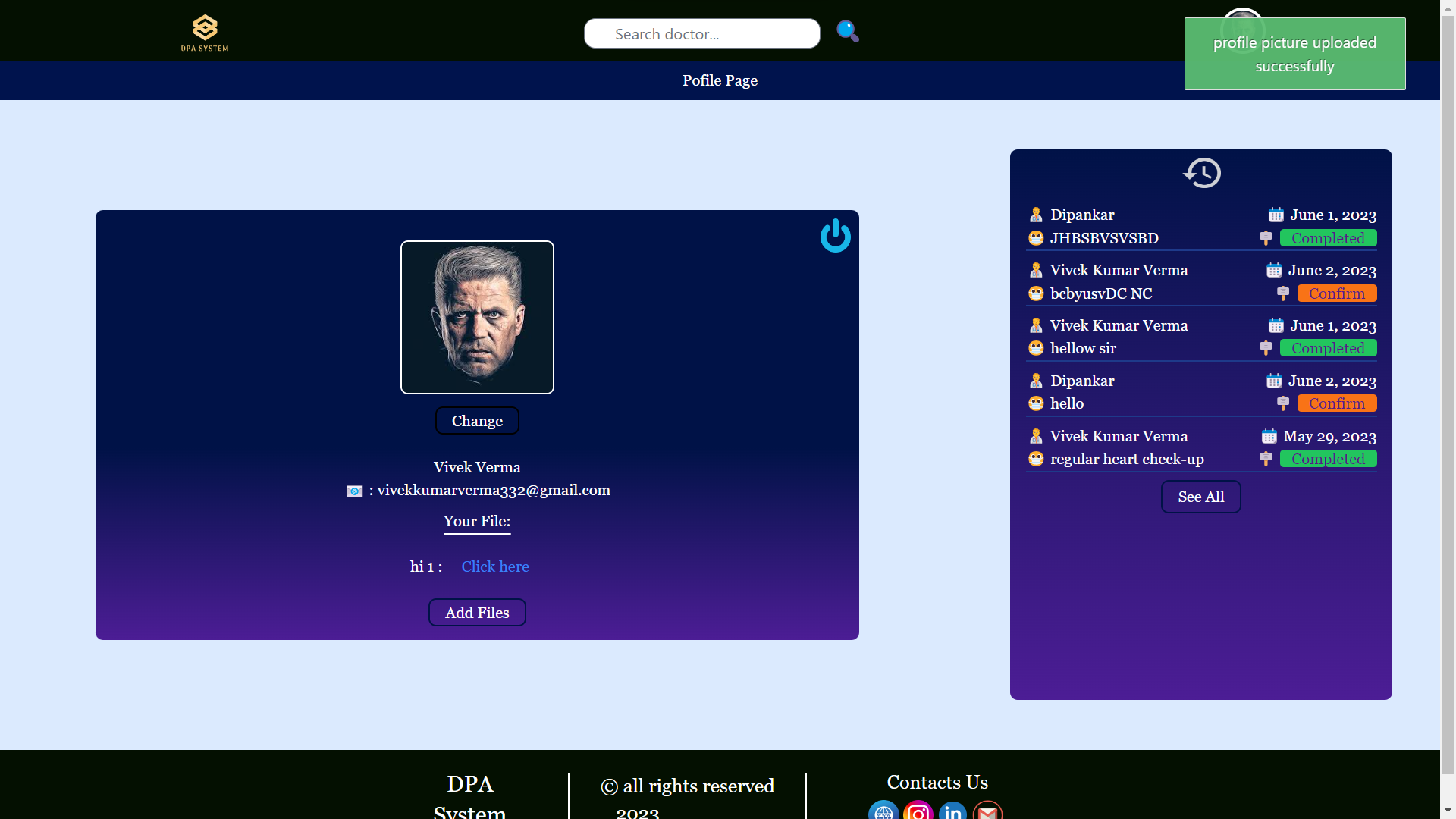
* Patient profile page



