SRINIVAS UNIVERSITY

Project Report on

Submitted to Srinivas University on partial completion of Sixth Semester

Bachelor of Computer Application degree

By

CHIRAG (3SU19SA011)

VI Semester BCA

Under the guidance of

Prof. P. Sridhara Acharya

Faculty of

Deparment of Computer and Information Sciences

Srinivas Institute of Management Studies

Pandeshwar, Managluru

2019-2022

DECLARATION

We hereby declare that the project work detail which is being presented in this report is in fulfillment of the requirement for the award of degree of Bachelor of Computer Application.

We hereby declare that we have undertaken our project work on **"Srinivas Exam Manager Software"** under the guidance of **Prof. P. Sridhara Acharya**, Professor and Head, Department of Computer Science and Information Science, Srinivas University, Mangalore.

We hereby declare that this project work report is our own work and the best of our knowledge and belief the matter embedded in report has not been submitted by us for the award of other degree to this or any other university.

Place: Mangalore **Mr. CHIRAG**

Date: **(3SU19SA011)**

ACKNOWLEDGEMENT

Any achievement big or small should have a catalyst and constant encouragement and advice of valuable and noble minds. The satisfy action and euphoria that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crowned our efforts with success.

We would like to express our sincere and grateful thanks to our **Prof. P. Sridhara Acharya,** Professor and Head, Department of Computer Science and Information Science, Srinivas University, Mangalore for the valuable guidance, encouragement, technical comments throughout our project work.

It is great pleasure to express our gratitude and indebtedness to our beloved principal **Dr. P.S. Aithal** for his continues effort in creating a competitive environment in our college and encouraging throughout this course.

We also wish to thank all the staff members, non-teaching staff members of the Department of Computer Science and Information Science who have helped us directly or indirectly in the completion of our final project successfully.

Finally, we are thankful to our parents, friends and loved ones, who are always our source of inspiration and for their continued moral and material support throughout the course and in helping us to finalize the project.

Mr. CHIRAG

ABSTRACT

Srinivas Exam Manager software is an application can provide the automation of exam management system. Where student can get their marks , halticket and marks card from this application.

This application is feasible to university and schools where the office works are done manually which take lots of time and effort. Here not only student work the office works will be also get reduced. Here after the payment of the exam fees students can get their hall ticket immediately.

CONTENTS

4. DETAILED DESIGN

CHAPTER – IV

5. DATABASE DESIGN

CHAPTER – V

11. BIBLIOGRAPHY

CHAPTER – 11

11.1 Books and Website reference

12. PLAGIARISM REPORT

CHAPTER – 12

12.1 Plagiarism

CHAPTER - 1

1. SYNOPSIS

1.1. Title of the project :

Srinivas Exam Management Software

1.2. Introduction :

Srinivas Exam Management Software is a client server-based software that starts with the registration of student information till the examination result generation.

The "Examination Management Software" has been developed to override the problems prevailing in the practicing manual system. This software is supported to eliminate and, in some cases, reduce the hardships faced by this existing system. Moreover, this system is designed for the particular need of the college to carry out operations smoothly and effectively.

1.3. Objective of the project :

- The main objective of the Exam management software is to manage examinations.
- Students can apply for the exams
- Teachers can update and give marks to students.
- Students can review their marks.

1.4. Innovative idea behind the project:

As of now, the university is using software built by Horizon, and it has high complexity and cost; we came up with the idea of building software that is more reliable to use and makes changes quickly.

1.5. Stakeholders:

- Office Staffs
- Evaluator
- Faculties
- Students

1.6. Project Category :

Web Application

1.7. Languages to be used :

Front-end : HTML, CSS, javascript, React JS

Back-end : MySQL, NodeJS

1.8. Hardware Interface :

- Processors: Intel Pentium dual-core or above
- RAM: 2 GB and above
- Hard disk Utilization: 40 GB and above
- Input Devices: Mouse, Keyboard

1.9. Description :

- Admin is in charge of registering the faculty, students, staff, and evaluator.
- The evaluator can upload marks.
- The Student has to enter their registration number to know the details of their marks.
- The Teacher should log in to the software and can manage the Student's details.

1.10 Module description :

Login & Registration

Sign in to use the application using a username and password.

Search Student details

Students can search for the student details using their Registration number.

Managing Student details

Users can manage the marks and details of the student.

1.11 Software Interface :

- Browser: Internet Explorer, Google Chrome, Mozilla Firefox
- Server: Apache
- IDE: Visual Studio Code

1.12 Limitations :

- This software is purely based on the examination part only.
- Less Security.

1.13 Future Scope :

- It can be integrated with mobile apps.
- Additional enhanced features to be added over time.

1.14 Team members :

- Rajath
- V Jeevan Kumar
- Chirag
- Siddharth KM

Chapter -2

2. SOFTWARE REQUIREMENT SPECIFICATION

2.1 INTRODUCTION

Srinivas Examination Manager is an examination software that automates the process of registration of each users till the marks card generation. Here admin and super admin takes care of every holder of this software, and office staff, faculty, exam coordinator, students are the users. Office staff can maintain their payment activities, and students can download their hall tickets from this software based on certain conditions they have to fulfill. After the examination, the exam coordinator can upload the marks, and the result can be generated quickly.

2.2 PURPOSE

The project mainly focuses on exam management, where we can perform all activities starting from registration of a student to marks card generation. This project is beneficial to any institution, such as a school or college. This project eases the work of manual work and maintains the student's record. Here, the generation of the hall ticket will be automatic, which will look through the specified condition that each student must overcome so that only those students are eligible to write the exam.

2.3 SCOPE

The project has an enormous scope in the upcoming days when the institutions can make their work automatic rather than manually. This will also save a lot of time. This project mainly contains registration, login, hall ticket and saves a lot of time on modules.

2.4 Definition, Acronyms, and Abbreviations

This particular software is defined as automated software for the evaluation and examination department of Srinivas University, and hereafter it is written in our document as:-

- Srinivas University -SU
- No Object Certificate-NOC
- Examination Management Software – EMS
- DFD - Data Flow Diagram
- Marks Card Generation-MCG
- RAM – Random Access Memory
- CSS – Cascading Style Sheet
- SEM -Srinivas Exam Manager

2.5 Book References:

1. Feller, J., & Fitzgerald, B. (2002). *Understanding open source software development*. Addison-Wesley Longman Publishing Co., Inc…
2. Boehm, B. W. (1984). Software engineering economics. *IEEE transactions on Software Engineering*, (1), 4-21.
3. McHugh, J., Abiteboul, S., Goldman, R., Quass, D., & Widom, J. (1997). Lore: A database management system for semistructured data. *ACM Sigmod Record*, *26*(3), 54-66.
4. Wiegers, K., & Beatty, J. (2013). *Software requirements*. Pearson Education.
5. Aggarwal, S. (2018). Modern web development using ReactJS. *International Journal of Recent Research Aspects*, *5*(1), 133-137.
6. Laksono, D. (2018, August). Testing spatial data deliverance in SQL and NoSQL database using NodeJS full stack web app. In *2018 4th International Conference on Science and Technology (ICST)* (pp. 1-5). IEEE.
7. Stonebraker, M. (2010). SQL databases v. NoSQL databases. *Communications of the ACM*, *53*(4), 10-11.
8. Schifreen, R. (2009). How to Create Websites and Applications with HTML, CSS, JavaScript, PHP, and MySQL.
9. Hartog, P., & Rhodes, E. C. (1936). Examination of examinations.
10. Fairburn, C. G., Cooper, Z., & O'Connor, M. (1993). The eating disorder examination. *International Journal of Eating Disorders*, *6*, 1-8.

2.6 Web References:

https://www.w3schools.com/js/js_loop_for.asp as on 24th March 2022

https://en.wikipedia.org/wiki/React_(JavaScript_library)#Components as on 25th March 2022

https://www.technipages.com/how-to-identify-specific-color-on-an-app-or-website as on 15 April 2022

https://dribbble.com/tags/webdesign as on 23th May 2022

https://wdexplorer.com/20-examples-beautiful-css-typography-design/ as on 4th June 2022

2.7 Overview

This SRS is organized into two parts, the first is the overall description, and the second is the specific requirements. The general description will describe the requirements of the exam management system. The particular requirement section describes the details of the system.

2.8 OVERALL DESCRIPTION:

This examination software mainly focuses on the automation of the examination process, which starts from the registration of students to the marks card generation. Here admin registers every detail of faculty, students, exam coordinator, and office staff. Then, the faculty can upload marks, attendance, and NOC so that only eligible students will only have to apply for the exams. After checking the payment details by office staff, the hall ticket will be automatically generated. After the exams, the evaluator enters the university exam marks, and after that, the result will be published.

2.9 Product Perspective:

The main objective of this EMS is the automation of examination processes. It is a web-based application with five interfaces: admin, faculty and student, exam coordinator, and office staff. All the information is stored in the database, which can be retrieved. The website works with desktops, laptops, and mobile with the browsers' help.

2.9.1 Product Function

- Registering student, faculty and office staff
- Ability to modify and upload data
- Record of fee payment
- We can get the hall ticket automatically after the payment is complete
- The evaluator can add marks
- Marks of the entire semester can be published and downloaded
- Details of each student's profile
- Allow admin to take control of the entire application.

2.10 General Constraints:

The developed system should run under any platform (Unix, Linux, Windows, etc.) that contains a web.

The user needs a hosting space so that we can access it from any remote location.

2.10.1 Browser That Support

- Google Chrome

- Microsoft Edge
- Mozilla Firefox

2.10.2 Assumption and dependencies

- The system should run 24/7
- The code should be error-free
- Roles and tasks are predefined
- Using Apache server
- The client needs to pay for a domain space

2.11 Specific Requirements

This section contains all the functionality and quality of the system

2.11.1 User Interface

- Admin
- Student
- Faculty
- Office Staff
- Exam coordinator

There are five users in this system

- Admin - The entire details of the course details and students are managed by the admin
- Student – Here, the student can view the marks, attendance, and other personal details
- Faculty – Faculty can add marks and attendance and can upload every detail of the students
- Exam coordinator - Here, the exam coordinator can view indent, attendance statement and approve to display the marks of semester exam
- Office Staff - The financial statements, attendance statement and indent are managed by these staff

2.11.2 Hardware Interface

- Processors: Intel Pentium dual-core or above
- RAM: 2 GB and above
- Hard disk Utilization: 40 GB and above
- Input Devices: Mouse, Keyboard
- Output Devices: Monitor, Printer

2.11.3 Software Interface

- Browser: Internet Explorer, Google Chrome, Mozilla Firefox
- Server: NodeJS
- IDE: Visual Studio Code
- Web components:HTML, CSS
- Language: React
- Scripting language: JavaScript
- Database: MySQL

2.11.4 Communication interface

This is a web-based system, and communication is done through internet and internet protocols(HTTP Protocol)

2.12 Modules

1. Registration Module
2. Application Module
3. Internal assessment and attendance Module
4. Time table Module
5. HallTicket Module
6. Exam coordinator
7. Exam coordinator

2.12.1 Registration Module:

- **Student Registration:** Registration form for Student Registration, in which Students are registered by Authorized Users/Faculty.
- **Faculty Registration:** Registration form for Faculty Registration.
- **Office Staff:** Registration form for staff
- **Registration Approval:**Approving the details which are given by students, faculty, evaluator and office staffs
- **Login:**A common login page for every users

2.12.2 Application Module

- **Regular:**This modules lists the current semester subject for applying exam
- **Repeater:**This module gives the previous semester details for applying exam
- **Payment:**Here the details of the payment are given by students
- **View Payment:**The payment details are viewed by office staffs

2.12.3 Internal assessment and attendance Module

- Here the faculty gives the marks and attendance of each student according to the subjects which they are teaching

2.12.4 Time table Module

- **Upload:**In this module the admin will upload the timetable
- **Approval:**The department head will approve the time table
- **Display:**The entire timetable will be displayed and their status also be displayed

2.12.5 HallTicket Module

- After approving the time table the hall ticket will be automatically generated.

2.12.6 Staff Module

- Here staffs has to approve the student and faculty who have registered
- Indent of the examination paper must be generated
- Attendance statement of each examination must be uploaded

2.12.7 Exam coordinator

- **Evaluator:** Here the exam coordinator selects the faculty for evaluation process and marks are been uploaded.
- **Attendance statement:** Here the office staff creates a attendance list of students who appeared for the exam and send it to evaluation dept.
- **Coding sheet:**A coding sheet will be generated according to the students register number
- **Marks card:**Marks card is automatically generated after the marks are uploaded

2.13 Functional Requirements

Registration

Student

- FName VARCHAR(20)
- LName VARCHAR(20)
- DOB Date()

- Phone INT(10)
- Address VARCHAR(20)
- Email VARCHAR(20)
- Gender BOOLEAN(2)
- BloodGrp VARCHAR(20)
- Religion VARCHAR(20)
- Caste VARCHAR(20)
- PlaceOfBirth VARCHAR(20)
- IdentityMark VARCHAR(20)
- DistrictOfBirth VARCHAR(20)
- CountryOfBirth VARCHAR(20)
- RNO VARCHAR(20)
- Password VARCHAR(20)
- Pin INT(6)

Faculty

- FName VARCHAR(20)
- LName VARCHAR(20)
- Email VARCHAR(20)
- Id VARCHAR(20)
- Phone INT(10)
- Gender BOOLEAN(2)
- Addr VARCHAR(20)
- bloodGrp VARCHAR(20)
- Religion VARCHAR(20)
- caste VARCHAR(20)
- Nationality VARCHAR(20)
- DOB DATE()
- IdentityMark VARCHAR(20)
- DOJ DATE()
- Exp FLOAT(5)
- Subject VARCHAR(20)
- Dept VARCHAR(20)
- password VARCHAR(20)

Staff

- FName VARCHAR(20)
- LName VARCHAR(20)
- Email VARCHAR(20)

- Id VARCHAR(20)
- Phone INT(10)
- Gender BOOLEAN(2)
- Addr VARCHAR(20)
- bloodGrp VARCHAR(20)
- Religion VARCHAR(20)
- caste VARCHAR(20)
- Nationality VARCHAR(20)
- DOB DATE()
- IdentityMark VARCHAR(20)
- DOJ DATE()
- Dept VARCHAR(20)
- password VARCHAR(20)

Evaluator

- FName VARCHAR(20)
- LName VARCHAR(20)
- Email VARCHAR(20)
- Id VARCHAR(20)
- Phone INT(10)
- Gender BOOLEAN(2)
- Addr VARCHAR(20)
- BloodGrp VARCHAR(20)
- Religion VARCHAR(20)
- caste VARCHAR(20)
- Nationality VARCHAR(20)
- DOB DATE()
- IdentityMark VARCHAR(20)
- DOJ DATE()
- Exp FLOAT(5)
- Subject VARCHAR(20)
- Dept VARCHAR(20)
- password VARCHAR(20)

Login

- Email VARCHAR(20)
- Password VARCHAR(20)

Application

Payment

- BName VARCHAR(20)
- acno INT(10)
- tid VARCHAR(20)
- dop DATE(10)

Internal Assessment

- RegNo VARCHAR(20)
- Name CHAR(20)
- Sem VARCHAR(10)
- Marks FLOAT(10)

Attendance

- RegNo VARCHAR(20)
- Name CHAR(20)
- Sem VARCHAR(10)
- Attendance FLOAT(10)

Time table

Display

- Course CHAR(10)
- Batch INT(5)
- Sem VARCHAR(10)

Evaluation

Attendance

- S_code INT(10)
- S_name VARCHAR(25)
- Tot_std BIGINT(10)
- Tot_absenties BIGINT(10)

Coding sheet

- Name VARCHAR(25)
- Regno INT(10)

- id INT(10)
- Bundleno VARCHAR(10)

2.14 System Attributes

2.14.1 Availability

This system will only available till the system on which it is installed is running. The system should available 24 hours.

2.14.2 Compatibility

This system will be compatible with almost all the web servers.

2.14.3 Flexibility

The system keeps on updating the data according to the changes that takes place.

2.14.4 Maintainability

There is maintenance required for the website. The database is provided by the administrator as well as the end-use.

2.14.5 Portability

This system can be run in any operating system and browser.

2.14.6 Reliability

This system is designed to have very simple database just to extract the details of every users. It is tested for all the constraints at development stage.

2.14.7 Security

This system is provided with authentication without which no user can pass. So only the legitimate users are allowed to use the application. If the legitimate users share the authentication information then the system is open to outsiders.

2.14.8 Timelines

The system carries out all the operations with consumptions of very less time

2.15 Other requirements

2.15.1 Safety Requirements

There are several user levels in SEM software, Access to the various subsystems will be protected by a user log in screen that requires a email id and password. This gives different views and accessible functions of user levels through the system. Maintaining backups ensure the system database security. System can be restored in any case of emergency.

2.15.2 Security Requirements

- Depending upon the category of user the access rights are decided.
- Admin has the maximum privilege to all subsystems.
- Only authenticated users can access this system.

2.15.3. Performance Requirements

In order to maintain an acceptable speed at maximum numbers of upload allowed from a particular users as any number of users can access to the system anytime. Also the connection to the server will be based on the attributes of the user like location and server will be working 24X7 times.

Chapter 3

SYSTEM DESIGN

3.1 INTRODUCTION

System design is the process of defining the components, modules, interfaces and data for a system to satisfy specified requirements. System design involves designing a new system that will meet the requirements identified during system analysis. The focus of system design is on deciding which modules are needed for system, the specifications of these modules and how the modules should be interconnected.

3.2 APPLICABLE DOCUMENTS

The documents used in system design is Software Requirement Specification Document.

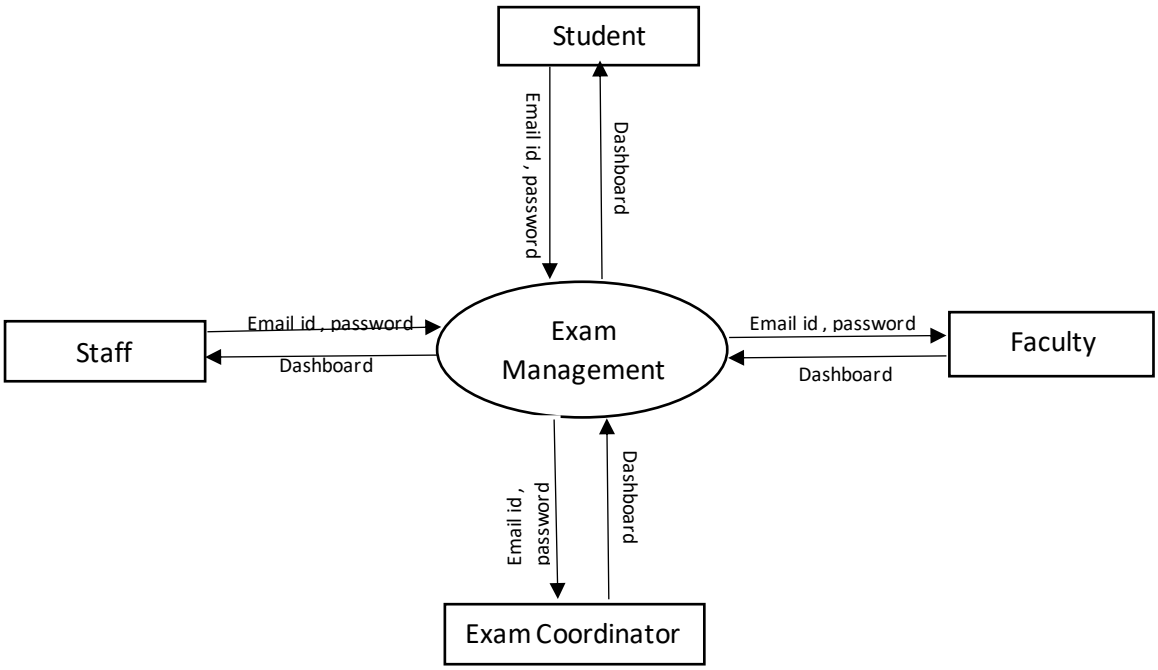
3.3 FUNCTIONAL DECOMPOSITION

The system can be decomposed into functional components as follows

- Registration component for students, faculties and office staffs
- Login components for students, faculties, office staffs, exam coordinator and administrators
- Student can apply for their semester examination and upload their payment details
- Marks and attendance can be entered by faculty
- Approval of student and faculty is done by office staff and admin
- Exam coordinator assigns the faculty for evaluation
- Hall ticket and marks card are automatically generated

3.4 CONTEX FLOW DIAGRAM

It defines the flows of information between the system and external entities. The entire software system is shown in single process. It's designed to be an abstraction view, showing the system as a single process with its relationship to external entities. It represent the entire system as single bubble with input and output data indicated by incoming/outgoing arrows.



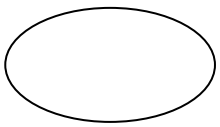
3.5 DATA FLOW DIAGRAM

Data Flow Diagram are the graphical way of showing the flow of data through an information system.It is the common practice for a designer to draw a context level DFD. A complete set of DFD provide a compact top down representation of the system.

It also express the requirement of the system and shows how the current system is implemented. Data Flow Diagram are commonly used during problem analysis. It views a system as a function that perform the input into the desired output.

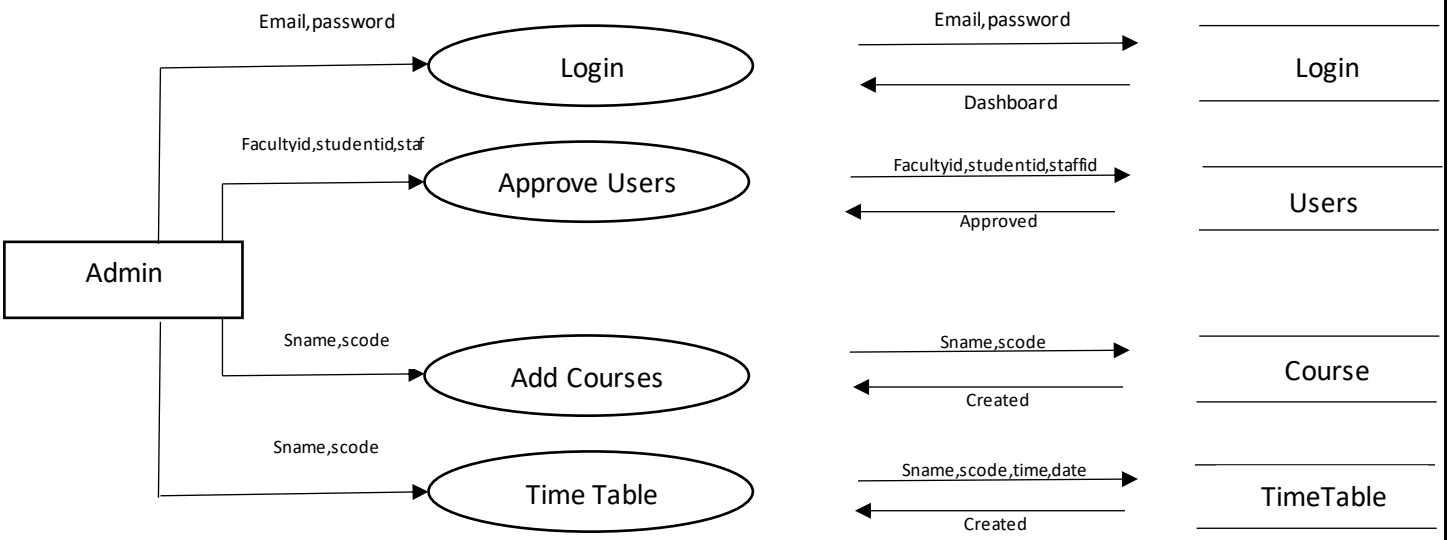
DFD shows the information moves through and how it is modified by a series of transformation. DFD may be used to represent the system at any level of abstraction. DFD can be used to provide the end users with the physical idea of where the data they input ultimately

3.5.1 DFD Symbols:

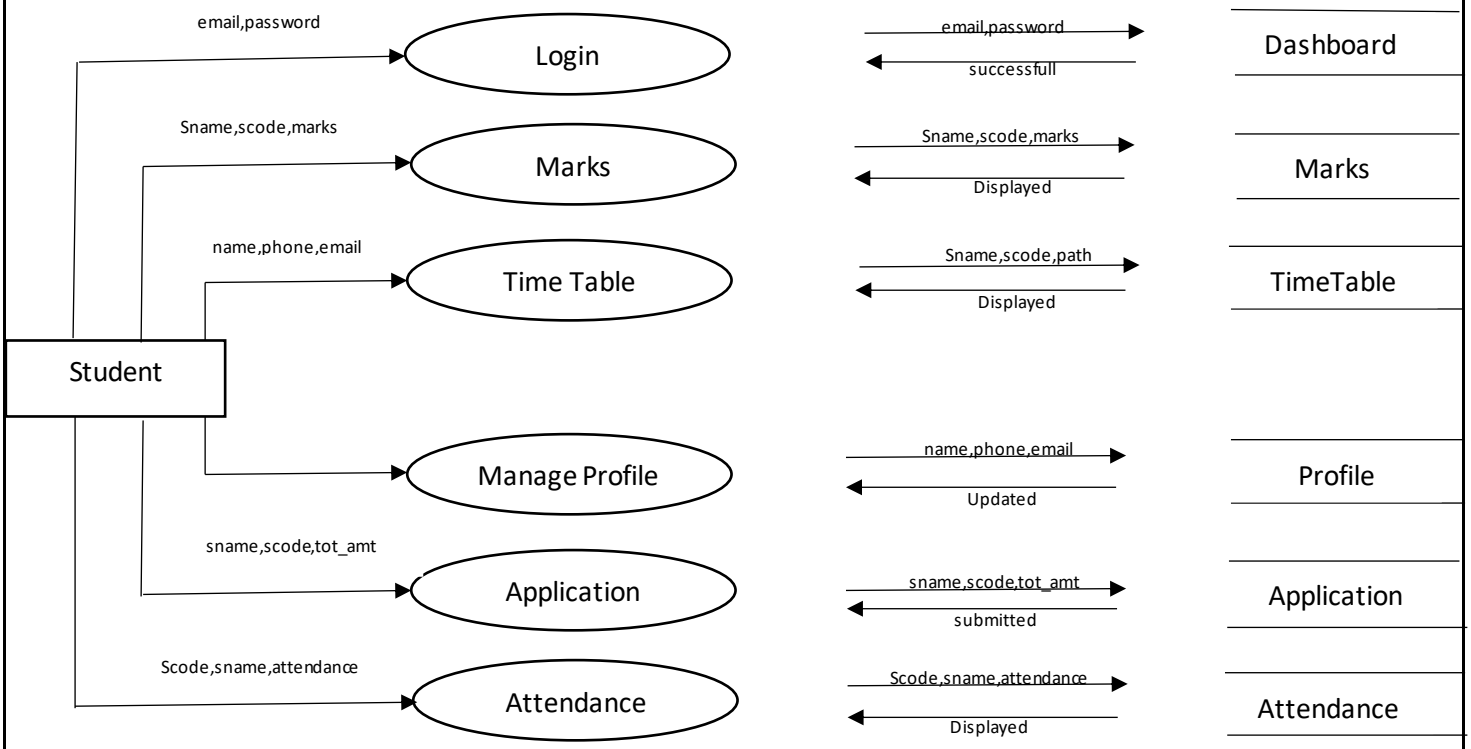
Name	Notation	Description
Process		A process transforms incoming data flow into outgoing data flow. The processes are shown by named circles.

