

SPORTS EVENT MANAGEMENT SYSTEM

Submitted by

MATHIKUMAR B

1P21MC031

In partial fulfillment of the requirements for the award of the Degree of

MASTER OF COMPUTER APPLICATIONS

From Bharathiar University, Coimbatore.

Under the internal supervision of

DR .G.PURUSOTHAMAN, M.Sc(IT), M.Phil(CS), Ph.D(CS)

Associate Professor



SCHOOL OF COMPUTER STUDIES

MASTER OF COMPUTER APPLICATIONS

RATHNAVEL SUBRAMANIAM COLLEGE OF ARTS AND SCIENCE

(AUTONOMOUS)

Sulur, Coimbatore – 641 402.

April 2023.

**RATHNAVEL SUBRAMANIAM COLLEGE OF ARTS AND
SCIENCE (AUTONOMOUS)**
Sulur, Coimbatore – 641 402.

**School of Computer Studies
Department of Computer Applications (MCA)**



April 2023.

Register Number: 1P21MC031

Certified bona fide original record work done by
MATHIKUMAR B

Guide

HoD

Submitted for the project Evaluation and Viva voce held on

Internal Examiner

External Examiner

DECLARATION

I, **MATHIKUMAR B**, hereby declare that the project entitled **SPORTS EVENT MANAGEMENT SYSTEM**, submitted to the School of Computer Studies, Rathnavel Subramaniam College of Arts and Science, in partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications is a record of original project work done by me during the period Jan 2023 to April 2023 under the internal supervision of **DR .G.PURUSOTHAMAN, M.Sc., (IT), PGDE-com., M.Phil.,(CS), Ph.D.,(CS) , ASSOCIATE PROFESSOR, RATHNAVEL SUBRAMANIAM COLLEGE OF ARTS AND SCIENCE (AUTONOMOUS)** From Bharathiar University, Coimbatore.

Signature of the Candidate

ACKNOWLEDGEMENTS

I express my sincere thanks to our Managing trustee **Dr.K.Senthil Ganesh MBA (USA)., MS (UK)., Ph.D.**, for providing us with the adequate faculty and laboratory resources for completing my project successfully.

I take this as a fine opportunity to express my sincere thanks to **Dr. T. Sivakumar M.Sc., M. Phil., Ph.D., Principal**, Rathnavel Subramaniam College of Arts And Science (Autonomous) for giving me the opportunity to undertake this project.

I express my sincere thanks to **Dr. P. Navaneetham M.Sc., M.Phil., Ph.D., Director (Administration), Department of Computer Science** for the help and advice throughout the project.

I express my sincere thanks to **Dr. S. Yamini M.Sc.,(CC)., M. Phil., Ph.D., Director (Academic), Department of Computer Science** for her valuable guidance and prompt correspondence throughout the curriculum to complete the project.

I express my sincere thanks to **Dr. C. Grace Padma MCA., M.Phil., MBA., Ph.D., Head of the Department(MCA)** for her support and advice throughout the project.

I express my gratitude to **DR .G.PURUSOTHAMAN, M.Sc., (IT), PGDE-com., M.Phil., (CS), Ph.D.,(CS)., Associate Professor (MCA)** for his valuable guidance, support, encouragement and motivation rendered by her throughout this project.

Finally, I express my sincere thanks to all other staff members and my dear friends, and all dear and near for helping me to complete this project.

MATHIKUMAR B

ABSTRACT

Sports Event Management Project brings the entire manual time-consuming Process of Event Management online. This project aims to simplify handling each sports event by providing a web interface for Admin and Students. This Project is built using FULL STACK (MERN)

The Master module consists of multiple modules to initiate with the sports event by staff adding the type of sports details, Editing the details if any updates or correction, viewing the report of the student's name list who are all registered for a sports event, adding the results of sports event held in college and adding the training schedule for the students registered in the event. The student part has come up with a home page that displays the event images. The event page has all the sports event information's (event name, date, time, register button) registration page the student can enter the details and register for the event, result page shows the results of sports event held in college, training schedule page has all the training detail and attendance for Student and finally about us page has the details of the college. The main desire our project is to establish a bridge between students and staff. So, there is no intermediate person to convey all information, students to get correct and quick updates regarding sports

CONTENTS

Declaration

Acknowledgements

Abstract

CHAPTER 1

1.Introduction

1.1 An overview of the project	1
1.2 Mission of the project	1
1.3 Background study	1
1.3.1 A study of the existing system	1

CHAPTER 2

2.System Analysis

2.1 A study of the proposed system	3
2.2 User requirement specification	3
2.2.1 Major modules	3
2.2.2 Sub modules	3
2.3 Software requirement specification	
2.3.1 Document Purpose	4
2.3.2 Product Scope	4
2.3.3 Design and Implementation	4
2.4 System specification	
2.4.1 React.js	4
2.4.2 Node.js	4
2.4.3 Hardware configuration	5
2.4.4 Software configuration	5

CHAPTER 3

3.System design and development

3.1 Fundamentals of design concepts	7
3.1.1 Abstraction	7

3.1.2 Refinement	7
3.1.3 Modularity	7
3.2 Design Notations	
3.2.1 System structure chart	9
3.2.2 System flow diagram	10
3.2.3 Data flow diagram / UML	12
3.2.4 Software Engineering Model	13

CHAPTER 4

4. Testing and implementation	
4.1 Testing	16
4.1.1 System testing	16
4.1.2 White box testing	16
4.1.3 Unit testing	17
4.2 System implementation	18
4.2.1 System maintenance	18

CHAPTER 5

5. Conclusion	
5.1 Directions for future enhancements references	20
BIBLIOGRAPHY	20

ANNEXURE

ANNEXURE - A - Output design	55
ANNEXURE - B - Source code	74

CHAPTER 1

INTRODUCTION

1.1 AN OVERVIEW OF THE PROJECT

Sports Event Management System (SEMS) is a Web Application that entire manual time-consuming Process of Event Management online dynamically. This project aims to simplify handling each sports event by providing a web interface for Admin and Students.

SEMS has a dynamic entries so that students can easily view the ongoing sports updates when an admin can create a sport event.

SEMS also have some other features like filter students by department wise so that event organizer view the registered students by department wise.

1.2 MISSION OF THE PROJECT

Students can apply to participate in sports online without any hassle. An Event organizer (admin) can view the registered student on the admin dashboard.

1.3 BACKGROUND STUDY

Students are satisfied with the Sports because they get value for money and are pressed with various facilities. Playground as well as the playing environment attracts students.

1.3.1 A STUDY OF EXISTING SYSTEM

The existing way to participate in sports event is the organizer announces in notice board and collecting the participating students list by asking them. Students try to catchup with the organizer to enroll his/her name in the participating students list.

CHAPTER 2

SYSTEM ANALYSIS

2.1 A STUDY OF PROPOSED SYSTEM:

The proposed system is an enhancement or an upgrade to an already existing system, this system can be used as an online registration for any sports event conducted by college.

The proposed system will overcome each and every single problem faced by existing system by reduce the paper notices

2.2 USER REQUIREMENT SPECIFICATION:

The requirements for the player are that they have to apply through the online application so easily and quickly. The player must have an account to login in otherwise they can easily create an account with in a second

2.2.1 MAJOR MODULES:

- Eventer Module
- Student Module

2.2.2 SUB-MODULES:

- Landing Page
- Common Login Module
- Student Register
- Profile Module
- Events Module
- Department Module

2.3 SOFTWARE REQUIREMENT SPECIFICATION

2.3.1 DOCUMENT PURPOSE

The software requirement specification will describe the processes and functions of the SEMS. The major portion of the product will be describe within this documentation, with a possible upgrade to the system

2.3.2 PRODUCT SCOPE

The SEMS provides the system work to reduce the manual work for both event creators and participants.

Eventer

Students

Each of them should have a separate module to use and has to authenticate as well as authorized each user by checking their roles. Both have a common login, even if the student tries to access the eventer login, it shows a message as you are not an authorized person.

2.3.3 DESIGN AND IMPLEMENTATION

SEMS is a multi-tenancy web application, it will be used by wide range of people as well as students, so , the design must be user friendly so that anybody can use the web application without the help or support.