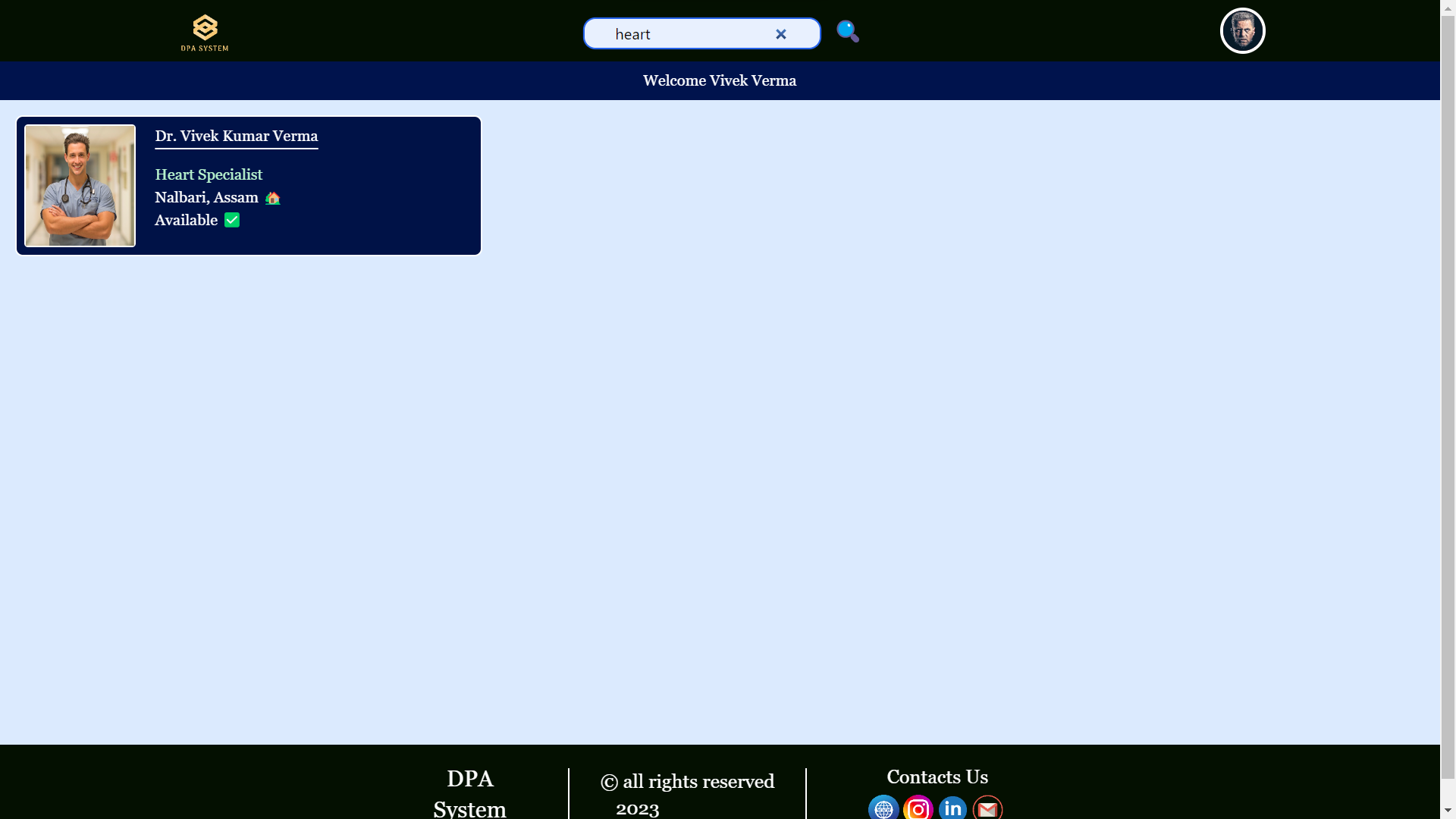
* Patient new profile picture upload page



