

A Project Report Submitted

On

HARDWARE MAINTENANCE SYSTEM

Submitted by

V. Bhaskar (16NG1A05B4)

In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

Under the Esteemed Guidance of

Mr. M. Samba Siva Rao
Associate Professor



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

USHA RAMA COLLEGE OF ENGINEERING AND TECHNOLOGY

(Approved by AICTE, Affiliated to JNTU Kakinada)

An ISO 9001:2015 Certified Institution

(ON NH 16, TELAPROLU, NEAR GANNAVARAM-521109)

2019-2020

A Project Report Submitted

on

HARDWARE MAINTENANCE SYSTEM

Submitted by

V. Bhaskar

(16NG1A05B4)

In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

Under the Esteemed Guidance of

Mr. M. SAMBA SIVA RAO

Associate Professor



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
USHA RAMA COLLEGE OF ENGINEERING AND TECHNOLOGY**

(Approved by AICTE, Affiliated to JNTU Kakinada)

An ISO 9001:2015 Certified Institution

(ON NH 16, TELAPROLU, NEAR GANNAVARAM-521109)

2019-2020

USHA RAMA COLLEGE OF ENGINEERING AND TECHNOLOGY



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(Approved by AICTE, Affiliated to JNTU Kakinada)

An ISO 9001:2015 Certified Institution

(ON NH 16, TELAPROLU, NEAR GANNAVARAM-521109)

CERTIFICATE

This is to certify that this project entitled **HARDWARE MAINTENANCE SYSTEM** is the bonafide work of **V. Bhaskar (16NG1A05B4)** who carried out the work under my supervision, and submitted in partial fulfillment of the requirements for the award of the degree in Bachelor of Technology in Computer Science & Engineering , during the academic year 2019-20.

Project Guide

Mr. M. Samba Siva Rao

Head of the Department

Dr. S. M. Roy Choudri

Signature of External Examiner

<https://usharama.edu.in/home> Tel: 0866 252755, +91 9949712255

DECLARATION

I hereby declare that the project entitled **“HARDWARE MAINTENANCE SYSTEM”** is the work done by me during the academic year 2019-2020 and is submitted in partial fulfillment of the requirements for the award of degree of Bachelor of technology in **COMPUTER SCIENCE AND ENGINEERING** from **JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, KAKINADA.**

ACKNOWLEDGEMENT

I express my sincere thanks where it is due

Foremost I would like to extend my heartfelt thanks to my beloved parents whose blessings and encouragement were always there as source of strength and inspiration.

I'm pleased to acknowledge my sincere thanks to our Honorable Chairman **Mr.Sunkara Rama Brahman** for providing support and stimulating environment for developing the project.

My sincere thanks to our beloved Director, **Dr. Kurra Raja Sekhar Rao** for his advice, guidance and resources provided.

I express sincere thanks to **Dr. G V K S V Prasad**, Principal of **USHARAMA COLLEGE OF ENGINEERING AND TECHNOLOGY**, Telaprolu; for providing us the resources for carrying out the project.

My sincere thanks to our Head of Department, **Dr. S M Roy Choudri** for his co-operation and guidance in helping me to make my project successful and complete in all aspects. I'm grateful for his precious guidance and suggestions.

I express my sincere thanks to my **Mr. M. SAMBA SIVA RAO**, Associate Professor in the **Department of CSE** for motivating me to make my project successful. I grateful for his precious guidance and suggestions.

Last but not the least, I also place my floral gratitude to all my teaching staff and lab technicians for their constant support and advice throughout the project.

V. Bhaskar
16NG1A05B4

HARDWARE MAINTENANCE SYSTEM

ABSTRACT

ABSTRACT

Computers and its peripherals have become a part of our daily life automating several tasks and producing outputs time efficiently. Yet, even their face with several hardware or software problems such as ram issues, power fault, storage disk etc. It is the duty of Lab Technician to take care of such problems. In general scenario, tech support maintains a ledger containing a record of all problems reported. As it is a non-digital record it is susceptible to several issues like getting lost, paper damage etc.

In this project, we intended to create a centralized data of systems and its peripherals of Usharama. All this data will be stored in a database (MongoDB) along with a backup. This project consists of six shareholders namely College Administrators, Head of Departments, Lab Technician, Lab Incharge, Guest (Students & Teachers) and Admin. It consists details of every laboratory, computers, printers and all hardware peripherals in Usharama.

Whenever a student or a faculty member faces an issue regarding system performance, he/she can login into the HMS by selecting guest login and rise a token. This token should be saved to know the status of the issue. This issue gets stored in the database and ne notified to technician dashboard.

Each technician will be assigned the work to be done automatically and after completing the work they would change the status in the dashboard; which has to be justified by the incharge. This entire process can be monitored by the administrators who can also get the status reports of each lab and its peripherals.

KEYWORDS:

Digitalized records, Token generation, Dashboard, Lazy Loading, Reports.

TABLE OF CONTENT

TOPIC	PAGE NO
1. INTRODUCTION	1
1.1. Literature Survey	2
1.1.2. Existing System	3
1.1.3. Proposed System	4
2. AIM & SCOPE	6
2.1. Feasibility study	7
2.1.1. Technical Feasibility	7
2.1.2. Operational Feasibility	8
2.1.3. Economic Feasibility	8
2.2 System Requirements Specification	9
2.2.1 Non Functional Requirements	9
2.2.2 Software Requirements	10
2.2.3 Hardware Requirements	10
3. CONCEPTS & METHODS	12
3.1. Problem Description	12
3.2. Proposed solution	13
3.2.1. Status Levels	13
3.2.2 Modules	14
3.3. System analysis methods	16
3.3.1. Use case Diagram	16
3.3.2. Activity Diagram	18
3.4. System design	21
3.4.1. Input Design	22
3.4.2. Output Design	24
3.4.3. Class diagram	25
3.4.4. Sequence Diagram	26

4. IMPLEMENTATION	28
4.1. Tools used	29
4.2. Database collections	40
4.3. Pseudo code	42
4.4. Component Diagram	79
4.5. Deployment Diagram	80
5. SCREEN SHOTS	82
6. TESTING	86
6.1. Test case-1	86
6.2. Test case-2	88
6.3. Test case-3	89
7. SUMMARY & CONCLUSION	91
8. FUTURE ENHANCEMENTS	93
9. BIBLIOGRAPHY	95

LIST OF FIGURES

Fig. No.	Figure Name	Page No
Fig 3.3.1.1	Usecase Diagram for Hardware Maintenance System	17
Fig 3.3.2.1	Activity Diagram for admin module	32
Fig 3.3.2.2	Activity Diagram for college administrator module	19
Fig 3.3.2.3	Activity Diagram for incharge module	19
Fig 3.3.2.4	Activity Diagram for technician module	20
Fig 3.3.2.5	Activity Diagram for guest module	20
Fig 3.4.3.1	Class Diagram for Hardware Maintenance System	25
Fig 3.4.4.1	Sequential Diagram for Hardware Maintenance System	26
Fig 4.1.1	Module Architecture	30
Fig 4.1.2	Starter Application	32
Fig 4.1.3	MongoDB Example	37
Fig 4.1.4	MongoDB vs RDBMS	39
Fig 4.4	Component Diagram for Hardware Maintenance System	79
Fig 4.5	Deployment Diagram for Hardware Maintenance System	80

LIST OF SCREENSHOTS

Fig. No.	Screenshot Name	Page No
Fig 5.1	Home Screen	82
Fig 5.2	Report Screen	82
Fig 5.3	Check Statue	83
Fig 5.4	Login Screen	83
Fig 5.5	Display list of reports	84
Fig 5.6	Create new lab	84
Fig 6.1.1	Entering credentials	86
Fig 6.1.2	Welcome Screen	87
Fig 6.1.3	Invalid Credentials	87
Fig 6.2.1	Reporting a problem	88
Fig 6.2.2	Checking status	88
Fig 6.3.1	Reported problems	89
Fig 6.3.2	In progress problems	89

CHAPTER-1

INTRODUCTION

1. INTRODUCTION:

The aim of this project is to have a centralized digital record of all computers and its peripherals of each lab in Usharama. This project helps to track down each problem easily and update its status. This project consists of six shareholders namely College Administrators, Head of Departments, Lab Technician, Lab Incharge, Guest (Students & Teachers) and Admin. It contains details of every lab equipment of Usharama stored digitally.

Now-a-days, computers and its peripherals have become a part of our daily life automating several tasks and producing outputs time efficiently. Yet, even their face with several hardware or software problems such as ram issues, power fault, storage disk etc. It is the duty of Lab Technician to take care of such problems. In general scenario, tech support maintains a ledger containing a record of all problems reported. As it is a non-digital record it is susceptible to several issues like getting lost, paper damage etc.

Currently all hardware peripherals don't have a digitalized record making things difficult for the lab technicians to solve them whenever a problem arises. All complaints regarding the systems or its peripherals has to be manual forwarded to technician using phone call or ledger which often results in delay of solution. Even after rectifying no one knows the result except the technician. Administrates are left blind regarding the peripherals used and status reports.

By adapting to a maintenance system, we would be able to have a centralized record of all peripherals available in every lab and department. Whenever a student or a faculty member faces an issue regarding system performance, he/she can login into the HMS by selecting guest login and rise a token. This token should be saved to know the status of the issue. This issue gets stored in the database and be notified to technician dashboard.

Each technician will be assigned the work to be done automatically and after completing the work they would change the status in the dashboard; which has to be justified by the incharge. This entire process can be monitored by the administrators who can also get the status reports of each lab and its peripherals.

1.1 LITERATURE SURVEY

Literature survey is the most important step in software development process. Before developing the tool, it is necessary to determine time factor, economy and company strength. Once these things are satisfied, then next steps are to be determine which operating system and language can be used for the developing tool.

Once the programmers start building the tool the programmers need a lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system, the above considerations are taken into account for developing the proposed system.

Computers and its components can get into several software/hardware related problems. It is the job of Technician Support team to rectify such problems. But due to manual entering of issues in ledger people often gets confused which problem to solve first in respective of its priorities and problem difficulty.

So, to solve this problem, we can maintain a centralized digital record of component data. In this way, it would be easy to apply filters and get required data easily which in returns help the lab incharge or technician to solve the issues quickly.

1.1.2. EXISTING SYSTEM

In current scenario, all hardware peripherals don't have a digitalized record making things difficult for the lab technicians to solve them whenever a problem arises. All complaints regarding the systems or its peripherals has to be manual forwarded to technician using phone call or meeting him directly and reporting the issue. Only then he will be aware of the problem and record it in ledger. This process is the time consuming one which often leads to delay of the problem rectification. Even after rectifying no one knows the result except the technician. Administrates are left blind regarding the peripherals used and status reports.

Now, after being aware of the problem; the technician starts working on it. After completing the issue, he again has to make a new entry in the ledger stating its status. Here only the technician who solved the problem is been aware of its status. No one else knows about its status. If someone has to know status of any particular system, then they have to spend a lot of time tracking its entry in the lodger; the day of issue reported, day of solved etc.

Again if any the administrators want to know the status of the lab components regarding its status, no.of working machines, no.of non-working machines; then the lab incharge has to go through every single page to track down each and every problem regarding his lab.

So even after doing all this heavy work, we cannot guarantee the safety of the ledger. As it is just a book without locks anyone can enter any misleading entries into it. Also, it can be stolen or the paper can be damaged due to water etc. Now, we have to do the same process again because we don't have any backup for such fault conditions.

LIMITATIONS:

- No security measures to ledger in case of thief or misleading entries.
- Doesn't have any backup in case of faults.
- Time consuming task to track down issues.
- No acknowledgement of problem status.

1.1.3. PROPOSED SYSTEM

In this project, we intend to create a centralized record of all peripherals available in every lab and department. Whenever a student or a faculty member faces an issue regarding system performance, he/she can login into the HMS by selecting guest login and rise a token. This token should be saved to know the status of the issue. This issue gets stored in the database and be notified to technician dashboard.

Each technician will be assigned the work to be done automatically and after completing the work they would change the status in the dashboard; which has to be justified by the incharge. HMS has various status levels to easily understand the status of the problem. They are working, reported, in_progress, awaiting, solved, need_care. Lab Technician can change the status to any of the above levels determining its condition.

The user(Guest) who have reported the user can track the problem to get its status to any given time by using its token ID. College Administrators and Head of Departments can get the status reports of any labs at any given time stating no.of working systems, no.of non-working, systems count etc.

ADVANTAGES:

- Secured data entry.
- Ease tracking of issue.
- Backup of data in case of faults.
- Time effective filtering and tracking.
- Different status levels to represent problem condition.

CHAPTER-2

AIM AND SCOPE

2.AIM AND SCOPE

Now-a-days, computers and its peripherals have become a part of our daily life automating several tasks and producing outputs time efficiently. Yet, even their face with several hardware or software problems such as ram issues, power fault, storage disk etc. It is the duty of Lab Technician to take care of such problems. In general scenario, tech support maintains a ledger containing a record of all problems reported. As it is a non-digital record it is susceptible to several issues like getting lost, paper damage etc.

The aim of this project is to have a centralized digital record of all computers and its peripherals of each lab in Usharama. This project helps to track down each problem easily and update its status. This project consists of six shareholders namely College Administrators, Head of Departments, Lab Technician, Lab Incharge, Guest (Students & Teachers) and Admin. It contains details of every lab equipment of Usharama stored digitally.

Whenever a student or a faculty member faces an issue regarding system performance, he/she can login into the HMS by selecting guest login and rise a token. This token should be saved to know the status of the issue. This issue gets stored in the database and be notified to technician dashboard.

Each technician will be assigned the work to be done automatically and after completing the work they would change the status in the dashboard; which has to be justified by the incharge. HMS has various status levels to easily understand the status of the problem. They are working, reported, in_progress, awaiting, solved, need_care. Lab Technician can change the status to any of the above levels determining its condition.

This entire process can be monitored by the administrators who can also get the status reports of each lab and its peripherals. The user who have reported the user can track the problem to get its status to any given time by using its token ID.

2.1.FEASIBILITY STUDY:

The feasibility of the project is analyzed in this phase. During system analysis the feasibility study of the proposed system is to be carried out. For feasibility analysis, some understanding of the major requirements for the system is essential.

The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

- Technical Feasibility
- Operational Feasibility
- Economical Feasibility

2.1.1 Technical Feasibility:

The technical issue usually raised during the feasibility stage of the investigation includes the following:

- Does the necessary technology exist to do what is suggested?
- Do the proposed equipment have the technical capacity to hold the data required to use the new system?
- Will the proposed system provide adequate response to inquiries, regardless of the number of users?
- Can the system be upgraded if developed?
- Are there technical guarantees of accuracy, reliability, ease of access and data security?

The current system developed is technically feasible. Thus it provides an easy access to the users. Therefore, it provides the technical guarantee of accuracy, reliability and security. The work for the project is done with the current equipment and existing software technology. Necessary bandwidth exists for providing a fast feedback to the users irrespective of the number of users using the system.

2.1.2 Operational Feasibility:

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization's operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. Some of the important issues raised are to test the operational feasibility of a project includes the following: -

- Is there sufficient support for the management from the users?
- Will the system be used and work properly if it is being developed and implemented?
- Will there be any resistance from the user that will undermine the possible application benefits?

This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits.

The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

2.1.3 Economic feasibility:

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economic feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs.

The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies.

2.2 System Requirements Specification:

A Software Requirements Specification (SRS) – a requirements specification for a software system– is a complete description of the behavior of a system to be developed. It includes a set of use-cases that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. Non-functional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).

System requirements specification is a structured collection of information that embodies the requirements of a system. A business analyst, sometimes titled system analyst, is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions.

2.2.1. Non-functional requirements:

The major non-functional Requirements of the system are as follows:

Usability:

The system is designed with semi automated process hence there is no or less user intervention.

Reliability:

The system is more reliable because of the qualities that are inherited from the chosen platform python. The code built by using python is more reliable.

Performance:

This system is developing in the high level languages and using the advanced front-end and back-end technologies it will give response to the end user on client system with in very less time.

Supportability:

The system is designed to be the cross platform supportable. The system is supported on a wide range of software platform, built into the system.

2.2.2. Software Requirements:

- MongoDB
- Express.js
- Angular
- NodeJS
- mongodb.js
- cors.js

2.2.3. Hardware requirements:

- Ram : 4GB
- Hard Disk : 100 GB
- Processor : Intel Xeon

CHAPTER-3

CONCEPTS AND METHODS

3.CONCEPTS AND METHODS

Student or a faculty member faces an issue regarding system performance, he/she can login into the HMS by selecting guest login and rise a token. This token should be saved to know the status of the issue. This issue gets stored in the database and be notified to technician dashboard. Each technician will be assigned the work to be done automatically and after completing the work they would change the status in the dashboard; which has to be justified by the incharge.

The Technology Acceptance Model(TAM) measure are used to evaluate the perceived effectiveness of our system. The motivation to introduce this concept is that: through my experiments, we found out that the ability to discriminate the target from its surrounding background is the key element that resolves the tracking success or failure.