Datastore		Data stores are repositories of data in the system. They are sometimes also referred to as files.
Dataflows		Data flows are pipelines through which packets of information flow. Label the arrows with the name of the data that moves through it.
External Entity		External entities are objects outside the system with which the system communicates. External Entities are sources and destinations of the system's inputs and outputs

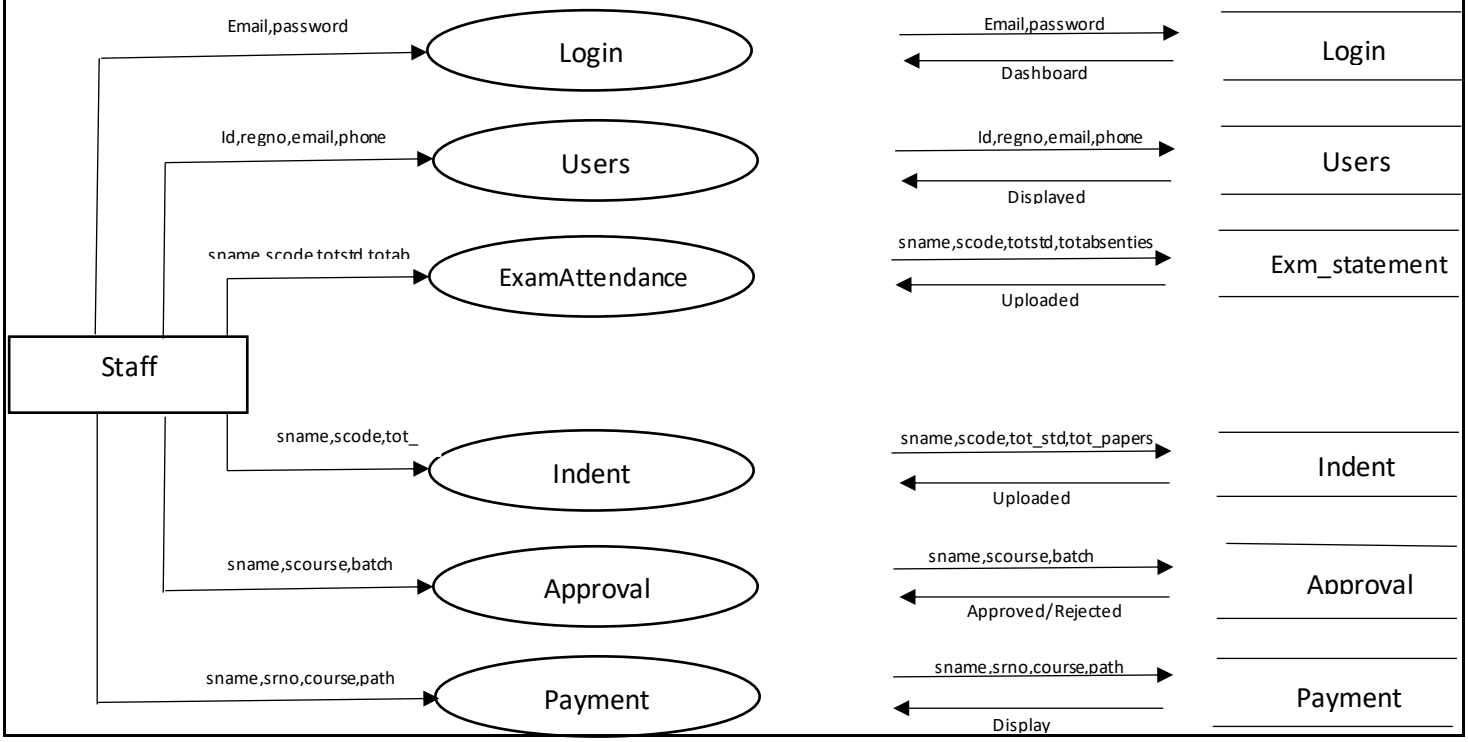
3.5.2 DFD LEVEL 1(ADMIN)



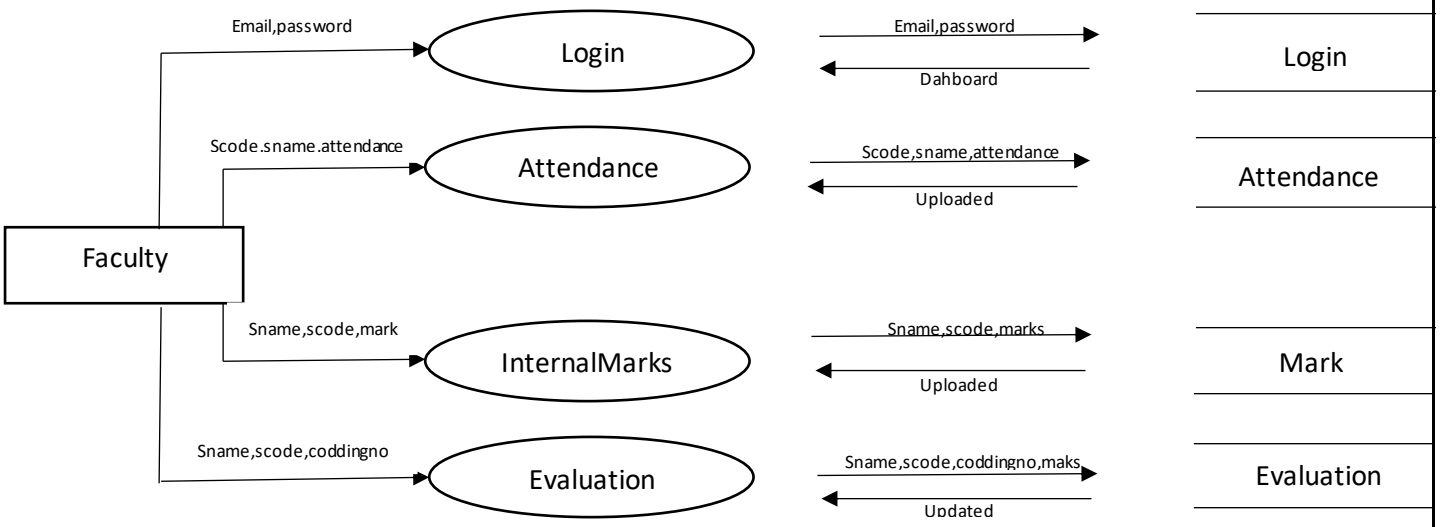
3.5.3 DFD LEVEL 1(STUDENT)



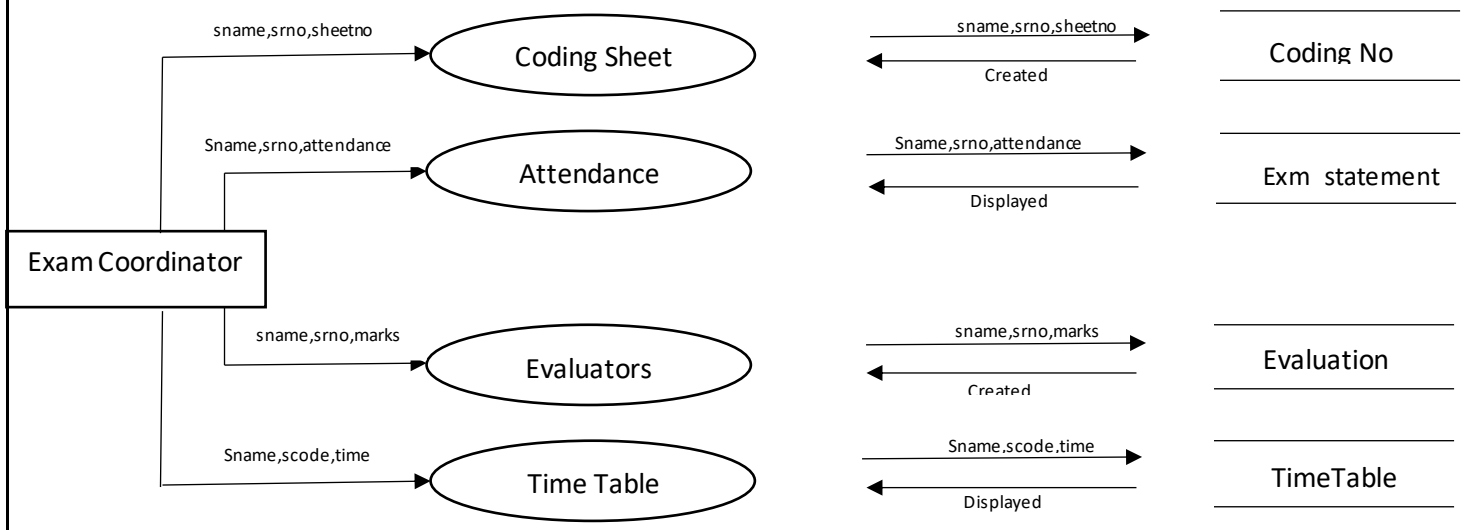
3.5.4 DFD LEVEL 1(OFFICE STAFF)



3.5.5 DFD LEVEL 1(FACULTY)



3.5.6 DFD LEVEL 1(EXAM COORDINATOR)



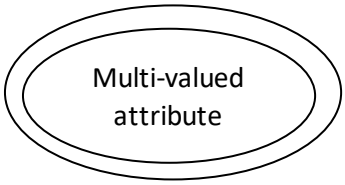
3.6 ENTITY RELATIONSHIP DIAGRAM

An Entity Relation of (ER) Diagram is a specialized graphics that illustrates the interrelationship between entities in a database. Entities are physical items or aggregations of data items that are important to the business that we analysis to the system. It is a framework using specialized symbols to define the relationship between entities. ER diagram is created based on three main components entities, attributes and relationship.

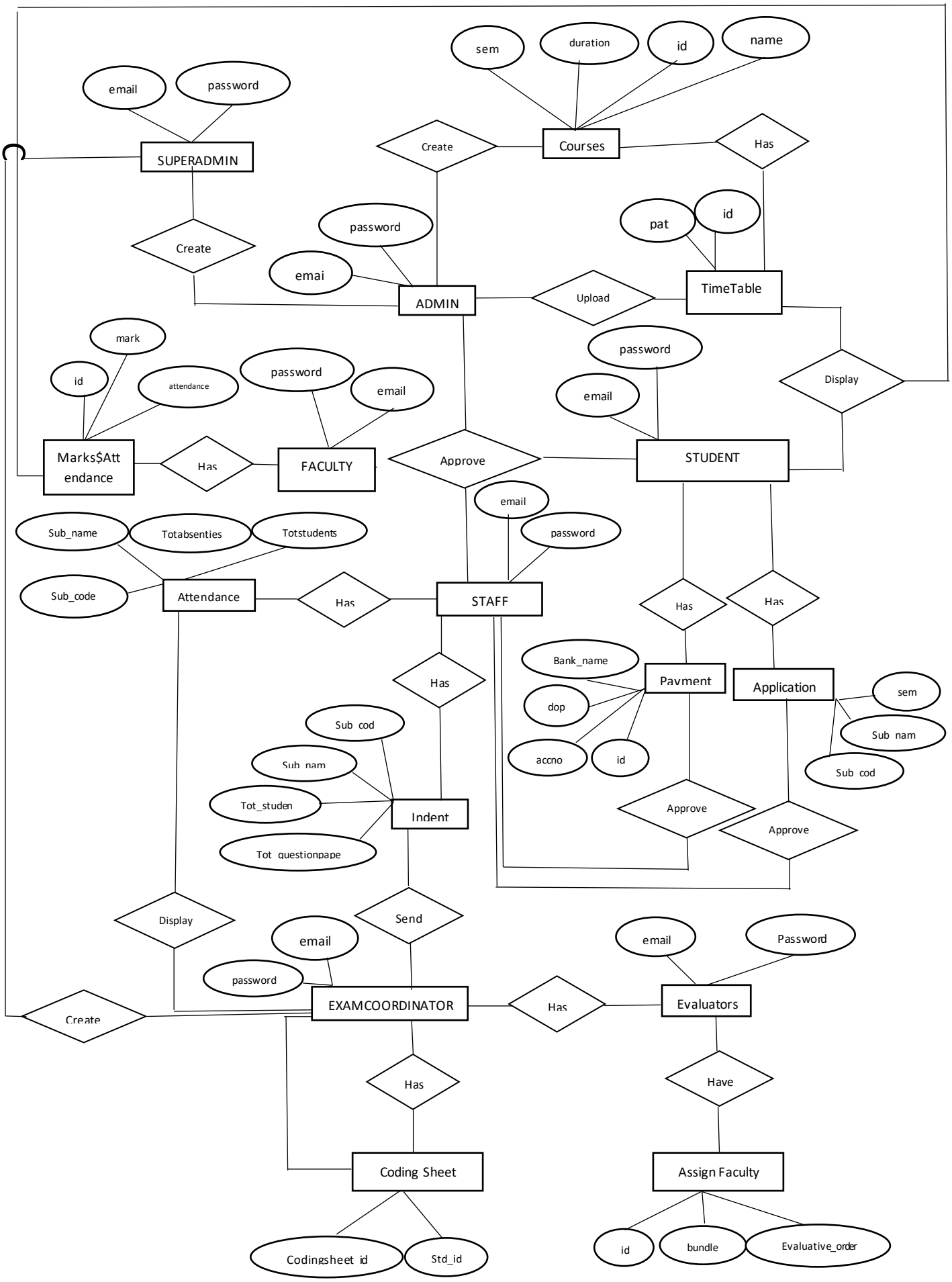
ER helps us conceptualize the database and help us know which fields need to be embedded for a particular entity. ER Diagram gives a better understanding of the information to be stored in a database. Reduces completely and saves time which allows you to build database quickly.

3.6.1 ER-Diagram Symbols:

Name	Notation	Description
Entity	Entity name	It may be an object with the physical existence or conceptual existence. It is represented by a Rectangle.
Attribute	Attribute name	The properties of the entity can be a attribute. It is represented by a Ellipse.
Relationship	Relationship	Whenever an attribute of one entity refers to another entity, some relationship exists. It is represented by a Diamond.
Link		Lines link attributes to entity sets and entity sets to relation.
Derived Attribute	Derived attribute	Dashed ellipse denotes derived attributes.
Key Attribute	Key Attribute	An entity type usually has an attribute whose values are distinct for each individual entry in the entity set. It is represented by a Underlined word in ellipse.

Multivalued Attribute		Attributes that have different numbers of values for a particular attribute. It is represented by a Double ellipse represents multi-valued attributes.
Cardinality Ratio	1) 1:1 2) 1:M 3) M:1 4) M:M	It specifies the maximum number of relationships instances that an entity can participate in. There are four cardinality ratios.

3.6.2 ER DIAGRAM



Chapter 4

4. DETAIL DESIGN

4.1 Introduction

The purpose of this document is to explain complete design details of Srinivas exam manager. It mainly consists of the general definition and features of the project, design constraints, the overall system architecture and data architecture.

Detail design essentially expands the system design and database design it contain a more detailed description of the processing logic and data structures so that design is sufficiently complete of coding

4.2 Applicable Documents

The documents used during detailed design are

- System Requirements Document
- System Design
- Database Design

4.3 Structure of the Software Package

The various functional components used are

- Super Admin module
- Admin module
- Student
- Faculty
- Staff
- Examcoordinator

4.4 Modular Decomposition Components

4.4.1 Super Admin Module

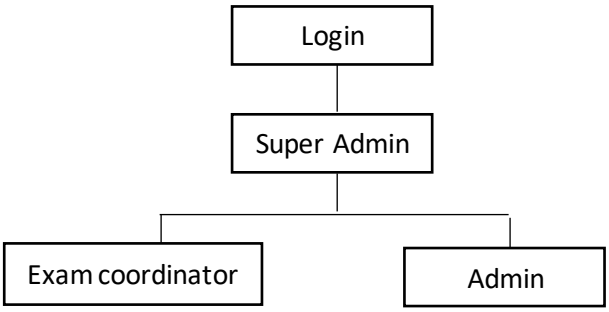
4.4.1.1 Design assumption

The module is designed with an intention to allow the super admin to perform various activities that are specific in the admin menu such as creating admin and exam coordinator.

4.4.1.2 Identification of the module:

- Login
- Create Admin
- Create Exam Coordinator

4.4.1.3 **Structured Chart**



4.4.2 Admin Module

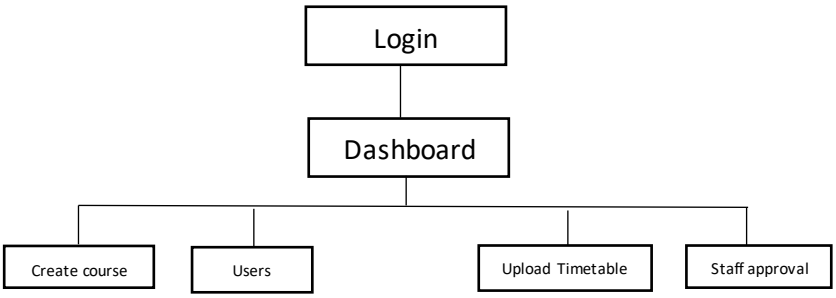
4.4.2.1 Design assumption

In this module it tells about how admin performs various activities such as creating courses, upload time table, staff approval and displaying users

4.4.2.2 Identification of the module:

- Login
- Dashboard
- Create course
- Users
- Upload Time table
- Staff approval

4.4.2.3 Hierarchy of the module:



4.4.3 Student Module

4.4.3.1 Design assumption

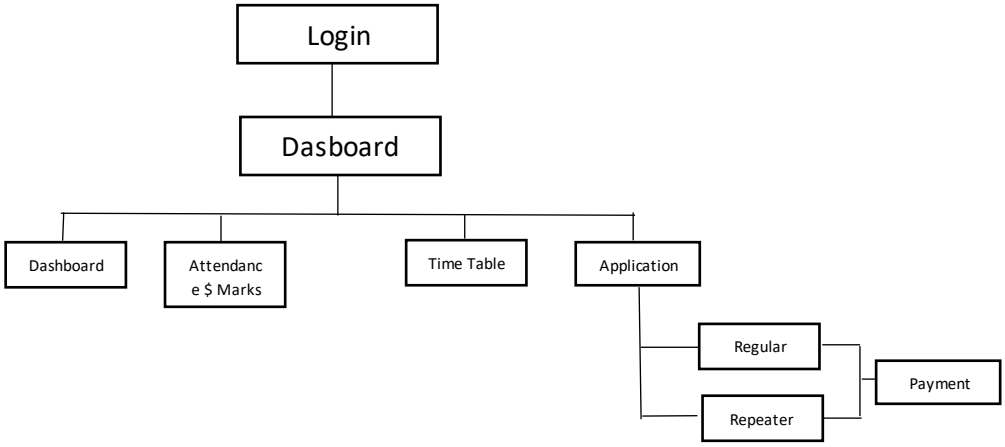
In this module it mainly built to specify certain roles such as displaying student time table, applying for their semester exam, displaying the time table of the exam.

4.4.3.2 Identification of the module:

- Dashboard

- Attendance $ Marks
- Application
- Displaying Time table
- Payment

4.4.3.3 Hierarchy of the module:



4.4.4 Faculty Module

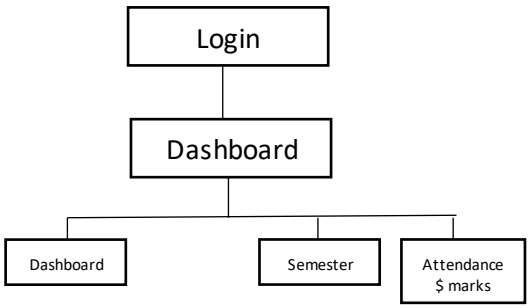
5.4.4.1 Design assumption

Here it mainly describe about how faculty module works and the role which hey perform such as assigning internal marks , attendance and semester marks

5.4.4.2 Identification of the module:

- Dashboard
- Attendance $ Marks
- Semester

5.4.4.3 Hierarchy of the module:



5.4.5 Staff Module

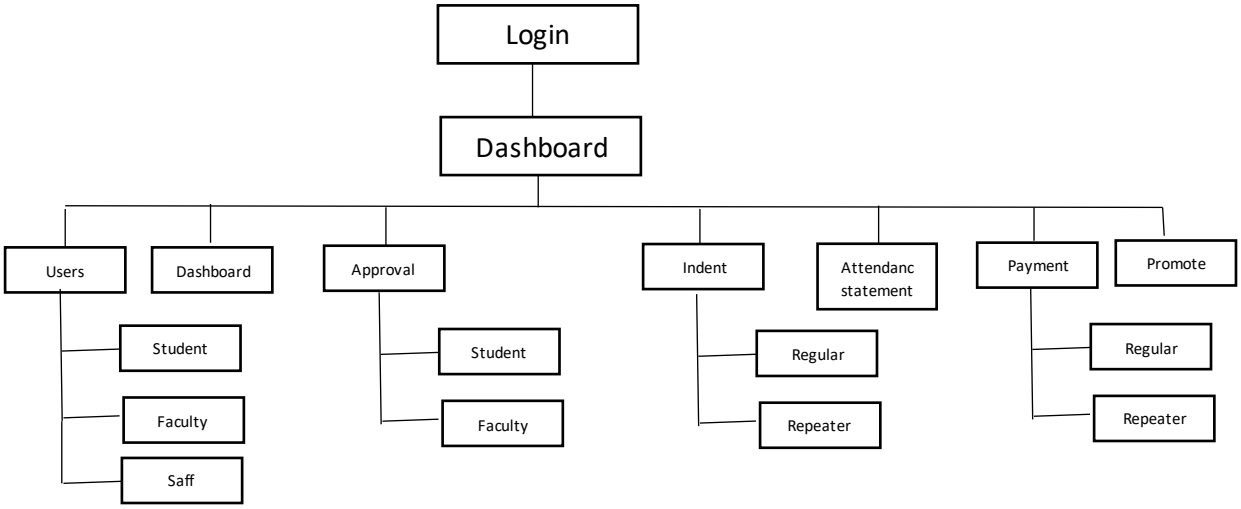
5.4.5.1 Design assumption

In this module it tells about how Staff performs various activities such as displaying user details, approval of students, indent report, attendance statement, payment approval

5.4.5.2 Identification of the module:

- Dashboard
- Displaying users
- Approval
- Indent report
- Attendance statement
- Payment approval

5.4.5.3 Hierarchy of the module:



4.4.6 Exam Coordinator Module

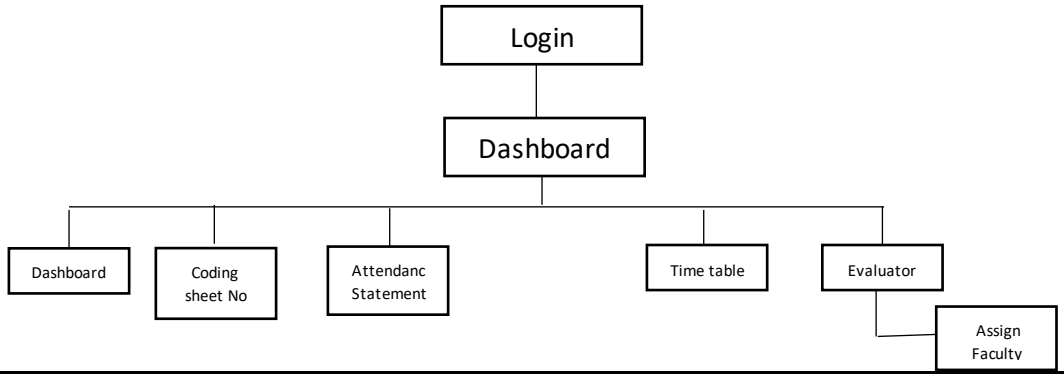
4.4.6.1 Design assumption

Here Exam coordinator function specific task such as assigning coding sheet no, displaying attendance statement, assigning faculty for evaluation and approving time table

4.4.6.2 Identification of the module:

- Dashboard
- Coding sheet No
- Displaying Attendance statement
- Evaluator
- Time table

4.4.6.3 Hierarchy of the module:



Chapter 5

5. DATABASE DESIGN

5.1 DATABASE DESCRIPTION

Database design is an important place in designing a system. The word "database" used to describe everything from a single set of data, to a complex set of tools, such as SQL server and a whole lot in between. The term data model to mean the conceptual description of the problem space. This includes the definition of entities, their attributes and the entity constraints. The data model also includes a description of the relationships between entities and any constraints on those relationships. The main advantage of this software is to reduce the manual work. During this phase care should be taken to avoid redundancy of information storing into a database since it leads to wastage of memory space.

