VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY AND RESEARCH (Deemed to be UNIVERSITY)

VADLAMUDI, GUNTUR DIST, ANDHRA PRADESH, INDIA, PIN-522 213

CERTIFICATE

This is to certify that the Internship Report entitled **" STUDENT SUPERVISION SYSTEM "** that is being submitted by **V. Naga Surya Satish (151FA04196)** in partial fulfilment for the award of B. Tech degree in Computer Science and Engineering to the Vignan's Foundation for Science, Technology and Research, Deemed to be University, is a record of bonafide work carried out by them at **Inked Creatively LLC** under the supervision of **Ms. Nikitha Reddy** under the co-guidance of the following faculty member of CSE Department.

K. Hema Bhargavi	External Examiner	HOD, CSE
Assistant Professor.		Dr. Venkatesulu Dondeti

DECLARATION

I hereby declare that the project entitled "**STUDENT SUPERVISION SYSTEM**" submitted for the **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**. This dissertation is our original work and the project has not formed the basis for the award of any degree, associate-ship, and fellowship or any other similar titles and no part of it has been published or sent for publication at the time of submission.

By

V. Naga Surya Satish (151FA04196)

Date: 01-December-2018.

ACKNOWLEDGMENT

Internship program is a golden opportunity for learning and self-development. Consider myself very lucky and honored to have so many wonderful people lead me through in the completion of this project.

I express my gratitude towards the Managing Director of **INKED CREATIVELY LLC, Ms. Nikitha Reddy,** for permitting me to undertake the **Internship Program** in their extreme company and for the help and cooperation throughout the course of my Internship Program.

We feel it our responsibility to thank **Ast Prof. K. Hema Bhargavi** under whose valuable guidance that the project came out successfully after each stage, and also it is our responsibility to extend our thanks to **Mr. D. YAKOBU, Department Internship Coordinator,** for extending his support towards the Internship Program in **Inked Creatively LLC**, in **Hyderabad**.

It is a great pleasure for me to express my sincere thanks to **Dr. Venkatesulu Dondeti, HOD, CSE** of VFSTR Deemed to be University**,** for providing me an opportunity to do my Internship Program.

It is our pleasure to extend our sincere thanks to our honorable Dean Engineering & Management, **Dr. V. MADHUSUDAN RAO,** for providing me an opportunity to do my academics in VFSTR Deemed to be University.

We extend our wholehearted gratitude to all our faculty members of Department of Computer Science and Engineering who helped us in our academics throughout course.

Finally, we wish to express thanks to our family members for the love and affection overseas and forbearance and cheerful depositions, which are vital for sustaining effort, required for completing this work.

With Sincere regards,
V. Naga Surya Satish (151FA04196)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS & ABBREVIATIONS

- **HTML:** Hyper Text Markup Language.
- **CSS:** Cascading Style Sheets.
- **IDE:** Integrated Development Environment
- **GUI:** Graphical User Interface
- **HTTP:** Hyper Text Transfer Protocol
- **API:** Application Programming Interface
- **E-R:** Entity-Relationship
- **UML:** Unified Modeling Language
- **OOAD:** Object-Oriented Analysis & Design.

INTERNSHIP SUMMARY

Location: Hyderabad

Center: Inked Creatively LLC

Duration: 4 months 25days.

Date of start: 5th July, 2018.

Date of submission: 30th November 2018.

Title of project: Student Supervision System.

Team Members:

V. Naga Surya Satish (151FA04196)
N. Srinivasa Pavan Kalyan (151FA04241)
P. Sai Kiran (151FA04246)

Name of the guide: Ms. Nikitha Reddy, Head HR, Inked Creatively LLC.

Name of Faculty guide: K. Hema Bhargavi, Assistant Professor, VFSTR University.

Project Area: Web Application Development for Student Supervision System.

Abstract:

This system is intended to ease the burden on administration duties for managing the office staff in processing Supervision and Management by making information retrieval and management faster, easier, and more efficient compared to the normal manual system.

Student supervision system is a software used by supervisors in order to monitor and reviews students' performance throughout a term and generate reports. This system is helpful for the academicians as well as school administrators to reduce the paperwork which most of the schools are familiar with.

.

Signature of Student **Signature of Faculty Guide**

Date: **Date:**

PROFILE OF THE COMPANY

What is Inked Creatively LLC?

Inked Creatively LLC is a state-of-the-art interface enabling students with knowledge-rich content on a hand-held platform. The outcome of this is that you can study, and learn on the go – ensured by an interactive and real-time feedback system.

We are an open, inclusive, and diverse organization where all team members are recognized for what they bring. Our mission is to establish Inked Creatively LLC as a leading student analytics platform in education space.

Edwisely is driven by alumni from TU, Draper, Deloitte, and successful entrepreneurs.

About Inked Creatively LLC

Inked Creatively LLC is an innovative AI - Driven learning platform which accommodates an institutional curriculum and blends it with contextual learning of students to give them highly personalized learning paths.

Our target market segments are State Universities, Private Universities, Autonomous Colleges, Private colleges affiliated to universities offering engineering courses. Through our platform, we generate valuable insights to the institutions on various key performance indicators of teachers, students and overall learning aspects of the education.

Vision:

Building blocks towards better learning.

Office address:

819/1, Manjeera Majestic Commercial,
JNTU Road,
Hyderabad 50007

CHAPTER - 1

INTRODUCTION

The chapter gives brief introduction of the project.

1. INTRODUCTION

1.1 Introduction

This system is intended to ease the burden on administration duties for managing the office staff in processing Supervision and Management by making information retrieval and management faster, easier, and more efficient compared to the normal manual system.

1.2 Literature Survey

Literature survey is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, the next steps are to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system the above consideration are taken into account for developing the proposed system.

- Survey on building Web applications.
- Implementation of a framework scripted in Node.js and Angular.
- Different online examples of data synchronization.
- Research on web server connectivity.
- Approached different Solutions for the requirement.
- Study on Customer Relationship Management.

1.2 Project Background

Student supervision system is a software used by supervisors in order to monitor and reviews students' performance throughout a term and generate reports. This system is helpful for the academicians as well as school administrators to reduce the paperwork which most of the schools are familiar with.

1.3 Objective

Student supervision system is a web application where all the Academic details like Student data, Supervisor data, complaints, results, news sharing and report generation takes at one place.

1.4 Project Description

In this section, all features in application are explained in brief.

1.4.1 Supervisor/User Data

In the corporate institution, the institution have so many students and supervisors. To maintain the student/supervisor data easily we can have all the details like personal information, contact information, educational background, previous experience, work information, results all together can have at one place.

1.4.2 Coursework

In this feature, both supervisor and user have the access. Supervisor can upload course work to the students and student in return can upload course work after completion to the supervisor. The supervisor can approve or reject the class work of the student based on work done, student will get the status info.

1.4.3 Results

This feature is used by both supervisor, student. In this, the supervisor can update results by submitting the required receipts of the expense. By verifying admin can approve or reject the expense. Supervisor will get the status.

1.4.4 Request for Supervisor

This feature is used only by the student inorder to request admin to assign a supervisor if not assigned because admin can only assign supervisors to the students.

1.4.5 Notifications

This feature is used to share the information about meetings, events that are going to held within the institution.

CHAPTER - 2

SOFTWARE REQUIREMENT SPECIFICATION

Gives the details of platform specifications, Hardware, and Software specifications.

2. SOFTWARE REQUIREMENTS SPECIFICATION

2.1 Requirement Analysis

For the purpose of easy maintenance of Student/Supervisor data, course work, results and other information we proposed to develop a web based application using Angular 6, Node.js and MongoDB. As a part of this system, we are going to develop Web based software in which data can be accessed easily. The required documents for these processes are as follows.

1. Problem statement
2. Data flow diagrams
3. Use case diagram
4. Other UML diagrams.

The above mentioned documents gives us diagrammatical view of the system what we are going to develop.

2.2 Problem Statement

The problem statement concentrates on to maintain the student data, supervisor data, course work, results on a single application.

2.3 Functional Requirements

- Functional requirements are nothing but the services provided by the system to the end users.

Dashboard:

- All the modules given by the admin will be displayed in the Dashboard. After which user can view any module from the Dashboard if module doesn't open, an internet connection error shows up else refresh the page by swiping the page down (swipe layout) or refresh icon in the toolbar.

Students list / Supervisor:

- The main aim of Students/Supervisor list is to display the data in the desired manner which is receiving from the admin panel. This list displays the empty list until and unless the records are added.
- All the stored records which are displayed in the list can be viewed by the user but can be deleted only by the admin.
- The admin will display all the fields in the list or can display some fields. It depends on the admin what fields display in the list.

Course Work:

- Course Work is used by both the Supervisor and the user.
- Supervisor can upload Course Work to the Student inorder to complete the course and student upload the course work to the supervisor after completion of work.
- Supervisor can approve or reject the course work send by the student.
- Student will know the status of the course work.

Results:

- Results can access by both student and supervisor.
- In this, the supervisor can update the results based on the course work and performance of the student.
- The student can see the results on their dashboard.
- Changes to the results can be done only by the database admin.

Request for Supervisor:

- This feature is used only by the student in order to request the admin for assigning the supervisor.
- This module has access to both Supervisor and Student.
- Only admin can assign the supervisors to the students.

Notifications:

- Admin can post the important information in this.
- The user can view the information shared by the admin.
- This is used to share important things by the admin within the company.

Update Password:

- This module is accessible by User/Supervisor to change the login credentials which are mentioned earlier.

2.4 Software Requirement Specification

The project is a web based application developed using Angular 6, Node.js, and MongoDB.

2.4.1 Purpose

The purpose of this document is to present a detailed description of **"SSS Application"**. It will explain the purpose and features of the SSS website that will provide, constraints under which it must operate and how the system will react. The document also describes the non-functional requirements of the system.

2.4.2 Scope of the project

The application has three main modules. They are admin module, supervisor and student module.

1) ADMIN

Admin can add supervisors to the students and admin can delete the users. In this, the Supervisor modules have some functionalities. They are:

i. View Supervisor data.
ii. Time/Expense updating.
iii. View Payslip.
iv. View News.
v. Logout.

2) SUPERVISOR

Supervisor module have special features and the credentials will be given by main admin. The functionalities are:

i. Student list.
ii. Notifications.
iii. Upload Results.
iv. Upload course work.
v. Update Password.
vi. Logout.

3) STUDENT

Student module have special features. The functionalities are:

i. Student list.
ii. Notifications.
iii. Upload Results.
iv. Upload course work.
v. Update Password.
vi. Logout.
vii. Request for Supervisor.

2.4.3 Technologies Used

Angular

Angular is a Type-Script based open source front-end web application platform led by the Angular Team at Google and by a community of individuals and corporations. Angular is a complete rewrite from the same team that built AngularJS.

Angular 6 was released on May 4, 2018. This is a major release focused less on the underlying framework, and more on the toolchain and on making it easier to move quickly with Angular in the future, like: ng update, ng add, Angular Elements, Angular Material + CDK Components, Angular Material Starter Components, CLI Workspaces, Library Support, Tree Shakable Providers, Animations Performance Improvements, and RxJS v6.

Node.js

Node.js is an open-sourse, cross-platform JavaScript run-time environment that executes JavaScript code outside of a browser. Typically, JavaScript is used primarily for client-side scripting, in which scripts written in JavaScript are embedded in a webpage's HTML and run client-side by a JavaScript engine in the user's web browser. Node.js lets developers use JavaScript to write Command Line tools and for server-side scripting running

scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web application development around a single programming language, rather than different languages for server side and client side scripts.

Visual Studio Code

It is a source code editor developed by Microsoft for Windows, linux and mac OS. It includes support for debugging, embedded git control, syntax highlighting,intelligent code, completion, snippets, and code refactoring. It is also customizable, so users can change the editor's theme, keyboard short cuts, and preferences.

Visual Studio Code is a source code editor. It supports a number of programming languages and a set of features that may or may not be available for a given language, as shown in the following table. Many of Visual Studio Code features are not exposed through menus or the user interface. Rather, they are accessed via the command palette or via a .json file (e.g., user preferences). The command palette is a command-line interface. However, it disappears if the user clicks anywhere outside it or presses a key combination on the keyboard to interact with something outside it. When this happens, the command in progress is cancelled.

In the role of a source code editor, Visual Studio Code allows changing the code page in which the active document is saved, the character that identifies line break (a choice between LF and CRLF), and the programming language of the active document.

2.4 MongoDB

Mongo DB is a cross-platform document-oriented database program that used to be free and open source but has since changed to a proprietary license (Server Side Public License). Classified as a NoSQL database program, MongoDB uses JSON -like documents with schemata. MongoDB is developed by MongoDB Inc.

MongoDB supports field, range query, and regular expression searches. Queries can return specific fields of documents and also include user-defined JavaScript functions. Queries can also be configured to return a random sample of results of a given size.