* Patient profile after updating profile picture



* Patient home page after searching doctor



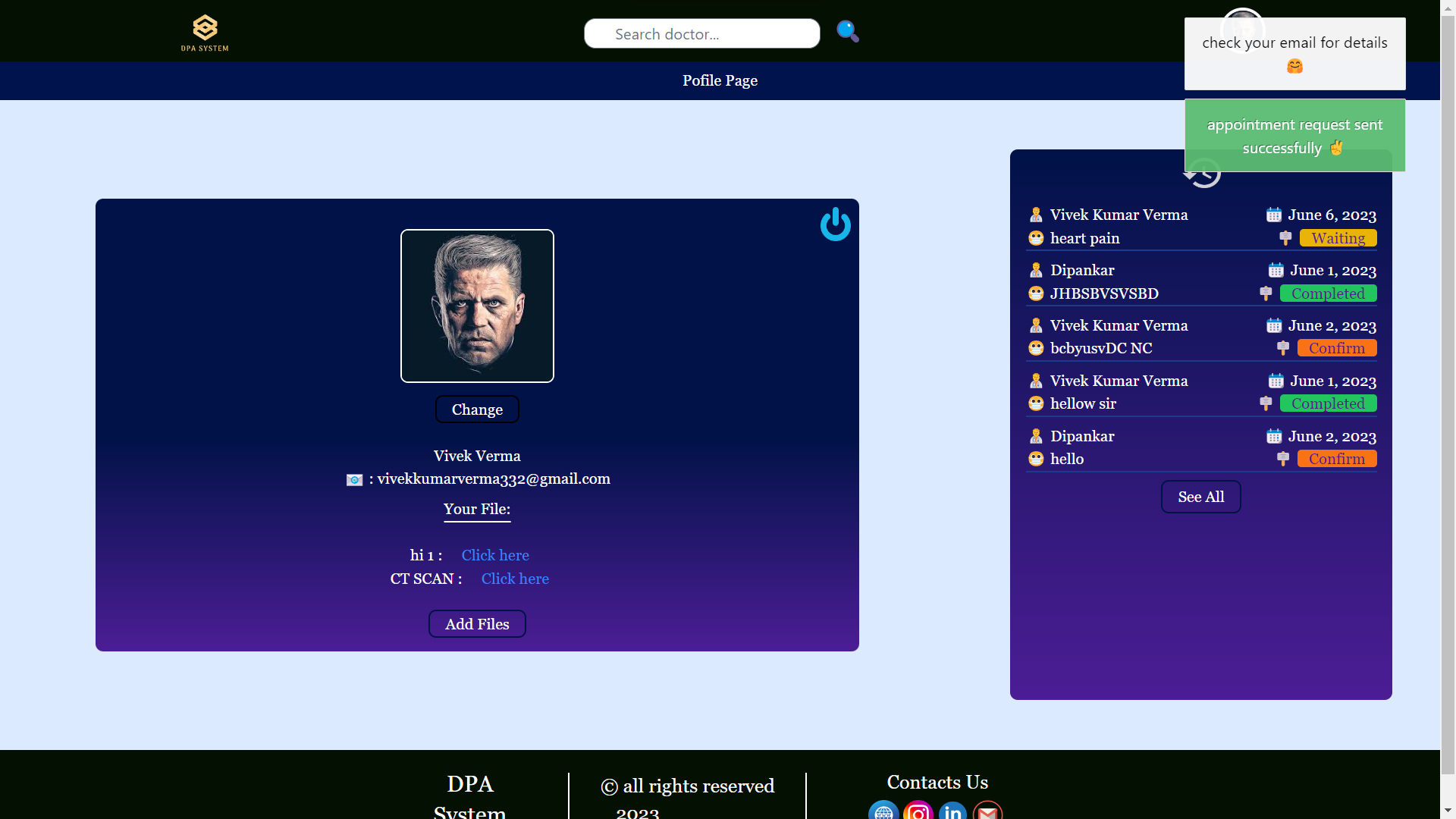