5.1.1 ADMIN TABLE

SLNO	Field Name	Data Type	Field size	Constraint	Description
1	Admin_id	Int	10	Primary key	Admin ID
2	first_name	varchar	255	Not Null	First Name
3	last_name	varchar	255	Not Null	Last Name
4	gender	varchar	255	Not Null	Gender
5	dob	varchar	100	Not Null	Date of birth
6	email	varchar	255	Not Null	Email Id
7	phone	bigint	20	Not Null	Phone No
8	address	varchar	255	Not Null	Address
9	password	varchar	255	Not Null	Password
10	dept_id	int	10	Foreign Key	Department ID

5.1.2 FACULTY TABLE

SLNO	Field Name	Data Type	Field size	Constraint	Description
1	faculty_id	varchar	255	Primary Key	Faculty ID
2	first_name	varchar	255	Not Null	First Name
3	last_name	varchar	255	Not Null	Last Name
4	gender	varchar	255	Not Null	Gender
5	dob	varchar	50	Not Null	Date Of Birth
6	email	varchar	255	Not Null	Email ID
7	phone	bigint	20	Not Null	Phone No
8	address	varchar	255	Not Null	Address
9	blood_group	varchar	255	Not Null	Blood Group
10	caste	varchar	255	Not Null	Caste
11	aadhar_no	bigint	20	Not Null	Aadhar No
12	religion	Varchar	25	Not Null	Religion
13	birth_place	varchar	255	Not Null	Birth Place
14	birth_district	Varchar	255	Not Null	Birth District
15	country	varchar	255	Not Null	Country
16	identity_mark	varchar	255	Not Null	Identity Mark
17	dept_id	int	10	Not Null	Department ID
18	pincode	int	11	Not Null	Pincode
19	password	varchar	155	Not Null	Password
20	f_name	varchar	255	Not Null	Fathers Name

21	f_occupation	varchar	255	Not Null	Fathers Ocuupation
22	f_email	varchar	255	Not Null	Fathers Email
23	f_phone	bigint	20	Not Null	Fathers Phone No
26	teaching_exp	int	4	Not Null	Teaching Experience
27	status	varchar	255	Not Null	Faculty Status
28	role	varachar	255	NotNull	Faculty Role
29	joining_year	int	5	NotNull	Joining Year

5.1.3 STAFF TABLE

SLNO	Field Name	Data Type	Field size	Constraint	Description
1	Staff_id	int	15	Primary Key	Staff ID
2	first_name	varchar	225	Not Null	First Name
3	last_name	varchar	225	Not Null	Last Name
4	gender	varchar	10	Not Null	Gender
5	dob	varchar	20	Not Null	Date Of Birth
6	email	varchar	225	Not Null	Email ID
7	phone	bigint	20	Not Null	Phone No
8	address	varchar	225	Not Null	Address
9	blood_group	varchar	225	Not Null	Blood Group
10	caste	varchar	225	Not Null	Caste
11	religion	Varchar	225	Not Null	Religion
12	documents	varchar	225	Not Null	Documents
13	identity_mark	varchar	225	Not Null	Identity Mark
14	password	varchar	225	Not Null	Password
15	staff_status	varchar	225	Not Null	Staff Status

5.1.4 STUDENT TABLE

SLNO	Field Name	Data Type	Field size	Constraint	Description
1	Std_id	int	10	Primary Key	Student ID
2	first_name	varchar	225	Not Null	First Name
3	last_name	varchar	225	Not Null	Last Name
4	gender	varchar	225	Not Null	Gender
5	dob	date	10	Not Null	Date Of Birth
6	email	varchar	225	Not Null	Email ID
7	phone	bigint	20	Not Null	Phone No
8	address	varchar	225	Not Null	Address
9	blood_group	varchar	225	Not Null	Blood Group
10	caste	varchar	225	Not Null	Caste
11	religion	Varchar	225	Not Null	Religion
12	aadhar_no	bigint	20	Not Null	Aadhar No
13	birth_place	varchar	225	Not Null	Birth Place
14	birth_district	Varchar	225	Not Null	Birth District
15	country	varchar	225	Not Null	Country
16	identity_mark	varchar	225	Not Null	Identity Mark
17	pincode	int	11	Not Null	Pincode
18	password	varchar	225	Not Null	Password
19	f_name	varchar	225	Not Null	Fathers Name
20	f_occupation	varchar	225	Not Null	Fathers Occupation
21	f_email	varchar	225	Not Null	Fathers Email
22	f_phone	bigInt	20	Not Null	Fathers Phone

23	m_name	varchar	225	Not Null	Mothers Name
24	m_occupation	varchar	225	Not Null	Mothers Occupation
25	m_email	varchar	225	Not Null	Mothers Email
26	m_phone	bigint	20	Not Null	Mothers Phone
27	g_name	varchar	225	Not Null	Guardian Name
28	g_occupation	varchar	225	Not Null	Guardian Occupation
29	g_email	varchar	225	Not Null	Guardian Email
30	g_phone	Bigint	20	Not Null	Guardian Phone
31	dept_id	int	10	Not Null	Department ID
32	course_id	int	11	Not Null	Course_ID
33	joining_year	int	11	Not Null	Joining Year
34	status	varchar	255	Not Null	Current Status
35	semester	Int	3	Not Null	Current Semester
36	eligibility	tinyint	1	Not Null	Student Eligibility

5.1.5 COURSE TABLE

SLNO	Field Name	Data Type	Field size	Constraint	Description
1	course_id	int	10	Primary Key	Course ID
2	dept_id	int	10	Foreign Key	Department ID
3	course_name	varchar	100	Not Null	Course Name
4	course_duration	int	11	Not Null	Course Duration
5	course_sem	int	11	Not Null	Course Semester

5.1.6 SEMESTER TABLE.

SLNO	Field Name	Data Type	Field size	Constraint	Description
1	sem_id	int	10	Primary Key	Semester ID
2	course_id	int	10	Foreign Key	Course ID
3	sem_name	varchar	225	Not Null	Semester Name
4	subj_name	vachar	225	Not Null	Subject Name
4	Subj_code	varchar	225	Not Null	Subject Code
5	dept_id	int	10	NotNull	Department ID

5.1.7 TIME TABLE

SLNO	Field Name	Data Type	Field size	Constraint	Description
1	timetable_id	int	10	Primary Key	Timetable ID
2	dept_id	int	11	Foreign Key	Department ID
3	course_id	int	10	Foreign Key	Course ID
4	semester	int	5	Not Null	Semester
5	t_id	varchar	100	Not Null	
6	subj_name	varchar	255	Not Null	Subject Name
7	subj_code	vachar	255	Not Null	Subject Code
8	exam_date	varchar	255	Not Null	Exam Date
9	exam_time	varchar	255	Not Null	Exam Time
10	status	varchar	255	Not Null	Current status
11	created_at	timestamp		Not Null	Created At

5.1.8 DEPARTMENT TABLE

SLNO	Field Name	Data Type	Field size	Constraint	Description
1	dept_id	int	10	Primary Key	Department ID
2	dept_name	varchar	255	Not Null	Department Name

5.1.9 PAYMENT TABLE

SLNO	Field Name	Data Type	Field size	Constraint	Description
1	payment_id	int	10	Primary Key	Payment ID
2	student_id	int	11	Foreign Key	Student ID
3	status	varchar	255	Not Null	Payment Status
4	dop	date		Not Null	Date Of Payment
5	bank_name	varchar	225	Not Null	Bank Name
6	transaction_id	varchar	225	Not Null	Transaction ID
7	acc_no	bigint	20	Not Null	Account No
8	receipt_path	varchar	225	Not Null	Receipt Path

5.1.10 MARKS ATTENDANCE

SLNO	Field Name	Data Type	Field size	Constraint	Description
1	id	int	10	Primary Key	Identity
2	std_id	int	11	Foreign Key	Student ID
3	marks	double	225	Not Null	Student Marks
4	attendance	double	225	Not Null	Student Attendance

5.1.11 ATTENDANCE STATEMENT

SLNO	Field Name	Data Type	Field size	Constraint	Description
1	Id	int	10	Primary Key	Identity
2	subject_name	varchar	255	Not Null	Subject Name
3	subject_code	varchar	155	Not Null	Subject Code
4	tot_students	int	11	Not Null	Total Students
5	tot_absenties	int	11	Not Null	Tot Absenties
6	regno	varchar	50	Not Null	Register No
7	booklet no	int	20	Not Null	Booklet No
8	add_sheet	int	20	Not Null	Additional Sheet
9	abs_no	int	10	Not Null	Absenties No
10	mal_no	int	10	Not Null	Malpractice No
11	name_exm	varchar	255	Not Null	Examiner Name
12	afflication	varchar	20	Not Null	Afflication

5.1.12 CODING SHEET

SLNO	Field Name	Data Type	Field size	Constraint	Description
1	id	Int	10	Primary Key	Identity
2	Std_id	varchar	11	Foreign Key	Student ID
3	Codingsheet_id	varchar	11	Not Null	Coding sheet ID
4	Bundle_no	varchar	10	Not Null	Bundle No

5.1.13 SEMESTER MARKS

SLNO	Field Name	Data Type	Field size	Constraint	Description
1	id	int	10	Primary Key	Identity
2	Codingsheet_id	varchar	225	Foreign Key	Coding sheet ID
3	marks	double	50	Not Null	Student Marks

5.1.14 EXAM COORDINATOR

SLNO	Field Name	Data Type	Field size	Constraint	Description
1	coord_id	int	11	Primary Key	Coordinator ID
2	dept_id	int	10	Foreign Key	Department ID
3	first_name	varchar	255	Not Null	First Name
4	last_name	varchar	255	Not Null	Last Name
5	gender	varchar	255	Not Null	Gender
6	dob	varchar	100	Not Null	Date of birth
7	email	varchar	255	Not Null	Email Id
8	phone	bigint	20	Not Null	Phone No
9	address	varchar	255	Not Null	Address
10	password	varchar	255	Not Null	Password

5.1.15 SUPER ADMIN

SLNO	Field Name	Data Type	Field size	Constraint	Description
1	S_admin_id	int	10	Primary Key	Super admin ID
2	First_name	varchar	70	Not Null	First Name
3	Last_name	varchar	70	Not Null	Last Name
4	email	varchar	200	Not Null	Email ID
5	password	varchar	200	Not Null	Password
6	role	varchar	20	Not Null	Admin Role

5.1.16 REPEATER SUBJECT

SLNO	Field Name	Data Type	Field size	Constraint	Description
1	Rep_id	int	10	Primary Key	Repeater ID
2	Dept_id	int	10	Foreign Key	Department ID
3	Payment_id	varchar	255	Not Null	Payment ID
4	Subj_name	varchar	255	Not Null	Subject Name
5	Subj_code	varchar	255	Not Null	Subject Code

Chapter 6

6. CODING

App.js

```
import "./App.css";
import "./pages/Registration/Registration.css";

import { BrowserRouter as Browser, Routes, Route, Navigate } from "react-router-dom";
import { ThemeProvider, createTheme } from "@mui/material/styles";
import React, { useState, useEffect } from "react";
import { useContextData } from "./hooks/useContextData";

import Login from "./pages/Login/Login";
import SpecialLogin from "./pages/Login/SpecialLogin";


import Dashboard from "./pages/Dashboard/Dashboard";
import Attendance from "./pages/Attendance";
import TimeTable from "./pages/TimeTable";
import PageNotFound from "./pages/PageNotFound/PageNotFound";
import Registration from "./pages/Registration/Registration";
import Student from "./pages/Registration/Student";
import Faculty from "./pages/Registration/Faculty";
import Staff from "./pages/Registration/Staff";

import ProtectedRoute from "./components/ProtectedRoute";

import Courses from "./components/Admin/Course/Course";
import Layout from "./components/Layout";
import Approval from "./components/Approval/Approval";
import ApprovalDetailsView from "./components/Approval/ApprovalDetailsView";
import TotalUsers from "./components/Users/TotalUsers";
import UserDetails from "./components/Users/UserDetails/UserDetails";
```

```
import Create from "./components/Admin/Course/Create/Create";

import CourseDetails from "./components/Admin/Course/CourseDetails/CourseDetails";

import InternalMarks from "./components/Faculty/InternalMarks/InternalMark";

import SemesterMarks from "./components/Faculty/SemesterMarks/SemesterMarks";

import IndentRegular from "./components/Staff/Indent/IndentRegular";

import IndentRepeater from "./components/Staff/Indent/IndentRepeater";

import PaymentsRegular from "./components/Staff/Payments/PaymentsRegular/PaymentsRegular";

import PaymentsRepeater from
"./components/Staff/Payments/PaymentsRepeater/PaymentsRepeater";

import ApplicationRegular from "./components/Student/Application/ApplicationRegular";

import ApplicationRepeater from "./components/Student/Application/ApplicationRepeater";

import Promote from "./components/Staff/Promote/Promote";


import Profile from "./pages/Profile/Profile";

import PaymentsRegularApproval from
"./components/Staff/Payments/PaymentsRegular/PaymentsRegularApproval";

import PaymentsRegularApproved from
"./components/Staff/Payments/PaymentsRegular/PaymentsRegularApproved";

import PaymentsRepeaterApproval from
"./components/Staff/Payments/PaymentsRepeater/PaymentsRepeaterApproval";

import PaymentsRepeaterApproved from
"./components/Staff/Payments/PaymentsRepeater/PaymentsRepeaterApproved";

import Coding from "./components/Evaluator/Coding/Coding";



import ExamAttendance from "./components/Staff/ExamAttendance/ExamAttendance";

import AttendanceStatement from
"./components/Evaluator/AttendanceStatement/AttendanceStatement";

import Evaluators from "./components/Evaluator/Evaluators/Evaluators";

import ExamcordTimeTable from
"./components/Evaluator/ExamcordTimeTable/ExamcordTimeTable";

import StudentTimeTable from "./components/Student/StudentTimeTable/StudentTimeTable";

import AssignFaculty from "./components/Evaluator/Evaluators/AssignFaculty/AssignFaculty";
```

```
import Departments from "./components/AdminSuper/Departments/Departments";
import NewDepartment from
"./components/AdminSuper/Departments/NewDepartment/NewDepartment";


import ExamCoordinator from "./components/AdminSuper/Examcoordinator/ExamCoordinator";
import NewExamCoordinator from
"./components/AdminSuper/Examcoordinator/NewCoordinator/NewCoordinator";
import { CircularProgress } from "@mui/material";
import axios from "axios";


function App() {
  const { setRole, setUser, setToken, token } = useContextData();


  const [loading, setLoading] = useState(true);
  axios.defaults.headers.common['Authorization'] = token;


  //MUI Components Fonts
  const theme = createTheme({
    typography: {
      fontFamily: ["Lato"].join(","),
    },
  });
  //MUI Components Fonts[/]


  //Check previous login credentials
  useEffect(() => {
    let prevUser = localStorage.getItem("user");
    prevUser = JSON.parse(prevUser);


    if (prevUser) {
      setToken(prevUser.token);
      setRole(prevUser.user.role);
      setUser(prevUser.user);
```

```
      setLoading(false);
    }
    setLoading(false);
  }, [setToken, setRole, setUser])

  return (
    <ThemeProvider theme={theme}>
      <div className="App">
        <Browser>
          <Routes>
            {/* Public Routes */}
            {!loading && <Route path="/login" element={!token ? <Login /> : <Navigate to="/" />} />}
            {/* {!token&&<Route path="/login" element={<Login />}/>} */}
            {/* <Route element={<Layout />}><Route path="/" element={<Dashboard/>}/></Route>}
*/}
            {/* <Route path="/login" element={!token?<Login/>:<Navigate to="/dashboard"/>} /> */}
            <Route path="/special" element={!token ? <SpecialLogin /> : <Navigate to="/" />} />
            <Route path="registration" element={!token ? <Registration /> : <Navigate to="/" />} />
            <Route path="registration/student" element={!token ? <Student /> : <Navigate to="/" />} />
            <Route path="registration/faculty" element={!token ? <Faculty /> : <Navigate to="/" />} />
            <Route path="registration/staff" element={!token ? <Staff /> : <Navigate to="/" />} />

            <Route element={<Layout />}>
              {/* Super Admin Access */}
              <Route element={<ProtectedRoute allowedRole={["super admin"]} />}>
                <Route path="departments" element={<Departments />} />
                <Route path="departments/create" element={<NewDepartment />} />
                <Route path="examcoordinator" element={<ExamCoordinator />} />
                <Route path="examcoordinator/create" element={<NewExamCoordinator />} />
              </Route>

              {/* Admin Access */}
              <Route element={<ProtectedRoute allowedRole={["admin"]} />}>
```

```
<Route path="courses" element={<Courses />} />
<Route
  path="courses/course-details/:courseId"
  element={<CourseDetails />}
/>
<Route path="courses/new-course" element={<Create />} />
<Route path="approve/staff" element={<Approval type="staff" />} />
<Route
  path="approve/staff/:id"
  element={<ApprovalDetailsView />}
/>
<Route path="users/examcoordinator" element={<TotalUsers type="examcoordinator" />}
/>
<Route path="users/examcoordinator/:userId" element={<UserDetails />} />
</Route>

{/* Staff Access*/}
<Route element={<ProtectedRoute allowedRole={["staff"]} />}>
  <Route path="approve/student" element={<Approval type="student" />} />
  <Route path="approve/faculty" element={<Approval type="faculty" />} />
  <Route
    path="approve/student/:id"
    element={<ApprovalDetailsView />}
  />
  <Route
    path="approve/faculty/:id"
    element={<ApprovalDetailsView />}
  />
  <Route path="indent/regular" element={<IndentRegular />} />
  <Route path="indent/repeater" element={<IndentRepeater />} />
  <Route path="payments/regular" element={<PaymentsRegular />}>
    <Route path="pending" element={<PaymentsRegularApproval />} />
    <Route
```

```
      path="approved"
      element={<PaymentsRegularApproved />}
    />
  </Route>
  <Route path="payments/repeater" element={<PaymentsRepeater />}>
    <Route
      path="pending"
      element={<PaymentsRepeaterApproval />}
    />
    <Route
      path="approved"
      element={<PaymentsRepeaterApproved />}
    />
  </Route>
  <Route path="/exam-attendance" element={<ExamAttendance />} />
  <Route path="/promote" element={<Promote />} />
</Route>

{/* Faculty Access*/}
<Route element={<ProtectedRoute allowedRole={["faculty"]} />}>
  <Route path="internal" element={<InternalMarks />} />
  <Route path="semester" element={<SemesterMarks />} />
</Route>

{/* Student Access*/}
<Route element={<ProtectedRoute allowedRole={["student"]} />}>

  <Route
    path="application/regular"
    element={<ApplicationRegular />}
  />
  <Route
```

```
    path="application/repeater"
    element={<ApplicationRepeater />}
  />
  <Route path="/studenttimetable" element={<StudentTimeTable />} />
</Route>

{/* Faculty and Student access*/}
<Route
  element={
    <ProtectedRoute allowedRole={["student", "faculty"]} />
  }
>
  <Route
    path="dashboard"
    element={<Dashboard />}
  />
  <Route path="attendance" element={<Attendance />} />
</Route>

{/*Admin, Student and Evaluator access*/}
<Route
  element={
    <ProtectedRoute
      allowedRole={["student", "exam coord", "faculty", "admin"]}
    />
  }
>
  <Route path="timetable" element={<TimeTable />} />
</Route>

{/* Admin and Staff Access */}
<Route
```

```
  element={<ProtectedRoute allowedRole={["admin", "staff"]} />}
>
  <Route path="users/student" element={<TotalUsers type="student" />} />
  <Route path="users/faculty" element={<TotalUsers type="faculty" />} />
  <Route path="users/staff" element={<TotalUsers type="staff" />} />

  <Route path="users/student/:userId" element={<UserDetails />} />
  <Route path="users/faculty/:userId" element={<UserDetails />} />
  <Route path="users/staff/:userId" element={<UserDetails />} />
</Route>

{/* Exam Coordinator Access */}
<Route element={<ProtectedRoute allowedRole={["exam coord"]} />}>
  <Route path="coding" element={<Coding />} />
  <Route
    path="/attendance-statement"
    element={<AttendanceStatement />}
  />
  <Route path="/evaluators" element={<Evaluators />} />
  <Route path="/evaluators/assign" element={<AssignFaculty />} />
  <Route path="/examcordtimetable" element={<ExamcordTimeTable />} />
</Route>

{/*Common Protected Routes */}
<Route
  element={
    <ProtectedRoute
      allowedRole={[
        "super admin",
        "admin",
        "student",
        "exam coord",
```

```
                "faculty",
                "staff",
               ]}
              />
             }
            >
             {/* {token?<Route path="/" element={<Dashboard />}/>:<Route path="/login"
element={<Login/>}/>} */}
             {/* {token&&<Route path="/" element={<Dashboard />}/>} */}
             {/* <Route path="/" element={<Dashboard />}/> */}
             {/* <Route>{token?<Route path="/" element={<Dashboard/>}/>:<Route
element={<Navigate to="/login"/>}/>}</Route> */}


             <Route path="/" element={token ? <Dashboard /> : <Navigate to="/login" />} />
             <Route path="/profile" element={token ? <Profile /> : <Navigate to="/login" />} />
             {/* Testing Route */}
             {/* <Route path="testing" element={<AnyTestComponent />} /> */}
            </Route>
           </Route>
           {/* Page Not Found Route */}
           <Route path="*" element={!loading ? <PageNotFound /> : <div style={{ height: '90v' }}
className="flex"><CircularProgress size={80} /></div>}></Route>
          </Routes>
         </Browser>
       </div>
     </ThemeProvider>
   );
}


export default App;
```