MongoDB provides high availability with replica sets. A replica set consists of two or more copies of the data. Each replica set member may act in the role of primary or secondary replica at any time. All writes and reads are done on the primary replica by default. Secondary replicas maintain a copy of the data of the primary using built-in replication. When a primary replica fails, the replica set automatically conducts an election process to determine which secondary should become the primary. Secondary's can optionally serve read operations, but that data is only eventually consistent by default.

2.4.4 Overview

This is a web based application. Which communicates with the users to update and view the information regarding timesheet, expense, and payroll?

2.4.5 Overall Description

The objective of this document is to formally describe the system's high level requirements including functional requirements, non-functional requirements. The detail structure of this document is organized as follows:

Section 1 of this document provides an overview of the SSS application. This includes a general description of the product, user characteristics, general constraints, and any assumptions for this system. This model demonstrates the development team understands of the application with limited fields as a proof of concept and serves to maximize the team's ability to build a system that truly does support the business.

Section 2 presents the detail requirements, which comprise the domain.

Section 3 consists of the modeling, low level and high level designing and screen captures demonstrating the functional requirements.

Section 4 shows testing the application using different testing techniques like load testing, performance testing, manual testing etc.

2.4.6 Specific Requirements

This section will describe the functions of actors, their roles in the system and the constraints faced by the system.

2.4.7 Product Perspective

The proposed system is used in Institution by the Admin, Supervisor, and Student. This system will provide the Admin and also faculty to add data to the database. The Admin Manage the all the records in a very Effective and Efficient Way. The data consist of institution information, student/supervisor data, results, course work within the institution.

2.5 Software Requirements

The software interface is the operating system, and application programming interface used for the development of the software.

Operating System: Windows XP or higher / Mac OS X 10.5.8 or later / Linux

IDE: Jet brains (webstrom).

Database: MongoDB.

Technologies used: Angular 6, Node.js.

2.6 Hardware Requirements

CLIENT		
Processor	Ram	Operating System
I3 processor or more.	4GB (or) more.	Windows7 (or) more

Table 2:1 Client Requirements

SERVER				
OPERATING SYSTEM	SOFTWARE	PROCESSOR	RAM	DISK SPACE
Windows7(or) more, Ubuntu 12.04 LTS	Angular 6 Node.js 8.11.3 MongoDB 4.0	IntelI XeonI CPU E31220	4GB (or) more	Minimum 250Mb

Table 2:2 Server Requirements

2.7 Functional Requirements (Modules)

The project has two main modes Student and Supervisor each having its own respective specific functionality.

2.7.1 Main Admin

This Main Admin module is to create company login. It contains the details of Company, Contact, etc**.**

2.7.2 Supervisor/User:

In the corporate institution, the institution have so many students and supervisors. To maintain the student/supervisor data easily we can have all the details like personal information, contact information, educational background, previous experience, work information, results all together can have at one place.

2.7.3 Course work:

In this feature, both supervisor and user have the access. Supervisor can upload course work to the students and student in return can upload course work after completion to the supervisor. The supervisor can approve or reject the class work of the student based on work done, student will get the status info.

2.7.4 Results:

This feature is used by both supervisor, student. In this, the supervisor can update results by submitting the required receipts of the expense. By verifying admin can approve or reject the expense. Supervisor will get the status.

2.7.5 Request for a Supervisor:

This feature is used only by the student inorder to request admin to assign a supervisor if not assigned because admin can only assign supervisors to the students.

2.7.6 Notifications:

This feature is used to share the information about meetings, events that are going to held within the institution.

2.7.7 Update Password:

This module is accessible by User/Supervisor to change the login credentials which are mentioned earlier.

2.8 Non-Functional Requirements:

2.8.1 Flexibility & Scalability

Google itself has given a set of applications with Php and HTML but the whole developer community can develop their own applications and they have access to same resource which are accessible to core applications.

2.8.2 Robust

It is quite robust because it can be access in any web browser with any version.

2.8.3 Performance:

There are two primary factors by which Web application performance is measured. The first factor is the efficiency with which the application performs central processing commands. The second factor is the speed with which the application renders two- and three-dimensional graphics on the Browser.

2.8.4 Reliability:

Since the application is being developed through Node.js, Html, and Angular 6 the most famous, efficient and reliable language, so it is reliable in every aspect until and unless there is an error in the programming side. Thus the application can be a compatible and reliable one.

2.8.5 Portability:

This System must be intuitive enough such that user with average background in using mobile phones, Laptops, Tabs can quickly experiment with the system and learn how to use the project. The system has user friendly interface.

2.9 External Interface Requirements

2.9.1 User Interface

A critical aspect of this project was examining how the app would look and its usability. When the App is launched, it displays the home activity that contains the logo of Company and asks the credentials of User. The layout displays Submit button requires online to log in, whereas Dashboard contains all modules. The application has a user friendly interface.

2.10 Feasibility study

A key part of the preliminary investigation that reviews anticipated costs and benefits and recommends a course of action based on operational, technical,

economic, and time factors. The purpose of the study is to determine if the systems request should proceed further.

2.10.1 Organisational Feasibility

The application would contribute to the overall objectives of the Company. It would provide a quick, error free and cost effective solution to the Company Management. It would provide a solution to many issues in the current system. As the new system is flexible and scalable it can also be upgraded and extended to meet other complex requirements which may be raised in the future. However, it is up to the organization to upgrade or extend it.

2.10.2 Economic Feasibility

The project is economically feasible as it only anywhere with internet. The users should be able to connect to internet and this would be the only cost incurred on the project.

2.10.3 Technical Feasibility

To develop this application, a high speed internet connection, a database server, a web server, and software are required. The current project is technically feasible as the application was successfully deployed.

2.10.4 Behavioural Feasibility

The application is behaviourally feasible since it requires no technical guidance, all the modules are user friendly and execute in a manner they were designed to.

CHAPTER - 3

ANALYSIS & DESIGN

this chapter gives the details of the system and data design.

3. ANALYSIS & DESIGN

3.1 Introduction

This system is intended to ease the burden on administration duties for managing the office staff in processing Supervision and Management by making information retrieval and management faster, easier, and more efficient compared to the normal manual system.

Student supervision system is a software used by supervisors in order to monitor and reviews students' performance throughout a term and generate reports.This system is helpful for the academicians as well as school administrators to reduce the paperwork which most of the schools are familiar with.

3.1.1 Purpose

In this section, the purpose of the document and the project is described.

3.1.1.1 Document Purpose

An SDD is a representation of a software system that is used as a medium for communicating software design information.

3.1.1.2 Project Purpose

The prime purpose of this "Student Supervision System application" is to create a fully-fledged Web application which would have all features to maintain a institution data as per requirement at one place. This application works when there is internet connectivity.

The application have student and supervisor module. In this admin can add/delete users, maintain institution data, maintain the student/supervisor data,results as per institution rules and Supervisor. Supervisor module can only access to view the student details and update the results, course work regularly and get the updates made by Student. Supervisor can view their pay slip generated by the Student. The information about meetings, events, projects will be shared by the admin within the institution and supervisor,student can view this.

3.1.2 Scope

In this section, the scope of the document and the project is explained in brief.

3.1.2.1 Document Scope

This document contains a thorough description of the high level architecture that will be used in developing the system. Communicating at a purposefully high level, it will only form the basis for the Software Detailed Design and implementation. However, the SDD itself will not be in sufficient detail to implement the code. It will convey the overall system design of the system, the user interface design and higher level module design (including web development tools). Design details that will not be included in the SDD are:

- Low level classes that will be used in the implementation. The full description of the implementation of each module is not needed, but the public modules that will be interfaced will be described.
- Exact detailed description of interactions within each module.

3.1.2.2 Project Scope

SSS application mainly consists of three modules with each having definite functionality. The purpose of admin is to add, view, maintain and edit the users details, results, course work. Supervisor/student works with login details provided by admin module.

3.2 System Overview

3.2.1 Development Tools

Web applications use certain development tools which are as follows:

3.2.1.1 HTML**Angular**

Angular is a Type-Script based open source front-end web application platform led by the Angular Team at Google and by a community of individuals and corporations. Angular is a complete rewrite from the same team that built AngularJS.

Angular 6 was released on May 4, 2018. This is a major release focused less on the underlying framework, and more on the toolchain and on making it easier to move quickly with Angular in the future, like: ng update, ng add, Angular Elements, Angular Material + CDK Components, Angular Material Starter Components, CLI Workspaces, Library Support, Tree Shakable Providers, Animations Performance Improvements, and RxJS v6.

3.2.1.2 Node.js

Node.js is an open-sourse, cross-platform JavaScript run-time environment that executes JavaScript code outside of a browser. Typically, JavaScript is used primarily for client-side scripting, in which scripts written in JavaScript are embedded in a webpage's HTML and run client-side by a JavaScript engine in the user's web browser. Node.js lets developers use JavaScript to write Command Line tools and for server-side scripting running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web application development around a single programming language, rather than different languages for server side and client side scripts.

3.2.1.3 Visual Studio Code

It is a source code editor developed by Microsoft for Windows, linux and mac OS. It includes support for debugging, embedded git control, syntax highlighting,intelligent code, completion, snippets, and code refactoring. It is also customizable, so users can change the editor's theme, keyboard short cuts, and preferences.

Visual Studio Code is a source code editor. It supports a number of programming languages and a set of features that may or may not be available for a given language, as shown in the following table. Many of Visual Studio Code features are not exposed through menus or the user interface. Rather, they are accessed via the command palette or via a .json file (e.g., user preferences). The command palette is a command-line interface. However, it disappears if the user clicks anywhere outside it or presses a key combination on the keyboard to interact with something outside it. When this happens, the command in progress is cancelled.

In the role of a source code editor, Visual Studio Code allows changing the code page in which the active document is saved, the character that identifies line break (a choice between LF and CRLF), and the programming language of the active document.

3.2.1.4 MongoDB

Mongo DB is a cross-platform document-oriented database program that used to be free and open source but has since changed to a proprietary license (Server Side Public License). Classified as a NoSQL database program, MongoDB uses JSON -like documents with schemata. MongoDB is developed by MongoDB Inc.

MongoDB supports field, range query, and regular expression searches. Queries can return specific fields of documents and also include user-defined JavaScript functions. Queries can also be configured to return a random sample of results of a given size.

MongoDB provides high availability with replica sets. A replica set consists of two or more copies of the data. Each replica set member may act in the role of primary or secondary replica at any time. All writes and reads are done on the primary replica by default. Secondary replicas maintain a copy of the data of the primary using built-in replication. When a primary replica fails, the replica set automatically conducts an election process to determine which secondary should become the primary. Secondary's can optionally serve read operations, but that data is only eventually consistent by default.

3.3 Data Design

3.3.1 Databases MongoDB

Name
StudentSupervision

Table 3:1 MongoDB Database

3.3.2 Tables

Name	Description
Users	Login credentials of the users.
Results	Contains Results.
Userdetails	Contains user profiles.
Notifications	Alert Messages.
File Upload	Stores the documents uploaded.

Table 3:2 List of Database Tables

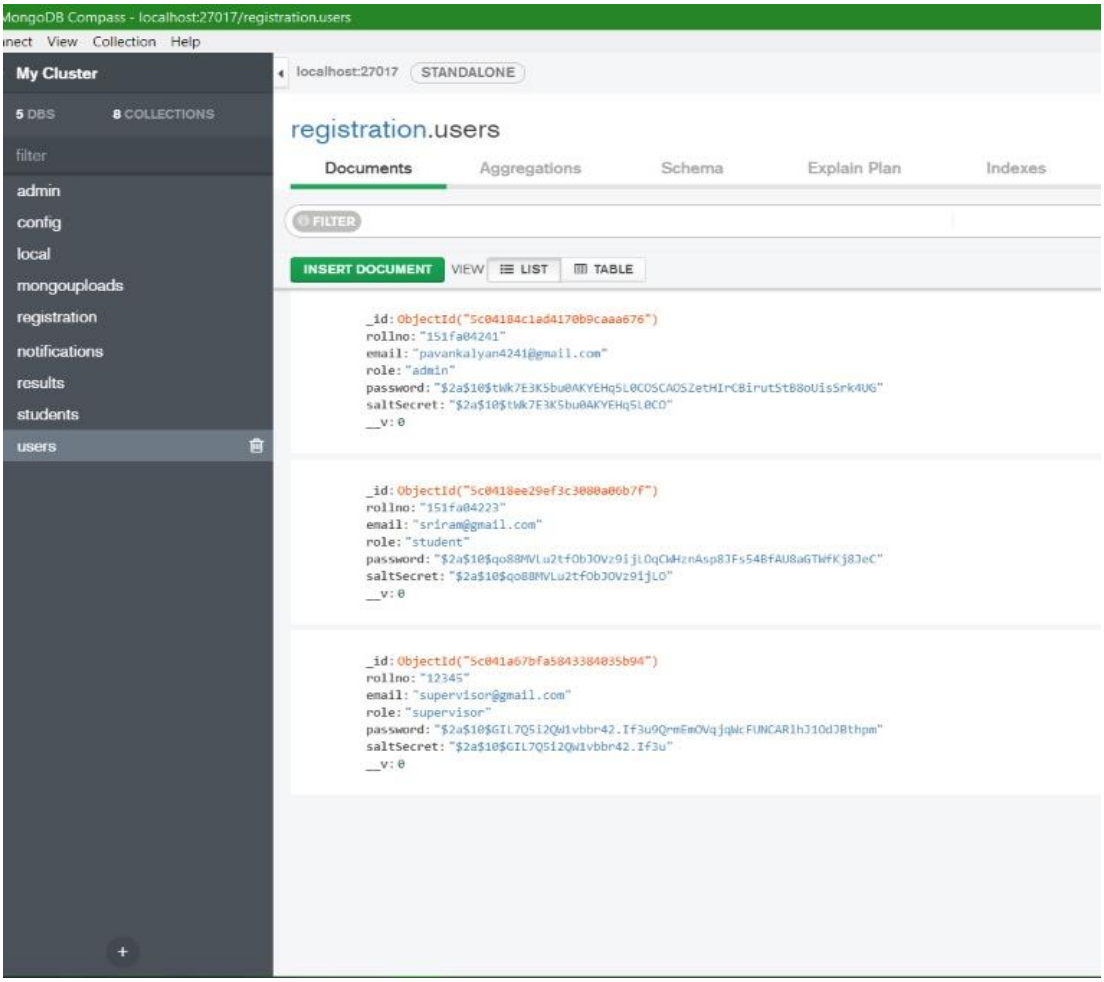

Fig. sample database table

CHAPTER - 4

MODELING

This chapter gives the unified modeling language diagrams.