* Doctor detail page for patient



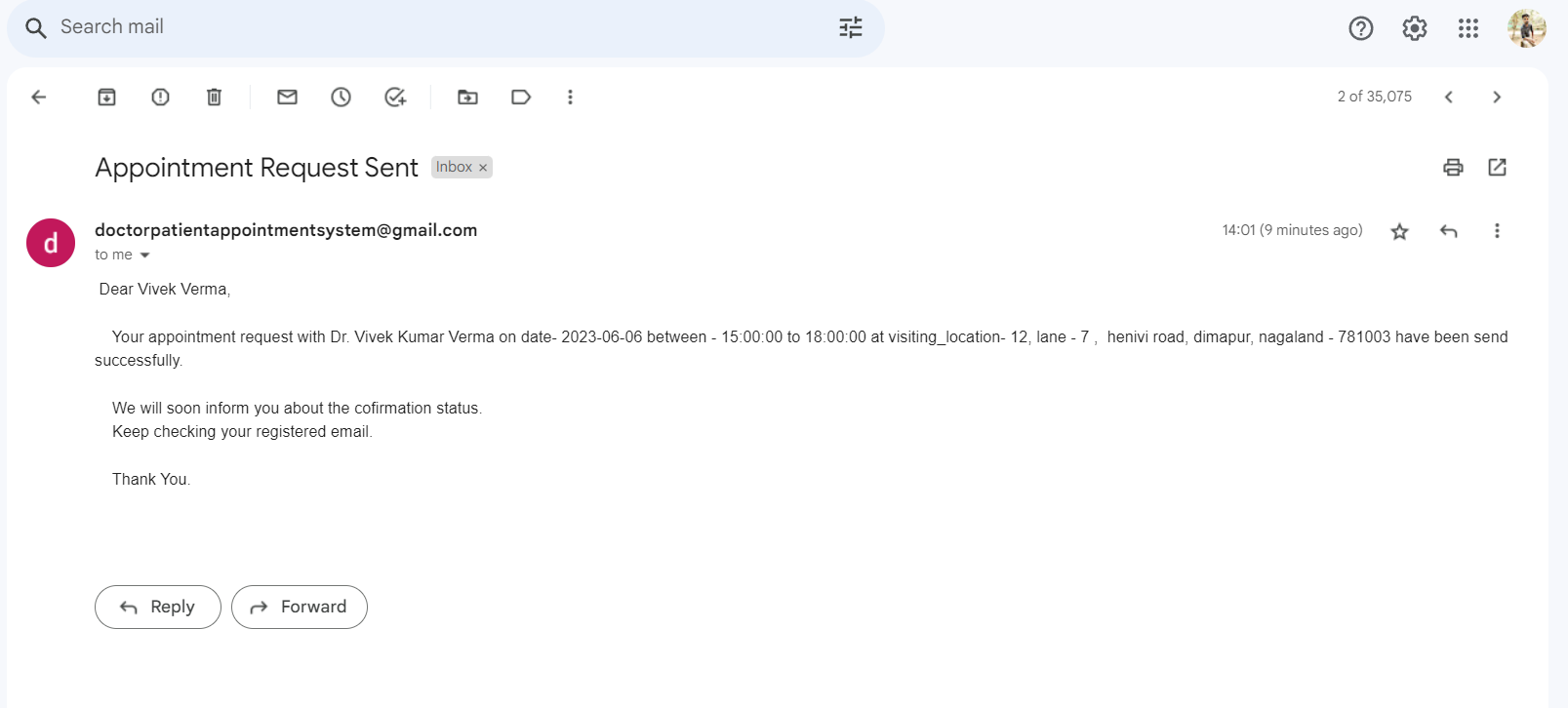
* Appointment booking page