NavLinks.js

```
import { Dashboard, Users, Approval, Courses, Calender, InternalMarks, Attendance, Payment,
Indent, Application, Coding, SemMarks } from "../Assets";
```

```
export const NavLinks = [
  {
    role: "super admin",
    links: [
      {
        title: "Dashboard",
        path: "/",
        icon: Dashboard,
      },
      {
        title: "Departments",
        path: "/departments",
        icon: Courses,
      },
      {
        title: "Exam Coordinator",
        path: "/examcoordinator",
        icon: Approval,
      },
    ],
  },
  {
    role: "admin",
    links: [
      {
        title: "Dashboard",
        path: "/",
        icon: Dashboard,
      },
      {
        title: "Courses",
        path: "/courses",
```

```
  icon: Courses,
},
{
  title: "Users",
  icon: Users,
  subMenu: [
    {
      title: "Student",
      path: "/users/student",
    },
    {
      title: "Faculty",
      path: "/users/faculty",
    },
    {
      title: "Staff",
      path: "/users/staff",
    },
    {
      title: "Exam Coordinator",
      path: "/users/examcoordinator",
    },
  ],
},
{
  title: "Time Table",
  path: "/timetable",
  icon: Calender,
},
{
  title: "Staff Approval",
  path: "approve/staff",
```

```
      icon: Approval
    }
  ],
},
{
  role: "faculty",
  links: [
    {
      title: "Dashboard",
      path: "/",
      icon: Dashboard,
    },
    {
      title: "Attendance",
      path: "/attendance",
      icon: Attendance
    },
    {
      title: "Internal Marks",
      path: "/internal",
      icon: InternalMarks
    },
    {
      title: "Semester Marks",
      path: "/semester",
      icon: SemMarks
    }
  ],
},
{
  role: "staff",
  links: [
```

```
{
  title: "Dashboard",
  path: "/",
  icon: Dashboard,
},
{
  title: "Users",
  icon: Users,
  subMenu: [
    {
      title: "Student",
      path: "/users/student",
    },
    {
      title: "Faculty",
      path: "/users/faculty",
    },
    {
      title: "Staff",
      path: "/users/staff",
    },
  ],
},
{
  title: "Approval",
  icon: Approval,
  subMenu: [
    {
      title: "Student",
      path: "/approve/student",
    },
    {
```

```
      title: "Faculty",
      path: "/approve/faculty",
    },
  ],
},
{
  title: "Indent",
  icon: Indent,
  subMenu: [
    {
      title: "Regular",
      path: "/indent/regular",
    },
    {
      title: "Repeater",
      path: "/indent/repeater",
    },
  ],
},
{
  title: "Attendance",
  icon: Indent,
  path: "/exam-attendance"
},
{
  title: "Payments",
  icon: Payment,
  subMenu: [
    {
      title: "Regular",
      path: "/payments/regular/approved",
    },
```

```
        {
          title: "Repeater",
          path: "/payments/repeater/approved",
        },
      ],
    },
    {
      title: "Promote",
      icon: Indent,
      path: "/promote"
    },
  ],
},
{
  role: "student",
  links: [
    {
      title: "Dashboard",
      path: "/studentdash",
      icon: Dashboard,
    },
    {
      title: "Attendance",
      path: "/attendance",
      icon: Attendance
    },
    {
      title: "Application",
      icon: Application,
      subMenu: [
        {
          title: "Regular",
```

```
          path: "/application/regular",
        },
        {
          title: "Repeater",
          path: "/application/repeater",
        },
      ],
    },
    {
      title: "Time Table",
      path: "/studenttimetable",
      icon: Calender,
    }
  ],
},
{
  role: "exam coord",
  links: [
    {
      title: "Dashboard",
      path: "/",
      icon: Dashboard,
    },
    {
      title: "Coding Sheet",
      path: "/coding",
      icon: Coding,
    },
    {
      title: "Attendance Statement",
      path: "/attendance-statement",
      icon: Coding,
```

```
      },
      {
        title: "Evaluators",
        path: "/evaluators",
        icon: Users,
      },
      {
        title: "Time Table",
        path: "/examcordtimetable",
        icon: Calender,
      },
    ],
  },
];
```

Context.js

```
import React, { useState } from "react";

export const Context = React.createContext();

const ContextProvider = (props) => {
  const [serverUrl] = useState('http://localhost:8080');
  const [role, setRole] = useState('');
  const [user, setUser] = useState({});
  const [token, setToken] = useState('');

  return (
    <Context.Provider
      value={{
        serverUrl,
        token,
        setToken,
        role,
```

```
        setRole,
        user,
        setUser,
      }}
    >
      {props.children}
    </Context.Provider>
  );
};

export default ContextProvider;
```

Create Course

```
import { useNavigate } from "react-router-dom";
import { FiArrowLeft } from "react-icons/fi";
import { HiPlus } from "react-icons/hi";
import {
  TextField,
  FormControl,
  Select,
  MenuItem,
  InputLabel,
  FormHelperText,
} from "@mui/material";
import axios from "axios";

import "./Create.css";
import SemList from "./SemList";
import { useReducer,useState} from "react";

const initialState = {
  name: '',
  duration: '',
```

```
  semesters: [],
};

let semCount = 0;
const courseDetailsReducer = (state, action) => {
  if (action.type === "ADD_SEM") {
    semCount++;
    const sem = {
      semName: "SEM " + semCount,
      subjects: [],
    };
    state.semesters.push(sem);
    return { ...state };
  }

  if (action.type === "REMOVE_SEM") {
    const oldState = [...state.semesters];
    const newState = oldState.filter((_, i) => i !== action.payload);
    state.semesters = newState;
    if (semCount > 0) semCount--;
    return { ...state };
  }

  if (action.type === "ADD_SUBJECT") {
    state.semesters.forEach((sem, i) => {
      if (action.payload.index === i) {
        const updatedSubjects = [...sem.subjects];
        action.payload.subjects.forEach((sub) => updatedSubjects.push(sub));
        state.semesters[i].subjects = updatedSubjects;
      } else {
        return;
      }
```

```
    });
    return { ...state };
  }

  if (action.type === "REMOVE_SUBJECT") {
    state.semesters
      .find((_, i) => i === action.payload.semIndex)
      .subjects.splice(action.payload.subIndex, 1);
    return { ...state };
  }

  if (action.type === "COURSENAME") {
    const newState = {...state};
    newState.name = action.payload;
    return newState;
  }

  if (action.type === "DURATION") {
    const newState = {...state};
    newState.duration= action.payload;
    return newState;
  }
};

const Create = () => {
  const [state, dispatch] = useReducer(courseDetailsReducer, initialState);
  const [errors,setErrors] = useState([]);

  const navigate = useNavigate();

  const addSem = (e) => {
    e.preventDefault();
```

```
  dispatch({ type: "ADD_SEM" });
};

const removeSem = (index) => {
  dispatch({ type: "REMOVE_SEM", payload: index });
};

const addSubjectsToReducer = (subjects) => {
  dispatch({ type: "ADD_SUBJECT", payload: subjects });
};

const removeSubject = (subIndex, semIndex) => {
  dispatch({ type: "REMOVE_SUBJECT", payload: { subIndex, semIndex } });
};
const newCourseSubmit = async(e) => {
  e.preventDefault();
  try {
    state.semesters.forEach((sem,i) => {
      if(!sem.subjects.length>0) {
        throw new Error('Add Subjects');
      }
    })
    const result = await axios.post('/admin/new-course',state);
    console.log(result);
  } catch(err) {
    if(err.response?.status===400) {
     return setErrors(err.response.data);
    }
    console.log(err);
    setErrors([]);
  }
};
```

```
return (
  <div className="create-course-main">
    <div className="back-btn flex" onClick={() => navigate(-1)}>
      <FiArrowLeft
        color="var(--light-grey)"
        className="create-svg"
        size={20}
      />
      <span>Back</span>
    </div>
    <h1 className="main-hdng">New Course</h1>
    <div className="course-details-wrapper">
      <form onSubmit={newCourseSubmit}>
        <div className="course-meta">
          <TextField
            label="Course"
            variant="outlined"
            size="small"
            fullWidth
            error={errors.some((err) => err.param === "name")}
            helperText={errors.find((err) => err.param === "name")?.msg}
            onChange={(e) => dispatch({type:'COURSENAME',payload:e.target.value})}
          />
          <FormControl className="course-duration-select">
            <InputLabel>Duration</InputLabel>
            <Select
              label="Duration"
              defaultValue=""
              size="small"
              type="number"
              error={errors.some((err) => err.param === "duration")}
              onChange={(e) => dispatch({type:'DURATION',payload:e.target.value})}
```

```
            >
              <MenuItem value="1">1 Year</MenuItem>
              <MenuItem value="2">2 Year</MenuItem>
              <MenuItem value="3">3 Year</MenuItem>
              <MenuItem value="4">4 Year</MenuItem>
            </Select>
            <FormHelperText error>{errors.find((err) => err.param ===
"duration")?.msg}</FormHelperText>
          </FormControl>
        </div>
        <div className="semester-wrapper">
          <div className="semester-header">
            <h2>Semesters</h2>
            <button
              className="btn-outlined new-sem-btn flex"
              onClick={(e) => addSem(e)}
            >
              <HiPlus
                size={20}
                color="var(--primary-color) :hover{color:var(--white)}"
              />
              <span>New Sem</span>
            </button>
          </div>
          {state.semesters.map((semester, index) => {
            return (
              <SemList
                key={Math.random()}
                details={semester}
                removeSem={removeSem}
                index={index}
                addSubjectsToReducer={addSubjectsToReducer}
                removeSubject={removeSubject}
```

```
            />
          );
        })}

        {semCount > 0 && (
          <button className="btn course-submit-btn" type="submit">
            Submit
          </button>
        )}
      </div>
    </form>
  </div>
</div>
);
};

export default Create;
```

New Department

```
import { useRef, useState } from "react";
import "./NewDepartment.css"
import Back from "../../../UI/Back/Back"
import Dob from "../../../UI/Dob";
import RadioInput from "../../../UI/RadioInput";
import { TextField } from "@mui/material";
import axios from "axios";
import dateFormat from "dateformat";

const NewDepartment = () => {
    const [gender,setGender] = useState('');
    const [passErr,setPassErr] = useState(false);
    const [errors,setErrors] = useState([]);
```

```
  const departmentNameRef = useRef();
  const firstNameRef = useRef();
  const lastNameRef = useRef();
  const dateRef = useRef();
  const monthRef = useRef();
  const yearRef = useRef();
  const dobRef = useRef({dateRef,monthRef,yearRef});
  const emailRef = useRef();
  const phoneRef = useRef();
  const addressRef = useRef();
  const passwordRef = useRef();
  const cPasswordRef = useRef();

const handleFormSubmit = async(e) => {
  e.preventDefault();
  const dob = `${dateRef.current.value}-${monthRef.current.value}-${yearRef.current.value}`
  const dobErr = dob.length>=10;
  const adminData = {
    departmentName : departmentNameRef.current.value,
    firstName : firstNameRef.current.value,
    lastName : lastNameRef.current.value,
    dob : dobErr&&dateFormat(dob,"dd-mm-yyyy"),
    gender: gender,
    email : emailRef.current.value,
    phone : phoneRef.current.value,
    address : addressRef.current.value,
    password : passwordRef.current.value,
    cPasword : cPasswordRef.current.value,
  }

  if(adminData.password!==adminData.cPasword) {
    setPassErr(true);
```

```
  } else {
    try {
      const result = await axios.post('/admin/department/new-department',adminData)
      console.log(result);
      setErrors([]);
      setPassErr(false);
    } catch(err) {
      if(err?.response?.status===400) {
       setErrors(err.response.data.err);
      }
      console.log(err)
    }
  }
}

return (
  <div className="newdept-container">
      <Back left="0em"/>
      <h2 className="newdept-title">Create New Department</h2>
      <form onSubmit={handleFormSubmit}>
          <div className="newdept-deptName">
              <TextField
                  label="Department Name"
                  variant="outlined"
                  size="small"
                  fullWidth
                  inputRef={departmentNameRef}
                  error={errors.some(err=>err.param==='departmentName')}
                  helperText={errors.find(err=>err.param==='departmentName')?.msg}
              />

              <h3>Admin Details</h3>
```

```
</div>

<div className="newdept-adminForm">
  <TextField
    label="First Name"
    variant="outlined"
    size="small"
    fullWidth
    inputRef={firstNameRef}
    error={errors.some(err=>err.param==='firstName')}
    helperText={errors.find(err=>err.param==='firstName')?.msg}
  />

  <TextField
    label="Last Name"
    variant="outlined"
    size="small"
    fullWidth
    inputRef={lastNameRef}
  />

  <TextField
    label="Phone"
    variant="outlined"
    size="small"
    type="number"
    fullWidth
    inputRef={phoneRef}
    error={errors.some(err=>err.param==='phone')}
    helperText={errors.find(err=>err.param==='phone')?.msg}
  />
```

```
  <TextField
    label="Email"
    variant="outlined"
    size="small"
    type="text"
    fullWidth
    inputRef={emailRef}
    error={errors.some(err=>err.param===='email')}
    helperText={errors.find(err=>err.param===='email')?.msg}
  />

  <Dob
  ref={dobRef}
  error={errors.some(err=>err.param===='dob')}
helperText={errors.find(err=>err.param===='dob')?.msg}
  />

  <RadioInput
  setGender={setGender}
  error={errors.some(err=>err.param===='gender')}
helperText={errors.find(err=>err.param===='gender')?.msg}
/>

  <TextField
    multiline
    label="Address"
    rows={2}
    className="textarea"
    inputRef={addressRef}
    error={errors.some(err=>err.param===='address')}
    helperText={errors.find(err=>err.param===='address')?.msg}
  />
```

```
            <TextField
              label="Password"
              variant="outlined"
              type="password"
              size="small"
              fullWidth
              inputRef={passwordRef}
              error={errors.some(err=>err.param==='password')}
              helperText={errors.find(err=>err.param==='password')?.msg}
            />

            <TextField
              label="Confirm Password"
              variant="outlined"
              type="password"
              size="small"
              error={passErr}
              helperText={passErr&&'Passwords does not match'}
              fullWidth
              inputRef={cPasswordRef}
            />
          </div>
          <input className="newdept-createBtn" type="submit" value="Create"/>
        </form>
    </div>
  )
}

export default NewDepartment;
```

Registration.js

```
import { Link } from "react-router-dom";
```

```
import StudentSvg from "../../Assets/Registration/student_reg.svg";
import FacultySvg from "../../Assets/Registration/faculty_reg.svg";
import StaffSvg from "../../Assets/Registration/staff_reg.svg";
import Navbar from "../../components/Navbar/Navbar";

const Registration = () => {
  return (
    <>
      <Navbar />
      <div className="registration-container">
        <h1>Register</h1>
        <div className="registration-wrapper">
          <Link to="student">
            <div className="registration-card">
              <img src={StudentSvg} alt="Student Svg"/>
              <h3>Student</h3>
            </div>
          </Link>
          <Link to="faculty">
            <div className="registration-card">
              <img src={FacultySvg} alt="Faculty Svg" />
              <h3>Faculty</h3>
            </div>
          </Link>
          <Link to="staff">
            <div className="registration-card">
              <img src={StaffSvg} alt="Staff Svg" />
              <h3>Staff</h3>
            </div>
          </Link>
```

```
        </div>
      </div>
    </>
  );
};

export default Registration;
```

Login.js

```
import { Link,useNavigate} from "react-router-dom";
import { useState} from "react";
import { FiMail, FiLock, FiArrowLeft } from "react-icons/fi";

import "./Login.css";
import { SrinivasLogo, LoginSvg } from "../../Assets";
import StudentSvg from "../../Assets/Registration/student_reg.svg";
import FacultySvg from "../../Assets/Registration/faculty_reg.svg";
import StaffSvg from "../../Assets/Registration/staff_reg.svg";
import axios from "axios";
import {useContextData} from "../../hooks/useContextData";
import { CircularProgress } from "@mui/material";

const Login = () => {
  const [emailfocus, setEmailFocus] = useState(false);
  const [passfocus, setPassFocus] = useState(false);
  const [loginUser, setLoginUser] = useState("");
  const [email,setEmail] = useState('');
  const [password,setPassword] = useState('');
  const [loading,setLoading] = useState(false);
  const [errors,setErrors] = useState('');
  const onEmailActive = () => setEmailFocus(true);
  const onPassActive = () => setPassFocus(true);
```

```
const onEmailBlur = () => setEmailFocus(false);
const onPassBlur = () => setPassFocus(false);

const emailAct = emailfocus ? "form-control active" : "form-control";
const passAct = passfocus ? "form-control active" : "form-control";

const {setRole,setUser,setToken} = useContextData();
const navigate = useNavigate();

const handleLogin = async(e) => {
  e.preventDefault();

  if(email===""||password===="") {
    return;
  }

  const data = {
    email,
    password,
    role:loginUser
  }

  try {
    setLoading(true);
    const result = await axios.post('/login',data);
    setRole(result.data.user.role);
    setToken(result.data.token);
    setUser(result.data.user);

    let userData = {
      role: result.data.user.role,
      token:result.data.token,
```

```
      user:result.data.user
    }
    localStorage.setItem("user",JSON.stringify(userData));
    navigate('/');
    setLoading(false);
  } catch(err) {
    if(err.response.status===404)
    setErrors(err.response.data.error);
    console.log(err);
    setLoading(false);
  }
}

return (
  <div className="login-container">
    {/* Login Side Design */}
    <div className="login-art">
      <div className="login-logo">
        <img width="50px" src={SrinivasLogo} alt="Login SVG" />
        <h1>Srinivas Exam Manager</h1>
      </div>

      <img
        className="login-vector"
        width="400px"
        src={LoginSvg}
        alt="Login SVG"
      />
    </div>

    {/* Login Form */}
    <div className="login-form">
```

```
<h1 className="login-hdng">{loginUser ? "Login as "+loginUser.charAt(0).toUpperCase() +
loginUser.slice(1) : "Select Login User"}</h1>


{!loginUser ? <div className="login-userSelectContain flex">
  <div className="login-userSelect">
    <div className="login-userBox" onClick={()=>{setLoginUser("student")}}>
        <img src={StudentSvg} alt="Student Svg" width="100px"/>
        <h3>Student</h3>
    </div>

    <div className="login-userBox" onClick={()=>{setLoginUser("faculty")}}>
        <img src={FacultySvg} alt="Faculty Svg" width="100px"/>
        <h3>Faculty</h3>
    </div>

    <div className="login-userBox" onClick={()=>{setLoginUser("staff")}}>
        <img src={StaffSvg} alt="Staff Svg" width="100px"/>
        <h3>Staff</h3>
    </div>
  </div>

  <div className="to-register flex">
      <p>Dont have an account yet ?</p>
      <Link to="/registration">Register</Link>
  </div>
</div>
:
<form onSubmit={handleLogin}>
  <div className="login-backBtn flex" onClick={()=>{setLoginUser("")}}>
    <FiArrowLeft color="var(--text-color)" size={25}/>
    <span>Back</span>
  </div>
```

```
<div className={emailAct}>
  <label className="login-label">Email</label>
  <div className="input-group">
    <input
      type="text"
      onFocus={onEmailActive}
      onBlur={onEmailBlur}
      onChange={(e) => setEmail(e.target.value)}
      placeholder="example@gmail.com"
    />
    <FiMail size={30} color="var(--light-grey)" />
  </div>
</div>
<div className={passAct}>
  <label className="login-label label-pass">Password</label>
  <div className="input-group">
    <input
      type="password"
      onFocus={onPassActive}
      onBlur={onPassBlur}
      onChange={(e) => setPassword(e.target.value)}
      placeholder="Password"
    />
    <FiLock size={30} color="var(--light-grey)" />
  </div>
</div>
{errors&&<div style={{color:'red',fontSize:'0.8em'}}>{errors}</div>}
<div className="forgot-pass">
  <Link to="#">Forgot Password ?</Link>
</div>
<div className="form-controls">
  <button type="submit" className="login-submit btn">{loading?<CircularProgress
color="inherit" size={20}/>:'Login'}</button>
```

```
          </div>
        </form>}
      </div>
    </div>
  );
};

export default Login;
```

app.js – backend

```
const express = require('express');
const cors = require('cors');
require('dotenv').config();
const bodyParser = require('body-parser');
const path = require('path');
const app = express();

const registrationRoutes = require('./Router/registrationRoutes');
const adminRoutes = require('./Router/adminRoutes');
const staffRoutes = require('./Router/staffRoutes');
const examcoordRoutes = require('./Router/examcoordRoutes');
const studentRoutes = require('./Router/studentRoutes');
const routes = require('./Router/routes');

app.use(bodyParser.urlencoded({ extended: false }));
app.use(bodyParser.json());
const imgPath = path.join(__dirname, 'uploads');
app.use(express.static(imgPath));

app.use(express.json());
app.use(cors());

app.use('/admin',adminRoutes);
```

```
app.use('/staff',staffRoutes);
app.use('/examcoord',examcoordRoutes);
app.use('/student',studentRoutes);
app.use(routes);
app.use(registrationRoutes);

const PORT = process.env.PORT||8080;
app.listen(PORT);
```

isAuth.js

```
const jwt = require("jsonwebtoken");

module.exports = (req, res, next) => {
    const headerToken = req.get("Authorization");
    if (!headerToken) {
        return res.status(401).json({ error: "Not Authorized" });
    }
    let decodedToken;
    try {
        decodedToken = jwt.verify(headerToken, process.env.SECRET_KEY);
        if(decodedToken) {
            req.deptId = decodedToken.deptId;
            req.userId = decodedToken.id;
        } else {
            throw new Error('Not Authorized');
        }
    } catch (err) {
        return res.status(401).json({ error: err });
    }
    next();
};
```

Multer.js

```
const multer  = require('multer');
```

```
const path = require('path');

const storage = multer.diskStorage({
    destination: function (req, file, cb) {
      const imgPath = path.join(__dirname,'..','uploads',file.fieldname+'s');
      cb(null, imgPath)
    },
    filename: function (req, file, cb) {
      const uid = (Math.random() + 1).toString(36).substring(2);
      const mimeType = file.mimetype.split('/')[1];
      cb(null, uid+'.'+mimeType)
    }
  })

const upload = multer({ storage: storage });

module.exports = upload;
```

adminControler.js

```
const db = require('../db');
const bcrypt = require('bcrypt');
const { validationResult } = require('express-validator');

exports.postNewCourse = async (req, res) => {
    const { duration, name, semesters } = req.body;
    const deptId = req.deptId;
    const err = validationResult(req).errors;
    if (err.length > 0) {
        return res.status(400).send(err);
    }
    const totalSemesters = semesters.length;
    const checkCourse = await db.execute(`select course_name from course where
course_name='${name}'`);
```

```
    if (checkCourse[0].length > 0) {
        return res.status(403).send('course Already exists');
    }
    const courseSql = 'insert into course(dept_id,course_name,course_duration,course_sem)
values(?,?,?,?)';
    db.execute(courseSql, [deptId, name, duration, totalSemesters])
        .then(() => {
            return db.execute(`select course_id from course where course_name='${name}'`)
        })
        .then(([result]) => {
            semesters.forEach(sem => {
                sem.subjects.forEach(sub => {
                    const semSql = `insert into semester(dept_id,course_id,sem_name,subj_name,subj_code)
values(?,?,?,?,?)`;
                    return db.execute(semSql, [deptId, result[0].course_id, sem.semName, sub.name,
sub.code])
                })
            })
        })
        .then((result) => {
            res.status(200).send({ success: true, result })
        })
        .catch(err => {
            console.log(err);
            res.status(500).send({ success: false, err });
        })
}

exports.postNewDepartment = async (req, res) => {
    const { departmentName, firstName, lastName, dob, email, gender, address, phone, password } =
req.body;
    const err = validationResult(req).errors;
    if (err.length > 0) {
        return res.status(400).send({ success: false, err });
```

```
    }
    try {
        const hashedPassword = await bcrypt.hash(password, 4);
        db.execute(`insert into department(dept_name) values(?)`, [departmentName])
            .then(() => {
                return db.execute(`select dept_id from department where
dept_name='${departmentName}'`);
            })
            .then(([result]) => {
                const deptId = result[0].dept_id;
                const adminSql = `insert into
admin(dept_id,first_name,last_name,gender,dob,email,address,phone,password)
values(?,?,?,?,?,?,?,?,?)`;
                return db.execute(adminSql, [deptId, firstName, lastName, gender, dob, email, address,
phone, hashedPassword])
            })
            .then(() => {
                res.send({ success: true, msg: 'Inserted Successfully' })
            })
            .catch(err => {
                // console.log(err);
                return res.status(500).send({ success: false, err });
            })
    } catch (err) {
        // console.log(err);
        return res.status(500).send({ success: false, err });
    }
}

exports.getDepartments = async (req, res) => {
    try {
        const result = await db.execute('select department.dept_id,dept_name,first_name from
department JOIN admin ON department.dept_id=admin.dept_id');
        res.send(result[0]);
```

```
    } catch (err) {
        res.status(500).send({ success: false, err })
    }
}

exports.getCourses = async (req, res) => {
    const deptId = req.deptId;
    try {
        const result = await db.execute(`select course_id,course_name,course_sem,course_duration from
course where dept_id=${deptId}`);
        res.send(result[0]);
    } catch (err) {
        console.log(err);
        res.status(404).send({ success: false });
    }
}

exports.getCourseDetails = async (req, res) => {
    const deptId = req.deptId;
    try {
        const result = await db.execute(`select sem_id,sem_name,subj_code,subj_name from semester
where dept_id=${deptId} AND course_id=${req.body.courseId} order by sem_name`);
        res.send(result[0]);
    } catch (err) {
        console.log(err);
        res.status(404).send({ success: false });
    }
}

exports.postNewCoordinator = async (req, res) => {
    const { departmentName, firstName, lastName, dob, email, gender, address, phone, password } =
req.body;
    const err = validationResult(req).errors;
```

```
    if (err.length > 0) {
        return res.status(400).send({ success: false, err });
    }
    try {
        const hashedPassword = await bcrypt.hash(password, 4);
        db.execute(`select dept_id from department where dept_name='${departmentName}'`)
            .then(([result]) => {
                const deptId = result[0].dept_id;
                const coordSql = `insert into
exam_coord(dept_id,first_name,last_name,gender,dob,email,address,phone,password)
values(?,?,?,?,?,?,?,?,?)`;
                return db.execute(coordSql, [deptId, firstName, lastName, gender, dob, email, address,
phone, hashedPassword])
            })
            .then(() => {
                res.send({ success: true, msg: 'Inserted Successfully' })
            })
            .catch(err => {
                // console.log(err);
                return res.status(500).send({ success: false, err });
            })
    } catch (err) {
        // console.log(err);
        return res.status(500).send({ success: false, err });
    }
}


exports.getExamCoordinators = async (req, res) => {
    try {
        const result = await db.execute('select dept_name,first_name,coord_id from exam_coord join
department on exam_coord.dept_id=department.dept_id');
        res.send(result[0]);
    } catch (err) {
        console.log(err);
```

```
        res.status(500).send(err);
    }
}

exports.getStaffApproveList = async (req, res) => {
    const deptId = req.deptId;
    try {
        const sql = `select staff_id,first_name,last_name,joining_year from staff where
dept_id=${deptId} and status='pending'`;
        const result = await db.execute(sql);
        res.send(result[0]);
    } catch (err) {
        res.status(500).send(err);
        console.log(err);
    }
}

exports.getApproveStaffDetail = async (req, res) => {
    const id = req.params.id;
    try {
        const result = await db.execute(`select * from staff join department on
staff.dept_id=department.dept_id where staff_id='${id}'`);
        res.status(200).send(result[0]);
    } catch (err) {
        console.log(err);
        res.status(500).send({ success: false })
    }
}

exports.postApproveStaff = async (req, res) => {
    const id = req.params.id;
    try {
        const result = await db.execute(`update staff set status='approved' where staff_id='${id}'`);
```

```
        res.status(200).send({ success: true, data: result });
    } catch (err) {
        console.log(err);
        res.status(500).send({ success: false })
    }
}

exports.postRejectStaff = async (req, res) => {
    const id = req.params.id;
    try {
        const result = await db.execute(`delete from staff where staff_id='${id}'`);
        res.status(200).send({ success: true, data: result[0] });
    } catch (err) {
        console.log(err);
        res.status(500).send({ success: false })
    }
}

exports.postNewTimeTable = (req, res) => {
    const { courseName, semester, timetable, tId } = req.body;
    const deptId = req.deptId;

    const sql = `insert into
timetable(dept_id,course_id,semester,t_id,subj_name,subj_code,exam_date,exam_time,status)
values(?,(select course_id from course where course_name=?),?,?,?,?,?,?,?)`;
    try {
        timetable.forEach(async ({ subjectName, subjectCode, examDate, examTime }) => {
            await db.execute(sql, [deptId, courseName, semester, tId, subjectName, subjectCode,
examDate, examTime, 'pending']);
        })
        res.status(200).send({ success: true, result: 'Inserted Successfully' });
    } catch (err) {
        console.log(err);
```

```
        res.status(500).send({ success: false })
    }
}


exports.getTimetables = async (req, res) => {
    const deptId = req.deptId;
    const sql = `select course_name,t_id,semester,count(subj_name) as
total_subjects,date_format(convert_tz(created_at,@@session.time_zone,'+05:30'),'%d %b-%Y %l:%i
%p') created_at,status from timetable join course on timetable.course_id=course.course_id where
timetable.dept_id=${deptId} group by t_id;`;
    try {
        const result = await db.execute(sql);
        res.status(200).send(result[0]);
    } catch (err) {
        console.log(err);
        res.status(500).send({ success: false })
    }
}
```