2.4 SYSTEM REQUIREMENT SPECIFICATION

2.4.1 REACT.JS

This project is developed using a react (A JavaScript-based UI) development library as frontend with MUI, React strap, bootstrap as CSS.

2.4.2 NODE.JS

This project is developed using Node as backend through which the database is mongo DB Atlas cluster

2.4.3 HARDWARE CONFIGURATION

System	:	Any
Processor	:	Intel core i3 (min) / Ryzen 3 (min)
Processor Speed	:	2.80 GHz (min)
Main Storage	:	4 GB (min)
Hard Disk Capacity	:	512 GB (min)

2.4.4 SOFTWARE CONFIGURATION

Operating System	:	Windows / MAC / Ubuntu / Linux
Editor	:	Visual Studio Code
Frontend	:	React 17
Backend	:	Node 15 16 17

CHAPTER 3

SYSTEM DESIGN AND DEVELOPMENT

3.1 FUNDAMENTALS OF DESIGN CONCEPTS:

The design concepts the software designer foundation from which more sophisticated methods can be applied. A set of fundamental design concept have evolved they are as follows

3.1.1 ABSTRACTION

Abstraction is the process result of generalization by reducing the information content of a concept or an observable, typically in order to retain only information which is relevant for a particular purpose. It is an act of representing essential features without including the background details or explanations.

In this project abstraction is used so that the purpose of the separate module of the project can be drawn utilized to its maximum and can be developed without any confusion, because the purpose and the end goal of the single module is already be devised so that it always be clear to the developers.

3.1.2 REFINEMENT

It is the process of elaboration. A hierarchy is developed by decomposing a macroscopic statement off function in a stepwise fashion until programming language statements are reached. In each step, one or several instructions of given program or decomposed into more detail instructions. Abstraction and refinement are complementary concepts.

3.1.3 MODULARITY

Software architecture is divided into components called nodules. In this project there are two main models which are

Eventer Module – An Eventer can create an events with appropriate details and can view the students registration details oriented to the sport event. So, he know the participation of the particular sport event. So, the total sport event conducting process will be simplified.

Student Module – A Student can register for a profile in SEMS and can choose which sport he can participate in sports even when the Eventer is creating the sport event.

3.2 DESIGN NOTATIONS

Design notations are used when planning and should be able to communicate the purpose of a program without the need for formal code. Commonly used design notations are:

- **PSEUDOCODE**

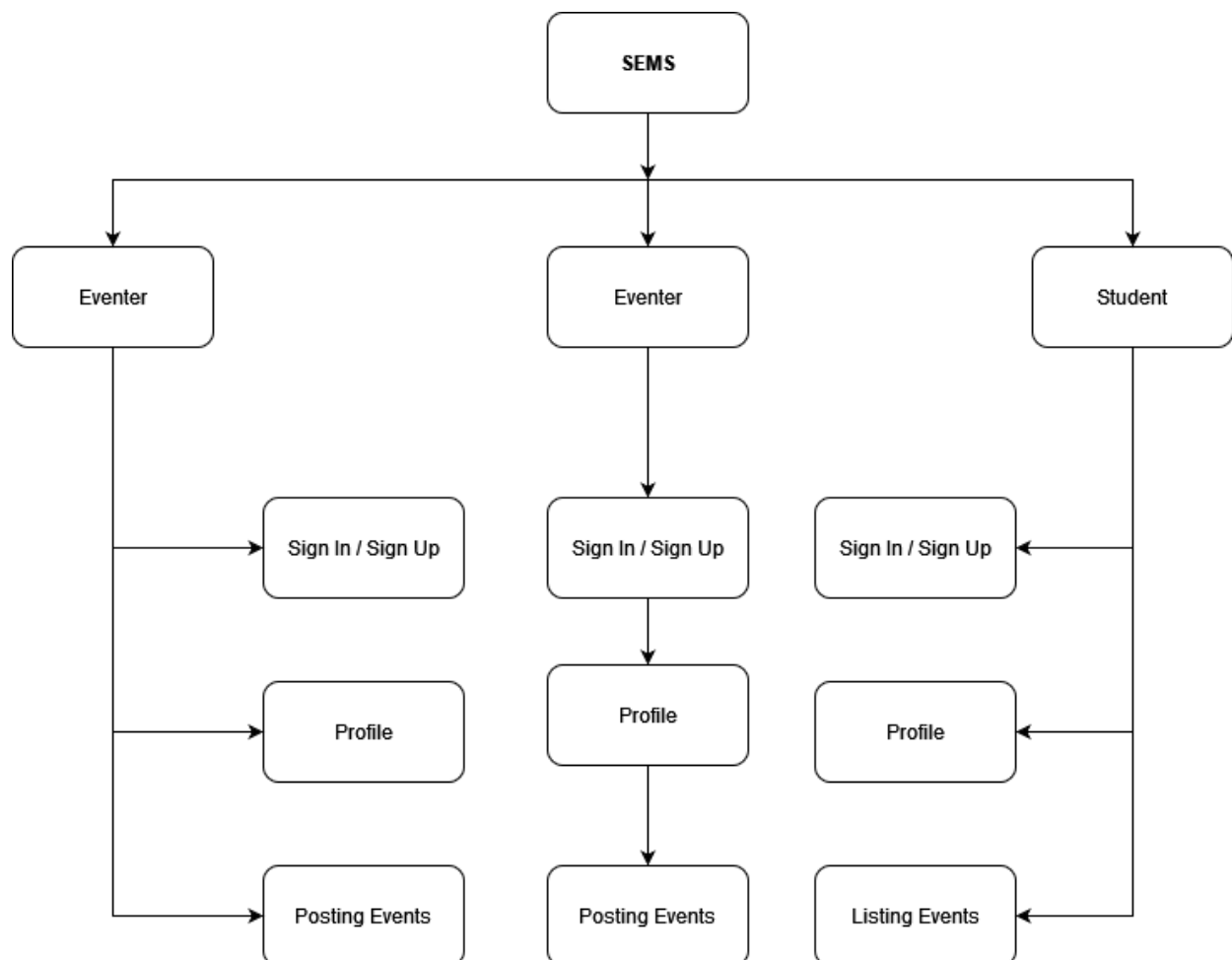
When designing a program, it is useful to lay out how the program might work, before writing it in a programming language, Pseudocode is a design notation that is closely related to the logic of how a program will work. It lets you detail what your program will do without having to worry about the particular syntax of your chosen programming language.

- **FLOW CHARTS**

Flowchart is a graphical representation of an algorithm. Programmers often use it as a program-planning tool to solve a problem. It makes use of symbols which are connected among them to indicate the flow of information and processing. The process of drawing a flowchart for an algorithm is known as "flowcharting."

3.2.1 SYSTEM STRUCTURE CHART

1. Structure Chart represent hierarchical structure of modules. It breaks down the entire system into lowest functional modules, describe functions and sub-functions of each module of a system to a greater detail.
2. Structure Chart partitions the system into black boxes (functionality of the system is known to the users but inner details are unknown).
3. Inputs are given to the black boxes and appropriate outputs are generated. Modules at top level called modules at low level.
4. Components are read from top to bottom and left to right. When a module call another. It views the called module as black box, passing required parameters and receiving results.



3.2.2 SYSTEM FLOW DIAGRAM

The system flow diagram is one of the graphical representations of the flow of data in a system in software engineering. The diagram consists of several steps that identify where the input is coming to the system and output going out of the system.

