A PROJECT REPORT

ON

"A WEB APPLICATION ON INVENTORY MANAGEMENT SYSTEM FOR SERVER CENTRE OF AIKTC"

Submitted to

UNIVERSITY OF MUMBAI

In Partial Fulfilment of the Requirement for the Award of

BACHELOR'S DEGREE IN
COMPUTER ENGINEERING

BY

KHAN AAMIR GOUS SHABANA	14CO25
ANSARI MOHAMMED AASIF AKHTAR ALI ZAITOON	16DCO46
ANSARI MD GHALIB NIZAMUDDIN AFSANA PARVEEN	15DCO54
PANDIT HERAMB SATISH VASANTI	15CO30

UNDER THE GUIDANCE OF
PROF. JAVED KHAN SHEIKH

DEPARTMENT OF COMPUTER ENGINEERING
Anjuman-I-Islam's Kalsekar Technical Campus
SCHOOL OF ENGINEERING & TECHNOLOGY
Plot No. 2 3, Sector - 16, Near Thana Naka,
Khandagaon, New Panvel - 410206
2018-2019

AFFILIATED TO

UNIVERSITY OF MUMBAI

A PROJECT II REPORT
ON

"A Web Application on Inventory Management System for Server Centre of AIKTC"

Submitted to
UNIVERSITY OF MUMBAI

In Partial Fulfilment of the Requirement for the Award of

**BACHELOR'S DEGREE IN
COMPUTER ENGINEERING**

BY

KHAN AAMIR GOUS SHABANA	14CO25
ANSARI MOHAMMED AASIF AKHTAR ALI ZAITOON	16DCO46
ANSARI MD GHALIB NIZAMUDDIN AFSANA PARVEEN	15DCO54
PANDIT HERAMB SATISH VASANTI	15CO30

UNDER THE GUIDANCE OF
PROF. JAVED KHAN SHEIKH

DEPARTMENT OF COMPUTER ENGINEERING
Anjuman-I-Islam's Kalsekar Technical Campus
SCHOOL OF ENGINEERING & TECHNOLOGY
Plot No. 2 3, Sector - 16, Near Thana Naka,
Khandagaon, New Panvel - 410206

2018-2019
AFFILIATED TO

UNIVERSITY OF MUMBAI

Anjuman-i-Islam's Kalsekar Technical Campus
Department of Computer Engineering
SCHOOL OF ENGINEERING & TECHNOLOGY
Plot No. 2 3, Sector - 16, Near Thana Naka,
Khandagaon, New Panvel - 410206

CERTIFICATE

This is certify that the project entitled

"A Web Application on Inventory Management System for Server Centre of AIKTC"

submitted by

KHAN AAMIR GOUS SHABANA	**14CO25**
ANSARI MOHAMMED AASIF AKHTAR ALI ZAITOON	**16DCO46**
ANSARI MD GHALIB NIZAMUDDIN AFSANA PARVEEN	**15DCO54**
PANDIT HERAMB SATISH VASANTI	**15CO30**

is a record of bonafide work carried out by them, in the partial fulfilment of the requirement for the award of Degree of Bachelor of Engineering (Computer Engineering) at ***Anjuman-I-Islam's Kalsekar Technical Campus, Navi Mumbai*** under the University of MUMBAI. This work is done during year 2018-2019, under our guidance.

Date: / /

(Prof. JAVED KHAN SHEIKH)
Project Supervisor

(Prof. KALPANA R. BODKE)
Project Coordinator

(Prof. TABREZ KHAN)
HOD, Computer Department

DR. ABDUL RAZAK HONNUTAGI
Director

External Examiner

Acknowledgements

We would like to take the opportunity to express our sincere thanks to our guide **Prof. Javed Khan Sheikh**, Assistant Professor, Department of Computer Engineering, AIKTC, School of Engineering, Panvel for his invaluable support and guidance throughout our project research work. Without his kind guidance & support this was not possible.

We are grateful to him/her for his timely feedback which helped us track and schedule the process effectively. His/her time, ideas and encouragement that he gave has helped us to complete our project efficiently.

We would like to express deepest appreciation towards **Dr. Abdul Razzak Honnutagi**, Director, AIKTC, Navi Mumbai, **Prof. Tabrez Khan**, Head of Department of Computer Engineering and **Prof. Kalpana R. Bodke**, Project Coordinator whose invaluable guidance supported us in completing this project.

At last we must express our sincere heartfelt gratitude to all the staff members of Computer Engineering Department who helped me directly or indirectly during this course of work.

Khan Aamir Gous Shabana

Ansari Mohammed Aasif Akhtar Ali Zaitoon

Ansari Md Ghalib Nizamuddin Afsana Parveen

Pandit Heramb Satish Vasanti

Project II Approval for Bachelor of Engineering

This project entitled *Ä Web Application on Inventory Management System for Server Centre of AIKTC¨* by ***Khan Aamir Gous Shabana, Ansari Mohammed Aasif Akhtar Ali Zaitoon, Ansari Md Ghalib Nizamuddin Afsana Parveen, Pandit Heramb Satish Vasanti*** is approved for the degree of ***Bachelor of Engineering in Department of Computer Engineering.***

Examiners

1. ....
2. ....

Supervisors

1. ....
2. ....

Chairman

....

Declaration

We declare that this written submission represents our ideas in our own words and where others ideas or words have been included, We have adequately cited and referenced the original sources. We also declare that We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Khan Aamir Gous Shabana
14CO25

Ansari Mohammed Aasif Akhtar Ali Zaitoon
14DCO46

Ansari Md Ghalib Nizamuddin Afsana Parveen
15DCO54

Pandit Heramb Satish Vasanti
15CO30

ABSTRACT

A Web Application on Inventory Management System for Server Center of AIKTC.

Considering the existing experience in AIKTC Server management center, we have analyzed basic aspects of manually maintained Inventory Management. As first step, we have detected the need for developing the proposed system, covering configuration management, functional area. The main goal of this system is to collect, process and store inventory data through users. This project minimizes the paper work, human mistakes, manual delay and speed up process. Inventory Management System will have the ability to track purchase and available inventory, tells admin when it's time to reorder and how much to purchase. Inventory Management System for server center focuses in the area of Inventory control and generates the various required reports that will increase efficiency and services and minimize the errors and reduce the manual maintenance and maintain data integrity.

Khan Aamir Gous Shabana
14CO25

Ansari Mohammed Aasif Akhtar Ali Zaitoon
14DCO46

Ansari Md Ghalib Nizamuddin Afsana Parveen
15DCO54

Pandit Heramb Satish Vasanti
15CO30

B.E(Computer Engineering)
University of Mumbai

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This project promises to reduce the workload of manual maintenance of AIKTC's Inventory management in Server Center with an approach of handling the Stock, Notifying the Administrator in advance about the low stock for specific components, To generate the reports as and when necessary by customizing them based on requirement and Generating reports in .pdf or .docx format.

For Maintaining Inventory in AIKTC is a hard thing in its own, considering the fact that the administrator has to maintain different manually maintained data in a register and to generate report depending on the register, it is a difficult task in its own, so to ease this load, the proposed system will be able to maintain stock, give new purchase order to the store, maintain information of the equipment purchased over time and also to maintain the information of the maintenance work carried out on the specific equipment.

1.1 Purpose

If we want to manage Inventory of Server Center of AIKTC, The Administrator and the other users faces many difficulties. To overcome this difficulties, We are implementing this system which will not only help us to manage inventory of server center, but also will be helpful in maintaining required stock for specific components and information regarding them. Requesting for some components and display the request processing time to both- the requester and the Administrator, To generate required reports for these requests based on its types and to maintain faulty components which will maintaining data for the same.

The purpose of this project is to create better understanding in redefining the inventory management of server centers out of the old traditional way, which will help to increase the transparency in between the users of the inventory.

1.2 Project Scope

In Inventory Management of AIKTC, The Scope of this project is to improve connectivity among the various departments for Server related services and issues. To increase Transparency, Security, To make work hassle free, To maintain stock, Give new purchase order to the store, To maintain information of the equipment purchased and maintenance work carried out on the specific equipment.

Hence, the Proposed System is an Inventory Management System which will primarily focus on the Stock maintenance, process requests and to generate reports based on the requests type as per the requirements.

1.3 Project Goals and Objectives

1.3.1 Goals

The Goals of this project is to enhance the Inventory Management in Server center to make it more secure, transparent and easy to use. This will help us to analyze the reports and also to keep a check on the available inventory.

1.3.2 Objectives

This project will help the Administrator of Server Center in maintaining and storing components data, Reviewing requests and generating reports for the requests.
Following are the objectives of the system:

- To study the existing manual work of Inventory Management.
- To create Data Flow Design depending upon the existing work.
- To develop User Interface using Django Framework.
- To design Database using Django Framework.

1.4 Organization of Report

1. In Chapter 1: We have considered project overview under which we have explained various important terminologies like Introduction of the project, Purpose of the project, Scope, Goals and Objectives.
2. In Chapter 2: We have discussed about various paper that we have referred for our project. We have mentioned the description, pros and cons and how the overcome the problem under every paper. A total of three paper have been referred.
3. In Chapter 3: we have discussed about the project planning under it we have discussed about members and their capabilities, their roles and responsibility in this project, the different assumptions we have made for the project, the constraints we have with the project, the approach we have used for project management, the project budget and timeline.
4. In Chapter 4: We have discussed the Software Requirement Specifications under it, we have explained various perspectives, features and constraints of our project. We have also explained various interfaces such as User Interface, Hardware Interface, Software Interface, Communication Interface; And also we have discussed the non-functional requirements such as Performance Requirements, Safety Requirements and Security Requirements.
5. In Chapter 5: we can see the system design and architecture various diagram can be seen in this chapter which represent the software , diagram includes our system architecture, sequence diagram, component diagram, class diagram and deployement diagram.
6. In Chapter 6: we have discussed about the implementation details the assumption and dependencies this part contains details of the implementation of methodology that we discussed earlier.
7. In Chapter 7: we have shown the test cases and result along with analytic discussion this part contains the result of the output of the project and also the software quality attributes.
8. Chapter 8 is a step by step guide about using the final product and screenshots of our project.

9. Chapter 9 is the closure to the book and tries to conclude the work in the project and also mentions the future scope as to where it would be used.

Chapter 2

Literature Survey

2.1 Inventory Management System for Water Supply Network.

We have made comparison between basic aspects of telecommunications network management and water supply network management. we have detected the need for developing water supply network inventory system, covering configuration management functional area. The main goal of this system is to collect, process and store inventory data through graphical user interface, to be presented to network operating personnel. Related work is researched and briefly presented. Complete network structure is modelled as information model in Data Warehouse. Graphical user interface, developed in web technology with Google maps as background, with basic knowledge integrated, is presented. Some basic functions of GUI (graphical user interface) are discussed, such as: logging of network maintenance and repair actions, network planning support and reporting.

2.1.1 Advantages of Paper

a. It combines the actual situation and management experience of the enterprise.

b. It combines google maps API to detect anolamy in the water supply network thus making it easier to find the fault area and resolve that fault.

2.1.2 Disadvantages of Paper

a. These systems is harder to implement since it covers wide area of water supply network.

b. Implementation of this scale of project needs lots of investments and need to be implemented with proper care.

2.1.3 How to overcome the problems mentioned in paper

a. Since the proposed system is being implemented over a network of an institution, It is easier to implement and maintain the system.

2.2 Web based Warehouse Management System.

Web-based warehouse management system is designed to solve the problem is caused by the simple and static way that the traditional enterprise warehouse management system used, which is unable to ensure the problem of efficient utilization of various enterprises resources and to provide companies with online, real-time insight into their warehouse operations and inventory availability. With the use of JAVA, XML and other technologies, the system promotes visibility and the exchange of information among the employees, customers, distributors, and manufacturers.

2.2.1 Advantages of Paper

a. This project combines the actual situation and management experience of the enterprise.

b. This project manages and traces customers orders, procurement orders, and integrated management of warehouse accurately and efficiently.

2.2.2 Disadvantages of Paper

a. These system may not be suitable for small organizations and businesses.

2.2.3 How to overcome the problems mentioned in Paper

a. The proposed system will be suitable for small organizations and businesses because any new module and update can be easily implemented as per the user's requirements.

2.3 Research and Design of The Intelligent Inventory Management System Based on RFID.

This system introduces the characteristics and basic application of RFID technology, analyses the data flow of intelligent inventory system from the perspective of business and function, then puts forward the specific framework programs and function modules of intelligent inventory management system based on IOT RFID technology, focuses on elaborating the design and implementation process of the intelligent inventory system. The system realizes full control and management of all products, faster in/out warehouse and dynamic inventory, utilizes warehouse efficiently and improves the capacity of warehouse by effectively combining with the ERP system in enterprise.

2.3.1 Advantages of Paper

a. This project uses the RFID technology for tracking of components within the warehouse.

b. This helps the enterprise to effectively manage logistics costs, and reducing labor costs, improving the operation accuracy and storage efficiency, etc.

2.3.2 Disadvantages of Paper

a. These system may not be suitable for small organizations and businesses.

b. This system is not cost efficient and development time for this system is lengthy based upon the wide range of APIs' being used in this Inventory System.

2.3.3 How to overcome the problems mentioned in Paper

a. Our Proposed system is cost efficient and can be customized based upon the User's requirements. Also this system is suitable for small organizations.

2.4 Technical Review

Django Framework

Django is a Python-based free and open-source web framework, which follows the model-view-template (MVT) architectural pattern. It is maintained by the Django Software Foundation (DSF), an independent organization established as a 501(c)(3) non-profit.

Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes reusability and "pluggability" of components, less code, low coupling, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models.

2.4.1 Advantages of Technology

a. A lightweight and standalone web server for development and testing

b. A form serialization and validation system that can translate between HTML forms and values suitable for storage in the database

c. A template system that utilizes the concept of inheritance borrowed from object-oriented programming

d. An interface to Python's built-in unit test framework

2.4.2 Reasons to use this Technology

a. Django is an open-source framework for backend web applications based on Python — one of the top web development languages. Its main goals are simplicity, flexibility, reliability, and scalability.

b. Django has its own naming system for all functions and components (e.g., HTTP responses are called "views"). It also has an admin panel, which is deemed easier to work with than in Lavarel or Yii, and other technical features, including:

  - Simple syntax.
  - Its own web server.

- MVC (Model-View-Template) core architecture.
- HTTP libraries.
- A Python unit test framework.

Chapter 3

Project Planning

3.1 Members and Capabilities

SR. No	Name of Member	Capabilities
1	Aamir Khan	Database
2	Aasif Ansari	Database, Backend
3	Ghalib Ansari	Database, Backend, UI Design
4	Heramb Pandit	Frontend

Table 3.1: Table of Capabilities

3.2 Roles and Responsibilities

SR. No	Name of Member	Role	Responsibilities
1	Aamir Khan	Team Leader	Database
2	Aasif Ansari	Member	Database, Backend
3	Ghalib Ansari	Member	Database, Backend,UI Design
4	Hermab Pandit	Member	Frontend

Table 3.2: Table of Responsibilities

3.3 Assumptions and Constraints

3.3.1 Assumptions

1. Data provided by the user is true.

2. The requests made for either issue component or maintenance request is genuine.

3.3.2 Constraints

1. Administrator not available for the day.

2. The waiting time for issue new component may vary depending upon supplier.

3. Maintenance engineer not available for pending maintenance requests.

3.4 Project Management Approach

We have use Agile methodology for the development of this project.AGILE methodology is a practice that promotes continuous iteration of development.The Agile Project Management Process is a value-centered methods of project management that allows projects to get processed in small phases or cycles. This methodology is one that is extremely flexible and error can be fixed in the middle of the project.Close communication with the user representative to understand the features. It minimizes overall risk and allows the product to adapt to changes quickly.

3.5 Ground Rules for the Project

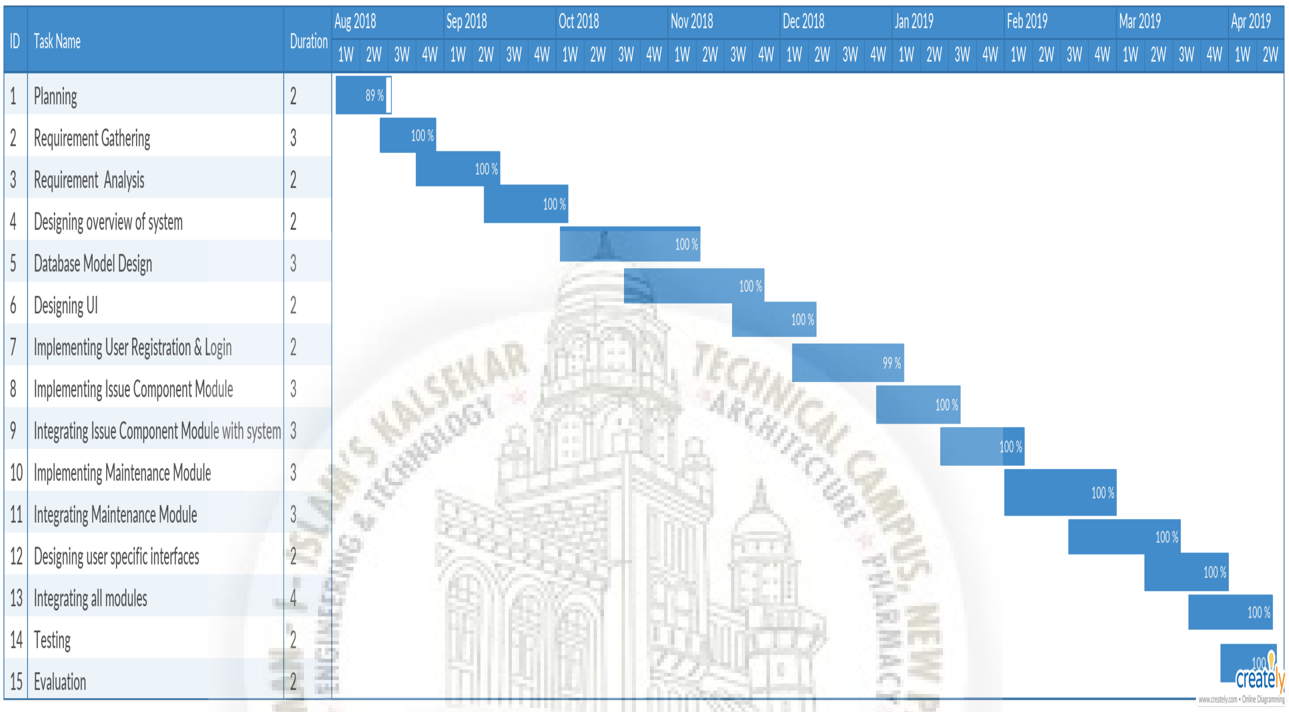
1. We treat each other with respect.
2. We intend to develop personal and professional relationships to enhance trust and open communication
3. As team members, we will pitch in to help where necessary to help solve problems and catch-up on behind schedule work.
4. One person talks at a time; there are no side discussions.
5. Additional meetings can be scheduled to discuss critical issues or tabled items upon discussion and agreement with the team leader.

3.6 Project Budget

The budget for the project is very low as we are using open source tools and services. The following are the tools and softwares we have used:

1. Django Framework based on Python language(Open Source) and SQLite3 Database.
2. Visual Studio Code(IDE).

3.7 Project Timeline



ID	Task Name	Duration
1	Planning	2
2	Requirement Gathering	3
3	Requirement Analysis	2
4	Designing overview of system	2
5	Database Model Design	3
6	Designing UI	2
7	Implementing User Registration & Login	2
8	Implementing Issue Component Module	3
9	Integrating Issue Component Module with system	3
10	Implementing Maintenance Module	3
11	Integrating Maintenance Module	3
12	Designing user specific interfaces	2
13	Integrating all modules	4
14	Testing	2
15	Evaluation	2

Figure 3.1: Project Timeline

Chapter 4

Software Requirements Specification

4.1 Overall Description

4.1.1 Product Perspective

The Web application for Inventory Management System for Server Center of AIKTC consists of the following perspective.
User's Perspective :

1. The users can be able to make a request for issuing a new component from the server inventory.
2. The users can be able to raise a maintenance request for a faulty component which was previously issued by the Administrator.

Administrator's Perspective :

1. The Administrator can be able to add items in inventory and can be able to see available inventory, also the administrator can add users, delete users and can generate reports as required.
2. The Administrator can issue an item which has been requested by the users and can assign a maintenance engineer for a maintenance job.

4.1.2 Product Features

The features of the products are mentioned below:

1. This product will make it easier for the users to make a request and keep a track on it.
2. The users can make a request and can follow up on the same.
3. It will be easier for the Administrator to keep track of the available inventory and manage the requests for the same.

4.1.3 User Classes and Characteristics

Users of this Web Application includes the Administrator, the requester, and the maintenance engineer.

The Administrator is the primary and most important user of the application, the administrator will add users, delete users, can add items to the inventory, edit the inventory and can view the available inventory, also they can Grant a request for either issuance of new component or they can revoke a request for the same. Also the Administrator can assign a maintenance engineer for a maintenance request received from the users of the application. In the end, they can generate the reports for the same was and when necessary.

The Requester can be able to request for issuing a new component, or they can request for maintenance of an existing component which is faulty. These users can see the status of their requests once they have logged in.

The maintenance engineers will be assigned by the administrator once a maintenance request has been encountered. They can view the maintenance job in their portal once they are logged in and can update the same once the fault has been cleared or otherwise.

4.1.4 Operating Environment

The Web Application will operate in the Intra Network of AIKTC, The Server Administrator of the AIKTC will host the Web application on the Intranet of the AIKTC and only the users inside the premisis of AIKTC can be able to access this system. Further required specification are mentioned below:

1. On the Client's side, any browser with HTML5 support (preferably Google Chrome or Mozilla Firefox).
   Any Operating System which supports the web browsers is required, preferably Linux or Windows.
2. On Server's side, any operating system which has support for cloud hosting services and a good processor (preferably Intel Xeon) which can take the load of the servers.

4.1.5 Design and Implementation Constraints

The major constraints that can occur during the design and implementation of this system are:

1. Designing the software in such a way that the user can understand how to use the system, for example, how to raise a issue request and how to view it once it is issued.
2. Implementing constraints includes handling the long awaited issue and maintenance requests as per the availability of the components and the maintenance engineer, etc.
3. Also to make sure the system is up and running everytime, we have to make sure that we have proper internet connection.

4.2 System Features

The major features for our system is that The management of Server Inventory should be made easy and hassle free. The new components should be added easily and then the issuance of those components by the users of the system should be as fast as possible depending upon the availability.

4.2.1 System Feature

1. User Authentication and Authorization.
2. Easy to manage inventory.
3. Easy for admin to manage users.

Description and Priority

1. User Authentication and Authorization: The Django authentication system handles both authentication and authorization. Briefly, authentication verifies a user is who they claim to be, and authorization determines what an authenticated user is allowed to do. Here the term authentication is used to refer to both tasks.
   The auth system consists of:
   - Users

- Permissions: Binary (yes/no) flags designating whether a user may perform a certain task.
- Groups: A generic way of applying labels and permissions to more than one user.
- A configurable password hashing system
- Forms and view tools for logging in users, or restricting content
- A pluggable backend system

The authentication system in Django aims to be very generic and doesn't provide some features commonly found in web authentication systems. Solutions for some of these common problems have been implemented in third-party packages:

- Password strength checking
- Throttling of login attempts
- Authentication against third-parties (OAuth, for example)
- Object-level permissions

2. Easy to manage inventory:
   This feature will help the Administrator to manage inventory by giving early warning to the Administrator regarding low inventory availability.
3. Easy for admin to manage users.
   The Administrator can be easily able to add new users and give them permission to use the system and can see the requests and can delete the users as and when necessary.

Stimulus/Response Sequences

For Issuing a new component:

1. Administrator will add components to the inventory.
2. Requester will add a new issue request for a component which is required and which is available in the inventory(assumed).
3. Administrator will review the issue request and check for availability in Inventory.

4. Requester will check the status of his request.
5. Administrator will issue the component to the requester.

For maintenance of faulty component:

1. Requester will make sure that the component is faulty.
2. Requester will raise a maintenance request for the faulty component.
3. Administrator will review the maintenance request.
4. Administrator will check for the available maintenance engineer and then the administrator will assign the maintenance job to that engineer.
5. The maintenance engineer will do the maintenance job on the faulty component and then the engineer will update on the system regarding job completion.

Functional Requirements

1. The Administrator will add users into the system.
2. The Administrator will add components to the inventory.
3. The Users will Log into the system.
4. The Users will raise a request(either issue or maintenance request).
5. The Administrator will review the request before approving it.
6. The Administrator will approve the request after reviewing it.
7. The Administrator will generate reports based upon the issuance of components and the maintenance records.

4.3 External Interface Requirements

4.3.1 User Interfaces

1. Users will be able to log in to the system.
2. Users can raise a Issue request for a specific component.
3. Users can raise a maintenance request for a faulty component.
4. Users can view the status of their requests based upon the request category.

4.3.2 Hardware Interfaces

The physical characteristics required for this system to run smoothly are mentioned below:

- Minimum 1GHz Dual Core Processor (32 bits).
- Minimum 1GB RAM.
- Minimum 32 GB Hard Disk.
- Ethernet Connection (LAN) or a Wireless adapter (Wi-Fi).

4.3.3 Software Interfaces

The minimum supported versions required for our system to run smoothly are mentioned below:

- Any browser with HTML5 support (eg: Google Chrome, Mozilla Firefox, Safari, etc.)
- Windows, MacOS or Linux Operating Systems.
- Any code editor(VS code, Sublime-Text, etc.)

4.3.4 Communications Interfaces

- The Web App will get support on all types of browser.
- The Web App has built-in protection against most types of CSRF(Cross-Site Request Forgery) attacks.
- The Interface between the Database and the Web App will be done by using HTTP Protocol.

4.4 Nonfunctional Requirements

4.4.1 Performance Requirements

- To make sure that our system performs up to the mark, we must ensure that system, when deployed, gets proper internet connection all the time to make sure that the web app runs efficiently.
- We need to make sure that when a new inventory is added into the system, the entry made is correct and precise to avoid any constraints.

4.4.2 Safety Requirements

- In case the server is down, we need to have another backup server available to make sure that the System is available for use to all the users.
- We need to save copies of the database, both locally and on cloud to make sure that if the servers are down, we can still have a count of the available inventory and other data.

4.4.3 Security Requirements

- If the Web App is hosted on a local server, we need to ensure that only the authorised users have access to the server room.
- The software used in the system must be up to date to ensure we have the latest security available for our system.

Chapter 5

System Design

5.1 System Requirements Definition

Our system is a Web Application on Inventory Management System for Server Center, The system's function is to provide easy access to the users to login and request for a new component and request for a maintenance check. There are two main modules in our system, First is the issuance of new component which has been requested by the user, The second module is the maintenance module which is the maintenance of the faulty component which has been previously issued by the Administrator. The Administrator will add new components into the inventory and can be able to add users, delete users and can modify access types to the users. Also the users can be able to view the requests in their portal for the status of their requests.

5.1.1 Functional requirements

- The Administrator should add components into the inventory of the system.
- The Administrator should add users to the system.
- The Users should Login to the system.
- The Users should raise a request(issue request or maintenance request) from the system.
- The Users should be able to see their requests in their portal once they are logged in.
- The Administrator should be able to generate reports based on the issuance of components and on the number of maintenance job completed by the maintenance engineer.

Use-case Diagram

The figure below illustrates the Use Case diagram of the system. In general, A Use Case diagram in its simplest, is a representation of a user's interaction with the system that shows the relationship between the users and the different use cases in which the user is involved. In the figure, there are three users which will interact with the system. The Users are the Requester, Maintenance Engineer and the Administrator.

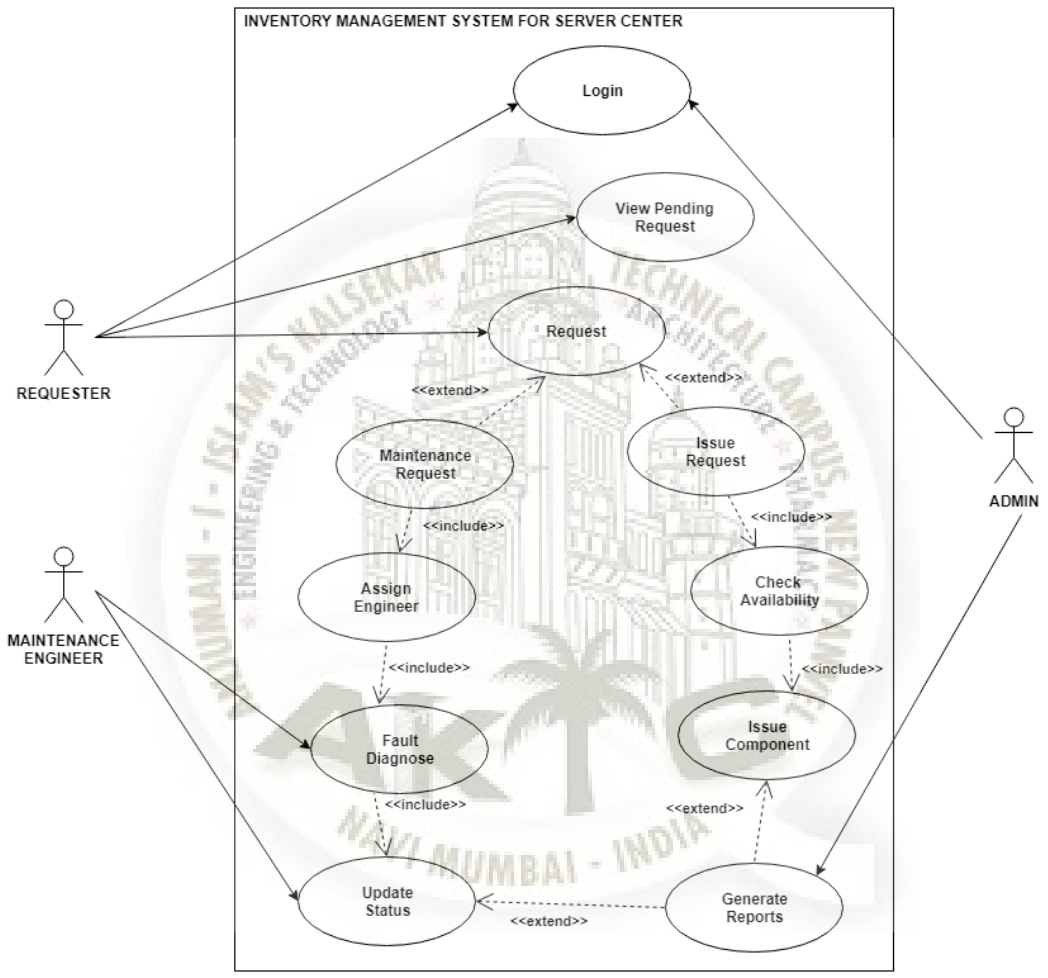


Figure 5.1: Use-Case Diagram

Data-flow Diagram

The figures below illustrates the Data-Flow Diagram of the system. A Data-Flow Diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. We have modelled the data flow of our project, In level 0, we have modelled the data flow between the users of the system and the system itself.

The second diagram is the level 1 Data Flow Diagram of the system in which the modules are briefly explained with their functionalities. These functionalities includes user authentication, requests, checking availability and their respective tables.

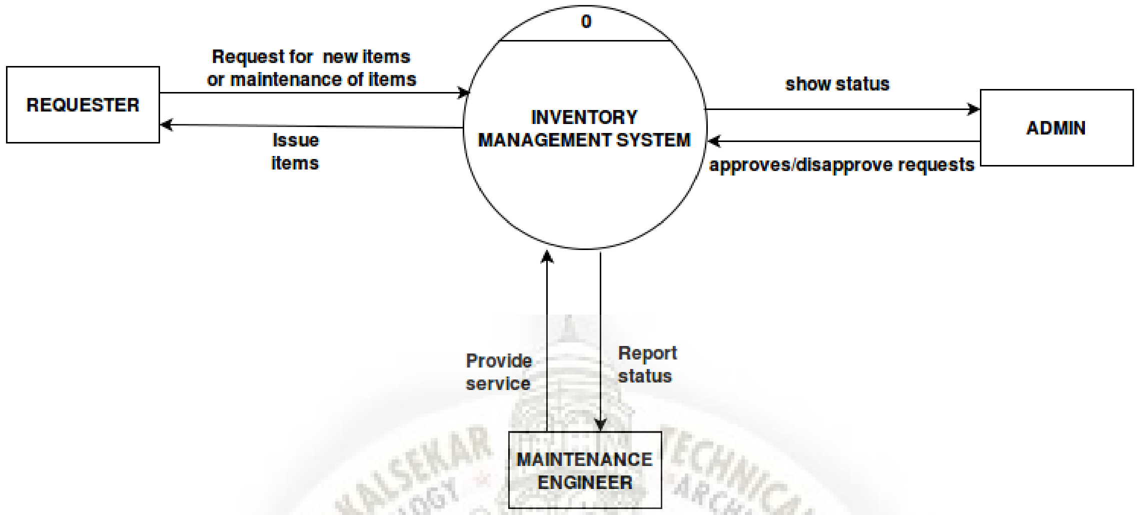

0
INVENTORY MANAGEMENT SYSTEM
REQUESTER
Request for new items or maintenance of items
Issue items
show status
approves/disapprove requests
ADMIN
Provide service
Report status
MAINTENANCE ENGINEER

Figure 5.2: Data Flow Diagram- Level 0

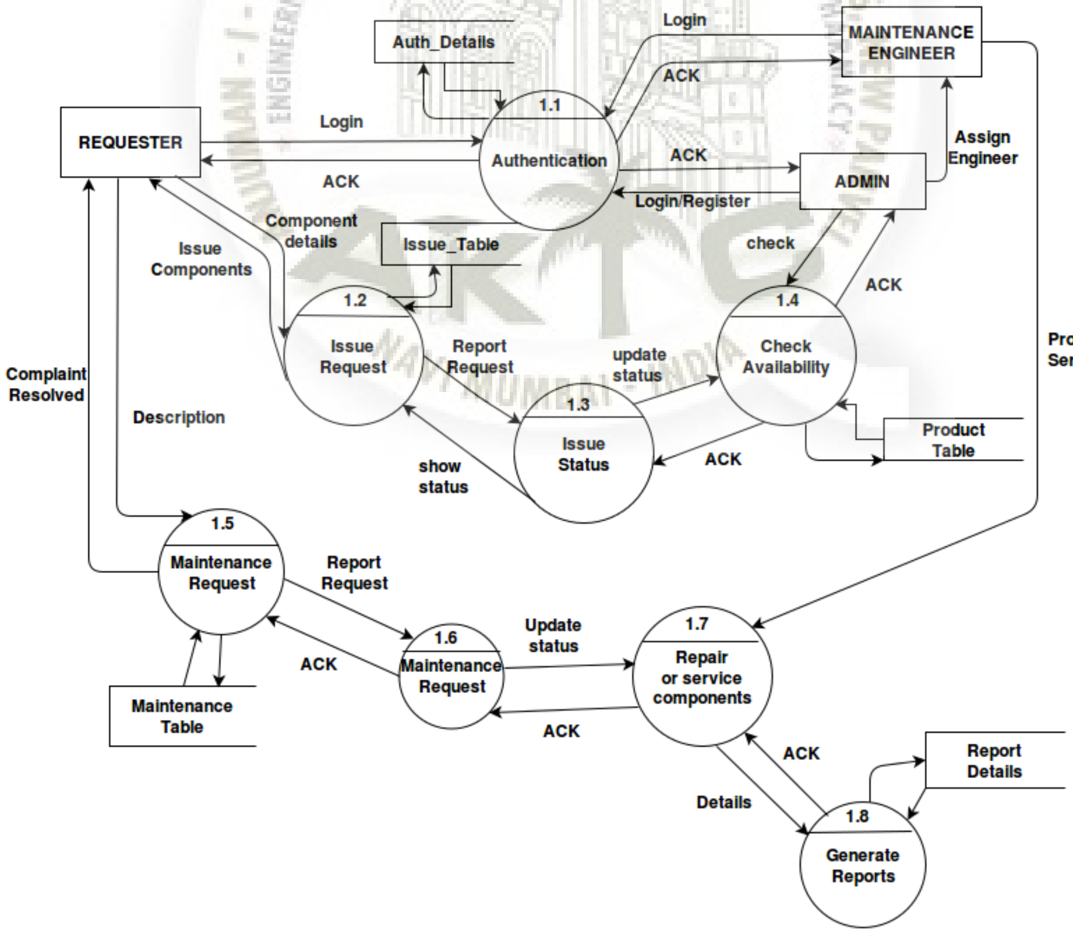


Figure 5.3: Data Flow Diagram- Level 1

5.1.2 System requirements (non-functional requirements)

Performance Requirements:

The system should have high performance and low failure rates. The hardware and software should be able to transmit and receive data from the servers. The requests mainly depends upon the availability of the components in the inventory. For requesting a specific component, User must be authenticated first and then authorized to request for that particular equipment.