4. MODELING

4.1 Design

Requirements gathering followed by careful analysis leads to a systematic Object Oriented Design (OOAD). Various activities have been identified and are represented using Unified Modeling Language (UML) diagrams. UML is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software-intensive system under development.

4.1.1. Use Case Diagram

In the Unified Modeling Language (UML), the use case diagram is a type of behavioral diagram defined by and created from a use-case analysis. It represents a graphical overview of the functionality of the system in terms of actors, which are persons, organizations or external system that plays a role in one or more interaction with the system. These are drawn as stick figures. The goals of these actors are represented as use cases, which describe a sequence of actions that provide something of measurable value to an actor and any dependencies between those use cases.

In this application there is only actor – soldier and below is the use case diagram of this application.

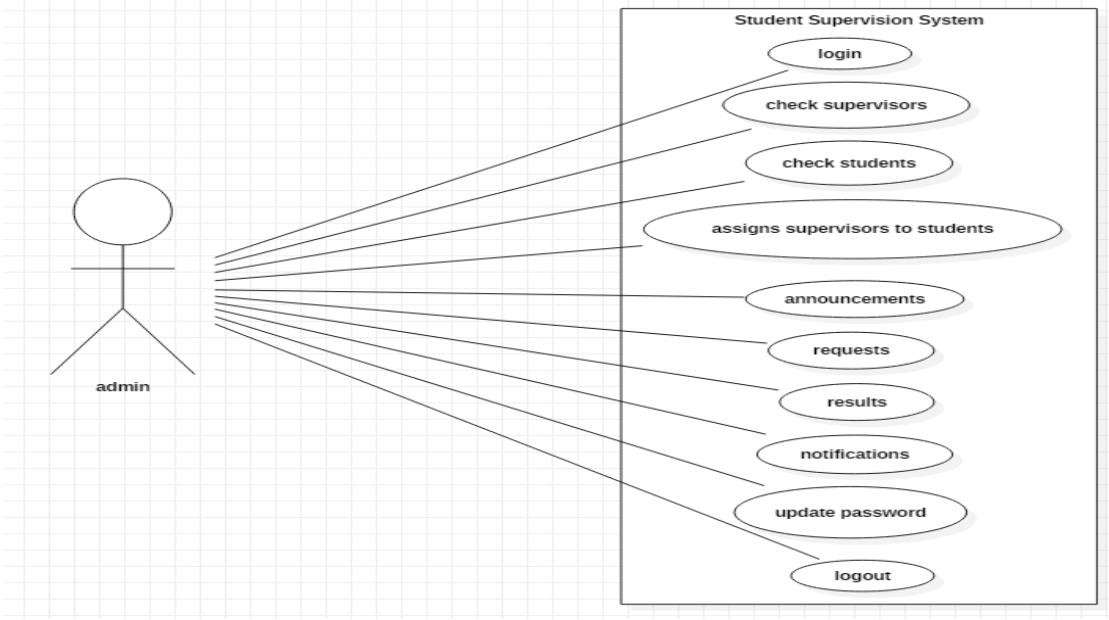

Figure 4-i Use Case Diagram for Admin

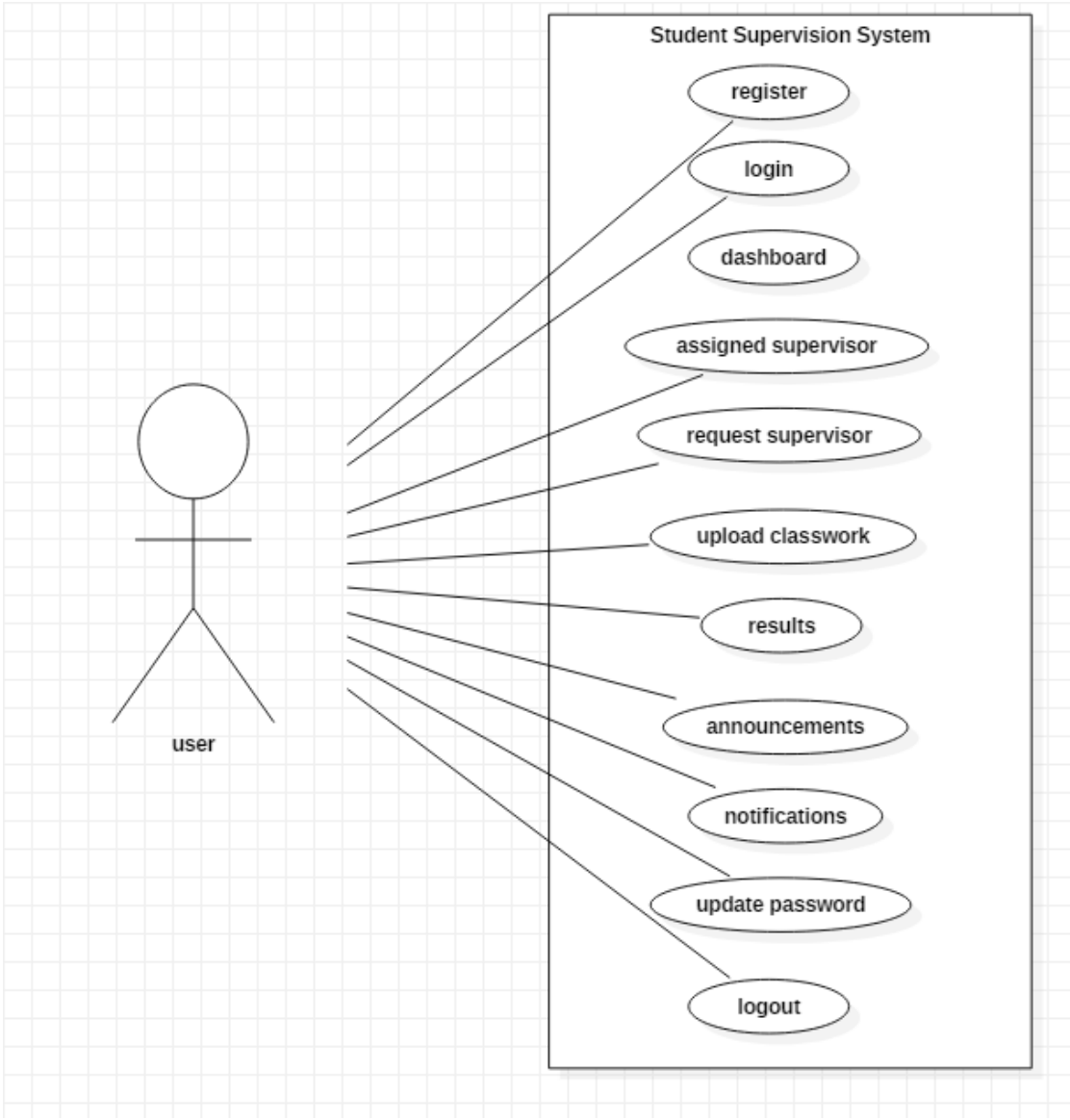

Figure 4-ii Use Case Diagram for User

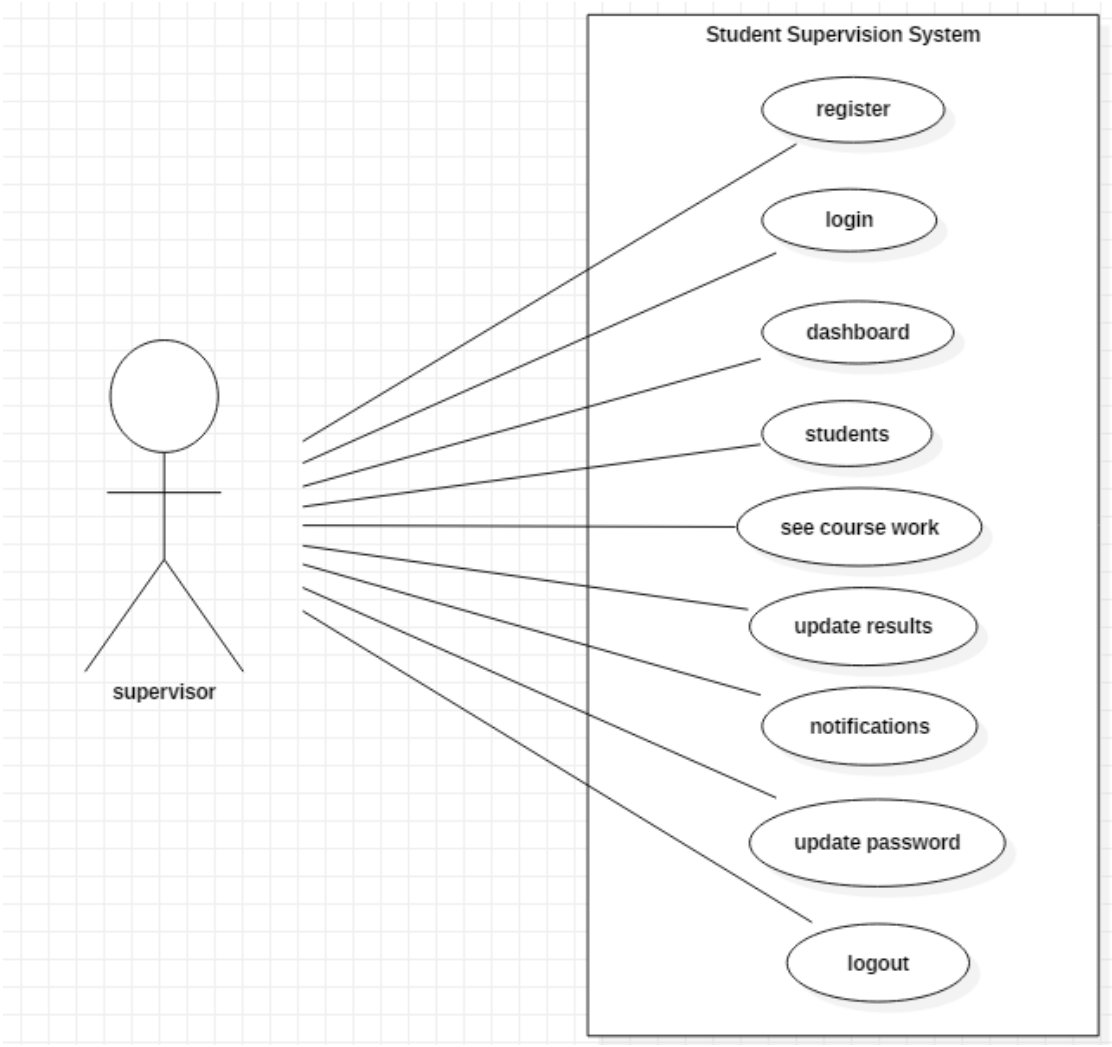

Figure 4-iii Use Case Diagram for Supervisor

4.1.2 Sequence Diagram

UML sequence diagrams are used to show how objects interact in a given situation. An important characteristic of a sequence diagram is that time passes from top to bottom: the interaction starts near the top of the diagram and ends at the bottom (i.e. Lower equals later).