3.1. Problem Description:

All hardware peripherals don't have a digitalized record making things difficult for the lab technicians to solve them whenever a problem arises. All complaints regarding the systems or its peripherals has to be manual forwarded to technician using phone call or meeting him directly and reporting the issue. Only then he will be aware of the problem and record it in ledger. This process is the time consuming one which often leads to delay of the problem rectification. Even after rectifying no one knows the result except the technician. Administrates are left blind regarding the peripherals used and status reports.

We cannot guarantee the safety of the ledger. As it is just a book without locks anyone can enter any misleading entries into it. Also, it can be stolen or the paper can be damaged due to water etc. Now, we have to do the same process again because we don't have any backup for such fault conditions.

3.2. PROPOSED SOLUTION:

So, instead of having a manual entry ledger, the proposed system intends to create a centralized record of all peripherals available in every lab and department. Whenever a student or a faculty member faces an issue regarding system performance, he/she can login into the HMS by selecting guest login and rise a token. This token should be saved to know the status of the issue. This issue gets stored in the database and be notified to technician dashboard.

Each technician will be assigned the work to be done automatically and after completing the work they would change the status in the dashboard; which has to be justified by the incharge. HMS has various status levels to easily understand the status of the problem. They are working, reported, in_progress, awaiting, solved, need_care. Lab Technician can change the status to any of the above levels determining its condition.

The user who have reported the user can track the problem to get its status to any given time by using its token ID. College Administrators and Head of Departments can get the status reports of any labs at any given time stating no.of working systems, no.of non-working, systems count etc.

3.2.1 Status Levels:

In this project, we have six status levels to determine the condition of the reported problems. They are :

- 1. Working:** Indicates that a system is working perfectly without any faults.
- 2. Reported:** Indicates that a user has reported a system as not working and needs technician attention.
- 3. In_progress:** Indicates that a technician is working on the problem reported.
- 4. Awaiting:** Indicates that technician has solved the issue and lab incharge must verify its condition.
- 5. Solved:** Indicates that the issue has been rectified, user can work on it again.
- 6. Need_care:** This level is used to indicate that technician has encountered some serious hardware/software fault and he needs extra variables such as buying new component to solve this problem.

3.2.2. MODULES:

1. Guest Module
2. Technician Module
3. Incharge Module
4. Head of Department Module
5. College Administrators Module
6. Admin Module

3.2.2.1. Description of each module:

1. Guest Module:

Guest can be anyone whether a student or a faculty member. When a user encounters an issue regarding system performance, he/she can login into the HMS by selecting guest login and rise a token. User will be prompted with a token ID which can be used to track his issue and check its status.

2. Technician Module:

Every lab has a technician assigned to rectify computer components issues. When a user reported a new problem, it raises a token and will be added to technician dashboard. Technician can next the reported issues and start working on them. After solving the user, he can update its status to awaiting or need_care.

3. Incharge Module:

Incharge is responsible for taking care of each and every system in his lab. After technician completes his work, this token will be notified to incharge dashboard. Incharge now check the system and again update its status to reported if not working or solved is the task is completed.

4.Head of Department Module:

Head of Department can raise a token by selecting any non working system in his department and allocate the task to any technician in the college. He can also get the status reports of all systems in his department.

5.College Administrators Module:

Chairman, Director, Principal comes under this category. They can get the overall status reports of all systems, labs, departments in the college.

6.Admin Module:

Admin is the person responsible to care of the proper execution of web application. He can either create, update and delete any lab, department, members in the college. His main responsible is to create labs and assign systems and lab incharge and lab technician to it. He has write permission to the database. He can also update status of lost issues.

3.3. SYSTEM ANALYSIS METHODS:

System Analysis is first stage according to System Development Life Cycle model. This System Analysis is a process that starts with the analyst. Analysis is a detailed study of the various operations performed by a system and their relationships within and outside the system. One aspect of analysis is defining the boundaries of the system and determining whether or not a candidate should consider other related systems.

During analysis, data is collected from the available files, decision points, and transactions handled by the present system. Logical system models and tools are used in analysis. Training, experience, and common sense are required for collection of the information needed to do the analysis

Logical system models and tools are used in analysis. Training, experience, and common sense are required for collection of the information needed to do the analysis. This System Analysis is a process that starts with the analyst. Analysis is a detailed study of the various operations performed by a system and their relationships within and outside the system.

3.3.1. Use-case diagram:

A use case is a set of scenarios that describing an interaction between a user and a system. Use case diagram displays the relationship among actors and use cases. The two main components of a use case diagram are use cases and actors.

An actor represents a user or another system that will interact with the system you are modeling. A use case is an external view of the system that represents some action the user might perform in order to complete a task.

Contents:

- Use cases
- Actors
- Dependency, Generalization, and association relationships
- System boundary

Use-case diagram for Hardware Maintenance System:

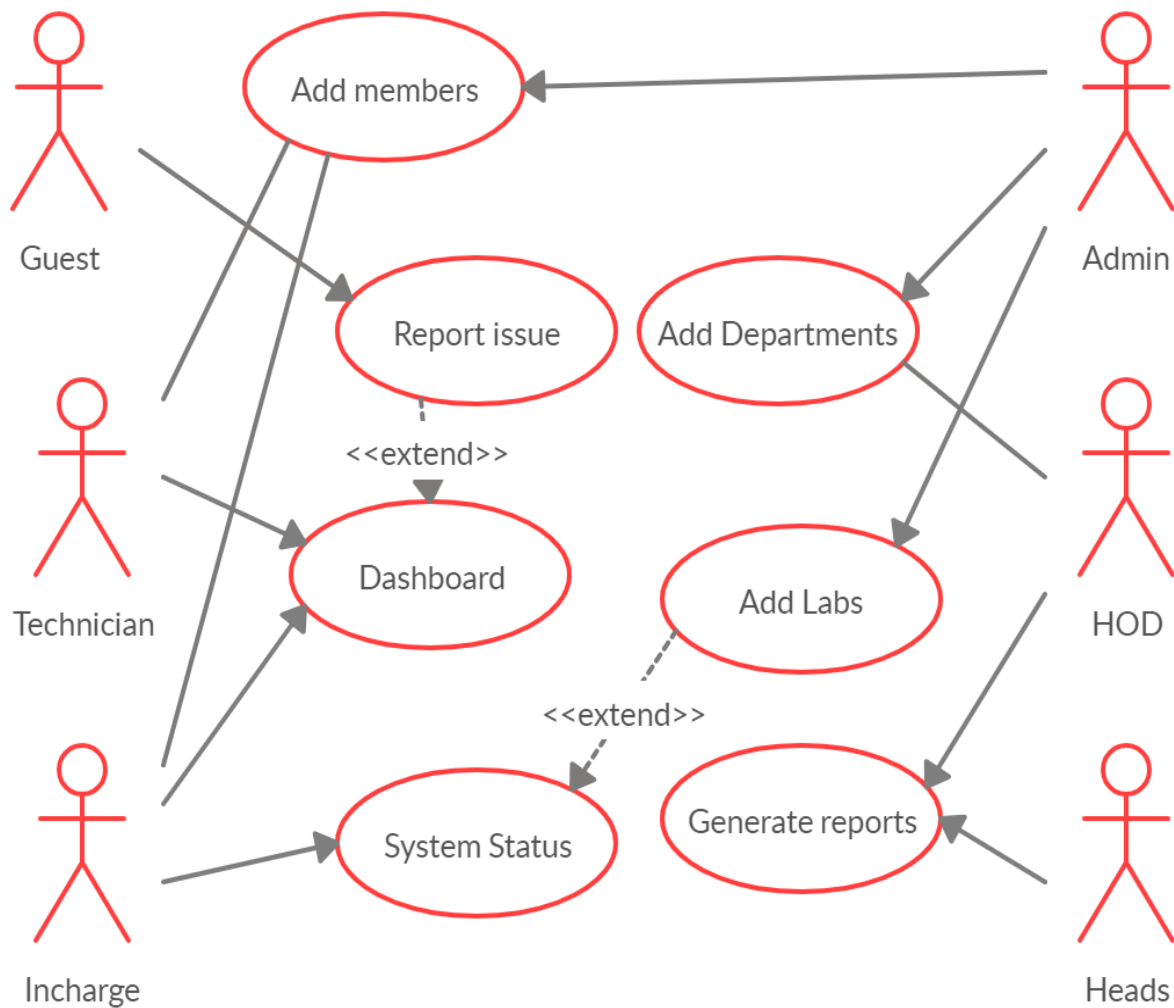


Fig 3.3.1.1: Use-case diagram for Hardware Maintenance System

Diagram Description:

The above figure shows us the use case diagram. Here Guest, Technician, Incharge, Admin, HOD and Heads can act as actors. Guest reports a problem which can be derived by dashboard of issues. Admin can add members who can performance operations accordingly. Head of Department and College Administrators can generate lab reports periodically.

3.3.2. Activity diagram:

Activity diagrams describe the workflow behavior of a system. Activity diagrams are similar to state diagrams because activities are the state of doing something. The diagrams describe the state of activities by showing the sequence of activities performed. Activity diagrams can show activities that are conditional or parallel. Activity diagrams show the flow of activities through the system. Diagrams are read from top to bottom.

Activity diagrams should be used in conjunction with other modeling techniques such as interaction diagrams and state diagrams. The main reason to use activity diagrams is to model the workflow behind the system being designed. Activity Diagrams are also useful for: analyzing a use case by describing what actions needs to take place and when they should occur; describing a complicated sequential algorithm; and modeling applications with parallel processes.

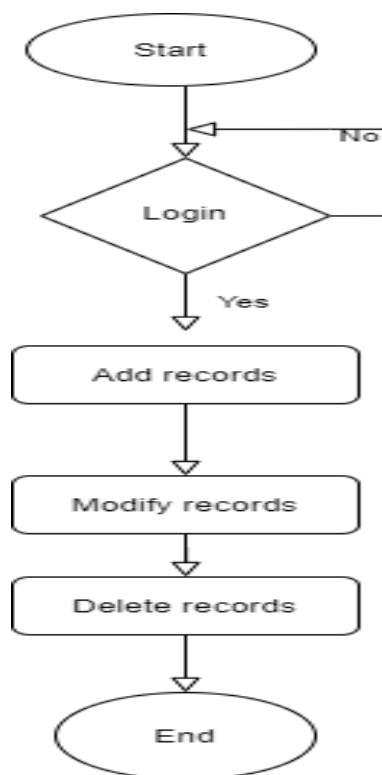


Fig 3.3.2.1: Activity Diagram for Admin Module

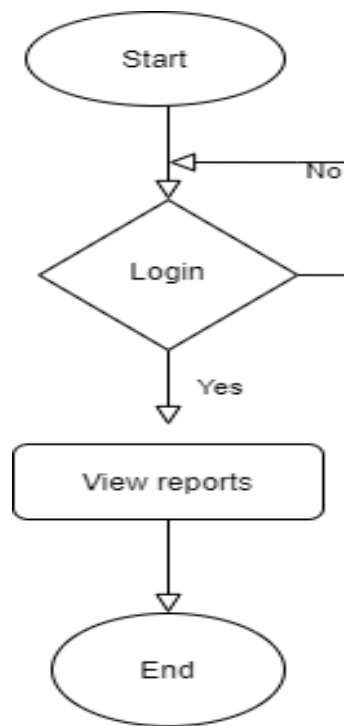


Fig 3.3.2.2 : Activity diagram for College Administrator Module

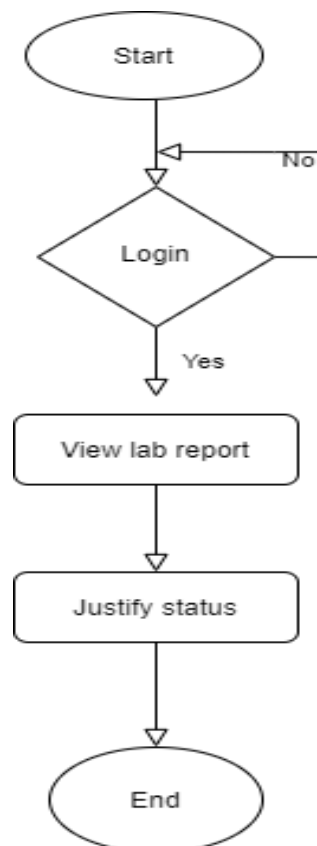


Fig 3.3.2.3 : Activity diagram for Incharge Module

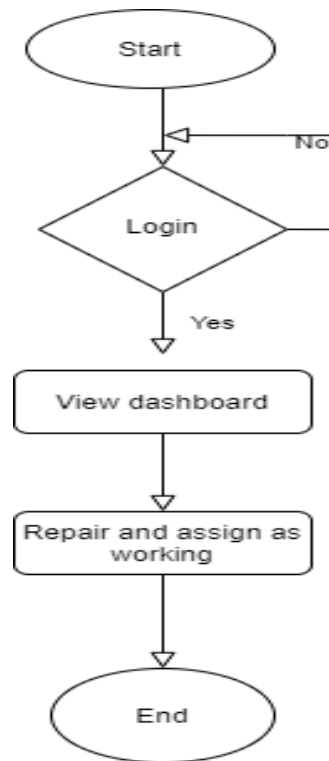


Fig 3.3.2.4 : Activity diagram for Technician Module

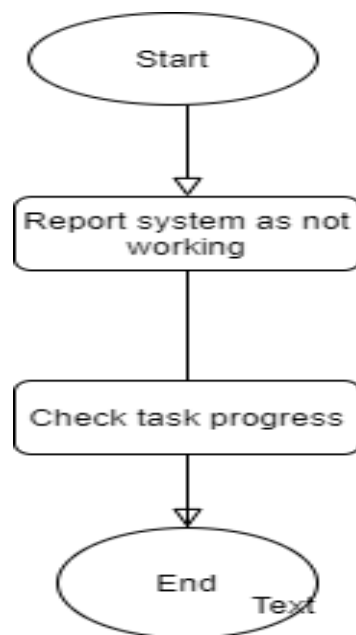


Fig 3.3.2.5 : Activity diagram for Guest Module

3.4. SYSTEM DESIGN:

The most creative and challenging phase of the life cycle is system design. The term design describes a final system and the process by which it is developed. It refers to the technical specifications that will be applied in implementations of the candidate system. The design may be defined as “the process of applying various techniques and principles for the purpose of defining a device, a process or a system with sufficient details to permit its physical realization”.

The designer’s goal is how the output is to be produced and in what format. Samples of the output and input are also presented. Second input data or database files have to be designed to meet the requirements of the proposed output. The processing phases are handled through the program Construction and Testing. Finally, details related to justification of the system and an estimate of the impact of the candidate system on the user and the organization are documented and evaluated by management as a step toward implementation.

The importance of software design can be stated in a single word “Quality”. Design provides us with representations of software that can be assessed for quality. Design is the only way where we can accurately translate a customer’s requirements into a complete software product or system. Without design we risk building an unstable system that might fail if small changes are made. It may as well be difficult to test, or could be one who’s quality can’t be tested. So, it is an essential phase in the development of a software product.

3.4.1. Input Design:

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy.

Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

Justification

Today it is common to specify systems on higher levels using some natural language (e.g. English). For large systems, where large amounts of information must be handled, problems arise with ambiguities and inconsistencies with such specifications. Errors that are introduced are often detected late in the design cycle - in the simulation of the design after much design work has already been carried out - if detected at all.

By making the initial system specifications in a formal language at a high abstraction level, functionality can be verified/ simulated earlier in the development process. Ambiguities and inconsistencies can be avoided, errors can be discovered earlier, and the design iteration cycles can be shortened, thereby reducing development times. It is of critical importance that the specification language provides modeling concepts at a high abstraction level to allow the representation of system functions at a conceptual level without introducing unnecessary details.

Further, most of the languages that are used for implementation of HW / SW designs (e.g. JavaScript, VScode, TypeScript) do not lend themselves well to formal verification. This is because they lack a formally defined semantics or because the semantics is complex. A lack of formal semantics sometimes causes ambiguities in the interpretation of the designs. Our goal is to develop functional system specification method for physically disabled people, to demonstrate its efficiency on an industrially relevant example and to develop a tool to support the mapping of such specifications to synthesizable Angular. The specification language in which the system level functions will be developed will have a formal semantics in order to support formal verification of specifications.

3.4.2. Output Design:

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the soft copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.
2. Select methods for presenting information.
3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- ❖ Convey information about past activities, current status or projections of the
- ❖ Future.
- ❖ Opportunities, problems, or warnings.
- ❖ Trigger an action.
- ❖ Confirm an action.