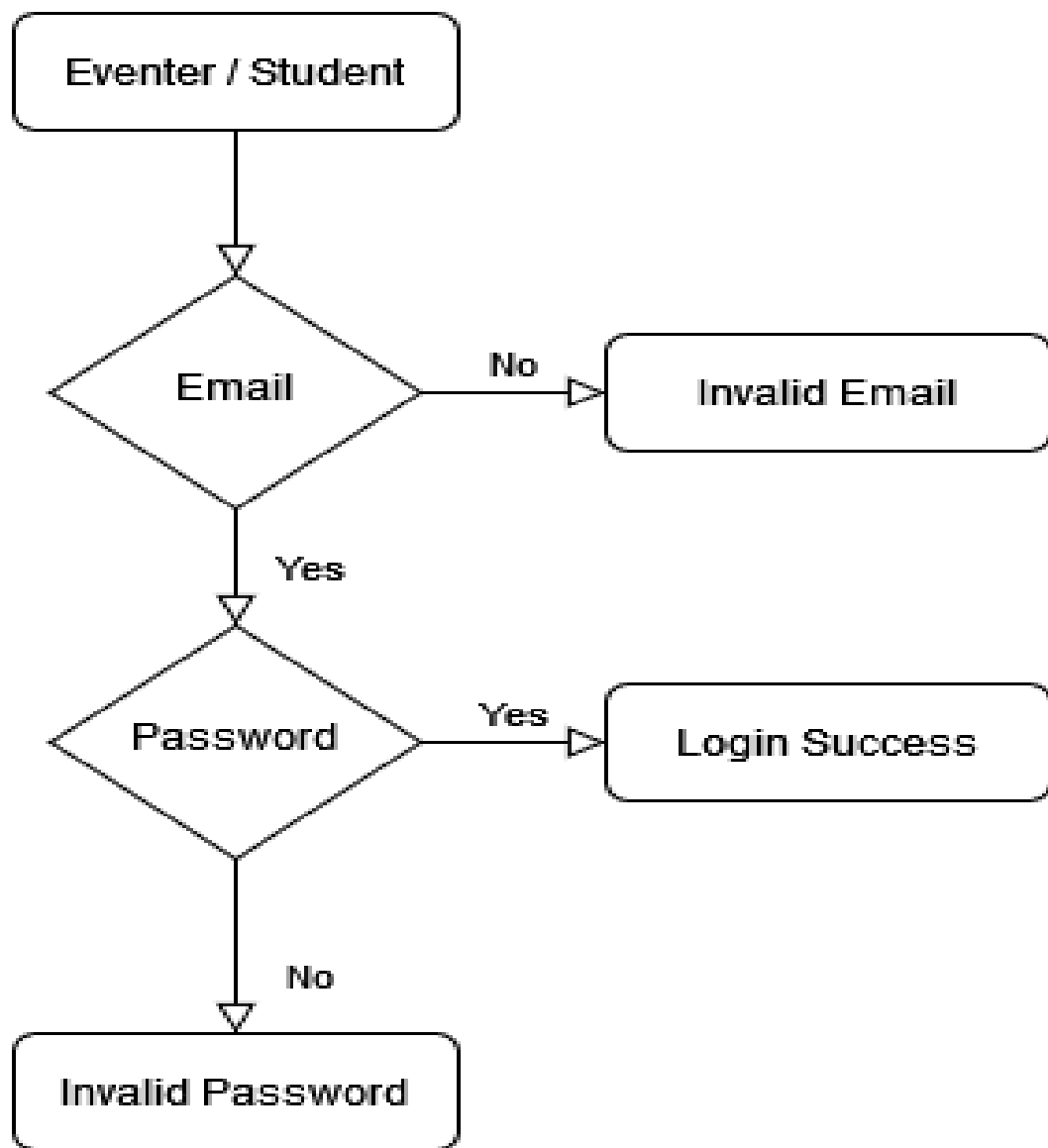
With the help of the diagram, it is possible to control the event decisions of the system and how data are flowing to the system.

Therefore, the system flow diagram is basically a visual representation of data flow, excluding the minor parts and including the major parts of the system in a sequential manner.

BENEFITS OF SYSTEM FLOW DIAGRAM

It provides a lot of benefits to the user as well for the organizations by its visual representations and the overall process in a sequential manner.

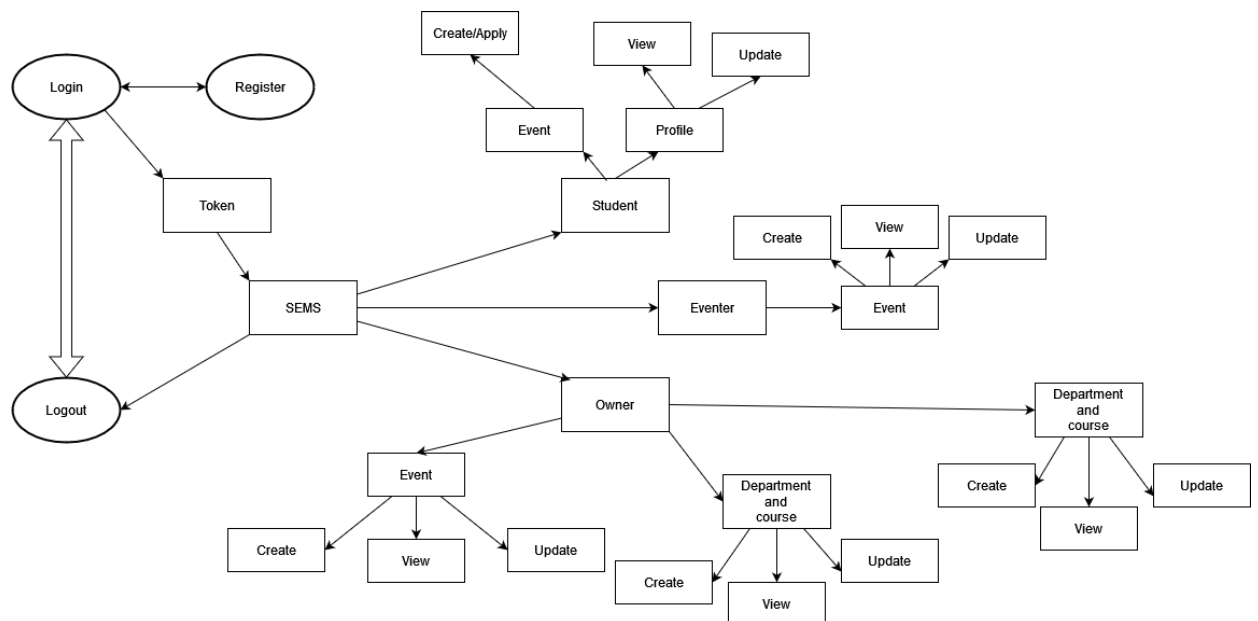
- System flow diagrams are used by various engineers and programmers to visualize the decisions and possible outcomes from the system helping the system designers to design according to it.
- System flow diagrams use symbols that are very easy to understand for the end-users in which the process is flowing and decisions are taken.
- The diagrams help in visualizing the major inputs for a system that takes into account and produces the major output using major steps that is beneficial for the organization for their requirements of the system.
- System flow diagrams are used by many popular organizations for their business purposes because it is very easy to understand by using symbols that eliminates the complexity of using complex process or codes that are mainly seen using a complex algorithm or in pseudo-codes.



3.2.3 DATAFLOW / UML DIAGRAM

A UML use case diagram is the primary form of system/software requirements for software program underdeveloped. Use cases specify the expected behavior (what), and not the exact method of making it happen (how). Use cases once specified can be denoted both textual and visual representation. A key concept of use case modelling is that it helps us design a system from the end user's perspective. It is an effective technique for communicating system behavior in the user's terms by specifying all externally visible system behavior.

A use case diagram is usually simple. It does not show the detail of the use cases: It only summarizes some of the relationships between use cases, actors, and systems, It does not show the order in which steps are performed to achieve the goals of each use case.