examcoordControler.js

```
const db = require('../db');


exports.getTimetables = async(req,res) => {
        const deptId = req.deptId;
        const sql = `select course_name,t_id,semester,count(subj_name) as
total_subjects,date_format(convert_tz(created_at,@@session.time_zone,'+05:30'),'%d %b %Y %l:%i
%p') created_at,status from timetable join course on timetable.course_id=course.course_id where
timetable.dept_id=${deptId} and timetable.status='pending' group by t_id;`;
        try {
            const result = await db.execute(sql);
            res.status(200).send(result[0]);
        } catch(err) {
            console.log(err);
            res.status(500).send({success:false})
        }
```

```
}

exports.getTimetableDetails = async(req,res) => {
    const deptId = req.deptId;
    const tId = req.params.tId;
    const sql = `select subj_name,subj_code,exam_date,exam_time from timetable join course on timetable.course_id=course.course_id where timetable.dept_id=${deptId} and t_id='${tId}' order by exam_date`;
    try {
        const result = await db.execute(sql);
        res.status(200).send(result[0]);
    } catch(err) {
        console.log(err);
        res.status(500).send({success:false})
    }
}

exports.postApproveTimetable = async(req,res) => {
    const deptId = req.deptId;
    const tId = req.params.tId;
    const sql = `update timetable set status='approved' where dept_id=${deptId} and t_id='${tId}'`;
    try {
        const result = await db.execute(sql);
        res.status(200).send(result[0]);
    } catch(err) {
        console.log(err);
        res.status(500).send({success:false})
    }
}

exports.postRejectTimetable = async(req,res) => {
    const deptId = req.deptId;
    const tId = req.params.tId;
```

```
    const sql = `update timetable set status='rejected' where dept_id=${deptId} and t_id='${tId}'`;
    try {
        const result = await db.execute(sql);
        res.status(200).send(result[0]);
    } catch(err) {
        console.log(err);
        res.status(500).send({success:false})
    }
}
```

registrationController.js

```
const { validationResult } = require("express-validator");
const db = require("../db");
const bcrypt = require("bcrypt");
const jwt = require('jsonwebtoken');

exports.postStudent = async (req, res) => {
  const data = req.body;
  const imagePath = `/studentProfiles/${req.file.filename}`;

  try {
    const hashedPassword = await bcrypt.hash(data.password, 4);

    const result = await db.execute(
      "insert into
student(regno,first_name,last_name,gender,dob,email,phone,address,blood_group,caste,aadhar_no,rel
igion,birth_place,birth_district,country,identity_mark,pincode,password,f_name,f_occupation,f_phon
e,f_email,m_name,m_occupation,m_phone,m_email,g_name,g_occupation,g_phone,g_email,dept_id,
course_id,image_path,joining_year,role,status,semester,eligibility)
values(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,(select dept_id from department where
dept_name=?),(select course_id from course where course_name=?),?,?,?,?,?,?)",
      [
        data.regno,
        data.firstName,
        data.lastName,
```

```
data.gender,
data.dob,
data.email,
data.phone,
data.address,
data.bloodGroup,
data.caste,
data.aadharNo,
data.religion,
data.birthPlace,
data.birthDistrict,
data.country,
data.identityMark,
data.pincode,
hashedPassword,
data.fatherName,
data.fatherOccupation,
data.fatherPhone,
data.fatherEmail,
data.motherName,
data.motherOccupation,
data.motherPhone,
data.motherEmail,
data.gName,
data.gOccupation,
data.gPhone,
data.gEmail,
data.department,
data.course,
imagePath,
data.joiningYear,
"student",
```

```
        "pending",
        1,
        false
      ]
    );
    res.status(200).send({ success: true });
  } catch (err) {
    res.status(500).send(err);
  }
};

exports.postFaculty = async (req, res) => {
  const data = req.body;
  const err = validationResult(req).errors;
  if (err.length > 0) {
    return res.status(400).send({ success: false, err });
  }
  const hashedPassword = await bcrypt.hash(data.password, 4);

  try {
    const result = await db.execute(
      "insert into
faculty(faculty_id,first_name,last_name,gender,dob,email,phone,address,blood_group,caste,aadhar_n
o,religion,birth_place,birth_district,country,identity_mark,pincode,password,f_name,f_occupation,f_p
hone,f_email,dept_id,teaching_exp,joining_year,role,status)
values(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,(select dept_id from department where
dept_name=?),?,?,?,?)",
      [
        data.facultyId,
        data.firstName,
        data.lastName,
        data.gender,
        data.dob,
        data.email,
```

```
        data.phone,
        data.address,
        data.bloodGroup,
        data.caste,
        data.aadharNo,
        data.religion,
        data.birthPlace,
        data.birthDistrict,
        data.country,
        data.identityMark,
        data.pincode,
        hashedPassword,
        data.fatherName,
        data.fatherOccupation,
        data.fatherPhone,
        data.fatherEmail,
        data.department,
        data.teachingExp,
        data.joiningYear,
        "faculty",
        "pending",
      ]
    );
    res.status(200).send({ success: true, data: result[0] });
  } catch (err) {
    console.log(err);
    res.status(500).send({ success: false, err });
  }
};

exports.postStaff = async (req, res) => {
  const data = req.body;
```

```
  const err = validationResult(req).errors;
  if (err.length > 0) {
    return res.status(400).send({ success: false, err });
  }
  const hashedPassword = await bcrypt.hash(data.password, 4);


  try {
    const result = await db.execute(
      "insert into
staff(first_name,last_name,gender,dob,email,phone,address,country,pincode,blood_group,caste,aadha
r_no,identity_mark,religion,birth_place,birth_district,password,f_name,f_occupation,f_phone,f_email
,joining_year,dept_id,role,status) values(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,(select dept_id from
department where dept_name=?),?,?)",
      [
        data.firstName,
        data.lastName,
        data.gender,
        data.dob,
        data.email,
        data.phone,
        data.address,
        data.country,
        data.pincode,
        data.bloodGroup,
        data.caste,
        data.aadharNo,
        data.identityMark,
        data.religion,
        data.birthPlace,
        data.birthDistrict,
        hashedPassword,
        data.fatherName,
        data.fatherOccupation,
        data.fatherPhone,
```

```
        data.fatherEmail,
        data.joiningYear,
        data.department,
        "staff",
        "pending",
      ]
    );
    res.status(200).send({ success: true, data: result[0] });
  } catch (err) {
    res.status(500).send({ success: false, err });
  }
};

exports.postLogin = async (req, res) => {
  const email = req.body.email;
  const password = req.body.password;
  const loginType = req.body.role;
  try {
    let sql;
    let id;
    if (loginType === 'student') {
      id = "regno";
      sql = `select
regno,first_name,last_name,email,password,phone,dept_id,semester,course_id,image_path,address,rol
e from ${loginType} where email='${email}' and status='approved'`;
    } else if (loginType === 'super admin') {
      id = 's_admin_id';
      sql = `select ${id},first_name,last_name,email,password,role from ${loginType.split(' ').join('_')}
where email='${email}'`;
    } else if (loginType === 'exam coord') {
      id = "coord_id";
      sql = `select ${id},first_name,last_name,email,password,phone,dept_id,address,role from
${loginType.split(' ').join('_')} where email='${email}'`;
    } else if (loginType === 'admin') {
```

```
      id = "admin_id";
      sql = `select ${id},first_name,last_name,email,password,phone,dept_id,address,role from ${loginType} where email='${email}'`;
    } else {
      if (loginType === 'faculty') {
        id = "faculty_id";
      } else if (loginType === 'staff') {
        id = "staff_id";
      }
      sql = `select ${id},first_name,last_name,email,password,phone,dept_id,address,role from ${loginType} where email='${email}' and status='approved'`;
    }
    const [user] = await db.execute(sql);
    // res.send({success:true});
    if (user.length === 0) {
      throw new Error("Invalid email or password");
    } else {
      const fetchedUser = user[0];
      bcrypt.compare(password, fetchedUser.password).then((isEqual) => {
        if (!isEqual) {
          return res.status(401).json({
            error: "Invalid Email or password",
          });
        }
        const token = jwt.sign(
          { email: fetchedUser.email, deptId: fetchedUser.dept_id, firstName: fetchedUser.first_name, lastName: fetchedUser.last_name, id: fetchedUser[id] },
          process.env.SECRET_KEY
        );
        res.status(200).json({
          token: token,
          user: {
            id: fetchedUser[id],
```

```
            first_name: fetchedUser.first_name,
            last_name: fetchedUser.last_name,
            imagePath: fetchedUser.image_path,
            email: fetchedUser.email,
            address: fetchedUser.address,
            phone: fetchedUser.phone,
            courseId: fetchedUser.course_id,
            deptId: fetchedUser.dept_id,
            role: fetchedUser.role,
            semester: fetchedUser.semester
          }
        });
      });
    }
  } catch (error) {
    res.status(404).json({ success: false, error: error.message });
  }
}
```

routeController.js

```
const db = require('../db');
const hallTicketTemplate = require('../hallticket');

exports.getDepartments = async(req,res) => {
    try {
      const result = await db.execute('select dept_id,dept_name from department');
      res.send(result[0]);
    } catch(err) {
        res.status(500).send(err);
    }
}

exports.getCourses = async(req,res) => {
```

```
    try {
      const departmentName = req.body.departmentName;
      const deptId = req.body.deptId;
      let sql;
      if(departmentName) {
        sql = `select course_name,course_id from course where dept_id=(select dept_id from department
where dept_name='${departmentName}')`;
      } else {
        sql = `select course_name,course_id from course where dept_id=${deptId}`;
      }
      const result = await db.execute(sql);
      res.send(result[0]);
    } catch(err) {
        res.status(500).send(err);
    }
}

exports.getSemesters = async(req,res) => {
  const courseName = req.body.courseName;
  try {
    const [result] = await db.execute(`select course_sem from course where
course_name='${courseName}'`);
    res.send(result[0]);
  } catch(err) {
      res.status(500).send(err);
  }
}

exports.getAllStudent = async (req,res) => {
  const deptId = req.deptId;
  const courseName = req.body.courseValue;
  const semester = req.body.semester;
  let sql;
```

```
  if(courseName&&semester) {

    sql=`select regno, first_name, last_name, course_name, joining_year, semester, eligibility from student join course on student.course_id=course.course_id where student.dept_id = ${deptId} and course_name='${courseName}' and semester=${semester} and student.status='approved' order by regno`;

  } else if(courseName) {

    sql=`select regno, first_name, last_name, course_name, joining_year, semester, eligibility from student join course on student.course_id=course.course_id where student.dept_id = ${deptId} and course_name='${courseName}' and student.status='approved' order by regno`;

  } else {

    sql=`select regno, first_name, last_name, course_name, joining_year, semester, eligibility from student join course on student.course_id=course.course_id where student.dept_id = ${deptId} and status='approved' order by regno`;

  }

  try{

    const result = await db.execute(sql);

    res.send(result[0]);

  }catch(err){

    res.status(500).send(err);

  }

}


exports.getAllFaculty = async(req,res) => {

  const deptId = req.deptId;

   let sql=`select faculty_id,email,first_name, last_name,joining_year from faculty where faculty.dept_id=${deptId} and status='approved' order by faculty_id`;

  try{

    const result = await db.execute(sql);

    res.send(result[0]);

  }catch(err){

    console.log(err)

    res.status(500).send(err);

  }

}
```

```
exports.getAllStaff = async(req,res) => {
  const deptId = req.deptId;
  let sql=`select staff_id, first_name, last_name, email, joining_year from staff where
staff.dept_id=${deptId} and status='approved'`;
  try{
    const result = await db.execute(sql);
    res.send(result[0]);
  }catch(err){
    res.status(500).send(err);
  }
}

exports.getAllExamCoord = async(req,res) => {
  const deptId = req.deptId;
  let sql=`select coord_id, first_name, last_name, email,dept_name from exam_coord join department
on exam_coord.dept_id=department.dept_id where exam_coord.dept_id=${deptId}`;
  try{
    const result = await db.execute(sql);
    res.send(result[0]);
  }catch(err){
    console.log(err);
    res.status(500).send(err);
  }
}

exports.getUserDetails = async(req,res) => {
  let type = req.body.type;
  const userId = req.body.uid;
  const idName = req.body.idName;
  if(type=="examcoordinator") type = "exam_coord";

  let sql=`select * from ${type} where ${idName}='${userId}'`;
  try{
```

```
    const result = await db.execute(sql);
    res.send(result[0]);
  }catch(err){
    console.log(err);
    res.status(500).send(err);
  }
}

exports.getSemFilteredStudent = async(req,res) => {
  const semester = req.body.semester;
  const courseName = req.body.courseName;

  let sql=`select course_name,eligibility,first_name,last_name,joining_year,regno,semester from
student join course on student.course_id=course.course_id where student.course_id=(select course_id
from course where course_name='${courseName}') and student.semester=${semester}`;
  try{
    const result = await db.execute(sql);
    res.send(result[0]);
  }catch(err){
    console.log(err);
    res.status(500).send(err);
  }
}
```

staffControler.js

```
const db = require('../db');
const Pdfmake = require('pdfmake');
const fs = require('fs');
const path = require('path');
const hallTicketTemplate = require('../hallticket');

exports.getStudentApproveList = async(req,res) => {
    const {courseName} = req.body;
    const deptId = req.deptId;
```

```
    try {
        let sql;
        if(!courseName) {
            sql = `select regno,first_name,last_name,course_name,joining_year,image_path from student
join course on student.course_id=course.course_id where student.dept_id=${deptId} and
student.status='pending' order by regno`;
        } else {
            sql = `select regno,first_name,last_name,course_name,joining_year,image_path from student
join course on student.course_id=course.course_id where student.dept_id=${deptId} and
course.course_name='${courseName}' and student.status='pending'`;
        }
       const result = await db.execute(sql);
       res.send(result[0]);
    } catch(err) {
        res.status(500).send(err);
        console.log(err);
    }
}


exports.getApproveStudentDetail = async(req,res) => {
    const id = req.params.id;
    try {
        const result = await db.execute(`select student.*,dept_name,course_name from student join
department on student.dept_id=department.dept_id join course on student.course_id=course.course_id
where regno='${id}'`);
        res.status(200).send(result[0]);
    } catch(err) {
        console.log(err);
        res.status(500).send({success:false})
    }
}


exports.postApproveStudent = async(req,res) => {
    const id = req.params.id;
```

```
    try {
      const result = await db.execute(`update student set status='approved' where regno='${id}'`);
       res.status(200).send({success:true,data:result});
    } catch(err) {
       console.log(err);
       res.status(500).send({success:false})
    }
}

exports.postRejectStudent = async(req,res) => {
    const id = req.params.id;
    const imgUrl = req.body.imageUrl;
    const imagePath = path.join(__dirname,'..','uploads',imgUrl);
    try {
       const result = await db.execute(`delete from student where regno='${id}'`);
       fs.unlink(imagePath,(err => {
          if(err) {
             console.log(err);
             throw new Error('Something went wrong');
          }
          res.status(200).send({success:true,data:result[0]});
       }));
    } catch(err) {
       console.log(err);
       res.status(500).send({success:false})
    }
}

exports.getFacultyApproveList = async(req,res) => {
    const deptId = req.deptId;
    try {
       const sql = `select faculty_id,first_name,last_name,joining_year from faculty where
dept_id=${deptId} and faculty.status='pending'`;
```

```
        const result = await db.execute(sql);
        res.send(result[0]);
    } catch(err) {
        res.status(500).send(err);
        console.log(err);
    }
}

exports.getApproveFacultyDetail = async(req,res) => {
    const id = req.params.id;
    try {
        const result = await db.execute(`select * from faculty join department on
faculty.dept_id=department.dept_id where faculty_id='${id}'`);
        res.status(200).send(result[0]);
    } catch(err) {
        console.log(err);
        res.status(500).send({success:false})
    }
}

exports.postApproveFaculty = async(req,res) => {
    const id = req.params.id;
    try {
        const result = await db.execute(`update faculty set status='approved' where faculty_id='${id}'`);
        res.status(200).send({success:true,data:result});
    } catch(err) {
        console.log(err);
        res.status(500).send({success:false})
    }
}

exports.postRejectFaculty = async(req,res) => {
    const id = req.params.id;
```

```
    try {
        const result = await db.execute(`delete from faculty where faculty_id='${id}'`);
        res.status(200).send({success:true,data:result[0]});
    } catch(err) {
        console.log(err);
        res.status(500).send({success:false})
    }
}

exports.generateBulkHallticket = async(req,res) => {
    const {semester,courseName} = req.body;

    try {
        const start = Date.now();
        const sql = `select regno,first_name,last_name,dept_name,semester,image_path from student join
department on student.dept_id=department.dept_id where student.course_id=(select course_id from
course where course_name='${courseName}') and student.semester=${semester} order by regno`;
        const [result] = await db.execute(sql);

        const timetableSql = `select subj_name,subj_code,exam_date,exam_time from timetable where
course_id=(select course_id from course where course_name='${courseName}') and
semester='${semester}' and status='approved' order by exam_date`;
        const [timetable] = await db.execute(timetableSql);

        const fonts = {
          Times: {
              normal: path.join(__dirname,'..','fonts','Times-New-Roman','times-new-roman.ttf'),
              bold: path.join(__dirname,'..','fonts','Times-New-Roman','times-new-roman-bold.ttf'),
              italics: path.join(__dirname,'..','fonts','Times-New-Roman','times-new-roman-italic.ttf'),
              bolditalics: path.join(__dirname,'..','fonts','Times-New-Roman','times-new-roman-bold-
italic.ttf'),
            },
        }
        const pdf = new Pdfmake(fonts);
```

```
        const doc = pdf.createPdfKitDocument(hallTicketTemplate(result,timetable),{});
        doc.pipe(res);
        doc.end();
        const end = Date.now();
        const total = end-start;
        console.log(total+'ms');
    } catch(err) {
        console.log(err);
    }
}

exports.setStudentEligibility = async(req,res) => {
    const {regno, eligibility} = req.body;
    console.log("YOOO",regno, eligibility);
    try {
        const sql = `update student set eligibility=${eligibility} where regno='${regno}'`;
        const result = await db.execute(sql);
        res.status(200).send({success:true,data:result[0]});
    } catch(err) {
        console.log(err);
        res.status(500).send({success:false})
    }
}
```

Student Controller.js

```
const db = require('../db');
const Pdfmake = require("pdfmake");
const hallTicketTemplate = require('../hallticket');
const fs = require('fs');
const path = require('path');
const {validationResult} = require('express-validator');

exports.getStudentSubjects = async(req,res) => {
```

```
  const {semester,courseId} = req.body;
  try {
    const sql = `select sem_id,subj_name,subj_code from semester where course_id=? and
sem_name=?`;
    const result = await db.execute(sql,[courseId,semester]);
    res.status(200).send(result[0]);
  } catch(err) {
    console.log(err);
    res.status(500).send(err);
  }
}

exports.postRegularPayment = async(req,res) => {
  const file = req.file;
  const deptId = req.deptId;
  const {bank,accno,transaction,date,semester,courseId,studentId,paymentId} = req.body;
  const recieptPath = `/reciepts/${file.filename}`;

  const err = validationResult(req).errors;
  if (err.length > 0) {
    res.status(400).send({ success: false, err });
    fs.unlink(file.path,(err)=>err&&res.status(500).send('Something went wrong'));
    return;
  }
try {
    const sql = `insert into
payment(dept_id,course_id,payment_id,regno,semester,bank_name,dop,transaction_id,acc_no,reciept
_path,exam_status,status) values(?,?,?,?,?,?,?,?,?,?,?,?)`;

    const result =
db.execute(sql,[deptId,courseId,paymentId,studentId,semester,bank,date,transaction,accno,recieptPath
,'regular','pending']);
    res.send(result[0]);
  } catch(err) {
```

```
    console.log(err)
  }
}

exports.getStudentTimetable = async(req,res) => {
    const {semester} = req.body;
    const deptId = req.deptId;
    const courseId = req.body.courseId;

    try{
        let sql = `select subj_code,subj_name,exam_date,exam_time from timetable where dept_id=?
and course_id=? and semester=? and status='approved'`;
       const result = await db.execute(sql,[deptId,courseId,semester]);
       res.send(result[0]);
    } catch(err) {
        res.status(500).send(err);
        console.log(err);
    }
}

exports.generateHallTicket = async(req,res) => {
    const deptId = req.deptId;
    const timetable = req.body.timetable;
    let stdId;
    if(req.userId) {
      stdId = req.userId;
    } else {
      stdId = req.body.regno;
    }

    const fonts = {
      Times: {
          normal: path.join(__dirname,'..','fonts','Times-New-Roman','times-new-roman.ttf'),
```

```
        bold: path.join(__dirname,'..','fonts','Times-New-Roman','times-new-roman-bold.ttf'),
        italics: path.join(__dirname,'..','fonts','Times-New-Roman','times-new-roman-italic.ttf'),
        bolditalics: path.join(__dirname,'..','fonts','Times-New-Roman','times-new-roman-bold-italic.ttf'),
      },
    }
  ;
    try {
      const start = Date.now();
      const [result] = await db.execute(`select regno,first_name,last_name,dept_name,course_name,semester,image_path from student join course on student.course_id=course.course_id join department on course.dept_id=department.dept_id where student.dept_id=? and regno=?`,[deptId,stdId]);


      const pdf = new Pdfmake(fonts);
      fs.readFile(`./pdfs/${stdId}.pdf`, 'utf8', (err, data) => {
        if (!err) {
          res.download(`./pdfs/${stdId}.pdf`);
        } else {
          const doc = pdf.createPdfKitDocument(hallTicketTemplate(result,timetable),{});
          doc.end();
          doc.pipe(res);
        }
      })
      const end = Date.now();
      console.log(end-start+'ms');
    } catch(err) {
      res.status(400).send(err);
      console.log(err);
    }
  }
```