3.4.3. Class diagram:

Class diagrams are widely used to describe the types of objects in a system and their relationships. Class diagrams model class structure and contents using design elements such as classes, packages and objects. Class diagrams describe three different perspectives when designing a system, conceptual, specification, and implementation. These perspectives become evident as the diagram is created and help solidify the design. Class diagrams are arguably the most used UML diagram type. It is the main building block of any object oriented solution. It shows the classes in a system, attributes and operations of each class and the relationship between each class. In most modeling tools a class has three parts, name at the top, attributes in the middle and operations or methods at the bottom. In large systems with many classes related classes are grouped together to create class diagrams. Different relationships between diagrams are show by different types of Arrows. Below is a image of a class diagram. Follow the link for more class diagram examples.

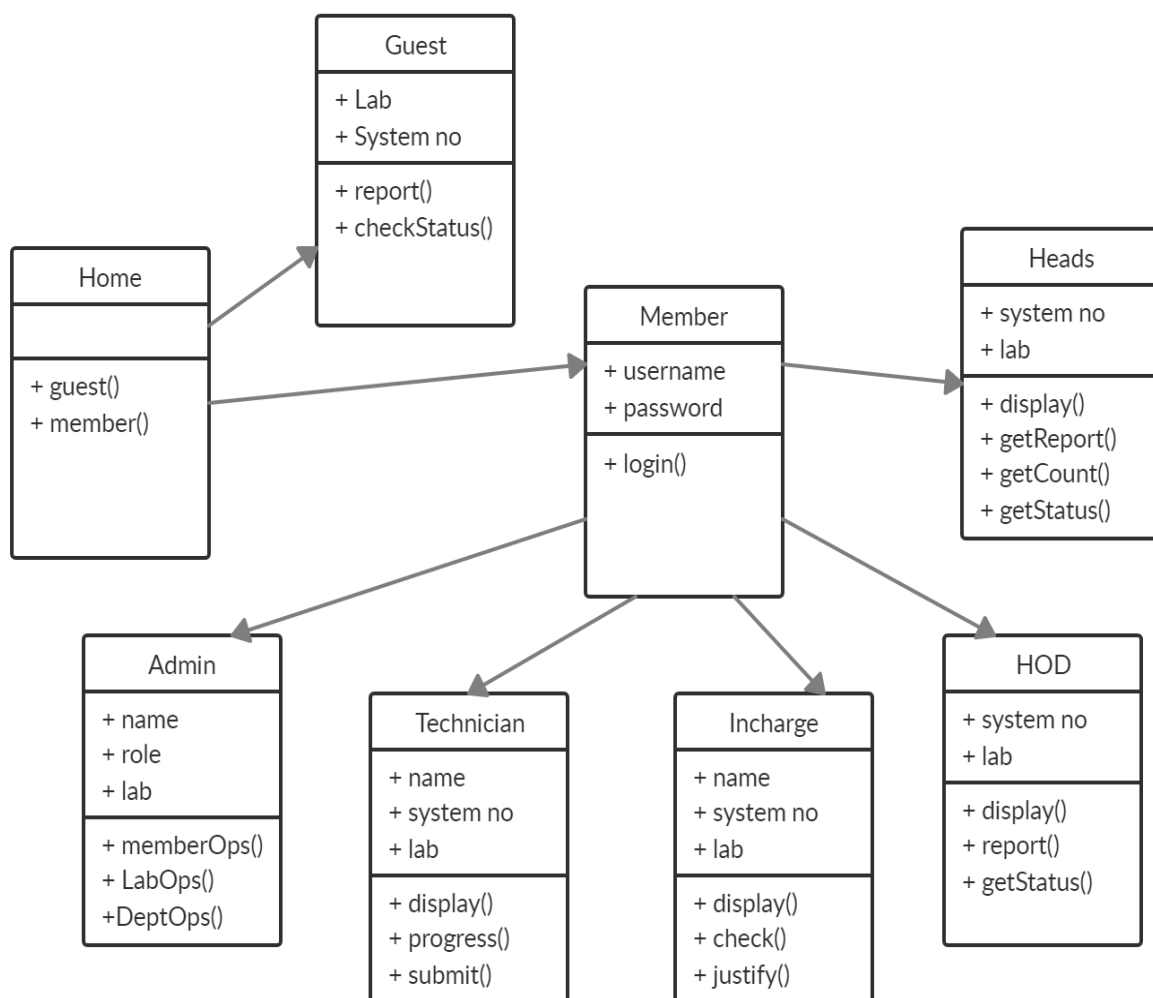


Fig 3.4.3.1 : Class diagram for Hardware Maintenance System

3.4.4. Sequence Diagram:

Sequence diagrams in UML shows how object interact with each other and the order those interactions occur. It's important to note that they show the interactions for a particular scenario. The processes are represented vertically and interactions are show as arrows. This article explains the purpose and the basics of Sequence diagrams.

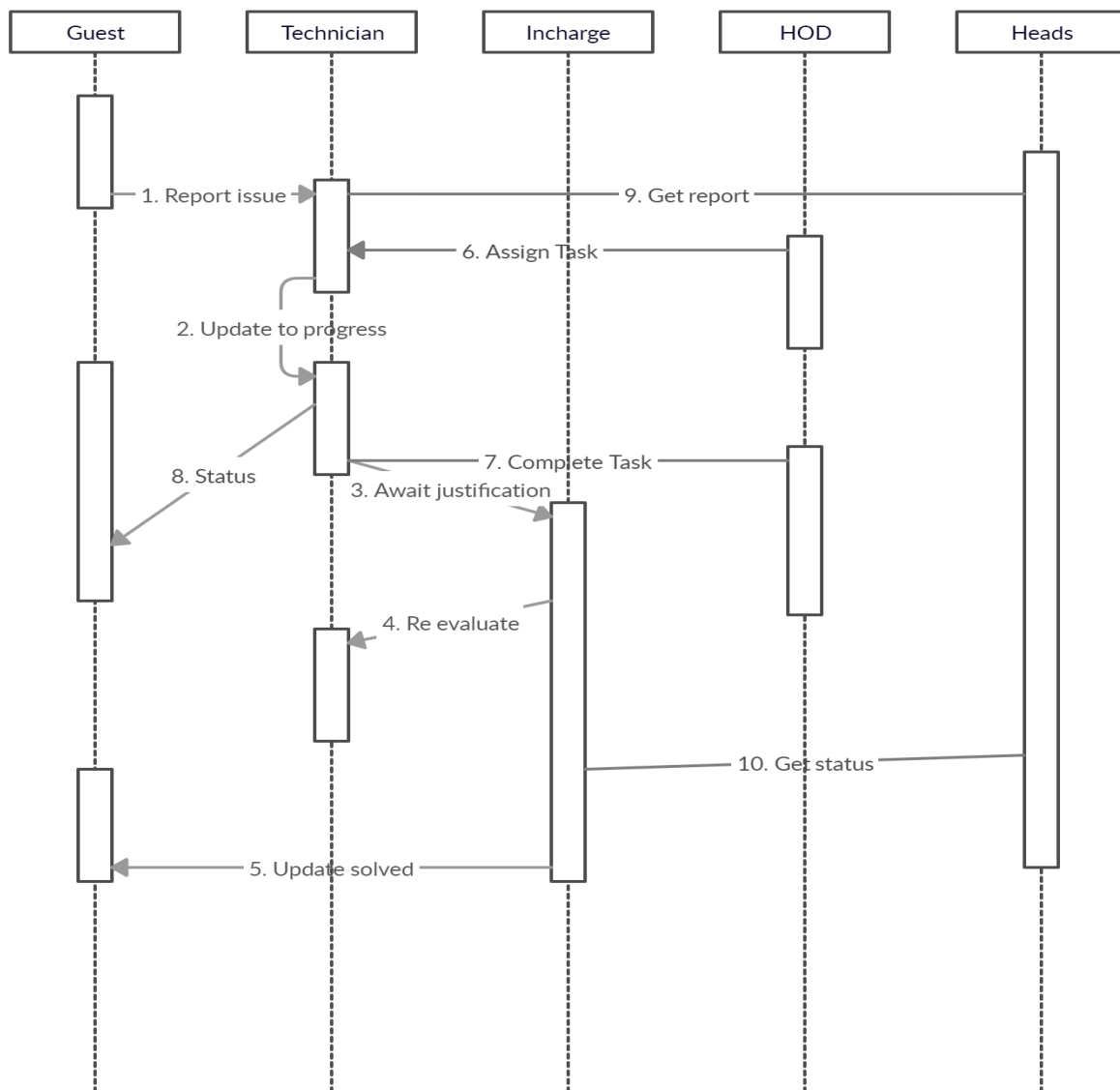


Fig 3.4.4.1 : Sequence diagram for Hardware Maintenance System

The above figure shows the sequential flow of actions performed in the project

CHAPTER-4

IMPLEMENTATION

4. IMPLEMENTATION:

Implementation is the process of having systems personnel check out and put new equipment into use, install the app. Depending on the size of the organization that will be involved in using the application and the risk associated with its use. Systems developers may choose to test the operation in only one area of the firm, say in one department or with only one or two persons. Sometimes they will run the old and new systems together to compare the result. In still other situation, developers will stop using the old system one-day and begin using the new one the next. As we will see, each implementation strategy has its merits, depending on the business situation in which it is considered. Regardless of the implementation strategy used, developers strive to ensure that the system's initial use is trouble-free.

Once installed, application is often used for many years. However, both the organization and the users will change, and the environment will be different over weeks and months. Therefore, the application will undoubtedly have to be maintained; modification and changes will be made to the software, files, or procedures to meet emerging user requirements. Since organization systems and the business environment undergo continual change, the information systems should keep pace. In this sense, implementation is ongoing process.

The characteristics required by a design language are:

- A fixed system of keywords that provide for all structured constructs data declaration and modularity characteristics.
- A free syntax of natural language that describes processing features.
- Data declaration facilities that should include both simple and complex data structures.
- Subprogram definition and calling techniques that support various nodes of interface description.

4.1. TOOLS USED:

Angular:

Angular is a platform and framework for building single-page client applications using HTML and TypeScript. Angular is written in TypeScript. It implements core and optional functionality as a set of TypeScript libraries that you import into your apps.

The framework is the brainchild of Google engineers, Misko Hevery and Adam Abrons. Google officially released the first version, AngularJS, in 2012, and has been maintaining it ever since. Before the release of AngularJS, there were other ways to create dynamic pages. However, they were not as convenient as the framework. AngularJS uses the Model-View-Controller (MVC) architecture, which is used in web app development.

This type of architecture consists of:

- **Model** - data structure that manages information and receives input from the controller
- **View** - the representation of information
- **Controller** - responds to input and interacts with the model

Advantages:

- Two way data binding
- Directives
- Code Structure
- Native Development
- Code splitting
- Powerful templates
- Angular CLI
- Animation support

Introduction to Modules

Angular apps are modular and Angular has its own modularity system called *NgModules*. NgModules are containers for a cohesive block of code dedicated to an application domain, a workflow, or a closely related set of capabilities. They can contain components, service providers, and other code files. They can import functionality that is exported from other NgModules, and export selected functionality to other NgModules.

Every Angular app has at least one NgModule class, the *root module*, which is conventionally named AppModule and resides in a file named app.module.ts. You launch your app by *bootstrapping* the root NgModule.

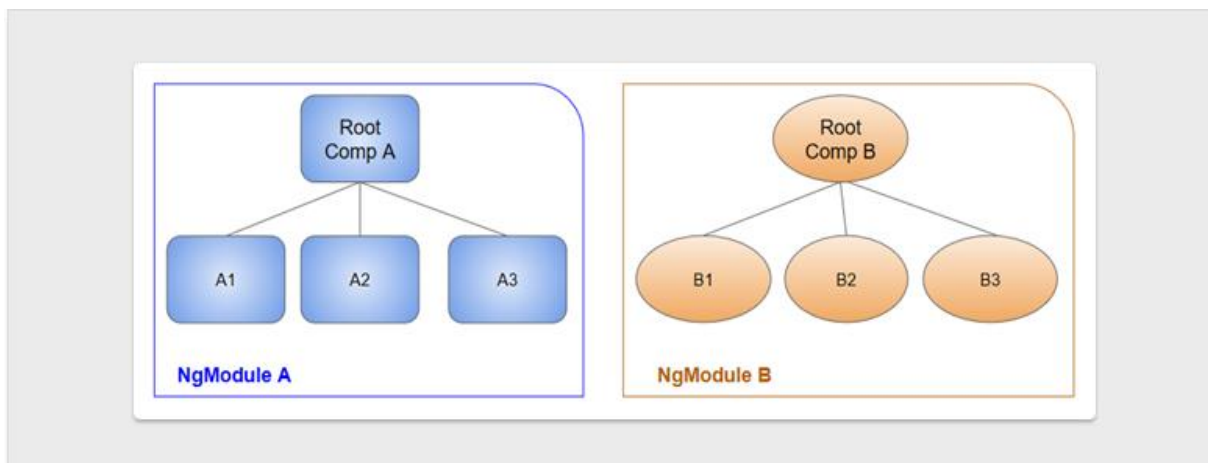


Fig 4.1.1 : Module Architecture

Module Metadata:

An NgModule is defined by a class decorated with @NgModule(). The most important properties are as follows.

- **declarations:** The components, *directives*, and *pipes* that belong to this NgModule.
- **exports:** The subset of declarations that should be visible and usable in the component templates of other NgModules.
- **imports:** Other modules whose exported classes are needed by component templates declared in this NgModule.
- **providers:** Creators of services that this NgModule contributes to the global collection of services; they become accessible in all parts of the app.
- **bootstrap:** The main application view, called the *root component*, which hosts all other app views. Only the root NgModule should set the bootstrap property.

Introduction to Components:

A *component* controls a patch of screen called a *view*. Angular creates, updates, and destroys components as the user moves through the application. Your app can take action at each moment in this lifecycle through optional lifecycle hooks, like `ngOnInit()`.

The component acquires the heroes from a service, which is a TypeScript parameter property on the constructor. The service is provided to the component through the dependency injection system.

Component Metadata:

The `@Component` decorator identifies the class immediately below it as a component class, and specifies its metadata. The metadata for a component tells Angular where to get the major building blocks that it needs to create and present the component and its view. In particular, it associates a *template* with the component, either directly with inline code, or by reference. Together, the component and its template describe a *view*. In addition to containing or pointing to the template, the `@Component` metadata configures, for example, how the component can be referenced in HTML and what services it requires.

This example shows some of the most useful `@Component` configuration options:

- **selector:** A CSS selector that tells Angular to create and insert an instance of this component wherever it finds the corresponding tag in template HTML. For example, if an app's HTML contains `<app-hero-list></app-hero-list>`, then Angular inserts an instance of the `HeroListComponent` view between those tags.
- **templateUrl:** The module-relative address of this component's HTML template. Alternatively, you can provide the HTML template inline, as the value of the `template` property. This template defines the component's host view.
- **providers:** An array of providers for services that the component requires. In the example, this tells Angular how to provide the `HeroService` instance that the component's constructor uses to get the list of heroes to display.

Setting up Angular Workspace:

To install angular you need both node.js and npm (node package manager) in your system.

Step 1 : Install the Angular CLI

To install angular globally you can use command:

```
npm install -g @angular/cli
```

Step 2 : Create a workspace and initial application

To create a new workspace and a initial starter app use command:

```
ng new project
```

Step 3 : Run the application

```
cd project
```

```
ng serve
```

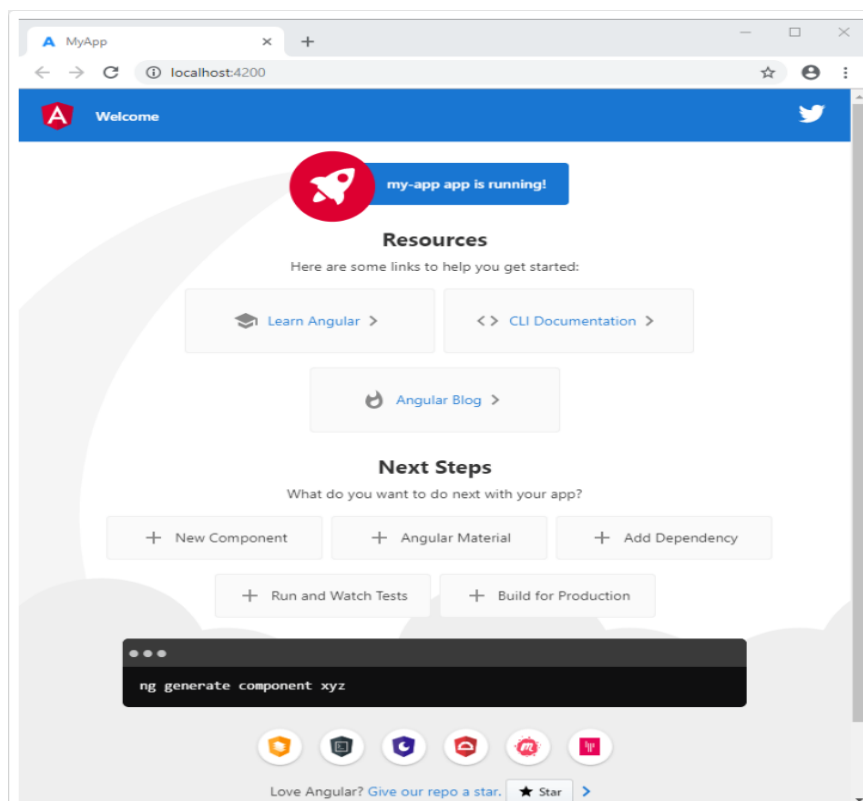


Fig 4.1.2 : Starter App

Node.js

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

Features:

- **Asynchronous and Event Driven** – All APIs of Node.js library are asynchronous, that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call.
- **Very Fast** – Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.
- **Single Threaded but Highly Scalable** – Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.
- **No Buffering** – Node.js applications never buffer any data. These applications simply output the data in chunks.