3.2.4 SOFTWARE ENGINEERING MODEL

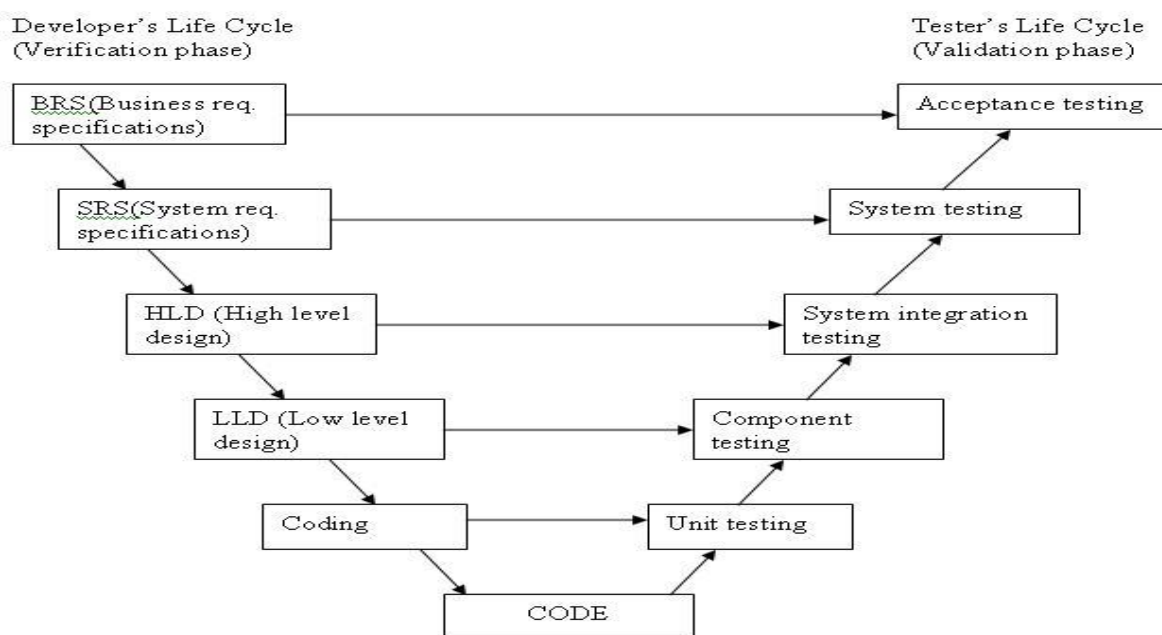
The software development models are the various processes or methodologies that are being selected for the development of the project depending on the project's aims and goals. There are many development life cycle models that have been developed in order to achieve different required objectives. The models specify the various stages of the process and the order in which they are carried out.

There are various Software development models or methodologies. They are as follows:

1. Waterfall model
2. V model
3. Incremental model
- 4 RAD model
5. Agile model
- 6 Iterative model
7. Spiral model
8. Prototype model

V-MODEL (VERIFICATION AND VALIDATION MODEL):

V-model means Verification and Validation model. Just like the waterfall model, the V-Shaped life cycle is a sequential path of execution of processes. Each phase must be completed before the next phase begins. V-Model is one of the many software development models.



THE VARIOUS PHASES OF V-MODEL ARE AS FOLLOWS:

- 1. Business Modelling:** The information flow among business functions is defined by answering questions like what data drives the business process, what data is generated, who generates it, where does the information go, who process it and so on.
- 2. Data Modelling:** The data collected from business modelling is refined into a set of data objects (entities) that are needed to support the business. The attributes (character of each entity) are identified, and the relation between these data objects (entities) is defined.
- 3. Process Modelling:** The information object defined in the data modelling phase are transformed to achieve the data flow necessary to implement a business function. Processing descriptions are created for adding, modifying, deleting, or retrieving a data object.
- 4. Application Generation:** Automated tools are used to facilitate construction of the software even they use the 4th GL techniques.
- 5. Testing & Turnover:** Many of the programming components have already been tested since V-Model emphasis reuse. This reduces the overall testing time. But the new part must be tested, and all interfaces must be fully exercised.

3.3 Design Process

3.3.1 Software Architecture

CHAPTER 4

TESTING AND IMPLEMENTATION

4.1 TESTING

Testing is the process of executing a program with the aim of finding errors. To make our software perform well it should be error-free. If testing is done successfully. It will remove all the errors from the software.

PRINCIPLES OF TESTING:

- All the test should meet the customer requirements
- To make our software testing should be performed by a third party

Exhaustive testing is not possible. As we need the optimal amount of testing based on the risk assessment of the application.

- All the test to be conducted should be planned before implementing it
- It follows the Pareto rule (80/20 rule) which states that 80% of errors come from 20% of program components.
- Start testing with small parts and extend it to large parts.

4.1.1 SYSTEM TESTING

This software is tested such that it works fine for the different operating systems. It is covered under the black box testing technique. In this, we just focus on the required input and output without focusing on internal working.

4.1.2 WHITE BOX TESTING

White Box Testing is software testing technique in which internal structure, design and coding of software is tested to verify flow of input-output and to improve design, usability and security. In white box testing, code is visible to testers so it is also called Clear box testing, open box testing, transparent box testing, Code-based testing and Glass box testing.

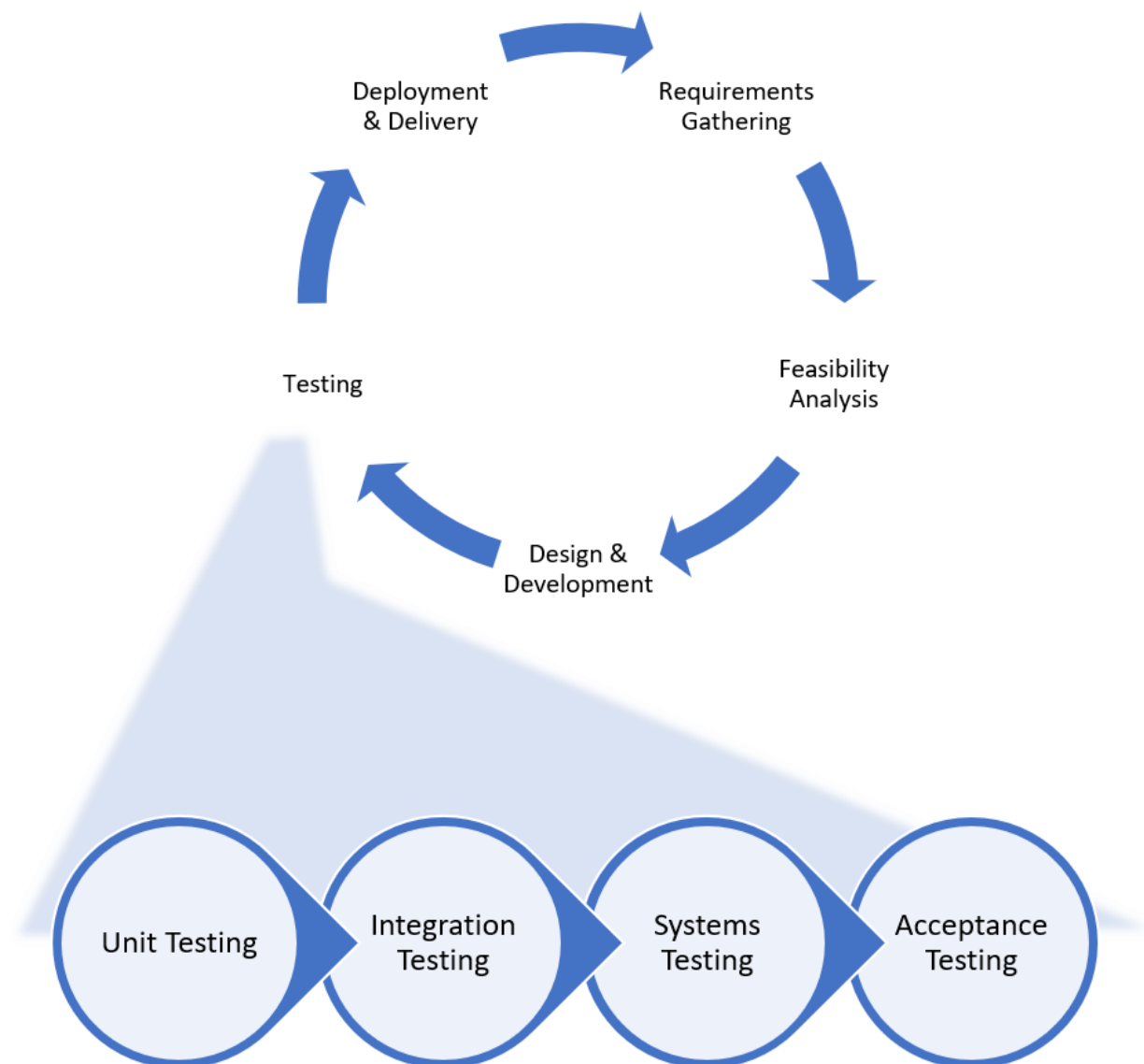
It is one of two parts of the Box Testing approach to software testing. Its counterpart, Black box testing, involves testing from an external or end-user type perspective. On the other hand, White box testing in software engineering is based on the inner workings of an application and revolves around internal testing

4.1.3 UNIT TESTING

Unit testing, a testing technique using which individual modules are tested to determine if there are any issue by the developer himself. It is concerned with functional correctness of the standalone modules.