5.1.3 Safety Requirements

If there is extensive damage to a wide portion of the database due to catastrophic failure such as a disk crash the recovery method restore a past copy of database that was backed up. To secure the system from unauthorized users we are using username, which requires to log in into the system which will provide a better safety feature. The system must never disclose the inventory information to the users that are not authorized. In case of database failure there must be backup of data or copy of database.

5.1.4 Security Requirements

The major security requirement for the system must be the safeguarding of the user data from any kind of exploit. Before any user want to access the system, they are required to input username and password. Each password must be between 8-12 characters.

Database Schema/ E-R Diagram

Figure below illustrates the ER diagram of our system. The ER or (Entity Relational Model) is a high-level conceptual data model diagram. Entity-Relation model is based on the notion of real-world entities and the relationship between them. ER modeling helps us to analyze data requirements systematically to produce a well-designed database. So, it is considered a best practice to complete ER modeling before implementing our database.

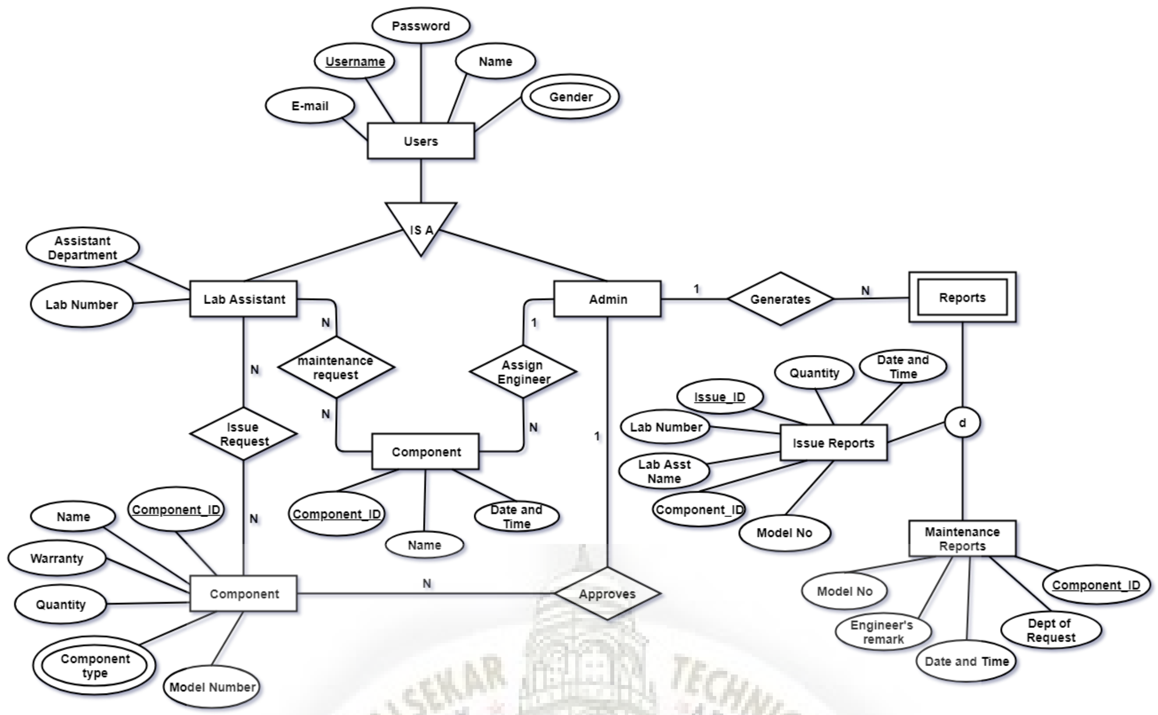


Figure 5.4: Entity-Relationship Diagram

5.2 System Architecture Design

Figure below is the System Architecture of the project. A system architecture or systems architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system.

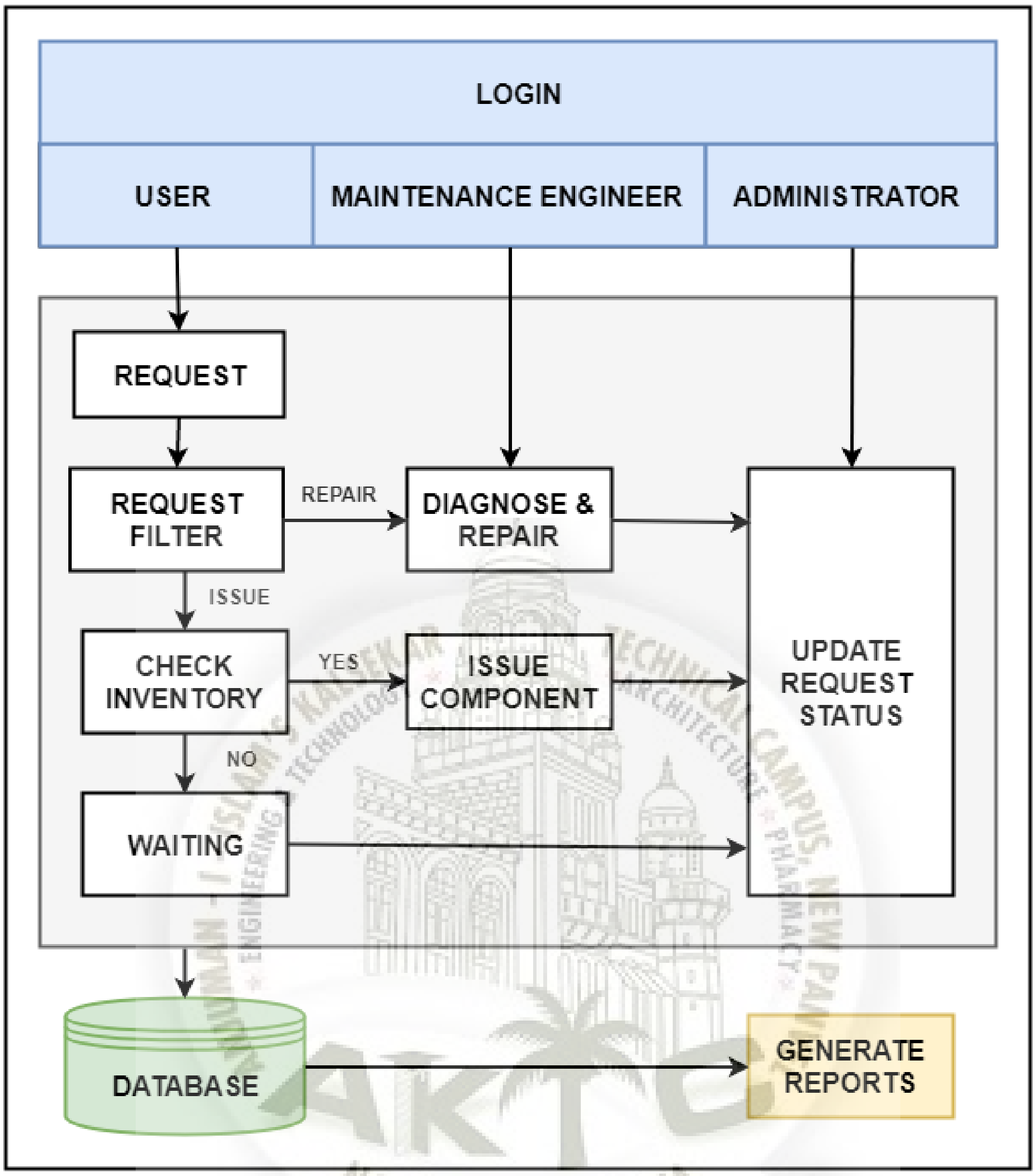


Figure 5.5: System Architecture

5.3 Sub-system Development

A modular description provides detailed information about the module and its supported components which is accessible in different manner. Following are the modules of our system:

5.3.1 Issue Component Module

Once the user is logged in, The user will raise a issue request by entering all the requirements in the system and will submit the details. The Administrator will verify the request, once verified, the Administrator will check the availability in the inventory. If the component is available then that component will be issued to that user and the database will get updated. Here, once the user has successfully raised a new request, they can be able to view their pending requests in their portal.

Module 1 Flow Diagram or Modular Diagram

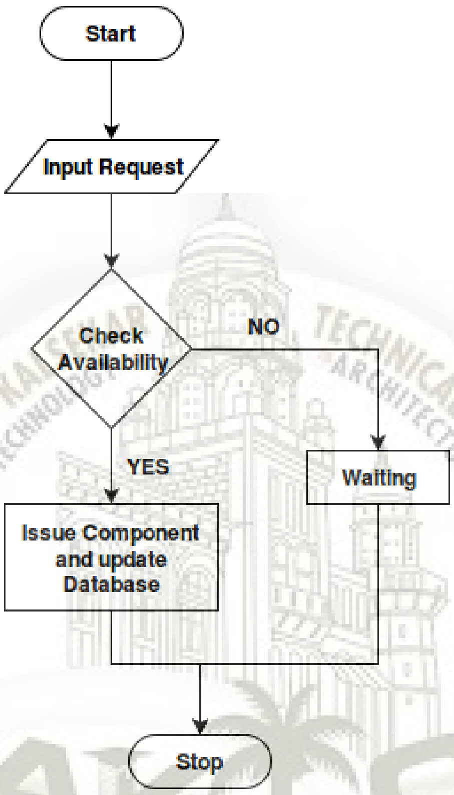


Figure 5.6: Modular Diagram for Issuing a Component

5.3.2 Component Maintenance Module

Once the user is logged in, The user will raise a maintenance request by entering all the requirements in the system and will submit the details. The Administrator will verify the request, once verified, the Administrator will assign a maintenance engineer for that job. The maintenance engineer will carry out the diagnosis of the component and once the job is completed then he will update the system for job completion. Here, once the user has successfully raised a new request, they can be able to view their pending requests in their portal.

Module 2 Flow Diagram or Modular Diagram

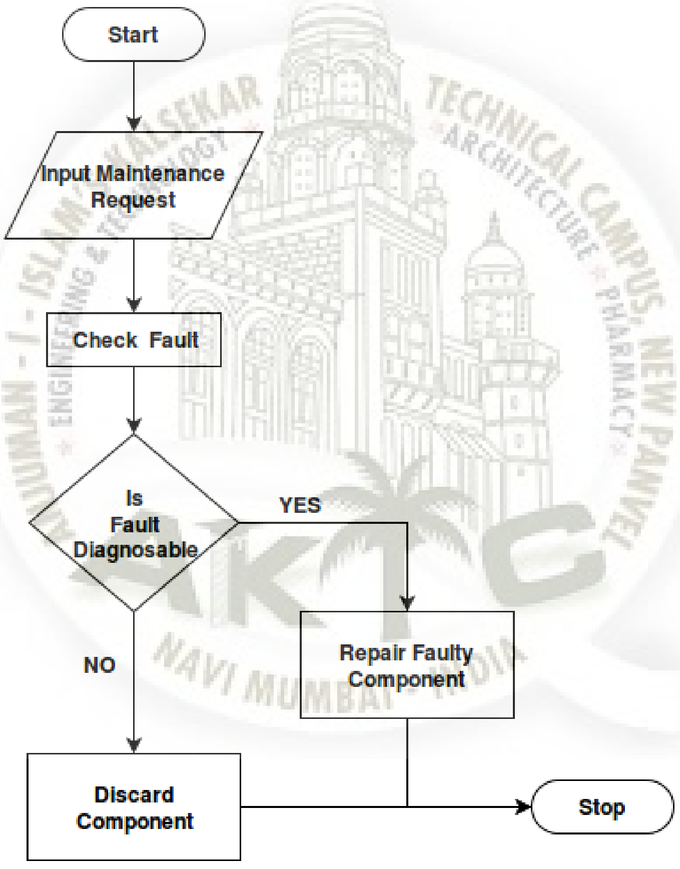


Figure 5.7: Modular Diagram for Maintenance of Component

5.4 Systems Integration

5.4.1 Class Diagram

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the structure of the application, and for detailed modeling translating the models into programming code. Administrator class contains attributes such as username, password, name, email and function such as add component,add user,assign component,reject issue request.component class contains attributes such as component type,component name,component size,serial number.Requester class has attributes username,password,name,email and function request issue item, maintenance request.component maintenance class contains attributes such as lab number,component type,fault description.Maintenance engineer class has function fault diagnosis, view maintenance request,update system.

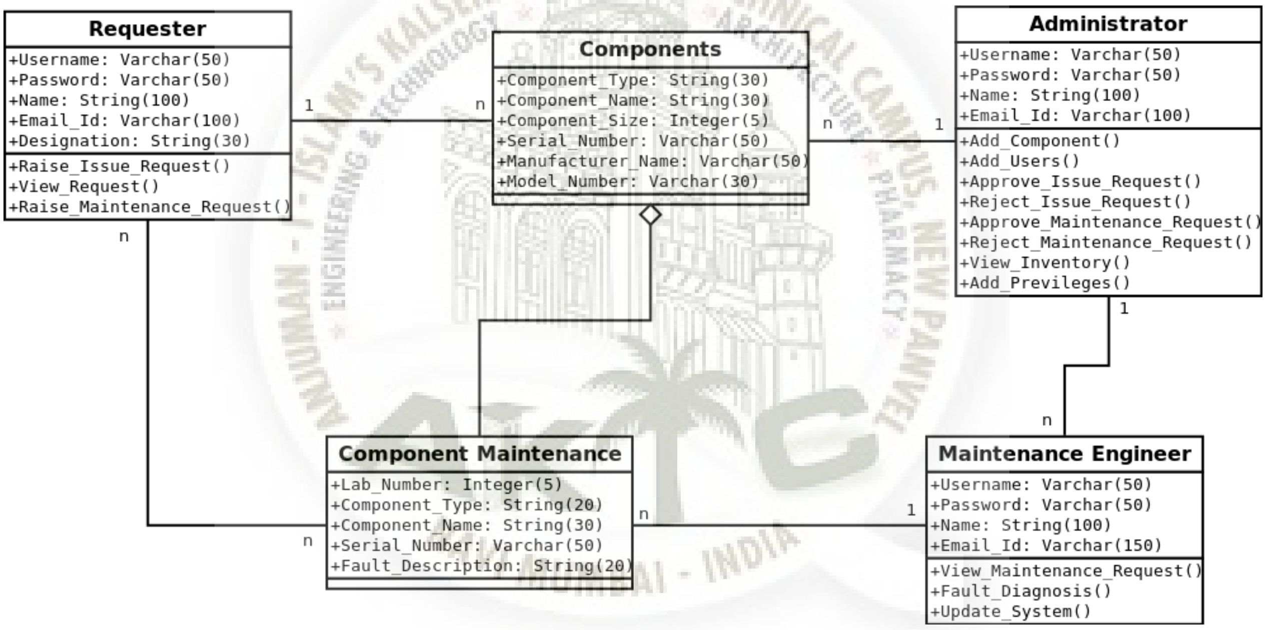


Figure 5.8: Class Diagram

5.4.2 Component Diagram

Component diagrams are different in terms of nature and behavior. Component diagrams are used to model the physical aspects of a system. Now the question is, what are these physical aspects? Physical aspects are the elements such as executables, libraries, files, documents, etc. which reside in a node. Component diagrams are used to visualize interactions between various components of our systems, namely the user, the userpanel, the system itself, inventory and the admin.

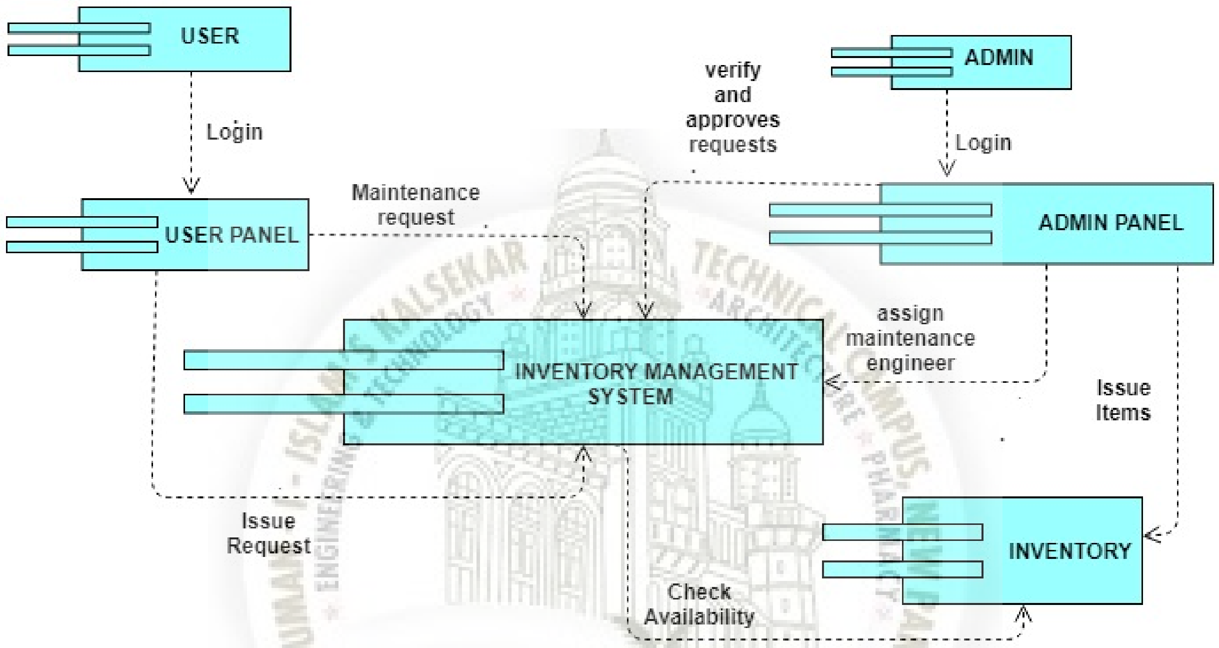


Figure 5.9: Component Diagram

5.4.3 Sequence Diagram

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. User login to the system and request for new component to the system. in this diagram, we have depicted the sequence of actions taken by the users of the system and the system itself to ensure the easy flow of instructions and data through our systems.

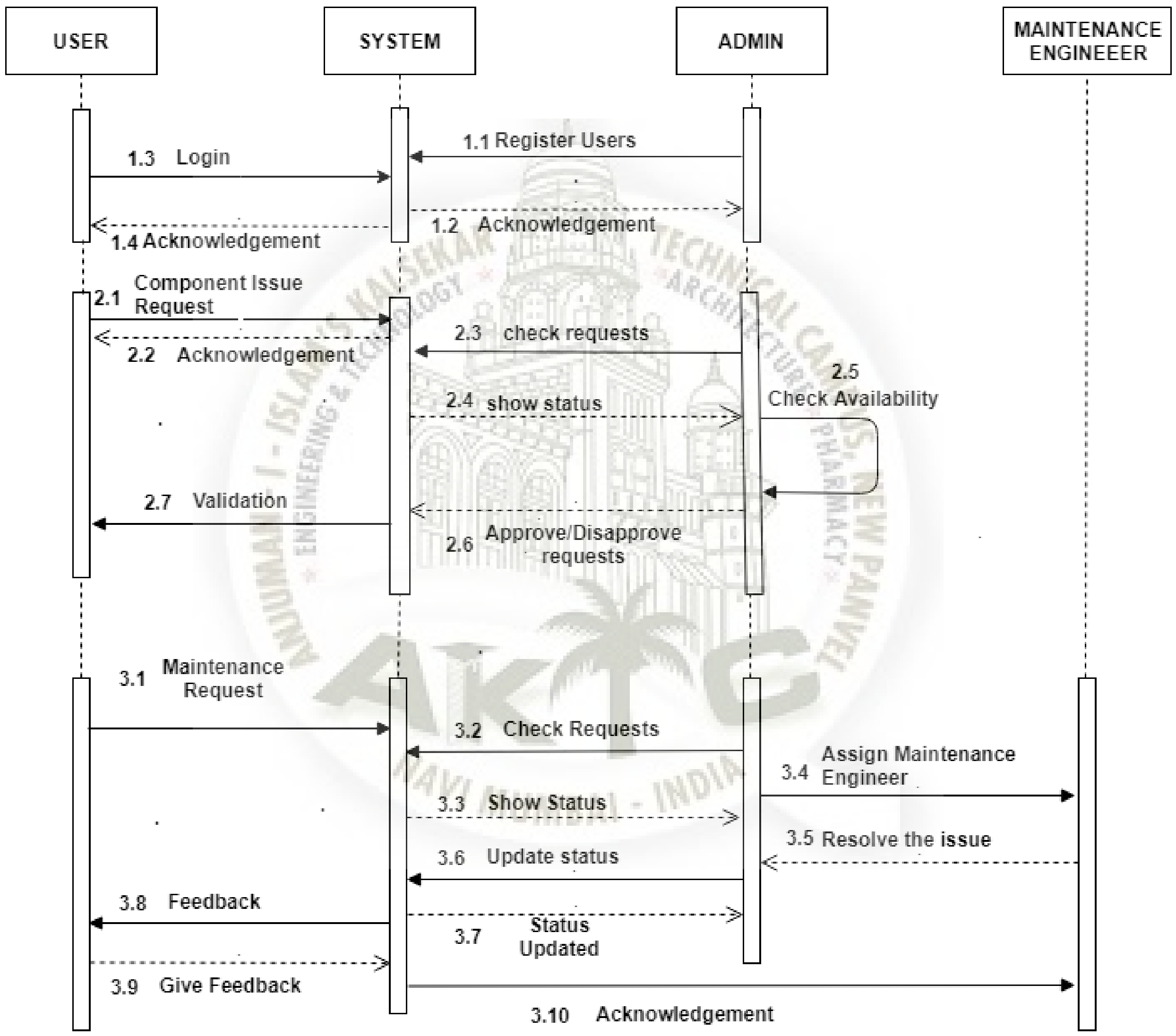


Figure 5.10: Sequence Diagram

5.4.4 Deployment Diagram

A deployment diagram is a UML diagram type that shows the execution architecture of a system, including nodes such as hardware or software execution environments, and the middleware connecting them. The figure below depicts the deployment diagram of our system. The various components depicts the interaction among them, namely the system itself, the requester and engineer component and the administrator component and the interaction between all of them.

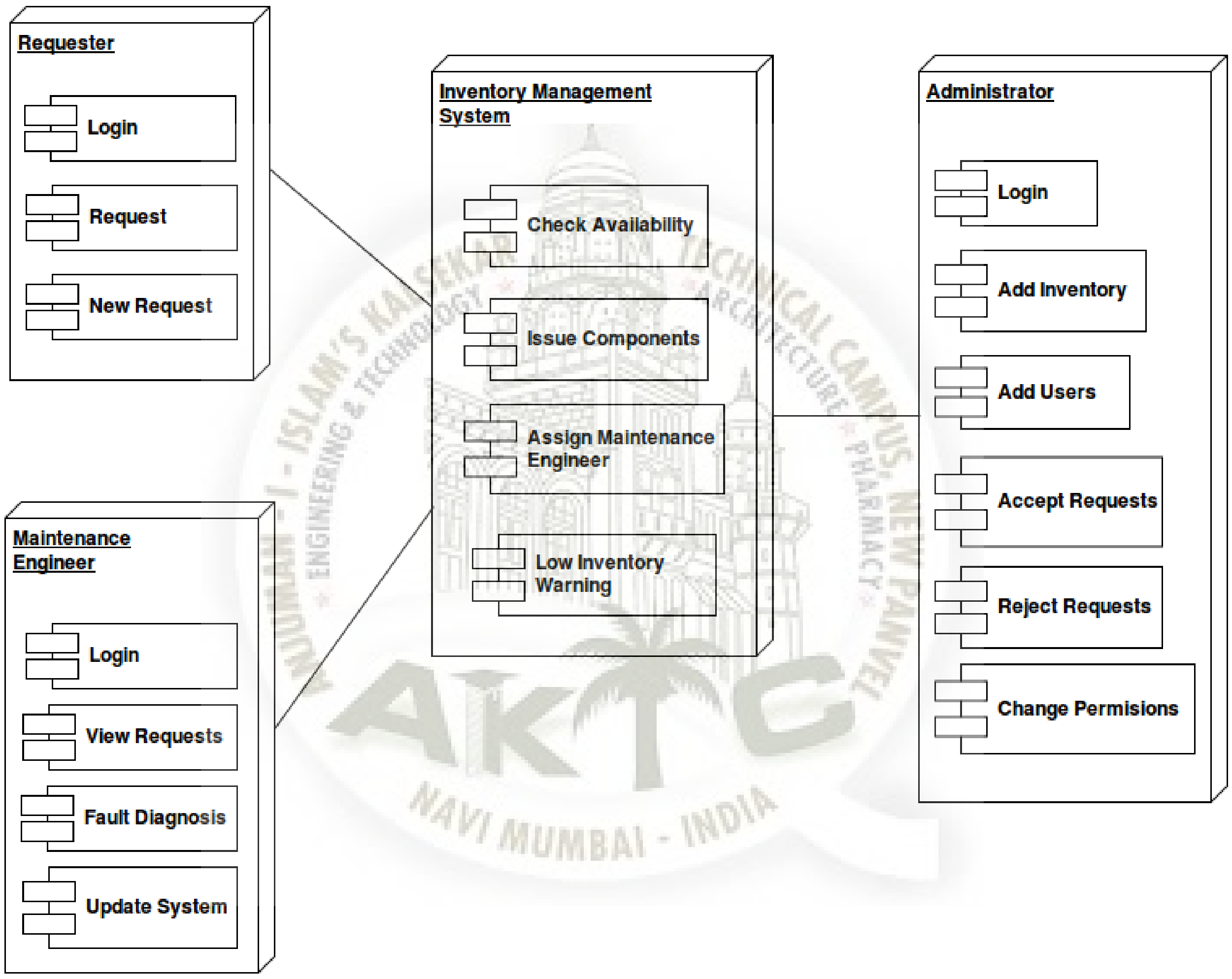


Figure 5.11: Deployment Diagram

Chapter 6

Implementation

6.1 Admin Panel

Administrator will first login to the system

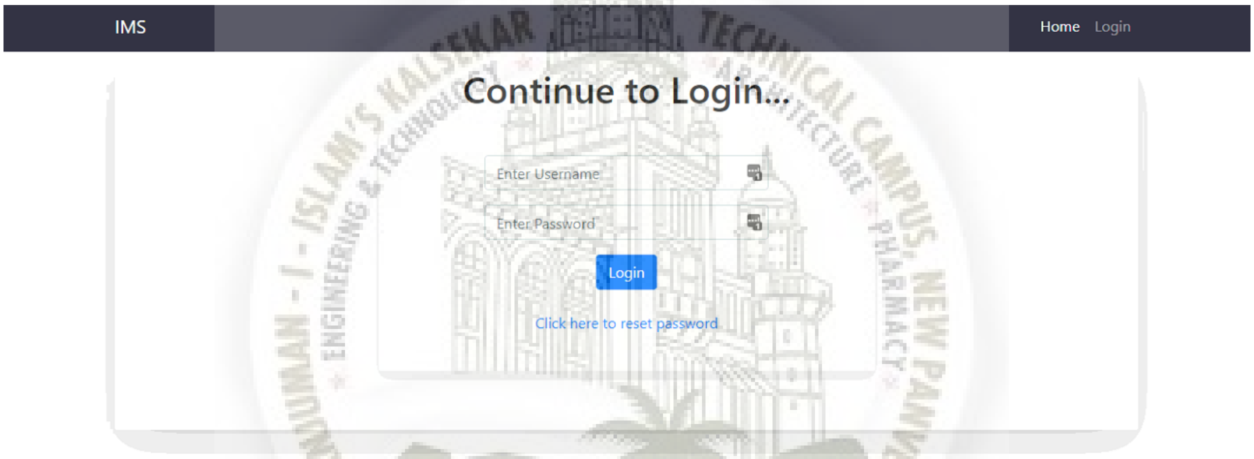


Figure 6.1: Administrator Login Panel

The below code illustrates the admin home page and the functions admin can perform

```
  <!doctype html>
<html lang="en">
    <head>
        <!-- Required meta tags -->
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1,
            shrink-to-fit=no">

        <!-- Bootstrap CSS -->
        <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/
            bootstrap/4.2.1/css/bootstrap.min.css" integrity="sha384-
            GJzZqFGwb1QTTN6wy59ffF1BuGJpLSa9DkKMp0DgiMDm4iYMj70gZWKYbI706tWS"
            crossorigin="anonymous">

        <title>IMS</title>

        <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <div class="container">
            <a class="navbar-brand" href="{% url 'accounts:home'%}">IMS</a>
            <button class="navbar-toggler" type="button" data-toggle="
                collapse" data-target="#navbarSupportedContent" aria-
                controls="navbarSupportedContent" aria-expanded="false" aria
                -label="Toggle navigation">
            <span class="navbar-toggler-icon"></span></button>

            <div class="collapse navbar-collapse" id="navbarSupportedContent
                ">
                <ul class="navbar-nav ml-auto">
                    <li class="nav-item active">
                        <a class="nav-link" href="{% url 'accounts:home' %}"
                            >Home <span class="sr-only">(current)</span></a>
                    </li>
                    <li class="nav-item active">
                        <a class="nav-link" href="{% url 'accounts:
                            admins_home' %}">Admins Home <span class="sr-
                            only">(current)</span></a>
                    </li>

                    {% if user.is_authenticated %}

                    <li class="nav-item dropdown">
                        <a class="nav-link dropdown-toggle" href="#" id="
                            navbarDropdown" role="button" data-toggle="
                            dropdown" aria-haspopup="true" aria-expanded="
                            false">
                        Services</a>
                        <div class="dropdown-menu" aria-labelledby="
                            navbarDropdown">
                        <a class="dropdown-item" href="{% url 'accounts:
                            admins_status' %}">Product Table</a>
                        <div class="dropdown-divider"></div>
                        <a class="dropdown-item" href="{% url 'accounts:
                            admins_issue_request_status' %}">Issue Request
                            Status</a>
                        <a class="dropdown-item" href="{% url 'accounts:
                            admins_maintenance_status' %}">Maintenance
                            Status</a>

                        </div>
                    </li>
                    <li class="nav-item dropdown">
                        <a class="nav-link dropdown-toggle" href="#" id="
                            navbarDropdown" role="button" data-toggle="
                            dropdown" aria-haspopup="true" aria-expanded="
                            false">
                        Reports</a>
                        <div class="dropdown-menu" aria-labelledby="
                            navbarDropdown">
                        <a class="dropdown-item" href="{% url 'accounts:
                            export_product_table' %}">Products Table</a>
                        <a class="dropdown-item" href="{% url 'accounts:
                            export_req_issue_item' %}">Request Issue item</a
                            >
                        <a class="dropdown-item" href="{% url 'accounts:
                            export_res_issue_item' %}">Response Issue item</
                            a>
                        <a class="dropdown-item" href="{% url 'accounts:
                            export_req_maintenance' %}">Request Maintenance
```

```
                                </a>
                            </div>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" href="{% url 'accounts:edit' %}"
                                >Edit Profile</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" href="{% url 'accounts:logout'
                                %}">Logout</a>
                        </li>

                        {% else %}

                        <li class="nav-item">
                            <a class="nav-link" href="{% url 'accounts:login' %}
                                ">Login</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" href="{% url 'accounts:
                                admins_login' %}">Admin</a>
                        </li>

                        {% endif %}

                    </ul>
                </div>
            </div>
        </nav>
    </head>

    <body>
        <div class="container">
            {% if messages %}
                {% for message in messages %}
                    <div class="alert alert-warning" role="alert">
                        <button class="close" data-dismissable="alert"><small><
                            sup>X</sup></small></button>
                        {{ message }}
                    </div>
                {% endfor %}
            {% endif %}

            {% block content %}
            {% endblock %}
        </div>

        <!-- Optional JavaScript -->
        <!-- jQuery first, then Popper.js, then Bootstrap JS -->
        <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity
            ="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8
            abtTE1Pi6jizo" crossorigin="anonymous"></script>
        <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.6/umd
            /popper.min.js" integrity="sha384-wHAiFfRlMFy6i5SRaxvfOCifBUQy1xHdJ/
            yoi7FRNXMRBu5WHdZYu1hA6ZOblgut" crossorigin="anonymous"></script>
        <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/js/
            bootstrap.min.js" integrity="sha384-B0UglyR+
            jN6CkvvICOB2joaf5I4l3gm9GU6Hc1og6Ls7i6U/mkkaduKaBhlAXv9k"
            crossorigin="anonymous"></script>
    </body>
</html>
```

The below code illustrates the admin login procedure

```
{% extends "accounts/admins_base.html" %}

{% block content %}

<div class="container shadow-lg p-3 mb-5 bg-white rounded">
    <h1 class="text-center">Continue to Login...</h1>
    <div class="col-md-6 offset-md-3 shadow p-3 mb-5 bg-white rounded">
        <br>
        <div class="col-md-8 offset-md-2">
            <form method="POST">
            {% csrf_token %}
                <div class="form-group">
                    <input type="text" class="form-control" placeholder="Enter
                       Username" name="username">
                </div>
                <div class="form-group">
                    <input type="password" class="form-control" placeholder="
                       Enter Password" name="password">
                </div>
                <div class="text-center">
                    <button type="submit" class="btn btn-primary center-block">
                       Login</button>
                </div>
            </form>

            <br>
            <div class="text-center">
            <a href="#">Click here to reset password</a>
            </div>
            <br>
        </div>
    </div>
</div>

{% endblock content %}
```

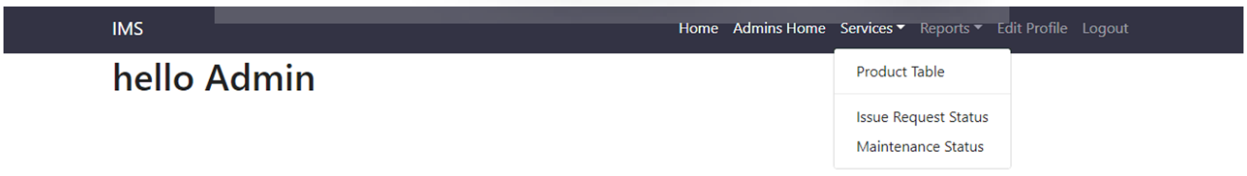
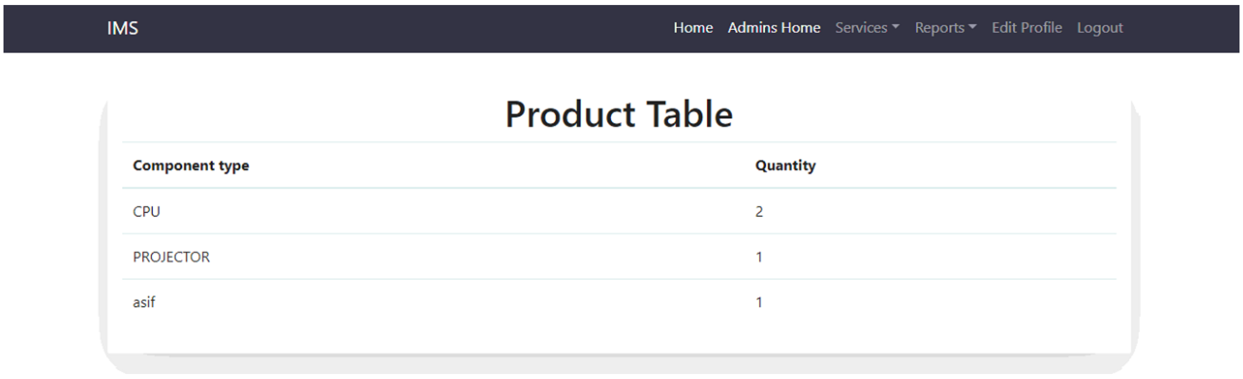