Installation:

Node.js is a runtime environment. It can be installed using the following link.

<https://nodejs.org/dist/v12.16.3/node-v12.16.3-x64.msi>

Express.js:

ExpressJS is a web application framework that provides you with a simple API to build websites, web apps and back ends. With ExpressJS, you need not worry about low level protocols, processes, etc.

The initial commit for Express was made on June 26, 2009 by TJ Holowaychuk and 660 commits later version 0.0.1 was released on January 2, 2010. The two main contributors at that time were TJ and Ciaron Jessup. At the time of the first release the framework was described as per the readme.md on github

Express provides a minimal interface to build our applications. It provides us the tools that are required to build our app. It is flexible as there are numerous modules available on **npm**, which can be directly plugged into Express.

ExpressJS is a prebuilt NodeJS framework that can help you in creating server-side web applications faster and smarter. Simplicity, minimalism, flexibility, scalability are some of its characteristics and since it is made in NodeJS itself, it inherits its performance as well.

In short, ExpressJS did for NodeJS what Bootstrap did for HTML/CSS and responsive web design. It made coding in NodeJS a piece of cake and gave programmers some additional features to extend their server-side coding. ExpressJS is hands down the most famous NodeJS framework- so much so that when most people talk about NodeJS they surely mean NodeJS+ExpressJS.

It is worth noting that NodeJS alone is not rocket science or anything. The only point we are trying to make here is that you can cut down your programming time in half through ExpressJS. Also, since NodeJS and ExpressJS are written in JavaScript, a very easy language to learn and manipulate, the framework is highly scalable and flexible. You can build websites, web application, and even **cross platform compatible mobile apps**.

Express has the biggest community not only out of the three frameworks compared here but out of all the web application frameworks for Node.js. It is the most matured framework out of the three, with almost 5 years of development behind it and now has StrongLoop taking control of the repository. It offers a simple way to get a server up and running and promotes code reuse with its built in router

Pug:

Pug (earlier known as Jade) is a terse language for writing HTML templates. It is one of the most popular template language used with Express.

- Produces HTML
- Supports dynamic code
- Supports reusability (DRY)

Node Package Manager(npm):

npm is the package manager for node. The npm Registry is a public collection of packages of open-source code for Node.js, front-end web apps, mobile apps, robots, routers, and countless other needs of the JavaScript community. npm allows us to access all these packages and install them locally. You can browse through the list of packages available on npm at [npmJS](https://www.npmjs.com/).

How to use npm?

There are two ways to install a package using npm: globally and locally.

- **Globally** – This method is generally used to install development tools and CLI based packages. To install a package globally, use the following code.

```
npm install -g <package-name>
```

- **Locally** – This method is generally used to install frameworks and libraries. A locally installed package can be used only within the directory it is installed. To install a package locally, use the same command as above without the **-g** flag.

```
npm install <package-name>
```


MongoDB:

MongoDB is a document-oriented NoSQL database used for high volume data storage. Instead of using tables and rows as in the traditional relational databases, MongoDB makes use of collections and documents. Documents consist of key-value pairs which are the basic unit of data in MongoDB. Collections contain sets of documents and function which is the equivalent of relational database tables. MongoDB is a database which came into light around.

With the rise in data all around the world, there has been an observable and increasing interest surrounding the wave of the non-relational database, also known as 'NoSQL'. Businesses and organizations are seeking new methods to manage the flood of data and are drawn toward the alternate database management tools and systems that are different from the traditional relational database systems. Here comes MongoDB into the picture.

MongoDB Features:

- Each database contains collections which in turn contains documents. Each document can be different with a varying number of fields. The size and content of each document can be different from each other.
- The document structure is more in line with how developers construct their classes and objects in their respective programming languages. Developers will often say that their classes are not rows and columns but have a clear structure with key-value pairs.
- The rows (or documents as called in MongoDB) doesn't need to have a schema defined beforehand. Instead, the fields can be created on the fly.
- The data model available within MongoDB allows you to represent hierarchical relationships, to store arrays, and other more complex structures more easily.

Key Components of MongoDB Architecture:

Below are a few of the common terms used in MongoDB

1. **_id** – This is a field required in every MongoDB document. The _id field represents a unique value in the MongoDB document. The _id field is like the document's primary key. If you create a new document without an _id field, MongoDB will automatically create the field. So, for example, if we see the example of the above customer table, Mongo DB will add a 24 digit unique identifier to each document in the collection.
2. **Collection** – This is a grouping of MongoDB documents. A collection is the equivalent of a table which is created in any other RDMS such as Oracle or MS SQL. A collection exists within a single database. As seen from the introduction collections don't enforce any sort of structure.
3. **Cursor** – This is a pointer to the result set of a query. Clients can iterate through a cursor to retrieve results.
4. **Database** – This is a container for collections like in RDMS wherein it is a container for tables. Each database gets its own set of files on the file system. A MongoDB server can store multiple databases.
5. **Document** - A record in a MongoDB collection is basically called a document. The document, in turn, will consist of field name and values.
6. **Field** - A name-value pair in a document. A document has zero or more fields. Fields are analogous to columns in relational databases.
7. **JSON** – This is known as JavaScript Object Notation. This is a human-readable, plain text format for expressing structured data. JSON is currently supported in many programming languages.

_id	CustomerID	CustomerName	OrderID
563479cc8a8a4246bd27d784	11	Guru99	111
563479cc7a8a4246bd47d784	22	Trevor Smith	222
563479cc9a8a4246bd57d784	33	Nicole	333

Fig 4.1.3 : MongoDB Example

Why use MongoDB?

Below are the few of the reasons as to why one should start using MongoDB

1. **Document-oriented** – Since MongoDB is a NoSQL type database, instead of having data in a relational type format, it stores the data in documents. This makes MongoDB very flexible and adaptable to real business world situation and requirements.
2. **Ad hoc queries** - MongoDB supports searching by field, range queries, and regular expression searches. Queries can be made to return specific fields within documents.
3. **Indexing** - Indexes can be created to improve the performance of searches within MongoDB. Any field in a MongoDB document can be indexed.
4. **Replication** - MongoDB can provide high availability with replica sets. A replica set consists of two or more mongo DB instances. Each replica set member may act in the role of the primary or secondary replica at any time. The primary replica is the main server which interacts with the client and performs all the read/write operations. The Secondary replicas maintain a copy of the data of the primary using built-in replication. When a primary replica fails, the replica set automatically switches over to the secondary and then it becomes the primary server.
5. **Load balancing** - MongoDB uses the concept of sharding to scale horizontally by splitting data across multiple MongoDB instances. MongoDB can run over multiple servers, balancing the load and/or duplicating data to keep the system up and running in case of hardware failure.

Data modelling in MongoDB:

As we have seen from the Introduction section, the data in MongoDB has a flexible schema. Unlike in SQL databases, where you must have a table's schema declared before inserting data, MongoDB's collections do not enforce document structure. This sort of flexibility is what makes MongoDB so powerful.

When modeling data in Mongo, keep the following things in mind

1. **What are the needs of the application** – Look at the business needs of the application and see what data and the type of data needed for the application. Based on this, ensure that the structure of the document is decided accordingly.
2. **What are data retrieval patterns** – If you foresee a heavy query usage then consider the use of indexes in your data model to improve the efficiency of queries.
3. **Are frequent inserts, updates and removals happening in the database?** Reconsider the use of indexes or incorporate sharding if required in your data modeling design to improve the efficiency of your overall MongoDB environment.

Difference between MongoDB & RDBMS

Below are some of the key term differences between MongoDB and RDBMS

RDBMS	MongoDB	Difference
Table	Collection	In RDBMS, the table contains the columns and rows which are used to store the data whereas, in MongoDB, this same structure is known as a collection. The collection contains documents which in turn contains Fields, which in turn are key-value pairs.
Row	Document	In RDBMS, the row represents a single, implicitly structured data item in a table. In MongoDB, the data is stored in documents.
Column	Field	In RDBMS, the column denotes a set of data values. These in MongoDB are known as Fields.
Joins	Embedded documents	In RDBMS, data is sometimes spread across various tables and in order to show a complete view of all data, a join is sometimes formed across tables to get the data. In MongoDB, the data is normally stored in a single collection, but separated by using Embedded documents. So there is no concept of joins in MongoDB.

Fig 4.1.4 : MongoDB vs RDBMS

4.2 Database Collections:

1. Members

```
{  
  _id : ObjectID  
  name : String  
  position : String  
  assigned : Boolean  
  password : String  
}
```

2. Departments

```
{  
  _id : Object  
  name : String  
  hod : String  
}
```

3. DeptPeri

```
{  
  _id : ObjectID  
  name : String  
  dept : String  
  type : String  
  assigned_to : String  
  message : String  
  status : String  
}
```

4. Labs

```
{  
  _id : ObjectID  
  name : String  
  incharge : String  
  technician : String  
  count : Int32  
  working : Int32  
  not : Int32  
}
```

5. Problems

```
{  
  _id : ObjectID  
  token : String  
  system : String  
  lab : String  
  assigned_to : String  
  status : String  
  problem : String  
  date : String  
  solved : String  
}
```

6. Systems

```
{  
  _id : ObjectID  
  name : String  
  lab : String  
  status : String  
}
```

4.3 PSEUDO CODE:

Backend code: (Node.js and Express.js)

```
var express = require('express')
var MongoClient = require('mongodb').MongoClient;
var M_url = "mongodb://localhost:27017/";
var app = express()
var cors = require('cors')
app.use(cors())

app.get("/getmembers",(req,res)=>{
  MongoClient.connect(M_url,function(err,db){
    if (err) throw err;
    dbo = db.db("HMS");
    dbo.collection("Members").find({}).project({_id:0,position:0,assigned:0}).toArray(function(err,res2){
      if (err) throw err;
      res.json(res2)
      db.close()
    })
  })
});

app.get("/auth/:pos/:user/:pass",(req,res)=>{
  var pos = ""
  if(req.params.pos=="technician"){
    pos = "Technician"
  }
  else if(req.params.pos=="administrator"){
    pos = "Administrator"
  }
  else if(req.params.pos == "incharge"){
    pos = "Incharge"
  }
  else if(req.params.pos == "admin"){
    pos = "admin"
  }
});
```

```

    }
    else if(req.params.pos == "hod"){
        pos = "HOD"
    }
    MongoClient.connect(M_url,function(err,db){
        if (err) throw err;
        dbo = db.db("HMS");
        console.log(pos+req.params.user+req.params.pass)
        dbo.collection("Members").find({$and:[{"position":pos,"name":req.params.user,"password":req.params.pass}]}).count(function(err,count){
            if (err) throw err;
            if(count==1){
                res.json({status:"success"})
            }
            else{
                res.json({status:"error"})
            }
            console.log(count)
            db.close()
        })
    })
})

app.get("/getNames/:pp",(req,res)=>{
    var pos = ""
    if(req.params.pp=="technician"){
        pos = "Technician"
    }
    else if(req.params.pp=="administrator"){
        pos = "Administrator"
    }
    else if(req.params.pp == "incharge"){
        pos = "Incharge"
    }
    else if(req.params.pp == "admin"){
        pos = "admin"
    }
}

```



```

else if(req.params.pp == "hod"){
    pos = "HOD"
}
mongoClient.connect(M_url,function(err,db){
    if (err) throw err;
    dbo = db.db("HMS");
    dbo.collection("Members").find({ "position":pos }).project({ _id:0,position:0,assigned:0 }).t
oArray(function(err,res2){
    if (err) throw err;
    console.log(res2)
    res.json(res2)
    db.close()
    })
    })
});

```

```

app.get("/getNotAsstech",(req,res)=>{
    mongoClient.connect(M_url,function(err,db){
        if (err) throw err;
        dbo = db.db("HMS");
        dbo.collection("Members").find({ "assigned":false,"position":"Technician" }).project({ _id:
0,position:0,assigned:0 }).toArray(function(err,res2){
            if (err) throw err;
            res.json(res2)
            db.close()
            })
        })
    });

```

```

app.get("/getNotAssInch",(req,res)=>{
    mongoClient.connect(M_url,function(err,db){
        if (err) throw err;
        dbo = db.db("HMS");
        dbo.collection("Members").find({ "assigned":false,"position":"Incharge" }).project({ _id:0,
position:0,assigned:0 }).toArray(function(err,res2){
            if (err) throw err;
            res.json(res2)

```

```

        db.close()
    })
    });

app.get("/getlabslist",(req,res)=>{
    mongoClient.connect(M_url,function(err,db){
        if (err) throw err;
        dbo = db.db("HMS");
        dbo.collection("Labs").find({}).project({_id:0,incharge:0,technician:0,count:0,working:0,
not:0}).toArray(function(err,res2){
            if (err) throw err;
            res.json(res2)
            db.close()
        })
    })
    });

app.get("/getNotAssHOD",(req,res)=>{
    mongoClient.connect(M_url,function(err,db){
        if (err) throw err;
        dbo = db.db("HMS");
        dbo.collection("Members").find({"assigned":false,"position":"HOD"}).project({_id:0,pos
ition:0,assigned:0}).toArray(function(err,res2){
            if (err) throw err;
            res.json(res2)
            db.close()
        })
    })
    });

app.get("/getDeptlist",(req,res)=>{
    mongoClient.connect(M_url,function(err,db){
        if (err) throw err;
        dbo = db.db("HMS");
        dbo.collection("Departments").find({}).project({_id:0,hod:0}).toArray(function(err,res2)
{

```

```

        if (err) throw err;
        res.json(res2)
        db.close()
    })
})
});

```

```

app.get("/getSystemslist/:lab",(req,res)=>{
    mongoClient.connect(M_url,function(err,db){
        if (err) throw err;
        dbo = db.db("HMS");
        dbo.collection("Systems").find({lab:req.params.lab,status:"working"}).project({_id:0,lab:
0,status:0}).toArray(function(err,res2){
            if (err) throw err;
            res.json(res2)
            db.close()
        })
    })
});

```

```

app.get("/insertMember/:name/:position/:pass",(req,res)=>{
    mongoClient.connect(M_url,function(err,db){
        if (err) throw err;
        dbo = db.db("HMS");
        var ins = {name : req.params.name,position : req.params.position, assigned : false, passw
ord : req.params.pass};
        dbo.collection("Members").insertOne(ins,function(err,res2){
            if(err) throw err;
            res.json([ {mes:"success" }])
            db.close();
        })
    })
});

```

```

app.get("/getMD/:name",(req,res)=>{
    mongoClient.connect(M_url,function(err,db){
        if (err) throw err;

```

```

        dbo = db.db("HMS");
        dbo.collection("Members").find({name:req.params.name}).project({_id:0,assigned:0}).toArray(function(err,res2){
            if (err) throw err;
            res.json(res2)
            db.close()
        })
    })
})

```

```

app.get("/updateMember/:ename/:cname/:position/:pass",(req,res)=>{
    mongoClient.connect(M_url,function(err,db){
        if (err) throw err;
        dbo = db.db("HMS");
        var qu = { name : req.params.ename }
        if(req.params.pass.length>0)
            var ins = { $set : { name : req.params.cname,position : req.params.position } };
        else
            var ins = { $set : { name : req.params.cname,position : req.params.position,password : req.params.pass } };
        dbo.collection("Members").updateOne(qu,ins,function(err,res2){
            if(err) throw err;
            res.json([ { mes:"success" } ])
            db.close();
        })
    })
});

```

```

app.get("/deleteMember/:name",(req,res)=>{
    mongoClient.connect(M_url,function(err,db){
        if (err) throw err;
        dbo = db.db("HMS");
        var qu = { name : req.params.name }
        dbo.collection("Members").find(qu).project({_id:0,name:0,position:0,password:0}).toArray(function(err,res3){
            console.log(res3[0].assigned)

```

```

if(!res3[0].assigned){
    dbo.collection("Members").deleteOne(qu,function(err,res2){
        if(err) throw err;
        res.json([ {mes:"success"} ])
        db.close();
    })
}
else{
    res.json([ {mes:"response"} ])
    db.close();
}
})
});

