Daffodil International University

PROJECT TITLE

SCHOOL MANAGEMENT SYSTEM (SMS)

Supervised by:

Md. Rafiqul Huq Rafi

Lecturer at Daffodil International University

Submitted by:

Name: Faisal Ahmed Anik

ID: 181-16-277

Submission Date:

23 June, 2020

Department of Computing and Information System (CIS),

DAFFODIL INTERNATIONAL UNIVERSITY

APPROVAL

This Project titled **"**School Management System (SMS)**"**, Submitted by Faisal Ahmed Anik ID No 181-16-277 to the Department of Computing & Information Systems, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Computing & Information Systems and approved as to its style and contents. The presentation has been held on 19-07-2020.

BOARD OF EXAMINERS

_____

Mr. Md Sarwar Hossain Mollah **Chairman**

Assistant Professor and Head

Department of Computing & Information Systems

Faculty of Science & Information Technology

Daffodil International University

_____

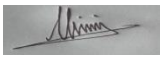
Ms. Nayeema Rahman **Internal Examiner**

Sr. Lecturer

Department of Computing & Information Systems

Faculty of Science & Information Technology

Daffodil International University



Mr. Minhaj Hosen **Internal Examiner**

Lecturer

Department of Computing & Information Systems

Faculty of Science & Information Technology

Daffodil International University

Dr. Saifuddin Md. Tareeq **External Examiner**

Professor

Department of Computer Science and Engineering

Dhaka University, Dhaka

Executive Summary

My whole project is for Schools. The system is going to make school management easier and works will be faster than ever. I will make multiple portals for users so that data can be shown specifically. I will make live attendance system so that attendance become faster and easier. I will provide multiple grading system so that schools can easily create their own grading system. I will create complain box so that teacher can create complain and that will directly ping in parent portal. I will create result mechanism so that result can be automated and easier to get. I will create ID card system so that Id card can be available easily. The system will be very user friendly. I will use current technology so that the performance of the system remains faster and secure.

Acknowledgement

I would like to specify my gratitude to my consultant Md. Rafiqul Huq Rafi sir for his guidance, support and his non-stop enthusiasm and encouragement at some point of the project. I am additionally very grateful and enlarge my sincere thanks to the parent and friends of the department of CIS at Daffodil International University for his or her cooperation by way of sharing the weight that was teaching to make me have time to work on this project and in the course of my study. Finally, many thanks to friends, who've helped and given me suggestions, helps and corrections throughout the project.

Table of Contents

Introduction 1

1.1. The system developed 1

1.2. Justification for the method or framework used 2

1.3. The solution that emerged 2

1.4. The main aims and objectives of the project 3

1.5. A short overview of the remaining chapters 3

Initial Study 4

2.1. Project Proposal 4

Initial Conception 4

2.1 Background of the project 10

2.2. Problem Area 10

2.3. Possible solution 11

Literature Review 12

3.1 Discussion on problem domain based on published articles 12

- Affordability 12
- User requirements 13
- Internet Connectivity 13
- Training 13
- Monitoring 13
- Affordable price 14
- Meet requirements 14

- Ensure Internet 14
- Train users 15
- Monitor the system 15
- SWOT-School management system 16
- Best Features: 16
- Limitations: 16
- ORA-School suit 17
- Best Features: 17
- Limitations: 17
- The Next Gen-School Management system 18
- Best Features: 18
- Limitations: 19
- Admin Panel 19
- Teacher Panel 19
- Student Panel 19

Methodology 20

- What to use 20
- Why to use 20
- Sections of methodology 21
  - Focus on the business need 21
  - Deliver on time 22

- Collaborate 22
- Never compromise quality 22
- Develop iteratively 22
- Communicate continuously and clearly 22

- Implementation plans 23
  - Feasibility Study 23
  - Business Study 24
  - Functional Model Iteration 24
  - Design and Build Iteration 24
  - Implementation 24

Planning 25

- Project Plan 25
  - Work Breakdown Structure (WBS) 25
  - Gantt Chart 26
  - Test Plan 26

Required tests 26

Test Case 26

User acceptance test plan 27

Feasibility 27

- All possible type of feasibility 27

Economic Feasibility 27

Technical Feasibility 28

Operational Feasibility 29
- Cost Benefit Analysis 29
- DSDM 30

Foundation 31
- Overall Requirement List 31

List of functional requirements 31

List of system-wide non-functional requirements 32
- What Technology to be implemented 32
- Recommendations and Justifications 33

Exploration 34
- Old Full System Use Case 34
- Old Full System Activity Diagram 35
- Prototype of new system 35

Engineering 37
- New System Modules 37
- Use Case 38
- Class Diagram 38
- ERD Diagram 40
- Sequence Diagram 41
- New System Modules 41
- System Interface Design 41

Deployment / Development 44

- Core Module Coding Samples 44
  - Possible problem breakdown 52

Testing 53

Test Case 53

- Unit Testing 53
- Module Testing 56
- Integration Testing 59

Implementation 63

Critical Appraisal and Evaluation 65

Conclusion 67

References 70

Test Scripts 71

User Guide 71

System Code 76

Plagiarism Report 83

Table of Figures:

Figure 1: Prototype 6
Figure 2:SWOT-School management system 16
Figure 3:ORA-School Suit 17
Figure 4:The Next Gen-School Management system 18
Figure 5:DSDM Life cycle 23
Figure 6:Gantt Chart for SMS 26
Figure 7:Old system use case 34
Figure 8:Old system activity diagram 35
Figure 9:Login Panel Prototype 36
Figure 10:Main Dashboard Prototype 36
Figure 11:Use case of School Management System 38
Figure 12:Initial Class Diagram 39
Figure 13:ERD diagram of School Management System 40
Figure 14:Sequence diagram 41
Figure 15:Login Panel 42
Figure 16:Admin Panel 42
Figure 17:Teacher panel 43
Figure 18:Student & Parent panel 43
Figure 19:Login Module Code 44
Figure 20:User CRUD Module code 46
Figure 21:Admission & Sibling management Module 48
Figure 22:Live attendance Module 49
Figure 23:Class schedule Module code 50
Figure 24:Grading Module Code 51
Figure 25:Result Module Code 52
Figure 26:Class Module code 53

Figure 27:Given Class details in fields 54
Figure 28:Class module working properly 54
Figure 29:Admission code 55
Figure 30:Admission form filled 55
Figure 31:Admission Working properly 56
Figure 32:Blank Login attempt 57
Figure 33:Fill login form with correct data 58
Figure 34:Login Successful 59
Figure 35:Login test with correct data 60
Figure 36:Login successful with correct data 61
Figure 37:Login failed with incorrect data 61
Figure 38:User create test with data 62
Figure 39:User created successfully with data 62
Figure 40:User creation failed with incorrect data 63
Figure 41:System link generation code for localhost 72
Figure 42:Successfully generated code 73
Figure 43:Admin Panel guide 73
Figure 44:Teacher panel Guide 74
Figure 45:Student & Parent panel Guide 75
Figure 46:Database connection code 76
Figure 47:Login Code 77
Figure 48:Admission Code 78
Figure 49:Attendance code 78
Figure 50:Routine code 79
Figure 51:Complain box code 79
Figure 52:Grading code 80
Figure 53:Profile code 80

Figure 54:Result code 81
Figure 55:Section code 81
Figure 56:User manage code 82
Figure 57:Sidebar code 82

Chapter 1

Introduction

1.1. The system developed

I developed a system which is called “School Management System”. This is a project which is for schools. In our country maximum of schools are not digitalized. The schools are still maintaining old method of keeping data manually. That’s why I tried to figure out what need to be done for making the school management system which will be helpful for all the related users of any school. I studied on field and also with the help of internet I found out many features which need to be implemented to build this project. To make the schools digital and advanced this system has so many features for all the users who are related to this system.

For Students, they can view their subject's grades, contact with the teachers, attendance report or present percentage, and they also up to date with all school's news or posts that publish by the other users.

For Admin, they have a full control on the system, like they can add a new, teachers and students with their subjects, create sections, courses, assign subjects to the teachers, send email anyone regarding notices.

For Teachers, they can add student’s grades or edit it for their own subjects only, and they can take attendance, see attendance reports, upload and download assignments of the related students.

1.2. Justification for the method or framework used

To develop my school management system, I thought about security, usability, comfort, efficiency a lot. That's why choosing programing method is very important thing. To make the project look good, simple and easy to use I chose html, css3, JavaScript. I made the site simple and responsive.

To power the backend and do the logics I used PHP procedural format and MySQL to handle the database. PHP is highly secure and fast as a backend data management method. The MySQL is also very secure to handle. Generally frameworks are open source and free. So that they may have some loopholes to hack them. But with procedural PHP I have full freedom of handling data as I want. That's why it becomes very secured as well as fast because big libraries don't need to be loaded every time. PHP is a cross platform language. Easy to use and stable for any web platform. This enables a lot of control to the developer to work with.

MySQL provides data security, on demand scalability, high performance and comprehensive transactional support. That's why I used the combination of all these methods in my project.

1.3. The solution that emerged

The school management system is created to make schools digitalized and advanced in terms of managing school. This system will bring a lot of comfort and features to the management of schools in our country. This will connect all the users in one portal. This will give secure data access and management. This will allow teachers to add assignments and manage those using the system. This will also allow students to see the reports and summery of their own data. It will also help improve students to improve their

attendance in all classes. This will allow admins send anyone emails from their portal. This will make notice sending easier to them. Overall this system will bring huge improvement to the existing manual management to school all over the country.

1.4. The main aims and objectives of the project

The main goal of making this project is to make the school management easier and faster. The school system has some users and work lists. My aim was to make sure all aspects are touched and made correctly inside the system. My system can manage maximum administration works, teacher and student works. It will stablish connection between student and teacher after the school also. All department interaction will be improved. Easy user interface and can be accessed from anywhere in the globe with connection of internet. The system is made responsive so that users can use it from any size of devices. All in all the concentration was fully on security, usability and efficiency. All these made the system unique of its kind.

1.5. A short overview of the remaining chapters

I have done some diagrams and testing during my analyses and development. I am going to show all my diagrams, descriptions, testing outcomes and other descriptions regarding the project. I am going to describe all these points in the remaining chapters.