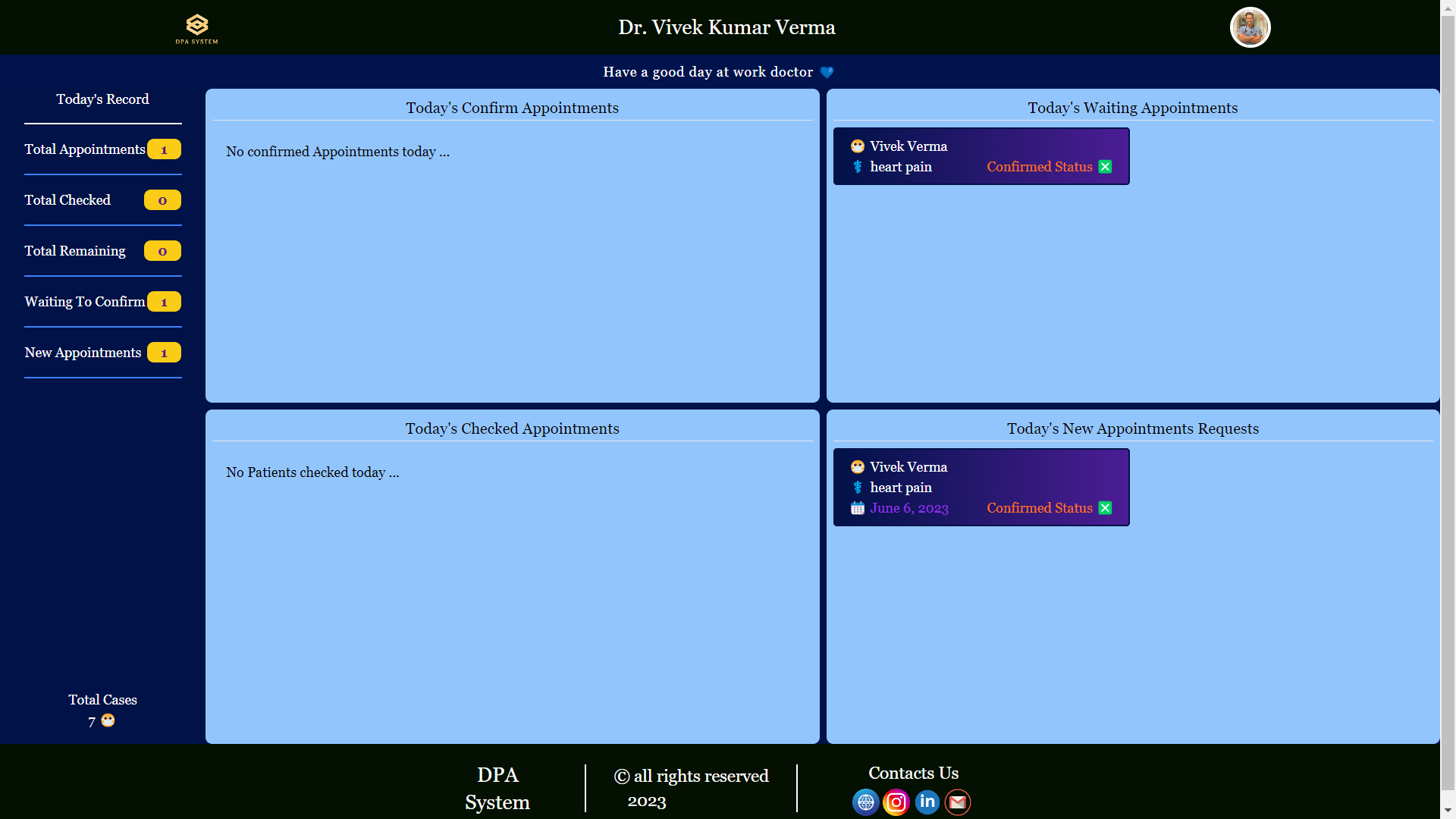
* Patient profile page with notification of successful appointment request sent.