The main aim to isolate each unit of the system to identify, analyze and fix the defects.

- Reduces Defects in the newly developed features or reduces bugs when changing the existing functionality.
- Reduces Cost of testing as defects are captured in very early phase.
- Improves design and allows better refactoring of code.
- Unit Tests, when integrated with build gives the quality of the build as well.



4.2 SYSTEM I IMPLEMENTATION

Implementation is a process of ensuring that the information system is operational. It involves:

- Constructing a new system from scratch
- Constructing a new system from the existing one.

Implementation allows the users to take over its operation for use and evaluation. It involves training the users to handle the system and plan for a smooth conversion.

TRAINING

The personnel in the system must know in detail what their roles will be, how they can use the system, and what the system will or will not do. The success or failure of well-designed and technically elegant systems can depend on the way they are operated and used.

4.2.1 SYSTEM MAINTENANCE

Maintenance means restoring something to its original conditions. Enhancement means adding, modifying the code to support the changes in the user specification. System maintenance conforms the system to its original requirements and enhancement adds to system capability by incorporating new requirements.

Thus, maintenance changes the existing system, enhancement adds features to the existing system, and development replaces the existing system. It is an important part of system development that includes the activities which corrects errors in system design and implementation, updates the documents, and tests the data.

Maintenance Types:

System Maintenance Can Be Classified into Three Types:

- Corrective Maintenance - Enables user to carry out the repairing.
- Adaptive Maintenance- Enables user to replace the functions of the programs.
- Perfective Maintenance - Enables user to modify or enhance the programs according to the users' requirements and changing needs.

CHAPTER 5

CONCLUSION

5.1 DIRECTION FOR FUTURE ENHANCEMENTS REFERENCE

This web Application provides facility to alert for students in sports event at college. It saves time as it allows number of users to be registered at a time and store their information. It is automatically generated by the server. Administrator has a privilege to create, modify and delete.

The SEMS was developed using React JS and Node JS with Mongo DB fully meets the objectivities of the system for which it has been developed.

The system is operated at a high level of efficiency and the administrators associated with the system understand its advantage.

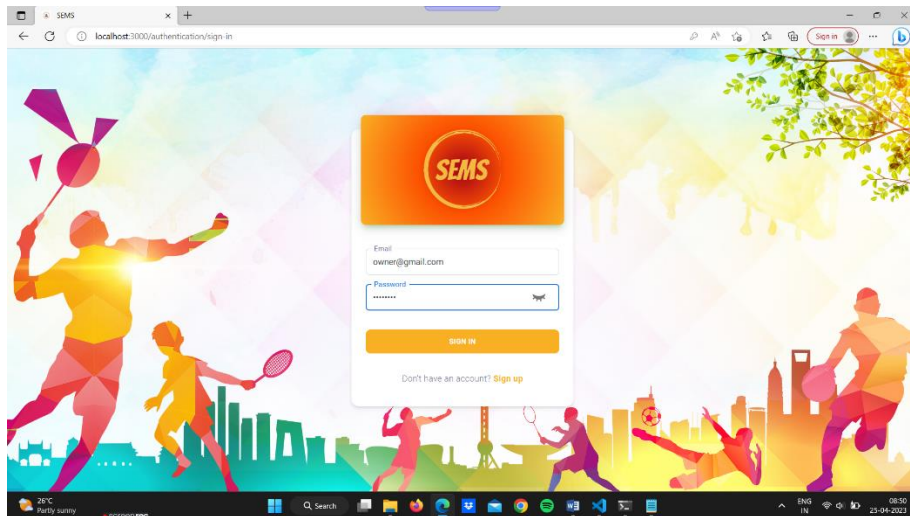
BIBLIOGRAPHY

- React (Online)
- Node (Online)
- Mongo DB (Online)
- Software Engineering(Book)

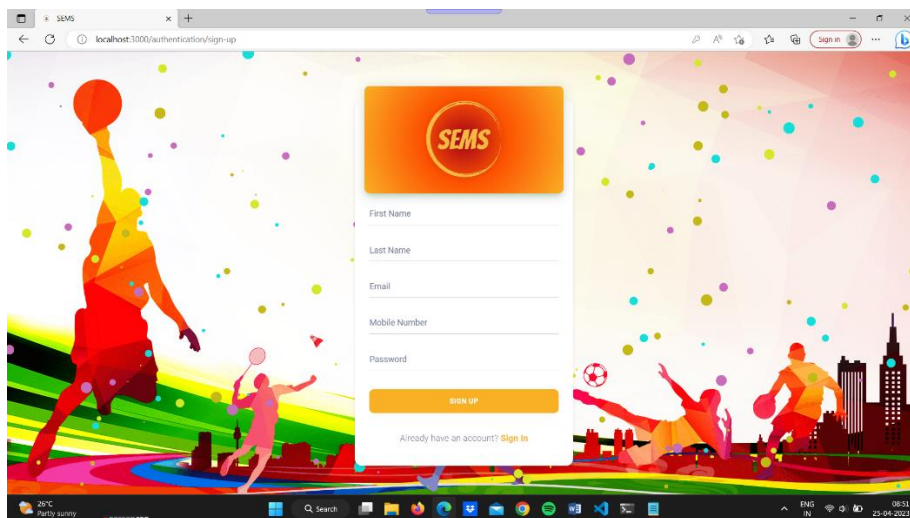
ANNEXURE

Annexure A - Output Design:

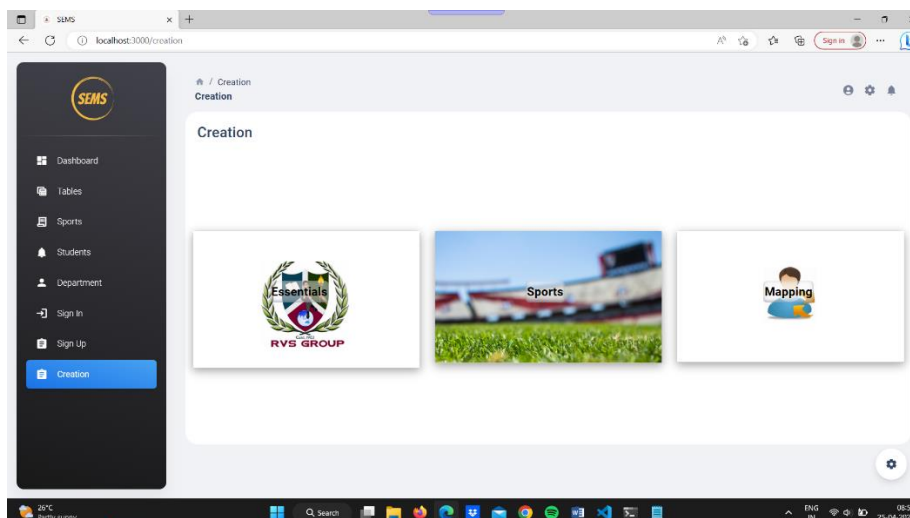
Login:



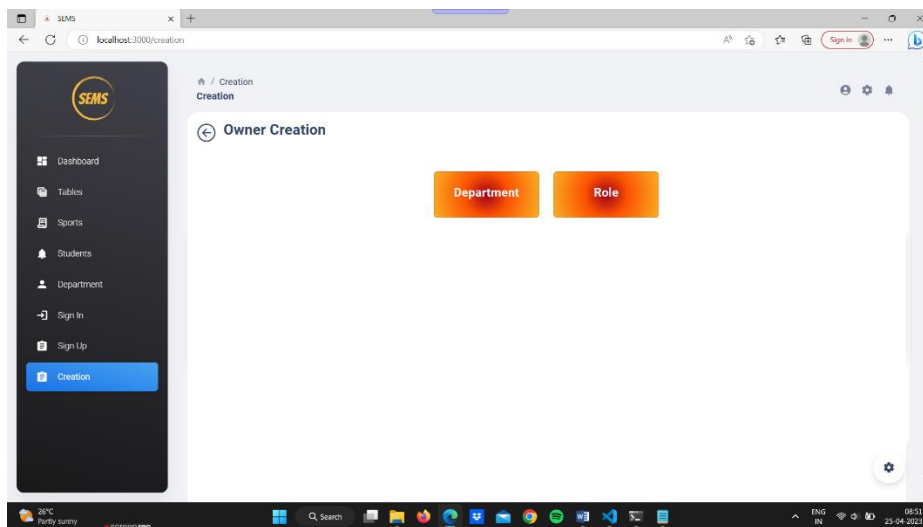
Register:



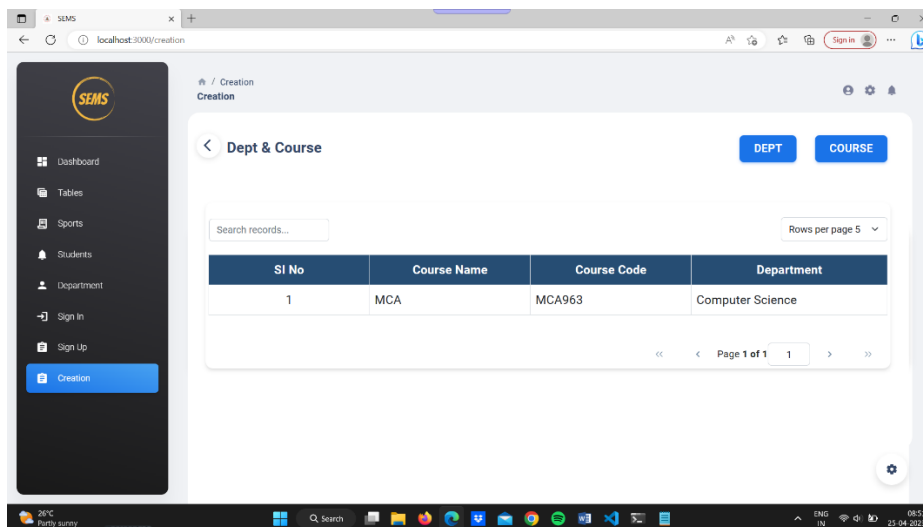
Creation:



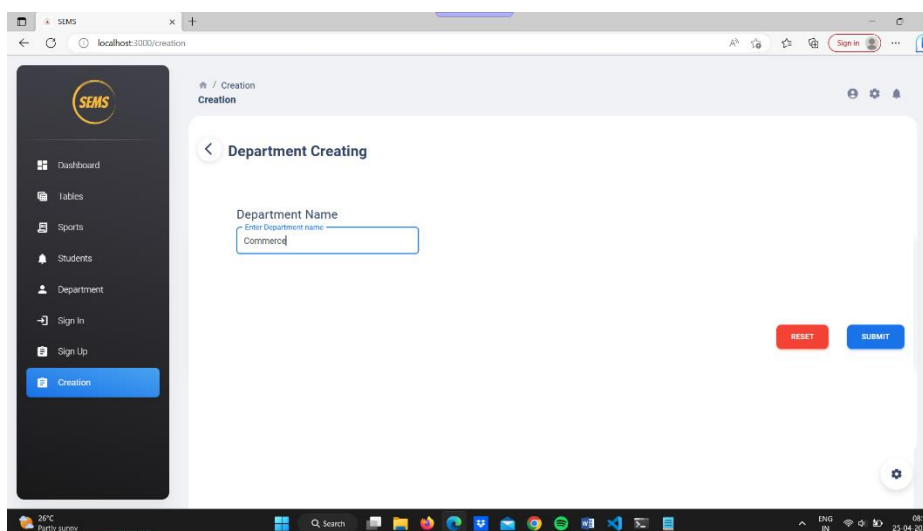
Owner Essential:



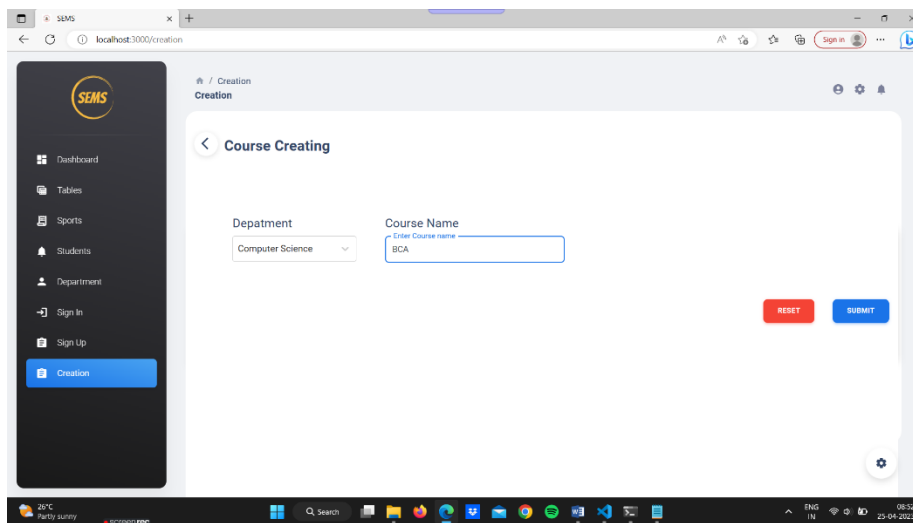
Department and Course:



Department Creation:

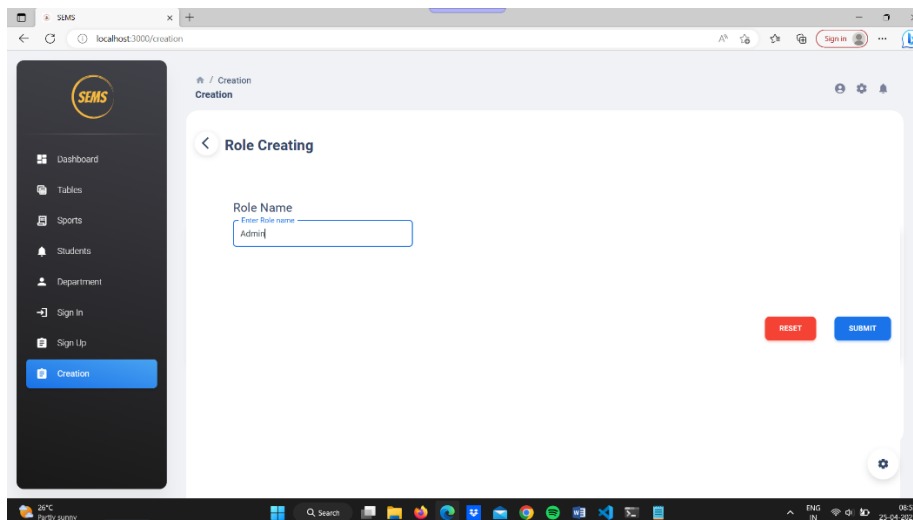


Course Creation:



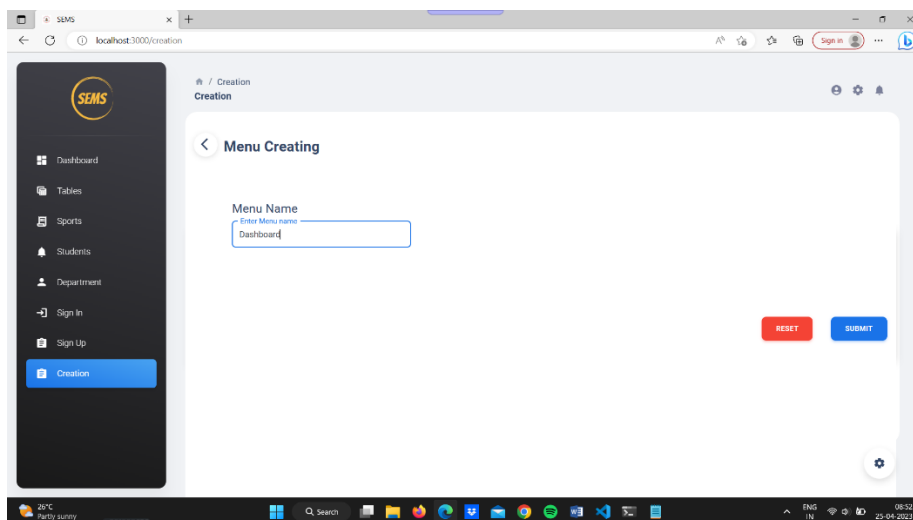
The screenshot shows the 'Course Creating' form in the SEMS application. The left sidebar contains a navigation menu with options: Dashboard, Tables, Sports, Students, Department, Sign In, Sign Up, and Creation (highlighted). The main content area has a header 'Creation' and a sub-header 'Course Creating'. Below this, there are two input fields: 'Department' with a dropdown menu showing 'Computer Science', and 'Course Name' with a text input field containing 'BCA'. At the bottom right of the form are two buttons: 'RESET' (red) and 'SUBMIT' (blue). The browser's address bar shows 'localhost:3000/creation' and the system tray at the bottom indicates the date and time as 08:52 on 23-04-2023.