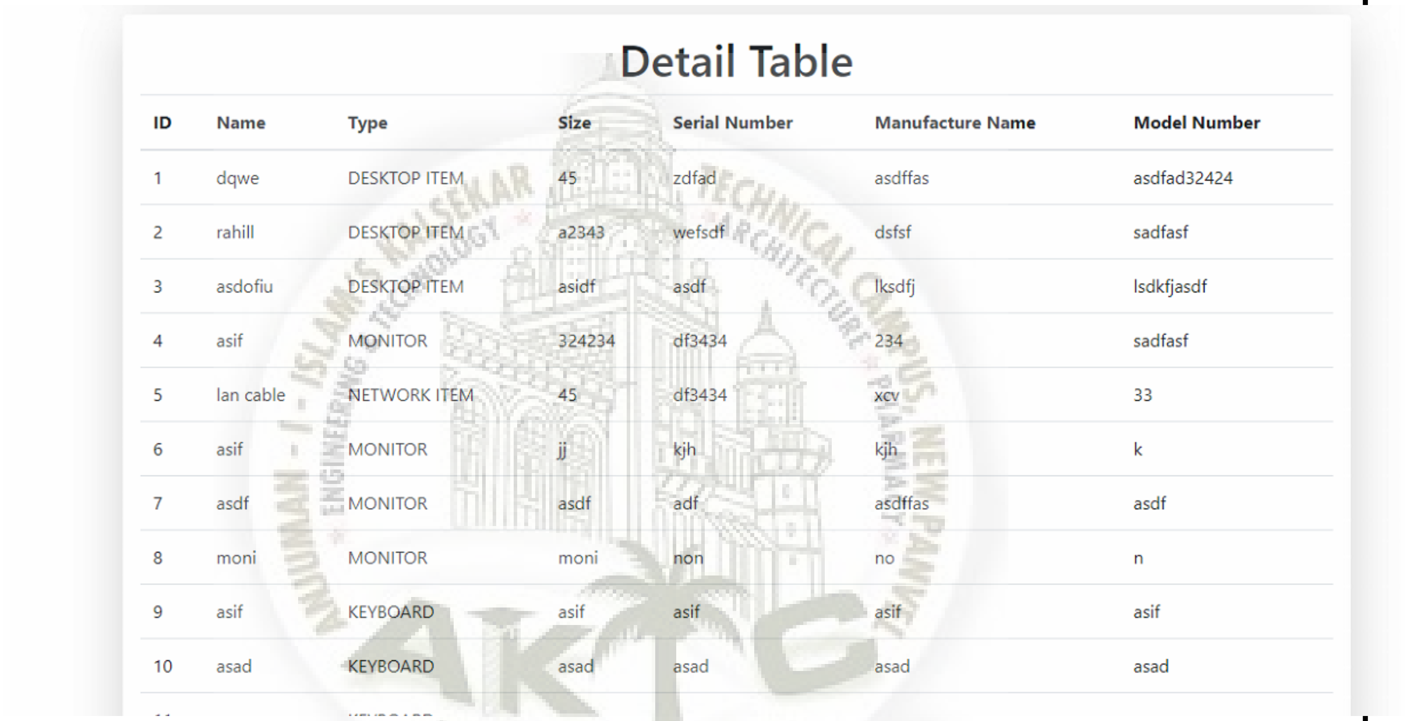
Figure 6.2: Administrator Services Panel



Product Table

Component type	Quantity
CPU	2
PROJECTOR	1
asif	1

Figure 6.3: Administrator Product Table



Detail Table

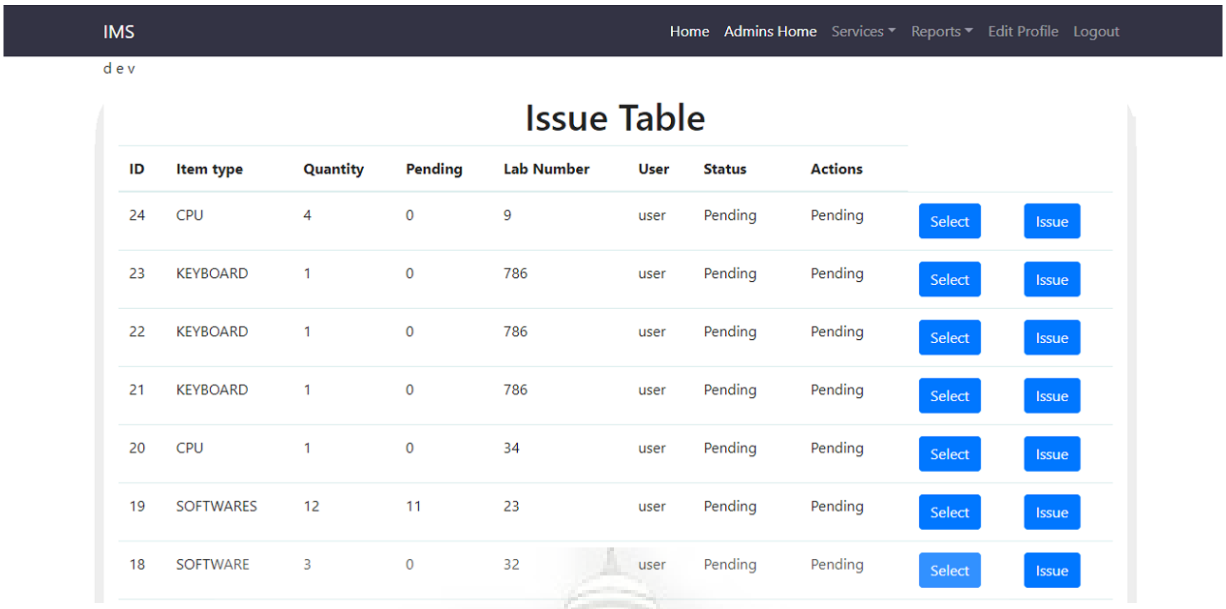
ID	Name	Type	Size	Serial Number	Manufacture Name	Model Number
1	dqwe	DESKTOP ITEM	45	zdfad	asdffas	asdfad32424
2	rahill	DESKTOP ITEM	a2343	wefsdf	dsfsf	sadfasf
3	asdofiu	DESKTOP ITEM	asidf	asdf	lksdfj	lsdkfjasdf
4	asif	MONITOR	324234	df3434	234	sadfasf
5	lan cable	NETWORK ITEM	45	df3434	xcv	33
6	asif	MONITOR	jj	kjh	kjh	k
7	asdf	MONITOR	asdf	adf	asdffas	asdf
8	moni	MONITOR	moni	non	no	n
9	asif	KEYBOARD	asif	asif	asif	asif
10	asad	KEYBOARD	asad	asad	asad	asad

Figure 6.4: Administrator Product Detail Table

The below code illustrates the admin's adding new product in the product table:

```
 {% extends "accounts/admins_base.html" %}

{% block content %}

 <form method="POST" action="{% url 'accounts:admins_add_item' %}">
 {% csrf_token %}
   <div class="form-group">
        <label> <h3>Component Name :</h3></label>
        <input type="text" class="form-control"  name="c_name" required>
    </div>
       <div class="form-group">
            <label >Component Type :</label>
            <select class="form-control"  name = "c_type" required>
                 <option>MONITOR</option>
                 <option>CPU</option>
                 <option>KEYBOARD</option>
                 <option>MOUSE</option>
```

```
                    <option>LAPTOP</option>
                    <option>STORAGE DEVICES</option>
                    <option>SOFTWARE</option>
                    <option>GATEWAY</option>
                    <option>ROUTER</option>
                    <option>SWITCH</option>
                    <option>REPEATER</option>
                    <option>LAN CABLE</option>
                    <option>PRINTER</option>
                    <option>PROJECTOR</option>
                    <option>POWER CABLE</option>
                    <option>PROJECTOR CABLE</option>
                </select>
          </div>        <div class="form-group">
              <label >Component Size :</label>
              <input type="Text" class="form-control" name="c_size" required>
          </div>

          <div class="form-group">
              <label >Component Serial Number :</label>
              <input type="text" class="form-control" name="c_sno" required>
          </div>


          <div class="form-group">
              <label >Model Number :</label>
              <input type="text" class="form-control"  name="c_mno" required>
          </div>

          <div class="form-group">
              <label >Manufacturer Name :</label>
              <input type="text" class="form-control"  name="c_mname" required>
          </div>

          <div>  <button type="submit" class="btn btn-primary center-block">Submit
              </button></div>
          </form>

{% endblock %}
```



Issue Table

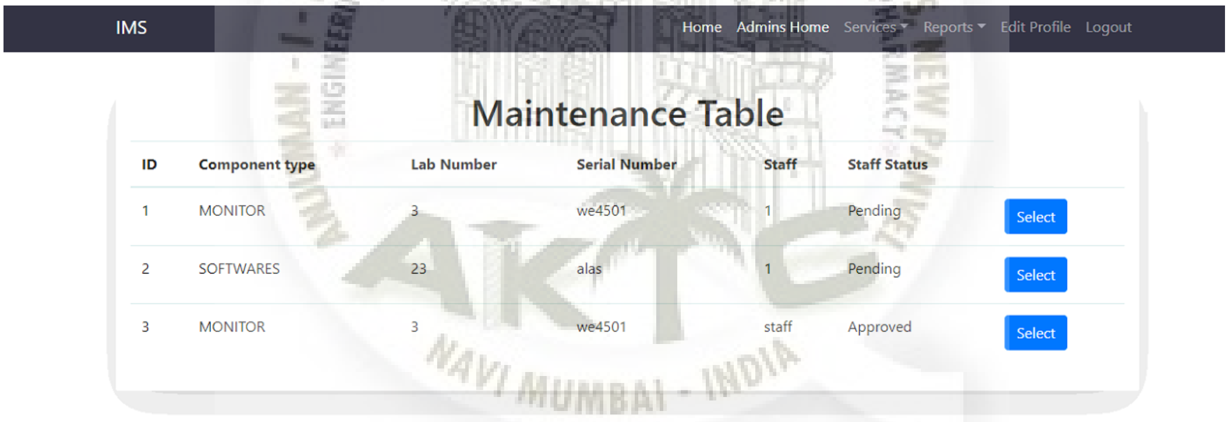
ID	Item type	Quantity	Pending	Lab Number	User	Status	Actions		
24	CPU	4	0	9	user	Pending	Pending	Select	Issue
23	KEYBOARD	1	0	786	user	Pending	Pending	Select	Issue
22	KEYBOARD	1	0	786	user	Pending	Pending	Select	Issue
21	KEYBOARD	1	0	786	user	Pending	Pending	Select	Issue
20	CPU	1	0	34	user	Pending	Pending	Select	Issue
19	SOFTWARES	12	11	23	user	Pending	Pending	Select	Issue
18	SOFTWARE	3	0	32	user	Pending	Pending	Select	Issue

Figure 6.5: Administrator Issue Request Table

The below code illustrates the code for administrator issue request table

```
 {% extends "accounts/admins_base.html" %}

{% block content %}
 {%for ris in rs%}
     {{ris}}
     {%endfor%}
<br>
<div class="container shadow-lg p-3 mb-5 bg-white rounded">
     <h1 class="text-center">Issue Table</h1>
     <table class="table">
          <thead>
               <tr>
                    <th scope="col">ID</th>
                    <th scope="col">Item type</th>
                    <th scope="col">Quantity</th>
                    <th scope="col">Pending</th>

                    <th scope="col">Lab Number</th>
                    <th scope="col">User</th>
                    <th scope="col">Status</th>
                    <th scope="col">Actions</th>
               </tr>
               </thead>
          <tbody>
          {% if es is null %}
               <tr>
                    <td>Null</td>
                    <td>Null</td>
                    <td>Null</td>
                    <td>Null</td>
                    <td>Null</td>
                    <td>Null</td>
                    <td>Null</td>


               </tr>
          {% else %}
```

```
                {% for eis in es %}
                <tr>
                    <td>{{ eis.id }}</td>
                    <td>{{ eis.component_type }}</td>
                    <td>{{ eis.quantity }}</td>
                    <td>{{ eis.pending_item }}</td>
                    <td>{{ eis.lab_number }}</td>
                    <td>{{ eis.user_name }}</td>
                    <td>{{ eis.status }}</td>
                    <td>{{ eis.admin_status }}</td>

                    <!--<td>{{ eis.description }}</td>-->
                    <td><a href="{% url 'accounts:admins_issue_update' eis.id %}"
                        class="btn btn-primary" role="button">Select</a></td>
                    <td><a href="{% url 'accounts:admin_issue_item' eis.id eis.
                        component_type eis.quantity %}" class="btn btn-primary" role
                        ="button">Issue</a></td>
                </tr>
                {% endfor %}
            {% endif %}
            </tbody>
        </table>
</div>

{% endblock %}
```



Maintenance Table

ID	Component type	Lab Number	Serial Number	Staff	Staff Status	
1	MONITOR	3	we4501	1	Pending	Select
2	SOFTWARES	23	alas	1	Pending	Select
3	MONITOR	3	we4501	staff	Approved	Select

Figure 6.6: Administrator Maintenance Request Table

The code below illustrates the admin panel of maintenance of component

```
{% extends "accounts/admins_base.html" %}

{% block content %}

<br>

<div class="container shadow-lg p-3 mb-5 bg-white rounded">
    <h1 class="text-center">Maintenance Table</h1>
    <table class="table">
        <thead>
            <tr>
                <th scope="col">ID</th>
                <th scope="col">Component type</th>
                <th scope="col">Lab Number</th>
                <th scope="col">Serial Number</th>
                <th scope="col">Staff </th>
                <th scope="col">Staff Status </th>
            </tr>
            </thead>
        <tbody>
        {% if es is null %}
            <tr>
                <td>Null</td>
                <td>Null</td>
                <td>Null</td>
                <td>Null</td>
                <td>Null</td>
                <td>Null</td>


            </tr>
        {% else %}
            {% for eis in es %}
            <tr>
                <td>{{ eis.id }}</td>
                <td>{{ eis.component_type }}</td>
                <td>{{ eis.lab_number }}</td>
                <td>{{ eis.component_serial_number }}</td>
                <td>{{ eis.staff }}</td>
                <td>{{ eis.admin_status }}</td>
                {% if eis.admin_status == "Faulty" %}
                    <td><a href="{% url 'accounts:update_product' eis.id %}"
                        class="btn btn-danger" role="button">Replace</a></td>
                {%else%}
                    <td><a href="{% url 'accounts:admins_maintenance_update' eis
                        .id %}" class="btn btn-primary" role="button">Select</a
                        ></td>
                {%endif%}

            </tr>
            {% endfor %}
        {% endif %}
        </tbody>
    </table>
</div>

{% endblock %}
```

The code below illustrates the admin panel of updating the maintenance request of component

```
{% extends "accounts/admins_base.html" %}

{% block content %}
<br>
<div class="container shadow-lg p-3 mb-5 bg-white rounded">
    <h1 class="text-center">Update Issue Request</h1>
    <div class="col-md-6 offset-md-3 shadow p-3 mb-5 bg-white rounded">
    <form method="POST" action="{% url 'accounts:admins_maintenance_edit' %}">
    {% csrf_token %}
    {% if es is not Null %}
    {% for eis in es %}

    <div class="form-group">
        <label> <h3>Request Number :</h3></label>
        <input type="Number" class="form-control" value="{{ eis.id }}" name="
           ids">
     </div>
        <div class="form-group">
            <label >Component type :</label>
            <input type="text" class="form-control" value="{{ eis.component_type
               }}" name="component_type" >
        </div>

        <div class="form-group">
            <label >Serial number :</label>
            <input type="text" class="form-control" value="{{ eis.
                component_serial_number }}" name="component_serial_number" >
        </div>

        <div class="form-group">
            <label >Lab Number :</label>
            <input type="text" class="form-control" value="{{ eis.lab_number}}"
                name="lab_number" >
        </div>
        <div class="form-group">
            <label >User Name :</label>
            <input type="text" class="form-control" value="{{ eis.user_name }}"
                name="user" >
        </div>

        <div class="form-group">
            <label >Status :</label>
            <input type="text" class="form-control" value="{{ eis.status }}"
                name = "status" >
        </div>

        <div class="form-group">
            <label >Staff :</label>
                <select class="form-control" name="staff">
                {%for sil in sl%}
                    <option>{{sil}}</option>
                {%endfor%}

                </select>
        </div>
        <div> <button type="submit" class="btn btn-primary center-block">Submit
            </button></div>
    {% endfor %}
    {% else %}<h1>Empty DataSet</h1>
    {% endif %}
    </form>
    </div>
</div>
{% endblock %}
```

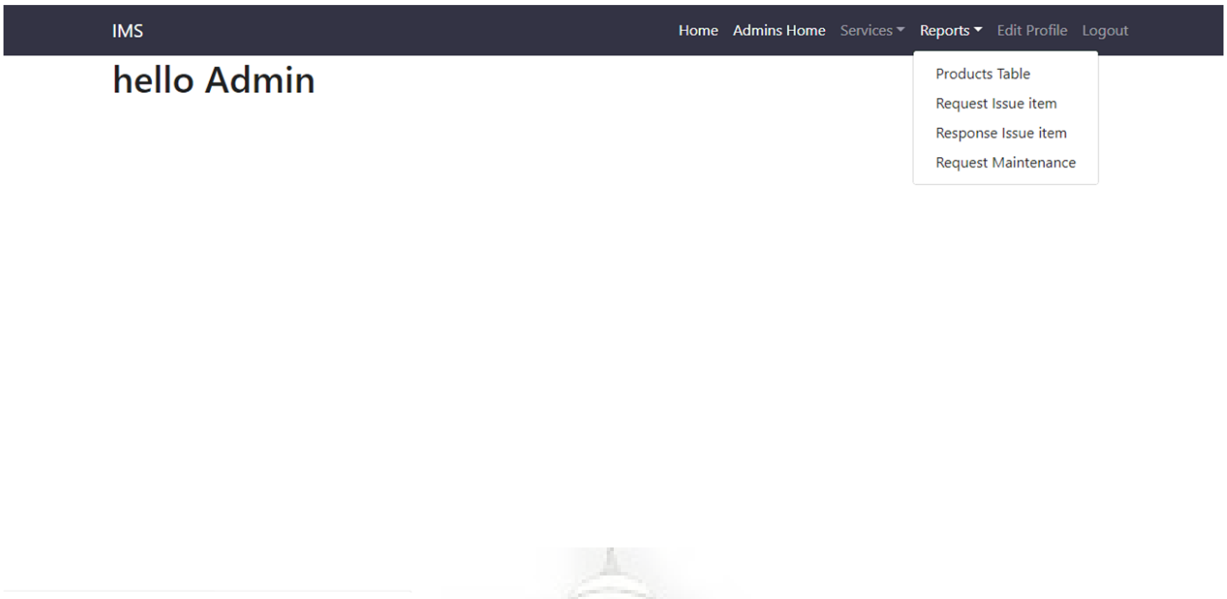


Figure 6.7: Administrator Reports

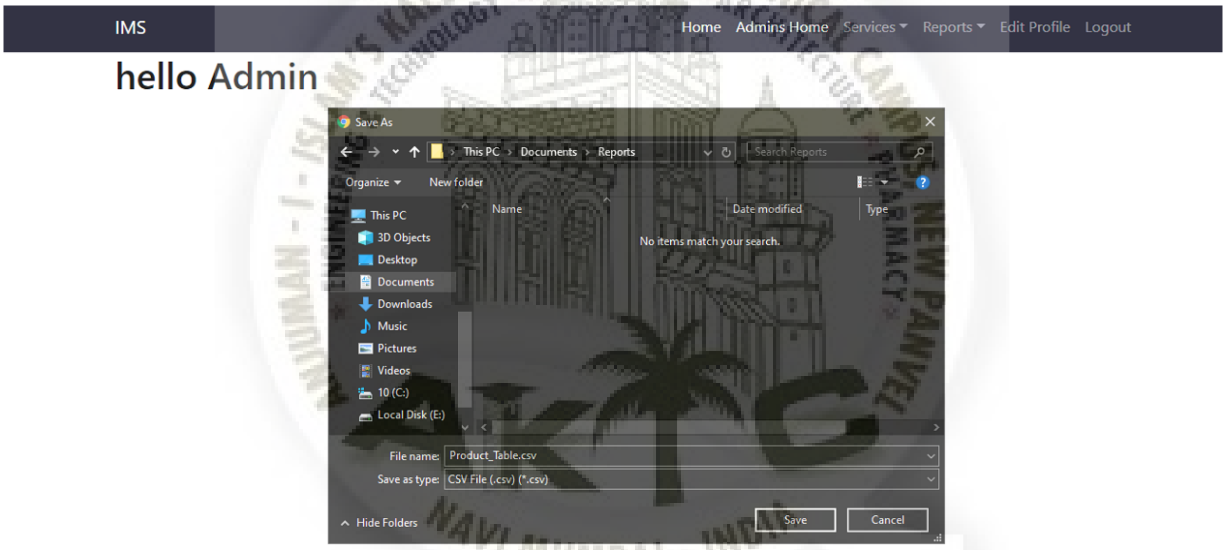
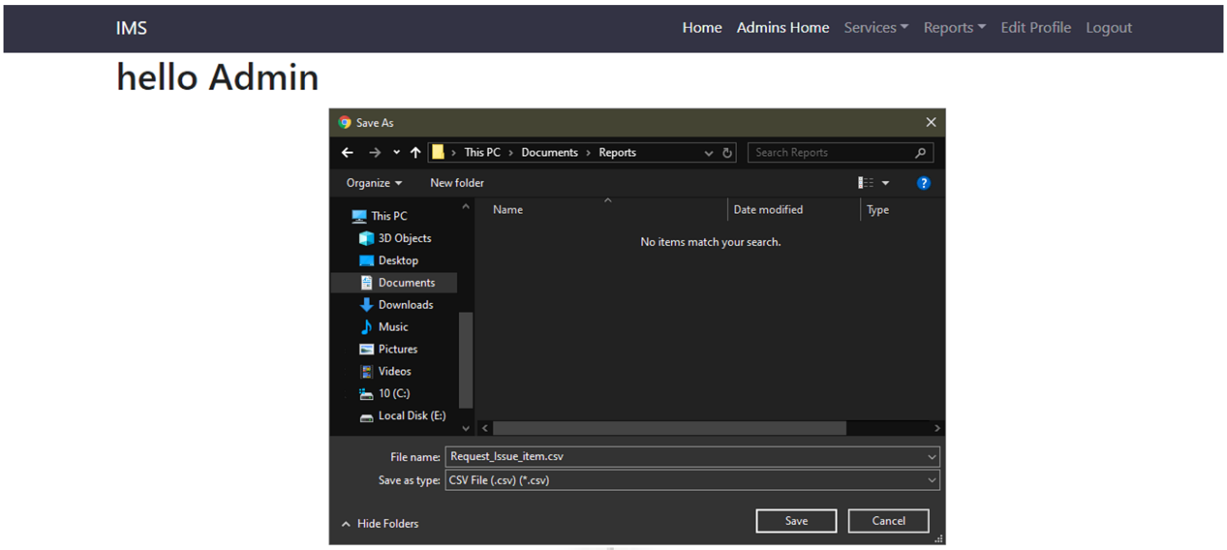


Figure 6.8: Product Table Reports

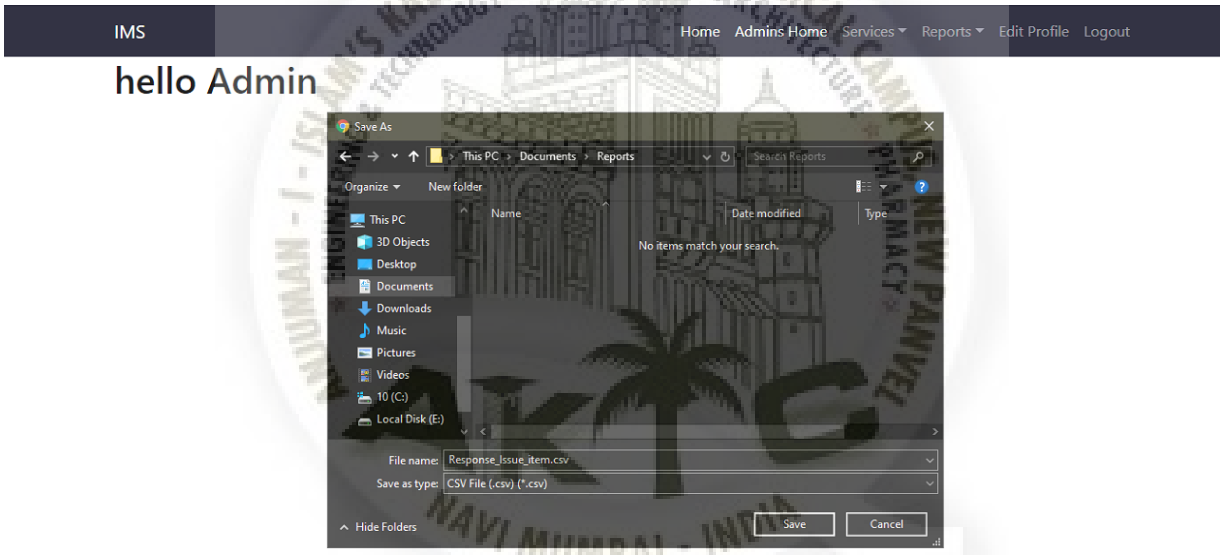


No items match your search.

File name: Request_Issue_item.csv

Save as type: CSV File (.csv) (*.csv)

Figure 6.9: Issue Requests Reports

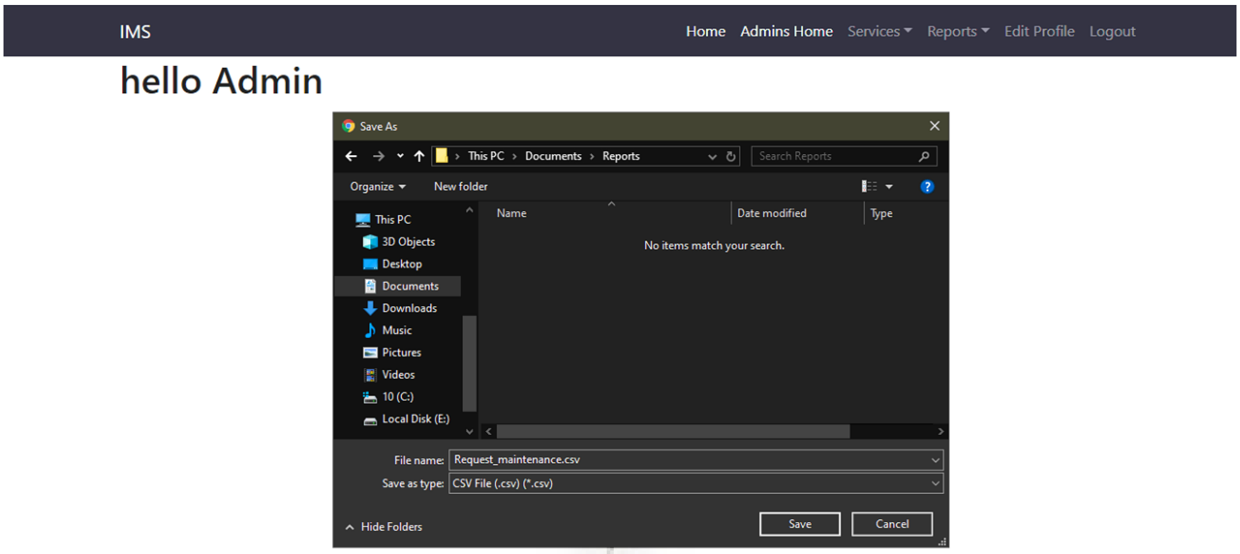


No items match your search.

File name: Response_Issue_item.csv

Save as type: CSV File (.csv) (*.csv)

Figure 6.10: Response of Issue Request Reports



No items match your search.

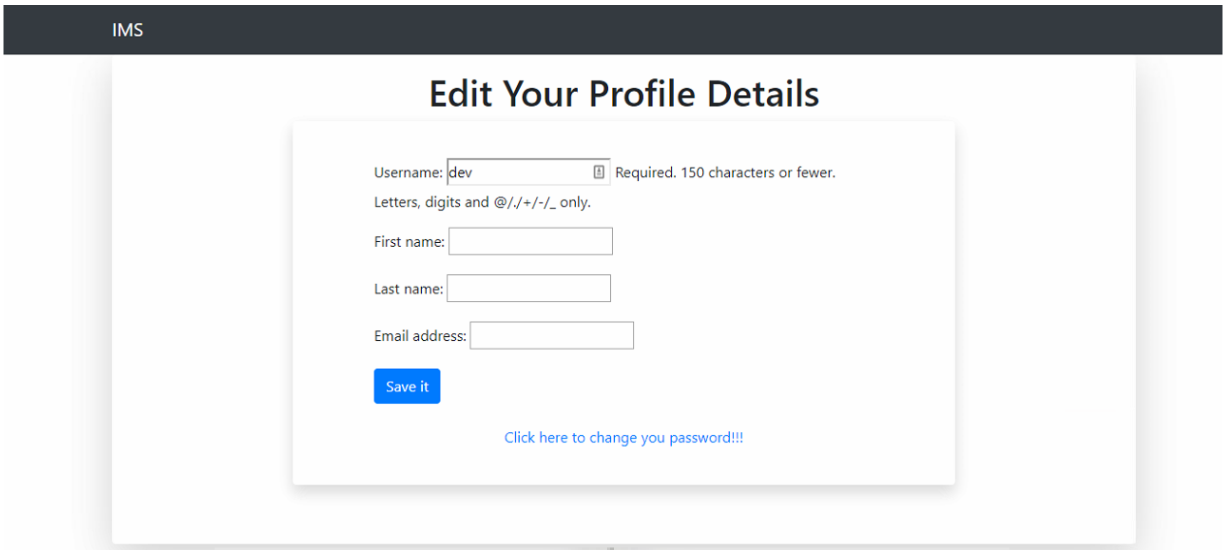
File name: Request_maintenance.csv

Save as type: CSV File (.csv) (*.csv)

Figure 6.11: Maintenance Requests Reports

The functions used to generate reports, the codes are illustrated below

```
def export_product_table(request):
    product_table_Resource = Product_table_Resource()
    dataset = product_table_Resource.export()
    response = HttpResponse(dataset.csv, content_type='text/csv')
    response['Content-Disposition'] = 'attachment; filename="Product_Table.csv"'
    return response

def export_req_issue_item(request):
    req_issue_item_Resource = Req_issue_item_Resource()
    dataset = req_issue_item_Resource.export()
    response = HttpResponse(dataset.csv, content_type='text/csv')
    response['Content-Disposition'] = 'attachment; filename="Request_Issue_item.
        csv"'
    return response

def export_res_issue_item(request):
    res_issue_item_Resource = Res_issue_item_Resource()
    dataset = res_issue_item_Resource.export()
    response = HttpResponse(dataset.csv, content_type='text/csv')
    response['Content-Disposition'] = 'attachment; filename="Response_Issue_item
        .csv"'
    return response

def export_req_maintenance(request):
    req_maintenance_Resource = Req_maintenance_Resource()
    dataset = req_maintenance_Resource.export()
    response = HttpResponse(dataset.csv, content_type='text/csv')
    response['Content-Disposition'] = 'attachment; filename="Request_maintenance
        .csv"'
    return response
```



Edit Your Profile Details

Username: dev Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

First name:

Last name:

Email address:

Save it

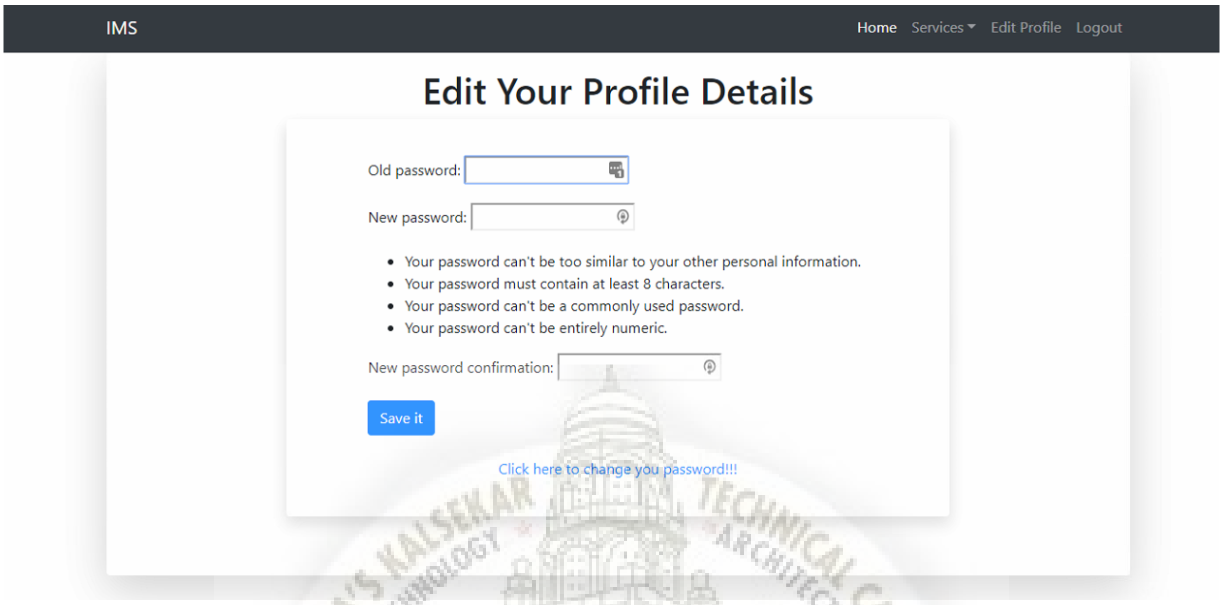
Click here to change you password!!!

Figure 6.12: Admin Edit Profile

The code below illustrates edit profile for admin

```
{% extends "accounts/base.html" %}

{% block content %}

<div class="container shadow-lg p-3 mb-5 bg-white rounded">
    <h1 class="text-center">Edit Your Profile Details</h1>
    <div class="col-md-8 offset-md-2 shadow p-3 mb-5 bg-white rounded">
        <br>
        <div class="col-md-10 offset-md-1">

            <form method="POST" action="{% url 'accounts:edit' %}">
                {% csrf_token %}
                {% if form.errors %}

                    <div class="alert alert-warning" role="alert">
                    <button class="close" data-dismissable="alert"><small><sup>X
                        </sup></small></button>
                    <p>Your Form has Errors...</p>
                    {% for field in form %}
                        {% if field.errors %}
                            {{ field.errors }}
                        {% endif %}
                    {% endfor %}
                    </div>
                {% endif %}
                {{ form.as_p }}
                <input type="submit" value="Save it" class="btn btn-primary
                    center-block"></input>
            </form>
            <br>
            <div class="text-center">
                <a href="{% url 'accounts:changepassword' %}">Click here to
                    change you password!!!</a>
            </div>
            <br>
        </div>
    </div>
```

```
</div>
{% endblock content %}
```



Edit Your Profile Details

Old password:

New password:

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

New password confirmation:

Save it

Click here to change you password!!!