Adminroutes.js

```
const router = require('express').Router();
const db = require('../db');
```

```
const adminController = require('../Controllers/adminController');
const {body} = require('express-validator');
const isAuth = require('../middleware/isAuth');

router.post('/new-course',isAuth,[
    body('name').notEmpty().withMessage('Enter course name'),
    body('duration').notEmpty().withMessage('Select Duration')
],adminController.postNewCourse);



router.post('/department/new-department',isAuth,[
    body("departmentName").trim().notEmpty().withMessage("Enter department name")
    .custom(async(value) => {
      const result = await db.execute(
        `SELECT dept_name FROM department WHERE dept_name='${value}'`
      );
      if (result[0].length > 0) {
        return Promise.reject("Department already exists");
      }
    }),
    body("firstName").trim().isLength({ min: 3 }).withMessage("Name must be atleast 3 characters
long"),
    body("gender").trim().isIn(["male", "female", "others"]).withMessage("Please specify Gender"),
    body("dob").trim().isDate({ format: "DD/MM/YYYY" }).withMessage("Please Enter Valid Date"),
    body("email").trim().isEmail().withMessage("Enter valid Email ID")
      .custom(async (value) => {
        const result = await db.execute(
          `SELECT email FROM admin WHERE email='${value}'`
        );
        if (result[0].length > 0) {
          return Promise.reject("E-mail already in use");
        }
      }),
```

```
    body("phone").trim().isLength({ min: 10, max: 10 }).withMessage("Enter valid phone number"),
    body("address").trim().isLength({ min: 1 }).withMessage("Address cannot be empty"),
    body("password").trim().isLength({ min: 5 }).withMessage("Password must contain atleast 5 characters")
],adminController.postNewDepartment);

router.get('/courses',isAuth,adminController.getCourses);

router.post('/course-details',isAuth,adminController.getCourseDetails);

router.get('/departments',isAuth,adminController.getDepartments);

router.get('/examcoordinators',isAuth,adminController.getExamCoordinators);

router.post('/registration/examcoordinator',isAuth,[
  body("departmentName").trim().notEmpty().withMessage("Select department"),
  body("firstName").trim().isLength({ min: 3 }).withMessage("Name must be atleast 3 characters long"),
    body("gender").trim().isIn(["male", "female", "others"]).withMessage("Please specify Gender"),
    body("dob").trim().isDate({ format: "DD/MM/YYYY" }).withMessage("Please Enter Valid Date"),
    body("email").trim().isEmail().withMessage("Enter valid Email ID")
      .custom(async (value) => {
        const result = await db.execute(
          `SELECT email FROM admin WHERE email='${value}'`
        );
        if (result[0].length > 0) {
          return Promise.reject("E-mail already in use");
        }
      }),
    body("phone").trim().isLength({ min: 10, max: 10 }).withMessage("Enter valid phone number"),
    body("address").trim().isLength({ min: 1 }).withMessage("Address cannot be empty"),
    body("password").trim().isLength({ min: 5 }).withMessage("Password must contain atleast 5 characters")
```

```
],adminController.postNewCoordinator);

router.post('/approvelist/staff',isAuth,adminController.getStaffApproveList);
router.get('/approve/staff/view/:id',isAuth,adminController.getApproveStaffDetail);
router.post('/approve/staff/:id',isAuth,adminController.postApproveStaff);
router.post('/reject/staff/:id',isAuth,adminController.postRejectStaff);

router.post('/timetable/new',isAuth,adminController.postNewTimeTable);
router.get('/timetables',isAuth,adminController.getTimetables);

module.exports = router;
```

registration_route.js

```
const router = require("express").Router();
const controllers = require("../Controllers/registrationController");
const db = require("../db");
const fs = require('fs');
const { body,check,validationResult} = require("express-validator");
const upload = require('../middleware/multer');

const validate = (req,res,next) => {
  const file = req.file;
  const err = validationResult(req).errors;
  if (err.length > 0) {
     res.status(400).send({ success: false, err });
     fs.unlink(file.path,(err)=>console.log(err));
     return;
  }
  next();
}

//Student Post Request
router.post(
```

```
  "/registration/student",
  upload.single("studentProfile"),[
    check("firstName").trim().isLength({ min: 3 }).withMessage("Name must be atleast 3 characters long"),
    check("regno").trim().isLength({ min: 5 }).withMessage("Enter valid Registration No.").custom(async (value) => {
      const result = await db.execute(
        `SELECT regno FROM student WHERE regno='${value}'`
      );
      if (result[0].length > 0) {
        return Promise.reject("This Register no. already registered");
      }
    }),
    check("gender").trim().isIn(["male", "female", "others"]).withMessage("Please specify Gender"),
    check("dob").trim().isDate({ format: "DD/MM/YYYY" }).withMessage("Please Enter Valid Date"),
    check("email").isEmail().withMessage("Enter valid Email ID").custom(async (value) => {
        const result = await db.execute(
          `SELECT email FROM student WHERE email='${value}'`
        );
        if (result[0].length > 0) {
          return Promise.reject("E-mail already in use");
        }
      }),
    check("phone").trim().isLength({ min: 10, max: 10 }).withMessage("Enter valid phone number"),
    check("address").trim().isLength({ min: 1 }).withMessage("Address cannot be empty"),
    check("bloodGroup").trim().notEmpty().withMessage("Enter a valid blood group"),
    check("caste").trim().isLength({ min: 1 }).withMessage("Caste cannot be empty"),
    check("aadharNo").trim().isLength({ min: 12, max: 12 }).withMessage("Enter valid Aadhar number"),
    check("religion").trim().notEmpty().withMessage("Enter religion"),
    check("birthPlace").trim().notEmpty().withMessage("Birth place cannot be empty"),
    check("birthDistrict").trim().notEmpty().withMessage("Birth District cannot be empty"),
    check("country").trim().notEmpty().withMessage("Country cannot be empty"),
```

```
    check("pincode").trim().isLength({ min: 6, max: 6 }).withMessage("Enter valid pincode"),

    check("password").trim().isLength({ min: 5 }).withMessage("Password must contain atleast 5 characters"),

    check("fatherName").trim().notEmpty().withMessage("Father name cannot be empty"),

    check("fatherOccupation").trim().notEmpty().withMessage("Father Occupation cannot be empty"),

    check("fatherPhone").trim().isLength({ min: 10, max: 10 }).withMessage("Enter valid Phone number"),

    check("fatherEmail").isEmail().withMessage("Enter valid Email ID"),

    check("motherName").trim().notEmpty().withMessage("Mother name cannot be empty"),

    check("motherOccupation").trim().notEmpty().withMessage("Mother Occupation cannot be empty"),

    check("motherPhone").trim().isLength({ min: 10, max: 10 }).withMessage("Enter valid Phone number"),

    check("motherEmail").isEmail().withMessage("Enter valid Email ID"),

    check("department").trim().notEmpty().withMessage("Select Department"),

    check("course").trim().notEmpty().withMessage("Select Course"),

    check("joiningYear").trim().notEmpty().withMessage("Enter Joining Year"),

  ],

  validate,

  controllers.postStudent

);


//Faculty Post Request

router.post(

  "/registration/faculty",[

    body("firstName").trim().isLength({ min: 3 }).withMessage("Name must be atleast 3 characters long"),

    body("facultyId").trim().isLength({ min: 5 }).withMessage("Enter valid Faculty ID").custom(async (value) => {

        const result = await db.execute(

          `SELECT faculty_id FROM faculty WHERE faculty_id='${value}'`

        );

        if (result[0].length > 0) {

          return Promise.reject("This FacultyID already registered");
```

```
      }
    }),
  body("gender").trim().isIn(["male", "female", "others"]).withMessage("Please specify Gender"),
  body("dob").trim().isDate({ format: "DD/MM/YYYY" }).withMessage("Please Enter Valid Date"),
  body("email").isEmail().withMessage("Enter valid Email ID").custom(async (value) => {
    const result = await db.execute(
      `SELECT email FROM faculty WHERE email='${value}'`
    );
    if (result[0].length > 0) {
      return Promise.reject("E-mail already in use");
    }
  }),
  body("phone").trim().isLength({ min: 10, max: 10 }).withMessage("Enter valid phone number"),
  body("address").trim().isLength({ min: 1 }).withMessage("Address cannot be empty"),
  body("bloodGroup").trim().notEmpty().withMessage("Enter a valid blood group"),
  body("caste").trim().isLength({ min: 1 }).withMessage("Caste cannot be empty"),
  body("aadharNo").trim().isLength({ min: 12, max: 12 }).withMessage("Enter valid Aadhar
number"),
  body("religion").trim().notEmpty().withMessage("Enter religion"),
  body("birthPlace").trim().notEmpty().withMessage("Birth place cannot be empty"),
  body("birthDistrict").trim().notEmpty().withMessage("Birth District cannot be empty"),
  body("country").trim().notEmpty().withMessage("Country cannot be empty"),
  body("pincode").trim().isLength({ min: 6, max: 6 }).withMessage("Enter valid pincode"),
  body("password").trim().isLength({ min: 5 }).withMessage("Password must contain atleast 5
characters"),
  body("fatherName").trim().notEmpty().withMessage("Father name cannot be empty"),
  body("fatherOccupation").trim().notEmpty().withMessage("Father Occupation cannot be empty"),
  body("fatherPhone").trim().isLength({ min: 10, max: 10 }).withMessage("Enter valid Phone
number"),
  body("department").trim().notEmpty().withMessage("Select Department"),
  body("teachingExp").trim().notEmpty().withMessage("Enter Teaching experience"),
  body("joiningYear").trim().notEmpty().withMessage("Enter Joining Year"),
],
```

```
  controllers.postFaculty
);

//Staff Post Request
router.post(
  "/registration/staff",[
    body("firstName").trim().isLength({ min: 3 }).withMessage("Name must be atleast 3 characters long"),
    body("gender").trim().isIn(["male", "female", "others"]).withMessage("Please specify Gender"),
    body("dob").trim().isDate({ format: "DD/MM/YYYY" }).withMessage("Please Enter Valid Date"),
    body("email").trim().isEmail().withMessage("Enter valid Email ID")
      .custom(async (value) => {
        const result = await db.execute(
          `SELECT email FROM staff WHERE email='${value}'`
        );
        if (result[0].length > 0) {
          return Promise.reject("E-mail already in use");
        }
      }),
    body("phone").trim().isLength({ min: 10, max: 10 }).withMessage("Enter valid phone number"),
    body("address").trim().isLength({ min: 1 }).withMessage("Address cannot be empty"),
    body("bloodGroup").trim().notEmpty().withMessage("Enter a valid blood group"),
    body("caste").trim().isLength({ min: 1 }).withMessage("Caste cannot be empty"),
    body("aadharNo").trim().isLength({ min: 12, max: 12 }).withMessage("Enter valid Aadhar number"),
    body("religion").trim().notEmpty().withMessage("Enter religion"),
    body("birthPlace").trim().notEmpty().withMessage("Birth place cannot be empty"),
    body("birthDistrict").trim().notEmpty().withMessage("Birth District cannot be empty"),
    body("country").trim().notEmpty().withMessage("Country cannot be empty"),
    body("pincode").trim().isLength({ min: 6, max: 6 }).withMessage("Enter valid pincode"),
    body("password").trim().isLength({ min: 5 }).withMessage("Password must contain atleast 5 characters"),
    body("fatherName").trim().notEmpty().withMessage("Father name cannot be empty"),
```

```
    body("fatherOccupation").trim().notEmpty().withMessage("Father Occupation cannot be empty"),
    body("fatherPhone").trim().isLength({ min: 10, max: 10 }).withMessage("Enter valid Phone
number"),
    body("department").trim().notEmpty().withMessage("Select Department"),
    body("joiningYear").trim().notEmpty().withMessage("Enter Joining Year"),
  ],
  controllers.postStaff
);

router.post('/login',controllers.postLogin);
module.exports = router;
```

route.js

```
const router = require('express').Router();
const routeContoller = require('../Controllers/routeController');
const isAuth = require('../middleware/isAuth');

router.get('/departments',routeContoller.getDepartments);
router.post('/courses',routeContoller.getCourses);
router.post('/semesters',routeContoller.getSemesters);

router.post('/users/student',isAuth,routeContoller.getAllStudent);
router.post('/users/faculty',isAuth,routeContoller.getAllFaculty);
router.post('/users/staff',isAuth,routeContoller.getAllStaff);
router.post('/users/examcoordinator',isAuth,routeContoller.getAllExamCoord);

router.post('/users/student/semfilter',isAuth,routeContoller.getSemFilteredStudent);

router.post('/users/details',isAuth,routeContoller.getUserDetails);

module.exports = router;
```

studentRoute

```
const router = require('express').Router();
```

```
const studentController = require('../Controllers/studentController');
const isAuth = require('../middleware/isAuth');
const {check} = require('express-validator');
const upload = require('../middleware/multer');

router.post('/application/regular',upload.single('reciept'),[
    check('bank').trim().isLength({min:3}).withMessage('Enter valid Bank name'),
    check('accno').trim().isLength({min:3}).withMessage('Enter valid accno'),
    check('transaction').trim().isLength({min:3}).withMessage('Enter valid transaction ID'),
    check('date').isDate({ format: "YYYY/MM/DD" }).withMessage('Enter valid Date')
],isAuth,studentController.postRegularPayment);

router.post('/application/subjects',isAuth,studentController.getStudentSubjects);

router.post('/timetable',isAuth,studentController.getStudentTimetable);

router.post('/hallticket',isAuth,studentController.generateHallTicket);

module.exports = router;
```

staff onlinement

```
const router = require('express').Router();
const staffController = require('../Controllers/staffController');
const isAuth = require('../middleware/isAuth');

router.post('/approvelist/student',isAuth,staffController.getStudentApproveList);
router.get('/approve/student/view/:id',isAuth,staffController.getApproveStudentDetail);
router.post('/approve/student/:id',isAuth,staffController.postApproveStudent);
router.post('/reject/student/:id',isAuth,staffController.postRejectStudent);

router.post('/approvelist/faculty',isAuth,staffController.getFacultyApproveList);
router.get('/approve/faculty/view/:id',isAuth,staffController.getApproveFacultyDetail);
router.post('/approve/faculty/:id',isAuth,staffController.postApproveFaculty);
```

```
router.post('/reject/faculty/:id',isAuth,staffController.postRejectFaculty);

router.post('/halltickets',isAuth,staffController.generateBulkHallticket);

router.post('/eligibility',isAuth,staffController.setStudentEligibility);

module.exports = router;
```

CHAPTER-7

7. TESTING

7.1 Introduction

In this software testing has been defined as the process of analysing a software item to detect the differences between existing and required conditions and to evaluate the features of the software item. Software testing is the process used to assess the quality of computer software. It involves operation of a system or application under controlled conditions and evaluating the results. The controlled conditions should include both normal and abnormal conditions. Testing should intentionally attempt to make things go wrong to determine if things happen when they should.

There are two types of software testing:

•Black box testing-internal system design is not considered in this type of testing. test are based on requirements and functionality.

•White box testing-this testing is based on knowledge of thaw internal logic of an applications code. Tests are based on coverage of code statements, branches, paths and conditions White box testing is applicable at the unit, integration and system levels of the software testing process.

7.2 Testing objectives

•Finding defects which may get created by the programmer while developing the software.

•Gaining confidence in and providing information about the level of quality

•To prevent defects

•To make sure that the end results meets the business and user requirements.

•To ensure that it satisfies the BRS that is Business Requirement Specification and SRS that is System Requirement Specification.

7.3 Testing steps

7.3.1 Unit Testing

Unit testing focuses efforts on the smallest unit of software design. This is known as module testing. The modules are tested separately. The test is carried out during programming stage itself. In this step, each module is found to be working satisfactory as regards to the expected output from the module.

7.3.2 Integration Testing

Data can be lost across an interface. One module can have an adverse effect on another, sub functions, when combined, may not be linked in desired manner in major functions. Integration testing is a systematic approach for constructing the program structure, while at the same time conducting test to uncover errors associated within the interface. The objective is to take unit tested modules and builds program structure. All the modules are combined and tested as a whole

7.3.3 Validation

At the culmination of the integration testing, Software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test begin in validation testing. Validation testing can be defined in many ways, but a simple definition is that the validation succeeds when the software functions in a manner that is expected by the customer. After validation test has been conducted, one of the three possible conditions exists.

•The function or performance characteristics confirm to specification and are accepted.

•A deviation from specification is uncovered and a deficiency lists is created.

•Proposed system under consideration has been tested by using validation test and found to be working satisfactory.

7.3.4 Output Testing

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in a specific format. The output format on the screen is found to be correct. The format was designed in the system design time according to the user needs. For the hard copy also; the output comes as per the specified requirements by the user. Hence output testing did not result in any correction for the system

7.3.5 User acceptance testing

User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for the user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes whenever required. This is done in regard to the following point:

•Input screen design.

•Output screen design.

•Online message should be guide to the user.

•Format of reports and other outputs

7.4 Test Cases

7.4.1 Registration

Sl No	Condition	Expected Result	Result
1	If any field in the form is empty	The user to enter all the fields and then proceed	Successful
2	If the name contains other than character values	The user to enter only characters and return to the same page	Successful
3	If phone number contains other than numeric value and exceeds after 10 digit	Enter a valid phone number	Successful

4	If E-mail ID is invalid	Enter a valid E-mail ID and return to same page	Successful
5	If age of student is less than 17 and greater than 25 from DOB	Enter valid DOB	Successful
6	If adhaar card number is greater than 12 and less than 12	Enter valid number	Successful
7	If registration number is greater than 10 and less than 10	Enter valid number	Successful
8	If pin code contains other than numeric values	Enter only numeric values	Successful
9	If password is not between 8 to 16 characters	Enter a password of length between 8 and 16	Successful
10	If confirm password and password does not match	Password is not matched	Successful
11	If max size of photo exceed by 4Mb and not in a jpeg, jpg and png format	Invalid image type	Successful
12	If student has entered same register number twice	This register number is already registered	Successful

7.4.2 Login

Sl No	Condition	Expected Result	Result
1	To identify admin	Valid username and password	Successful
2	If email is blank but password is entered	Error message Invalid details	Successful
3	If the email and password is incorrect	Invalid details	Successful
4	The valid email and valid password is entered	Displays home page	Successful

7.4.3 Exam Coordinator Registration

Sl No	Condition	Expected Result	Result
1	If any field in the form is empty	The user to enter all the fields and then proceed	Successful
2	If the name contains other than character values	The user to enter only characters and return to the same page	Successful
3	If phone number contains other than numeric value and exceeds after 10 digit	Enter a valid phone number	Successful
4	If E-mail ID is invalid	Enter a valid E-mail ID and return to same page	Successful
5	If password is not between 8 to 16 characters	Enter a password of length between 8 and 16	Successful
6	If confirm password and password does not match	Password is not matched	Successful

7.4.4 Department Registration

Sl No	Condition	Expected Result	Result
1	If any field in the form is empty	The user to enter all the fields and then proceed	Successful
2	If the department name contains other than character values	The user to enter only characters and return to the same page	Successful
3	If same department is created twice	This department is already created	Successful
4	If department clicked the current date in DOB	Enter a valid DOB	Successful
5	If password is not between 8 to 16 characters	Enter a password of length between 8 and 16	Successful
6	If confirm password and password does not match	Password is not matched	Successful

7.4.5 Creating New Course

Sl No	Condition	Expected Result	Result
1	If the Course name contains other than character values	The admin is to enter only characters and return to the same page	Successful
2	If the subject name contains other than character values	The admin is to enter only characters and return to the same page	Successful
3	If the subject code contains other than character or numeric values	The admin is to enter only characters or numeric and return to the same page	Successful
4	If admin enters same subject name and subject code	Subject name and subject code is already entered	Successful
5	If any field in the form is empty	The user to enter all the fields and then proceed	Successful

7.4.6 Upload Time Table

Sl No	Condition	Expected Result	Result
1	If the subject name contains other than character values	The admin is to enter only characters and return to the same page	Successful
2	If admin entered the subject which is not created in course	Entered valid subject name	Successful
3	If the subject code contains other than character or numeric values	The admin is to enter only characters or numeric and return to the same page	Successful
4	If admin entered date is not in a correct format	Entered the date in format dd-mm-yyyy	Successful
5	If admin entered time is not in a correct format	Enter the time in hh-mm-ss format	Successful
6	If admin is entering the same subject name	This subject is already entered	Successful

7	If any field in the form is empty	The user to enter all the fields and then proceed	Successful

7.4.7 Coding sheet

Sl No	Condition	Expected Result	Result
1	If the coding sheet no. contains other than character or numeric values	Entered a valid coding sheet no.	Successful
2	If the bundle no. contains other than character or numeric values	Entered a valid bundle no.	Successful
3	If any field in the form is empty	The user to enter all the fields and then proceed	Successful

7.4.8 Evaluators

Sl No	Condition	Expected Result	Result
1	If any field in the form is empty	The user to enter all the fields and then proceed	Successful
2	If exam coordinator selects the same evaluator order for different faculty	This number is already taken	Successful
3	If same bundle no. is assign to two different faculty	This bundle number is already assigned	Successful

7.4.9 Indent

Sl No	Condition	Expected Result	Result
1	If total no of students contains other than numeric values	The staff is to enter only numeric and return to the same page	Successful
2	If the question paper contains other than numeric values	The staff is to enter only numeric and return to the same page	Successful
3	If total no of students is equal to total no of question paper	Total no of papers must be greater than tot. no of students	Successful
4	If any field in the form is empty	The staff I used to enter all the fields and then proceed	Successful

7.4.10 Attendance Statement

Sl No	Condition	Expected Result	Result
1	If the subject name contains other than character values	The staff is to enter only characters and return to the same page	Successful
2	If staff entered the subject which is not created in course	Entered valid subject name	Successful

3	If the subject code contains other than character or numeric values	The staff is to enter only characters or numeric and return to the same page	Successful
4	If total no of students is greater than total no of absenties	Enter a valid absenties no	Successful
5	If total no of students contains other than numeric values	The staff is to enter only numeric value	Successful
6	If any field in the form is empty	The staff is used to enter all the fields and then proceed	Successful

7.4.11 Attendance

Sl No	Condition	Expected Result	Result
1	If the attendance is entered other than numeric values	The faculty is used to enter only numeric and return to the same page	Successful
2	If any field in the form is empty	The faculty to enter all the fields and then proceed	Successful
3	If attendance entered is greater than hundred	Entered valid attendance	Successful

7.4.12 Internal Marks

Sl No	Condition	Expected Result	Result
1	If the internal mark is entered other than numeric values	The faculty is used to enter only numeric and return to the same page	Successful
2	If any field in the form is empty	The faculty to enter all the fields and then proceed	Successful

7.4.13 Semester Marks

Sl No	Condition	Expected Result	Result
1	If the semester mark is entered other than numeric values	The faculty is used to enter only numeric and return to the same page	Successful
2	If any field in the form is empty	The faculty to enter all the fields and then proceed	Successful

7.4.14 Payment

Sl No	Condition	Expected Result	Result
1	If the bank name is entered other than character values	Only characters are used	Successful
2	If the account no is entered other than numeric values	Only numeric value is entered	Successful

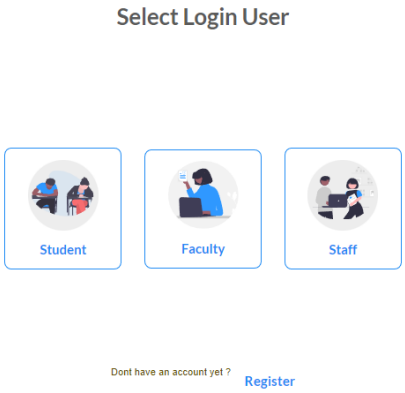
3	If any field in the form is empty	The faculty to enter all the fields and then proceed	Successful
4	If date is not entered in correct format	Enter the date in correct format	Successful