Role Creation:



The screenshot shows the 'Role Creating' form in the SEMS application. The left sidebar is identical to the previous screenshot, with 'Creation' highlighted. The main content area has a header 'Creation' and a sub-header 'Role Creating'. Below this, there is a single text input field for 'Role Name' containing the text 'Admin'. At the bottom right of the form are two buttons: 'RESET' (red) and 'SUBMIT' (blue). The browser's address bar shows 'localhost:3000/creation' and the system tray at the bottom indicates the date and time as 08:52 on 23-04-2023.

Menu Creation:



The screenshot shows the 'Menu Creating' form in the SEMS application. The left sidebar is identical to the previous screenshots, with 'Creation' highlighted. The main content area has a header 'Creation' and a sub-header 'Menu Creating'. Below this, there is a single text input field for 'Menu Name' containing the text 'Dashboard'. At the bottom right of the form are two buttons: 'RESET' (red) and 'SUBMIT' (blue). The browser's address bar shows 'localhost:3000/creation' and the system tray at the bottom indicates the date and time as 08:52 on 23-04-2023.

Mapping:

The screenshot shows the 'Role & Menu' mapping interface in the SEMS application. The left sidebar contains navigation links: Dashboard, Tables, Sports, Students, Department, Sign In, Sign Up, and Creation (highlighted). The main content area is titled 'Role & Menu' and includes a 'MAPPING' button. Below the title is a search bar and a table with the following data:

SI No	Role Name	Role Code	Status
1	Student	STU654	

At the bottom of the table, there is a pagination control showing 'Page 1 of 1'.

Role Menu Mapping:

The screenshot shows the 'Role Menu Mapping' interface. It features a dropdown menu for 'Select the Role' with 'Student' selected, and another dropdown for 'Select the Menu' with 'Creation' selected. Below these are four toggle switches for 'Get', 'Create', 'Update', and 'Delete'. The 'Get' toggle is turned on, while the others are turned off. At the bottom right, there are 'RESET' and 'SUMMIT' buttons.

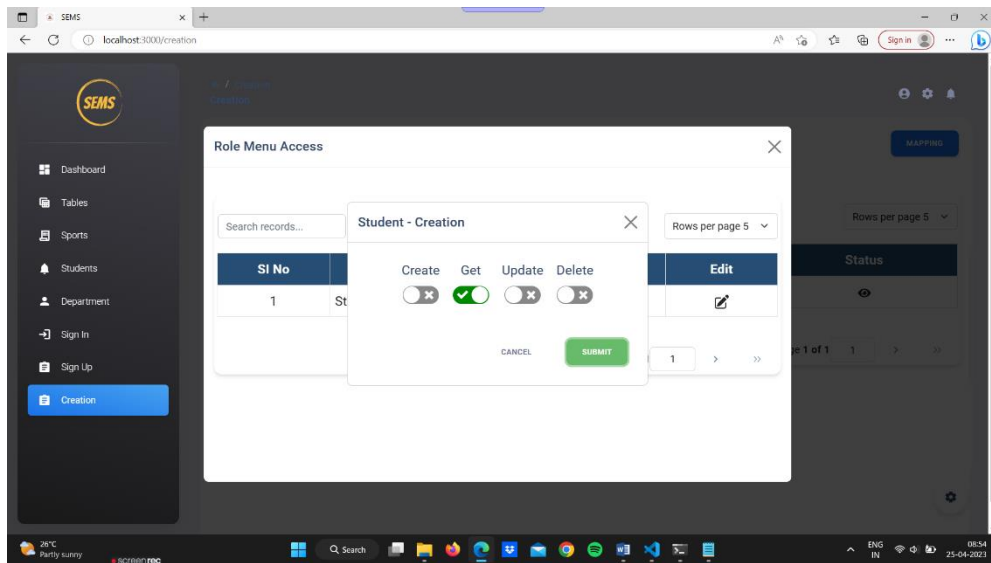
Access view:

The screenshot shows the 'Role Menu Access' view. It displays a table with the following data:

SI No	Role Name	Menu Name	Access	Edit
1	Student	Creation	Get	

The table has a search bar and a 'Rows per page 5' dropdown. The pagination control at the bottom shows 'Page 1 of 1'.

Access Edit:



Annexure B – Source Code:

Frontend:

App.js

```
import { useState, useEffect, useMemo, Suspense } from "react";

// react-router components
import { Routes, Route, Navigate, useLocation } from "react-router-dom";

// @mui material components
import { ThemeProvider } from "@mui/material/styles";
import CssBaseline from "@mui/material/CssBaseline";
import Icon from "@mui/material/Icon";
import SignIn from "layouts/authentication/sign-in";
import Dashboard from "layouts/dashboard";
// Material Dashboard 2 React components
import MDBox from "components/MDBox";

// Material Dashboard 2 React example components
import Sidenav from "examples/Sidenav";
import Configurator from "examples/Configurator";

// Material Dashboard 2 React themes
import theme from "assets/theme";
import themeRTL from "assets/theme/theme-rtl";

// Material Dashboard 2 React Dark Mode themes
import themeDark from "assets/theme-dark";
import themeDarkRTL from "assets/theme-dark/theme-rtl";

// RTL plugins
import rtlPlugin from "stylis-plugin-rtl";
```

```

import { CacheProvider } from "@emotion/react";
import createCache from "@emotion/cache";
import Page404 from "layouts/authentication/page404/Page404.js";
import Page500 from "layouts/authentication/page500/Page500.js";
// Material Dashboard 2 React routes
import routes from "routes";
// Material Dashboard 2 React contexts
import { useMaterialUIController, setMiniSidenav, setOpenConfigurator } from "context";

// Images
import brandWhite from "assets/images/logos/SEMS.png";
import brandDark from "assets/images/logos/SEMS.png";

export default function App() {
  const [controller, dispatch] = useMaterialUIController();

  let getToken=localStorage.getItem('sems-token')
  let token;
  if(getToken){
    token=JSON.parse(getToken)
  }

  const {
    miniSidenav,
    direction,
    layout,
    openConfigurator,
    sidenavColor,
    transparentSidenav,
    whiteSidenav,
    darkMode,
  } = controller;
  const [onMouseEnter, setOnMouseEnter] = useState(false);
  const [rtlCache, setRtlCache] = useState(null);

```

```

const { pathname } = useLocation();

// Cache for the rtl
useMemo(() => {
  const cacheRtl = createCache({
    key: "rtl",
    stylisPlugins: [rtlPlugin],
  });

  setRtlCache(cacheRtl);
}, []);

// Open sidenav when mouse enter on mini sidenav
const handleOnMouseEnter = () => {
  if (miniSidenav && !onMouseEnter) {
    setMiniSidenav(dispatch, false);
    setOnMouseEnter(true);
  }
};

// Close sidenav when mouse leave mini sidenav
const handleOnMouseLeave = () => {
  if (onMouseEnter) {
    setMiniSidenav(dispatch, true);
    setOnMouseEnter(false);
  }
};

const loading = (
  <div className="pt-3 text-center">
    <div className="sk-spinner sk-spinner-pulse"></div>
  </div>
)