Figure 6.13: Admin Change Password

The code below illustrates how to change password

```
{% extends "accounts/base.html" %}
{% block content %}
<div class="container shadow-lg p-3 mb-5 bg-white rounded">
   <h1 class="text-center">Change your Pawword.</h1>
    <div class="col-md-8 offset-md-2 shadow p-3 mb-5 bg-white rounded">
        <br>
        <div class="col-md-10 offset-md-1">

            <form method="POST" action="{% url 'accounts:edit' %}">
                {% csrf_token %}
                {% if form.errors %}

                    <div class="alert alert-warning" role="alert">
                    <button class="close" data-dismissable="alert"><small><sup>X
                        </sup></small></button>
                    <p>Your Form has Errors...</p>
                    {% for field in form %}
                        {% if field.errors %}
                            {{ field.errors }}
                        {% endif %}
                    {% endfor %}
                    </div>
                {% endif %}
                {{ form.as_p }}
                <input type="submit" value="Change Password" class="btn btn-
                    primary center-block"></input>
            </form>
        </div>
    </div>
</div>
{% endblock content %}
```

6.2 Requester Panel

The below code illustrates the base file for the overview of the requester's panel

```
  <!doctype html>
<html lang="en">
    <head>
        <!-- Required meta tags -->
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1,
            shrink-to-fit=no">

        <!-- Bootstrap CSS -->
        <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/
            bootstrap/4.2.1/css/bootstrap.min.css" integrity="sha384-
            GJzZqFGwb1QTTN6wy59ffF1BuGJpLSa9DkKMp0DgiMDm4iYMj70gZWKYbI706tWS"
            crossorigin="anonymous">
        <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.
            min.js"></script>
        <script src="https://cdnjs.cloudflare.com/ajax/libs/lodash.js/4.17.11/
            lodash.js"></script>

        <title>IMS</title>

        <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
            <div class="container">
                <a class="navbar-brand" href="{% url 'accounts:home'%}">IMS</a>
                <button class="navbar-toggler" type="button" data-toggle="
                    collapse" data-target="#navbarSupportedContent" aria-
                    controls="navbarSupportedContent" aria-expanded="false" aria
                    -label="Toggle navigation">
                <span class="navbar-toggler-icon"></span></button>

                <div class="collapse navbar-collapse" id="navbarSupportedContent
                    ">
                    <ul class="navbar-nav ml-auto">

                        {% if user.is_superuser %}
                       <script>window.location="http://127.0.0.1:8000/
                           admins_home";</script>
                        {% comment %} <li class="nav-item active">
                            <a class="nav-link" href="{% url 'accounts:
                                admins_home' %}">Admins Home <span class="sr-
                                only">(current)</span></a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" href="{% url 'accounts:logout'
                                %}">Logout</a>
                        </li> {% endcomment %}

                        {% elif user.is_staff %}

                                    <li class="nav-item active">
                                        <a class="nav-link" href="{% url 'accounts:
                                            staff_handle_req' %}"> Maintenance
                                            Request<span class="sr-only">(current)</
                                            span></a>
                                    </li>
                                    <li class="nav-item">
                                        <a class="nav-link" href="{% url 'accounts:
                                            logout' %}">Logout</a>
                        </li>

                {% elif user.is_authenticated and not user.is_staff %}

                <li class="nav-item active">
                    <a class="nav-link" href="{% url 'accounts:home' %}"
                        >Home <span class="sr-only">(current)</span></a>
                </li>

                <li class="nav-item dropdown">
                    <a class="nav-link dropdown-toggle" href="#" id="
                        navbarDropdown" role="button" data-toggle="
                        dropdown" aria-haspopup="true" aria-expanded="
                        false">
                    Services </a>
                    <div class="dropdown-menu" aria-labelledby="
                        navbarDropdown">
                    <a class="dropdown-item" href="{% url 'accounts:
                        status' %}">Status </a>
                    <a class="dropdown-item" href="{% url 'accounts:
                        user_issue_status' %}">Issue Status </a>
                    <a class="dropdown-item" href="{% url 'accounts:
                        user_maintenance_status' %}">Maintenance Status
                        </a>
                    <div class="dropdown-divider"></div>
                    <a class="dropdown-item" href="{% url 'accounts:req'
                        %}">New Issue Request </a>
                    <a class="dropdown-item" href="{% url 'accounts:
                        maintenance' %}">New Maintenance Request </a>
                    </div>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="{% url 'accounts:edit' %}"
                        >Edit Profile </a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="{% url 'accounts:logout'
                        %}">Logout </a>
                </li>

                {% else %}

                <li class="nav-item active">
                    <a class="nav-link" href="{% url 'accounts:home' %}"
                        >Home <span class="sr-only">(current)</span></a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="{% url 'accounts:login' %}
                        ">Login </a>
                {% comment %} </li>
                <li class="nav-item">
                    <a class="nav-link" href="{% url 'accounts:
                        admins_login' %}">Admin</a>
                </li> {% endcomment %}

                {% endif %}




            </ul>
```

```
                </div>
            </div>
        </nav>
    </head>

    <body>
        <div class="container">
            {% if messages %}
                {% for message in messages %}
                    <div class="alert alert-warning" role="alert">
                        <button class="close" data-dismissable="alert"><small><
                            sup>X</sup></small></button>
                        {{ message }}
                    </div>
                {% endfor %}
            {% endif %}

            {% block content %}
            {% endblock %}
        </div>

        <!-- Optional JavaScript -->
        <!-- jQuery first, then Popper.js, then Bootstrap JS -->
        <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity
            ="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8
            abtTE1Pi6jizo" crossorigin="anonymous"></script>
        <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.6/umd
            /popper.min.js" integrity="sha384-wHAiFfRlMFy6i5SRaxvfOCifBUQy1xHdJ/
            yoi7FRNXMRBu5WHdZYu1hA6ZOblgut" crossorigin="anonymous"></script>
        <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/js/
            bootstrap.min.js" integrity="sha384-B0UglyR+
            jN6CkvvICOB2joaf5I4l3gm9GU6Hc1og6Ls7i6U/mkkaduKaBhlAXv9k"
            crossorigin="anonymous"></script>
    </body>
</html>
```

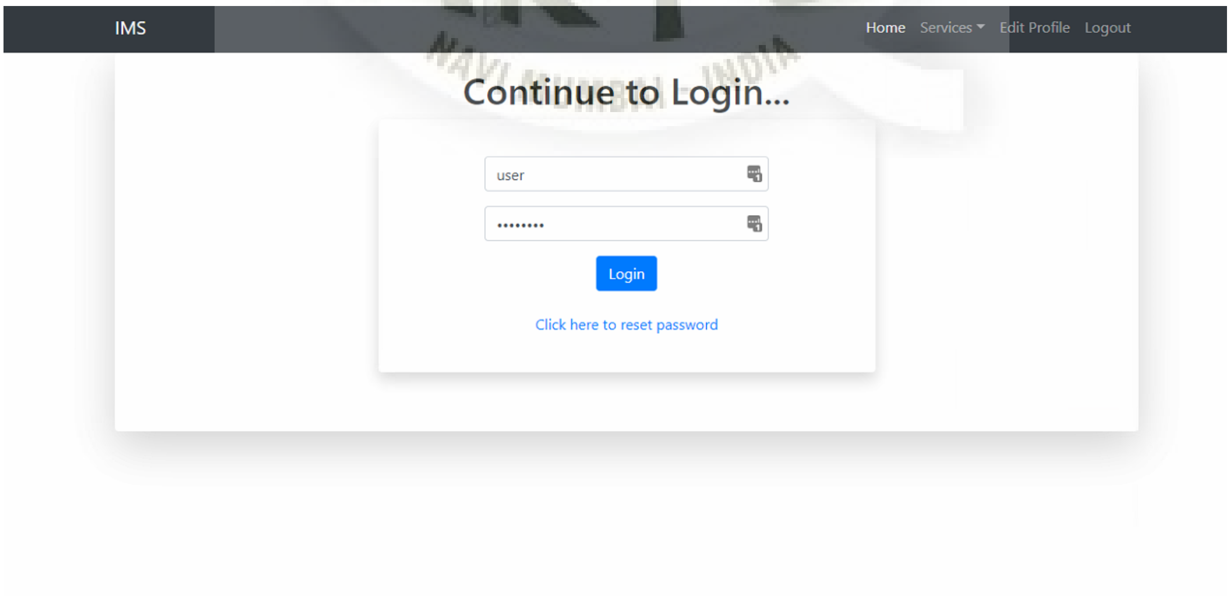
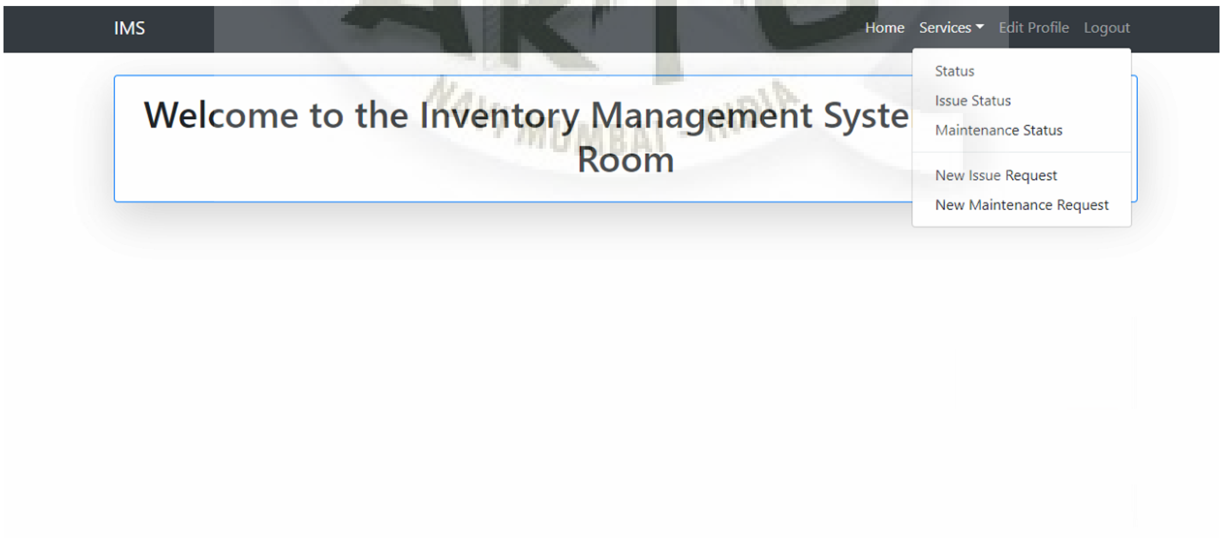


Figure 6.14: Requester Login Portal

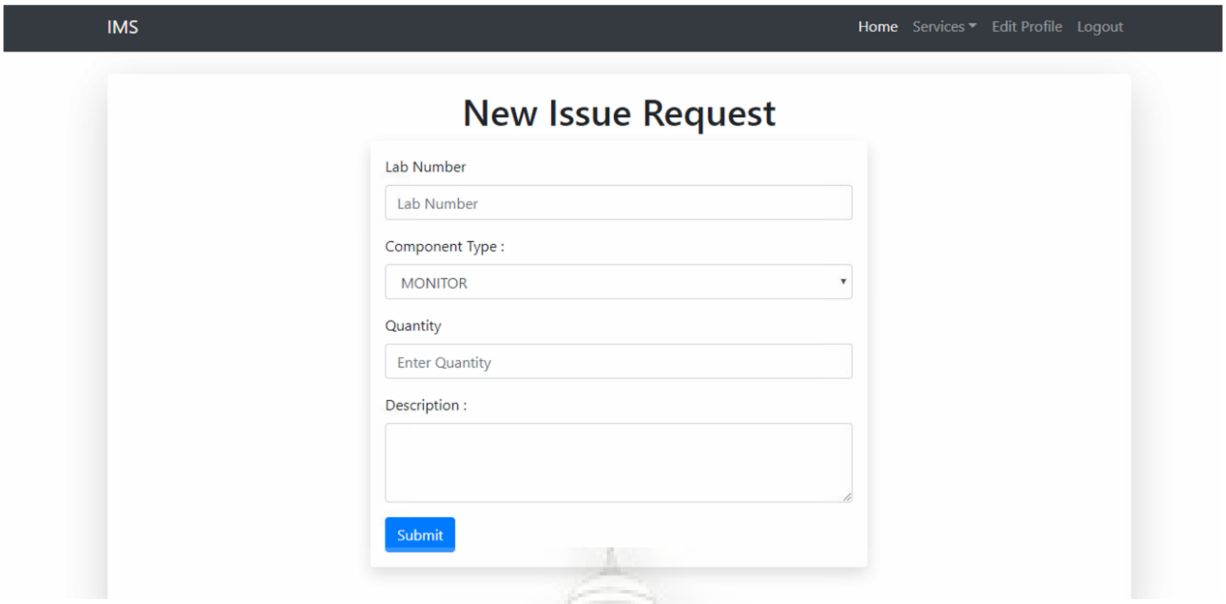
The below code illustrates the login system for the Requester

```
{% extends "accounts/base.html" %}
{% block content %}
<div class="container shadow-lg p-3 mb-5 bg-white rounded">
    <h1 class="text-center">Continue to Login...</h1>
    <div class="col-md-6 offset-md-3 shadow p-3 mb-5 bg-white rounded">
        <br>
        <div class="col-md-8 offset-md-2">
            <form method="POST">
            {% csrf_token %}
                <div class="form-group">
                    <input type="text" class="form-control" placeholder="Enter
                       Username" name="username">
                </div>
                <div class="form-group">
                    <input type="password" class="form-control" placeholder="
                       Enter Password" name="password">
                </div>
                <div class="text-center">
                    <button type="submit" class="btn btn-primary center-block">
                       Login</button>
                </div>
            </form>

            <br>
            <div class="text-center">
            <a href="#">Click here to reset password</a>
            </div>
            <br>
        </div>
    </div>
</div>
{% endblock content %}
```



Welcome to the Inventory Management Syste
Room

Figure 6.15: Requester Services



New Issue Request

Lab Number
Lab Number
Component Type :
MONITOR
Quantity
Enter Quantity
Description :
Submit

Figure 6.16: New Issue Request

The code below illustrates new issue request into the system

```
{% extends "accounts/base.html" %}

{% block content %}

<br>
<div class="container shadow-lg p-3 mb-5 bg-white rounded">
    <h1 class="text-center">New Issue Request</h1>
    <div class="col-md-6 offset-md-3 shadow p-3 mb-5 bg-white rounded">

    <form method="POST" action="{% url 'accounts:req' %}">
    {% csrf_token %}
        <div class="form-group">
            <label >Lab Number</label>
            <input type="Number" class="form-control" name = "lab_number"
                placeholder="Lab Number">
        </div>

        <div class="form-group">
            <label >Component Type :</label>
            <select class="form-control"  name = "component_type">
                <option>MONITOR</option>
                <option>CPU</option>
                <option>KEYBOARD</option>
                <option>MOUSE</option>
                <option>LAPTOP</option>
                <option>STORAGE DEVICES</option>
                <option>SOFTWARES</option>
                <option>GATEWAY</option>
                <option>ROUTER</option>
                <option>SWITCH</option>
                <option>REPEATER</option>
                <option>LAN CABLE</option>
                <option>PRINTER</option>
                <option>PROJECTOR</option>
                <option>POWER CABLE</option>
                <option>PROJECTOR CABLE</option>
                </select>
            </div>
            <div class="form-group">
                <label >Quantity </label>
                <input type="Number" class="form-control" placeholder="Enter
                    Quantity" name = "quantity">
            </div>
            <div class="form-group">
                <label >Description :</label>
                <textarea class="form-control"  rows="3" name = "description"></
                    textarea>
            </div>
            <div> <button type="submit" class="btn btn-primary center-block">Submit
                </button></div>
        </form>
        </div>
</div>
{% endblock content %}
```

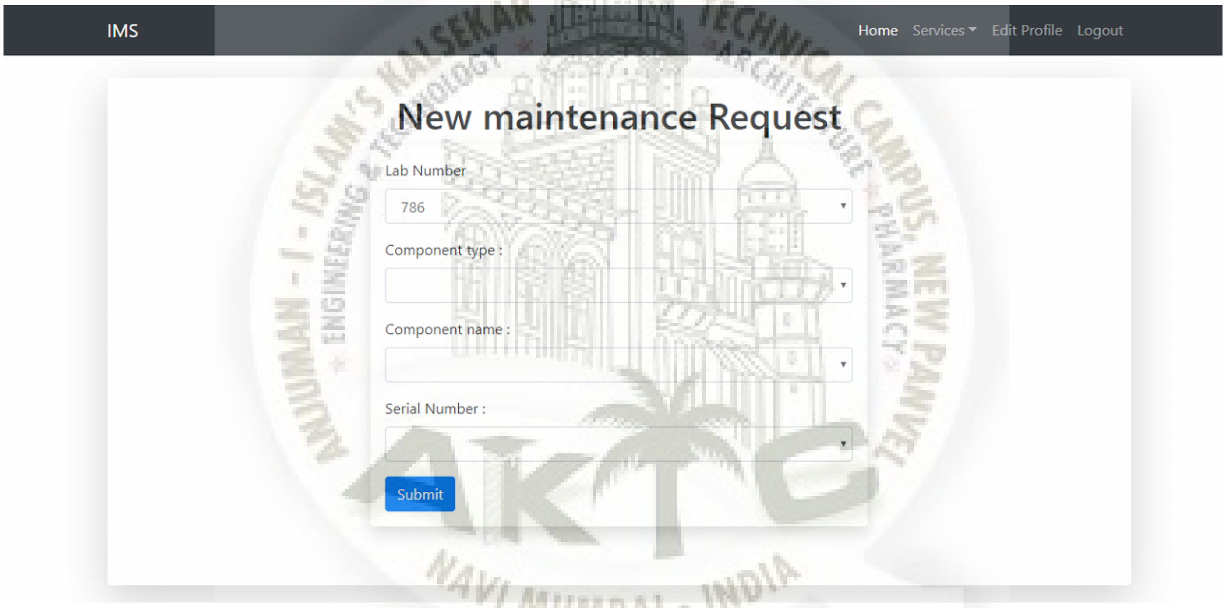


Figure 6.17: New Maintenance Request

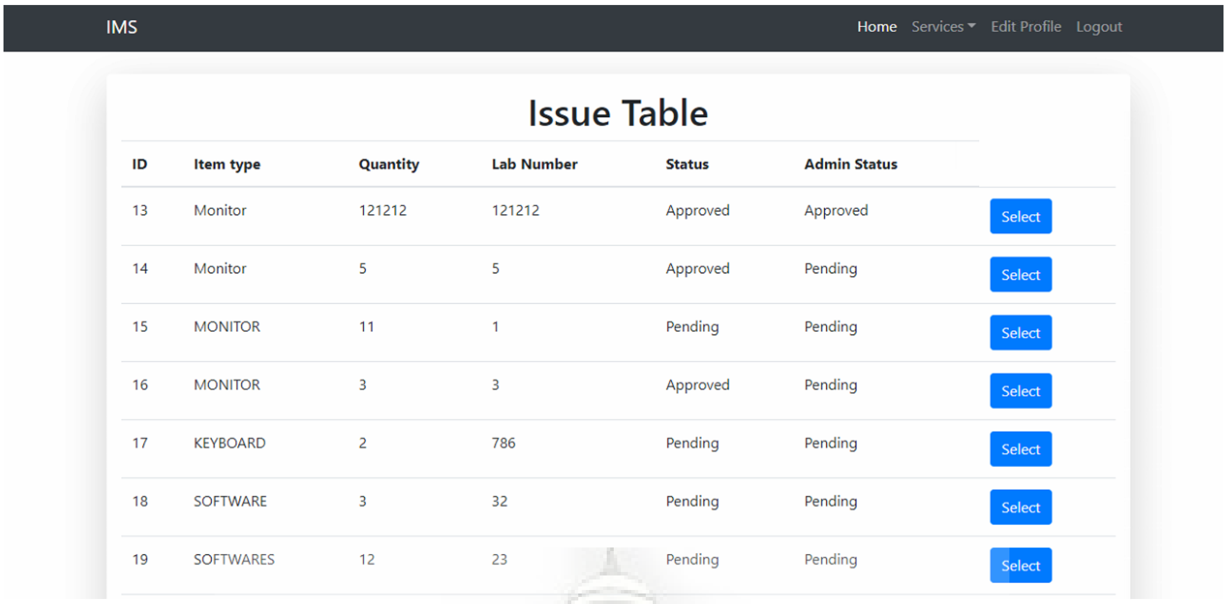
The code below illustrates new maintenance request into the system

```
{% extends "accounts/base.html" %}
{% block content %}
<br>
{{ mt }}
<div class="container shadow-lg p-3 mb-5 bg-white rounded">
    <h1 class="text-center">New maintenance Request</h1>
    <div class="col-md-6 offset-md-3 shadow p-3 mb-5 bg-white rounded">

    <form method="POST" action="{% url 'accounts:user_maintenance_req' %}">
    {% csrf_token %}
        <div class="form-group">
            <label for="exampleFormControlSelect1">Lab Number</label>
            <select onchange="labNumber(this.value)" class="form-control" id="
                labType" name="lab_number">
            <script>
```

```
                    var val = ('{{mt}}').toString();
                    //console.log(val);
                    var obj=val.replace(/&quot;/g,'"');
                    var obj= JSON.parse(obj);
                   // console.log(obj);
                   // console.log(obj.length);
                    var labs = [];
                    const uniqueArray = (arr) => [...new Set(arr)];
                    let arr=[];
                    for(i=1;i<=obj.length;i++){           if($.inArray(""+obj[i-1].
                        fields.lab_number,labs))
                            {                               arr.push(obj[i-1].fields.
                                lab_number);
                              // document.write('<option>'+obj[i-1].fields.
                                  lab_number+'</option>');
                            }
                    }
                     uniqueArray(arr).forEach(function(entry){
                        document.write('<option>'+entry+'</option>');
                    })
    //              console.log("Array Logging");
    //               console.log(uniqueArray(arr));

               </script>
               <script>
               labNumbers = 0;
               componentTypes = "";
               componentNames = "";
               serialNumbers = "";

                       function labNumber(selectedLab){
                           labNumbers = selectedLab;
                      var val = ('{{mt}}').toString();
                     //console.log(val);
                      var obj=val.replace(/&quot;/g,'"');
                      var obj= JSON.parse(obj);
                       var output = [];
                       var c = [];
                          for(var i=0;i<obj.length;i++)
                          {
                              for(var name in obj[i])
                              {
                                  var lab = obj[i][name].lab_number;
                                  if(lab == selectedLab)
                                  {
                                      if(output.includes('<option value="'+obj[i][
                                          name].component_type+'">'+obj[i][name].
                                          component_type+'</option>'))
                                      {

                                      }
                                      else
                                      {
                                       output.push('<option value="'+obj[i][name].
                                           component_type+'">'+obj[i][name].
                                           component_type+'</option>');

                                      }
                                  }
                              }
                          }
```

```
                                    $('#componentType').html(output.join(''));



                    }
                </script>
                </select>
            </div>
        <script>
            function selectComponent(selectedComponent)
            {
                    componentTypes = selectedComponent;
                         var val = ('{{mt}}').toString();
                        //console.log(val);
                         var obj=val.replace(/&quot;/g,'"');
                         var obj= JSON.parse(obj);
                          var output = [];
                            for(var i=0;i<obj.length;i++)
                            {
                                for(var name in obj[i])
                                {
                                    var cType = obj[i][name].component_type;
                                    if(cType == selectedComponent && labNumbers ==
                                        obj[i][name].lab_number)
                                    {

                                         output.push('<option value="'+obj[i][name].
                                            component_name+'">'+obj[i][name].
                                            component_name+'</option>');
                                    }
                                   $('#componentName').html(output.join(''));
                                }
                            }
            }
        </script>
            <div class="form-group">
                <label >Component type :</label>
                {% comment %} <select onchange="val_change(event)" class="form-
                    control" id="exampleFormControlSelect1"> {% endcomment %}
                <select onclick="selectComponent(this.value)" class="form-control"
                    id="componentType" name="component_type">
                <script>
                    var val = ('{{mt}}').toString();
                    //console.log(val);
                    var obj=val.replace(/&quot;/g,'"');
                    var obj= JSON.parse(obj);
    //              console.log(obj);
     //             console.log(obj.length);
                    val = e.target.value
                    for(i=1;i<=obj.length;i++){
                        if obj[i-1].fields.lab_number == val
                        document.write('<option>'+obj[i-1].fields.component_type+'</
                            option>');
                    }
                </script>
                </select>
                {% comment %}
                {% for eis in mt %}
                    <option>{{ eis.id }}</option>
                {% endfor %}
                </select>
                <input type="text" class="form-control" value="{{ mt.1.
                    component_type }}" name="component_type" disabled> {% endcomment
                    %}
            </div>

            <script>
                function selectComponentName(componentName)
                {
                    componentNames = componentName;
                        var val = ('{{mt}}').toString();
                        //console.log(val);
                        var obj=val.replace(/&quot;/g,'"');
                        var obj= JSON.parse(obj);
                         var output = [];
                           for(var i=0;i<obj.length;i++)
                           {
                               for(var name in obj[i])
                               {
                                   var nameType = obj[i][name].component_name;
                                   if(nameType == componentName && componentTypes
                                       == obj[i][name].component_type
                                   && componentNames == obj[i][name].component_name
                                       )
                                   {
                                    output.push('<option value="'+obj[i][name].
                                        component_serial_number+'">'+obj[i][name].
                                        component_serial_number+'</option>');
                                   }
                                  $('#serialNumber').html(output.join(''));
                               }
                           }
                }
            </script>
            <div class="form-group">
                <label >Component name :</label>
                <select onclick="selectComponentName(this.value)" class="form-
                    control" id="componentName" name="component_name">
                <script>
                    var val = ('{{mt}}').toString();
                    //console.log(val);
                    var obj=val.replace(/&quot;/g,'"');
                    var obj= JSON.parse(obj);
        ///            console.log(obj);
            //          console.log(obj.length);
                    for(i=1;i<=obj.length;i++){
//                        document.write('<option >'+obj[i-1].fields.component_name
    +'</option >');
                    }
                </script>
                </select>
            </div>

<script>
    function selectSerial(s)
    {
        serialNumbers  = s;
    }
</script>
            <div class="form-group">
                <label >Serial Number :</label>
                <select onclick="selectSerial(this.value)" class="form-control" id="
```

```
                    serialNumber" name="serial_number">
                {<script>
    //              var val = ('{{mt}}').toString();
                  //console.log(val);
   //               var obj=val.replace(/&quot;/g,'"');
  //                var obj= JSON.parse(obj);
                //      console.log(obj);
                 //     console.log(obj.length);
 //                 for(i=1;i<=obj.length;i++){
//                      document.write('<option>'+obj[i-1].fields.
    component_serial_number+'</option>');
  //                }
                </script>
                </select>
            </div>

            <div> <button type="submit" class="btn btn-primary center-block">Submit
                </button></div>

    {% comment %} {% else %}<h1>Empty DataSet</h1>
    {% endif %} {% endcomment %}
    </form>
    </div>
</div>

<script>
    function val_change(e)
    {
    //      console.log(e.target.value);
        var labNo = e.target.value;
    }

    //console.log(val);
    function convert(){
        var val = ('{{mt}}').toString();
        var obj=val.replace(/&quot;/g,'"');
        var obj= JSON.parse(obj);
        return obj;
       // {% comment %} console.log(obj);
       // console.log(obj.length); {% endcomment %}
    }

    //console.log("converted"+val.replace(/&quot;/g,'"'));
    //eval(val.quote());

    //console.log(val);
    for(i=0;i<obj.length;i++){
    }
</script>
{% endblock content %}
```



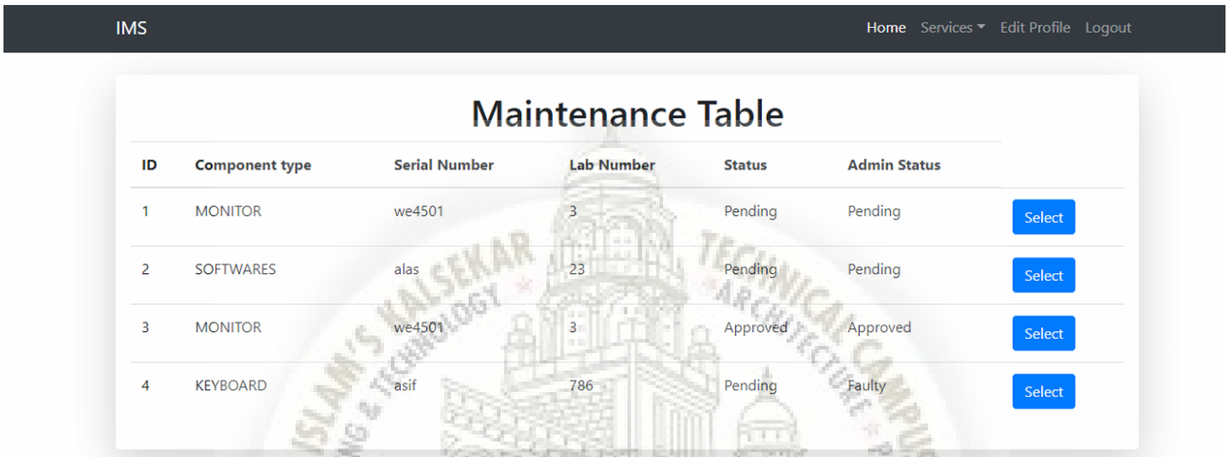
Issue Table

ID	Item type	Quantity	Lab Number	Status	Admin Status	
13	Monitor	121212	121212	Approved	Approved	Select
14	Monitor	5	5	Approved	Pending	Select
15	MONITOR	11	1	Pending	Pending	Select
16	MONITOR	3	3	Approved	Pending	Select
17	KEYBOARD	2	786	Pending	Pending	Select
18	SOFTWARE	3	32	Pending	Pending	Select
19	SOFTWARES	12	23	Pending	Pending	Select

Figure 6.18: View Issue Requests

The below code illustrates the working of view issue request from the requester point of view

```
{% extends "accounts/base.html" %}
{% block content %}
<br>

<div class="container shadow-lg p-3 mb-5 bg-white rounded">
    <h1 class="text-center">Issue Table</h1>
    <table class="table">
        <thead>
            <tr>
                <th scope="col">ID</th>
                <th scope="col">Item type</th>
                <th scope="col">Quantity</th>
                <th scope="col">Lab Number</th>
                <th scope="col">Status</th>
                <th scope="col">Admin Status</th>
            </tr>
            </thead>
        <tbody>
        {% if es is null %}
            <tr>
                <td>Null</td>
                <td>Null</td>
                <td>Null</td>
                <td>Null</td>
                <td>Null</td>
            </tr>
        {% else %}
            {% for eis in es %}
            <tr>
                <td>{{ eis.id }}</td>
                <td>{{ eis.component_type }}</td>
                <td>{{ eis.quantity }}</td>
                <td>{{ eis.lab_number }}</td>
                <td>{{ eis.status }}</td>
                <td>{{ eis.admin_status }}</td>
                <td><a href="{% url 'accounts:user_issue_update' eis.id %}"
                    class="btn btn-primary" role="button">Select</a></td>

            </tr>
            {% endfor %}
        {% endif %}
        </tbody>
    </table>
</div>
{% endblock %}
```


IMS Home Services Edit Profile Logout

Maintenance Table

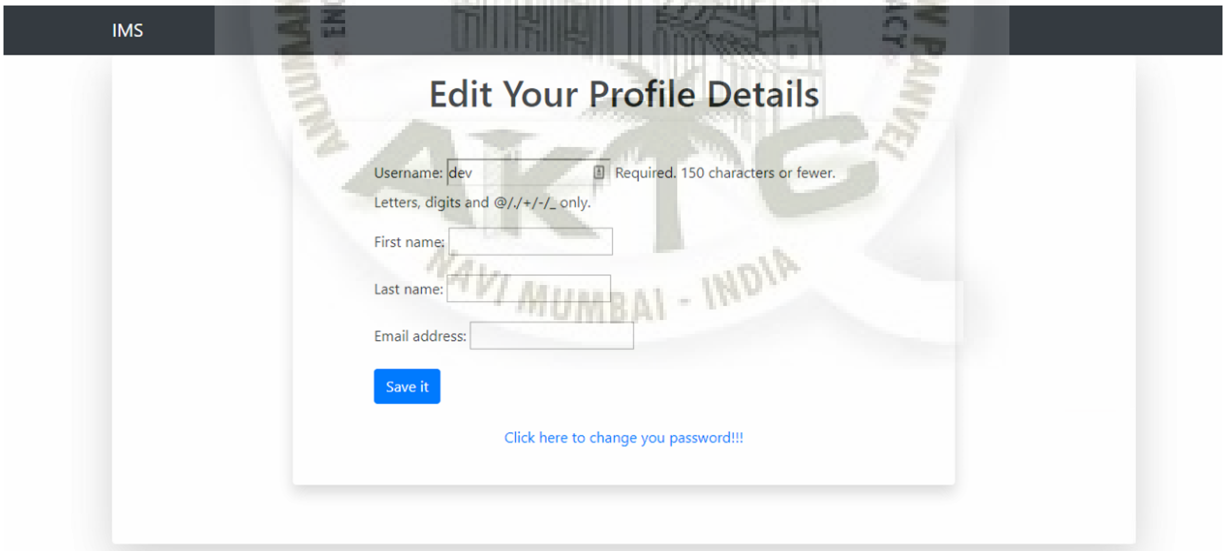
ID	Component type	Serial Number	Lab Number	Status	Admin Status	
1	MONITOR	we4501	3	Pending	Pending	Select
2	SOFTWARES	alas	23	Pending	Pending	Select
3	MONITOR	we4501	3	Approved	Approved	Select
4	KEYBOARD	asif	786	Pending	Faulty	Select

Figure 6.19: View Maintenance Request

The below code illustrates the working of view maintenance requests from requester perspective

```
{% extends "accounts/base.html" %}

{% block content %}

<br>

<div class="container shadow-lg p-3 mb-5 bg-white rounded">
    <h1 class="text-center">Maintenance Table</h1>
    <table class="table">
        <thead>
            <tr>
                <th scope="col">ID</th>
                <th scope="col">Component type</th>
                <th scope="col">Serial Number</th>
                <th scope="col">Lab Number</th>
                <th scope="col">Status </th>
                <th scope="col">Admin Status </th>
            </tr>
            </thead>
        <tbody>
        {% if es is null %}
```

```
            <tr>
                <td>Null</td>
                <td>Null</td>
                <td>Null</td>
                <td>Null</td>
                <td>Null</td>
                <td>Null</td>

            </tr>
        {% else %}
            {% for eis in es %}
            <tr>
                <td>{{ eis.id }}</td>
                <td>{{ eis.component_type }}</td>
                <td>{{ eis.component_serial_number }}</td>
                <td>{{ eis.lab_number }}</td>
                <td>{{ eis.status }}</td>
                <td>{{ eis.admin_status }}</td>
                <td><a href="{% url 'accounts:user_maintenance_update' eis.id %}
                    " class="btn btn-primary" role="button">Select</a></td>

            </tr>
            {% endfor %}
        {% endif %}
        </tbody>
    </table>
</div>
{% endblock %}
```



Edit Your Profile Details

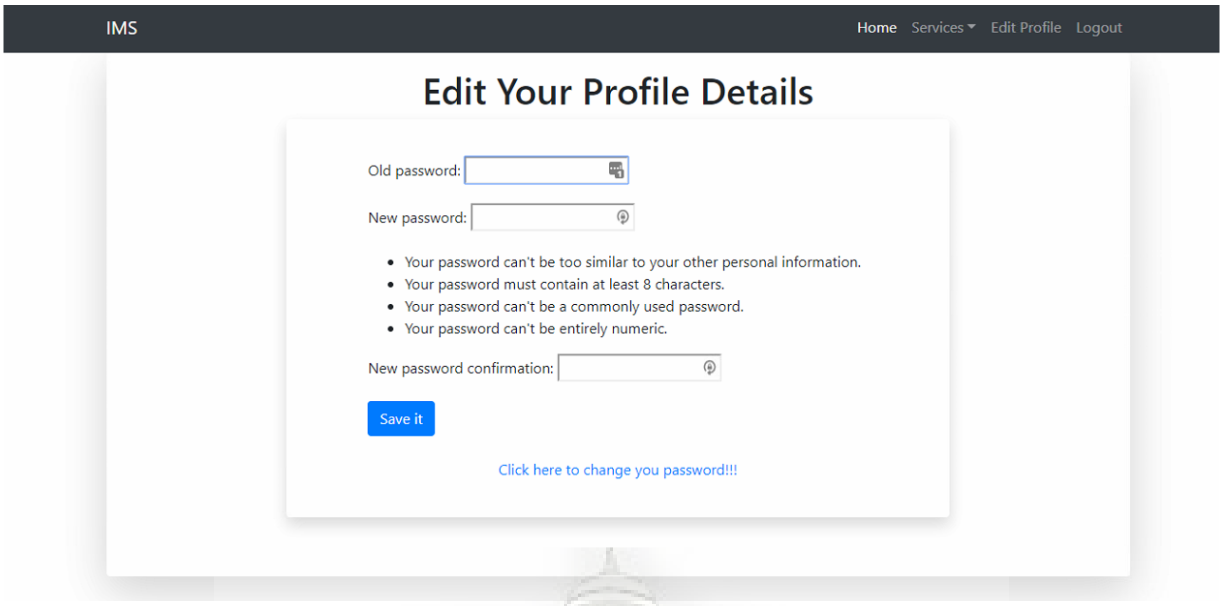
Username: dev Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

First name:

Last name:

Email address:

Figure 6.20: User Edit Profile



Edit Your Profile Details

Old password:

New password:

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

New password confirmation:

Save it

Click here to change you password!!!