A popular use for them is to document the dynamics in an object-oriented system. For each key, collaboration diagrams are created that show how objects interact in various representative scenarios for that collaboration.

Sequence diagram is the most common kind of interaction diagram, which focuses on the message interchange between a numbers of lifelines.

The following nodes and edges are typically drawn in a UML sequence diagram: lifeline, execution specification, message, combined fragment, interaction use, state invariant, continuation, destruction occurrence.

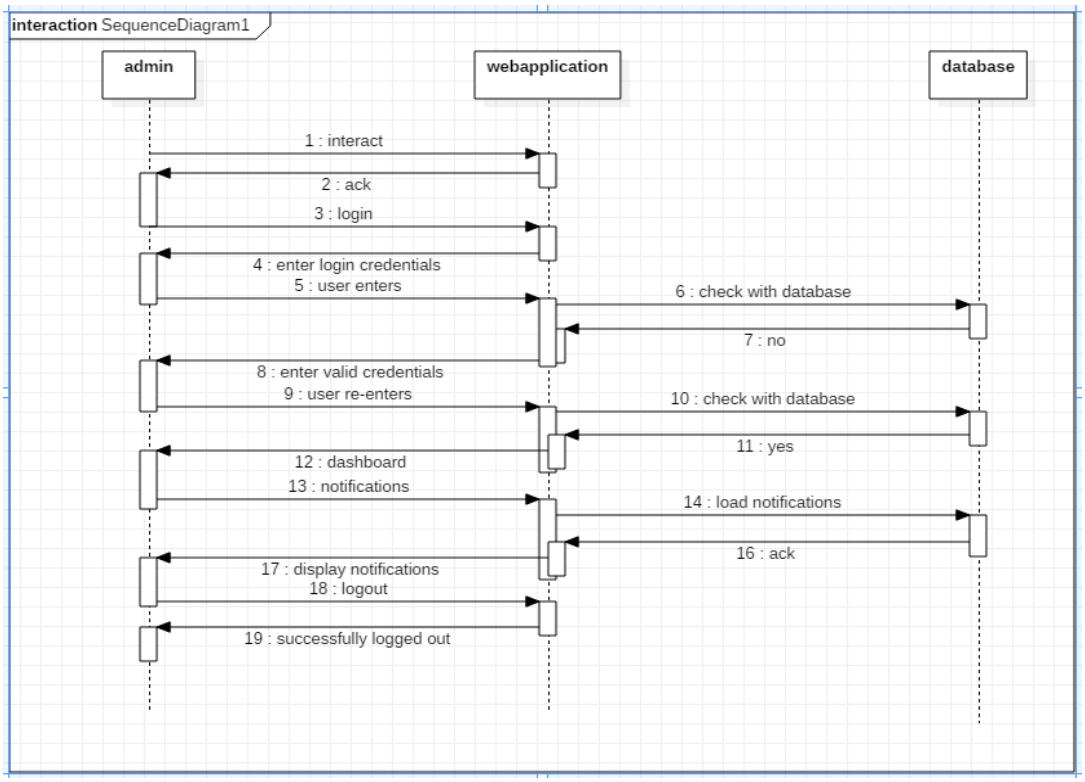

Figure 4-iv Sequence Diagram for Admin Module

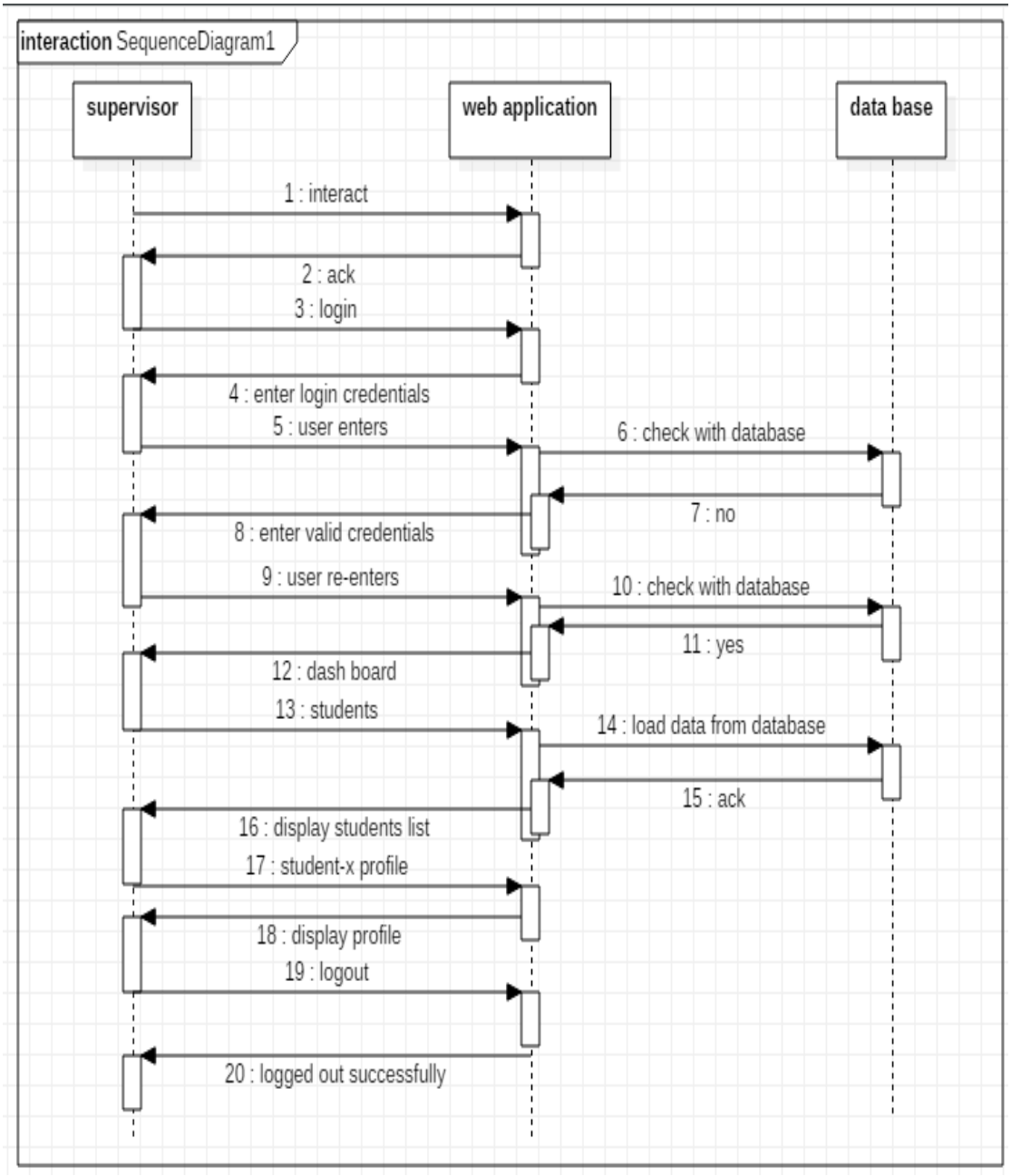

Figure 4-v Sequence Diagram for Supervisor

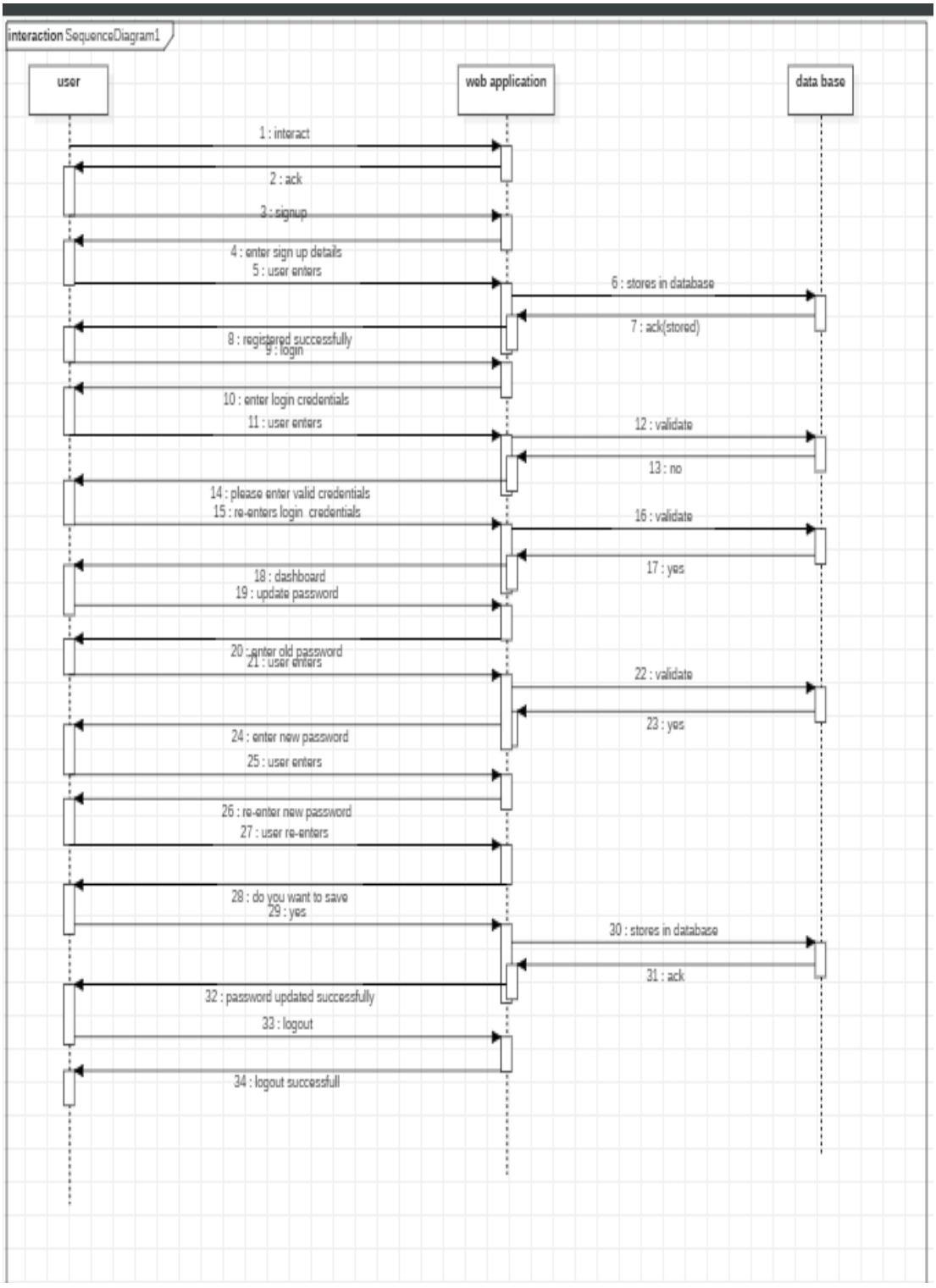

Figure 4-vi Sequence Diagram for User

4.1.3 Activity Diagram

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagrams deals with all type of flow control by using different elements like fork, join etc. Activity is a particular operation of the system.