```

```

// Change the openConfigurator state
const handleConfiguratorOpen = () => setOpenConfigurator(dispatch, !openConfigurator);

// Setting the dir attribute for the body element
useEffect(() => {
  document.body.setAttribute("dir", direction);
}, [direction]);

// Setting page scroll to 0 when changing the route
useEffect(() => {
  document.documentElement.scrollTop = 0;
  document.scrollingElement.scrollTop = 0;
}, [pathname]);

const getRoutes = (allRoutes) =>
allRoutes.map((route) => {
  if (route.collapse) {
    return getRoutes(route.collapse);
  }

  if (route.route) {
    return <Route exact path={route.route} element={route.component} key={route.key} />;
  }

  return null;
});

const configsButton = (
  <MDBBox
    display="flex"
    justifyContent="center"
    alignItems="center"
    width="3.25rem"

```

```

height="3.25rem"
bgColor="white"
shadow="sm"
borderRadius="50%"
position="fixed"
right="2rem"
bottom="2rem"
zIndex={99}
color="dark"
sx={{ cursor: "pointer" }}
onClick={handleConfiguratorOpen}
>
  <Icon fontSize="small" color="inherit">
    settings
  </Icon>
</MDBBox>
);

return direction === "rtl" ? (
  <CacheProvider value={rtlCache}>
    <ThemeProvider theme={darkMode ? themeDarkRTL : themeRTL}>
      <CssBaseline />
      {layout === "dashboard" && (
        <>
          <Sidenav
            color={sidenavColor}
            brand={(transparentSidenav && !darkMode) || whiteSidenav ? brandDark : brandWhite}
            brandName="SEMS"
            routes={routes}
            onMouseEnter={handleOnMouseEnter}
            onMouseLeave={handleOnMouseLeave}
          />
          <Configurator />
          {configsButton}

```

```

    </>
  ))
  {layout === "vr" && <Configurator />}
  <Routes>
    {getRoutes(routes)}
    <Route path="*" element={<Navigate to="/dashboard" />} />
  </Routes>
</ThemeProvider>
</CacheProvider>
):(
  <ThemeProvider theme={darkMode ? themeDark : theme}>
    <CssBaseline />
    {layout === "dashboard" && (
      <
        <Sidenav
          color={sidenavColor}
          brand={(transparentSidenav && !darkMode) || whiteSidenav ? brandDark : brandWhite}
          brandName="SEMS"
          routes={routes}
          onMouseEnter={handleOnMouseEnter}
          onMouseLeave={handleOnMouseLeave}
        />
        <Configurator />
        {configsButton}
      </>
    )}
    {layout === "vr" && <Configurator />}

    <Routes>
      {getRoutes(routes)}
      <Route path="*" element={<Navigate to="/authentication/sign-in" />} />
    </Routes>
  </ThemeProvider>
)}

```


Login.js

```
import { useState } from 'react';

// react-router-dom components
import { Link, useNavigate } from 'react-router-dom';
import { toast, Toaster } from 'react-hot-toast';
import Image from 'mui-image'
// @mui material components
import SignUpSEMS from "assets/images/SignUpSEMS.png"
import IconButton from '@mui/material/IconButton';
import { InputAdornment } from '@mui/material';
import { RiEyeCloseFill } from 'react-icons/ri';
import { IoEye } from 'react-icons/io5';
// import VisibilityIcon from "@mui/icons-material/Visibility";
// import VisibilityOffIcon from "@mui/icons-material/VisibilityOff";

import Card from '@mui/material/Card';
import Switch from '@mui/material/Switch';
import Grid from '@mui/material/Grid';
import MuiLink from '@mui/material/Link';
import SignInBG from 'assets/images/SignInBG.jpg';

// @mui icons
import FacebookIcon from '@mui/icons-material/Facebook';
import GitHubIcon from '@mui/icons-material/GitHub';
import GoogleIcon from '@mui/icons-material/Google';

// Material Dashboard 2 React components
import MDBBox from 'components/MDBBox';
import MDTypography from 'components/MDTypography';
import MDInput from 'components/MDInput';
import MDButton from 'components/MDButton';
```

```

// Authentication layout components
import BasicLayout from 'layouts/authentication/components/BasicLayout';

import {
  userLogin,
  // getRoleMenuAccessById,
} from 'utility/apiService';

function SignIn() {
  const navigate = useNavigate();

  const [rememberMe, setRememberMe] = useState(false);
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");

  const [emailError, setEmailError] = useState("");
  const [passwordError, setPasswordError] = useState("");

  const [showPassword, setShowPassword] = useState(false);
  const handleClickShowPassword = () => setShowPassword(!showPassword);
  const handleMouseDownPassword = () => setShowPassword(!showPassword);
  const handleSetRememberMe = () => setRememberMe(!rememberMe);

  const handleSubmit = async (e) => {
    if (email === "") {
      setEmailError('Email is required');
    } else {
      setEmailError("");
    }
    if (password === "") {
      setPasswordError('Password is required');
    } else {
      setPasswordError("");
    }
  }
}

```

```

    }
    if (email !== "" && password !== "") {
        const body = {
            email: email,
            password: password,
        };
        try {
            const response = await userLogin(body);
            console.log(response);
            if (response.ok) {
                toast.success(response.data.message);
                localStorage.setItem('sems-token', JSON.stringify(response.data.token));
                return navigate('/dashboard');
            } else {
                toast.error(response.data.message);
            }
        } catch (error) {
            console.log(error);
        }
    }
};

return (
    <BasicLayout image={SignInBG}>
        <Card>
            <MDBBox
                variant="gradient"
                borderRadius="lg"
                coloredShadow="success"
                mx={2}
                mt={-3}
                p={3}
                mb={1}
                textAlign="center"

```

```

    style={{
      background:'radial-gradient(circle, rgb(163, 0, 28) 0%, rgb(253, 83, 2) 38%, rgb(250, 176,
37) 100%)'
    }}
  >
  <Image src={SignUpSEMS} style={{
    width:'100%',
    height:'100%',
  }}/>
</MDBBox>
  <MDBBox pt={4} pb={3} px={3}>
    <MDBBox component='form' role='form'>
      <MDBBox mb={2}>
        <MDInput
          type='email'
          label='Email'
          fullWidth
          value={email}
          style={{ width:'100%' }}
          onChange={(e) => setEmail(e.target.value)}/>
        {emailError ? (
          <p
            style={{
              paddingLeft: '14px',
              color: 'red',
              fontSize: '14px',
            }}
            color='red'>
              {emailError}
            </p>
          ) : null}
      </MDBBox>
      <MDBBox mb={2}>
        <MDInput

```

```

type={showPassword ? 'text' : 'password'}
label='Password'
fullwidth
style={{ width:'100%' }}
value={password}
onChange={(e) => setPassword(e.target.value)}
InputProps={ {
  // <-- This is where the toggle button is added.
  endAdornment: (
    <InputAdornment position='end'>
      <IconButton
        aria-label='toggle password visibility'
        onClick={handleClickShowPassword}
        onMouseDown={handleMouseDownPassword}>
        {showPassword ? <IoEye /> : <RiEyeCloseFill />}
      </IconButton>
    </InputAdornment>
  ),
}></MDInput>
{passwordError ? (
  <p
    style={{
      paddingLeft: '14px',
      color: 'red',
      fontSize: '14px',
    }}
    color='red'>
    {passwordError}
  </p>
) : null}
</MDBBox>

```

```

<MDBBox mt={4} mb={1}>

```

```

    <MDBButton variant="gradient"
    style={{ backgroundColor:'#fab025',color:"white" }}
    fullWidth
    onClick={handleSubmit}>
      Sign In
    </MDBButton>
  </MDBBox>
  <MDBBox mt={3} mb={1} textAlign='center'>
    <MDTypography variant='button' color='text'>
      Don't have an account?{' '}
      <MDTypography
        component={Link}
        to='/authentication/sign-up'
        variant='button'
        color='lightorange'
        fontWeight='medium'
        textGradient>
        Sign up
      </MDTypography>
    </MDTypography>
  </MDBBox>
</MDBBox>
</MDBBox>
</Card>
<Toaster />
</BasicLayout>
);
}

```

```
export default SignIn;
```