```

```

app.get("/addLab/:lname/:inch/:tech/:num",(req,res)=>{
    mongoClient.connect(M_url,function(err,db){
        if (err) throw err;
        dbo = db.db("HMS");
        var qu = { name : req.params.lname,incharge:req.params.inch,technician:req.params.tech,
count:req.params.num,working:req.params.num,not:0};
        dbo.collection("Labs").insertOne(qu,function(err,res2){
            if(err) throw err;

        });
        var data = Array();
        for(var i=1;i<=req.params.num;i++){
            var sys = { name : req.params.lname+"sys"+i,lab : req.params.lname, status : "working
"}
            data.push(sys);
        }
        var dinch = { name : req.params.inch}
        var dtech = { name : req.params.tech}
        var ins = { $set : { assigned : true } };
        dbo.collection("Members").updateOne(dinch,ins,function(err,rr){
            if(err) throw err;
        })
    })

```

```

    dbo.collection("Members").updateOne(dtech,ins,function(err,rr){
        if(err) throw err;
    })
    dbo.collection("Systems").insertMany(data,function(err,res3){
        if(err) throw err;
        res.json([ {mes:"success" }])
        db.close()
    })

    });

app.get("/updateLabs/:lab/:inch/:tech",(req,res)=>{
    mongoClient.connect(M_url,function(err,db){
        if (err) throw err;
        dbo = db.db("HMS");
        var pinch = ""
        var ptech = ""
        var inch = (req.params.inch!=null)?req.params.inch:"null"
        var tech = (req.params.tech!=null)?req.params.tech:"null"

        dbo.collection("Labs").find({ name:req.params.lab}).project({ _id:0,name:0,count:0,worki
ng:0,not:0}).toArray(function(err,res2){
            if(err) throw err;
            this.pinch = (inch!="null"?res2[0].incharge:"null"
            this.ptech = (tech!="null"?res2[0].technician:"null"
            var cond2 = { $or : [{name:res2[0].incharge},{name:res2[0].technician}]}
            var ins2 = { $set : {assigned : false} }
            console.log(cond2)
            dbo.collection("Members").updateMany(cond2,ins2,function(err,res4){
                if(err) throw err;
            })
        })
        dbo.collection("Labs").updateOne({ name:req.params.lab},{ $set : { incharge : inch,technic
ian : tech} },function (err,rrr){
            if(err) throw err;

```

```

    })
    var cond = { $or : [{name:inch},{name:tech}]}
    var ins = { $set : {assigned : true}}
    dbo.collection("Members").updateMany(cond,ins,function(err,res3){
        if(err) throw err;
        dbo.collection("Problems").updateMany({lab:req.params.lab},{ $set:{ assigned_to:tech
    }},function(err,re4){
        if(err) throw err;
        res.json([{mes:"success"}])
    })
    db.close()
})

})
});

app.get("/deleteLab/:name",(req,res)=>{
    mongoClient.connect(M_url,function(err,db){
        if (err) throw err;
        dbo = db.db("HMS");
        dbo.collection("Labs").find({name:req.params.name}).project({_id:0,name:0,count:0,working:0,not:0}).toArray(function(err,res2){
            if(err) throw err;
            var cond2 = { $or : [{name:res2[0].incharge},{name:res2[0].technician}]}
            var ins2 = { $set : {assigned : false}}
            console.log(cond2)
            dbo.collection("Members").updateMany(cond2,ins2,function(err,res4){
                if(err) throw err;
            })
        })
        var qu = { name : req.params.name}
        dbo.collection("Labs").deleteOne(qu,function(err,res2){
            if(err) throw err;
        })
        dbo.collection("Systems").deleteMany({lab:req.params.name},function(err,res3){
            if(err) throw err;
            res.json([{mes:"success"}])
        })
    })
})

```

```

        db.close();
    })
    });

app.get("/insertDept/:name/:hod",(req,res)=>{
    mongoClient.connect(M_url,function(err,db){
        if (err) throw err;
        dbo = db.db("HMS");
        var ins = { name : req.params.name,hod : req.params.hod,};
        dbo.collection("Departments").insertOne(ins,function(err,res2){
            if(err) throw err;

        })
        dbo.collection("Members").updateOne({ name:req.params.hod,position:"HOD" },{ $set: { as
signed:true } },function(err,res3){
            if(err) throw err;
            res.json([ {mes:"success" }])
            db.close();
        })
    });

app.get("/modDept/:dept/:chod",(req,res)=>{
    mongoClient.connect(M_url,function(err,db){
        if (err) throw err;
        dbo = db.db("HMS");
        dbo.collection("Departments").find({ name:req.params.dept}).project({ _id:0,name:0}).to
Array(function(err,res3){
            if(err) throw err;
            dbo.collection("Members").updateOne({ name:res3[0].hod,position:"HOD" },{ $set: { ass
igned:false } },function(err,res4){
                if(err) throw err;
            })
            dbo.collection("Members").updateOne({ name:req.params.chod,position:"HOD" },{ $set
:{ assigned:true } },function(err,res4){
                if(err) throw err;

```



```

        })
    })
    dbo.collection("Departments").updateOne({ name:req.params.dept },{ $set: { hod:req.params.chod } },function(err,res3){
        if(err) throw err;
        res.json([ { mes:"success" } ])
        db.close();
    })
    })
    });

```

```

app.get("/addPeriDept/:dept/:name/:type",(req,res)=>{
    mongoClient.connect(M_url,function(err,db){
        if(err) throw err;
        dbo = db.db("HMS");
        dbo.collection("DeptPeri").insertOne({ name:req.params.name,dept:req.params.dept,type:
req.params.type,assigned_to:"",mes:"",status:"working" },function(err,res2){
            if(err) throw err;
            res.json([ { mes:"success" } ])
            db.close()
        })
    })
    });

```

```

app.get("/deleteDept/:dept",(req,res)=>{
    mongoClient.connect(M_url,function(err,db){
        if (err) throw err;
        dbo = db.db("HMS");
        dbo.collection("Departments").find({ name:req.params.dept }).project({ _id:0,name:0 }).to
Array(function(err,res3){
            if(err) throw err;
            dbo.collection("Members").updateOne({ name:res3[0].hod,position:"HOD" }, { $set: { ass
igned:false } },function(err,res4){
                if(err) throw err;
            })
        })
        var qu = { name : req.params.dept}

```

```

        dbo.collection("Departments").deleteMany(qu,function(err,res2){
            if(err) throw err;
            res.json([ {mes:"success" }])
            db.close();
        })
    })
});

app.get("/insertProblem/:lab/:sys/:mes",(req,res)=>{
    mongoClient.connect(M_url,function(err,db){
        if (err) throw err;
        dbo = db.db("HMS");
        dbo.collection("Problems").find({ }).count(function(err,count){
            if(err) throw err;
            var dtk = count +1
            dbo.collection("Labs").find({ name:req.params.lab }).project({ _id:0,name:0,incharge:0,
count:0}).toArray(function(err,res2){
                if(err) throw err;
                console.log(res2)
                var tech = res2[0].technician
                var dworking = res2[0].working-1
                var dnott = res2[0].not+1
                var today = new Date()

                var dd = today.getDate()
                var mm = today.getMonth()+1
                var yyyy = today.getFullYear()

                if(dd<10) { dd='0'+dd;}

                if(mm<10) { mm='0'+mm; }

                today = mm+'/'+dd+'/'+yyyy

                dbo.collection("Systems").updateOne({ name:req.params.sys,lab:req.params.lab },{$s
et:{status:"reported" } },function(err,res6){
                    if(err) throw err;

```

```

    })

    dbo.collection("Labs").updateOne({ name:req.params.lab},{ $set:{ working:dworking
,not:dnott}},function(err,res3){
        if(err) throw err;
        var ins = {tk: "tk"+dtk,system:req.params.sys,lab:req.params.lab,assigned_to:tech,
status:"reported",problem:req.params.mes,date:today,solved:""}
        dbo.collection("Problems").insertOne(ins,function(err,res5){
            if(err) throw err;
            res.json([{ mes:"success",id:"tk"+dtk}])
            db.close();
        })
    })
})
})
});

```

```

app.get("/getStatus/:token",(req,res)=>{
    mongoClient.connect(M_url,function(err,db){
        if (err) throw err;
        dbo = db.db("HMS");
        dbo.collection("Problems").find({tk:req.params.token}).project({_id:0,tk:0}).toArray(function(err,res2){
            if (err) throw err;
            res2.push({ mes:"success"})
            res.json(res2)
            db.close()
        })
    })
});

```

```

app.get("/getTotalSystemCount/:inch",(req,res)=>{
    mongoClient.connect(M_url,function(err,db){
        if (err) throw err;
        dbo = db.db("HMS");
        dbo.collection("Labs").find({ incharge:req.params.inch}).project({_id:0,technician:0,coun

```

```

t:0,working:0,not:0}).toArray(function(err,res2){
    if(err) throw err;
    var labdb = res2[0].name
    dbo.collection("Systems").find({lab:labdb}).project({_id:0}).count(function(err,count)
{
    if(err) throw err;
    res.json(count)
    console.log(count)
    db.close()
    })
    })
    });

app.get("/getSystemsIncharge/:p/:inch",(req,res)=>{
    mongoClient.connect(M_url,function(err,db){
        if (err) throw err;
        dbo = db.db("HMS");
        dbo.collection("Labs").find({incharge:req.params.inch}).project({_id:0,technician:0,coun
t:0,working:0,not:0}).toArray(function(err,res2){
            if(err) throw err;
            var labdb = res2[0].name
            if(req.params.p==1){
                dbo.collection("Systems").find({lab:labdb}).project({_id:0}).limit(10).toArray(func
tion(err,res3){
                    if(err) throw err;
                    res3.push({mes:"success"})
                    res.json(res3)
                    db.close()
                })
            }
            else{
                dbo.collection("Systems").find({lab:labdb}).project({_id:0}).skip((req.params.p-
1)*10).limit(10).toArray(function(err,res3){
                    if(err) throw err;
                    res3.push({mes:"success"})
                    res.json(res3)

```

```

        db.close()
    })
}

    })
})
});

app.get("/getAwaitSystemCount/:inch",(req,res)=>{
    mongoClient.connect(M_url,function(err,db){
        if (err) throw err;
        dbo = db.db("HMS");
        dbo.collection("Labs").find({ incharge:req.params.inch }).project({ _id:0,technician:0,count:0,working:0,not:0 }).toArray(function(err,res2){
            if(err) throw err;
            var labdb = res2[0].name
            dbo.collection("Systems").find({ lab:labdb,status:"awaiting" }).project({ _id:0 }).count(function(err,count){
                if(err) throw err;
                res.json(count)
                console.log(count)
                db.close()
            })
        })
    })
});

app.get("/getAwaitSystemsIncharge/:p/:inch",(req,res)=>{
    mongoClient.connect(M_url,function(err,db){
        if (err) throw err;
        dbo = db.db("HMS");
        dbo.collection("Labs").find({ incharge:req.params.inch }).project({ _id:0,technician:0,count:0,working:0,not:0 }).toArray(function(err,res2){
            if(err) throw err;
            var labdb = res2[0].name
            if(req.params.p==1){
                dbo.collection("Systems").find({ lab:labdb,status:"awaiting" }).project({ _id:0 }).limit(

```

```

10).toArray(function(err,res3){
    if(err) throw err;
    res3.push({ mes:"success"})
    res.json(res3)
    db.close()
  })
}
else{
  dbo.collection("Systems").find({lab:labdb,status:"awaiting"}).project({_id:0}).skip((
req.params.p-1)*10).limit(10).toArray(function(err,res3){
    if(err) throw err;
    res3.push({ mes:"success"})
    res.json(res3)
    db.close()
  })
}

  ))
})
});

```

```

app.get("/updateStatusSys/:name/:status",(req,res)=>{
  mongoClient.connect(M_url,function(err,db){
    if (err) throw err;
    dbo = db.db("HMS");
    dbo.collection("Systems").updateOne({ name:req.params.name},{ $set:{ status:req.params.
status} },function(err,res2){
      if(err) throw err;
      console.log(req.params.name)
      console.log(req.params.status)
      dbo.collection("Problems").updateMany({ system:req.params.name},{ $set:{ status:req.p
arams.status} },function(err,res3){
        if(err) throw err;
        res.json([ {mes:"success"} ])
        db.close()
      })
    })
  })
});

```

```

    })
  });

app.get("/getTotalReportedSystemCount/:tech",(req,res)=>{
  mongoClient.connect(M_url,function(err,db){
    if (err) throw err;
    dbo = db.db("HMS");
    dbo.collection("Problems").find({ assigned_to:req.params.tech,status:"reported" }).count(function(err,count){
      if(err) throw err;
      res.json(count)
      console.log(count)
      db.close()
    })
  })
});

app.get("/getReportedSystemsTechnician/:p/:tech",(req,res)=>{
  mongoClient.connect(M_url,function(err,db){
    if (err) throw err;
    dbo = db.db("HMS");

    if(req.params.p==1){
      dbo.collection("Problems").find({ assigned_to:req.params.tech,status:"reported" }).project({ _id:0,assigned_to:0,status:0,solved:0 }).limit(10).toArray(function(err,res3){
        if(err) throw err;
        res3.push({ mes:"success" })
        res.json(res3)
        db.close()
      })
    }
    else{
      dbo.collection("Problems").find({ assigned_to:req.params.tech,status:"reported" }).project({ _id:0,assigned_to:0,status:0,solved:0 }).skip((req.params.p-1)*10).limit(10).toArray(function(err,res3){
        if(err) throw err;

```

```

        res3.push({ mes:"success"})
        res.json(res3)
        db.close()
    })
}
})
});

```

```

app.get("/updateStatusTech/:tk/:system/:status",(req,res)=>{
    mongoClient.connect(M_url,function(err,db){
        if (err) throw err;
        dbo = db.db("HMS");
        dbo.collection("Systems").updateOne({ name:req.params.system},{ $set:{ status:req.param
s.status } },function(err,res2){
            if(err) throw err;
            dbo.collection("Problems").updateOne({ tk:req.params.tk},{ $set:{ status:req.params.stat
us } },function(err,res3){
                if(err) throw err;
                res.json([ { mes:"success" } ])
                db.close()
            })
        })
    })
});

```

```

app.get("/getTotalSelectedSystemCount/:tech",(req,res)=>{
    mongoClient.connect(M_url,function(err,db){
        if (err) throw err;
        dbo = db.db("HMS");
        dbo.collection("Problems").find({ assigned_to:req.params.tech,status:"in progress"}).count
(function(err,count){
            if(err) throw err;
            res.json(count)
            console.log(count)
            db.close()
        })
    })
});

```



```

    })
  });

app.get("/getSelectedSystemsTechnician/:p/:tech",(req,res)=>{
  mongoClient.connect(M_url,function(err,db){
    if (err) throw err;
    dbo = db.db("HMS");

    if(req.params.p==1){
      dbo.collection("Problems").find({assigned_to:req.params.tech,status:"in progress"}).project({_id:0,assigned_to:0,status:0,solved:0}).limit(10).toArray(function(err,res3){
        if(err) throw err;
        res3.push({mes:"success"})
        res.json(res3)
        db.close()
      })
    }
    else{
      dbo.collection("Problems").find({assigned_to:req.params.tech,status:"in progress"}).project({_id:0,assigned_to:0,status:0,solved:0}).skip((req.params.p-1)*10).limit(10).toArray(function(err,res3){
        if(err) throw err;
        res3.push({mes:"success"})
        res.json(res3)
        db.close()
      })
    }
  })
});

```

```

app.get("/getTotalAwaitSystemCount/:tech",(req,res)=>{
  mongoClient.connect(M_url,function(err,db){
    if (err) throw err;
    dbo = db.db("HMS");
    dbo.collection("Problems").find({assigned_to:req.params.tech,status:"awaiting"}).count(function(err,count){
      if(err) throw err;

```

```

        res.json(count)
        console.log(count)
        db.close()
    })
})
});

app.get("/getAwaitSystemsTechnician/:p/:tech",(req,res)=>{
    mongoClient.connect(M_url,function(err,db){
        if (err) throw err;
        dbo = db.db("HMS");

        if(req.params.p==1){
            dbo.collection("Problems").find({ assigned_to:req.params.tech,status:"awaiting"}).project({_id:0,assigned_to:0,status:0,solved:0}).limit(10).toArray(function(err,res3){
                if(err) throw err;
                res3.push({ mes:"success"})
                res.json(res3)
                db.close()
            })
        }
        else{
            dbo.collection("Problems").find({ assigned_to:req.params.tech,status:"awaiting"}).project({_id:0,assigned_to:0,status:0,solved:0}).skip((req.params.p-1)*10).limit(10).toArray(function(err,res3){
                if(err) throw err;
                res3.push({ mes:"success"})
                res.json(res3)
                db.close()
            })
        }
    })
});

```

```

app.get("/getTotalSolvedSystemCount/:tech",(req,res)=>{
    mongoClient.connect(M_url,function(err,db){
        if (err) throw err;

```

```

    dbo = db.db("HMS");
    dbo.collection("Problems").find({ assigned_to:req.params.tech,status:"working" }).count(f
unction(err,count){
        if(err) throw err;
        res.json(count)
        console.log(count)
        db.close()
    })
    })
});