Chapter 2

Initial Study

2.1. Project Proposal

Initial Conception

a) Brief Discussion of the concept

The main goal is to build a “school management system”. The system will be a web-based system which will provide solution for all types of solution of day to day activities. The system will automate the repeated and main tasks of a school so that work becomes faster. It will be done using web supported languages like html, css, javascript, bootstrap for front end and php for backend logical operations. I am going to describe all functionalities of the school management system below

1. Admin panel Activities

- Register Teacher
- Register Students
- See all overall reports
- Admit students
- See characteristic report of students
- Send notices
- See attendance report
- See results

- Create routine for teachers
- Create routine for students

2. Teacher panel Activities

- See routine and classtime
- Take attendance
- Give assignments
- Take assignments
- Mark students
- Generate results
- Multiple grading system
- Report wrong activities of any student

3. Student panel Activities

- See routine and class time
- See and download assignment materials
- Upload assignments
- See attendance report
- See marks
- See results
- See notices

4. Parents panel activities

- Notify important notice
- See routine and class time
- See attendance report
- See marks
- See results
- See notices

b) Proof of Concept

i) Prototyping

I am providing a prototype of the system which will be looking like that:

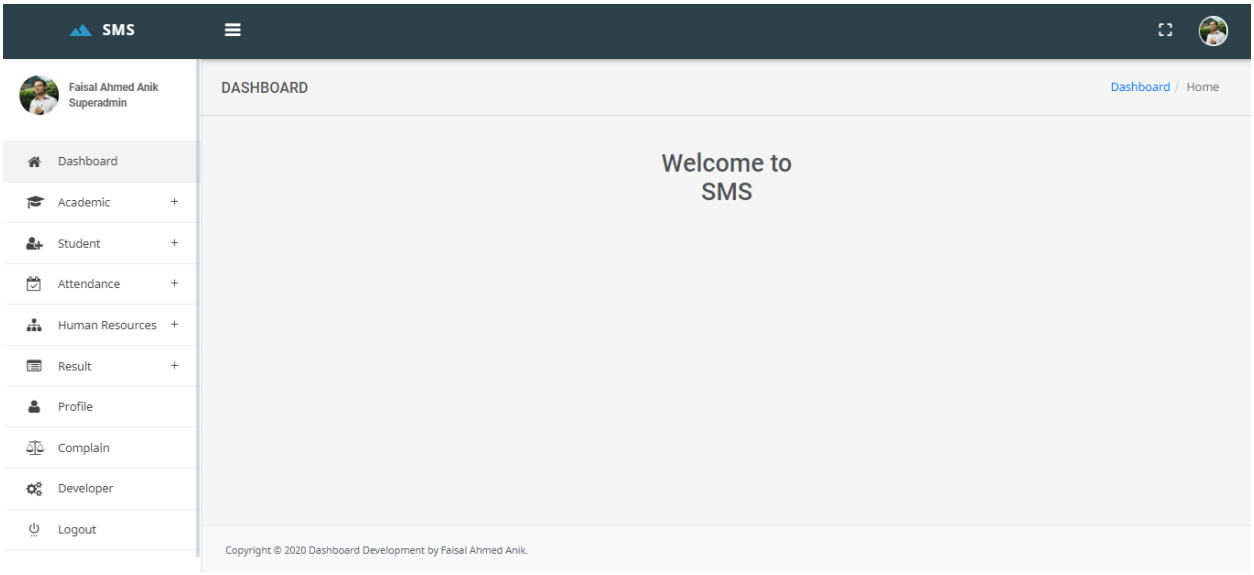


Figure 1: Prototype

ii) Initial Research

Market viability

School management system is a system which will make all repeated and main functional works automated for school officials, students and parents. This will make the communication, connection and work smoother and faster.

In our country most of the school are managing their work in analogue way. They don't have any system to make the works digital. Our government is also very much focused on making the schools digital and automated. So the system will be very much viable for our country. In the system there will be

multiple grading system which will help selling the system to internationally also. The system will be adaptive to many scenarios.

Comparative analysis

There are some school management available in the comparative market. But those has some problems and issues. Some system has grading system but not multiple grading system. Some has not any grading system. Some doesn't have any attendance system.

In my system I kept all nessecery features which are needed. I kept multi grading system, attendance system, marking system etc. those features made this system a complete school management system.

Identification of key milestones with dates

Task	Start date	End date	Duration
Starting project(generate Idea	**05/11/2019**	**10/11/2019**	**6 days**

and market research)			
Project proposal and submission	**11/11/2019**	**12/11/2019**	**2 days**

Analysis	**13/11/2019**	**16/11/2019**	**4 days**
Design	**17/11/2019**	**22/11/2019**	**6 days**
Development	**23/11/2019**	**01/12/2019**	**9 days**
Testing	**02/12/2019**	**04/12/2019**	**3 days**
Documentation	**05/12/2019**	**17/12/2019**	**13 days**
Total=			**43 days**

c) Outline of key activities

I have done researching the project fully. I have outlined the key activities to complete it. I have to make a system architecture design before building. I will collect new design and develop ideas to build the system fully. After that I will make and ERD diagram, DFD, Use case, class diagram and sequence diagram. Those will allow me to build the system more efficiently.

After that I will design and develop the system and test the full system module by module. After finishing testing and bug fixing, I will create a detailed documentation on it.

2.1 Background of the project

Schools take care of all of their works by analogue system here in our country. By this process they have to deal with a good amount of expense behind papers, employees and other sectors. They don't even find this easy to find or create any report for their work because of analogue system. All the data has to write repeatedly and time passes away.

So, to make things easier, efficient, faster and comfortable I decided to make a school management system. If I can allow the schools to do their major works through system then they will can save money and make the work more easier and faster. I will allow them to take attendance through the system, keep multiple grading system so that they can use their most liked system, multiple users like Admin, teacher, student and parent. Every user can use their required functionalities, I will also keep routine management system so that teacher and students both can see their routines through system.

So, the system will reduce a lot of workload for the schools and make all works easier and faster. This was the background of my project.

2.2. Problem Area

Here in Bangladesh there are some problem areas. In Bangladesh there are lest technical people to operate a system. Many management systems

available are not so user friendly so that they feel frustrated using them. There are many systems available which does not meet the requirements of the schools of our country. All teachers, students, parents, administration works doesn't meet by many systems which are available in market. All in all, the main problem area is user friendliness, meeting requirements.

2.3. Possible solution

I have figured out the possible solutions. I already visited some schools on fields which are still using analogue process. I have taken their requirements and also studied online to find out more required features. I will implement all types of required features so that requirement doesn't become an issue. I also tried to make the interface design so simple so that users can find out very easy to use the whole system. To make the schools personnel feel comfortable I can give them a short training or make a video demonstration of the system so that they become very tied up with the system easily. In these ways I can overcome possible problems.

Chapter 3

Literature Review

3.1 Discussion on problem domain based on published articles.

In Bangladesh there are some problem areas of using “School Management System”. One of the most important reasons on why it becomes thus exhausting to implement a school management system is that the software system doesn't continuously meet the wants and requirements of the business. There are continuously some specific tasks that such a system is anticipated to fulfil and also the issues happen once these expectations aren't met. this can be the rationale why it's therefore vital that you just, select a system that may do what you expect it to do. you wish to arrange the complete work properly and execute it well. this may ensure that the school management system is enforced properly.

I am going to discuss on some problems below which are mainly vital for school management systems:

- **Affordability**

Budget is one amongst the most important problems that might confront you once you attempt to implement the ERP system. this can be why before you are trying to implement such a system it's important for you to assess the type of cash that you simply would be ready to spare for the

aim. In fact, you wish to appear at the complete budget of your school so as to know things properly enough.

- **User requirements**

It is vital that the school management software system getting used in your school is in a position to satisfy the wants and expectations that you simply have from an equivalent.

- **Internet Connectivity**

This is forever a significant issue once it involves implementing school management systems. The factor regarding this can be that the systems utilized by schools currently are web-based. this is often why sensible connectivity is required in order that the system may be supplied with all the support that it wants.

- **Training**

The success of a school management system is often captivated with however well the end users are ready to use it. it's vital that everyone the technical and non-technical workers' members in your school skills to use the system. this is often applicable for the teachers furthermore since they might be operating with the system as well. you must ensure that they're ready to build the fullest use of the system.

- **Monitoring**

It would even be nice if the vendor is in a position to observe however the system goes when you've got implemented it. this is often a key component in ensuring that the system functions well and therefore the implementation

is productive. this might be in serious trouble a period of three months at once when the system has been put in place.

3.2 Discussion on problem solutions based on published articles

As there are problems, there are solutions also. I am providing the solutions of the problems.

- **Affordable price**

While setting price the vendor should analyze the market and also analyze the target audience budget range also. Of course the vendor should also think about their profit. But if the price is too high then the budget of the targeted audience then the system will not be sold any more. So setting a fair price is the solution of this problem.

- **Meet requirements**

The users are the soul of any system. If the requirements are not fulfilled, then the system cannot be sold. The most basic and important thing is to study the requirements properly, take user feedback and make the system easy for the users. By meeting the requirements this system can be resolved.

- **Ensure Internet**

This is the most basic requirement of any web based system. So the vendor should clarify the buyer properly that the system needs internet connectivity to perform perfectly. Nowadays everywhere data connection is

very much cheap and easy to get. So ensuring internet is an important thing.

- **Train users**

Generally, every system has its own way to operate. No matter how user friendly the system is. User may find some things hard to do. So training is the best solution to solve this issue. This is often why the end users – the employees – are provided active training so the school management system can be implemented in an exceedingly booming manner. it's additionally vital during this context to create positive that the management system marketer encompasses a technical team that will add a fanatical manner and causes you to get all the assistance you would like so as to implement it properly.

- **Monitor the system**

Every system needs maintenance and monitoring for taking out the best out of it. Generally, in every three months every system needs a checkup and maintenance. There are sure to be problems – it's necessary that the vendor addresses them. If the necessity is, the technical team of the vendor ought to take requests from you for facilitate and help. the simplest} factor to try and do during this regard is for them to offer their senior product specialists to you so the problems will be resolved at the earliest and within the most effective means.