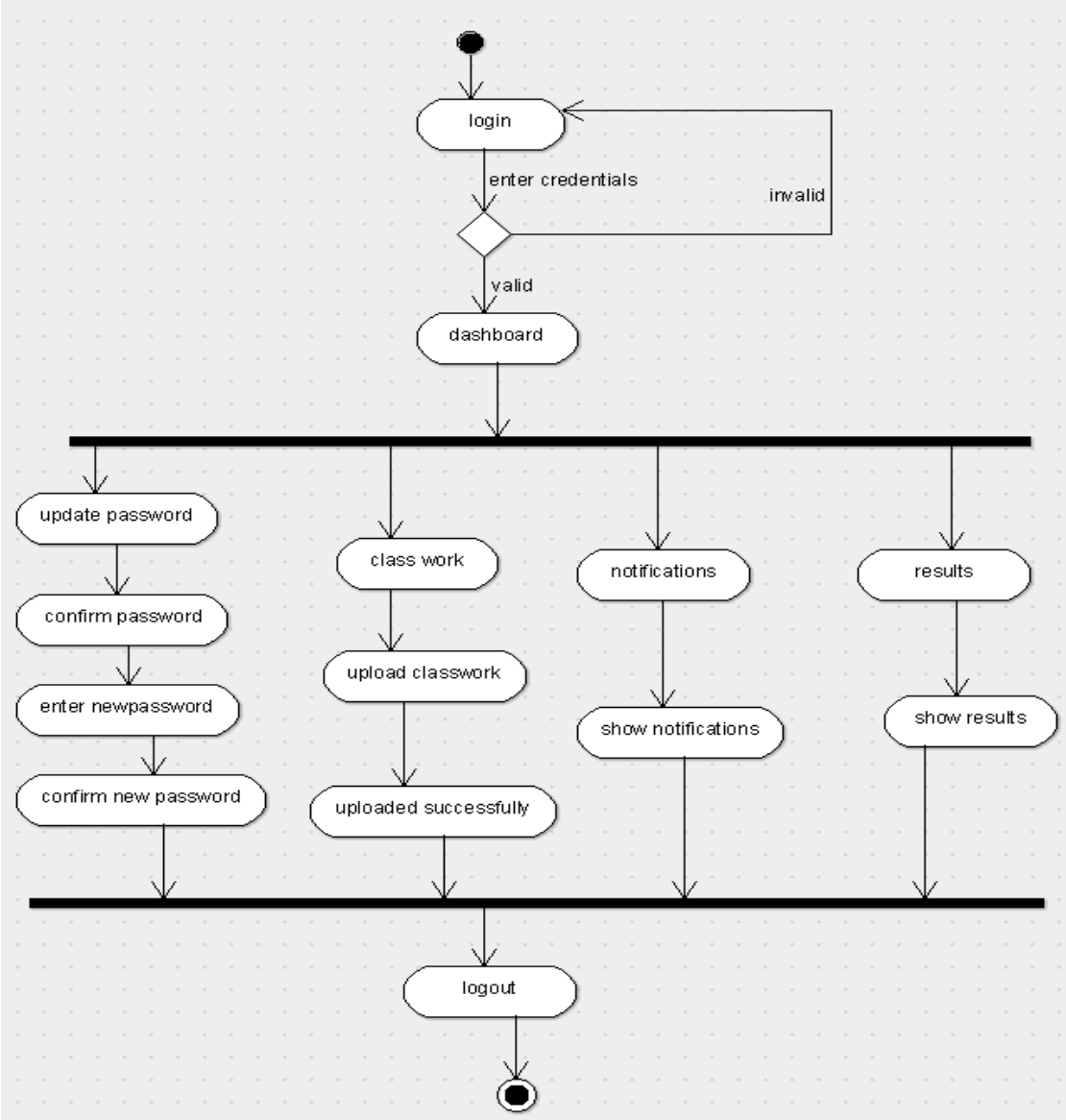

Figure 4-vii Activity Diagram

4.1.4 Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes.

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the 29modeling29c of the application and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed.

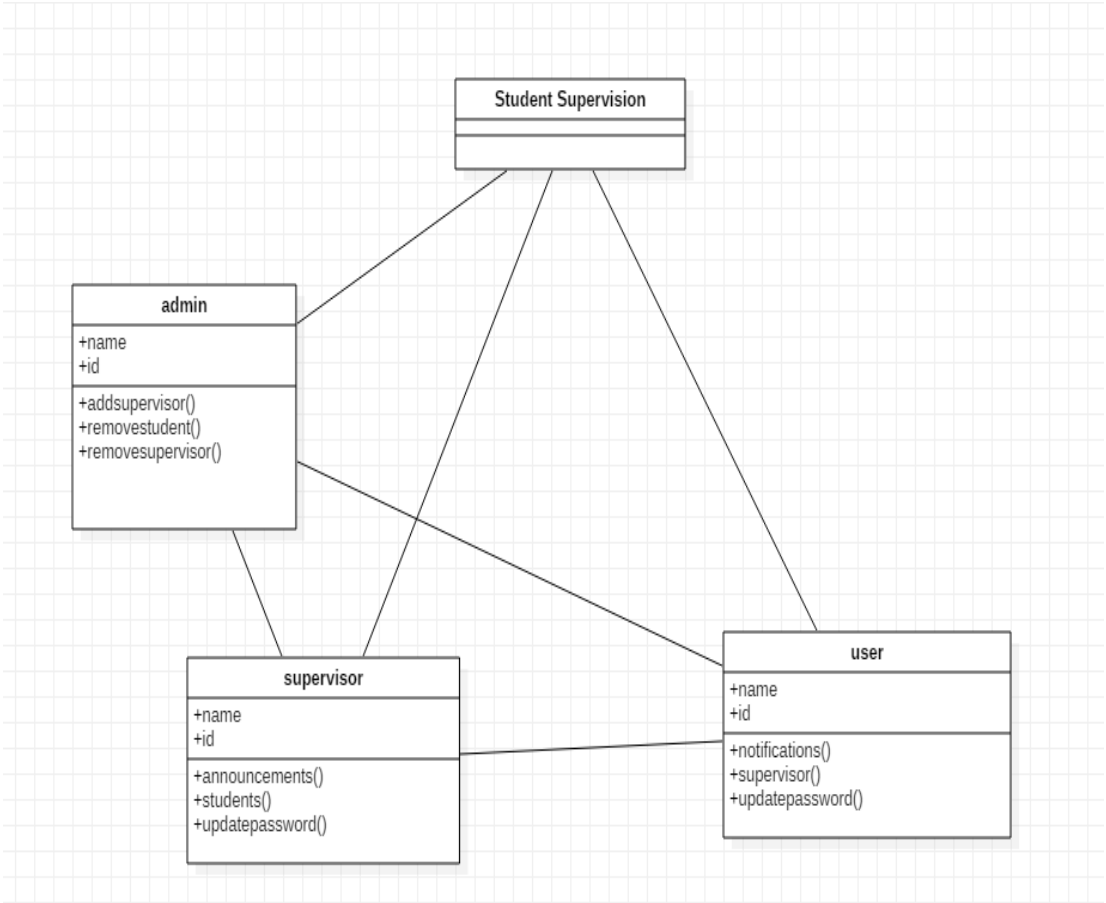

Figure 4-viii Class Diagram for SSS.

4.1.5 Deployment Diagram

Deployment diagram shows execution architecture of systems that represent the assignment (deployment) of software artifacts to deployment targets (usually nodes).

Nodes represent either hardware devices or software execution environments. They could be connected through communication paths to create network systems of arbitrary complexity. Artifacts represent concrete elements in the physical world that are the result of a development process and are deployed on nodes.

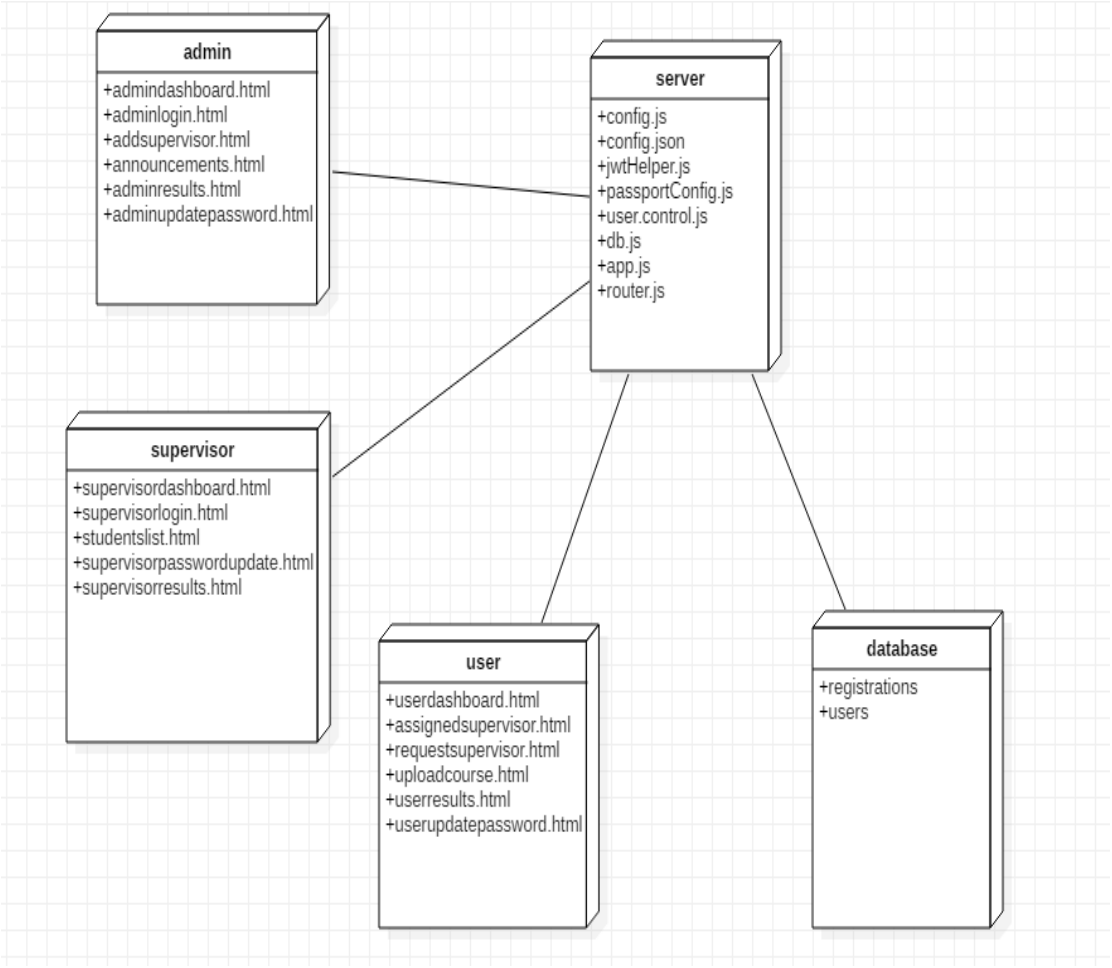

Figure 4-ix Deployment Diagram of the system

4.1.6 ER Diagram

An ER model is an abstract way to describe a database. Describing a database usually starts with a relational database, which stores data in tables. Some of the data in these tables point to data in other tables - for instance, your entry in the database could point to several entries for each of the phone numbers that are yours. The ER model would say that you are an entity, and each phone number is an entity, and the relationship between you and the phone numbers is 'has a phone number'. Diagrams created to design these entities and relationships are called entity-relationship diagrams or ER diagrams.

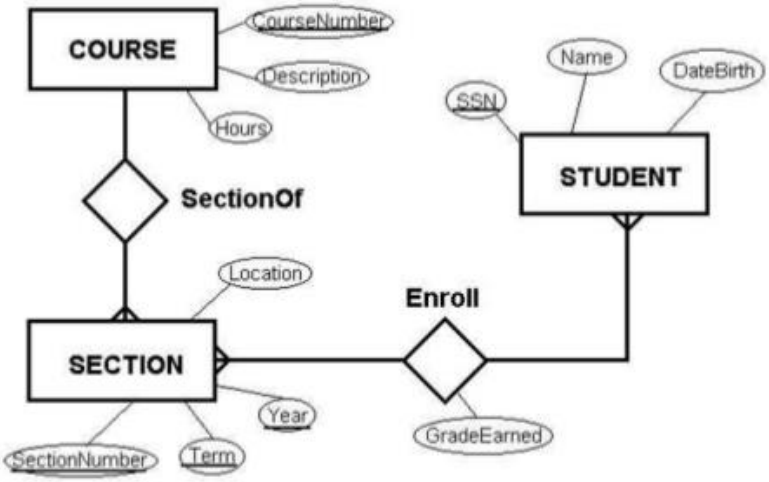

Figure 4-8 ER Diagram

CHAPTER - 5

IMPLEMENTATION

The chapter gives the details of the implementation.