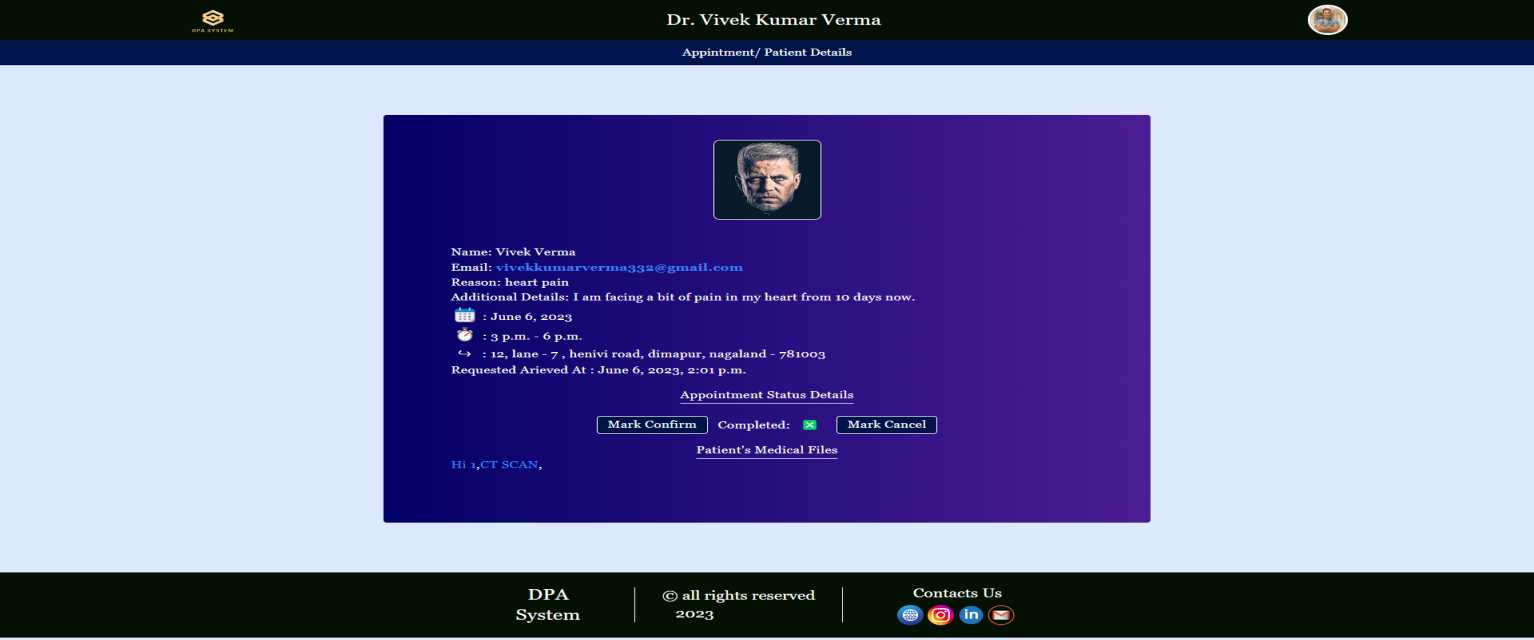
* Email for successful appointment request.