3.3 Comparison of three/four leading solutions

- **SWOT-School management system**

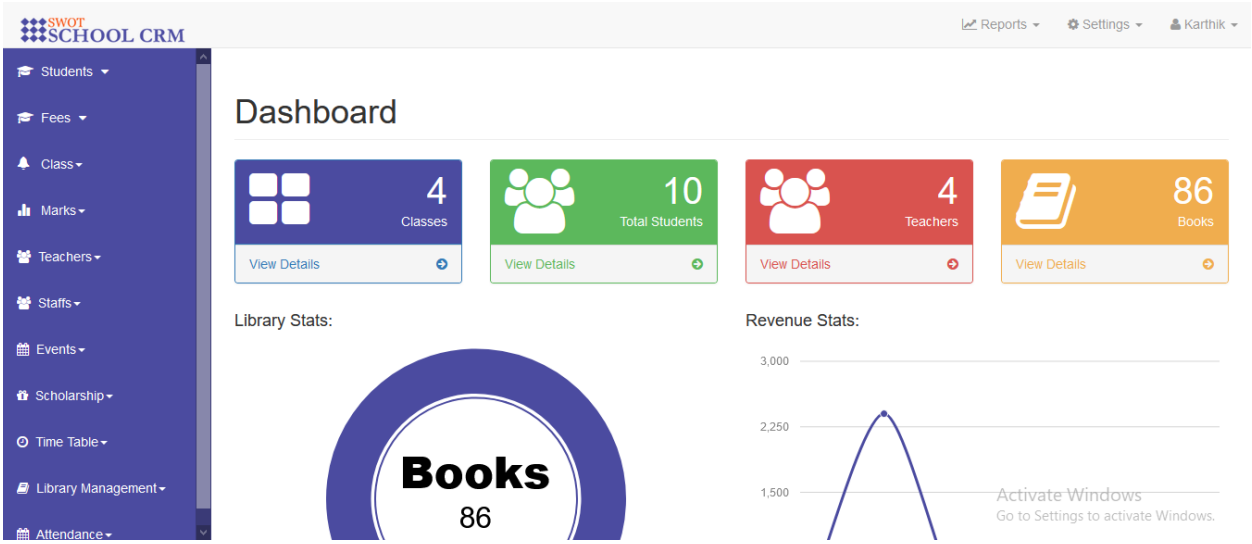


Figure 2:SWOT-School management system

SWOT SCHOOL is a revamped online management software for total school, college or any educational institutions management. You can keep student's records from admission to transfer, exams, marks, events, complains and scholarships, teachers' management, payroll, staffs and also library management. (http://demo.swot.co.in/sms/public/home)

- **Best Features:**
  - Total School or any type of Educational Institute Management System
  - Class Room & Time Table Management
  - Teachers & Staffs Management
  - Library Management
  - Events Management
  - Fees Management
  - Attendance Management

- **Limitations:**
  - No multiple user portal system

- No multiple grading system
- No grading system
- No result management system

- **ORA-School suit**

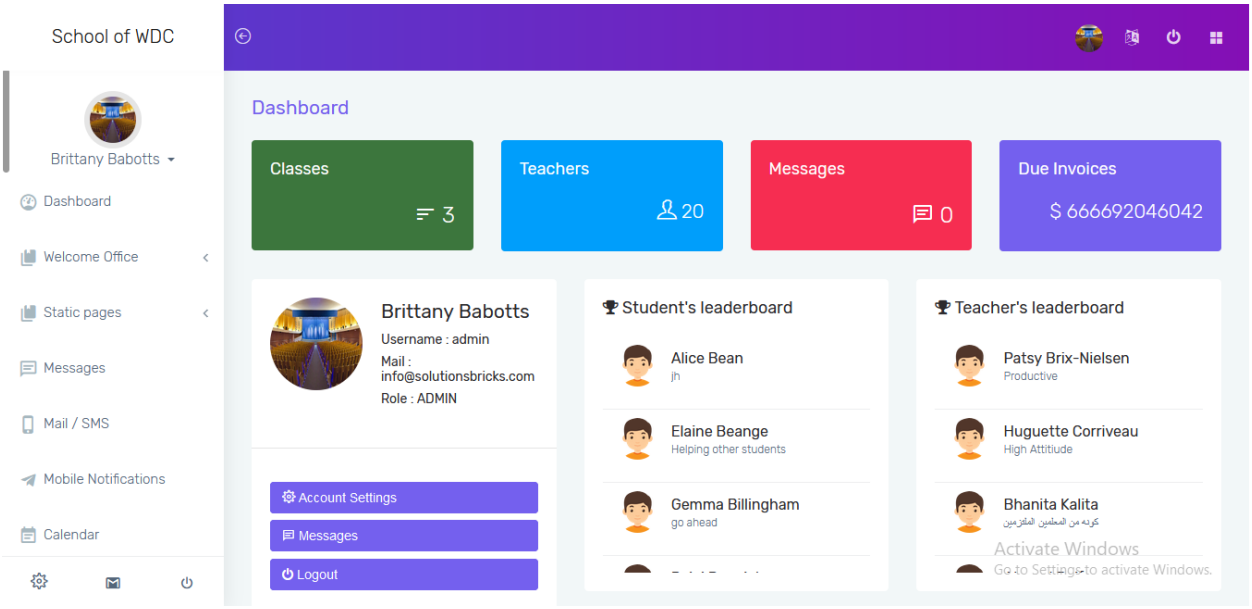


Figure 3:ORA-School Suit

Schoex application name has changed to OraSchool, New name with the same leadership and suite. (http://demo.solutionsbricks.com/schoex/login)

Best Features:

- ID cards module generation
- “My Payroll” User can see his payroll summary
- Support biometric for employees
- Support Department & design for teachers
- Support for custom headers for HTTP SMS configurations
- Support for controlling which user role can message another user role

- **Limitations:**

- No multiple grading system

- No grading system
- No result management system

- **The Next Gen-School Management system**

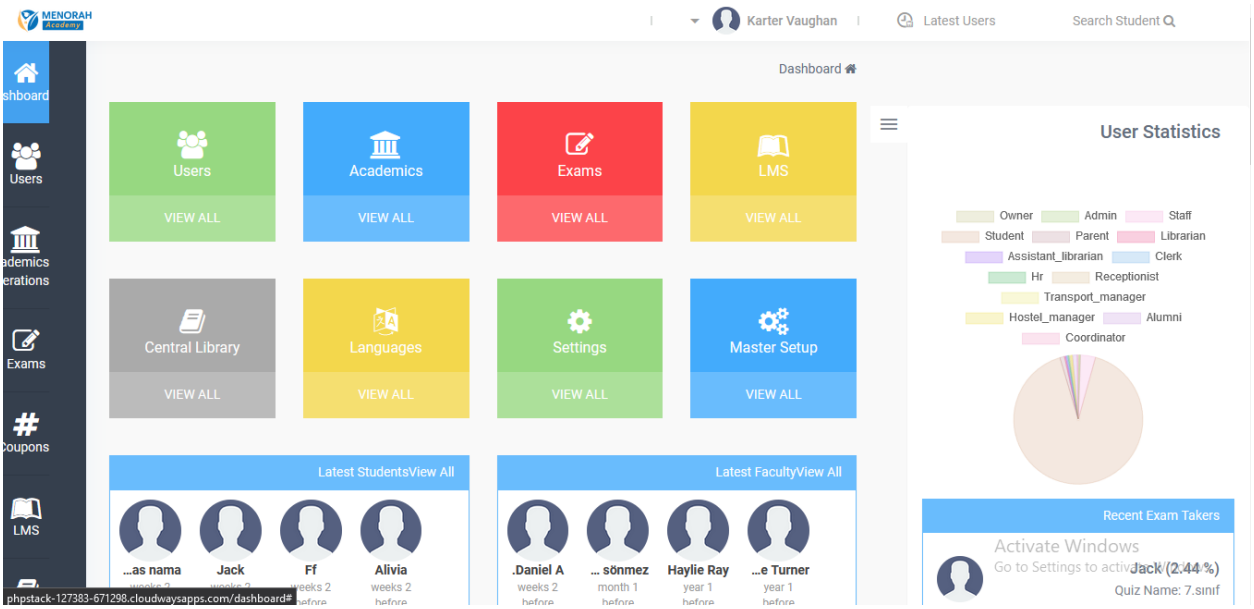


Figure 4:The Next Gen-School Management system

Menorah Academy is a multipurpose school management software which is suitable for colleges, schools and universities for academic, administration and management related activities. (http://phpstack-127383-671298.cloudwaysapps.com/login)

- **Best Features:**
  - Single system for School/College/University or All together
  - 2 Steps Installation process / Free Installation support
  - Courses Management with and without semesters
  - Complicated modules are made easy with Drag and Drop option
  - Create and manage Time table on fly with flexible print options
  - Generate own timetable for Staff/Student with print option.

- **Limitations:**
    - Portal is not user friendly
    - Not fully responsive
    - No multiple grading system
    - No grading system
    - No result management system

3.4 Recommended approach

After studying all the available systems, I found some points which are very important to have in all the user portals. User friendliness, responsiveness is the most important thing to have. Other points are given below:

- **Admin Panel**
    - Admission system should be available
    - Report generation should be there
    - Multiple grading system
    - Result management
    - Attendance report
    - Routine management
- **Teacher Panel**
    - Routine view
    - Result management
    - Grade management
    - Attendance management
- **Student Panel**
    - Result view
    - Grade view
    - Attendance report

- Notice view

These are some recommended approach which can be made to make any system a better system.

Chapter 4

Methodology

What to use

Methodology is very important part of any development. Setting correct and perfect methodology is a vital part of development. The methodology should be compatible with development area. Agile is a very famous and effective framework. In agile there is a methodology names Dynamic Systems Development Method (DSDM). This is a methodology which gives a framework to build and manage a system development. It came from the version of Sociologist Principle. DSDM is an iterative code methodology among which each and every iteration follows the 80% rule that just enough work is required for each increment to facilitate movement to the subsequent increment. The remaining detail is usually completed later once plenty of business requirements are noted or changes are requested and accommodated.

Why to use

DSDM features a broader focus than most alternative Agile approaches therein it deals with projects instead of simply the development and delivery of a product (typically software). The project context needs attention on the broader business need and every one aspects of the answer that evolves

to fulfill that require. DSDM features a long record of successful Agile project delivery altogether forms of company environments, and has verified to be totally scalable, operating effectively in tiny simple businesses, large, complicated organizations and in extremely regulated environments. It additionally has been shown to be equally effective for each IT and non-IT projects, for instance business modification projects. I am going to state some points for which this methodology will be best for this project.

- The business is more likely to feel ownership of the solution as it evolves and, most importantly, as it transitions into live use
- Prioritization will enable a project to be delivered on time whilst protecting the quality of what is being delivered
- The risk of building the wrong solution is greatly reduced
- The final solution is more likely to meet the real business need
- Deployment is more likely to go smoothly, because of the co-operation of all parties concerned throughout development

These are some points which made DSDM a perfect methodology for this project.