CHAPTER-8

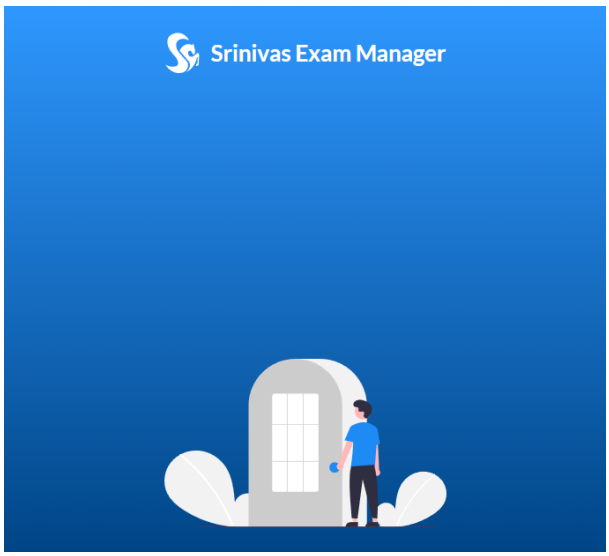
8. USER INTERFACE

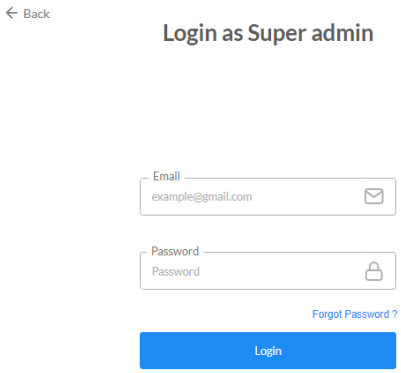
8.1 Screenshort

Main Page

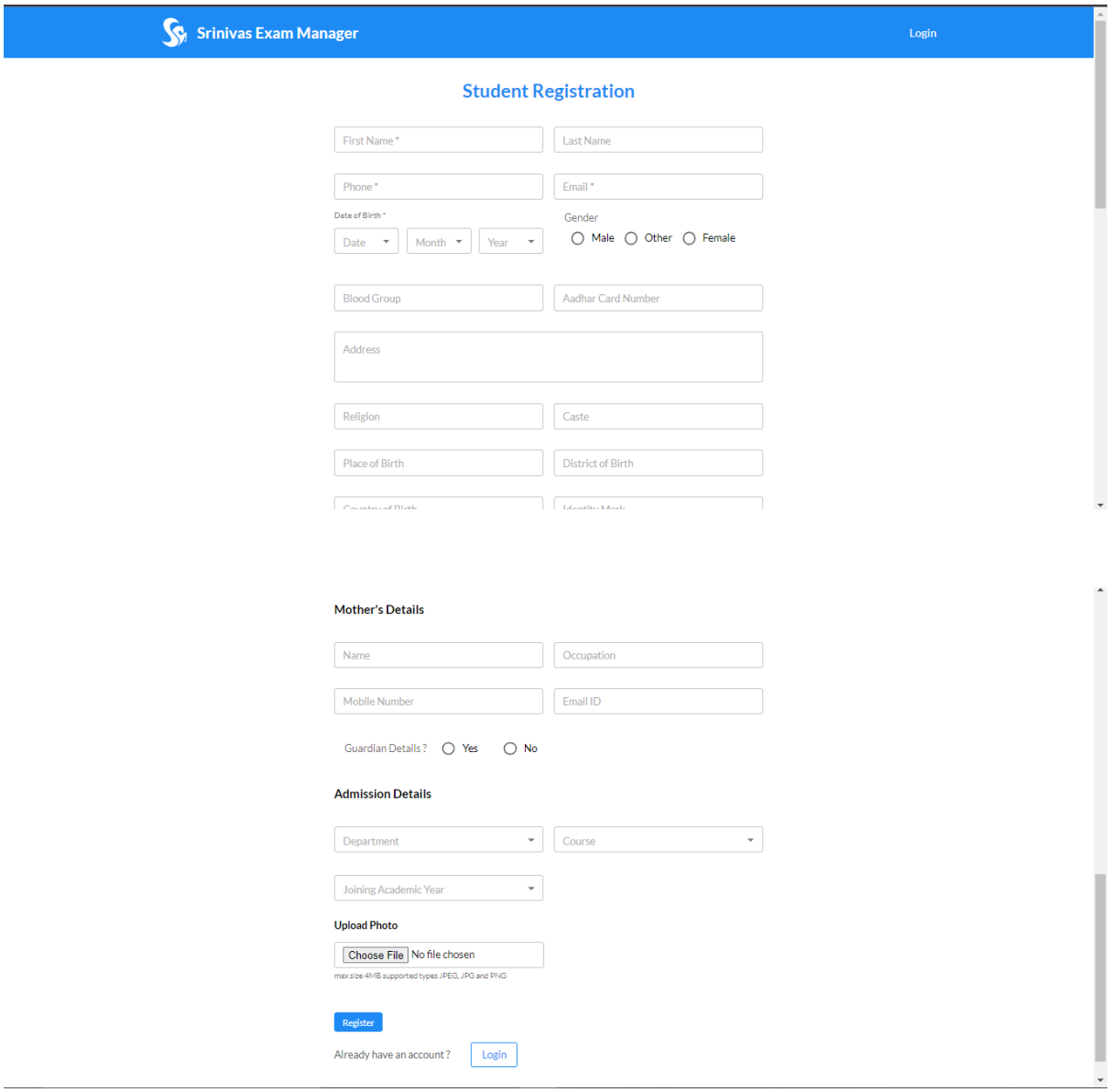


Login Page



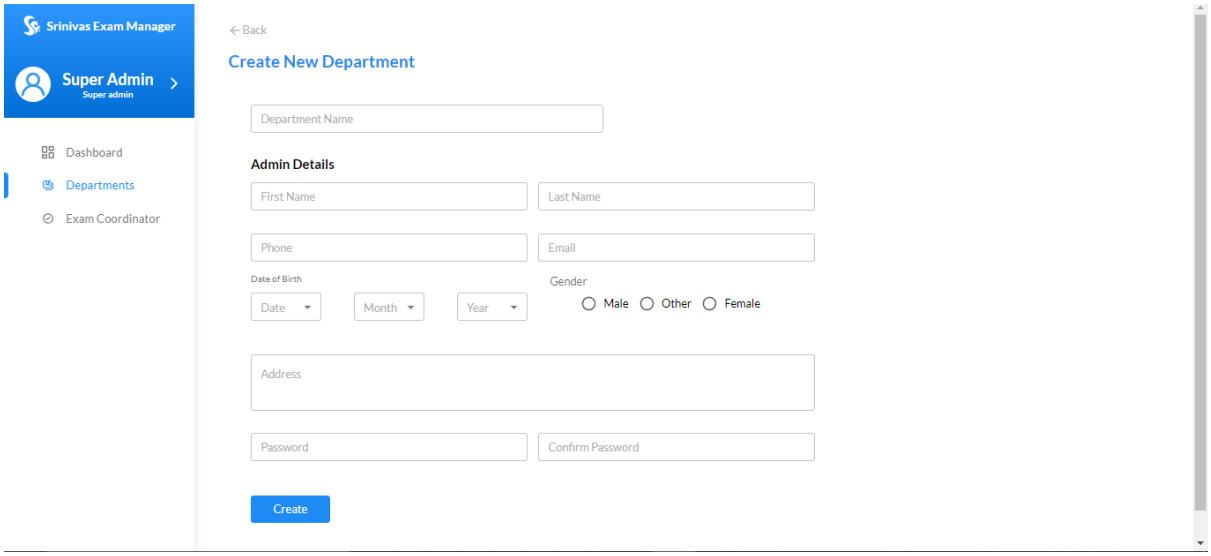


Registration

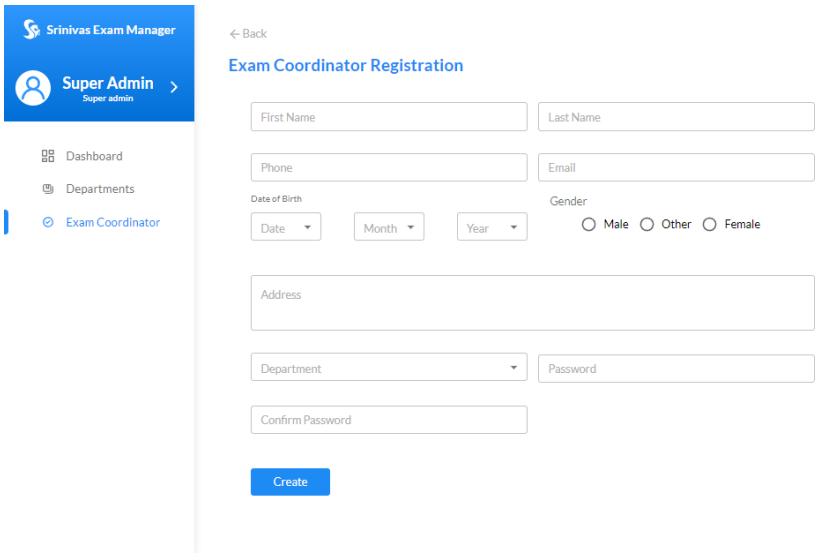


Dashboard Page

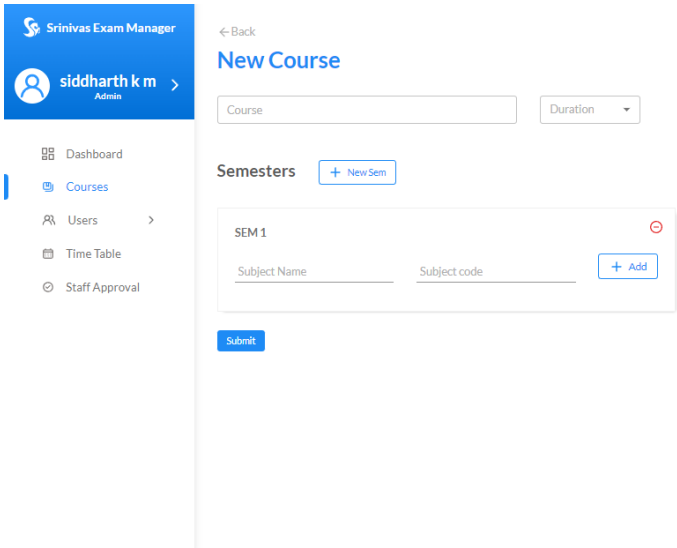
Create Department



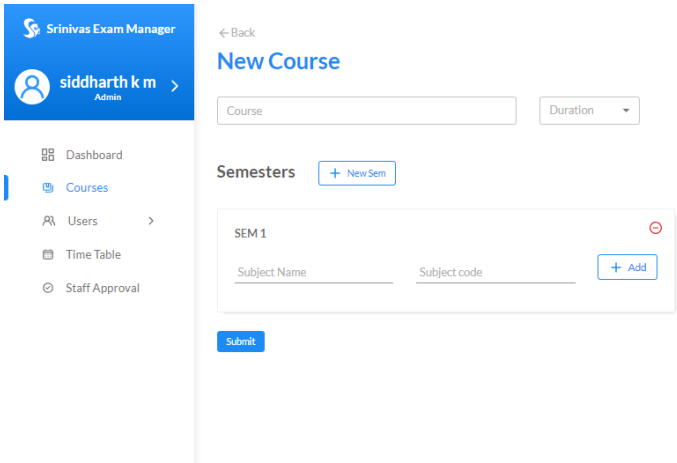
Exam coordinator



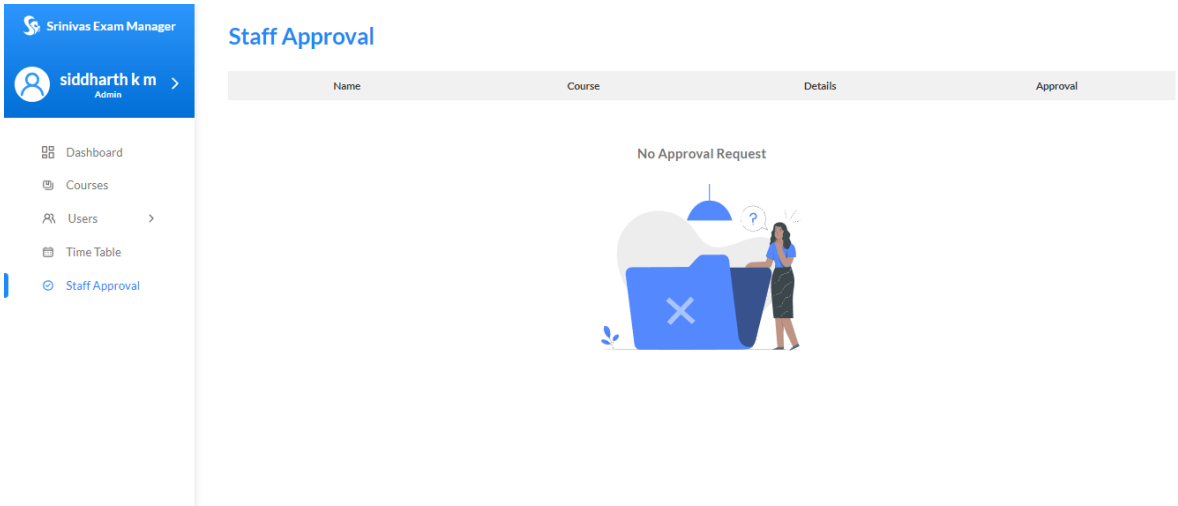
Create course



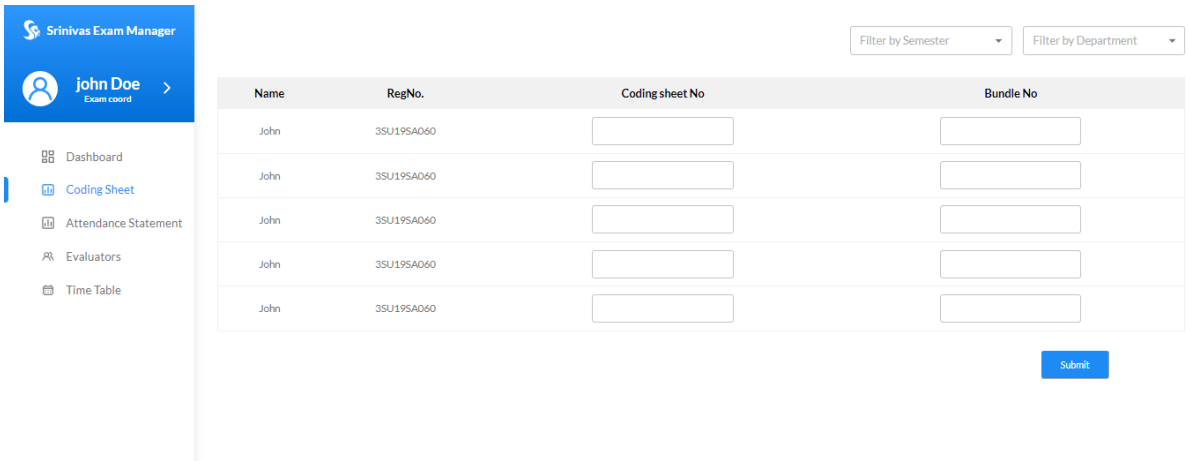
Upload Time Table



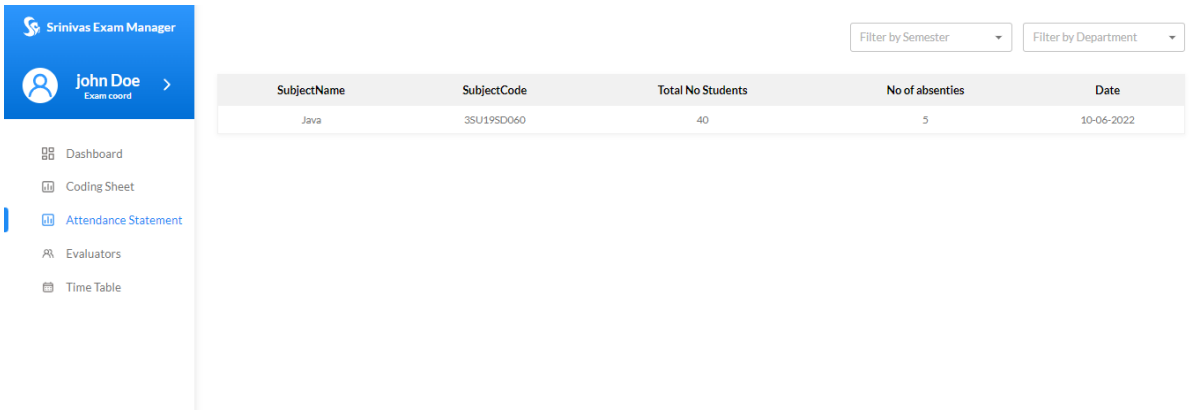
Staff Approval



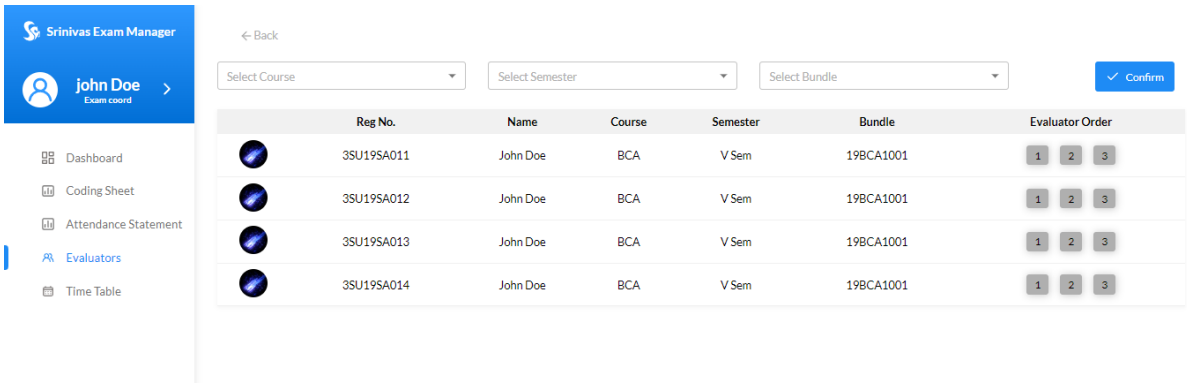
Coding Sheet



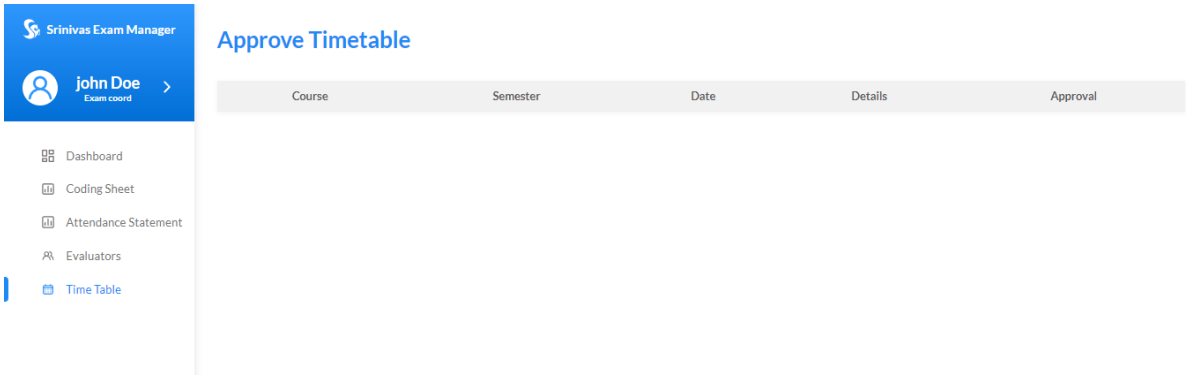
Attendance Statement



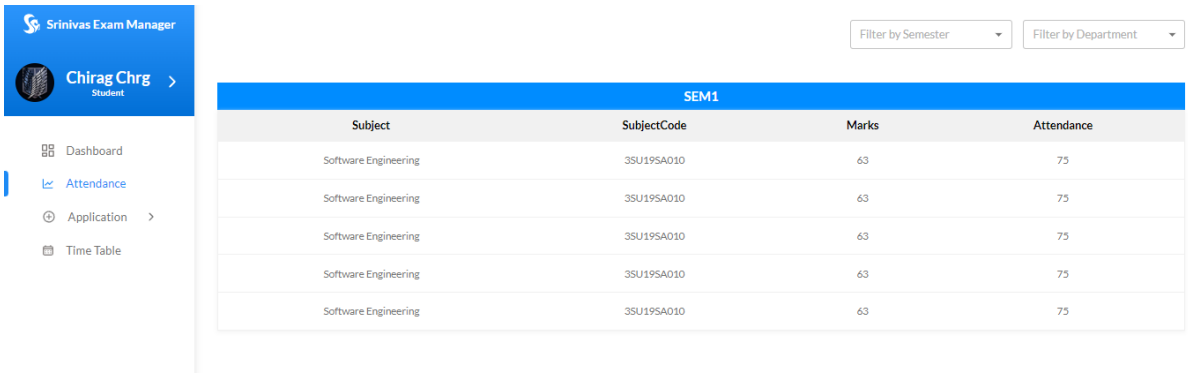
Evaluators



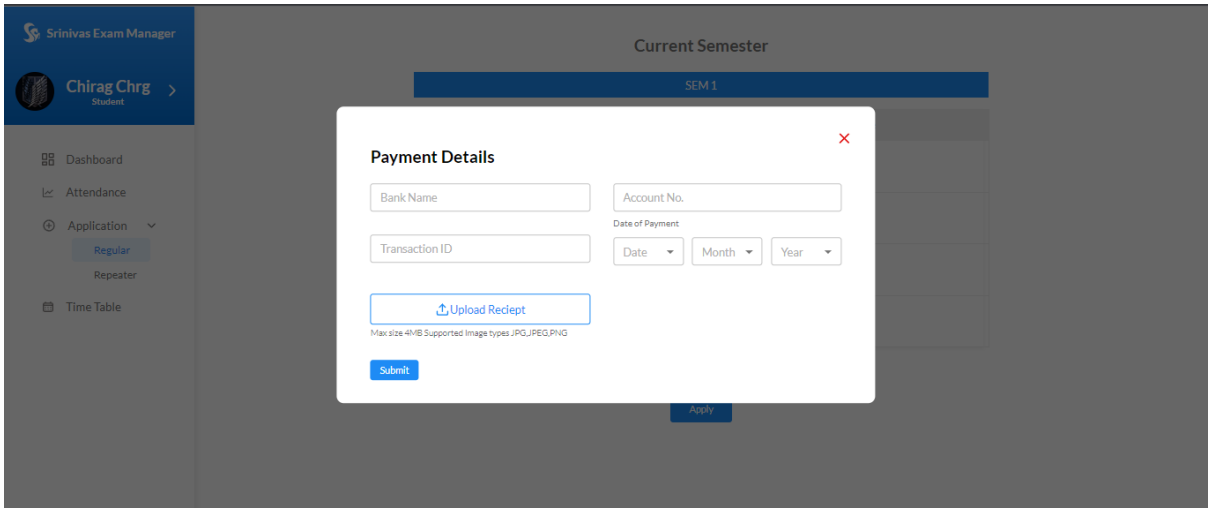
Approve Time Table



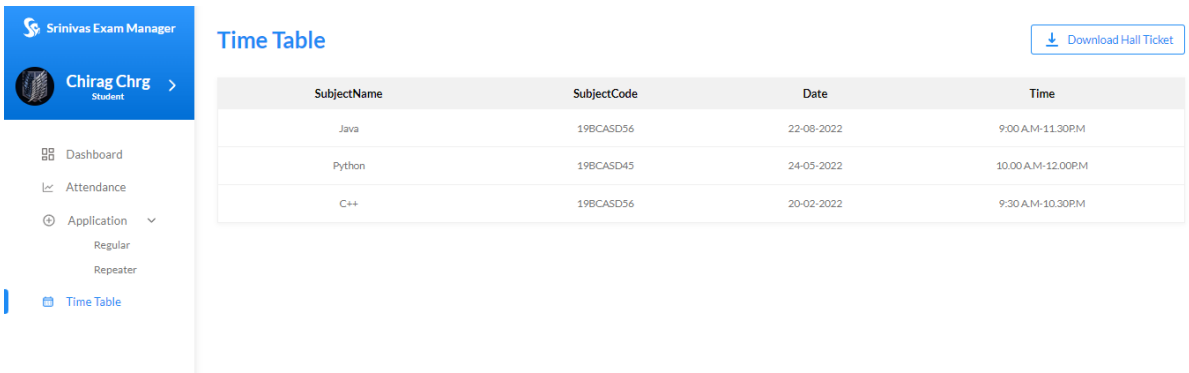
Student Attendance



Payment Application

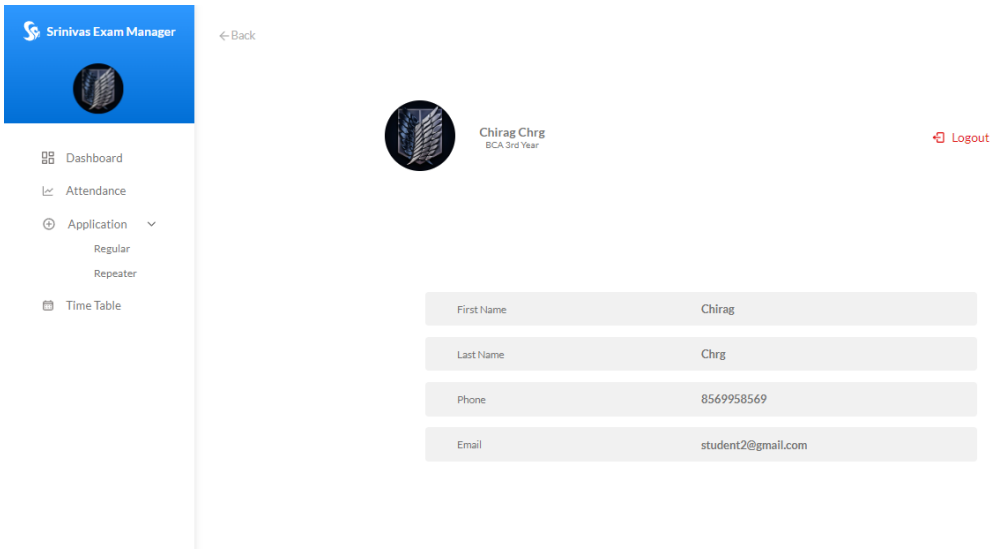


Time Table and Hall ticket



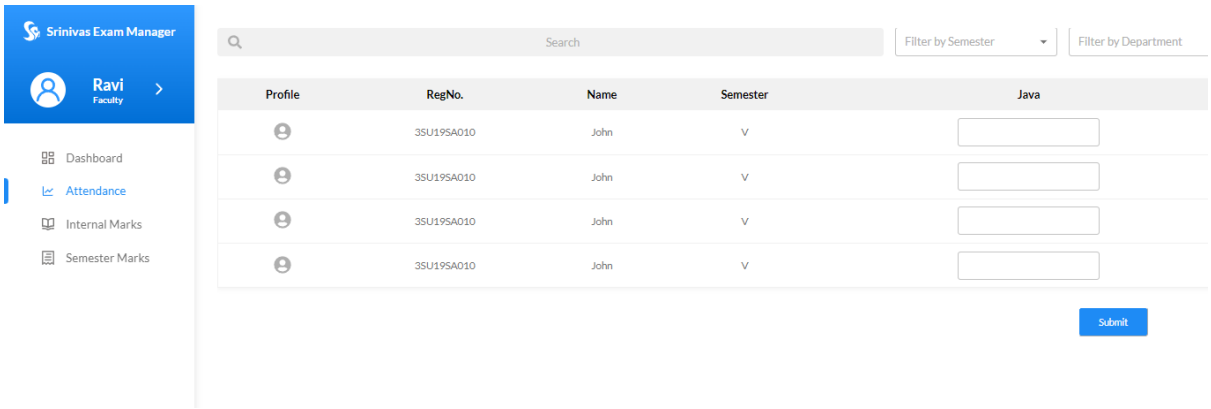
SubjectName	SubjectCode	Date	Time
Java	19BCASD56	22-08-2022	9:00 A.M-11.30P.M
Python	19BCASD45	24-05-2022	10.00 A.M-12.00P.M
C++	19BCASD56	20-02-2022	9:30 A.M-10.30P.M