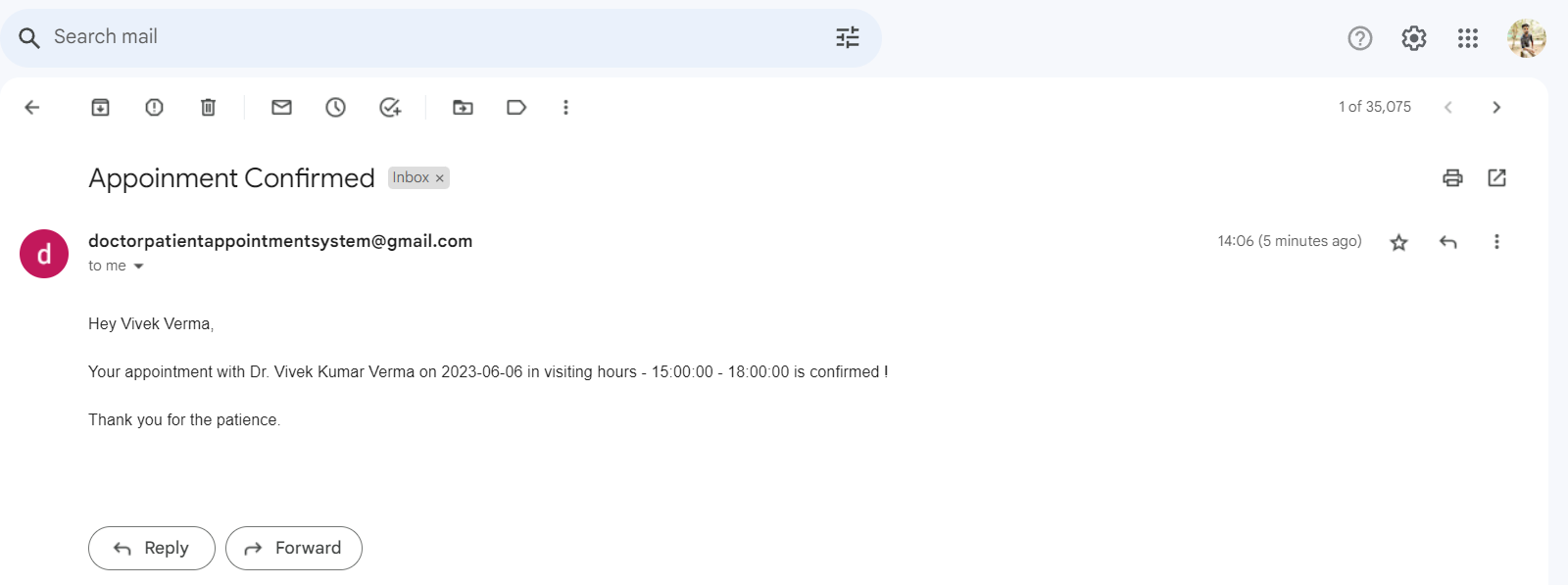
* Doctor’s home page after receiving appointment.