```

```

app.get("/getTotalSystemsTechnician/:p/:tech",(req,res)=>{
    mongoClient.connect(M_url,function(err,db){
        if (err) throw err;
        dbo = db.db("HMS");

        if(req.params.p==1){
            dbo.collection("Problems").find({ assigned_to:req.params.tech,status:"working" }).proje
ct({ _id:0,assigned_to:0,status:0,solved:0}).limit(10).toArray(function(err,res3){
                if(err) throw err;
                res3.push({ mes:"success"})
                res.json(res3)
                db.close()
            })
        }
        else{
            dbo.collection("Problems").find({ assigned_to:req.params.tech,status:"working" }).proje
ct({ _id:0,assigned_to:0,status:0,solved:0}).skip((req.params.p-
1)*10).limit(10).toArray(function(err,res3){
                if(err) throw err;
                res3.push({ mes:"success"})
                res.json(res3)
                db.close()
            })
        }
    })
});

```

```

app.get("/systemCounts",(req,res)=>{
  mongoClient.connect(M_url,function(err,db){
    if(err) throw err;
    dbo = db.db("HMS")
    dbo.collection("Labs").find({}).project({_id:0,}).toArray(function(err,res2){
      if(err) throw err;
      res.json(res2)
      db.close()
    })
  })
});

app.get("/allSolved",(req,res)=>{
  mongoClient.connect(M_url,function(err,db){
    if(err) throw err;
    dbo = db.db("HMS")
    dbo.collection("Problems").find({status:"working"}).count(function(err,count){
      if(err) throw err;
      res.json(count)
    })
  })
});

app.get("/getSolvedPages/:p",(req,res)=>{
  mongoClient.connect(M_url,function(err,db){
    if(err) throw err;
    dbo = db.db("HMS")

    if(req.params.p==1){
      dbo.collection("Problems").find({status:"working"}).project({_id:0,status:0}).limit(10)
      .toArray(function(err,res3){
        if(err) throw err;
        res3.push({mes:"success"})
        res.json(res3)
        db.close()
      })
    }
  })
});

```

```

        })
    }
    else{
        dbo.collection("Problems").find({status:"working"}).project({_id:0,status:0}).skip((req
.params.p-1)*10).limit(10).toArray(function(err,res3){
            if(err) throw err;
            res3.push({ mes:"success"})
            res.json(res3)
            db.close()
        })
    }
})
});

```

```

app.get("/allPresent",(req,res)=>{
    mongoClient.connect(M_url,function(err,db){
        if(err) throw err;
        dbo = db.db("HMS")
        dbo.collection("Problems").find({status:{$ne:"working"}}).count(function(err,count){
            if(err) throw err;
            res.json(count)
        })
    })
});

```

```

app.get("/getPresentPages/:p",(req,res)=>{
    mongoClient.connect(M_url,function(err,db){
        if(err) throw err;
        dbo = db.db("HMS")

        if(req.params.p==1){
            dbo.collection("Problems").find({status:{$ne:"working"}}).project({_id:0,solved:0}).li
mit(10).toArray(function(err,res3){
                if(err) throw err;
                res3.push({ mes:"success"})
                res.json(res3)
                db.close()
            })
        }
    })
});

```

```

        })
    }
    else{
        dbo.collection("Problems").find({status:{$ne:"working"}}).project({_id:0,solved:0}).s
kip((req.params.p-1)*10).limit(10).toArray(function(err,res3){
            if(err) throw err;
            res3.push({ mes:"success"})
            res.json(res3)
            db.close()
        })
    }
})
});

```

```

app.get("/getHodDept/:name",(req,res)=>{
    mongoClient.connect(M_url,function(err,db){
        if(err) throw err;
        dbo = db.db("HMS")
        dbo.collection("Departments").find({hod:req.params.name}).project({_id:0,hod:0}).toArr
ay(function(err,res2){
            if(err) throw err;
            res.json(res2)
            db.close()
        })
    })
});

```

```

app.get("/getDeptSys/:dept",(req,res)=>{
    mongoClient.connect(M_url,function(err,db){
        if(err) throw err;
        dbo = db.db("HMS")
        dbo.collection("DeptPeri").find({ dept:req.params.dept}).project({_id:0}).toArray(funcio
n(err,res2){
            if(err) throw err;
            res.json(res2)
            db.close()
        })
    })
});

```

```

    })
  });

app.get("/updateDeptPro/:dept/:name/:tech/:mes/:status",(req,res)=>{
  mongoClient.connect(M_url,function(err,db){
    if(err) throw err;
    dbo = db.db("HMS")
    var cond = {dept:req.params.dept,name:req.params.name}
    if(req.params.status=="working"){
      var ins = {$set:{assigned_to:"",mes:"",status:req.params.status}}
    }
    else{
      var ins = {$set:{assigned_to:req.params.tech,mes:req.params.mes,status:req.params.sta
tus}}
    }
    dbo.collection("DeptPeri").updateOne(cond,ins,function(err,res2){
      if(err) throw err;
      res.json([ {res:"success"} ])
      db.close()
    })
  })
});

app.get("/getDeptTechSys/:tech",(req,res)=>{
  mongoClient.connect(M_url,function(err,db){
    if(err) throw err;
    dbo = db.db("HMS")
    dbo.collection("DeptPeri").find({assigned_to:req.params.tech}).project({_id:0}).toArray(
function(err,res2){
  if(err) throw err;
  res.json(res2)
  db.close()
})
})
});
app.listen(3000);

```

Frontend Code: (Angular)

App root component:

app.component.ts

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';
```

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'project';

  constructor(public router : Router){
    this.router.navigateByUrl("home");
  }
}
```

app.component.html

```
<router-outlet></router-outlet>
```

app-route.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { MemberComponent } from './member/member.component';
import { GuestComponent } from './guest/guest.component';
import { HomeComponent } from './home/home.component';
import { AdminComponent } from './member/admin/admin.component';
import { AdministratorComponent } from './member/administrator/administrator.component';
```



```

import { InchargeComponent } from './member/incharge/incharge.component';
import { TechnicianComponent } from './member/technician/technician.component';
import { HodComponent } from './hod/hod.component';

const routes: Routes = [
  {path:"member",component:MemberComponent},
  {path:"guest",component:GuestComponent},
  {path:"home",component:HomeComponent},
  {path:"member/admin",component:AdminComponent},
  {path:"member/administrator",component:AdministratorComponent},
  {path:"member/incharge",component:InchargeComponent},
  {path:"member/technician",component:TechnicianComponent},
  {path:"member/hod",component:HodComponent}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }

```

Guest Component:

guest.component.ts

```

import { Component, OnInit, ViewChild, ElementRef } from '@angular/core';
import { HttpClient } from '@angular/common/http'

@Component({
  selector: 'app-guest',
  templateUrl: './guest.component.html',
  styleUrls: ['./guest.component.css']
})
export class GuestComponent implements OnInit {

```

```
rep = false;
tk = false;
sta = false;

tk_db : string;

labs_db = []
systems_db = []

status_db = false
db_sys : string;
db_lab : string;
db_tech : string;
db_sta : string;
db_rdate : string;
db_sdate : string;
db_mes : string;

constructor(public http:HttpClient) { }

ngOnInit() { }

ngAfterViewInit(){ }

changed(lab:string){

    this.systems_db = []

    this.http.get("http://localhost:3000/getSystemslist/"+lab)
        .subscribe((data)=>{
            Object.values(data).forEach(ele=>{
                this.systems_db.push(ele.name)
            })
            console.log(this.systems_db)
        },
```

```
(error)=>{
    console.log(error)
}
)
```

```
report(){
    this.rep = true;
    this.sta = false;
    this.tk = false;
    this.labs_db = []
```

```
this.http.get("http://localhost:3000/getlabslist")
    .subscribe((data)=>{
        Object.values(data).forEach(ele=>{
            this.labs_db.push(ele.name)
        })
        console.log(this.labs_db)
    },
```

```
(error)=>{
    console.log(error)
}
)
```

```
reportSer(lab:string,sys:string,mes:string){
    this.http.get("http://localhost:3000/insertProblem/"+lab+"/"+sys+"/"+mes)
    .subscribe((data)=>{
        console.log(data[0])
        if(data[0].mes=="success"){
            this.tk = true;
            this.rep = false;
            this.tk_db = data[0].id;
        }
    },
```

```
(error)=>{  
    console.log(error)  
}  
);  
}
```

```
status(){  
    this.rep = false;  
    this.sta = true;  
    this.tk = false;  
    this.status_db = false;  
}
```

```
statusSer(token:string){  
    this.http.get("http://localhost:3000/getStatus/"+token)  
        .subscribe((data)=>{  
            if(data[1].mes===("success")){  
                this.db_sys = data[0].system  
                this.db_lab = data[0].lab  
                this.db_tech = data[0].assigned_to  
                this.db_sta = data[0].status  
                this.db_rdate = data[0].date  
                this.db_sdate = data[0].solved  
                this.db_mes = data[0].problem  
                this.status_db = true  
            }  
        }  
    },
```

```
(error)=>{  
    console.log(error)  
}  
);  
}  
}
```

guest.component.html

```
<h1>Welcome! How can I help you</h1>
<div class="menu">
  <p (click)="report()">Report</p>
  <p (click)="status()">Check status</p>
</div>
<div>
  <ng-container *ngIf=rep>
    <div id="report">
      <h1>Please select which system you want to report problematic</h1>
      <form class="form">
        <table style="">
          <tr>
            <td> <h3>Lab</h3>
            </td>
            <td>
              <select #slab (change)="changed(slab.value)">
                <option>select</option>
                <option *ngFor="let lab of labs_db" value={{ lab }}>{{ lab }}</option>
              </select>
            </td>
          </tr>
          <tr>
            <td> <h3>System</h3>
            </td>
            <td>
              <select #ssys>
                <option *ngFor="let sys of systems_db" value={{ sys }}>{{ sys }}</option>
              </select>
            </td>
          </tr>
          <tr>
            <td> <h3>Message</h3>
            </td>
```

```

        <td>
            <input type="text" #prob/>
        </td>
    </tr>
    <tr>
        <td>
            <button value="submit" (click)="reportSer(slab.value,ssys.value,prob.value)">
Submit</button>
        </td>
    </tr>
</table>
</form>
</div>
</ng-container>
<ng-container *ngIf=tk>
    <div>
        <p>Your token id is {{tk_db}}</p>
    </div>
</ng-container>
<ng-container *ngIf=sta>
<div id="status">
    <form class="form">
        <table >
            <tr>
                <td> <h5>Please enter your token number :</h5>
                </td>
                <td> <input type="text" #token/>
                </td>
            </tr>
            <tr>
                <td colspan="2">
                    <button value="submit" (click)="statusSer(token.value)">Submit</button>
                </td>
            </tr>
        </table>

```

```
<ng-container *ngIf=status_db>
```

```
  <table >
```

```
    <tr>
```

```
      <td><p>System :</p>
```

```
    </td>
```

```
      <td> <p>{{ db_sys }}</p>
```

```
    </td>
```

```
      <td><p>Lab :</p>
```

```
    </td>
```

```
      <td> <p>{{ db_lab }}</p>
```

```
    </td>
```

```
  </tr>
```

```
  <tr>
```

```
    <td> <p>Technician :</p>
```

```
  </td>
```

```
    <td> <p>{{ db_tech }}</p>
```

```
  </td>
```

```
    <td><p>Status :</p>
```

```
  </td>
```

```
    <td> <p>{{ db_sta }}</p>
```

```
  </td>
```

```
  </tr>
```

```
  <tr>
```

```
    <td><p>Reported date :</p>
```

```
  </td>
```

```
    <td><p>{{ db_rdate }}</p>
```

```
  </td>
```

```
    <td><p>Solved data:</p>
```

```
  </td>
```

```
    <td><p>{{ db_sdate }}</p>
```

```
  </td>
```

```
  </tr>
```

```
</table>
```

```
<tr>
```

```

        <td><p>Problem :</p>
        </td>
        <td> <p>{{ db_mes }}</p>
        </td>
    </tr>
</ng-container>
</form>
</div>
</ng-container>
</div>

```

Member Component:

member.component.ts

```

import { Component, OnInit, NgModule } from '@angular/core';
import { DatashareService } from '../datashare.service';
import { Router } from '@angular/router';
import { HttpClient } from '@angular/common/http';

@Component({
  selector: 'app-member',
  templateUrl: './member.component.html',
  styleUrls: ['./member.component.css']
})

export class MemberComponent implements OnInit {

  name : "#";
  incharge = "C";
  preNameList = [];
  warn = ""

  constructor(public share:DatashareService, public router:Router,public http:HttpClient) { }

```



```

ngOnInit() {
  this.share.logged = false;
}

auth(np:any,user:any,pass:any){
  this.share.incharge=this.incharge;
  this.http.get("http://localhost:3000/auth/"+np+"/"+user+"/"+pass)
    .subscribe((data)=>{
      console.log(Object(data).status)
      if(Object(data).status=="success"){
        this.router.navigate(["member/"+np]);
        this.share.name = user;
        this.share.logged = true;
      }
      else{
        this.warn = "Wrong username or password"
        this.share.logged = false;
        console.log("error")
      }
    },
    (error)=>{
      console.log(error)
    }
  );
}

```

```

getNames(pp:any){
  console.log(pp)
  this.http.get("http://localhost:3000/getNames/"+pp)
    .subscribe((data)=>{
      this.preNameList = []
      Object.values(data).forEach(ele=>{
        this.preNameList.push(ele.name)
      })
      console.log(this.preNameList)
    },

```

```

(error)=>{
  console.log(error)
}
);
}
}

```

guest.component.html

```

<h1>Welcome back! Please login with your credentials</h1>
<form>
  <table style="overflow-x: auto;margin: auto;">
    <tr>
      <td><h3>Designation :</h3>
      </td>
      <td>
        <select name="designation" (change)="getNames(sss.value)" #sss>
          <option>Select</option>
          <option value="admin">Admin</option>
          <option value="administrator">Heads</option>
          <option value="hod">HOD</option>
          <option value="incharge">Incharge</option>
          <option value="technician">Technician</option>
        </select>
      </td>
    </tr>
    <tr>
      <td><h3>Username :</h3>
      </td>
      <td>
        <select [(ngModel)]="name" name="name" #sname>
          <option *ngFor="let nam of preNameList">{{ nam }}</option>
        </select>
      </td>
    </tr>
  </table>

```

```

<tr>
  <td><h3>Password :</h3>
  </td>
  <td><input type="password" #pass/><br>
  </td>
</tr>
<tr>
  <td>
    <p>{{ warn }}</p>
    <button (click)="auth(sss.value,sname.value,pass.value)">Login</button>
  </td>
</tr>
</table>
</form>
<app-incharge [incha]='incharge' *ngIf=false></app-incharge>

```

DataShare Service:

datashare.service.ts