Profile page

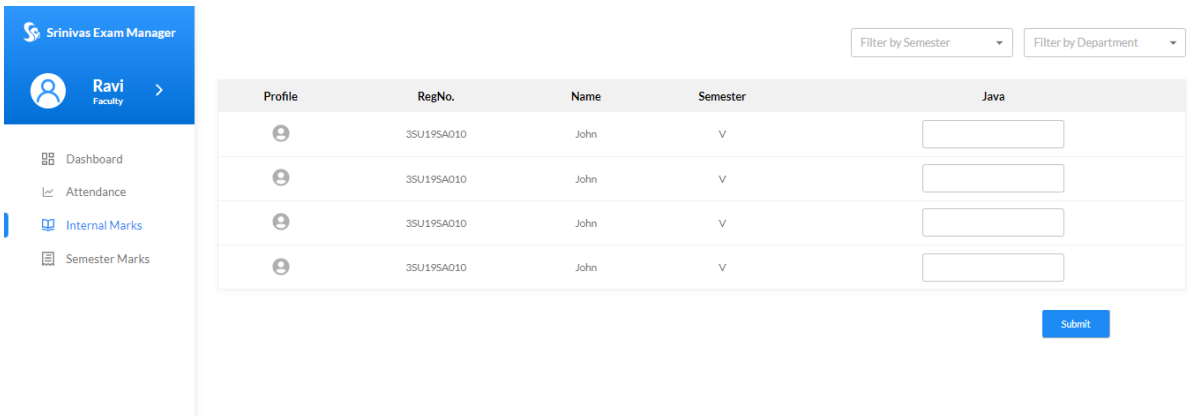


First Name	Chirag
Last Name	Chrg
Phone	8569958569
Email	student2@gmail.com

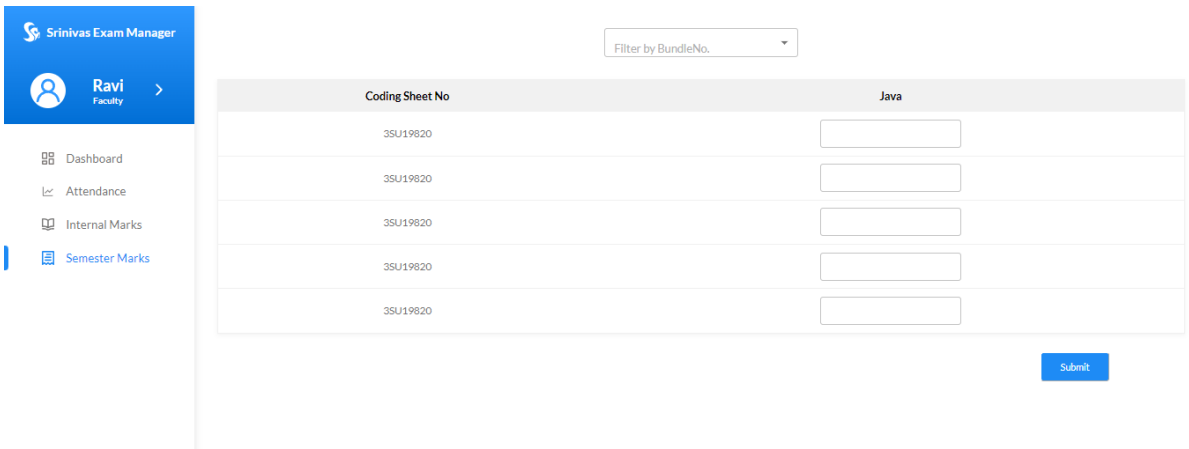
Attendance assigning



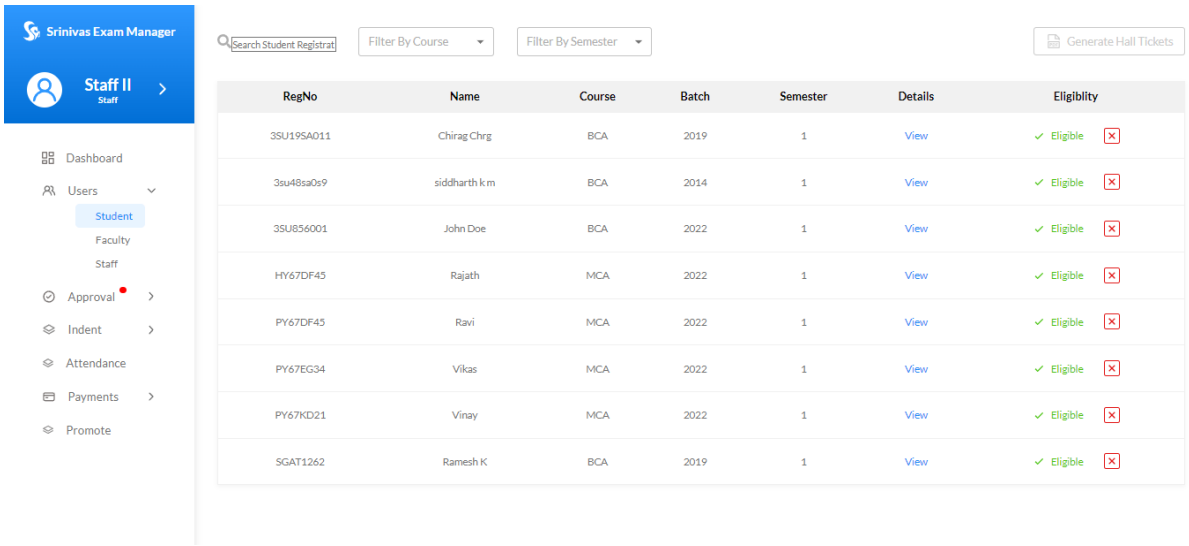
Internal Marks



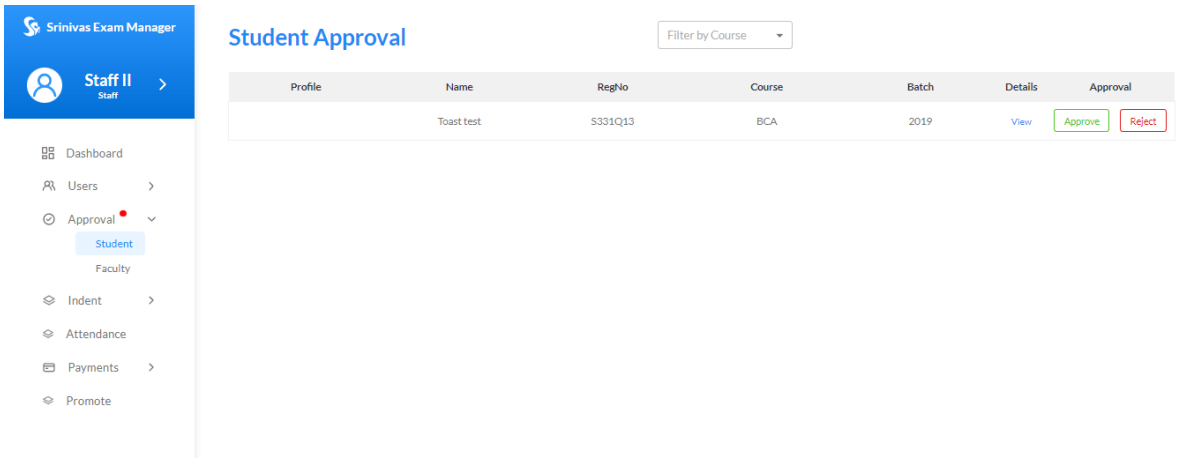
Semester Marks



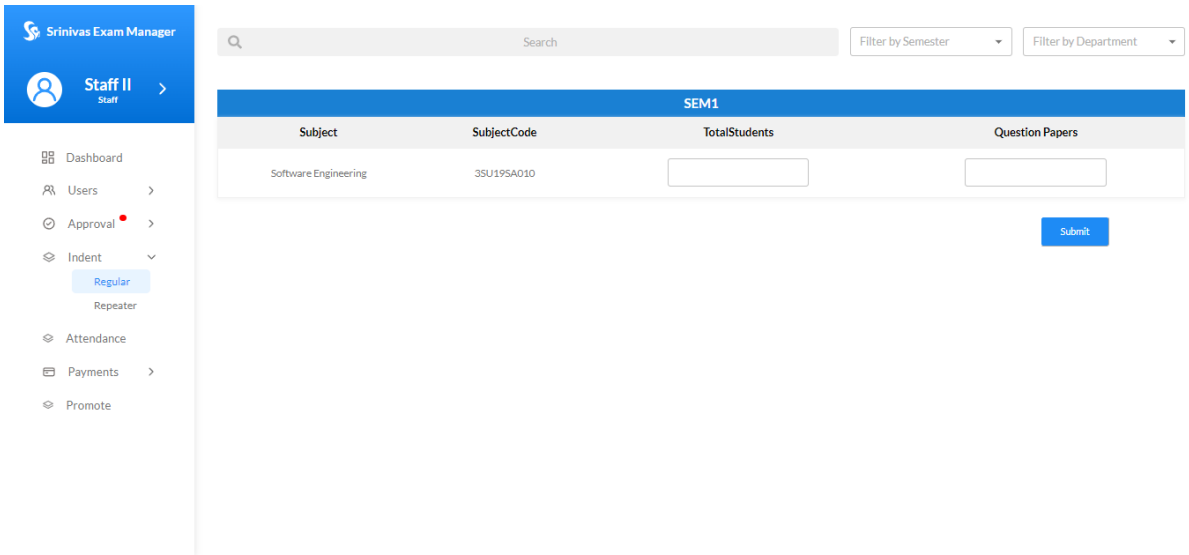
Users



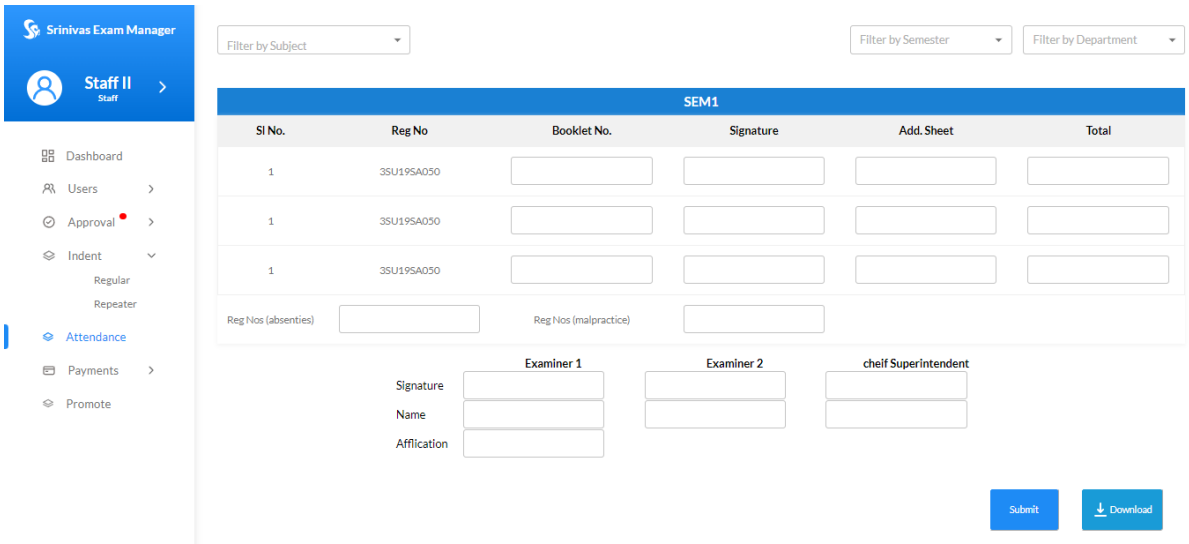
Approval Page



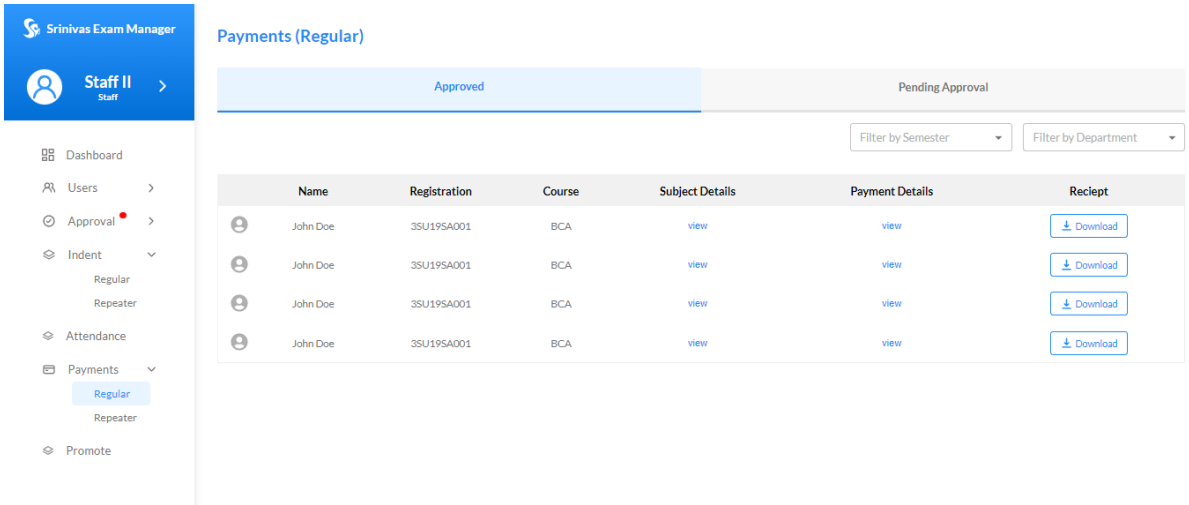
Indent Page



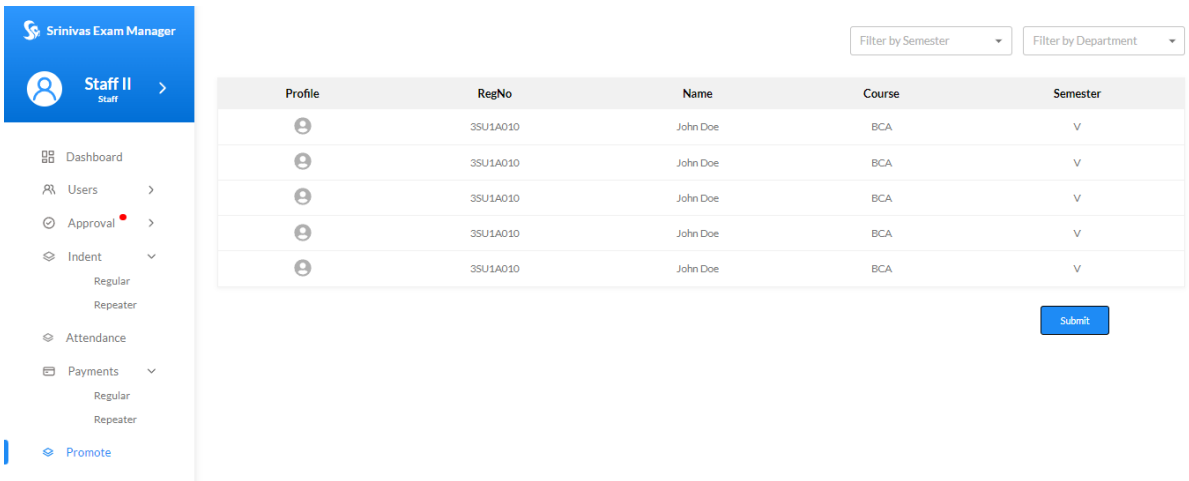
Attendance List Page



Payment Approval



Promote Page



CHAPTER-9

9. USER MANUAL

9.1 Introduction

Srinivas Exam Manager is a simple software where mainly it is used by students, faculty and office staffs where their manual work is fully automated. Here the students can register with their details and they can log in to the system where the marks and attendance details are displayed. Here the software is built in such a way that it is simple and user friendly. These software can be used in different college where they can reduce the office works. In todays world everything is click away that's the parole of these modern times we live. Consequently, user guides are significant aspect for every IT developer, and much-needed addition to their customer services

9.2 Hardware Requirements:

- Intel dual core Processor or higher
- 2 GB RAM and above
- 40 GB hard disk
- Mouse/Keyboard

9.3 Software Requirements:

- Language: Javascript
- User interface design: Html, CSS, Bootstrap
- Scripting language: JavaScript
- Operating system: Windows
- Sublime Text editor
- Database (Back End): MySQL
- Apache server
- Browsers**:** Firefox, Chrome or any other browser

CHAPTER-10

10. CONCLUSION

The Srinivas Exam Manager plays an important role in the automation of existing manual system which is going on in different universities. These can be implemented in schools and colleges. Its UI is built in such a way that it is user friendly and everyone can use it easily. The project is designed to meet the requirements of different users in the university. It has been developed using React, Mysql keeping in mind the specification of the system. The software takes care of all requirements and provides easy and effective storage of information related to users. It is a great improvement over the manual system and speed up the process.

Future Scope

- The project has an enormous scope in the upcoming days when the institutions can make their work automatic rather than manually.
- This system can be used on any platform because it is basically designed in platform independent language.
- Flexible to enhance the system
- It can be integrated with mobile apps.
- Additional enhanced features to be added over time

CHAPTER-11

11. BIBLIOGRAPHY

11.1 Book References:

1. Feller, J., & Fitzgerald, B. (2002). *Understanding open source software development*. Addison-Wesley Longman Publishing Co., Inc…
2. Boehm, B. W. (1984). Software engineering economics. *IEEE transactions on Software Engineering*, (1), 4-21.
3. McHugh, J., Abiteboul, S., Goldman, R., Quass, D., & Widom, J. (1997). Lore: A database management system for semistructured data. *ACM Sigmod Record*, *26*(3), 54-66.
4. Wiegers, K., & Beatty, J. (2013). *Software requirements*. Pearson Education.
5. Aggarwal, S. (2018). Modern web development using ReactJS. *International Journal of Recent Research Aspects*, *5*(1), 133-137.
6. Laksono, D. (2018, August). Testing spatial data deliverance in SQL and NoSQL database using NodeJS full stack web app. In *2018 4th International Conference on Science and Technology (ICST)* (pp. 1-5). IEEE.
7. Stonebraker, M. (2010). SQL databases v. NoSQL databases. *Communications of the ACM*, *53*(4), 10-11.
8. Schifreen, R. (2009). How to Create Websites and Applications with HTML, CSS, JavaScript, PHP, and MySQL.
9. Hartog, P., & Rhodes, E. C. (1936). Examination of examinations.
10. Fairburn, C. G., Cooper, Z., & O'Connor, M. (1993). The eating disorder examination. *International Journal of Eating Disorders*, *6*, 1-8.

11.2 Web References:

https://www.w3schools.com/js/js_loop_for.asp as on 24th March 2022

https://en.wikipedia.org/wiki/React_(JavaScript_library)#Components as on 25th March 2022

https://www.technipages.com/how-to-identify-specific-color-on-an-app-or-website as on 15 April 2022

https://reactjs.org/ as on 26th April 2022

https://reactjs.org/docs/hooks-intro.html as on 14th May 2022

https://dribbble.com/tags/webdesign as on 23th May 2022

https://wdexplorer.com/20-examples-beautiful-css-typography-design/ as on 4th June 2022

CHAPTER-12

PLAGIARISM

DrillBit

The Report is Generated by DrillBit Plagiarism Detection Software

Submission Information

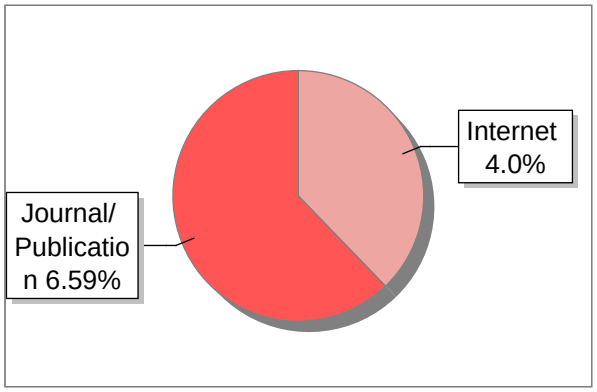
Author Name	Siddharth KM
Title	Srinivas Exam Manager
Paper/Submission ID	545613
Submission Date	2022-06-22 13:23:48
Total Pages	27
Document type	Project Work

Result Information

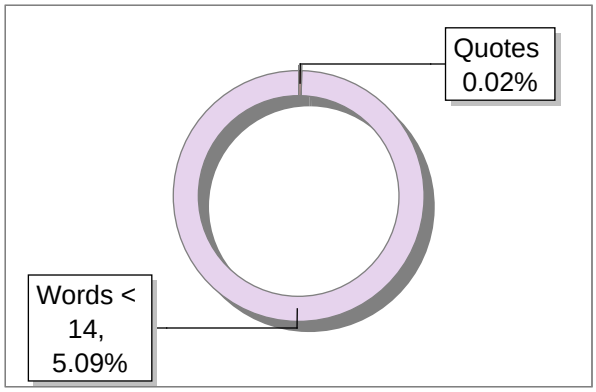
Similarity **8 %**

1 10 20 30 40 50 60 70 80 90

Sources Type



Report Content



Exclude Information

Quotes	Not Excluded
References/Bibliography	Excluded
Sources: Less than 14 Words Similarity	Excluded
Excluded Source	**0 %**
Excluded Phrases	Not Excluded

A Unique QR Code use to View/Download/Share Pdf File

DrillBit

DrillBit Similarity Report

8	9	A	A-Satisfactory (0-10%) B-Upgrade (11-40%) C-Poor (41-60%) D-Unacceptable (61-100%)
SIMILARITY %	MATCHED SOURCES	GRADE	

LOCATION	MATCHED DOMAIN	%	SOURCE TYPE
1	citeseerx.ist.psu.edu	1	Internet Data
3	ijsret.com	1	Publication
4	qdoc.tips	1	Internet Data
5	qdoc.tips	1	Internet Data
6	pdfcoffee.com	1	Internet Data
7	www.aalimec.ac.in	1	Publication
8	Paper Published in International Journal of Science and Research (IJSR) www.ijsr.net	1	Publication
9	Preparation of Foreign Language Teachers Fact vs Fiction A Reply to Concentr by Edwar-1945	<1	Publication
10	Answering XML Queries Using Path-Based Indexes A Survey by Kam-Fa-2006	<1	Publication