- **Sections of methodology**

DSDM always focuses on some principals which helps deliver a perfect product to the user.

➢ Focus on the business need

This always keeps and focuses on the business needs. This will study the needs and requirements of any business first then start building the project using those requirements.

- ## Deliver on time

This DSDM methodology always makes sure that the project is delivered on time always. This makes the process in iterations and all iteration has its time boxing. So my meeting all time boxes, delivery on time is set.

- ## Collaborate

DSDM always collaborates with the client. Always keeps in touch in every iteration. If the clients are happy then process moves. So collaboration is always ensured.

- ## Never compromise quality

Quality is the key thing to go and meet success. So DSDM ensures quality in every iteration. If quality is not ensured, then it is sent back and fixed to go forward.

- ## Develop iteratively

Iteration is a basic feature of Agile framework. It helps the system to be built perfectly from foundation. Each iteration completes perfectly then work moves to the next iteration.

- ## Communicate continuously and clearly

In every iteration communication with customer is available. If needed customer communication happens continuously and ensured high quality always.

- **Implementation plans**

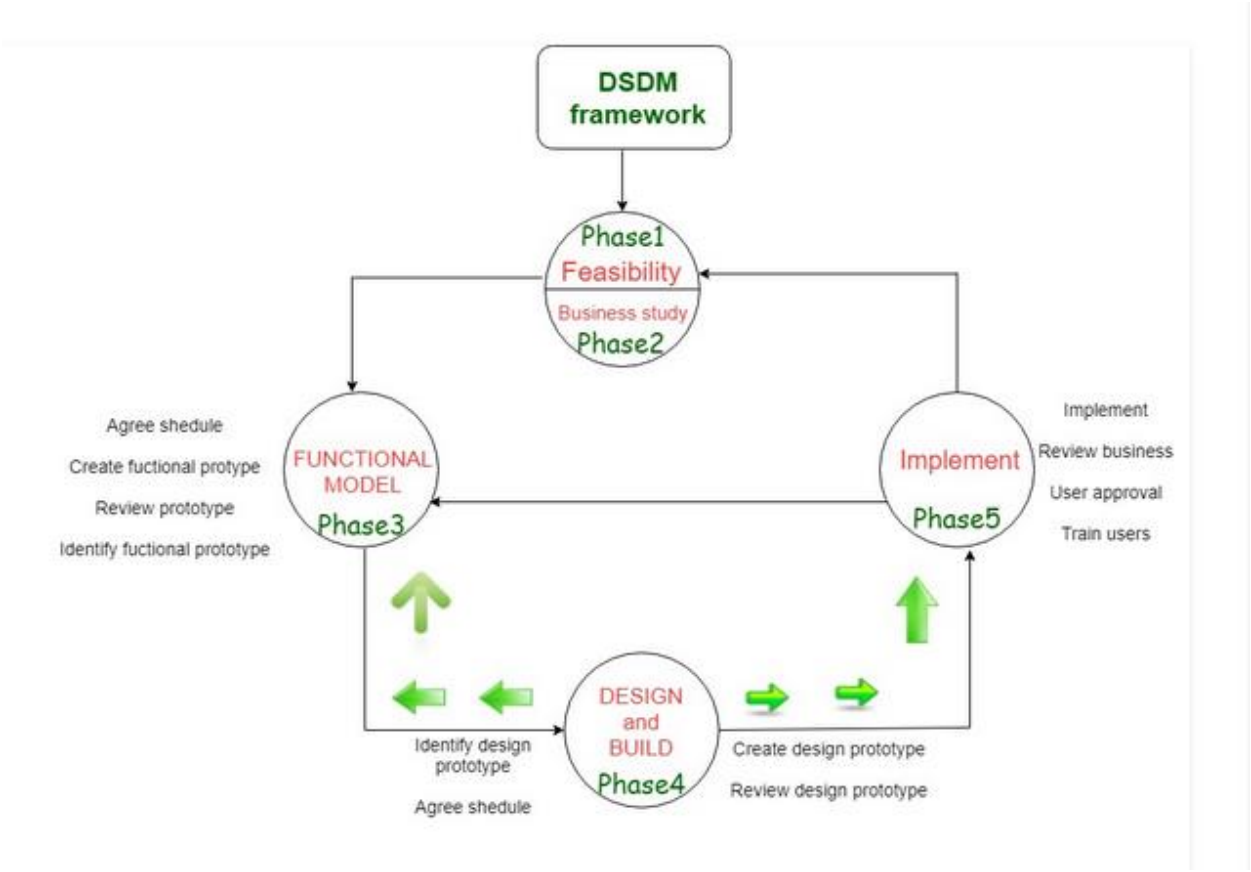


Figure 5:DSDM Life cycle

To implement the DSDM in the project development, I have to follow some steps which will allow me implement DSDM perfectly.

➢ Feasibility Study

It establishes the essential business wants and constraints involving the applying to be designed then assesses whether or not or not the applying may be a viable candidate for the DSDM methodology.

➢ Business Study

It establishes the utilization and data wants that will allow the applying to produce business value; in addition, it's the essential application style and identifies the maintainability wants for the applying.

➢ Functional Model Iteration

It produces a group of progressive prototypes that demonstrate usefulness for the shopper. The intent throughout this unvarying cycle is to gather any wants by eliciting feedback from users as they exercise the paradigm.

➢ Design and Build Iteration

It revisits prototypes designed throughout helpful model iteration to form certain that everybody has been designed throughout a way that will alter it to produce operational business value for end users. In some cases, helpful model iteration and elegance and build iteration occur at an equivalent time.

➢ Implementation

It places the most recent code increment (an “operationalized” prototype) into the operational surroundings. It is good to know that:

a) The increment may not be100% complete.
b) Changes are also requested because the increment is already being in place.

In both the cases, DSDM development work continues by returning to the helpful model iteration activity.

Chapter 5

Planning

- ## Project Plan

Project planning refers to everything you are doing to line up your project for achievement. it's the method you bear to ascertain the steps needed to outline your project objectives, clarify the scope of what has to be done and develop the task list to try and do it. Project planning software system is commonly wont to facilitate craft thorough project plans.

➢ Work Breakdown Structure (WBS)

Task	Start date	End date	Duration
Starting project(generate Idea and market research)	05/11/2019	10/11/2019	6 days
Project proposal and submission	11/11/2019	12/11/2019	2 days

➢ Gantt Chart

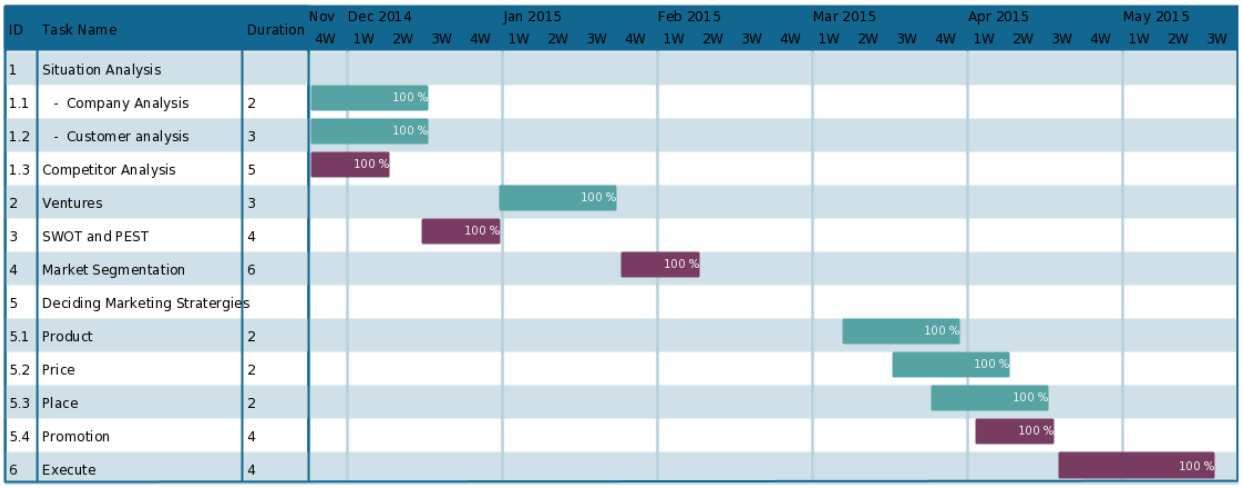


Figure 6:Gantt Chart for SMS

➢ Test Plan

Required tests

The system has to be tested in some methods so that the system become perfect and doesn't contain any bugs. The tests will help to make the system more perfect for real world use. Without them software may contain a lot of bugs which will cause serious issues in real life. The needed tests are given below:

1. Unit testing
2. Module testing
3. Integration testing

Test Case

Test case is a set of conditions or variables under which a tester will determine whether a system under test satisfies requirements or works correctly.

User acceptance test plan

I have plan to make user acceptance testing of this system. I have though to giving my system to a school for using free so that I can take their feedbacks and learn the user acceptance of the system. This will help me build future developments also.

Chapter 6

Feasibility

- **All possible type of feasibility**

Economic Feasibility

Development Costs

Development fee may be very minimal because the gear and technology used are to be had online. It's a project done by myself so there is no employee cost. Development time is properly deliberate and will no longer have an effect on different operations and activities of the individuals. Once the project has been developed, the schools purchasing the project will be offered with a guide for training purposes will be supplied for those individuals that need it. There is no need to purchase new hardware since the existing computer systems can nonetheless be used with a connection with internet.

Benefits

Performance advantages:

Augmented speed of report production. quicker creation, access, modification and retrieval of information. Diminished redundancy or duplication of information. Timely access of students' data and school update. Improved interaction between students and teachers.

Cost-Avoidance Benefits:

Avoid prices of buying information backups and storage since everything are on-line and automatic. Avoid prices of oldsters having to travel long distances to check their kids' performance. Employee Reductions can cut back the price of salaries since and integrated system will want simply a number of employees, no want for employees in each department. it's evident that the system is economically feasible, the advantages outweigh the system prices among the defined period of your time acceptable to the user/client. the utilization of system places the school at a competitive advantage.