```
import { Injectable } from '@angular/core';
```

```

@Injectable({
  providedIn: 'root'
})

```

```
export class DatashareService {
```

```
  incharge:string;
```

```
  name : string;
```

```
  logged = false;
```

```
  constructor() { }
```

```
}
```

4.4 COMPONENT DIAGRAM:

A component diagram displays the structural relationship of components of a software system. These are mostly used when working with complex systems that have many components. Components communicate with each other using interfaces. The interfaces are linked using connectors. Below images shows a component diagram.

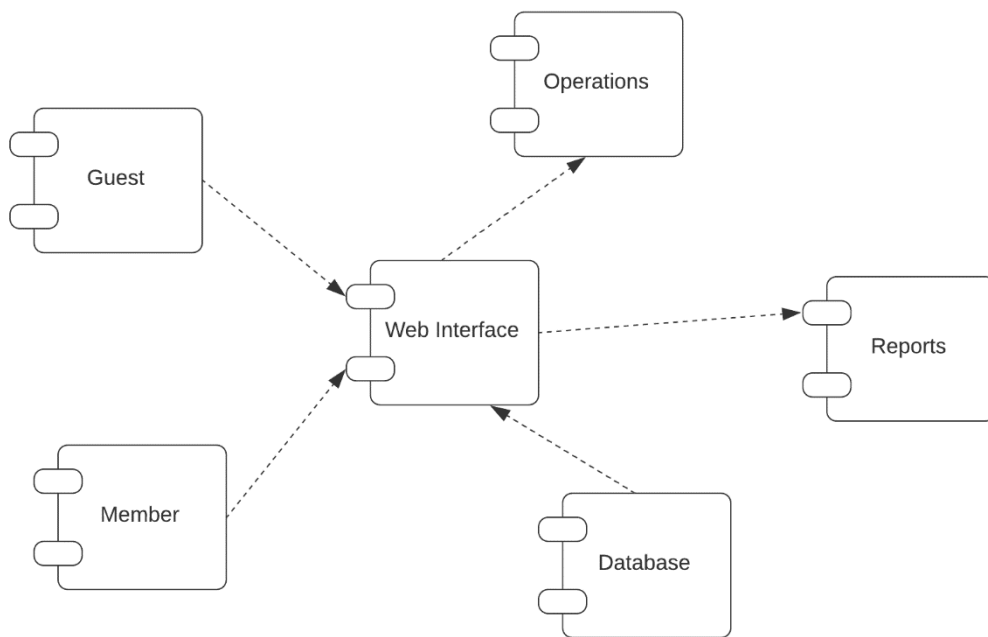


Fig 4.4: Component diagram for Hardware Maintenance System

4.5 DEPLOYMENT DIAGRAM:

Deployment diagram is a structure diagram which shows architecture of the system as deployment (distribution) of software artifacts to deployment targets. Artifacts represent concrete elements in the physical world that are the result of a development process.

Examples of artifacts are executable files, libraries, archives, database schemas, configuration files, etc. Deployment target is usually represented by a node which is either hardware device or some software execution environment. Nodes could be connected through communication paths to create networked systems of arbitrary complexity.

Deployment diagrams could describe architecture at specification level (also called type level) or at instance level (similar to class diagrams and object diagrams).

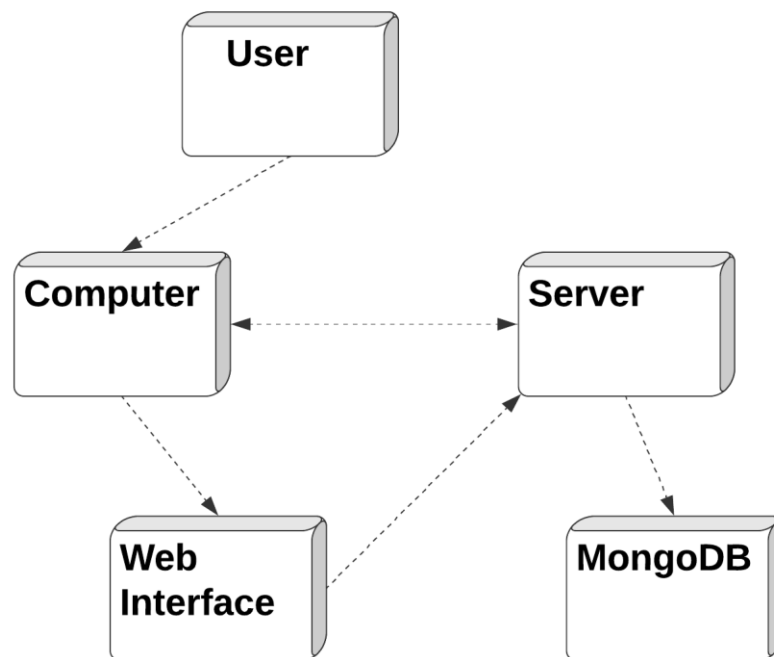


Fig4.5:Deployment diagram for Hardware Maintenance System

CHAPTER-5

SCREEN SHOT

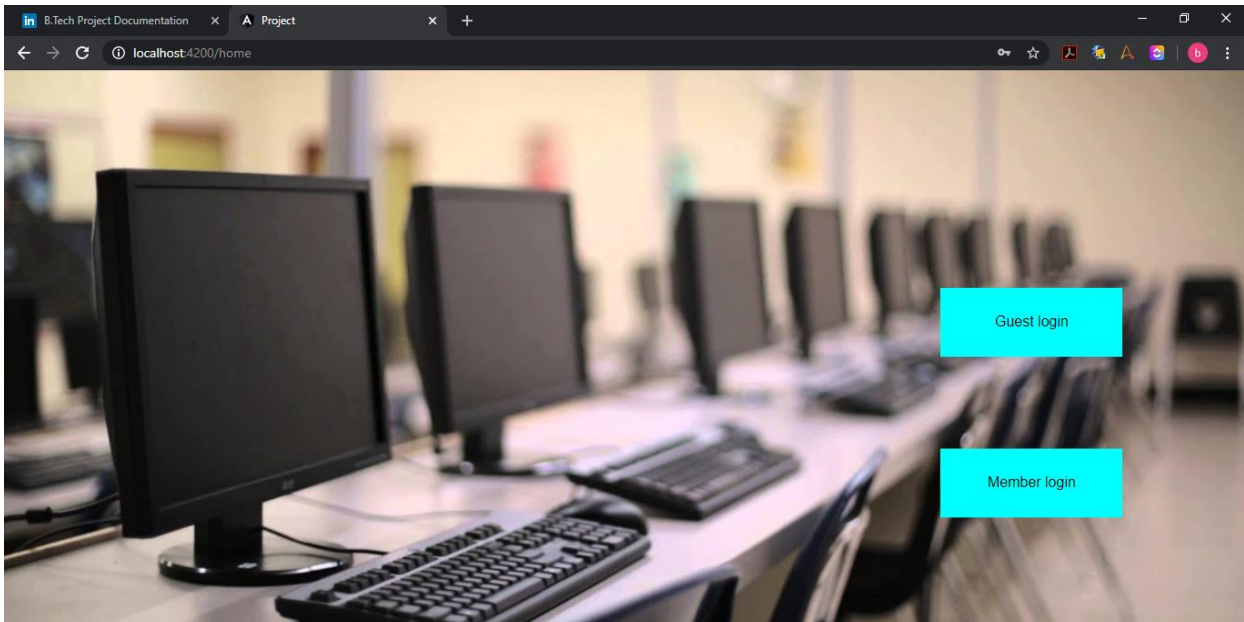


Fig 5.1: Login Screen

The above figure shows the home page of the website where users can choose their logins.

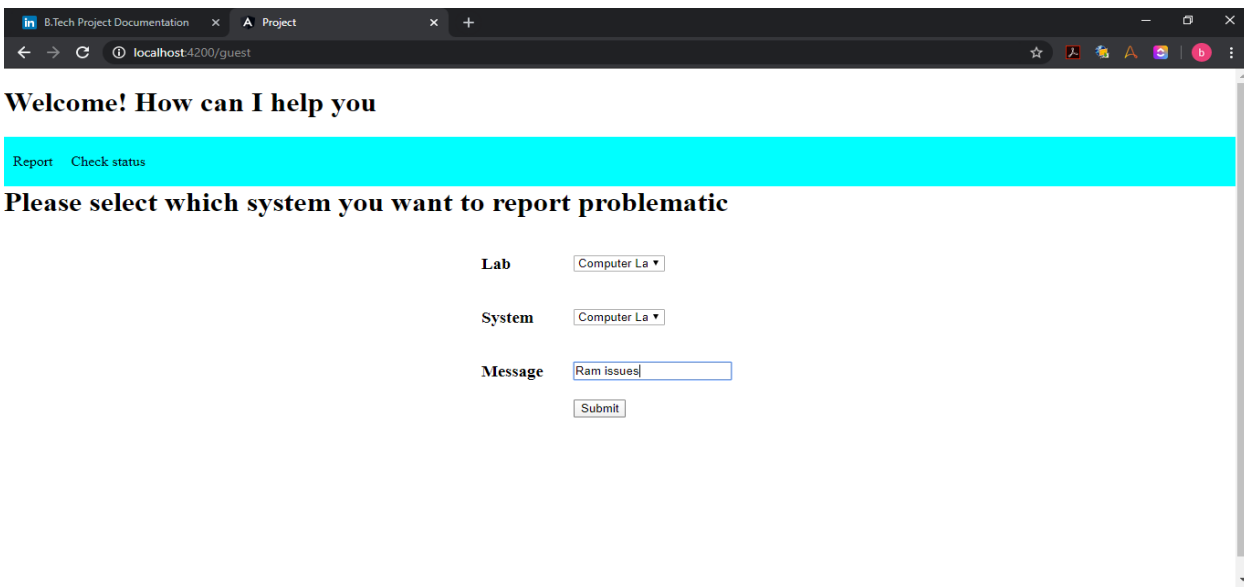


Fig 5.2 : Report Screen

The above figure shows the page where user can select the problematic system and report it.

Welcome! How can I help you

Report Check status

Please enter your token number :

System : Computer Labsys5 Lab : Computer Lab

Technician : john2 Status : reported

Reported date : 05/12/2020 Solved data:

Problem : Ram issues

Fig 5.3 : Check Status

The above figure shows the page where users can check the status of the reported issue

Welcome back! Please login with your credentials

Designation :

Username :

Password :

Fig 5.4 : Login Screen

The above figure shows the page where members can login with their credentials

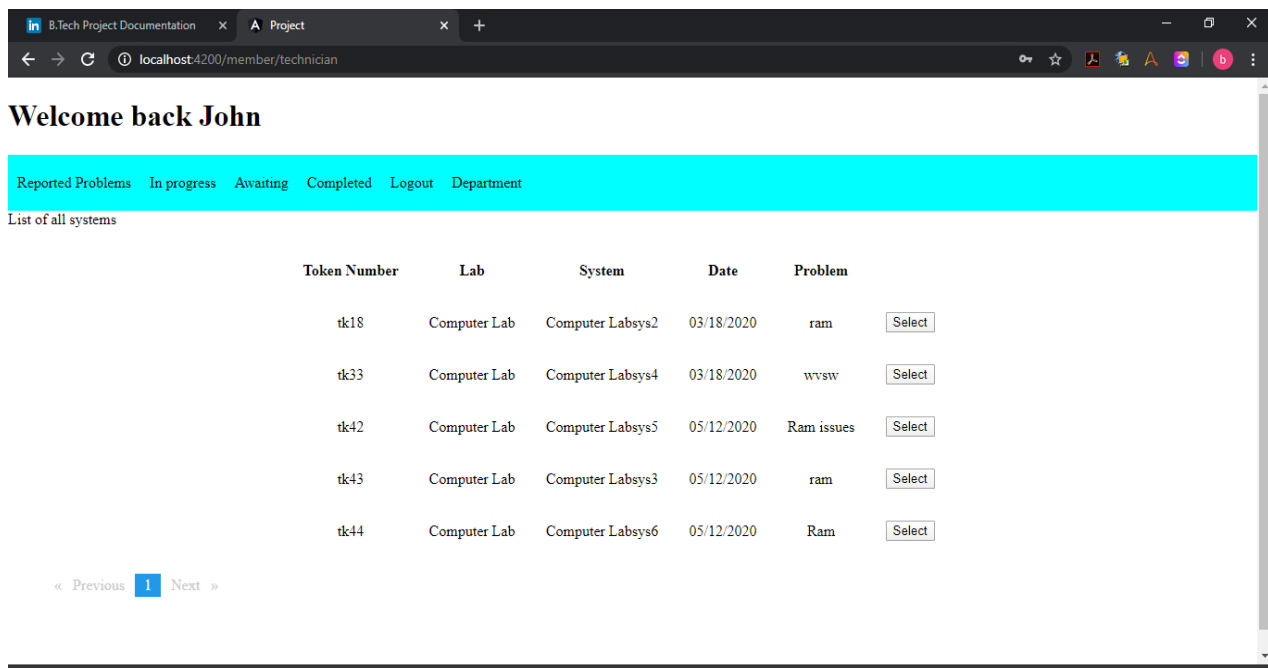


Fig 5.5 : Display list of reports

The above figure show the list of all problems that have been reported to certain technician

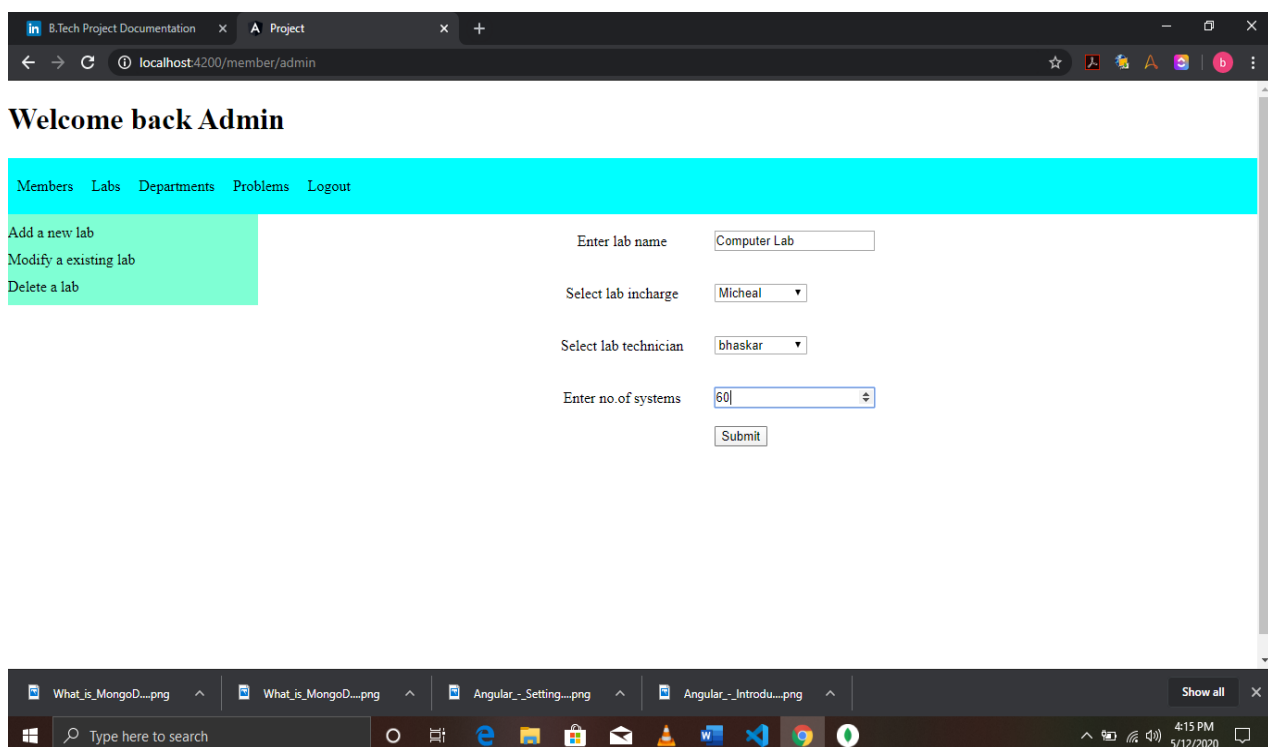


Fig 5.6 : Creating a new Lab

The above figure shows how admin creates a new lab using existing pool of members

CHAPTER-6

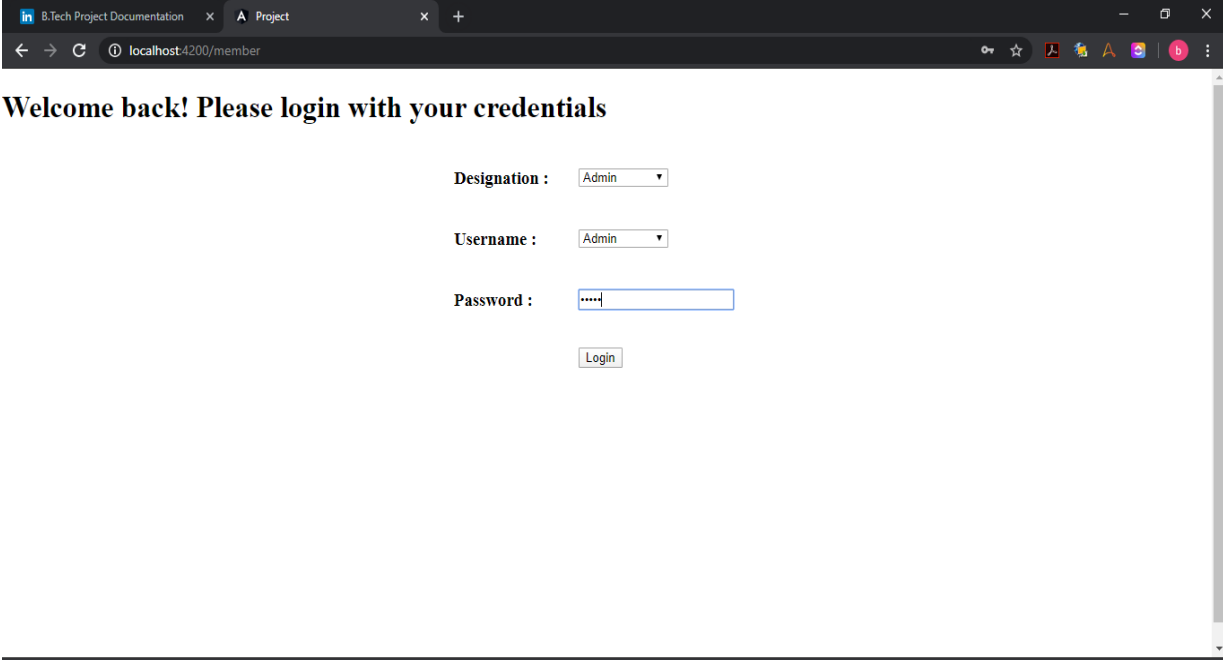
TESTING

6.TESTING:

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet –undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hang together. System testing requires a test consists of several key activities and steps for run program, string, system and is important in adopting a successful new system.

6.1. TEST CASE 1:

In this test case we are going to test whether the application is secured or not i.e., the app must be logged in as shown in fig 6.1.2 only when the member enters correct credentials as shown in fig 6.1.1 else it must prompt wrong credentials as shown in fig 6.1.3.



Designation :

Username :

Password :

Fig 6.1.1 : Entering credentials

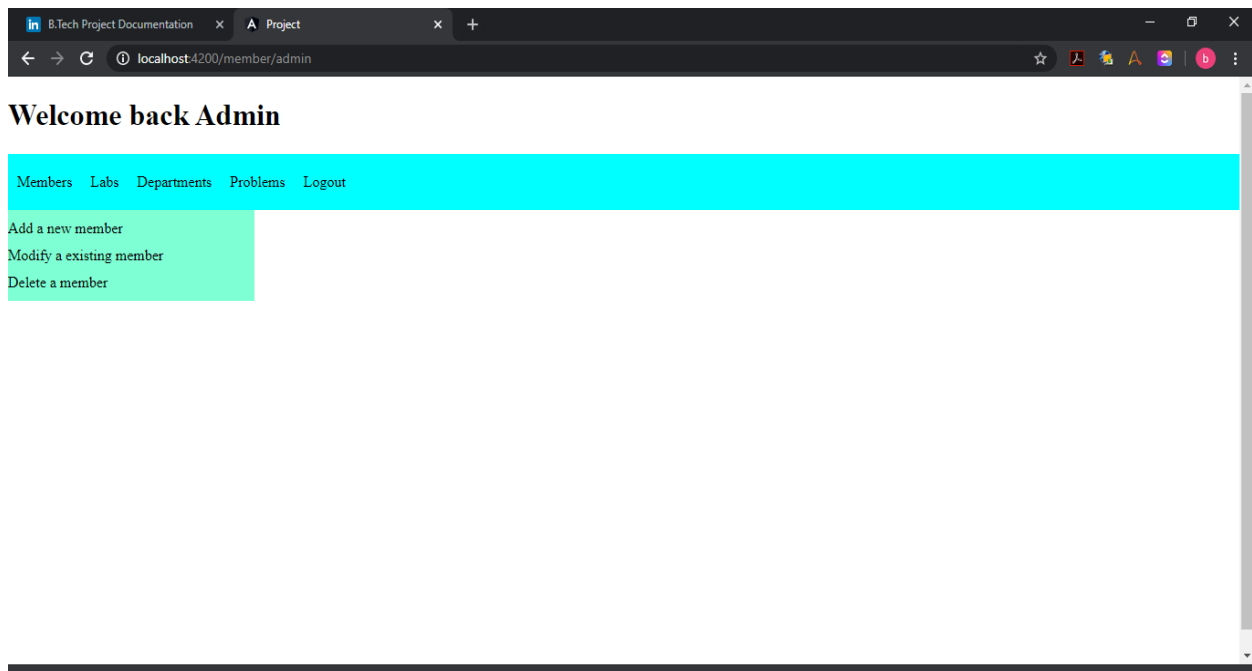


Fig 6.1.2 : Welcome screen

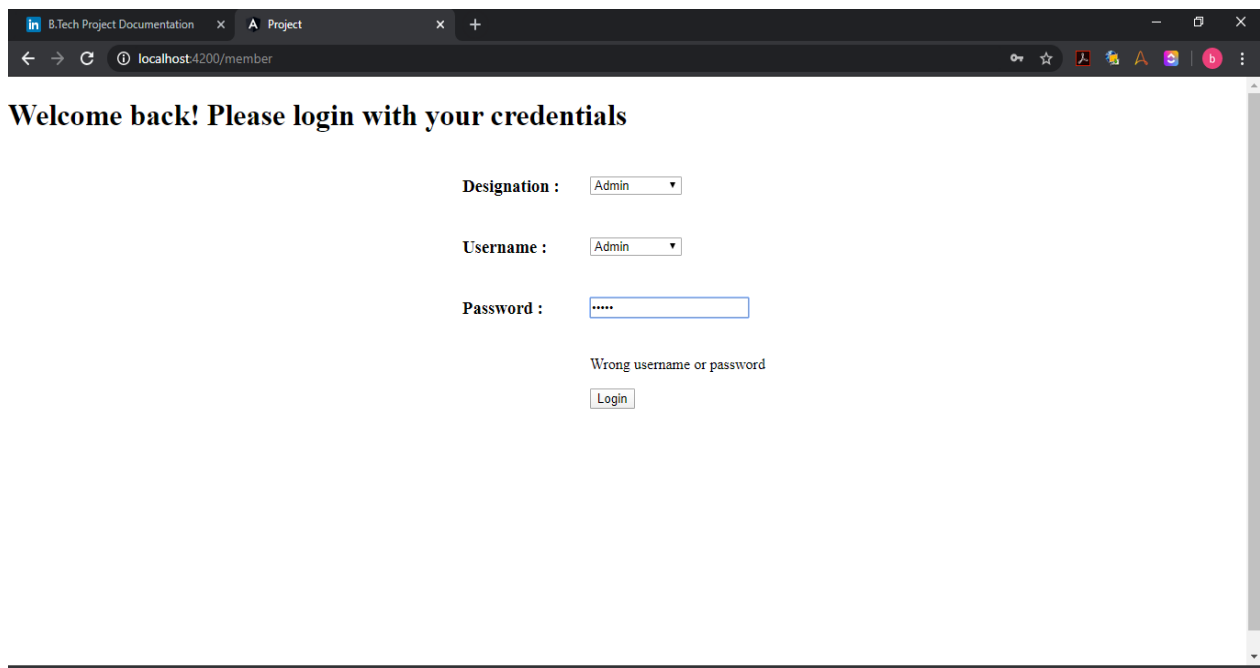
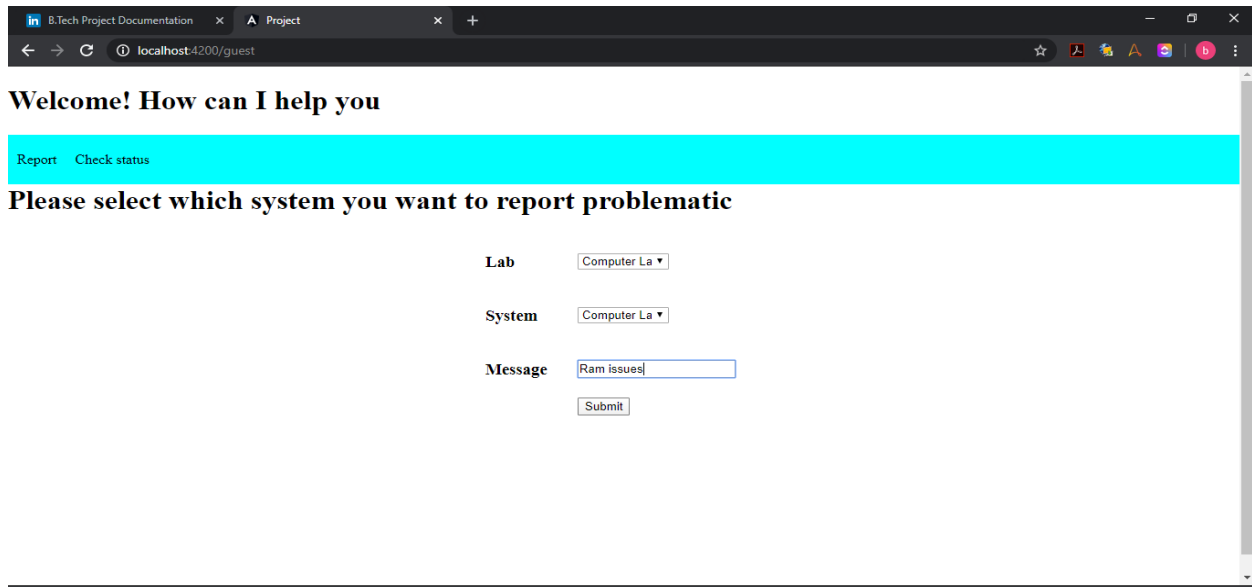


Fig 6.1.3 : Invalid credentials

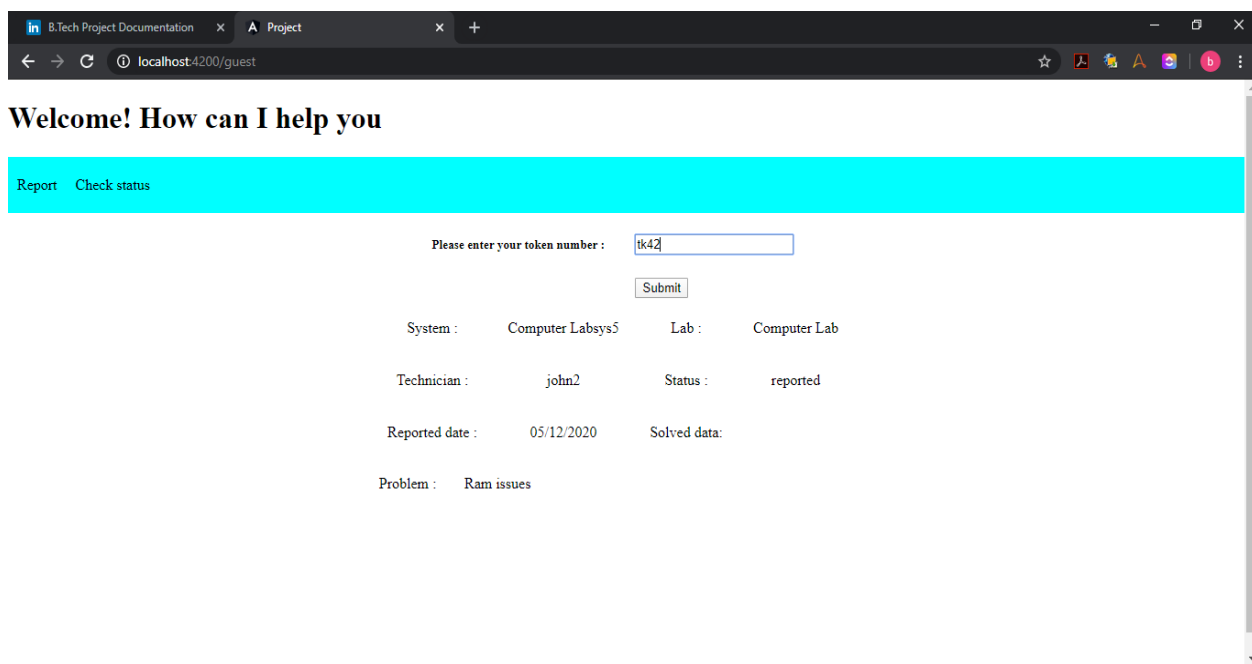
6.2 TEST CASE 2:

In this test case we are going to test the condition whether the reported problem shown in fig 6.2.1 is being reported to the proper technician as shown in fig 6.2.2.



The screenshot shows a web browser window with the URL `localhost:4200/guest`. The page has a header with the text "Welcome! How can I help you" and a navigation bar with "Report" and "Check status" links. Below the navigation bar, there is a section titled "Please select which system you want to report problematic". This section contains three dropdown menus: "Lab" (selected "Computer La"), "System" (selected "Computer La"), and "Message" (containing the text "Ram issues"). A "Submit" button is located below the "Message" dropdown.

Fig 6.2.1 : Reporting a problem



The screenshot shows the same web browser window, but the "Check status" link is active. The page displays the status of the reported problem. It includes a "Please enter your token number :" field with the value "tk42" and a "Submit" button. Below this, the status information is displayed in a table-like format:

System :	Computer Labsys5	Lab :	Computer Lab
Technician :	john2	Status :	reported
Reported date :	05/12/2020	Solved data:	
Problem :	Ram issues		

Fig 6.2.2 : Checking status

6.3 TEST CASE 3 :