Figure 6.21: User Change Password

The code below illustrates changing user profiles and change password

```
{% extends "accounts/base.html" %}

{% block content %}

<div class="container shadow-lg p-3 mb-5 bg-white rounded">
    <h1 class="text-center">Edit Your Profile Details</h1>
    <div class="col-md-8 offset-md-2 shadow p-3 mb-5 bg-white rounded">
        <br>
        <div class="col-md-10 offset-md-1">

            <form method="POST" action="{% url 'accounts:edit' %}">
                {% csrf_token %}
                {% if form.errors %}

                    <div class="alert alert-warning" role="alert">
                    <button class="close" data-dismissable="alert"><small><sup>X
                        </sup></small></button>
                    <p>Your Form has Errors...</p>
                    {% for field in form %}
                        {% if field.errors %}
                            {{ field.errors }}
                        {% endif %}
                    {% endfor %}
                    </div>
                {% endif %}
                {{ form.as_p }}
                <input type="submit" value="Save it" class="btn btn-primary
                    center-block"></input>
            </form>
            <br>
            <div class="text-center">
                <a href="{% url 'accounts:changepassword' %}">Click here to
                    change you password!!!</a>
            </div>
            <br>
        </div>
    </div>
```

```
</div>

{% endblock content %}
```

```

{% extends "accounts/base.html" %}

{% block content %}

<div class="container shadow-lg p-3 mb-5 bg-white rounded">
   <h1 class="text-center">Change your Pawword.</h1>
    <div class="col-md-8 offset-md-2 shadow p-3 mb-5 bg-white rounded">
        <br>
        <div class="col-md-10 offset-md-1">

            <form method="POST" action="{% url 'accounts:edit' %}">
                {% csrf_token %}
                {% if form.errors %}

                    <div class="alert alert-warning" role="alert">
                    <button class="close" data-dismissable="alert"><small><sup>X
                        </sup></small></button>
                    <p>Your Form has Errors...</p>
                    {% for field in form %}
                        {% if field.errors %}
                            {{ field.errors }}
                        {% endif %}
                    {% endfor %}
                    </div>
                {% endif %}
                {{ form.as_p }}
                <input type="submit" value="Change Password" class="btn btn-
                    primary center-block"></input>
            </form>
        </div>
    </div>
</div>

{% endblock content %}
```

6.3 Maintenance Engineer Panel

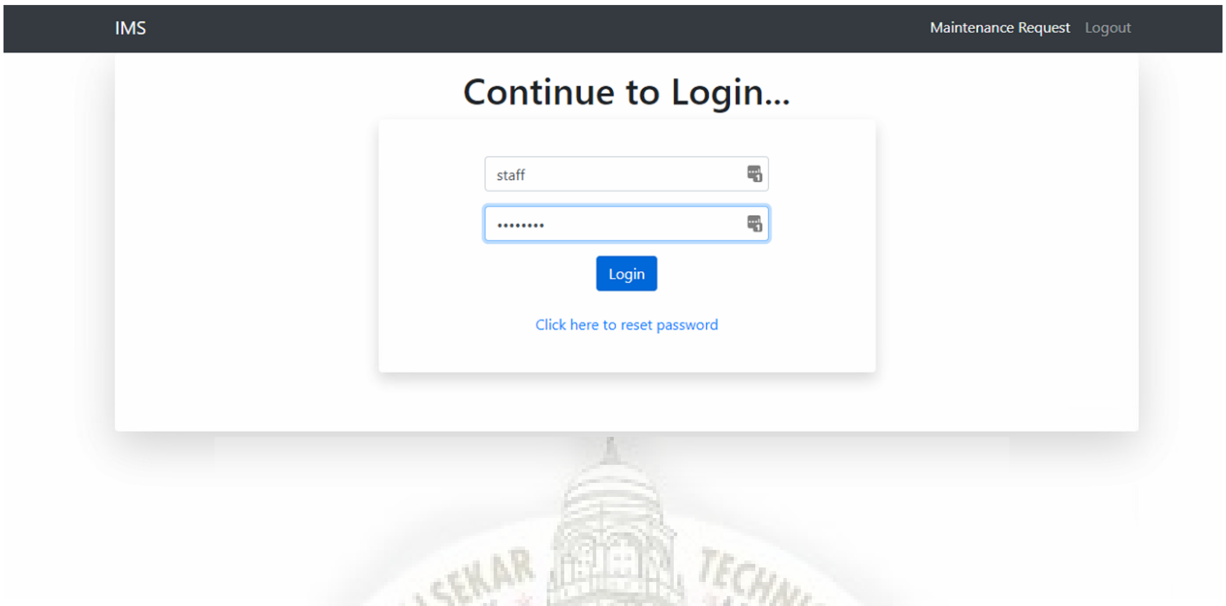
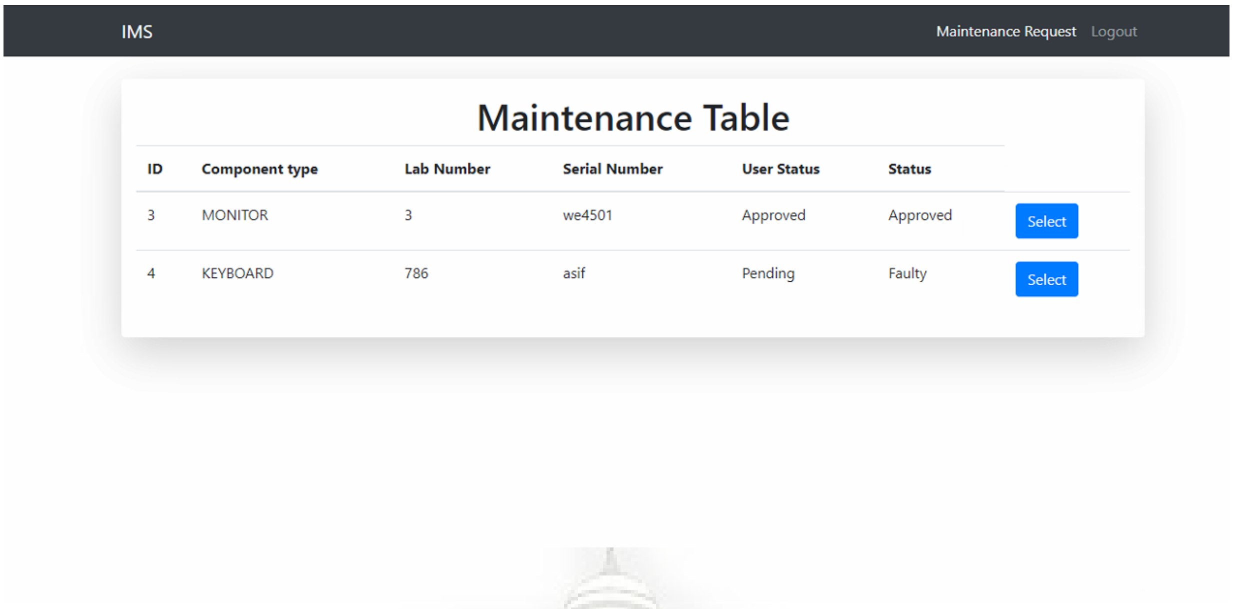


Figure 6.22: Maintenance Engineer Login Form

```
{% extends "accounts/base.html" %}
{% block content %}
<div class="container shadow-lg p-3 mb-5 bg-white rounded">
    <h1 class="text-center">Continue to Login...</h1>
    <div class="col-md-6 offset-md-3 shadow p-3 mb-5 bg-white rounded">
        <br>
        <div class="col-md-8 offset-md-2">
            <form method="POST">
            {% csrf_token %}
                <div class="form-group">
                    <input type="text" class="form-control" placeholder="Enter
                       Username" name="username">
                </div>
                <div class="form-group">
                    <input type="password" class="form-control" placeholder="
                       Enter Password" name="password">
                </div>
                <div class="text-center">
                    <button type="submit" class="btn btn-primary center-block">
                       Login</button>
                </div>
            </form>

            <br>
            <div class="text-center">
            <a href="#">Click here to reset password</a>
            </div>
            <br>
        </div>
    </div>
</div>

{% endblock content %}
```



Maintenance Table

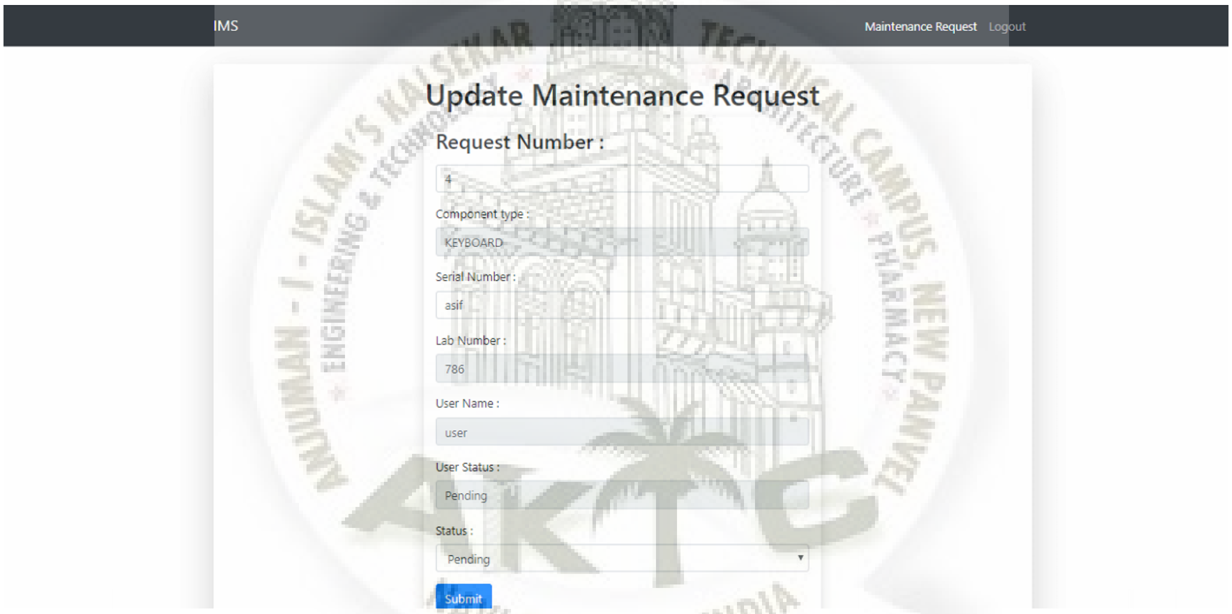
ID	Component type	Lab Number	Serial Number	User Status	Status	
3	MONITOR	3	we4501	Approved	Approved	Select
4	KEYBOARD	786	asif	Pending	Faulty	Select

Figure 6.23: View Maintenance Requests

The maintenance Engineer will view the maintenance jobs assigned to him by the Administrator

```
 {% extends "accounts/base.html" %}

{% block content %}

<br>

<div class="container shadow-lg p-3 mb-5 bg-white rounded">
    <h1 class="text-center">Maintenance Table</h1>
    <table class="table">
        <thead>
            <tr>
                <th scope="col">ID</th>
                <th scope="col">Component type</th>
                <th scope="col">Lab Number</th>
                <th scope="col">Serial Number</th>
                <th scope="col">User Status</th>
                <th scope="col">Status</th>

            </tr>
            </thead>
        <tbody>
        {% if remo is null %}
            <tr>
                <td>Null</td>
                <td>Null</td>
                <td>Null</td>
                <td>Null</td>
                <td>Null</td>
                <td>Null</td>


            </tr>
        {% else %}
            {% for eis in remo %}
            <tr>
```

```
                    <td>{{ eis.id }}</td>
                    <td>{{ eis.component_type }}</td>
                    <td>{{ eis.lab_number }}</td>
                    <td>{{ eis.component_serial_number }}</td>
                    <td>{{ eis.status }}</td>
                    <td>{{ eis.admin_status }}</td>
                    <td><a href="{% url 'accounts:staff_handle_update' eis.id %}"
                        class="btn btn-primary" role="button">Select </a></td>

                </tr>
                {% endfor %}
            {% endif %}
            </tbody>
        </table>
</div>

{% endblock %}
```



Update Maintenance Request

Request Number :

4

Component type :

KEYBOARD

Serial Number :

asif

Lab Number :

786

User Name :

user

User Status :

Pending

Status :

Pending

Figure 6.24: Update Maintenance Requests

The Maintenance Engineer, Once done with fault diagnosis will update the request stats of the component into the system

```

{% extends "accounts/base.html" %}

{% block content %}

<br>

<div class="container shadow-lg p-3 mb-5 bg-white rounded">
    <h1 class="text-center">Update Issue Request</h1>
    <div class="col-md-6 offset-md-3 shadow p-3 mb-5 bg-white rounded">

    <form method="POST" action="{% url 'accounts:staff_handle_edit' %}">
    {% csrf_token %}
    {% if remo is not Null %}
    {% for eis in remo %}

    <div class="form-group">
         <label> <h3>Request Number :</h3></label>
         <input type="Number" class="form-control" value="{{ eis.id }}" name="
             ids" >
    </div>
        <div class="form-group">
            <label >Component type :</label>
            <input type="text" class="form-control" value="{{ eis.component_type
                 }}" name="component_type" disabled>
        </div>
        <div class="form-group">
            <label >Serial Number :</label>
            <input type="text" class="form-control" value="{{ eis.
                component_serial_number }}" name="serial_number" >
        </div>

        <div class="form-group">
            <label >Lab Number :</label>
            <input type="Number" class="form-control" value="{{ eis.lab_number
                }}" name="lab_number" disabled>
        </div>

        <div class="form-group">
            <label >User Name :</label>
            <input type="text" class="form-control" value="{{ eis.user_name }}"
                name="user" disabled>
        </div>

        <div class="form-group">
            <label >User Status :</label>
            <input type="text" class="form-control" value="{{ eis.status }}"
                name = "status" disabled>
        </div>

        <div class="form-group">
            <label >Status :</label>
                <select class="form-control" name="admin_status">
                    <option>Pending</option>
                    <option>Approved</option>
                    <option>Faulty</option>
                </select>
        </div>
        <div> <button type="submit" class="btn btn-primary center-block">Submit
            </button></div>
    {% endfor %}
    {% else %}<h1>Empty DataSet</h1>
    {% endif %}
    </form>
    </div>
</div>
{% endblock %}
```

Chapter 7

System Testing

To make sure that our system does not have any constraints, we need to do system testing to make sure our system is performing well. Following are the lists of tests to be made.

7.1 Test Cases and Test Results

Test ID	Test Case Title	Test Condition	System Behavior	Expected Result
T01	Login	Should be register user	Will connect to database	Should login into system.
T02	Enter Issue Request	The values must be true	Will get added to the database if correct entries are made	Request gets added to the system
T03	Enter Maintenance Request	The values must be true	Will get added to the database if correct entries are made	Request gets added to the system
T04	Issue a component	The component should get issued	component should get decremented from available inventory	Component gets issued to the requester
T05	Update a maintenance request	There must be a maintenance request	Engineer should be able to update the system	Request gets updated from the system

Table 7.1: Table of Test Cases

7.2 Login Test Case

Title: Login – Authenticated Successfully from database.
Description: A registered user should be able to login successfully.

Precondition: the user must already be registered with username and password.

Assumption: All the credentials are true.

Test Steps:

1. Navigate to Login Page
2. In the Username field, enter the Username of the registered user.
3. Enter the password of the registered user
4. Click 'Log In'

Expected Result: A new page displaying the users functionalities and statuses.

Actual Result: User gets logged in successfully and then the user portal is appeared.

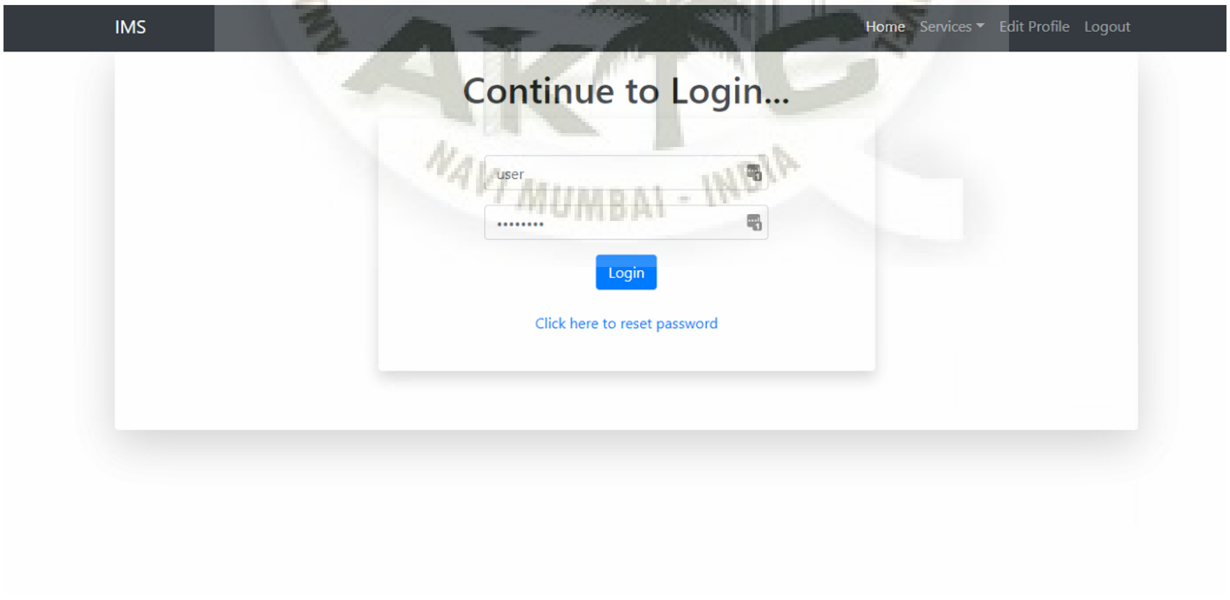
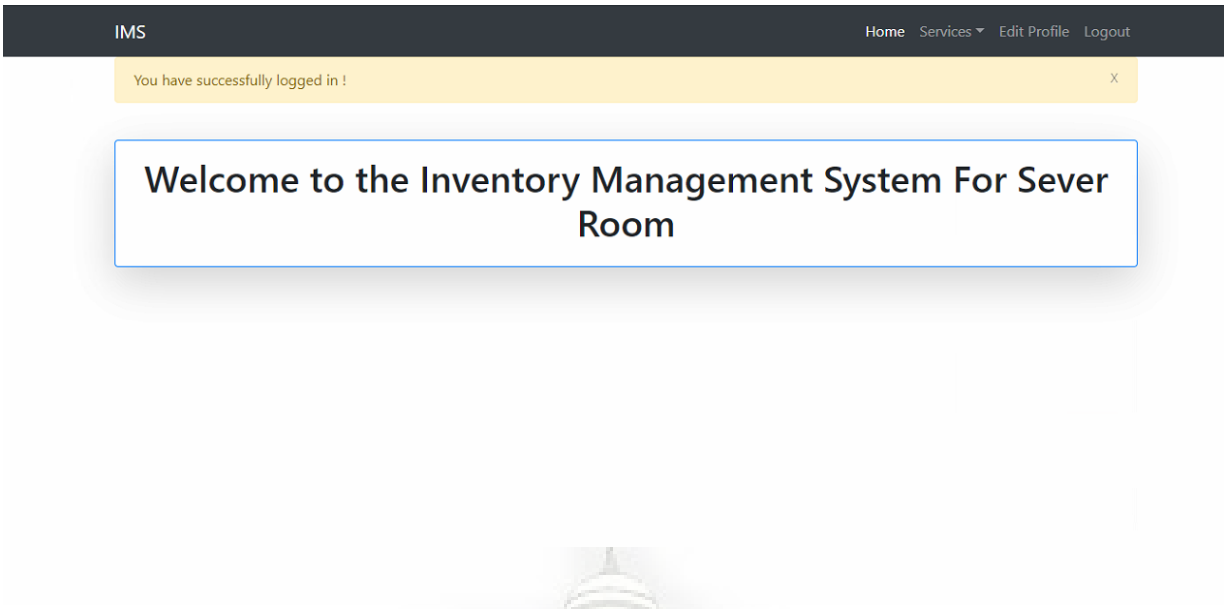


Figure 7.1: Test Case Before Login



You have successfully logged in !

Welcome to the Inventory Management System For Sever Room

Figure 7.2: Test Case After Login

7.3 New Issue Request Test Case

Title: Enter Issue request – Request successfully added to the system.
Description: A registered user should be able to raise an issue request to the administrator.

Precondition: The details entered by the user are up to the requirements.

Assumption: Requested component is available in the Inventory for issuing.

Test Steps:

1. Navigate to New Issue Request Page
2. The requester must enter the Lab Number.
3. Then the requester must enter the component name that is requested
4. The requester must enter the quantity of components required
5. Then the requester must enter the description of the request
6. Then click on Submit button

Expected Result: The Request is updated on the issue request table.

Actual Result: The Request gets updated in the system and administrator can view the requests.

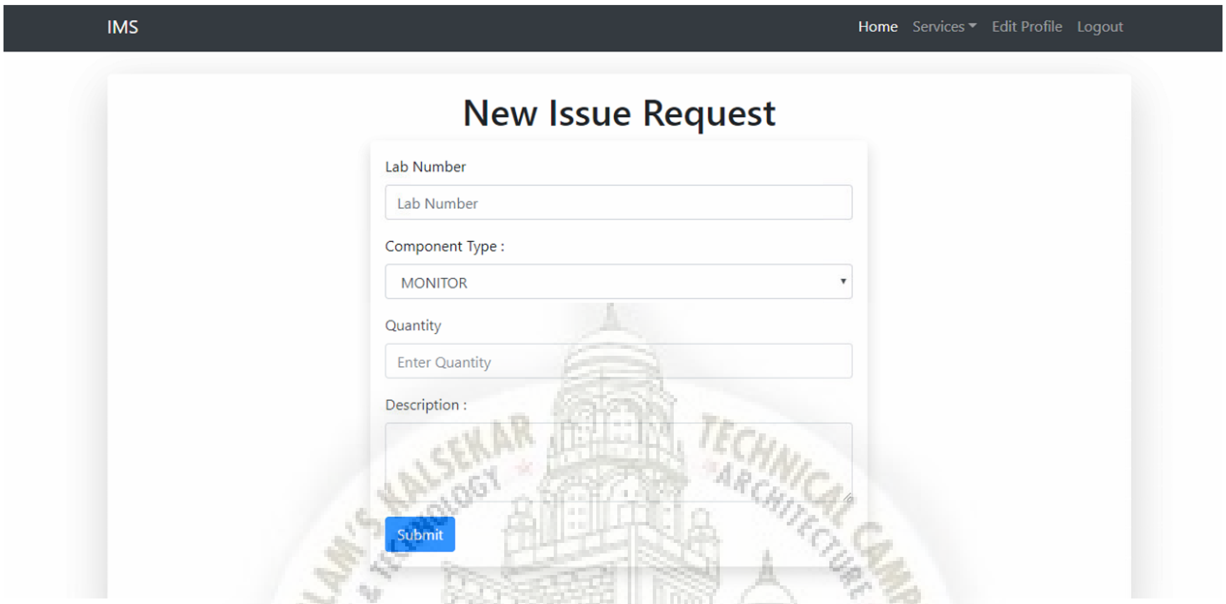
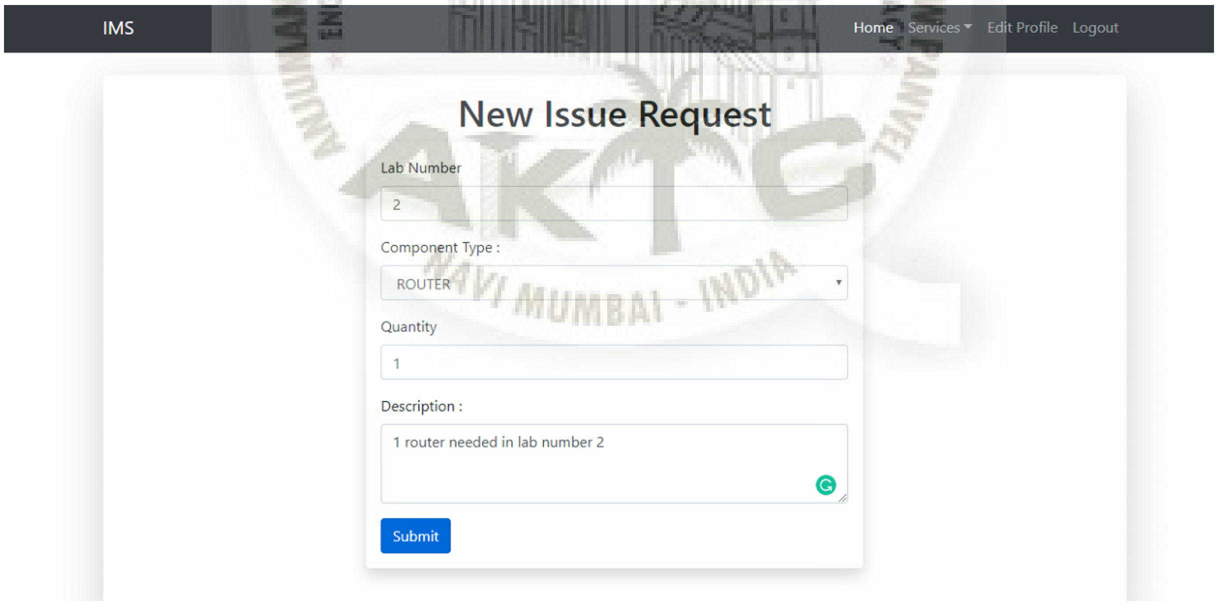


Figure 7.3: Test Case before Entering Data



Lab Number: 2
Component Type : ROUTER
Quantity: 1
Description : 1 router needed in lab number 2

Figure 7.4: Test Case After Entering Data and Submitting

7.4 New Maintenance Request Test Case

Title: Enter Maintenance Request – Request successfully added to the system.
Description: A registered user should be able to raise a maintenance request.

Precondition: the details entered by the user is true.

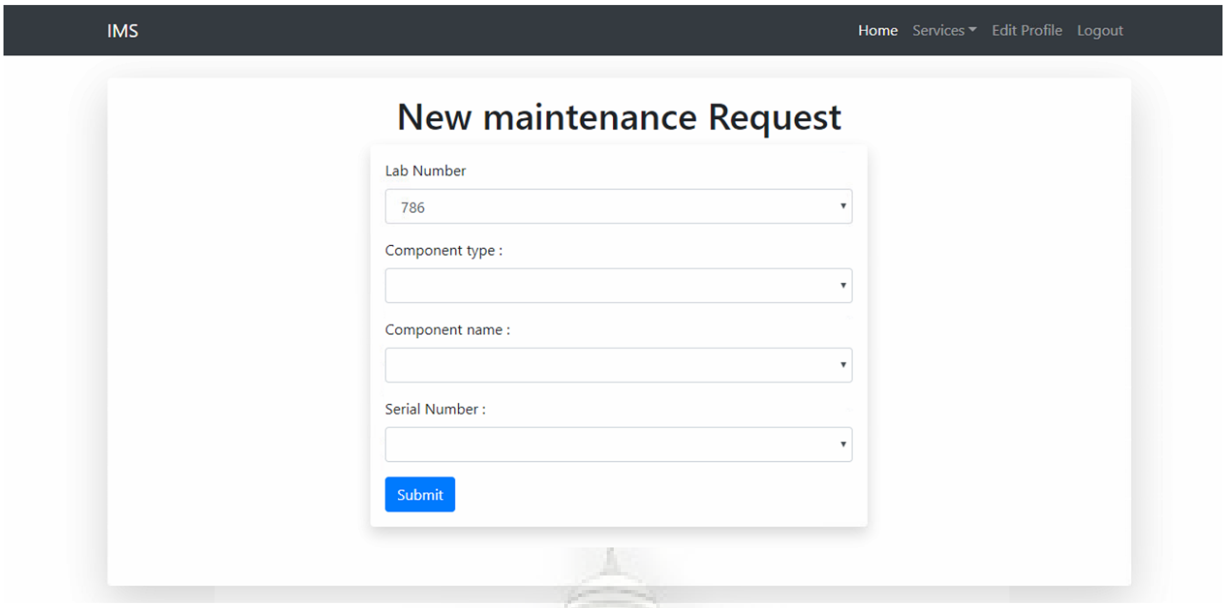
Assumption: The component which is up for maintenance should be indeed faulty to get it diagnosed by the maintenance engineer.

Test Steps:

1. The requester must enter the lab number.
2. Then the requester must enter the component type which is up for maintenance.
3. Then the requester must enter the component name.
4. Then the serial number of the component which is to be maintained.
5. Then click on the Submit button to submit the request in the system.

Expected Result: The request is updated on the maintenance table.

Actual Result: The request gets updated in the system and the administrator can view the request and assign an engineer for the job.



New maintenance Request

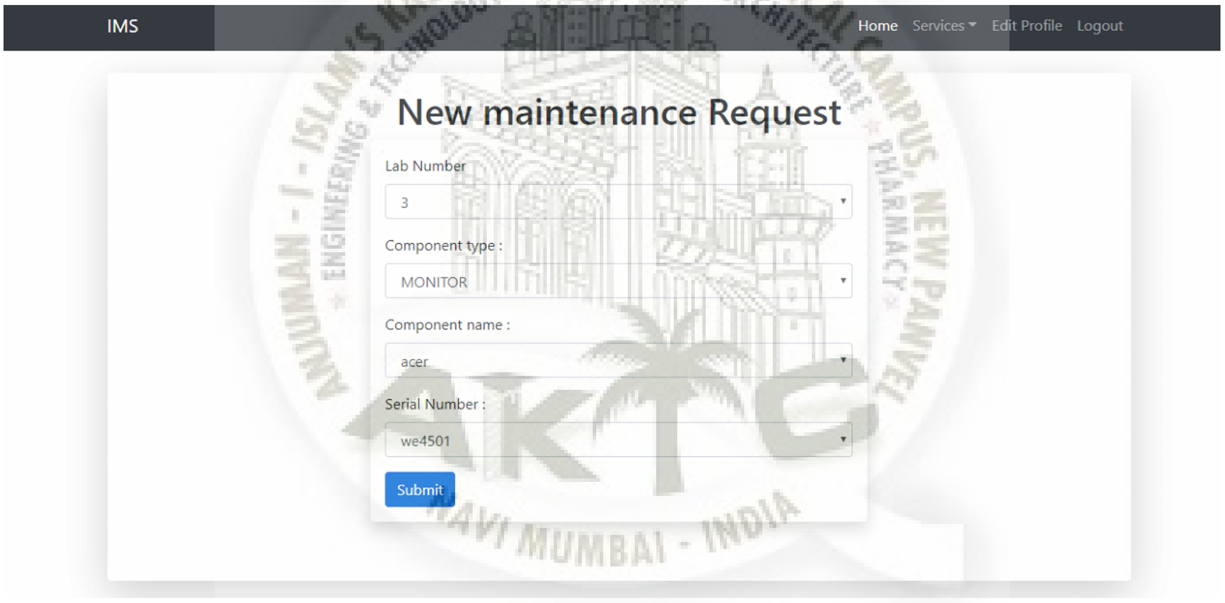
Lab Number
786

Component type :

Component name :

Serial Number :

Figure 7.5: Test Case for Maintenance Request



New maintenance Request

Lab Number
3

Component type :
MONITOR

Component name :
acer

Serial Number :
we4501

Figure 7.6: Test Case for Maintenance Request after Entering Data

7.5 Admin Issue Component Test Case

Title: Issue a component – Admin should issue from its panel.
Description: The Administrator should be able to a component to the requester.

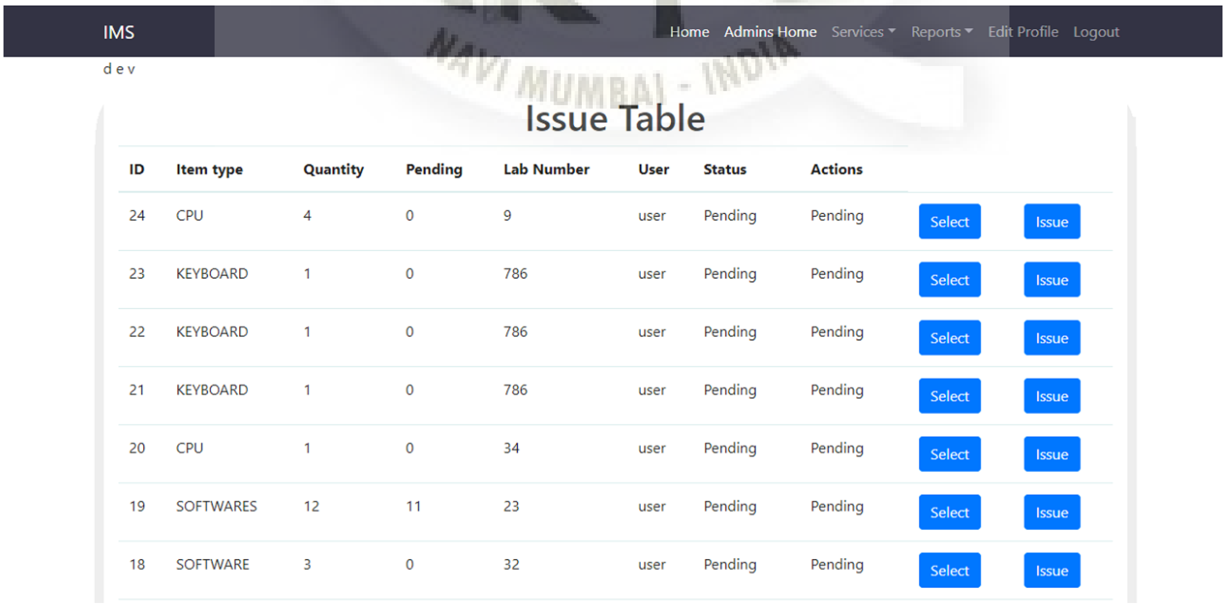
Precondition: There must be a valid issue request from the registered user of the system.

Assumption: There is inventory available for the requested component.

Test Steps:

1. The requester must enter an issue request to the system.
2. Then the administrator will view the issue request in its portal.
3. Then the administrator will view the available inventory.
4. Then the administrator will issue the component to the requester and the system gets updated.

Expected Result: The component gets issued to the requester.
Actual Result: The request gets updated when administrator issues the component and then the inventory gets updated.



ID	Item type	Quantity	Pending	Lab Number	User	Status	Actions		
24	CPU	4	0	9	user	Pending	Pending	Select	Issue
23	KEYBOARD	1	0	786	user	Pending	Pending	Select	Issue
22	KEYBOARD	1	0	786	user	Pending	Pending	Select	Issue
21	KEYBOARD	1	0	786	user	Pending	Pending	Select	Issue
20	CPU	1	0	34	user	Pending	Pending	Select	Issue
19	SOFTWARES	12	11	23	user	Pending	Pending	Select	Issue
18	SOFTWARE	3	0	32	user	Pending	Pending	Select	Issue

Figure 7.7: Test Case for issuing a component

7.6 Admin Update Maintenance Request Test Case

Title: Update Maintenance Request – Maintenance Engineer should update the maintenance request.
Description: The Maintenance Engineer should be able to update the system after performing maintenance fault diagnosis.

Precondition: There must be a valid maintenance request for a faulty component which was previously issued to the registered user of the system.