5. IMPLEMENTATION

5.1 Sample Code

5.1.1 Code for SSS

SIGN UP

```
<form #signUpForm="ngForm" (ngSubmit)="signUpForm.valid && onSubmit(signUpForm)">
  <input type="text" #rollno="ngModel" [(ngModel)]="userService.selectedUser.rollno" name="rollno" placeholder="rollno" required [ngClass]="{'invalid-textbox' :signUpForm.submitted && !rollno.valid }">
  <div *ngIf="signUpForm.submitted && !rollno.valid">
    <label class="validation-message">This field is required.</label>
  </div>
  <input type="text" #email="ngModel" [(ngModel)]="userService.selectedUser.email" name="email" placeholder="Email" required [pattern]="emailRegex" [ngClass]="{'invalid-textbox' :signUpForm.submitted && !email.valid }">
  <div *ngIf="signUpForm.submitted && email.errors">
    <label *ngIf="email.errors.required" class="validation-message">This field is required.</label>
    <label *ngIf="email.errors.pattern" class="validation-message">Invalid email address.</label>
  </div>
  <input type="password" #password="ngModel" [(ngModel)]="userService.selectedUser.password" name="password" placeholder="Password" minlength="4" required [ngClass]="{'invalid-textbox' :signUpForm.submitted && !password.valid }">
  <div *ngIf="signUpForm.submitted && password.errors">
    <label *ngIf="password.errors.required" class="validation-message">This field is required.</label>
```

```
      <label *ngIf="password.errors.minlength" class="validation-message">Enter atleast 4 characters.</label>
    </div>
    <input type="submit" value="Sign Up">
</form>

<!-- Success message -->
<div class="success" *ngIf="showSucessMessage">
    Saved successfully
</div>

<!-- Error message -->
<div class="alert" *ngIf="serverErrorMessages">
    {{serverErrorMessages}}
</div>
```

SIGN UP COMPONENT

```
import { Component, OnInit } from '@angular/core';
import { NgForm } from '@angular/forms';
import { Router } from '@angular/router';

import { UserService } from '../../shared/user.service';

@Component({
  selector: 'app-sign-up',
  templateUrl: './sign-up.component.html',
  styleUrls: ['./sign-up.component.css']
})
export class SignUpComponent implements OnInit {
  // tslint:disable-next-line:max-line-length
  emailRegex =
/^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+)*)|(".+"))@((\[[0-
9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-
Z]{2,}))$/;
  showSucessMessage: boolean;
  serverErrorMessages: string;

  constructor(private userService: UserService, private router: Router) { }

  ngOnInit() {
  }

  onSubmit(form: NgForm) {
    this.userService.postUser(form.value).subscribe(
      res => {
```

```
        this.showSucessMessage = true;
        setTimeout(() => this.showSucessMessage = false, 2000);
        setTimeout(() => { this.router.navigate(['/login']); }, 2000);
        this.resetForm(form);
      },
      err => {
        if (err.status === 422) {
          this.serverErrorMessages = err.error.join('<br/>');
        } else {
          this.serverErrorMessages = 'Something went wrong.Please contact admin.';
        }
      }
    );
  }

  resetForm(form: NgForm) {
    this.userService.selectedUser = {
      rollno: '',
      email: '',
      password: ''
    };
    form.resetForm();
    this.serverErrorMessages = '';
  }

}
```

LOGIN

```
<form #signInForm="ngForm" (ngSubmit)="signInForm.valid && onSubmit(signInForm)">
   <input type="text" name="email" #email="ngModel" [(ngModel)]="model.email" placeholder="Email" [pattern]="emailRegex" required [ngClass]="{'invalid-textbox' :signInForm.submitted && !email.valid }">
   <div *ngIf="signInForm.submitted && email.errors?.pattern">
      <label class="validation-message">Invalid email address.</label>
   </div>
   <input type="password" name="password" #password="ngModel" [(ngModel)]="model.password" placeholder="Password" required minlength="4" [ngClass]="{'invalid-textbox' :signInForm.submitted && !password.valid }">
   <div *ngIf="signInForm.submitted && password.errors?.minlength">
      <label class="validation-message">Minimum 4 characters.</label>
   </div>
   <input type="submit" value="Sign In">
   <span (click)="forget()">Forget password?</span>
</form>
<!-- Error message -->
<div class="alert" *ngIf="serverErrorMessages">
   {{serverErrorMessages}}
</div>
```

LOGIN COMPONENT

```
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
import { NgForm } from '@angular/forms';
import { Router } from '@angular/router';

// tslint:disable-next-line:semicolon
import { UserService } from '../../shared/user.service'



@Component({
  selector: 'app-sign-in',
  templateUrl: './sign-in.component.html',
  styleUrls: ['./sign-in.component.css'],
  // encapsulation: ViewEncapsulation.None
})
export class SignInComponent implements OnInit {
  userDetails;
  constructor(private userService: UserService, private router: Router) { }

  model = {
    email : '',
    password : ''
  };
  // tslint:disable-next-line:max-line-length
  emailRegex     =     /^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+)*)|(".+"))@((\[[0-
9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/;
  showSucessMessage: boolean;
  serverErrorMessages: string;
  ngOnInit() {
    if (this.userService.isLoggedIn()) {
   this.router.navigateByUrl('/userprofile');
    }
  }
```

BACKEND FULL CODE

```
const mongoose = require('mongoose');
const passport = require('passport');
const _ = require('lodash');
var nodemailer = require("nodemailer");
const request = require('request');
var http = require("http");
const express = require('express');
const User = mongoose.model('User');
var ObjectId = require('mongoose').Types.ObjectId;
var { notification } = require('../models/notification.model.js');


module.exports.register = (req, res, next) => {
    console.log(req.body.rollno);
    var user = new User();
    user.rollno = req.body.rollno;
    user.email = req.body.email;
    user.role = 'student';
    user.password = req.body.password;
    user.save((err, doc) => {
        if (!err)
            res.send(doc);
        else {
            if (err.code == 11000)
                res.status(422).send(['Duplicate email adrress found.']);
            else
                return next(err);
        }

    });
};
```

```
module.exports.putnoti = (req, res, next) => {
    var noti = new notification();
    noti.head = req.body.head;
    noti.body = req.body.body;
    noti.save((err, doc) => {
        if (!err)
            res.send(doc);
        else {
            if (err.code == 11000)
                res.status(422).send(['Duplicate email adrress found.']);
            else
                return next(err);
        }

    });
};
module.exports.sturegister = (req, res, next) => {
    var user = new User();
    user.rollno = req.body.rollno;
    user.email = req.body.email;
    user.role = 'student';
    user.password = req.body.password;
    user.save((err, doc) => {
        if (!err)
            res.send(doc);
        else {
            if (err.code == 11000)
                res.status(422).send(['Duplicate email adrress found.']);
            else
                return next(err);
        }

    });
};
module.exports.supregister = (req, res, next) => {
    var user = new User();
    user.rollno = req.body.rollno;
```

```
    user.email = req.body.email;
    user.role = 'supervisor';
    user.password = req.body.password;
    user.save((err, doc) => {
        if (!err)
            res.send(doc);
        else {
            if (err.code == 11000)
                res.status(422).send(['Duplicate email adrress found.']);
            else
                return next(err);
        }

    });
};

module.exports.getnoti = (req, res) => {
    notification.find((err, docs) => {
        if (!err) { res.send(docs); } else { console.log('Error in Retriving Student :' +
JSON.stringify(err, undefined, 2)); }
    });
};

module.exports.authenticate = (req, res, next) => {
    // call for passport authentication
    passport.authenticate('local', (err, user, info) => {
        // error from passport middleware
        if (err) return res.status(400).json(err);
        // registered user
        else if (user) return res.status(200).json({ "token": user.generateJwt() });
        // unknown user or wrong password
        else return res.status(404).json(info);
    })(req, res);
};

module.exports.userprofile = (req, res, next) => {
    User.findOne({ _id: req._id },
```

```
        (err, user) => {
            if (!user)
                return res.status(404).json({ status: false, message: 'User record not found.' });
            else
                return res.status(200).json({ status: true, user: _.pick(user, ['rollno', 'email', 'role'])
});
        }
    );
};

module.exports.updatepassword = (req, res, next) => {
    var password = req.body.password;
    console.log(req.body.password);
    bcrypt.genSalt(10, (err, salt) => {
        bcrypt.hash(this.password, salt, (err, hash) => {
            req.body.password = hash;
            this.saltSecret = salt;
            next();
        });
    });
    console.log(req.body.password);

    User.findByIdAndUpdate(req.body._id, { $set: emp }, { new: true }, (err, doc) => {
        if (!err) { res.send(doc); } else { console.log('Error in student Update :' +
JSON.stringify(err, undefined, 2)); }
    });
};

var smtpTransport = nodemailer.createTransport({
    service: "Gmail",
    auth: {
        user: "pavankalyan4241@gmail.com",
        pass: "pavan.kalyan"
    }
});
var rand, mailOptions, host, link;
```

```
module.exports.send = (req, res, next) => {
   rand = Math.floor((Math.random() * 100) + 54);
   host = req.get('host');
   email = req.query.to;
   link = "http://" + req.get('host') + "/api" + "/verify?id=" + rand + "&email=" + req.query.to;
   mailOptions = {
      to: req.query.to,
      subject: "Please confirm your Email account",
      html: "Hello,<br> Please Click on the link to verify your email.<br><a href=" + link +
">Click here to verify</a>"
   };
   console.log(mailOptions);
   smtpTransport.sendMail(mailOptions, function(err, res) {
      if (err) {
         console.log(err);
         res.end("error");
      } else {
         console.log("Message sent: " + res.message);
         res.end("sent");
      }
   });
};

module.exports.verify = (req, res, next) => {
   rands = Math.floor((Math.random() * 100) + 54);
   console.log(req.protocol + ":/" + req.get('host'));
   if ((req.protocol + "://" + req.get('host')) == ("http://" + host)) {
      console.log("Domain is matched. Information is from Authentic email");
      if (req.query.id == rand && req.query.email == email) {
         res.render("update/updatepassword", {
            viewTitle: req.query.email
         });
      } else {
         console.log("email is not verified");
         res.end("<h1>Bad Request</h1>");
      }
   } else {
```

```
        res.end("<h1>Request is from unknown source");
    }
};

module.exports.updatepassword = (req, res, next) => {
    return res.status(422).send(['Duplicate email adrress found.']);
};


// recap
module.exports.sub = (req, res) => {
    var homepage = 'homepage';
    console.log("homepage");
    if (
        req.body.success == false
    ) {
        return res.status(422).send(['false']);
    } else if (
        req.body.score == 0.0
    ) {
        return res.status(422).send(['score is 0.0']);
    } else if (
        req.body.action == homepage
    ) {
        return res.status(422).send(['action not valid']);
    }

    // Secret Key
    const secretKey = '6Lc9jXgUAAAAAB-99dmv4dbGVXcJLeMzBizqZQWB';

    // Verify URL
    const verifyUrl = `https://google.com/recaptcha/api/siteverify?secret=${secretKey}&response=${req.body.captcha}&remoteip=${req.connection.remoteAddress}`;

    // Make Request To VerifyURL
    request(verifyUrl, (err, response, body) => {
```

```
        body = JSON.parse(body);
        console.log(body);

        // If Not Successful
        if (body.success != false && !body.success) {
            return res.status(422).send(['Failed captcha verification']);
        }

        //If Successful
        return res.status(200).send(['captcha verification']);
    });
};



module.exports.userdet = (req, res) => {
    User.find({ role: req.query.super }, (err, docs) => {
        if (!err) { res.send(docs); } else { console.log('Error in Retriving Supervisors :' + JSON.stringify(err, undefined, 2)); }
    });
};
module.exports.deluser = (req, res) => {
    User.findByIdAndRemove(req.query._id, (err, doc) => {
        if (!err) { res.send(doc); } else { console.log('Error in Student Delete :' + JSON.stringify(err, undefined, 2)); }
    });
};




// Github
module.exports.github = () => {
    passport.authenticate('github')
};

module.exports.callback = () => {
```

```
    passport.authenticate('github', { failureRedirect: '/failure' }),
    authController.sendJWTToken
};


module.exports.authenticates = () => {
    passport.authenticate('jwt', { session: false }), authController.authenticate
};


module.exports.logout = () => {
    passport.authenticate('jwt', { session: false }), authController.logout
};
```

5.2 Screen Captures

5.2.1 Login Screen

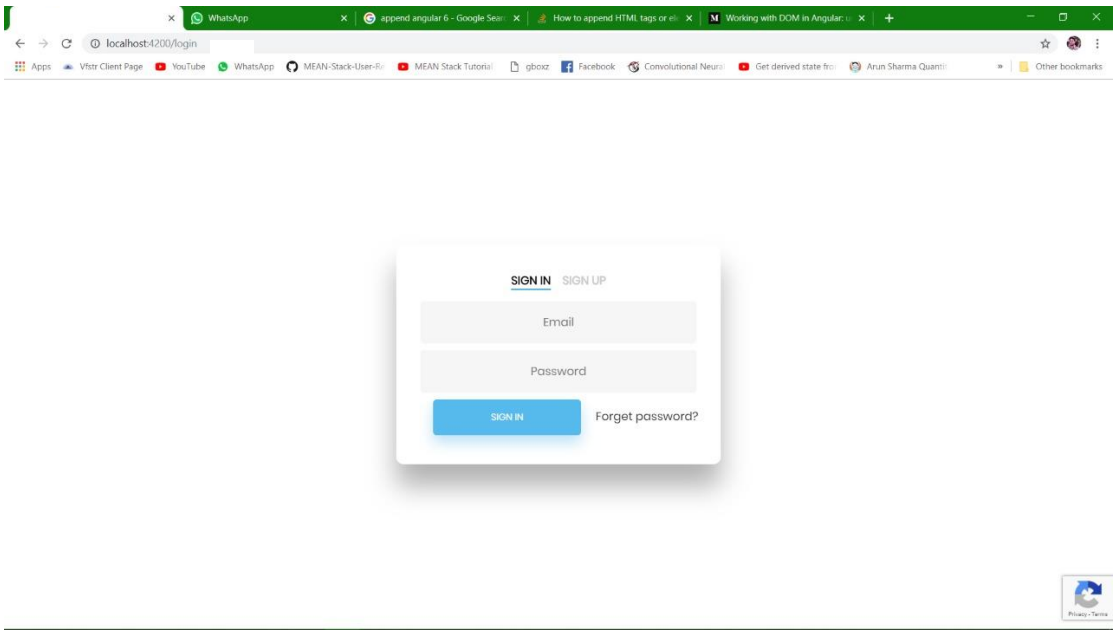

Figure 5-i Login Activity

Description: The application has a login which is used by both supervisor and student.