Technical Feasibility

The school management system is internet based and so will be accessed through any browsers. The solution is sensible since the objectives of the system development are possible and realistic. The technology to be used is obtainable, this embrace use of programming language PHP, markup language HTML, CSS for styling and MySQL database to develop internet based applications. Schools who will get this project, they will not have to purchase many things just computer with internet will make it possible for use. No intense training will need to be given.

Operational Feasibility

The users will be very comfortable using this project. As this is a web based software so users can access it from anywhere in the world with internet connection. It is less costly and user friendly. It will resolve many repetitive works for admin, teacher and students. Everything will be in front of eyes. It requires less equipment to make it workable. Just computer and internet will make this workable. All departments will be connected that's why interaction will be developed by using it. Overall operational advantages are a lot with this project.

- ## Cost Benefit Analysis

Cost benefit analysis in project management is one more tool in your toolbox. This one has been devised to evaluate the cost versus the benefits in your project proposal. It begins with a list, as so many processes do.

The purpose of cost benefit analysis in project management is to have a systemic approach to figure out the pluses and minuses of various paths through a project, including transactions, tasks, business requirements and investments. Cost benefit analysis gives you options, and it offers the best approach to achieve your goal while saving on investment. (projectmanager, 2020)

The approximate cost analysis is given below:

Project Name: SMS **Date: 06 May, 2020**

Hardware	Year 1	Year 2	Year 3	Total
Server	1500	3000	4500	9000

Software	800	800	800	2400
Internet	3000	3000	3000	9000
Training	500	0.00	0.00	500
Development	3000	2000	2000	7000
Total	8,800	8,800	10,300	27,900

- **DSDM**

DSDM (Dynamic System Development Method) is vendor-independent, covers the entire lifecycle of a project and provides best practice guidance for on-time, in-budget delivery of projects, with proven scalability to address projects of all sizes and for any business sector. (agilebusiness, 2020)

DSDM is good for this project because of its agile based vendor-independent system.

Chapter 7

Foundation

- **Overall Requirement List**

List of functional requirements

Requirement Category	Explanation
Must have	The school management system must have authentication system The school management system must have admin, teacher and student portal
Should have	The school management system should have better communication system
Could have	The school management system could have parent separate portal The school management system could have announcement system
Won't have	The school management system won't have any installation process due to it is a web based PHP project.

List of system-wide non-functional requirements

The list for the school management system non-functional requirement are given below:

- User friendly system.
- Highly secure because many school student, teacher data will be available.
- Make confidential data secured and not allow others to edit those data.
- Reports and summaries should be available.
- Student should only see own data in the student's portal.
- The site should have secured password authentication system.
- The site should be fully responsive for any kind of devices.

- **What Technology to be implemented**

To develop the School Management, I used PHP based framework called Laravel as the main programming language. I also used a huge amount of Vue.Js also. Beside that for validation, searching there is slight use of JavaScript as well.

I have chosen PHP as my main programing language for some reasons.

- PHP and Laravel is Open Source; which means you never need wait for release the next version if something is not working or pay for upgrades.
- PHP can be extended.
- Maximum databases are supported.
- PHP can run on most of the platforms. That's why it is platform independent.
- Compatible with most of the servers like IIS and APACHE.
- PHP requires low development and maintenance cost.
- PHP provides high performance and reliability.
- Vue.Js gives a huge user experience.
- Vue.Js loads the data from server on a blink by not reloading the page.

For all these benefits and advantages, I have chosen PHP based Laravel framework and Vue.Js. For web based applications PHP is the most popular and largely used language for a huge amount of time. It gives efficiency, comfort, reliability and most importantly security.

- **Recommendations and Justifications**

To develop my school management system, I thought about security, usability, comfort, efficiency a lot. That's why choosing programing method is very important thing. To make the project look good, simple and easy to use I chose html, css3, JavaScript. I made the site simple and responsive.

To power the backend and do the logics I used PHP based Laravel Framework and MySQL to handle the database. PHP is highly secure and fast as a backend data management method. The MySQL is also very secure to handle. With Latavel framework of PHP I have full freedom of handling data as I want. That's why it becomes very secured as well as fast because big libraries don't need to be loaded every time. PHP is a cross platform language. Easy to use and stable for any web platform. This enables a lot of control to the developer to work with. I used Vue.Js also in a huge scale. It allows to load any type of data from database instantly without loading. Everything operates real-time.

MySQL provides data security, on demand scalability, high performance and comprehensive transactional support. That's why I used the combination of all these methods in my project.

Chapter 8

Exploration

- **Old Full System Use Case**

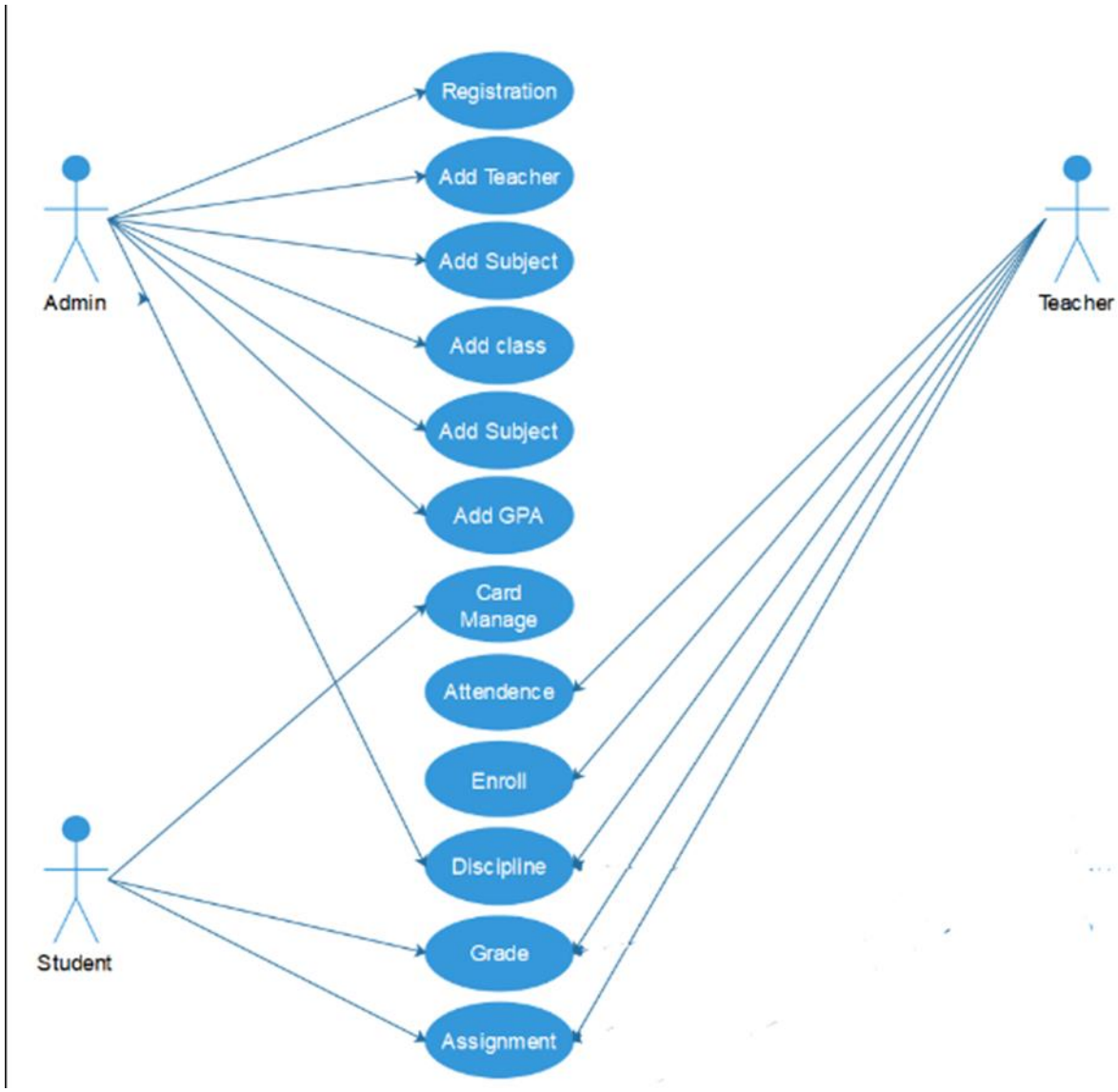


Figure 7:Old system use case

- **Old Full System Activity Diagram**

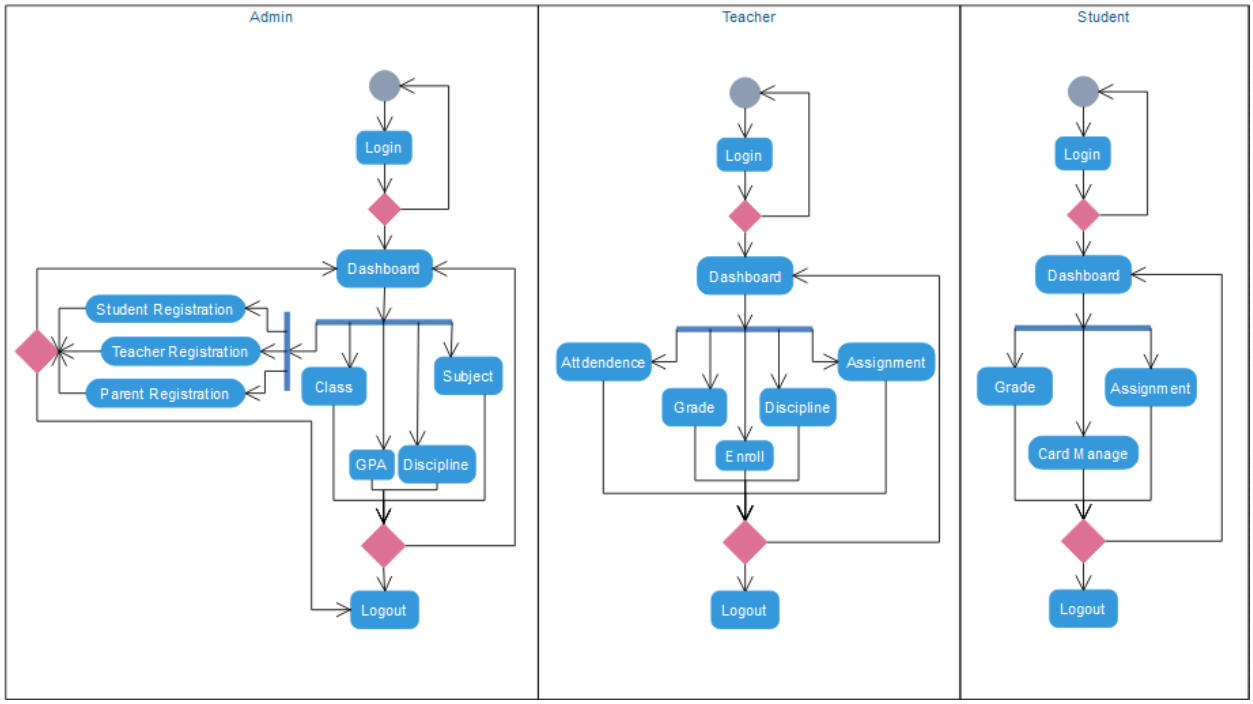


Figure 8:Old system activity diagram

- **Prototype of new system**

Login Panel Prototype:

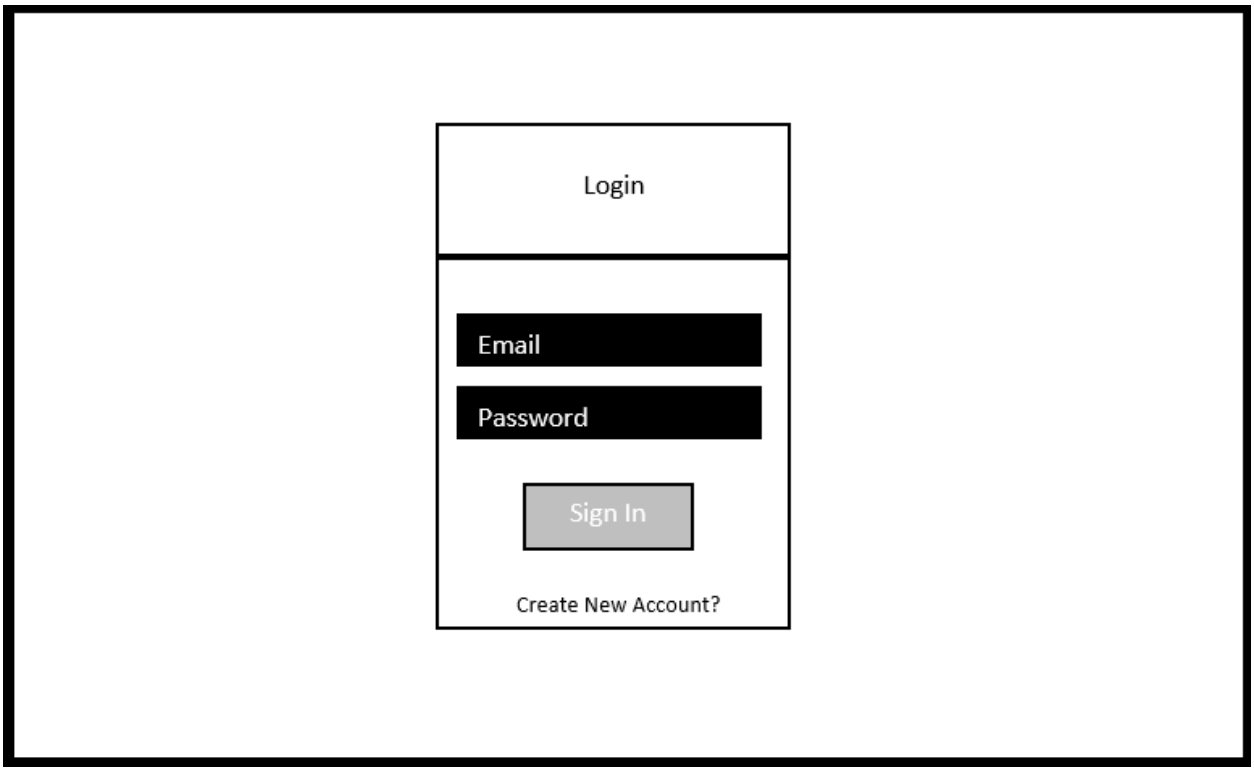


Figure 9:Login Panel Prototype

Main Dashboard prototype:

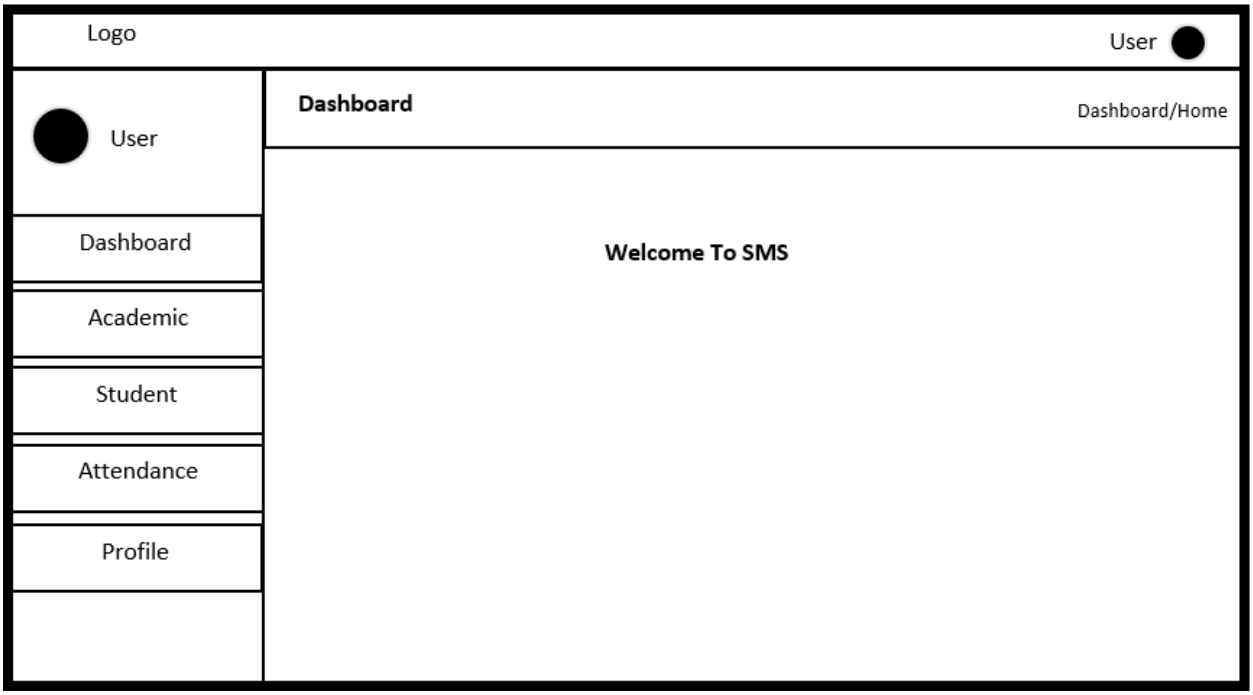


Figure 10:Main Dashboard Prototype

Chapter 9

Engineering

- **New System Modules**

Previously I have discussed the full system features and modules which will be helpful to understand the things.

- **Use Case**

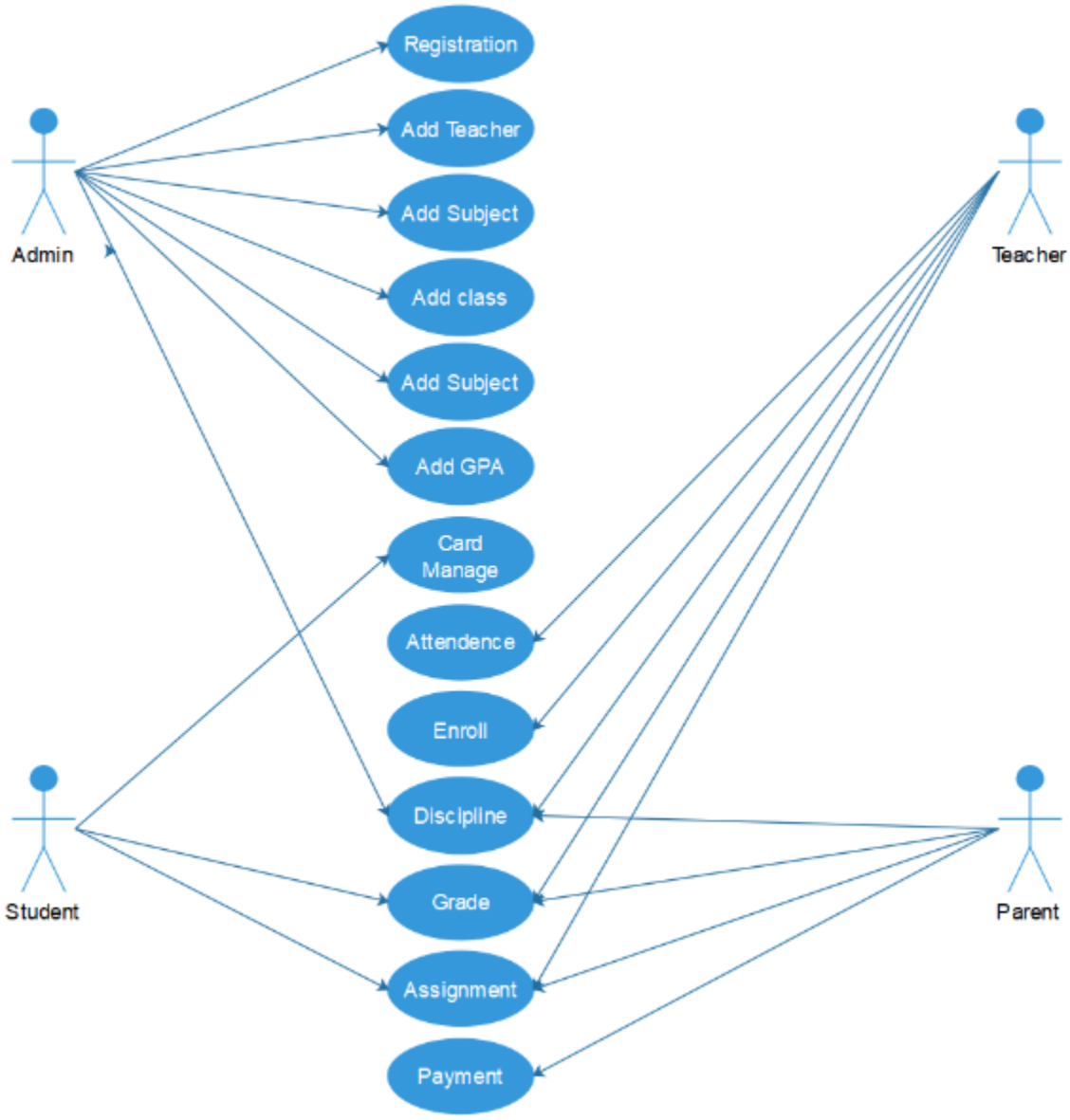


Figure 11:Use case of School Management System

- **Class Diagram**

As the system is made using procedural format that's why it doesn't have any definite class diagram. I prepared an initial class diagram which is provided below.