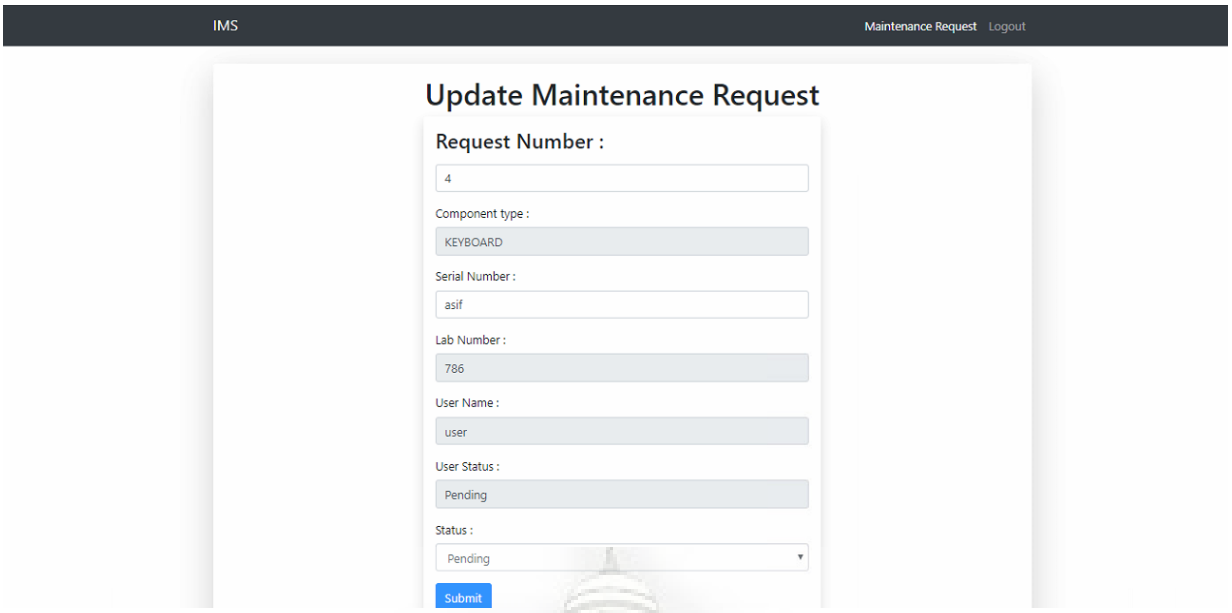
Assumption: There is a maintenance request in the system and the administrator has assigned an Engineer to do the job.

Test Steps:

1. The requester must enter a maintenance request to the system.
2. Then the administrator will view the maintenance request in its portal.
3. Then the administrator will assign the engineer to do the fault diagnosis.
4. Then the maintenance engineer will do the fault diagnosis.
5. Then the maintenance engineer will update the maintenance request to the system.

Expected Result: The maintenance work is carried out and system gets updated.

Actual Result: The administrator assigns the maintenance engineer and the administrator carries out fault maintenance and then the request is updated by the engineer for the request.



Update Maintenance Request

Request Number : 4

Component type : KEYBOARD

Serial Number : asif

Lab Number : 786

User Name : user

User Status : Pending

Status : Pending

Figure 7.8: Test Case for updating an maintenance request

7.7 Software Quality Attributes

7.7.1 Static Quality Attributes

Static Quality Attributes reflects the system's structure and organization. Static Quality Attributes are directly related to the architecture and design of the system and the way in which the system is implemented. To avoid coupling and cohesion, we need to ensure that the system is well maintained and if we need to add new modules in near future, it must be taken care of by good users of the system. Also we need to ensure that the coding style is by the international standards and naming of the files and documents is given professionally.

7.7.2 Dynamic Quality Attributes

Dynamic Quality Attributes reflects the behaviour of the system during its execution. The system should be designed in such a way that the system is always available to the users and it is robust. It should be scalable and should have the ability of detecting and correcting human errors while entering any unwanted or unuseful data. To ensure robustness of our system, we need to make sure it occupies less memory in the servers to make sure the execution time is less.

Chapter 8

Screenshots of Project

8.1 Administrator Section:

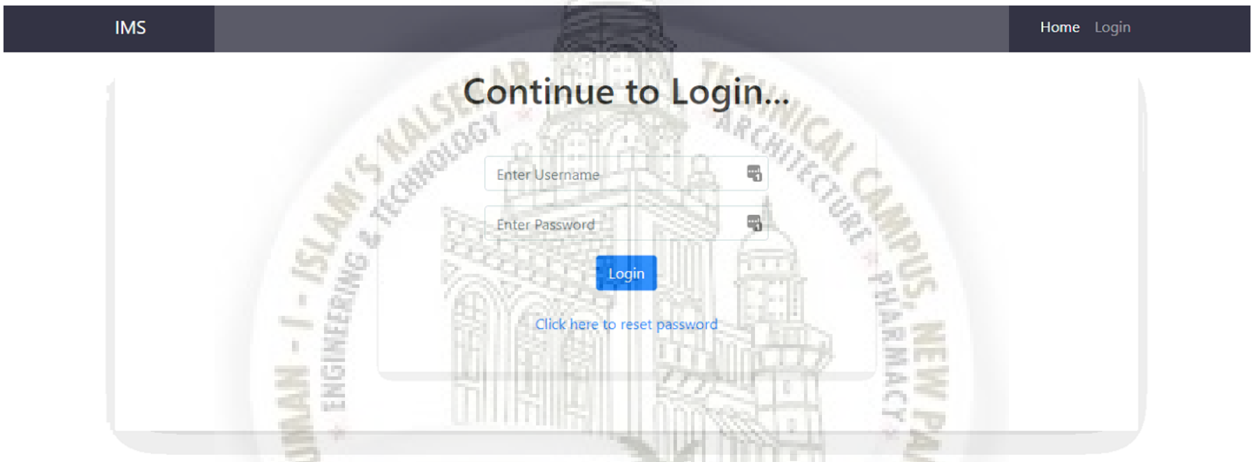


Figure 8.1: Administrator Login

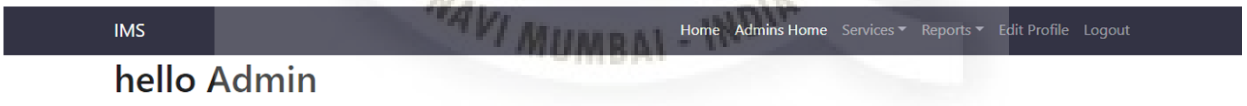
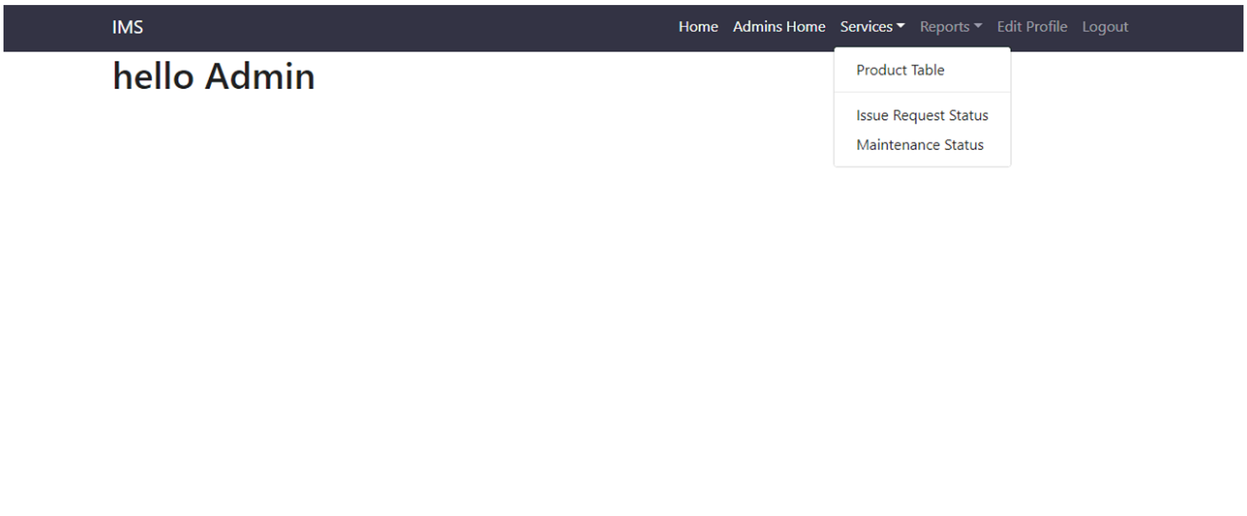
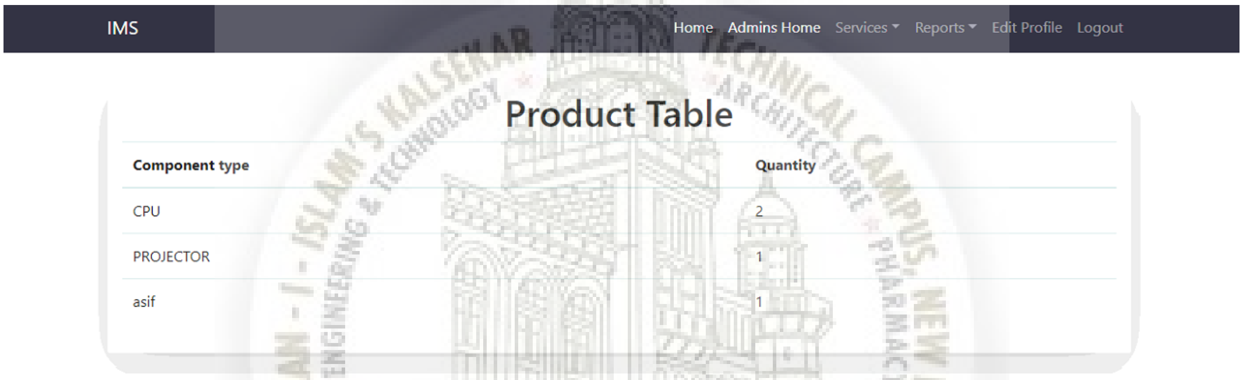


Figure 8.2: Administrator Login Successful



hello Admin

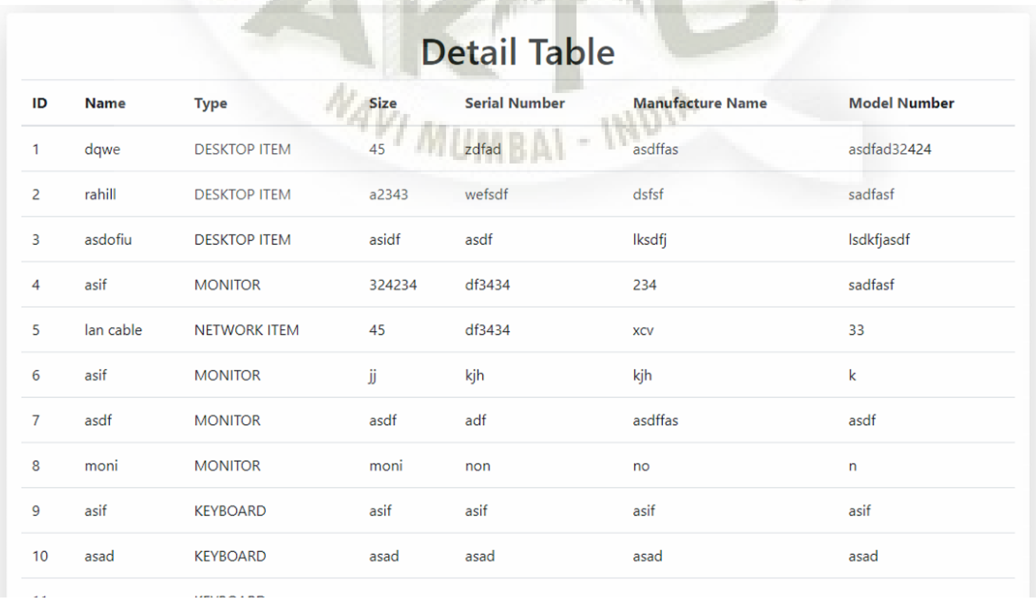
Figure 8.3: Administrator Services



Product Table

Component type	Quantity
CPU	2
PROJECTOR	1
asif	1

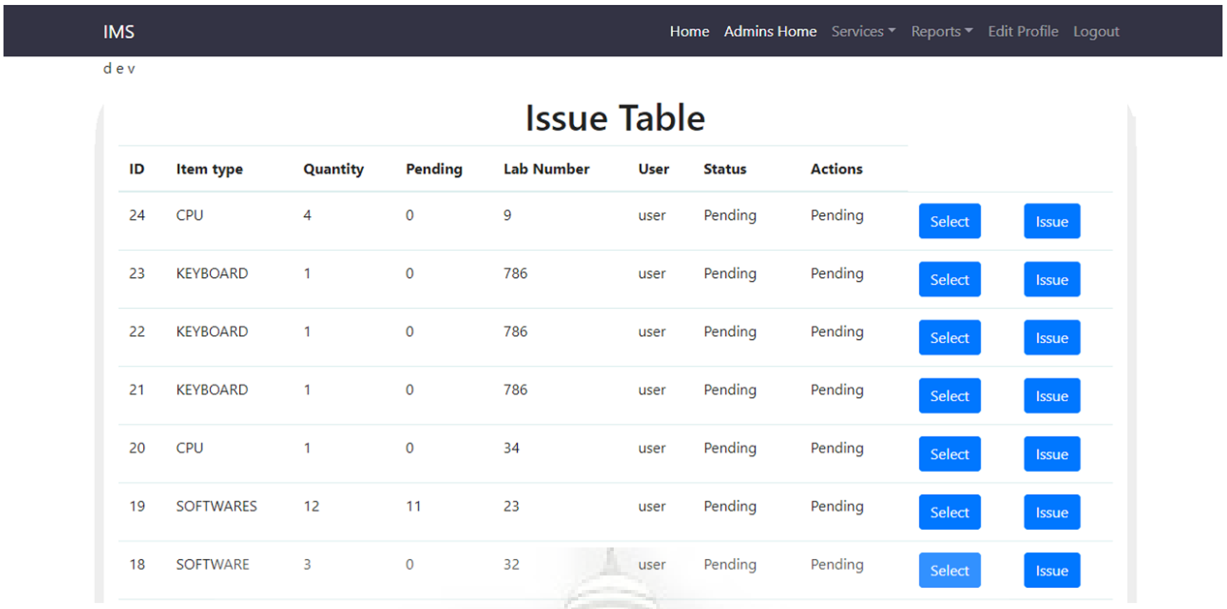
Figure 8.4: Administrator View Inventory



Detail Table

ID	Name	Type	Size	Serial Number	Manufacture Name	Model Number
1	dqwe	DESKTOP ITEM	45	zdfad	asdffas	asdfad32424
2	rahill	DESKTOP ITEM	a2343	wefsdf	dsfsf	sadfasf
3	asdofiu	DESKTOP ITEM	asidf	asdf	lksdfj	lsdkfjasdf
4	asif	MONITOR	324234	df3434	234	sadfasf
5	lan cable	NETWORK ITEM	45	df3434	xcv	33
6	asif	MONITOR	jj	kjh	kjh	k
7	asdf	MONITOR	asdf	adf	asdffas	asdf
8	moni	MONITOR	moni	non	no	n
9	asif	KEYBOARD	asif	asif	asif	asif
10	asad	KEYBOARD	asad	asad	asad	asad

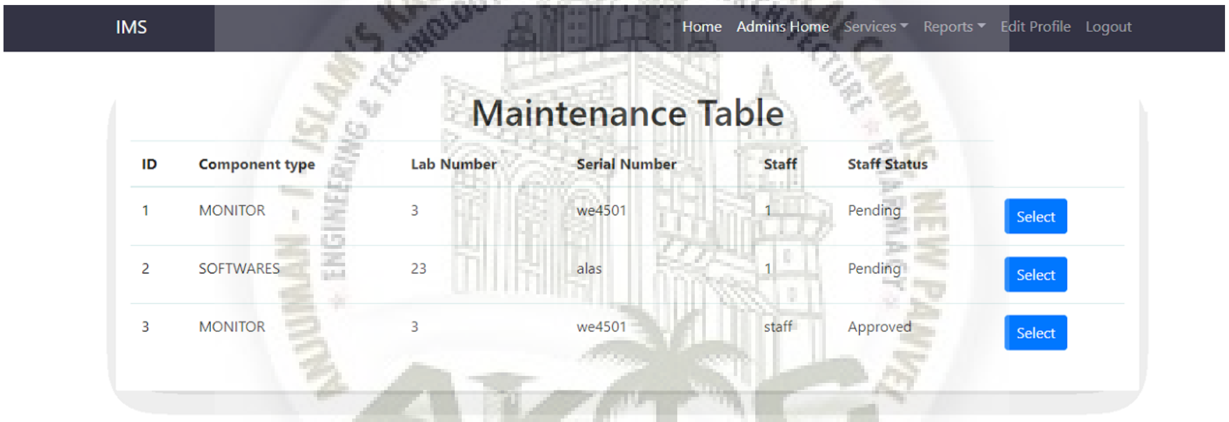
Figure 8.5: Administrator View Detail Inventory



Issue Table

ID	Item type	Quantity	Pending	Lab Number	User	Status	Actions		
24	CPU	4	0	9	user	Pending	Pending	Select	Issue
23	KEYBOARD	1	0	786	user	Pending	Pending	Select	Issue
22	KEYBOARD	1	0	786	user	Pending	Pending	Select	Issue
21	KEYBOARD	1	0	786	user	Pending	Pending	Select	Issue
20	CPU	1	0	34	user	Pending	Pending	Select	Issue
19	SOFTWARES	12	11	23	user	Pending	Pending	Select	Issue
18	SOFTWARE	3	0	32	user	Pending	Pending	Select	Issue

Figure 8.6: Administrator Issue Item



Maintenance Table

ID	Component type	Lab Number	Serial Number	Staff	Staff Status	
1	MONITOR	3	we4501	1	Pending	Select
2	SOFTWARES	23	alas	1	Pending	Select
3	MONITOR	3	we4501	staff	Approved	Select

Figure 8.7: Administrator Assign Engineer for Maintenance

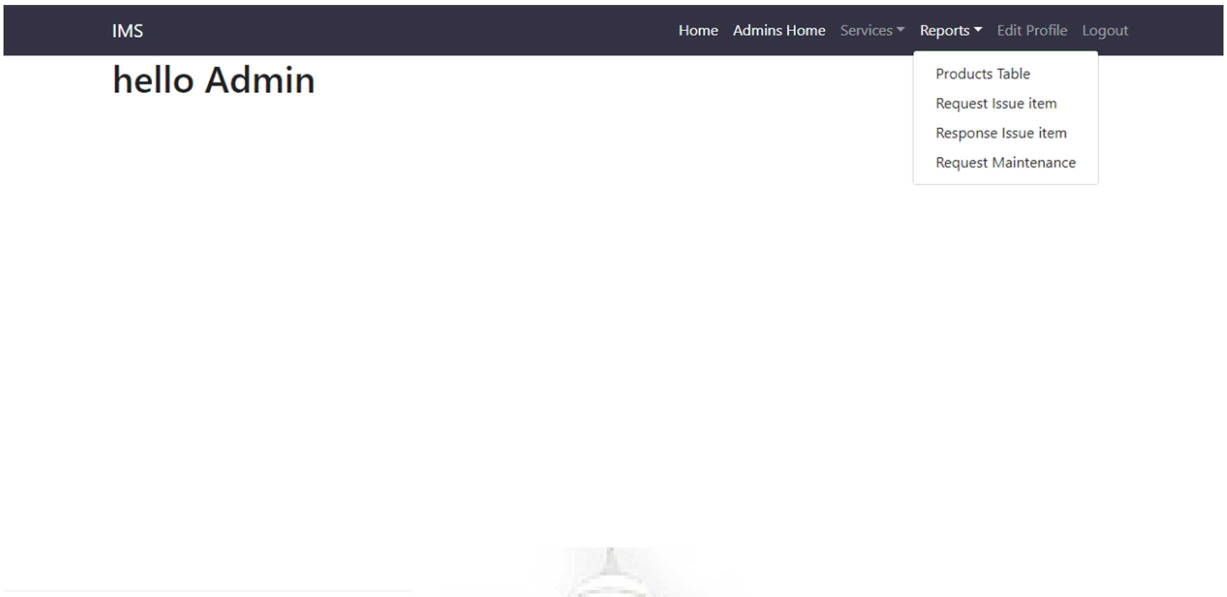
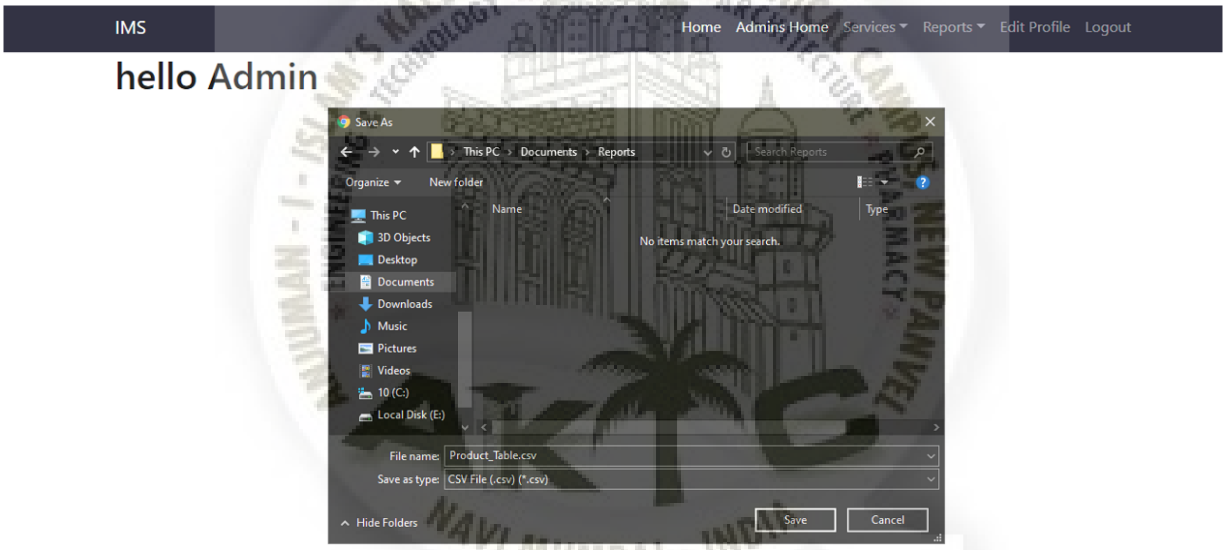


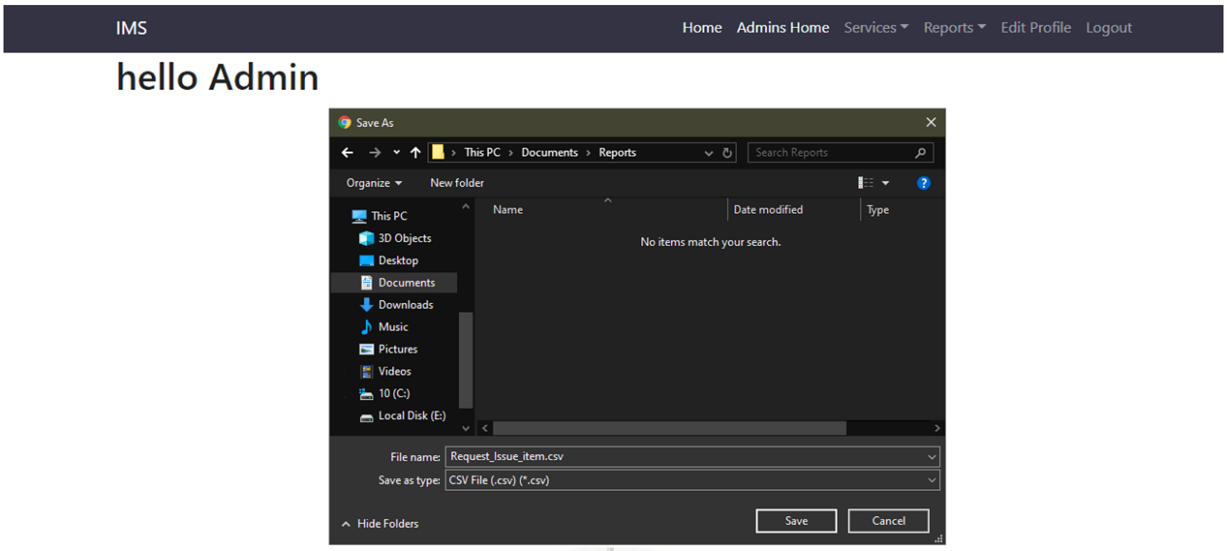
Figure 8.8: Administrator Generate Reports



No items match your search.

File name: Product_Table.csv
Save as type: CSV File (.csv) (*.csv)

Figure 8.9: Generate Product Table Report

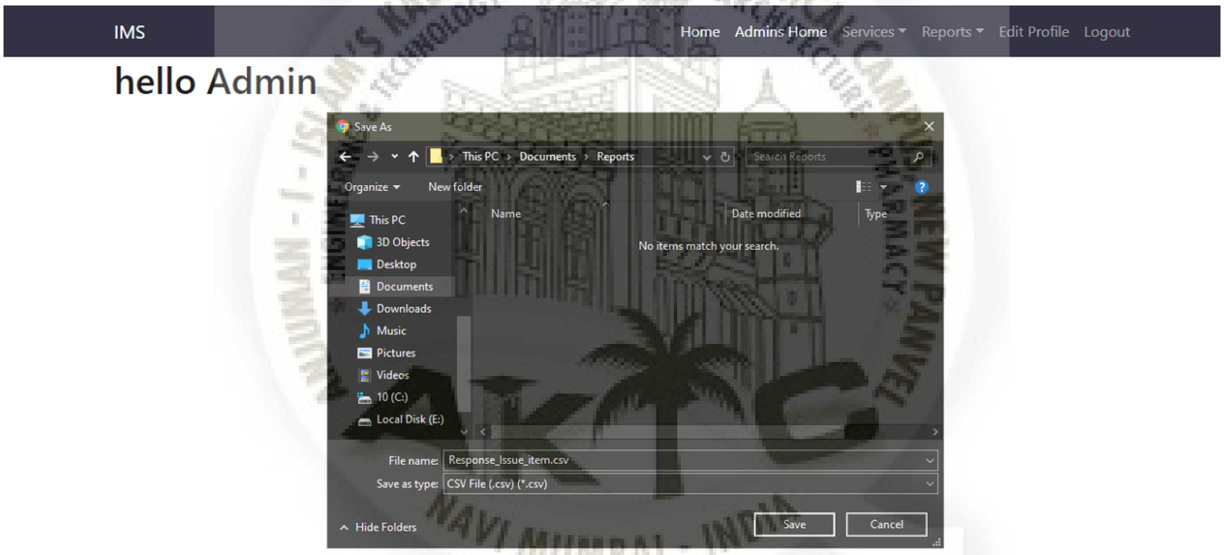


No items match your search.

File name: Request_Issue_item.csv

Save as type: CSV File (.csv) (*.csv)

Figure 8.10: Generate Issue Requests Report

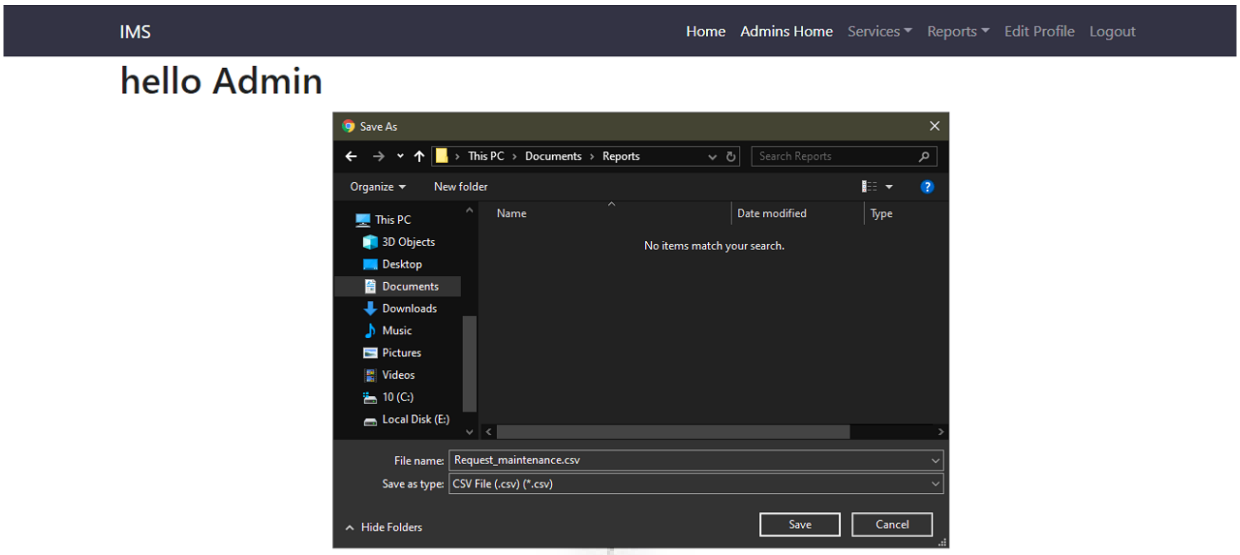


No items match your search.

File name: Response_Issue_item.csv

Save as type: CSV File (.csv) (*.csv)

Figure 8.11: Generate Issue Response Report

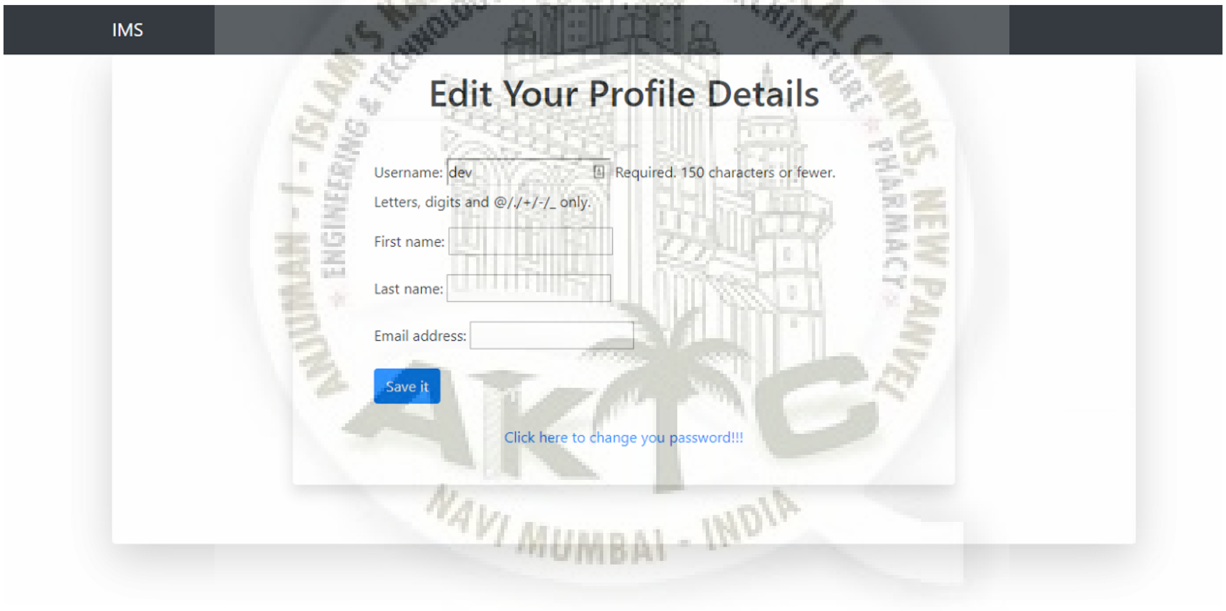


No items match your search.

File name: Request_maintenance.csv

Save as type: CSV File (.csv) (*.csv)

Figure 8.12: Generate Maintenance Request Report



Edit Your Profile Details

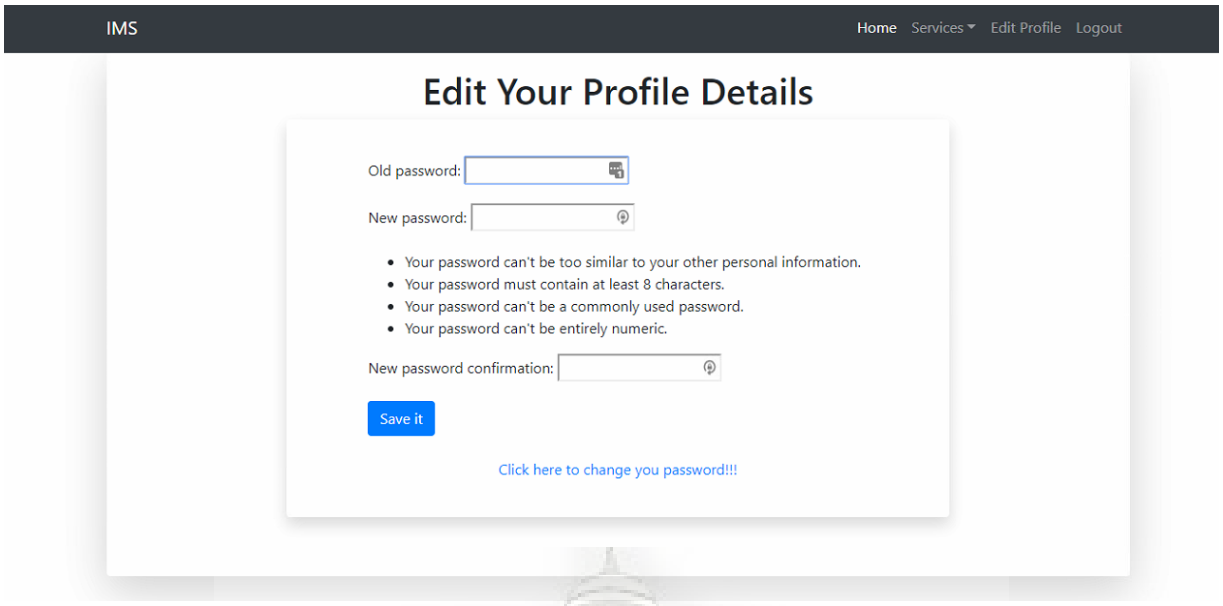
Username: dev Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

First name:

Last name:

Email address:

Figure 8.13: Edit Profile Administrator


IMS
Home Services Edit Profile Logout

Edit Your Profile Details

Old password:

New password:

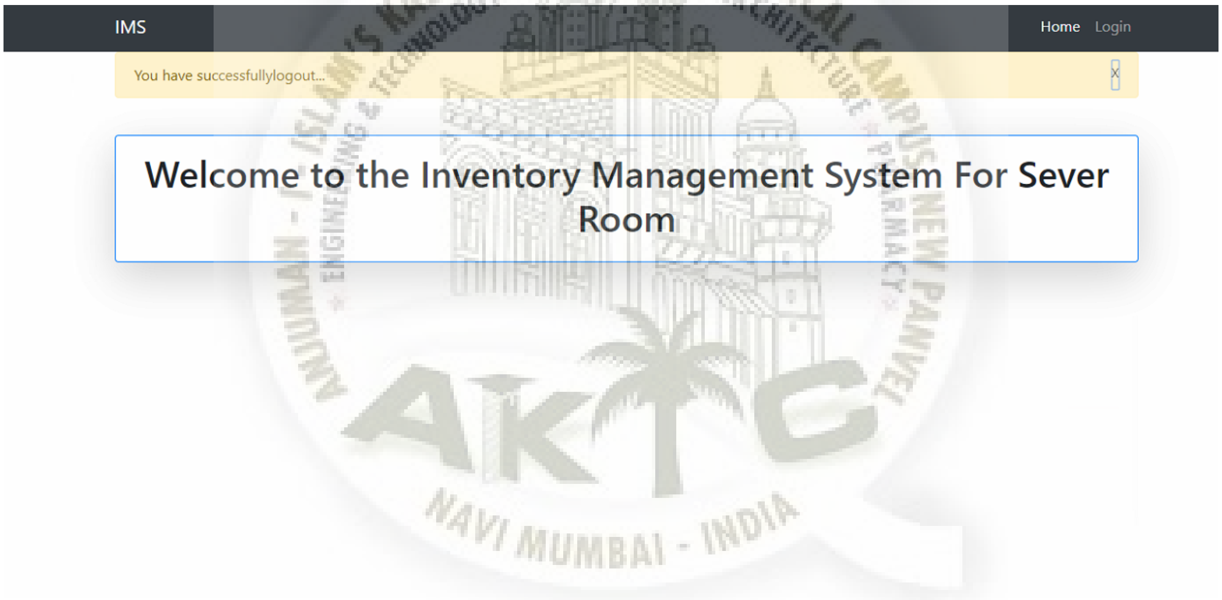
- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

New password confirmation:

Save it

Click here to change you password!!!

Figure 8.14: Administrator Change Password


IMS
Home Login

You have successfullylogout...