5.2.2 Sign Up Screen

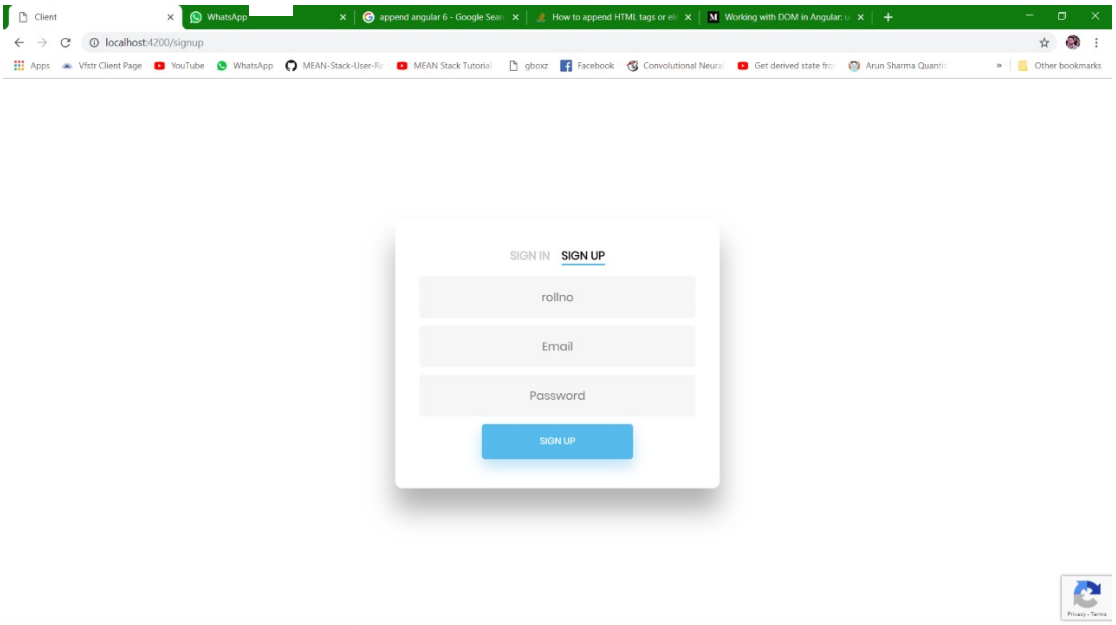

Figure 5-ii Sign up Screen.

Description: The application has a different signup for supervisor and student. The needs to contact Student for login credentials.

5.2.4 Home Screen

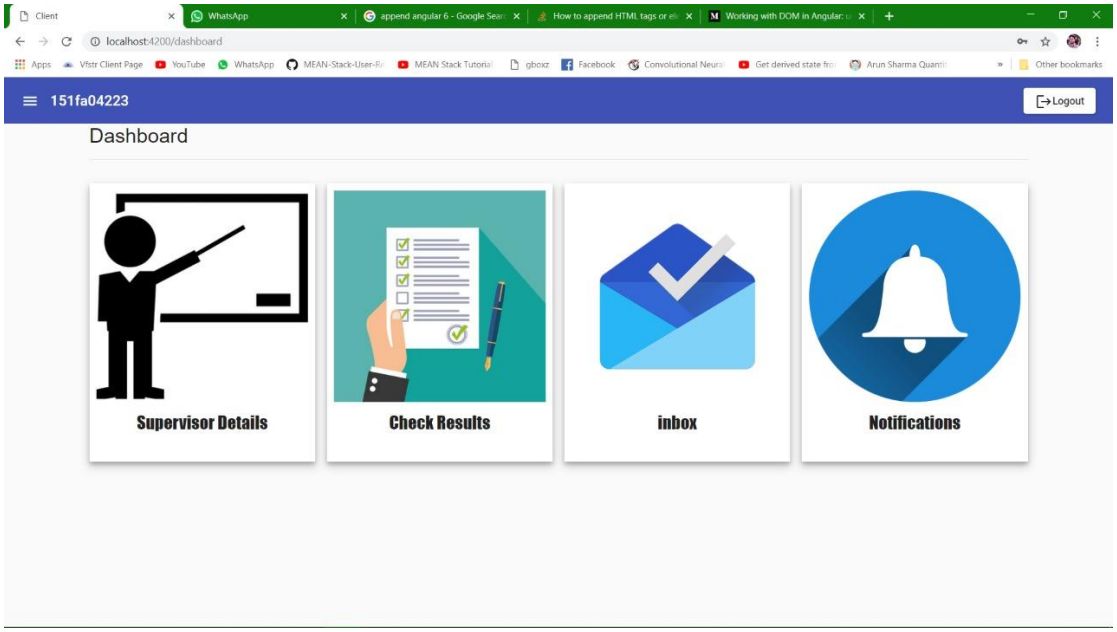

Figure 5-iii Home Screen Activity

Description: The Home Screen Activity contains all the links like supervisor, results, notifications, inbox, logout. On clicking, we can move to particular page and perform operations.

5.2.6 Notifications

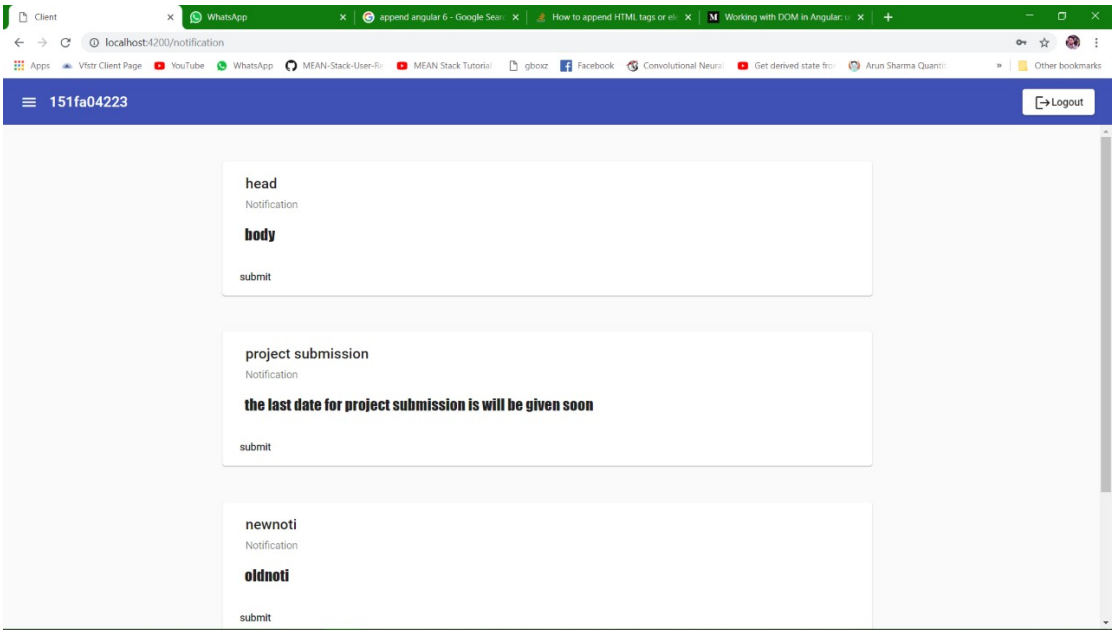

Figure 5-iv notifications on screen.

Description: Get notification information on the screen.

5.2.7 Results

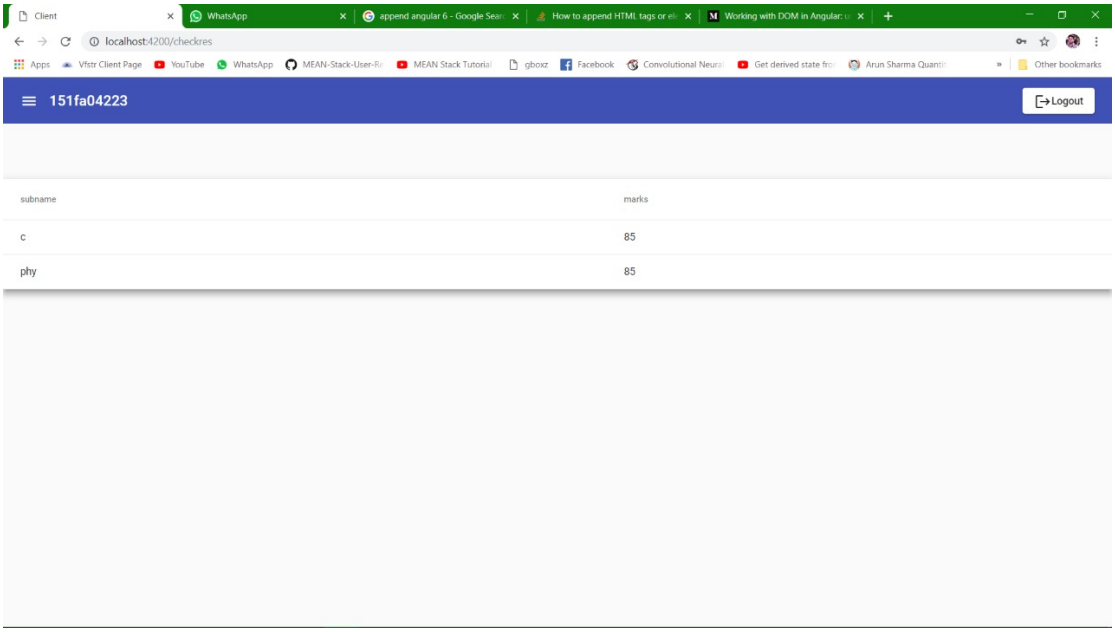

Figure 5-v Results screen.

Description: Results can be viewed by the supervisor and the student on screen.

5.2.8 File Upload

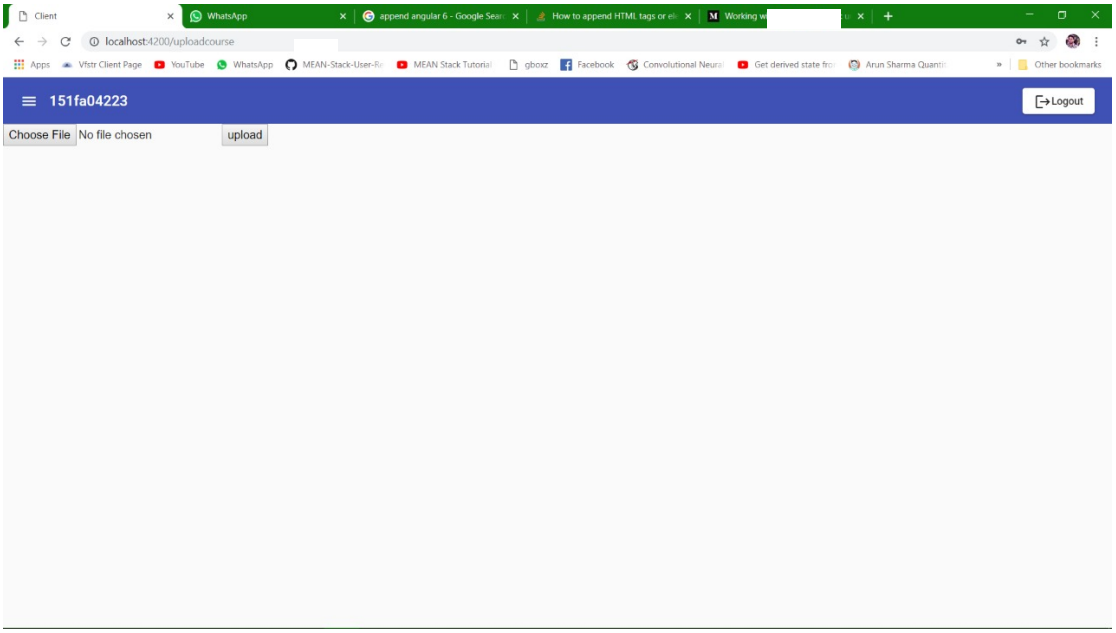

Figure 5-vi File Upload screen

Description: Supervisor and Students can send course work each other by uploading the document.

CHAPTER - 6

TESTING

The chapter shows the various test cases.

6. TESTING

6.1 Software Testing

Software testing is the process of validating and verifying that a software application meets the technical requirements which are involved in its design and development. It is also used to uncover any defects/bugs that exist in the application. It assures the quality of the software. There are many types of testing software viz., manual testing, unit testing, black box testing, performance testing, stress testing, regression testing, white box testing etc. Among these performance testing and load testing are the most important one for an android application and next sections deal with some of these types.

6.2 Black box Testing

Black box testing treats the software as a "black box"—without any knowledge of internal implementation. Black box testing methods include equivalence partitioning, boundary value analysis, all-pairs testing, fuzz testing, model-based testing, traceability matrix, exploratory testing, and specification-based testing.