In this test case we are going to check whether the task status are being updated properly or not. Here we can see tk42 in fig 6.3.1 which is in reported status, now we will mark it as in progress as shown in fig 6.3.2

Token Number	Lab	System	Date	Problem	
tk18	Computer Lab	Computer Labsys2	03/18/2020	ram	Select
tk33	Computer Lab	Computer Labsys4	03/18/2020	wvsw	Select
tk42	Computer Lab	Computer Labsys5	05/12/2020	Ram issues	Select
tk43	Computer Lab	Computer Labsys3	05/12/2020	ram	Select
tk44	Computer Lab	Computer Labsys6	05/12/2020	Ram	Select

Fig 6.3.1 : Reported Problems

Token Number	Lab	System	Date	Problem	
tk18	Computer Lab	Computer Labsys2	03/18/2020	ram	Select
tk19	Computer Lab	Computer Labsys1	03/18/2020	pendrive	Select
tk42	Computer Lab	Computer Labsys5	05/12/2020	Ram issues	Select

Fig 6.3.2 : In progress Problem

CHAPTER-7
SUMMARY &CONCLUSION

7. SUMMARY AND CONCLUSION:

By adapting to this project, we would be able to have a centralized record of all peripherals available in every lab and department. The issue which are reported can be stored in the database and be used to effectively measure and monitor the future investments in computer components.

All the systems will be properly utilized and accounted for. It would be so easy to track down any system issues or the management can also easily measure the performance of the lab technicians and incharges. In case of any damage to the record can also be revived due to periodic backup of collections.

- Secured data entry.
- Ease tracking of issue.
- Backup of data in case of faults.
- Time effective filtering and tracking.
- Different status levels to represent problem condition.

CHAPTER-8

FUTURE ENCHANCEMENT

8. FUTURE ENHANCEMENT:

In the near future, we would like to implement speech recognition and facial unlocking features as an optional feature to the product. These implementations would be very helpful to the user for providing easily navigation of the application and also hands-free navigation.

In addition to these features we would also like to make it completely automated software which would auto detect any problems related to the computer and its peripherals such as keyboard, mouse etc. Basic concept of this idea is to build a windows service file which would be running in the background of the computer and be triggered by hardware related events. Then the service would be periodically checking any error codes in the machine. If any error code is encountered then it would send response to the central server using rest api.

In this way, we can auto detect the issues without any one checking and passing over to the technicians. After solving the issue, technicians can also check the compatibility themselves using this service without any third party justification.

CHAPTER-9

BIBLIOGRAPHY

9. BIBLIOGRAPHY:

- ["Angular Docs". angular.io.](#)
- ["What's the difference between AngularJS and Angular?". gorrior.io.](#)
September 19, 2017. Retrieved 2018-01-28.
- ["Angular: Branding Guidelines for AngularJS".](#) Retrieved 2017-03-04.
- Coman Hamilton. ["A sneak peek at the radically new Angular 2.0".](#)
Retrieved 2015-10-21.
- ["Ok... let me explain: it's going to be Angular 4.0". angularjs.blogspot.kr.](#)
Retrieved 2016-12-14.
- ["Angular 4.0.0 Now Available". angularjs.blogspot.ca.](#) Retrieved 2017-03-23.
- ["Angular 4 coming in 2017, to be backwards compatible with Angular 2".](#) react-etc.net. Retrieved 2016-12-14.
- Fluin, Stephen. ["Version 5.0.0 of Angular Now Available".](#) Retrieved 2 November 2017.
- ["Angular 5 JavaScript framework delayed".](#)
- ["Version 6.0.0 of Angular Now Available".](#) Retrieved 4 May 2018.
- Fluin, Stephen (2018-10-18). ["Version 7 of Angular — CLI Prompts, Virtual Scroll, Drag and Drop and more".](#) Angular Blog. Retrieved 2019-06-07.
- ["node-v0.x-archive on GitHub".](#) Retrieved 2 August 2014.
- ["Node.js 14 ChangeLog".](#) Retrieved 6 May 2020 – via [GitHub](#).

- Jump up to: a b "nodejs/node". GitHub.
- "node/LICENSE at master". GitHub. Node.js Foundation. 17 September 2018. Retrieved 17 September 2018.
- "The MIT License". Open Source Initiative. 17 September 2018. Retrieved 17 September 2018.
- "Express 4.x changelog". expressjs.com.
- <https://github.com/expressjs/express/releases/latest>
- "Express.js home page".
- Case study: How & why to build a consumer app with Node.js. VentureBeat.com.
- Holowaychuck, TJ. "Express 1.0beta". Archived from the original on 2015-07-06.
- "Mean.io: The Friendly & Fun Javascript Fullstack for your next web application". Archived from the original on 13 June 2019. Retrieved 15 July 2019.
- "TJ Holowaychuk Passes Sponsorship of Express to StrongLoop". StrongLoop. Archived from the original on 11 October 2016. Retrieved 11 February 2016.
- "IBM snaps up StrongLoop to add Node.js smarts to BlueMix". Infoworld. IDG. Retrieved 11 February 2016.
- "Node.js Foundation to shepherd Express Web framework". Infoworld. IDG. Retrieved 11 February 2016.
- "Companies using Express". expressjs.com. Retrieved 2018-12-04.