Welcome to the Inventory Management System For Sever Room

Figure 8.15: Administrator Logout

8.2 Requester's Section

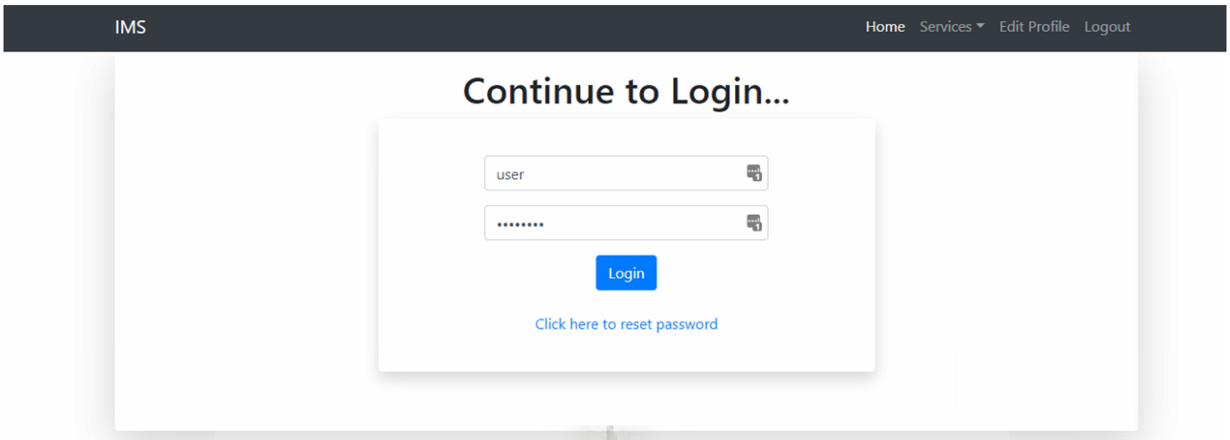
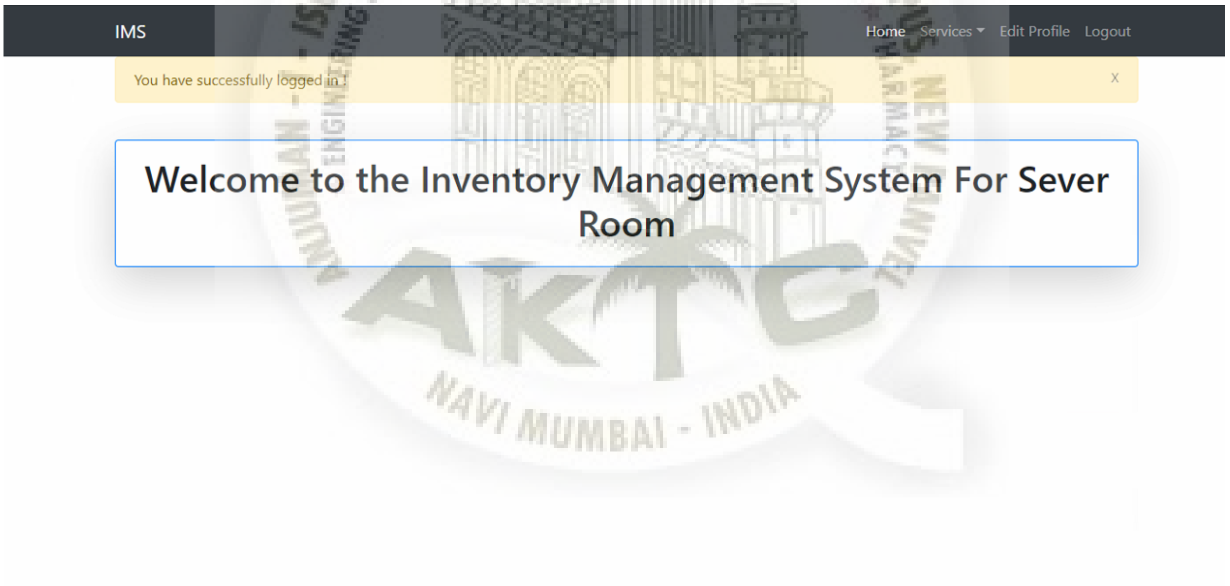


Figure 8.16: Requester Login



You have successfully logged in !

Welcome to the Inventory Management System For Sever Room

Figure 8.17: User Login Successful

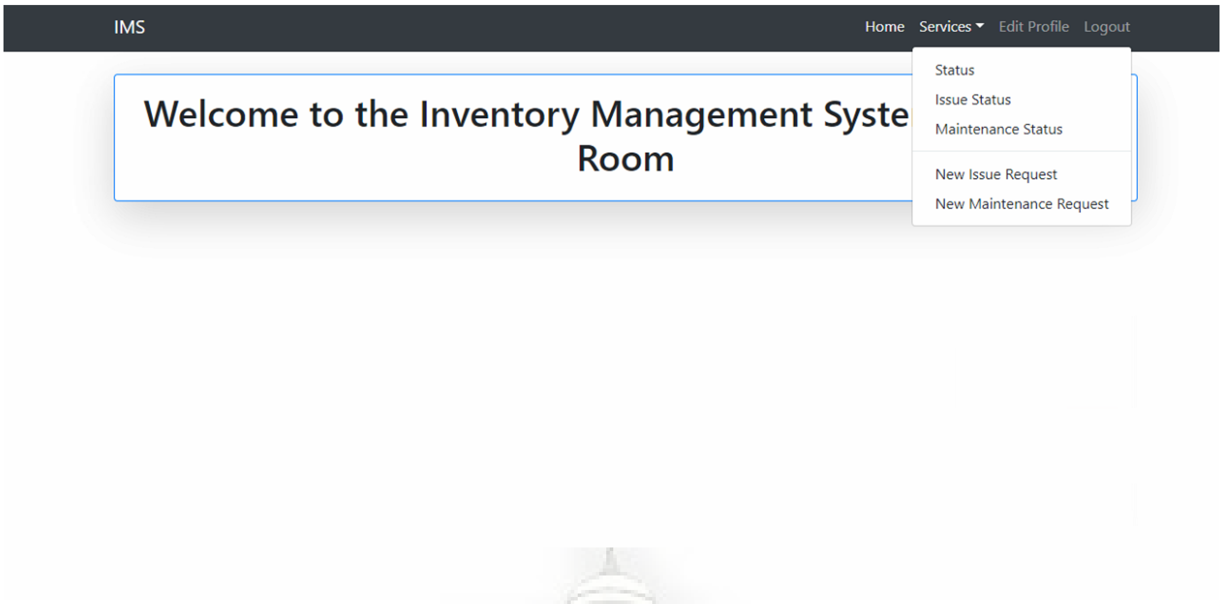


Figure 8.18: User Services

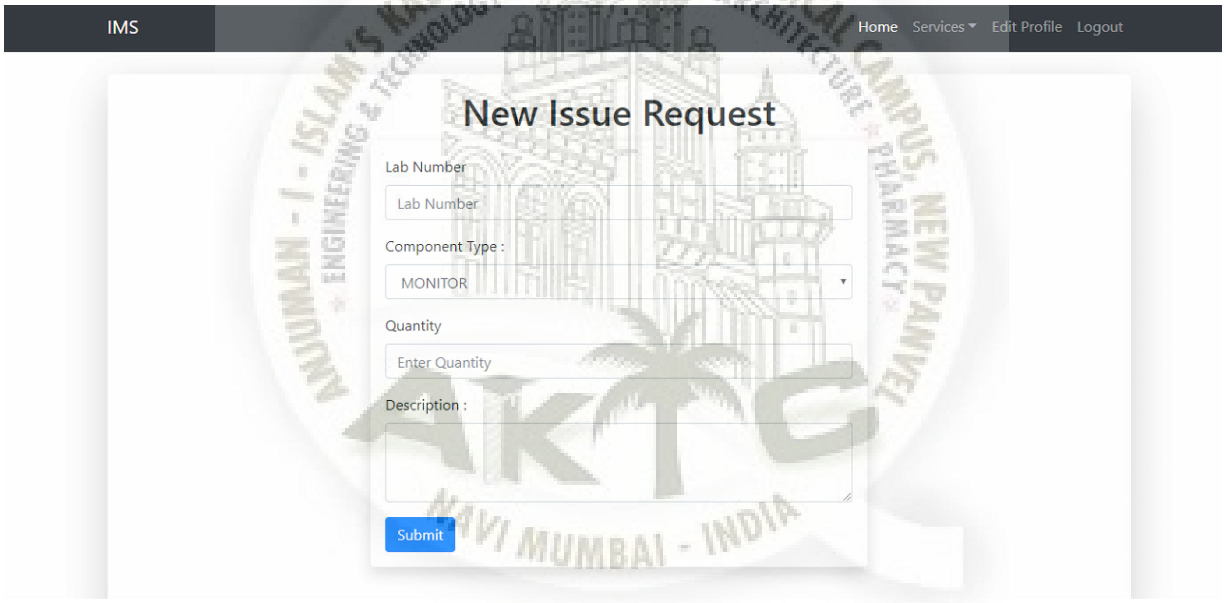
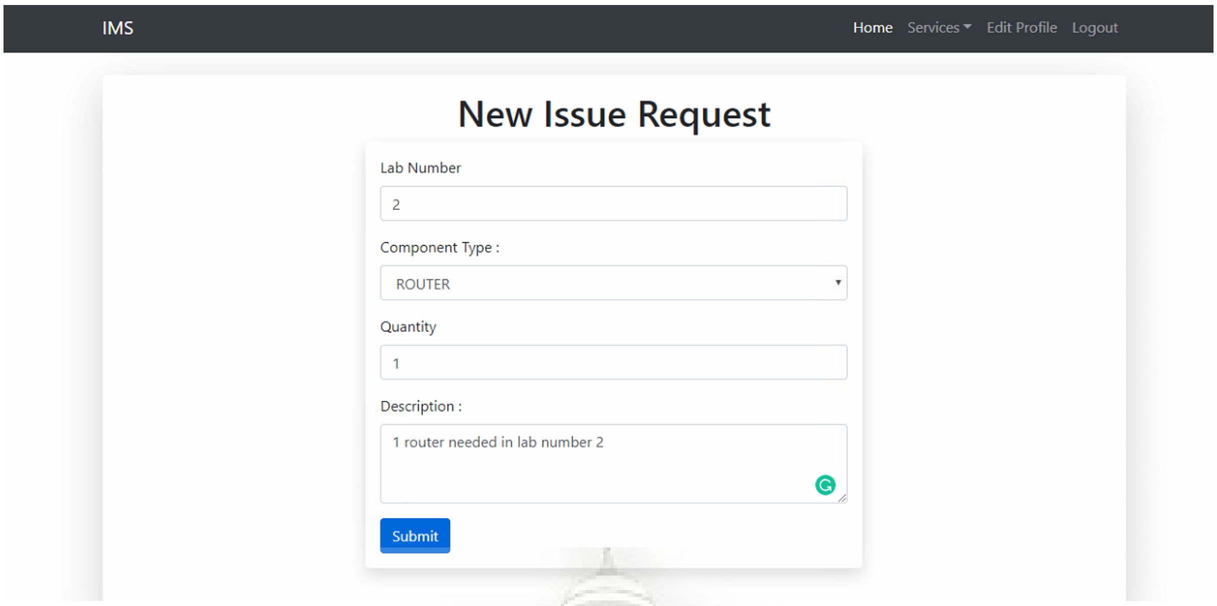


Figure 8.19: Empty Issue Request Form



New Issue Request

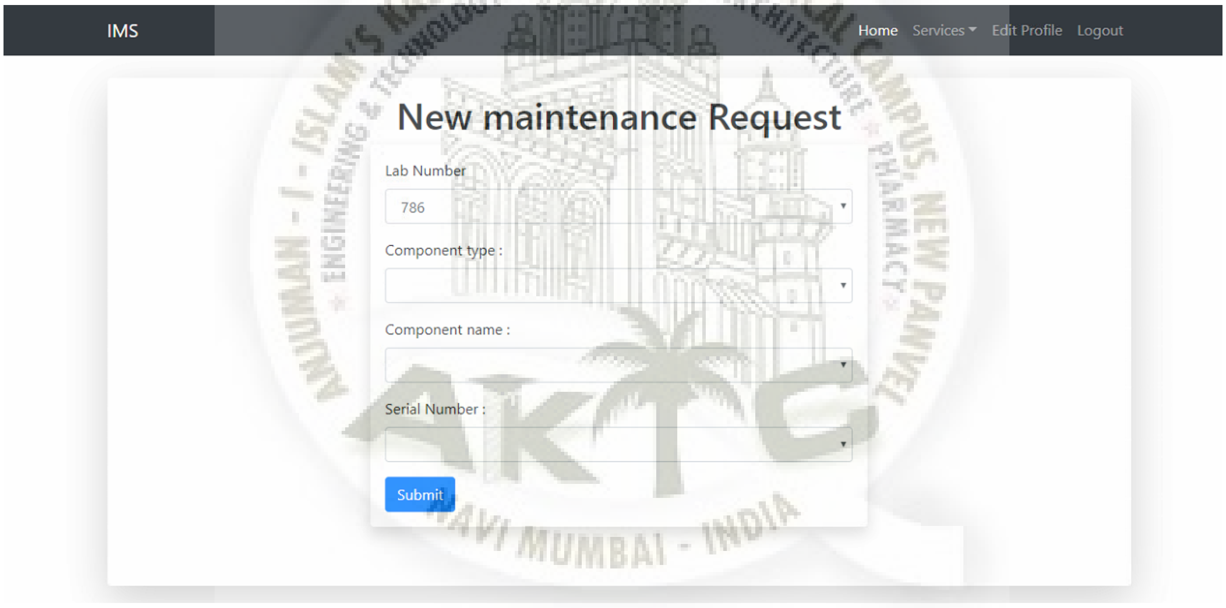
Lab Number
2

Component Type :
ROUTER

Quantity
1

Description :
1 router needed in lab number 2

Figure 8.20: Issue Request Form



New maintenance Request

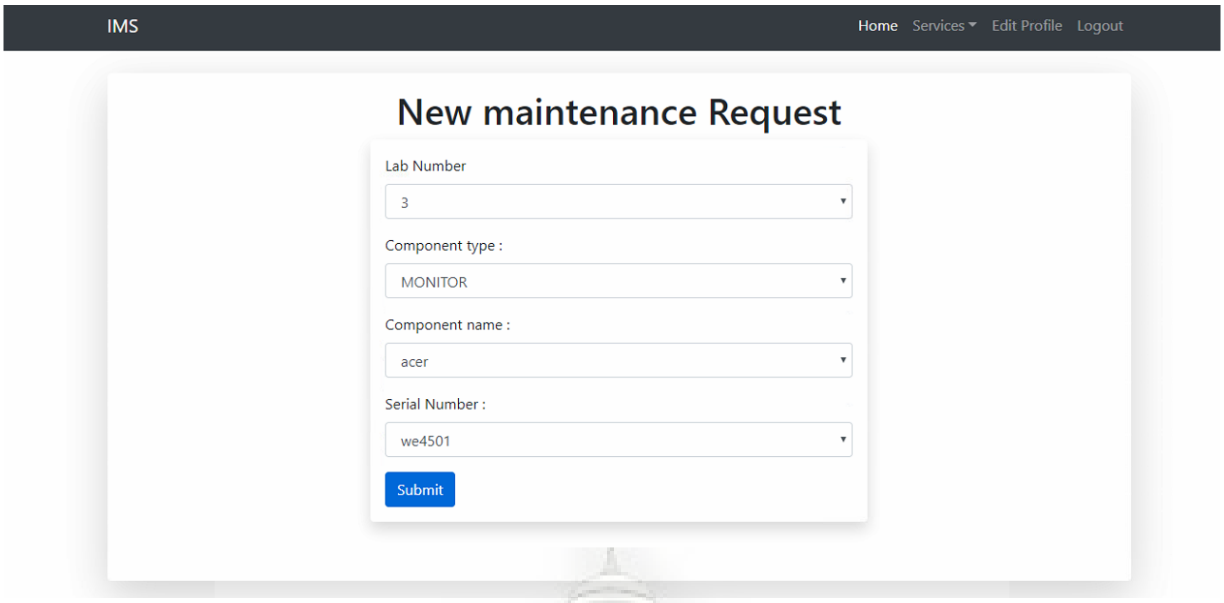
Lab Number
786

Component type :

Component name :

Serial Number :

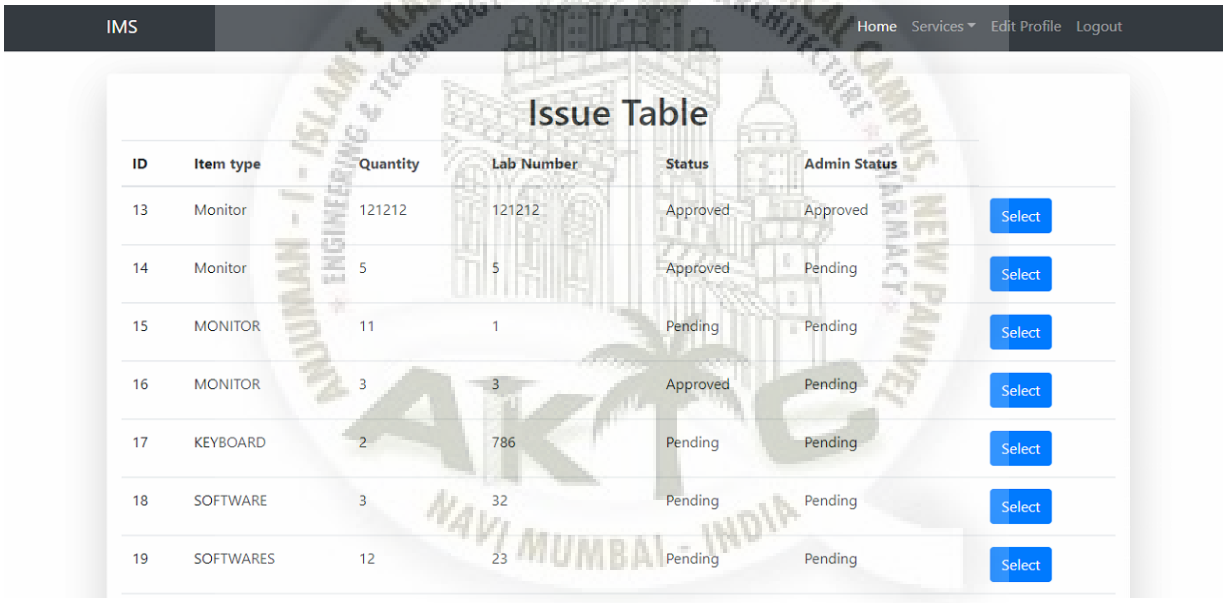
Figure 8.21: Empty Maintenance Request Form



New maintenance Request

Lab Number	3
Component type :	MONITOR
Component name :	acer
Serial Number :	we4501

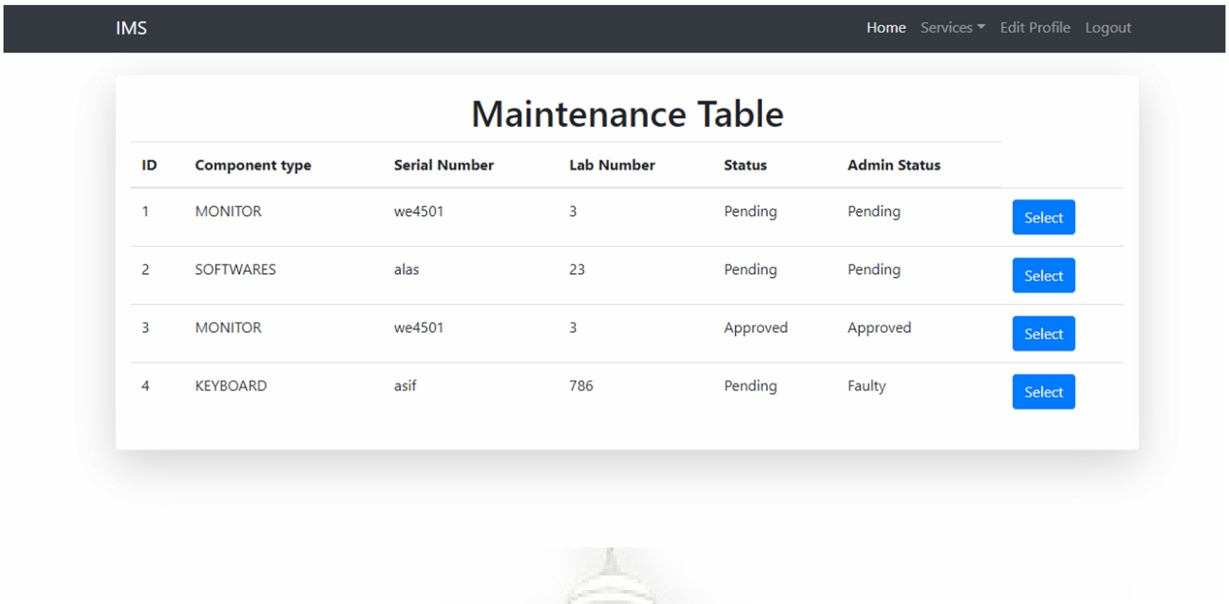
Figure 8.22: Maintenance Request Form



Issue Table

ID	Item type	Quantity	Lab Number	Status	Admin Status	
13	Monitor	121212	121212	Approved	Approved	Select
14	Monitor	5	5	Approved	Pending	Select
15	MONITOR	11	1	Pending	Pending	Select
16	MONITOR	3	3	Approved	Pending	Select
17	KEYBOARD	2	786	Pending	Pending	Select
18	SOFTWARE	3	32	Pending	Pending	Select
19	SOFTWARES	12	23	Pending	Pending	Select

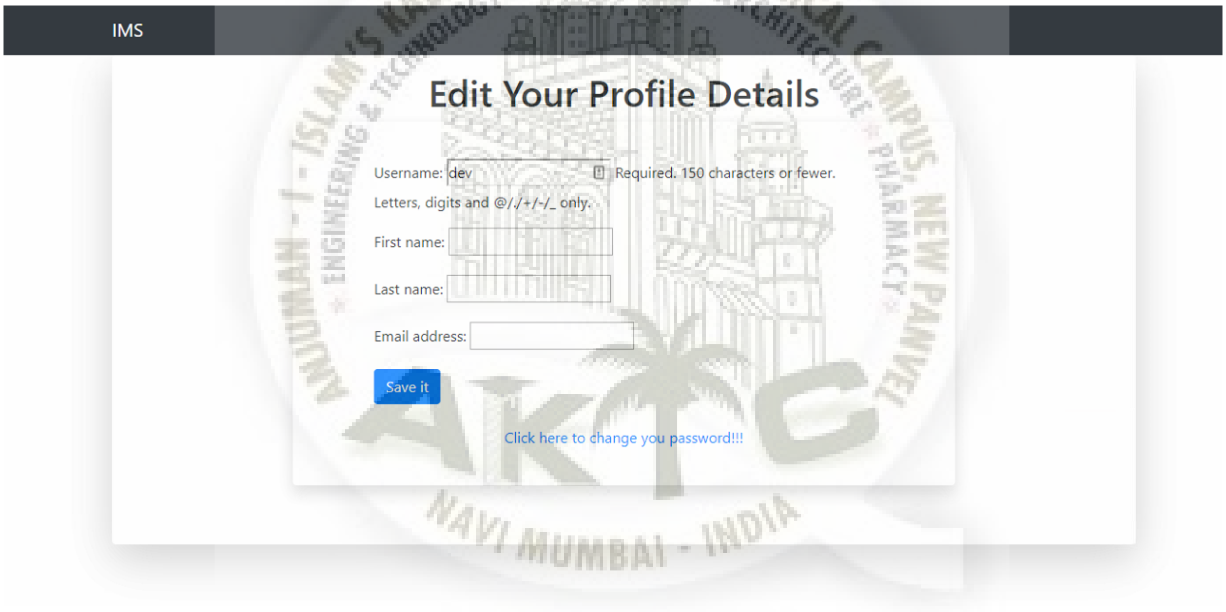
Figure 8.23: Requester View Issue Request



Maintenance Table

ID	Component type	Serial Number	Lab Number	Status	Admin Status	
1	MONITOR	we4501	3	Pending	Pending	Select
2	SOFTWARES	alas	23	Pending	Pending	Select
3	MONITOR	we4501	3	Approved	Approved	Select
4	KEYBOARD	asif	786	Pending	Faulty	Select

Figure 8.24: Requester View Maintenance Request



Edit Your Profile Details

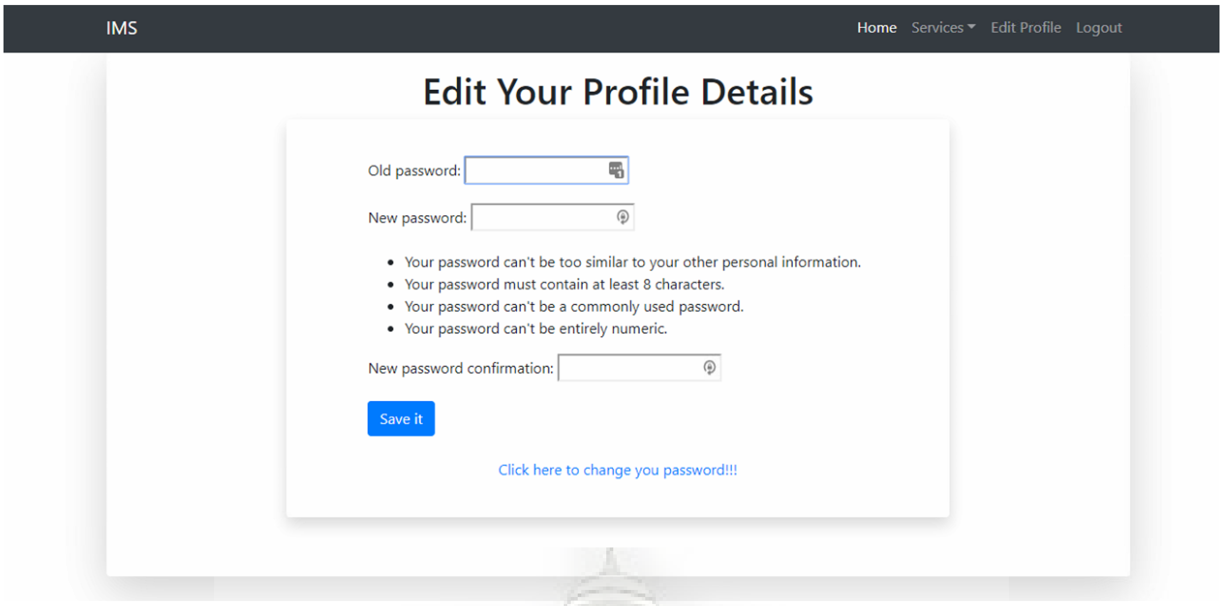
Username: dev Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

First name:

Last name:

Email address:

Figure 8.25: Requester Edit Profile



Edit Your Profile Details

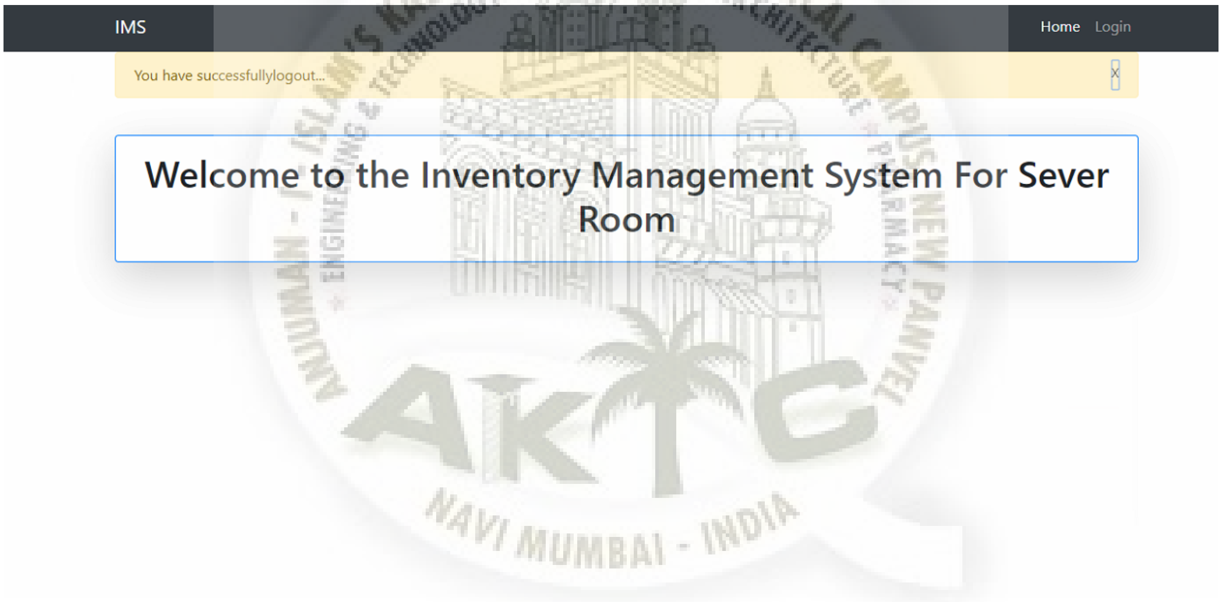
Old password:

New password:

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

New password confirmation:

Figure 8.26: Requester Change Password



You have successfullylogout...

Welcome to the Inventory Management System For Sever Room

Figure 8.27: Requester Logout

8.3 Maintenance Engineer Section

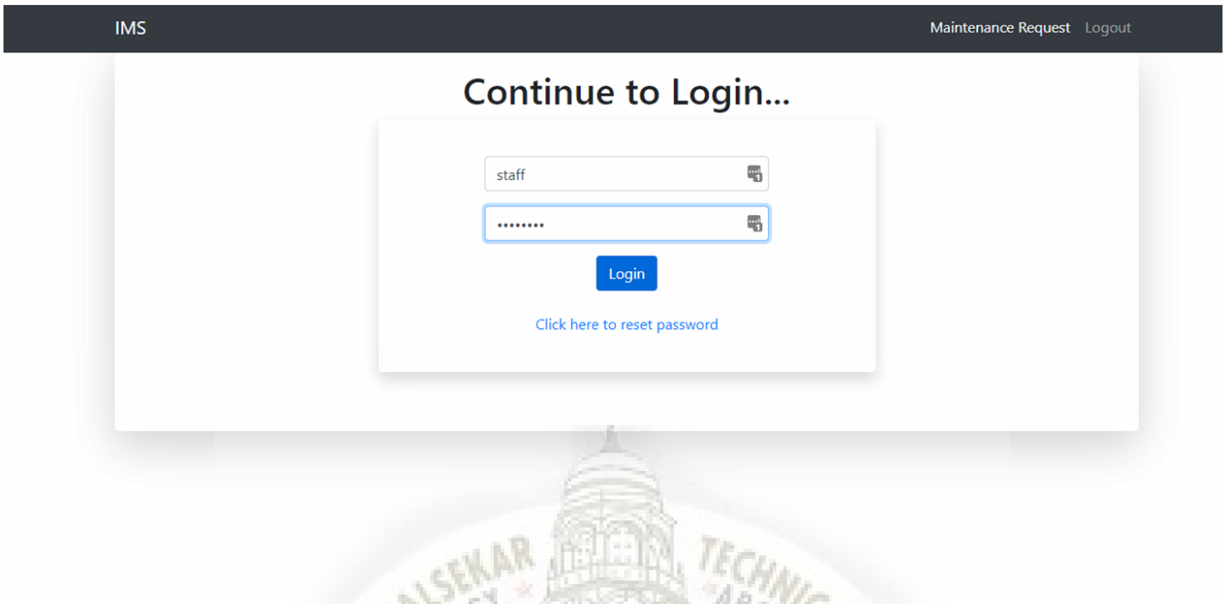
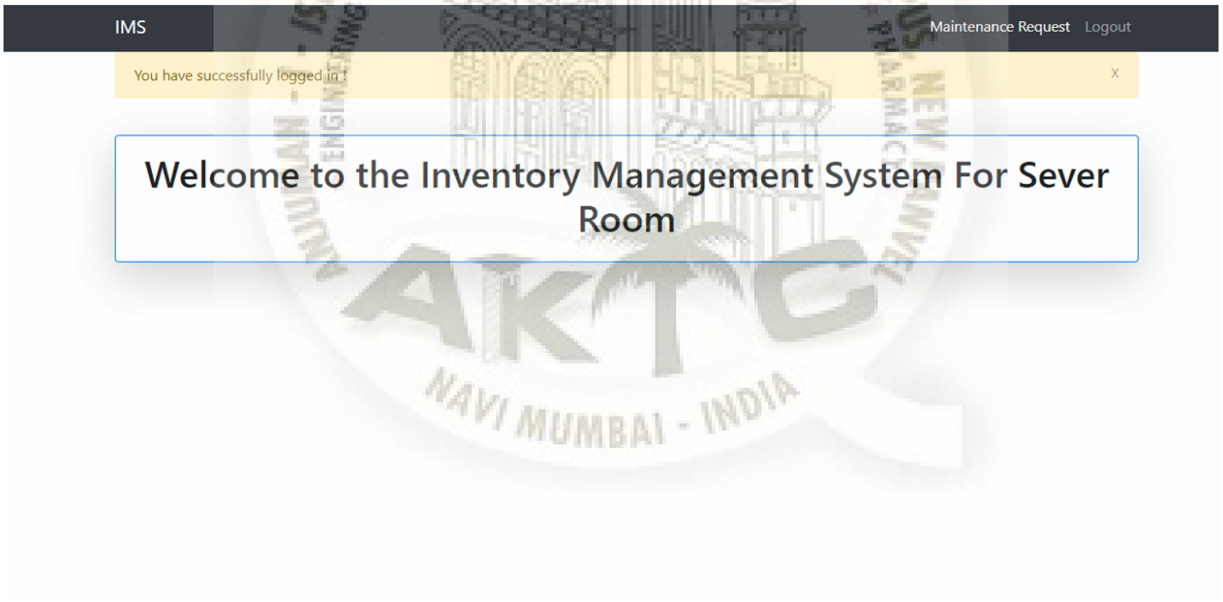


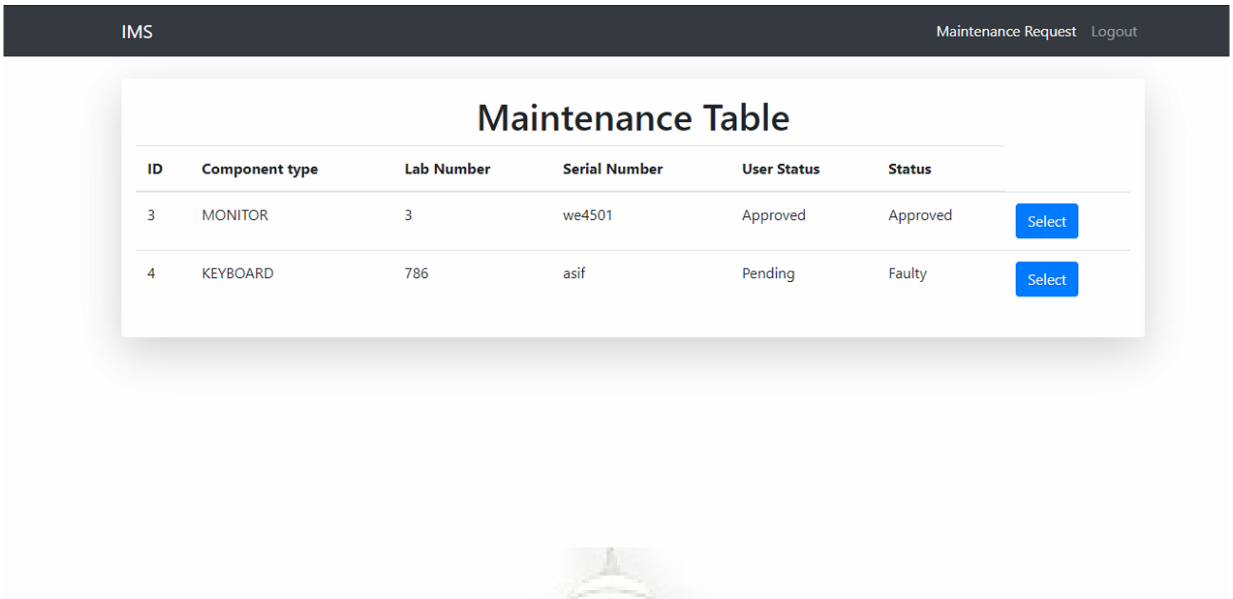
Figure 8.28: Maintenance Engineer Login Portal



You have successfully logged in !