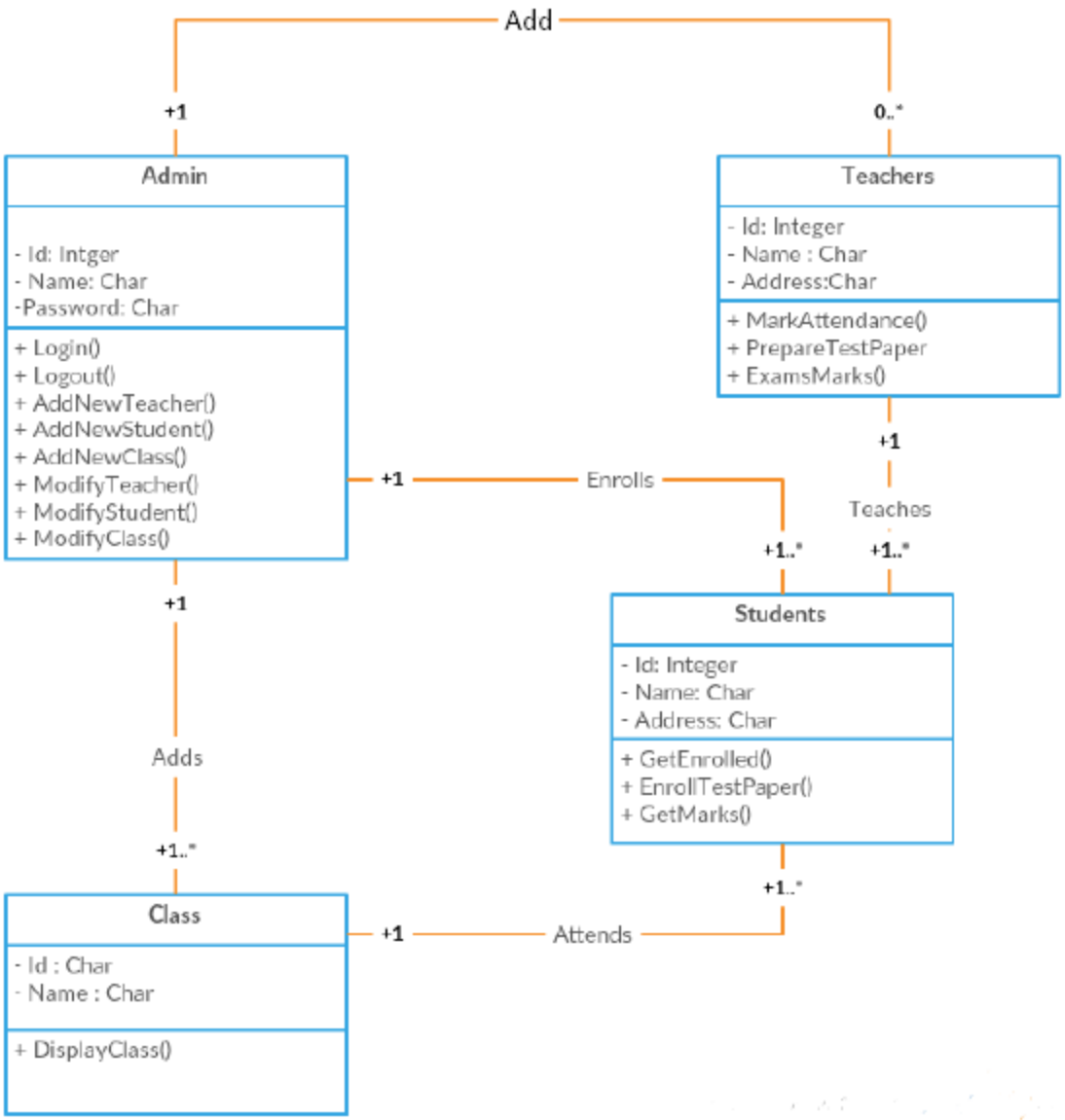


Figure 12:Initial Class Diagram

Here I prepared an initial class diagram for my School Management System. Here I prepared it as Admin adds Teacher, enrolls Students, adds classes. Teachers can mark attendance, give marks. Students can view their classes. This was my initial class diagram.

- **ERD Diagram**

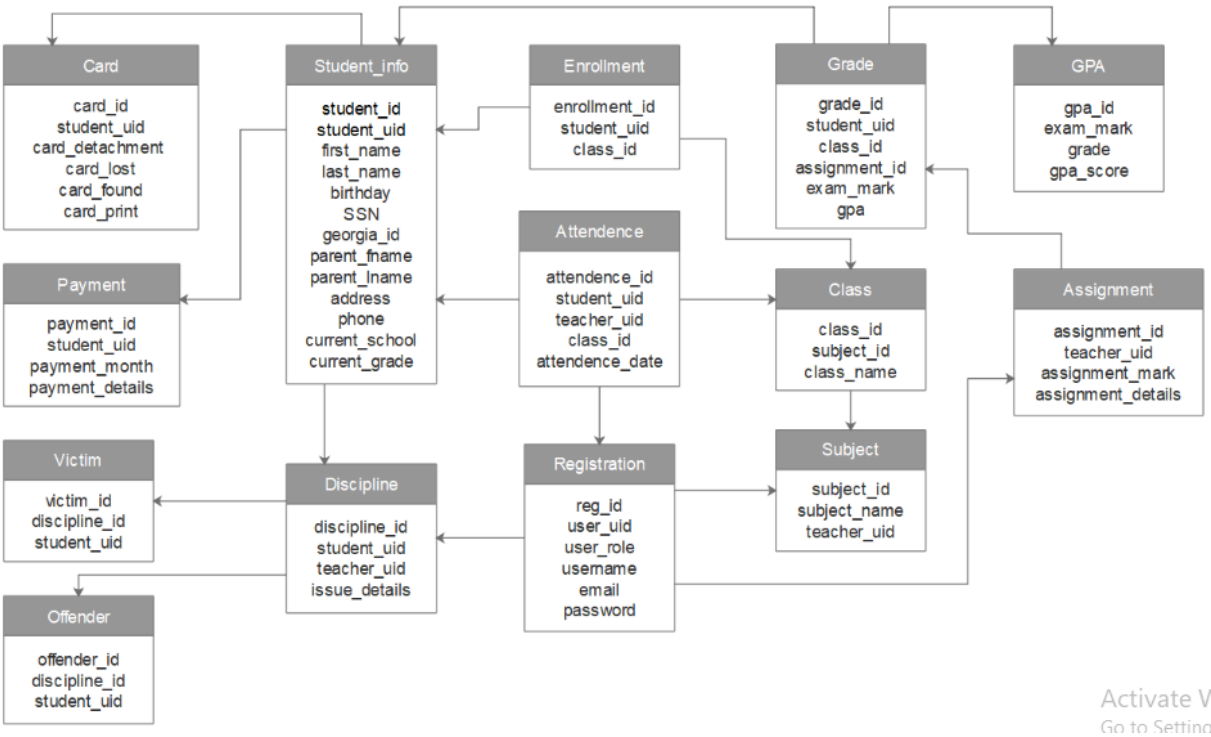


Figure 13:ERD diagram of School Management System

- **Sequence Diagram**

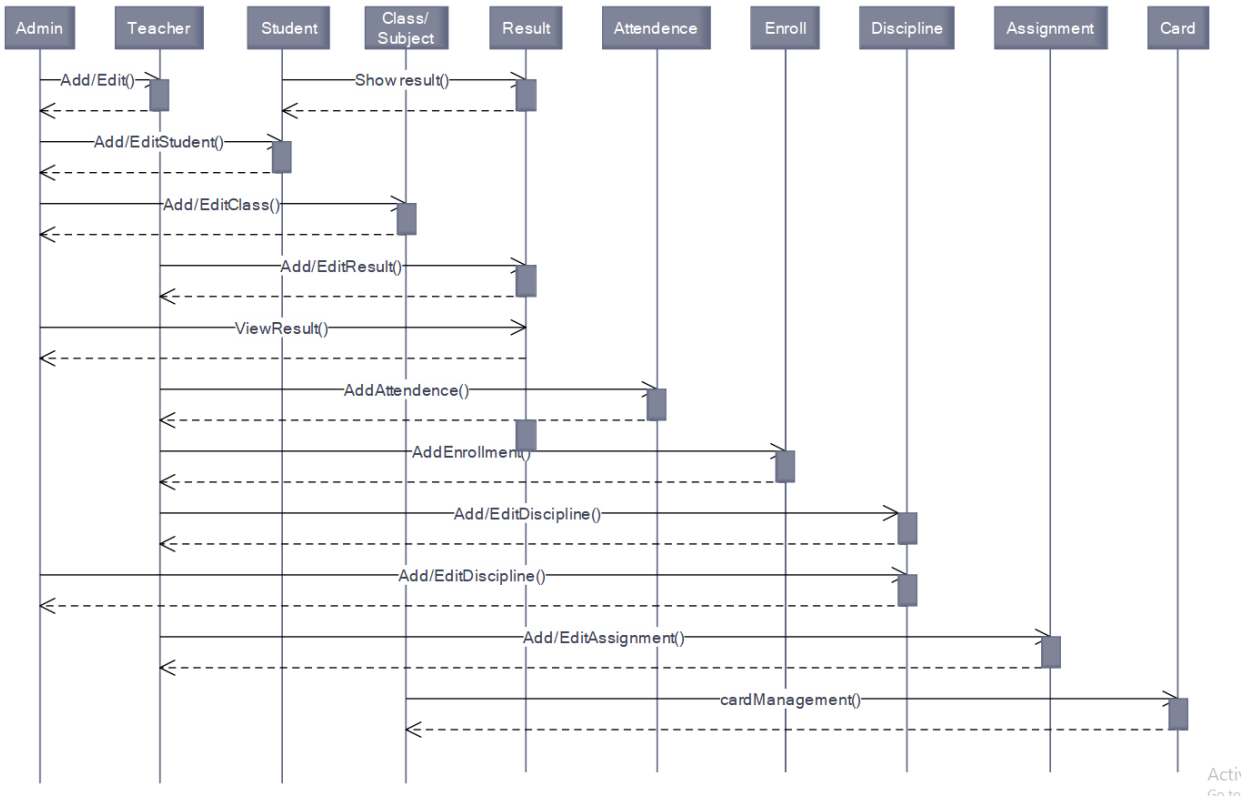


Figure 14:Sequence diagram

- **New System Modules**

The new system will have a lot more that the previous. This system has Laravel and vue.js features which are faster and improved. It has better error handling. It has attendance system. Student portal for student and parent. Complain box system for complaining. Multiple grading system. Result mechanism. Id card system etc.