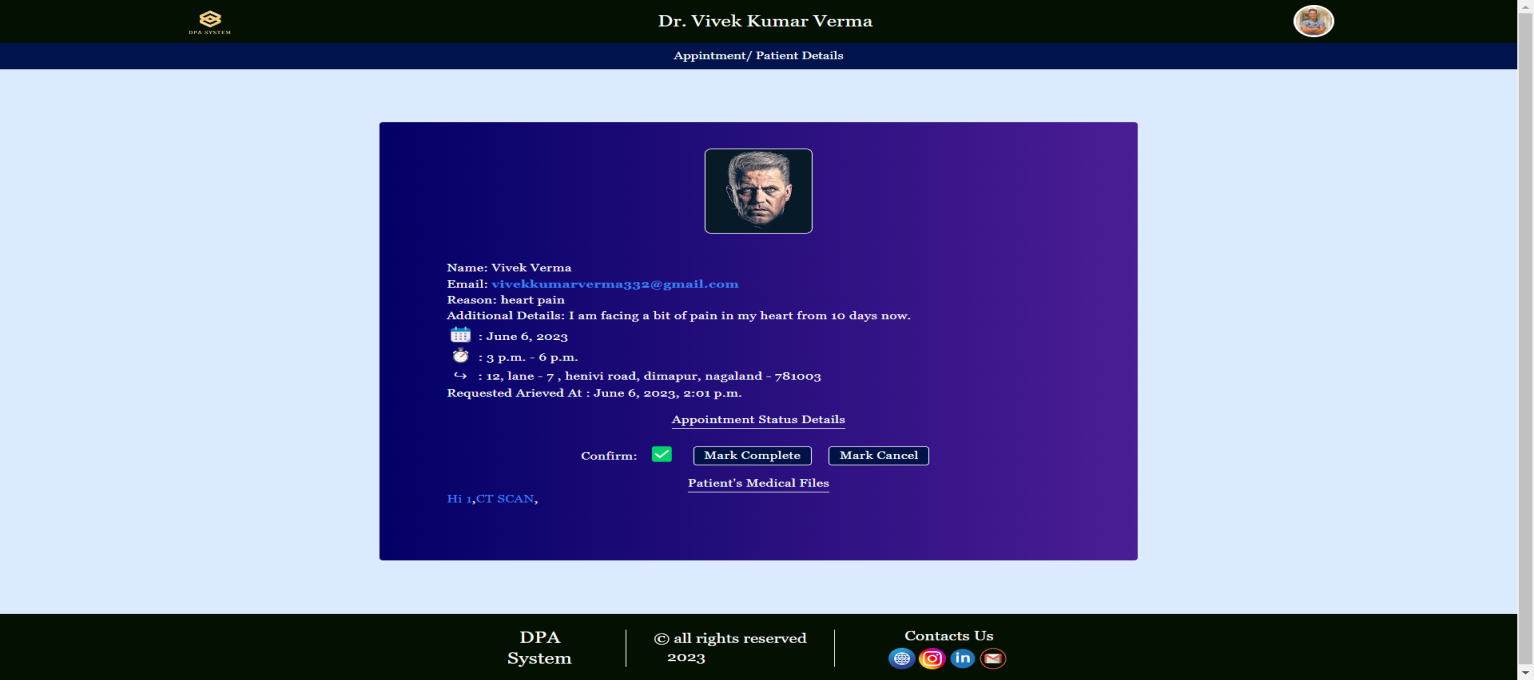
* Appointment detail page for doctor



* Appointment confirmation mail to patient.



* Appointment details after confirmation.



* Doctor’s home page after appointmnet is marked complete by the doctor.



**Chapter 10**

**Conclusion & Future Scope**

10.1 Conclusion:

In conclusion, the Doctor-Patient Appointment System developed using Django, PostgreSQL, Jinja, Sweet Alert, and SMTP mail with multithreading offers an efficient and user-friendly solution for managing appointments in healthcare settings. The system streamlines the appointment booking process, enhances communication between doctors and patients, and improves overall operational efficiency. It comes with ability of easy to scale for the large user base and ease of integration of new features.

10.2 Future Scope:

For future scope, the project can be extended to include additional features such as integration with electronic health records (EHR) systems, real-time chat functionality, and advanced analytics for appointment scheduling optimization. Furthermore, mobile applications can be developed to provide a more accessible and convenient experience for both patients and doctors. Continuous enhancements and updates can be made to improve system performance, security, and user experience.

**Chapter 11**

**Bibliography**

* [1] Django :
  + - <https://www.djangoproject.com/>
    - Bairagi, Ishan Adhikari, et al. "UNO: A Web Application using Django." *2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*. IEEE, 2021.

<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9725577&isnumber=9725386>

* + - P. V, V. P, R. K. K, S. P, O. Khan and C. N. Krishna, "A Django Web Application to Promote Local Service Providers," 2022 6th International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2022, pp. 1517-1521

<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9754099&isnumber=9753686>

* [2] SMTP (Simple Mail Transfer Protocol) :

<https://www.geeksforgeeks.org/simple-mail-transfer-protocol-smtp/>

* [3] PostgreSQL:

<https://www.postgresql.org/>

* [4] Tailwind CSS :

<https://tailwindcss.com/docs/installation/>

* [5] HTML :

<https://www.w3schools.com/html/>

* [6] Sweet Alert :

<https://common.olemiss.edu/examples/sweet-alert/>