6.3 White box Testing

White box testing is when the tester has access to the internal data structures and algorithms including the code that implement these.

6.4 Performance Testing

Performance testing is executed to determine how fast a system or sub-system performs under a particular workload. It can also serve to validate and verify other quality attributes of the system such as scalability, reliability and resource usage.

6.5 Load Testing

Load testing is primarily concerned with testing that can continue to operate under specific load, whether that is large quantities of data or a large number of users.

6.6 Manual Testing

Manual Testing is the process of manually testing software for defects. Functionality of this application is manually tested to ensure the correctness. Few examples of test case for Manual Testing are discussed later in this chapter.

Test Case 1	
Test Case Name	Empty Signup fields testing
Description	In the Signup screen if the username and password fields are empty.
Output	Signup fails showing an alert “This field is required”.

Table 6:1 Test Case for Empty Signup Fields

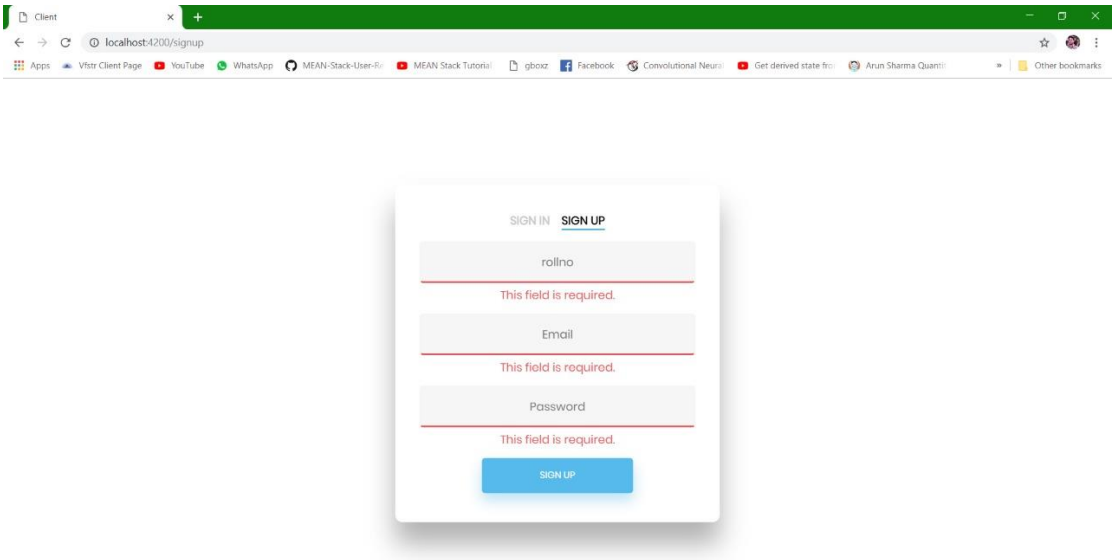

Figure 6-i Test Case for Empty Signup Fields

Test Case 2	
Test Case Name	Wrong login fields testing
Description	A unique username and password are required. On entering wrong username or password gives.
Output	Login fails showing an alert “Enter email/password entered is wrong”.

Table 6:2 Test Case for Wrong Login Fields

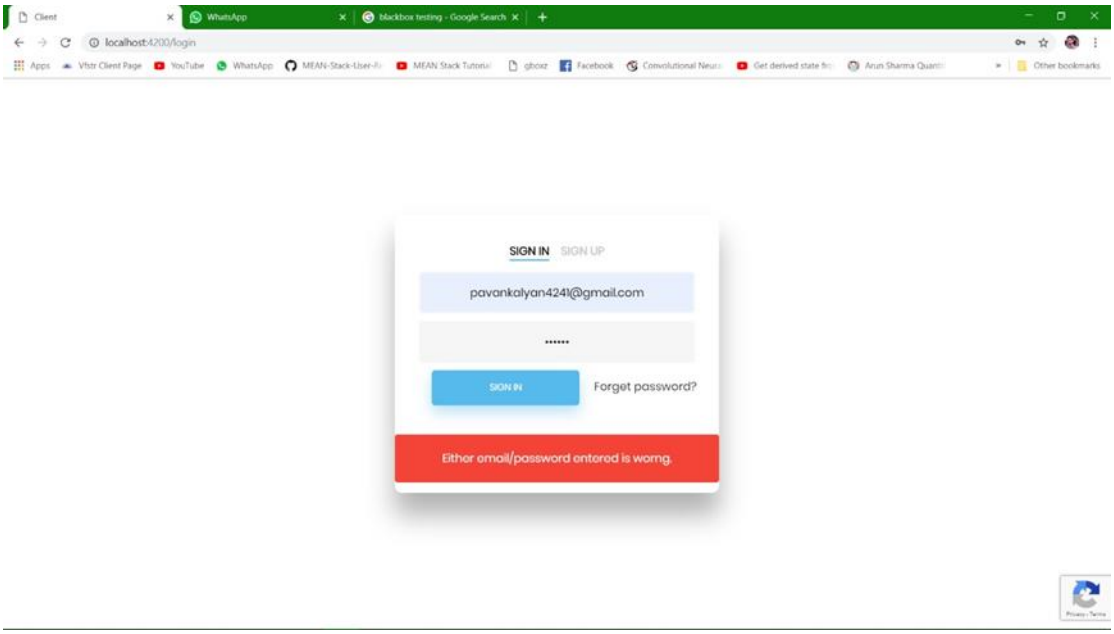

Figure 6-ii Test Case for Wrong Login Fields

Test Case 3	
Test Case Name	Signup Fails.
Description	Signup need to provide all data.
Output	Signup Fails and a alert “Invalid email address”.

Table 6:3 Test Case for Signup fail

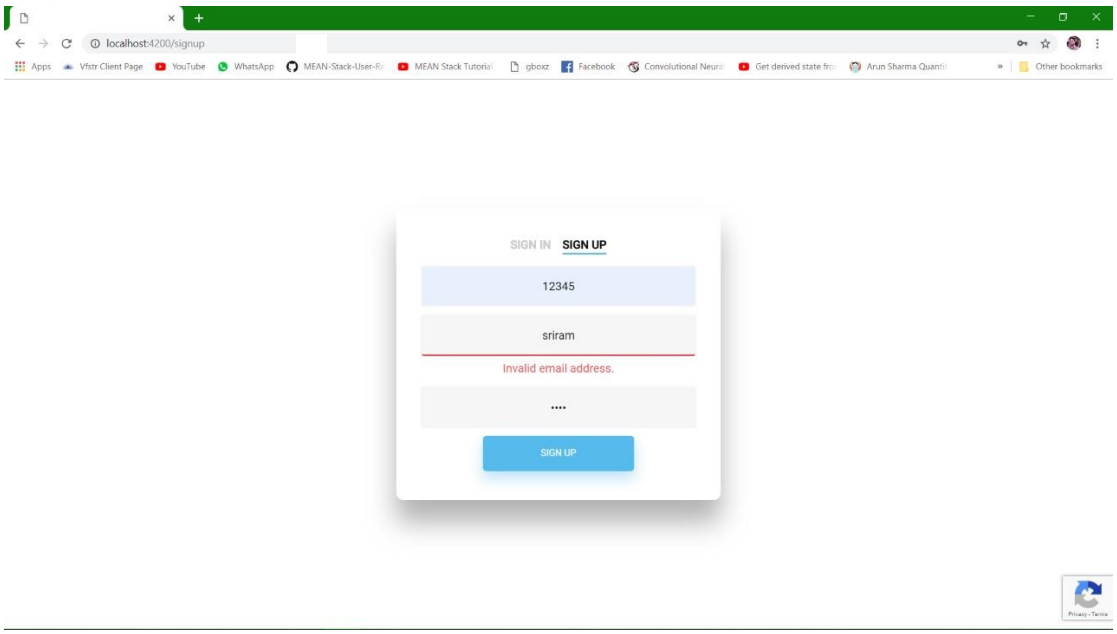

Figure 6-iii Signup fail

Test Case 4	
Test Case Name	Duplicate Signup
Description	Can’t perform signup operation with already registered email.
Output	Alert message shown as “Duplicate email address found”.

Table 6:4 Test Case for Duplicate signup

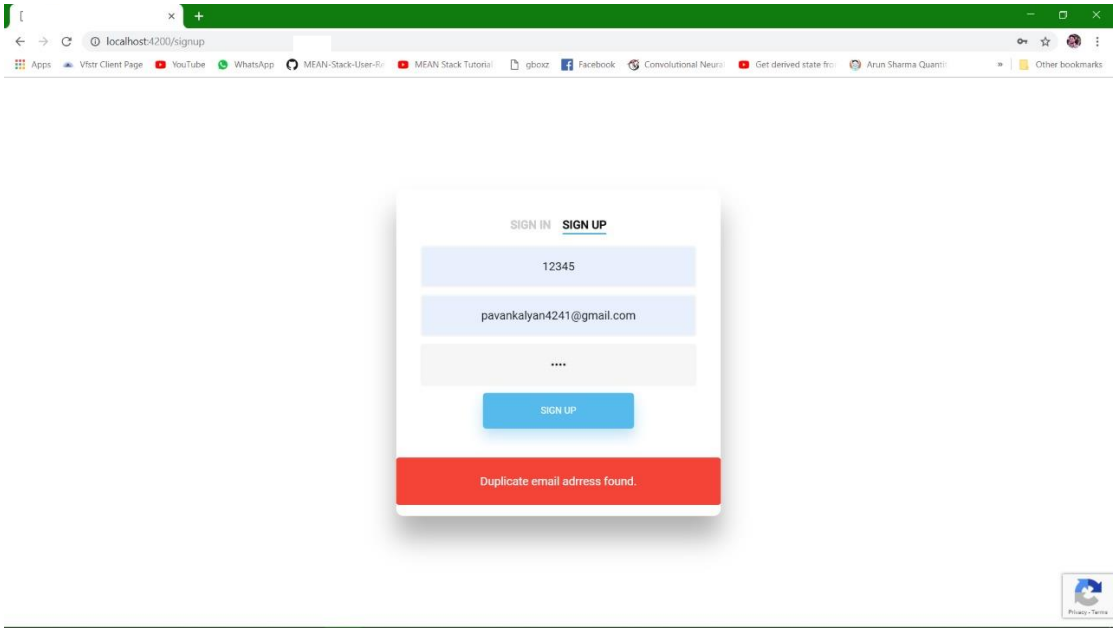

Figure 6-4 Test Case for Duplicate signup.

CHAPTER - 7

RESULTS &CHALLENGES

The chapter describes the results and challenges faced in the project.

7. RESULTS AND CHALLENGES

7.1 Results

The current application is developed using Html, Angular 6, MongoDB, Node.js. It can be used by small companies for maintaining the Company, Supervisor data and to maintain timesheet, expenses, payroll, share the information.

At the time of submission of my application was capable of doing the following:

- Displaying a welcome screen
- Authentication of user by using login screen
- Home screen to display based on Supervisor or student.
- After successful login of User, they can update course work, results and notifications regarding events, meetings.
- Add, update, view, delete the User details.
- Admin can view the remaining User details, edit their own information.
- Supervisor can update results, course work and view the status in their dashboard.
- User can edit the their details.
- Logout and end the session.

7.2 Challenges

- Understanding the client requirements was one of the crucial tasks of the whole project.
- Graphics User Interface (GUI) design was a difficult task as there are many types of devices like desktop, tablet, mobile with varying screen size and resolutions.
- Implementing synchronization with server was a challenging task.
- Learning different technologies and frameworks with little guidance.

CHAPTER - 8

CONCLUSIONS & FUTURE WORK

The chapter gives brief introduction of the project.

8. CONCLUSION

8.1 Conclusions

The application has been designed successfully to meet all the user requirements. I found this project to be far more difficult than I ever anticipated. Without doubt, this has been the most challenging and at the same time rewarding programming project I have undertaken since I started college.

8.2 Scope for future work

The application can further be modified in the following ways:

- Finish the form validations for entire application.
- Introduce a feature to call contacts directly from within the application.
- Integrate with built-in SMS, and Email.
- Fix all existing bugs

8.3 Limitations

The current application is with lack of validations. It can take input data without proper validations. The user log need to be fix in the server. Forgot password need to be add and as well change password also. Document verification need to be add.

BIBLIOGRAPHY

Code snippets for Node.js development

http://stackoverflow.com/

Node.js Development Guide

https://www.w3schools.com/nodejs/

Angular 6 Guide

https://www.w3schools.com/

Connection between Node.js and MongoDB

https://www.w3schools.com/nodejs_mongodb_connect.asp

Software Testing

http://en.wikipedia.org/wiki/Software_testing

Manual Testing

http://en.wikipedia.org/wiki/Manual_testing

Performance Testing

http://en.wikipedia.org/wiki/Software_performance_testing