Welcome to the Inventory Management System For Sever Room

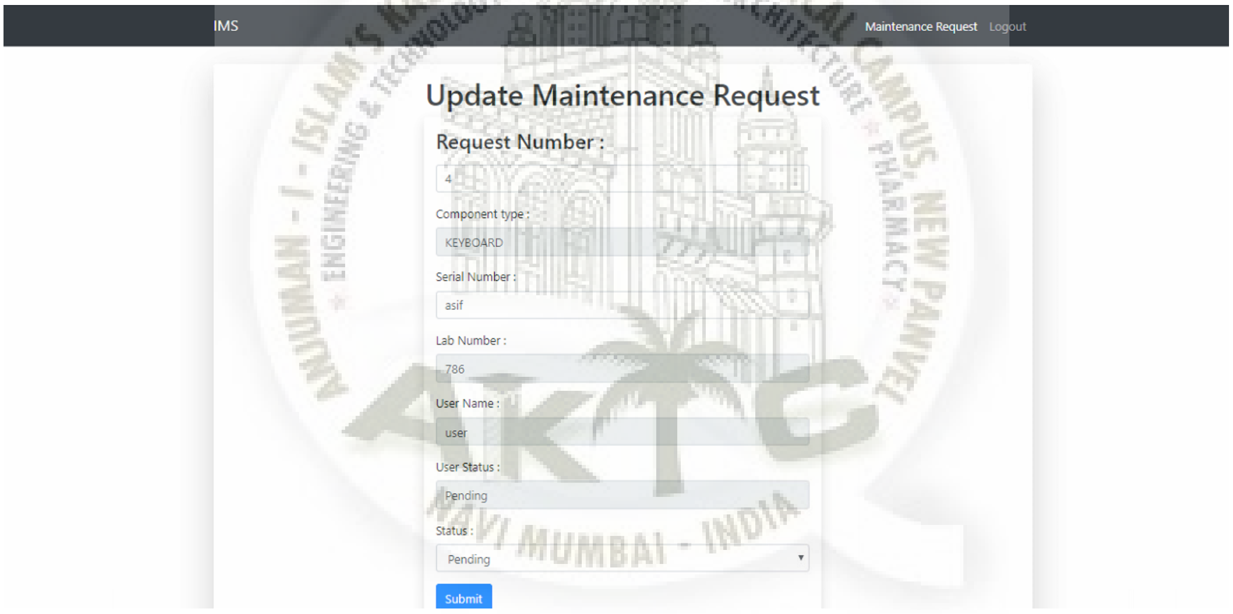
Figure 8.29: Maintenance Engineer Logged In



Maintenance Table

ID	Component type	Lab Number	Serial Number	User Status	Status	
3	MONITOR	3	we4501	Approved	Approved	Select
4	KEYBOARD	786	asif	Pending	Faulty	Select

Figure 8.30: Maintenance Engineer View Maintenance Requests



Update Maintenance Request

Request Number :	4
Component type :	KEYBOARD
Serial Number :	asif
Lab Number :	786
User Name :	user
User Status :	Pending
Status :	Pending

Figure 8.31: Maintenance Engineer Update Maintenance Requests

Chapter 9

Conclusion and Future Scope

9.1 Conclusion

This system will ease the manual maintenance of Inventory Management System by giving transparency,security and maintaining integrity of the inventory. Other tasks such as generating reports on demand ,carrying out maintenance of devices and thus maintaining data for all of these devices.

9.2 Future Scope

Following are some of the functionalities that can be integrated in our project in future we needed.

- As the technology emerges,it is possible to upgrade the system and can be adaptable to desired environment.Because it is a Web application ,any further changes can be easily adaptable.
- We can add Real Time network analyzing and troubleshooting which will help the administrator to rectify the defect in the network of AIKTC and also to troubleshoot if the defect is not a physical defect.
- We can also extend the Inventory Management System for the other departments of our institute as well so as to ease the load of manual paperwork.
- As the need arises,we can also develop Android Application for Inventory Management for Server Centres

References

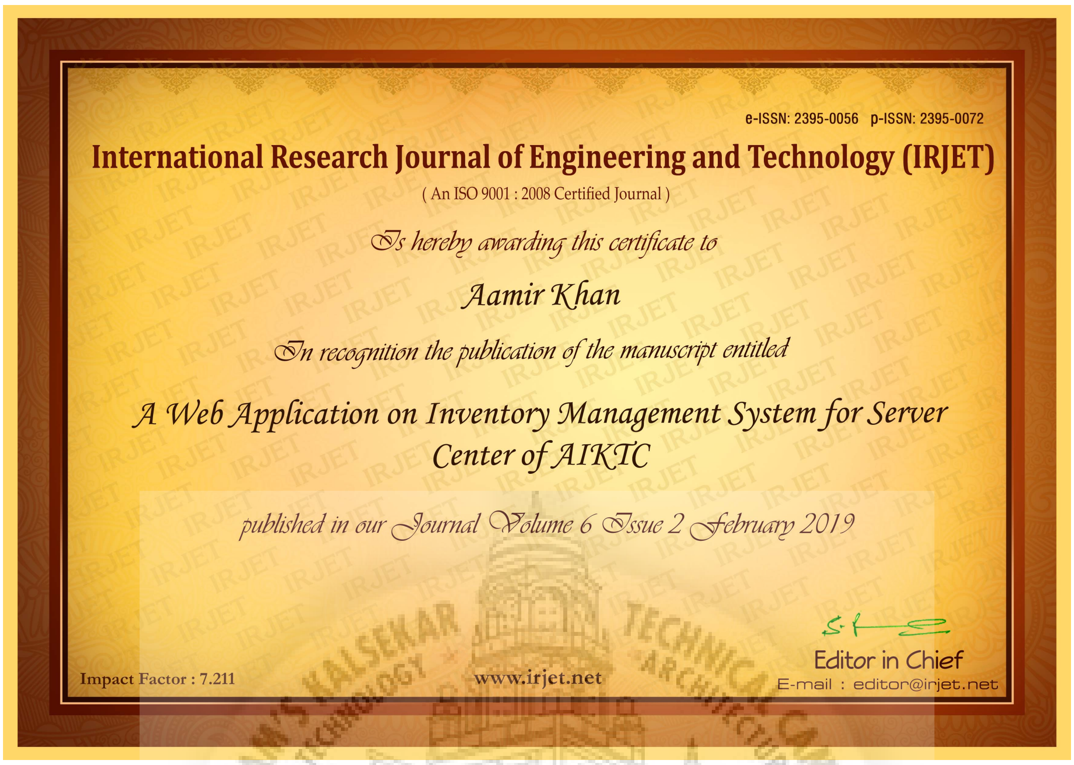
[1] *"The Design of Web-based Warehouse Management System"* ; Zizhuo Yang,Jun Wang,Qianmin Su,Bocheng Zhong,11th World Congress on Intelligent Control and Automation,vol 2(14), ISSN 978-1-4799-5825,February 2014

[2] *"Inventory Management system for water supply network"*;O.Juki,I.Hei,MIPRO 2014,26-30 May 2014,,Opatija,Croatia.

[3] *"Research and Design of the Intellegent Inventory Management System based on RFID"*,Xiaojun Jing, Peng Tang,2013 Sixth International Symposium on Computational Intelligence and Design,Volume:05 ,Issue: 04,ISSN: 978-0-7695-5079,April 2013 .

Achievements

1. Publications
   (a) *"A WEB APPLICATION ON INVENTORY MANAGEMENT SYSTEM FOR SERVER CENTER OF AIKTC"*; Aamir Khan, Aasif Ansari, MD Ghalib Nizamuddin, Heramb Pandit; International Research Journal of Engineering and Technology(IRJET), Volume 6, Issue 2, February 2019, (`https://www.irjet.net/archives/V6/i2/IRJET-V6I2436.pdf`)
2. Project Competitions
   (a) *"A WEB APPLICATION ON INVENTORY MANAGEMENT SYSTEM FOR SERVER CENTER OF AIKTC"*; Aamir Khan, Aasif Ansari, MD Ghalib Nizamuddin, Heramb Pandit; Avalon 2019(National Level Technical Paper Presentation, March 2019 (Venue: New Mumbai)



e-ISSN: 2395-0056 p-ISSN: 2395-0072

International Research Journal of Engineering and Technology (IRJET)

(An ISO 9001 : 2008 Certified Journal)

Is hereby awarding this certificate to

Aamir Khan

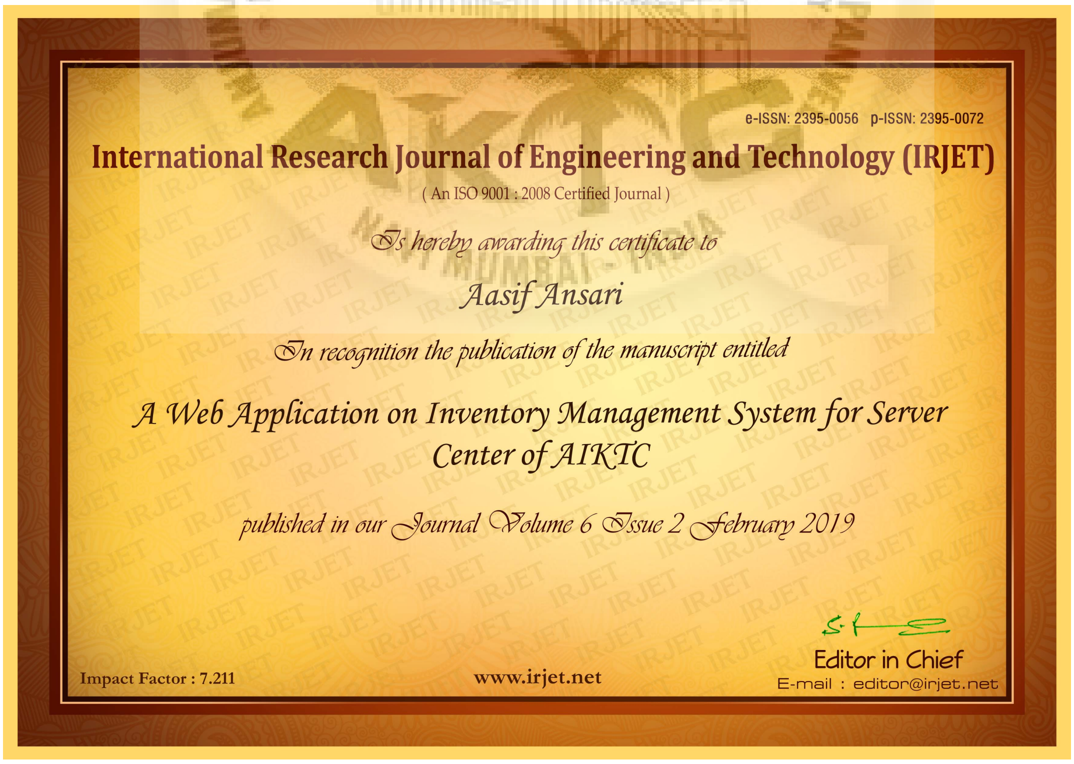
In recognition the publication of the manuscript entitled

A Web Application on Inventory Management System for Server Center of AIKTC

published in our Journal Volume 6 Issue 2 February 2019

Impact Factor : 7.211 www.irjet.net Editor in Chief E-mail : editor@irjet.net

Figure 9.1: Publication Certificate- Aamir Khan



e-ISSN: 2395-0056 p-ISSN: 2395-0072

International Research Journal of Engineering and Technology (IRJET)

(An ISO 9001 : 2008 Certified Journal)

Is hereby awarding this certificate to

Aasif Ansari

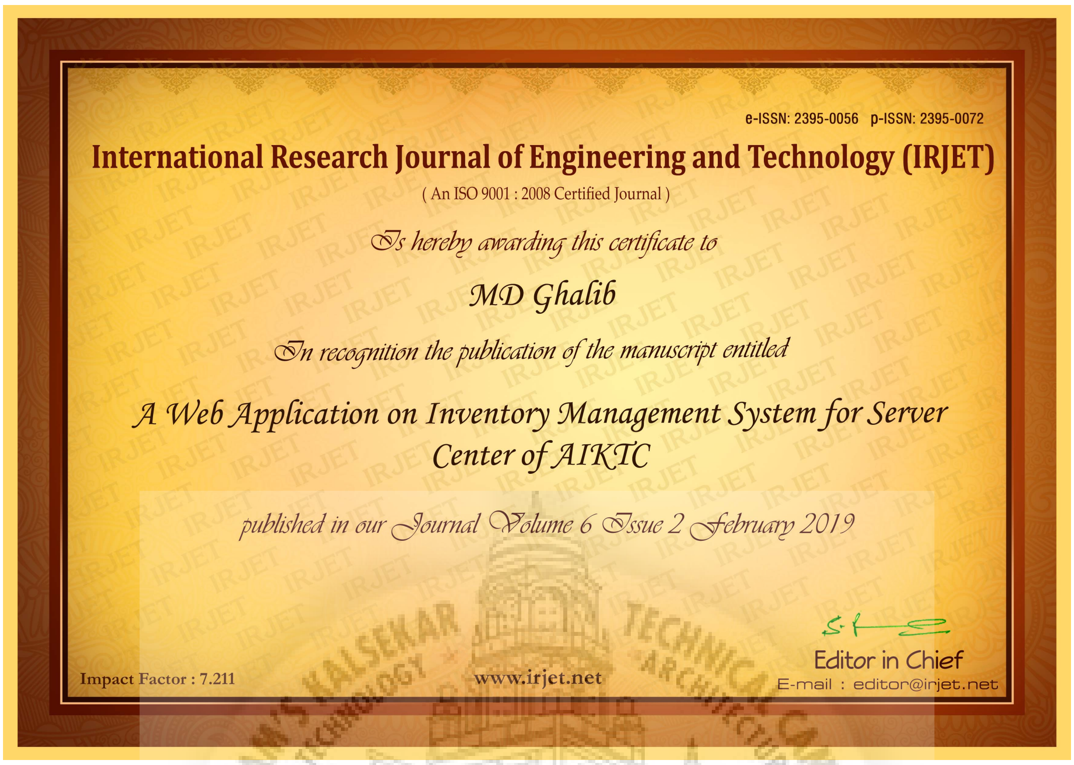
In recognition the publication of the manuscript entitled

A Web Application on Inventory Management System for Server Center of AIKTC

published in our Journal Volume 6 Issue 2 February 2019

Impact Factor : 7.211 www.irjet.net Editor in Chief E-mail : editor@irjet.net

Figure 9.2: Publication Certificate- Aasif Ansari



e-ISSN: 2395-0056 p-ISSN: 2395-0072

International Research Journal of Engineering and Technology (IRJET)

(An ISO 9001 : 2008 Certified Journal)

Is hereby awarding this certificate to

MD Ghalib

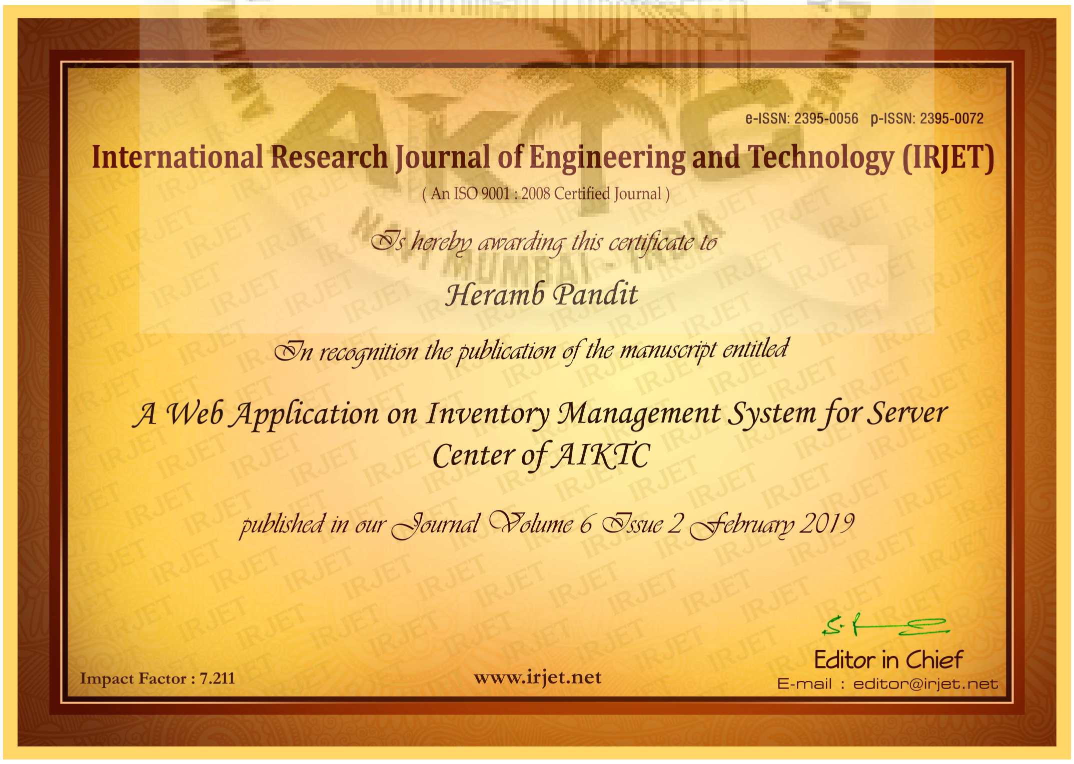
In recognition the publication of the manuscript entitled

A Web Application on Inventory Management System for Server Center of AIKTC

published in our Journal Volume 6 Issue 2 February 2019

Impact Factor : 7.211 www.irjet.net Editor in Chief E-mail : editor@irjet.net

Figure 9.3: Publication Certificate- MD Ghalib



e-ISSN: 2395-0056 p-ISSN: 2395-0072

International Research Journal of Engineering and Technology (IRJET)

(An ISO 9001 : 2008 Certified Journal)

Is hereby awarding this certificate to

Heramb Pandit

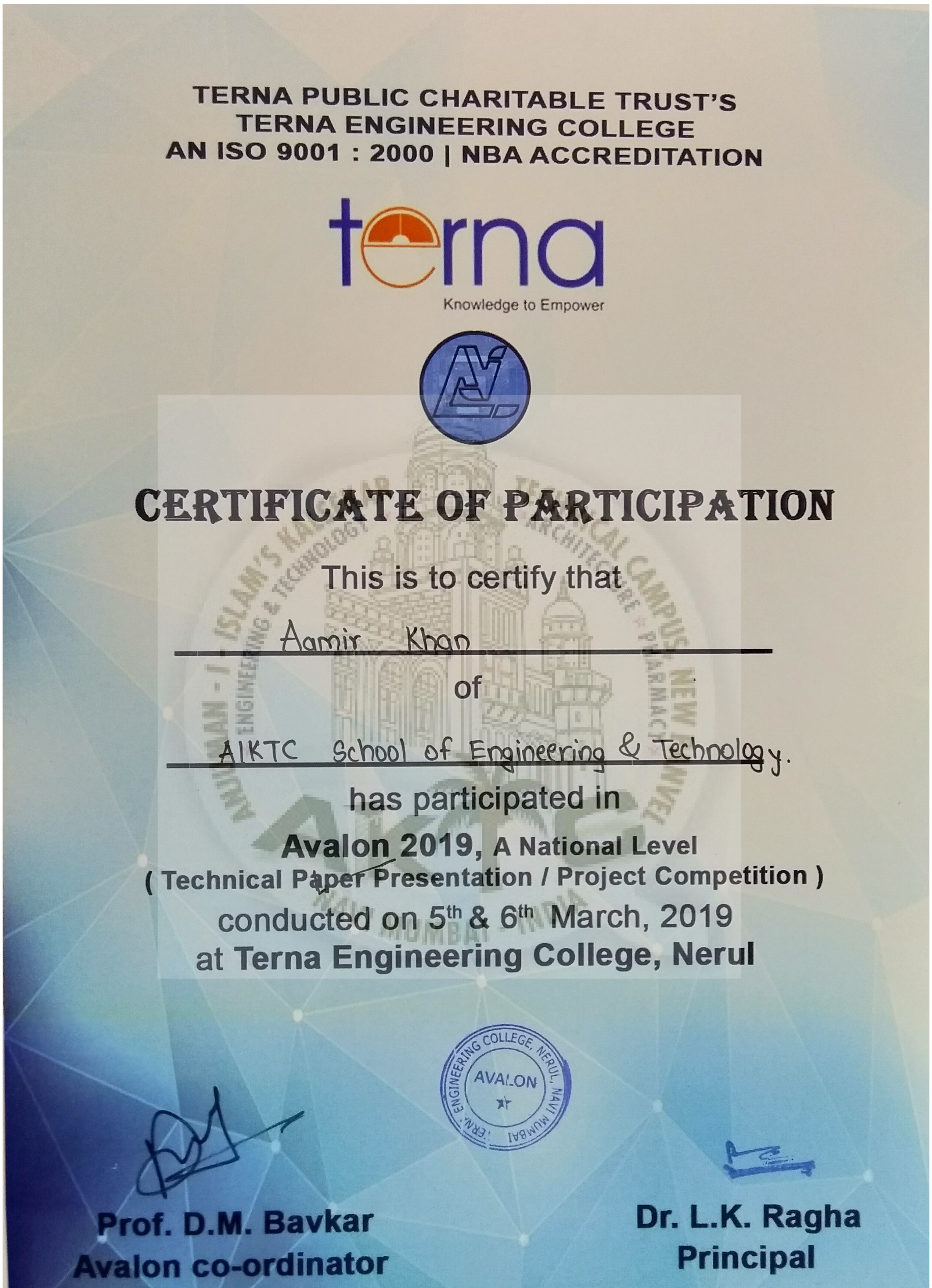
In recognition the publication of the manuscript entitled

A Web Application on Inventory Management System for Server Center of AIKTC

published in our Journal Volume 6 Issue 2 February 2019

Impact Factor : 7.211 www.irjet.net Editor in Chief E-mail : editor@irjet.net

Figure 9.4: Publication Certificate- Heramb Pandit



TERNA PUBLIC CHARITABLE TRUST'S
TERNA ENGINEERING COLLEGE
AN ISO 9001 : 2000 | NBA ACCREDITATION

terna
Knowledge to Empower

CERTIFICATE OF PARTICIPATION

This is to certify that

Aamir Khan

of

AIKTC School of Engineering & Technology.

has participated in

Avalon 2019, A National Level
(Technical Paper Presentation / Project Competition)
conducted on 5th & 6th March, 2019
at **Terna Engineering College, Nerul**

TERNA ENGINEERING COLLEGE, NERUL, NAVI MUMBAI
AVALON

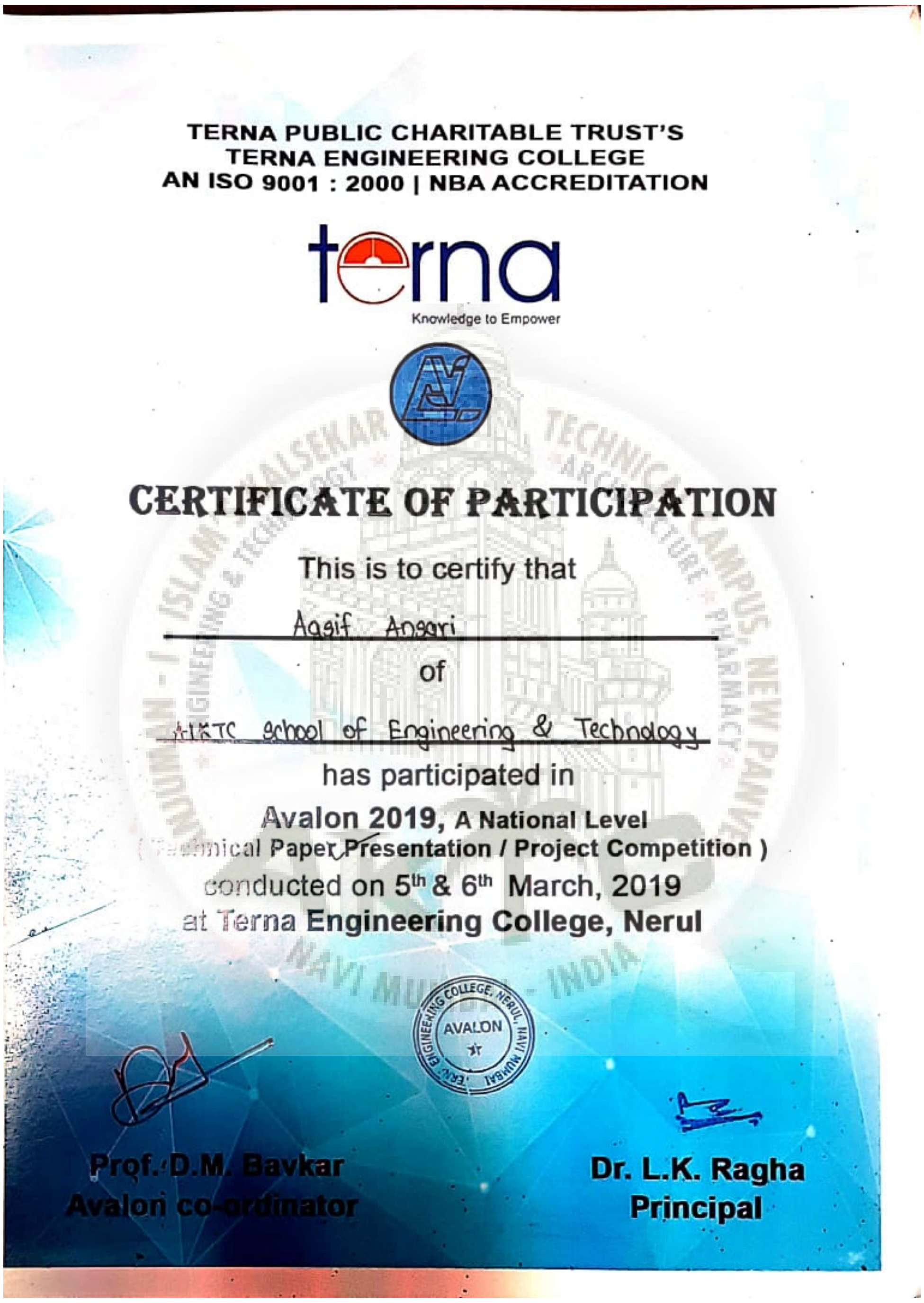
[illegible]

Prof. D.M. Bavkar
Avalon co-ordinator

[illegible]

Dr. L.K. Ragha
Principal

Figure 9.5: Paper Presentation Certificate- Aamir Khan



TERNA PUBLIC CHARITABLE TRUST'S
TERNA ENGINEERING COLLEGE
AN ISO 9001 : 2000 | NBA ACCREDITATION

terna
Knowledge to Empower

CERTIFICATE OF PARTICIPATION

This is to certify that

Aasif Ansari

of

AIKTC School of Engineering & Technology

has participated in

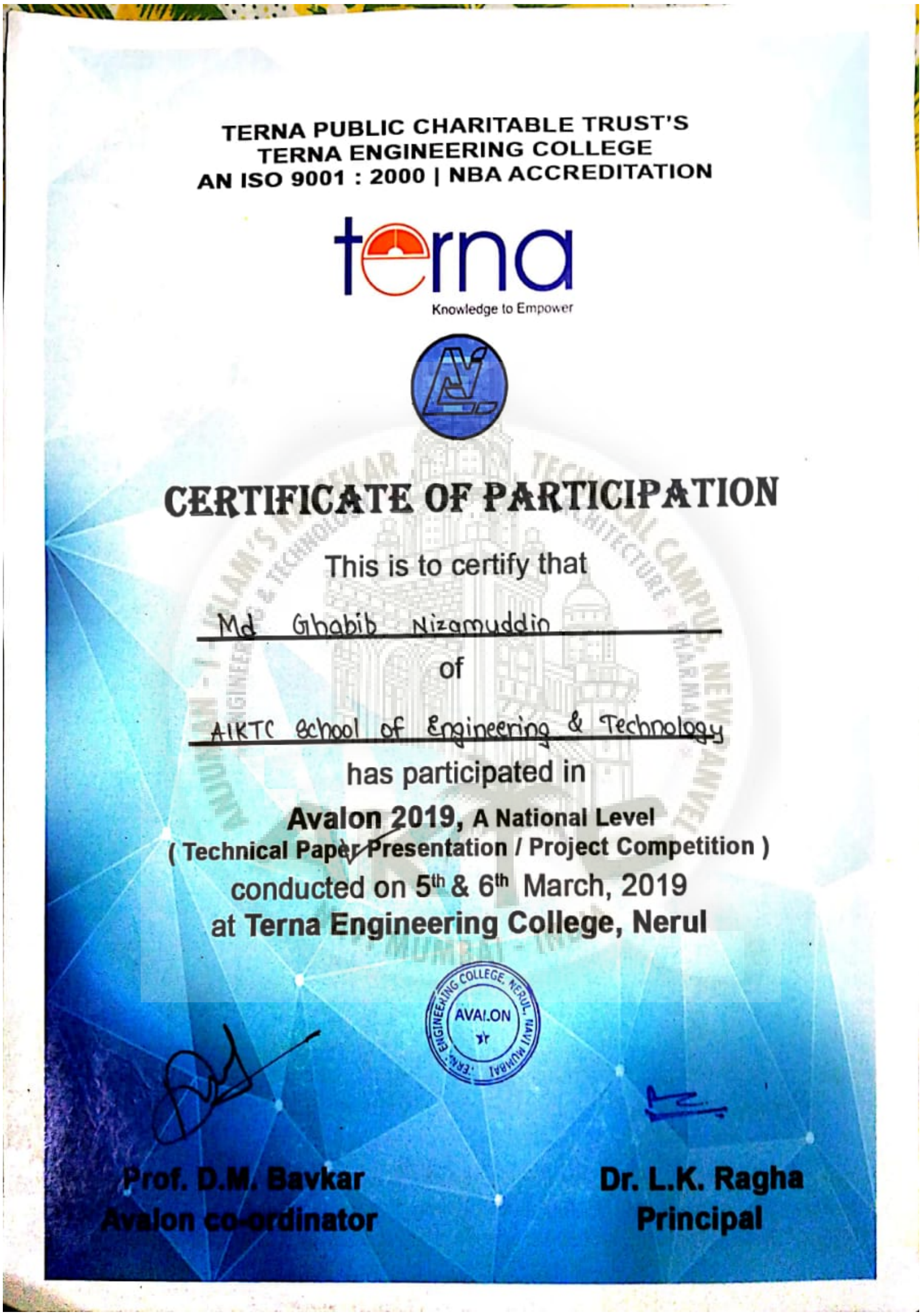
Avalon 2019, A National Level
(Technical Paper Presentation / Project Competition)
conducted on 5th & 6th March, 2019
at Terna Engineering College, Nerul

TERNA ENGINEERING COLLEGE, NERUL, NAVI MUMBAI AVALON

[illegible]
Prof. D.M. Bavkar
Avalon co-ordinator

[illegible]
Dr. L.K. Ragha
Principal

Figure 9.6: Paper Presentation Certificate- Aasif Ansari



TERNA PUBLIC CHARITABLE TRUST'S
TERNA ENGINEERING COLLEGE
AN ISO 9001 : 2000 | NBA ACCREDITATION

terna
Knowledge to Empower

CERTIFICATE OF PARTICIPATION

This is to certify that

Md Ghabib Nizamuddin

of

AIKTC School of Engineering & Technology

has participated in

Avalon 2019, A National Level
(Technical Paper Presentation / Project Competition)
conducted on 5th & 6th March, 2019
at Terna Engineering College, Nerul

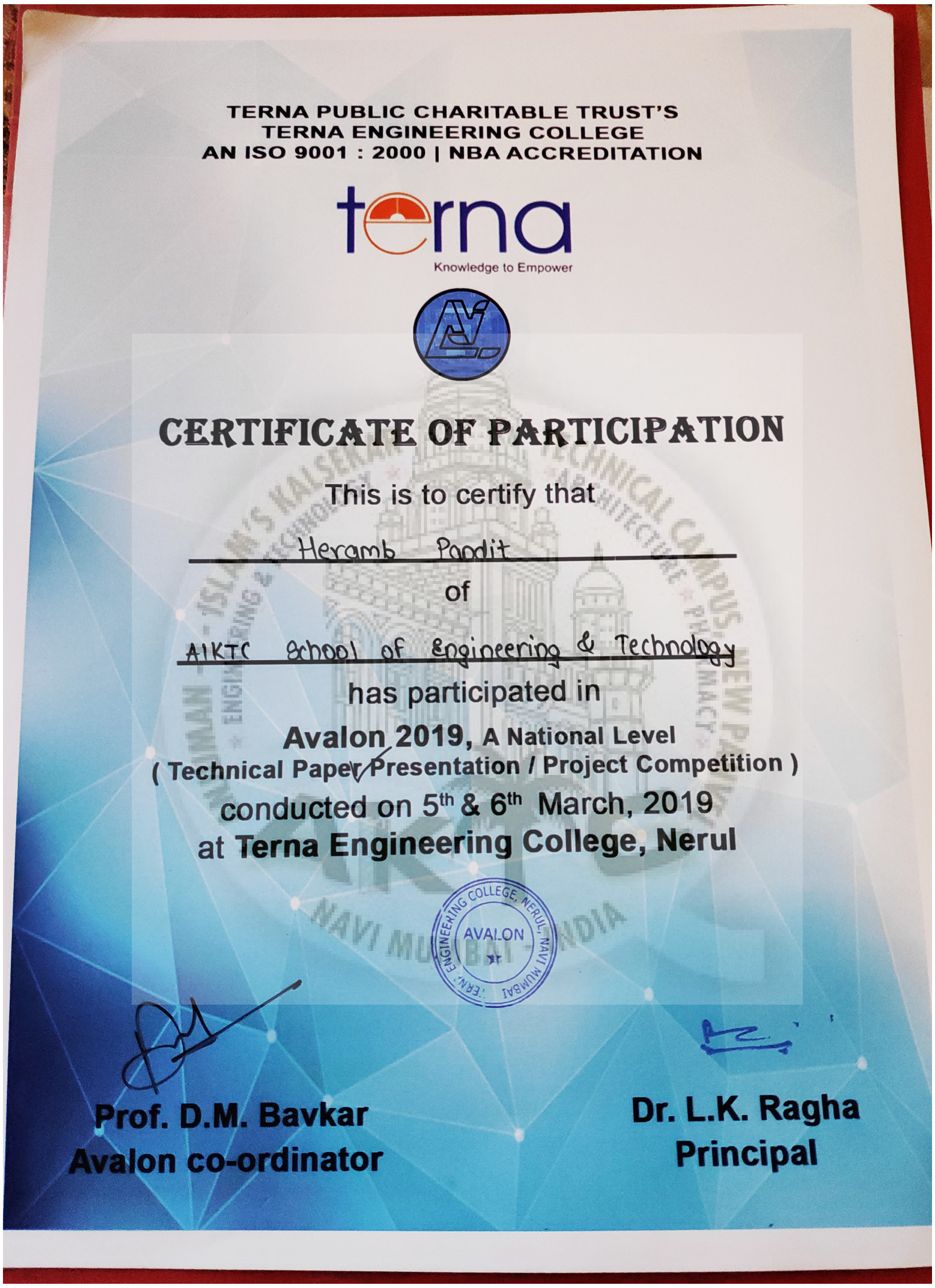
TERNA ENGINEERING COLLEGE, NERUL, NAVI MUMBAI
AVALON

[illegible] [illegible]

Prof. D.M. Bavkar
Avalon co-ordinator

Dr. L.K. Ragha
Principal

Figure 9.7: Paper Presentation Certificate- Md Ghalib



TERNA PUBLIC CHARITABLE TRUST'S
TERNA ENGINEERING COLLEGE
AN ISO 9001 : 2000 | NBA ACCREDITATION

terna
Knowledge to Empower

CERTIFICATE OF PARTICIPATION

This is to certify that

Heramb Pandit

of

AIKTC School of Engineering & Technology

has participated in

Avalon 2019, A National Level
(Technical Paper Presentation / Project Competition)
conducted on 5th & 6th March, 2019
at Terna Engineering College, Nerul

TERNA ENGINEERING COLLEGE, NERUL, NAVI MUMBAI AVALON

[illegible]

Prof. D.M. Bavkar
Avalon co-ordinator

[illegible]

Dr. L.K. Ragha
Principal

Figure 9.8: Paper Presentation Certificate- Heramb Pandit