- **System Interface Design**

Login Panel:

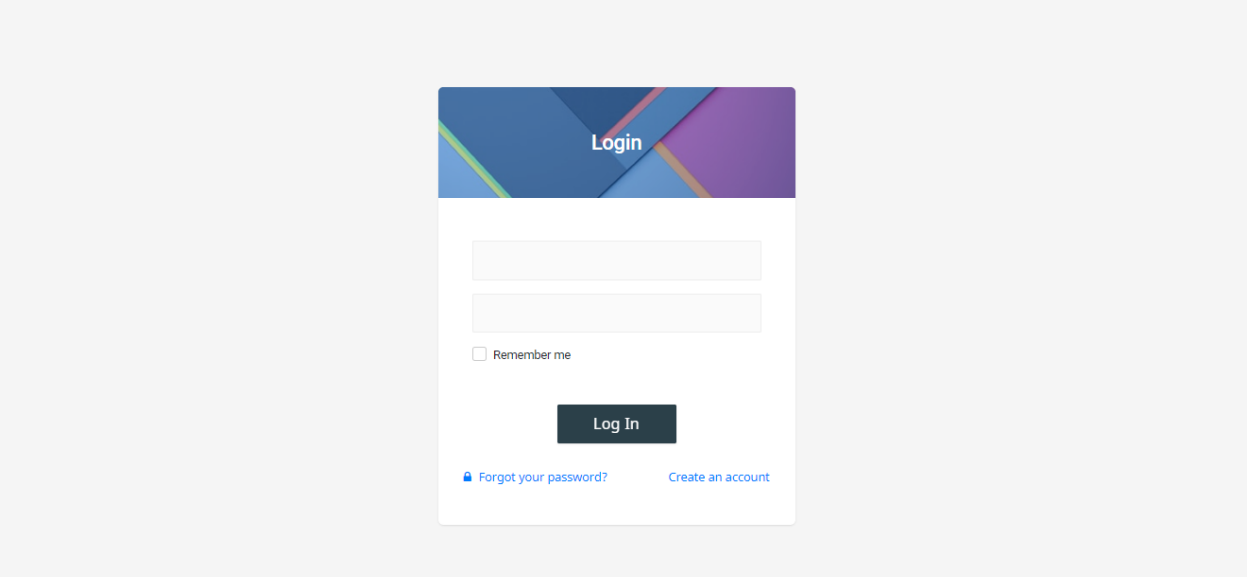


Figure 15:Login Panel

Admin panel:

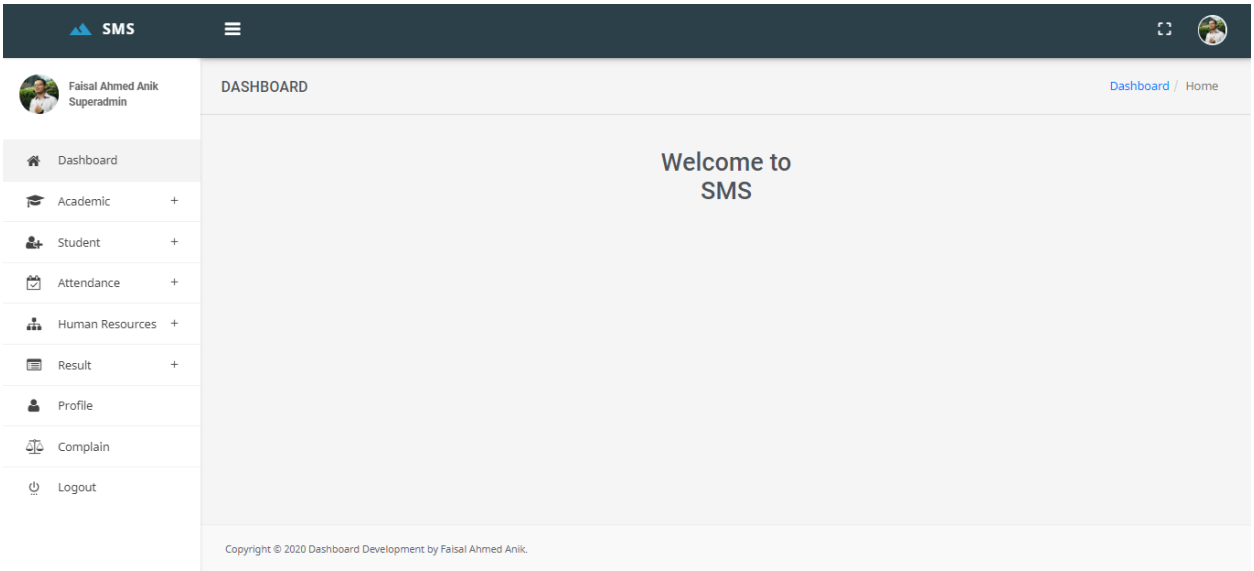


Figure 16:Admin Panel

Teacher Panel:

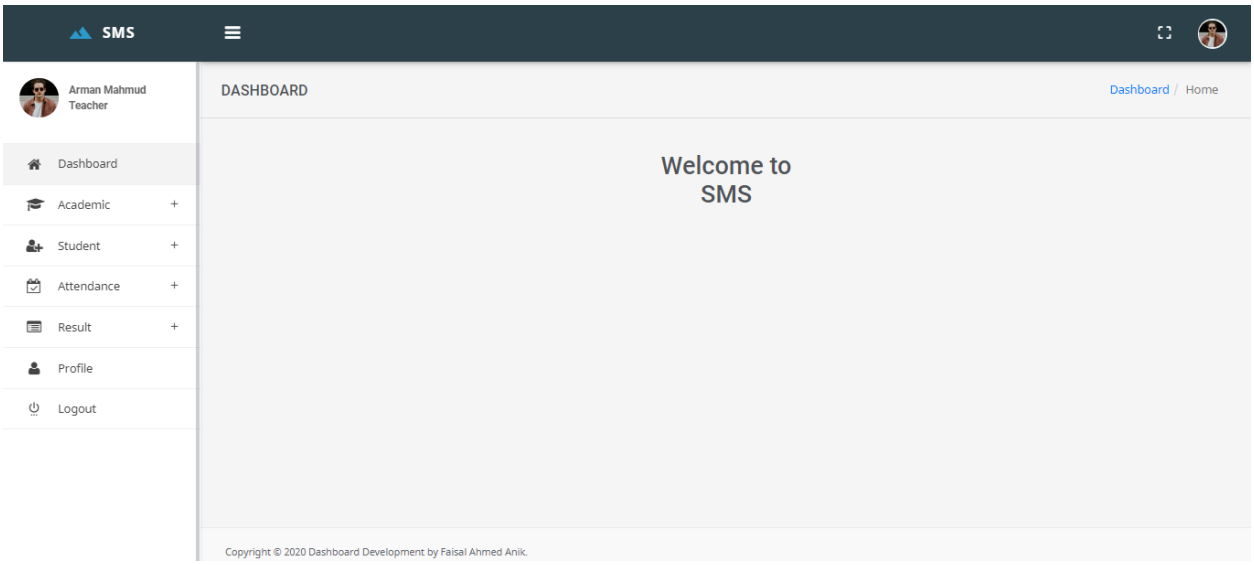


Figure 17:Teacher panel

Student & Parent panel:

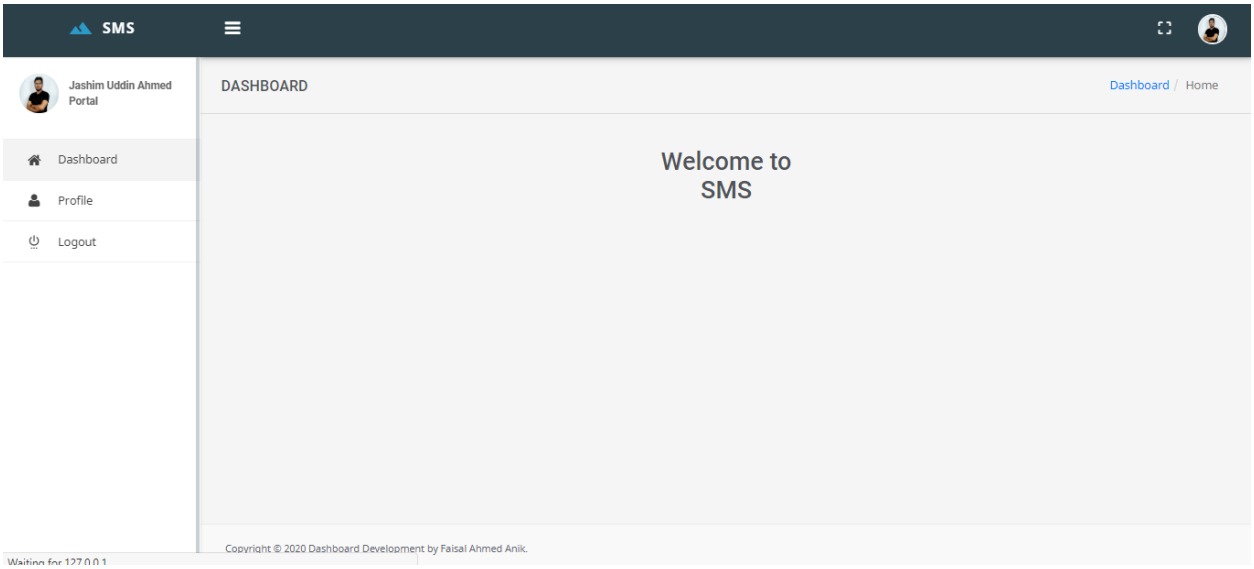


Figure 18:Student & Parent panel

Chapter 10

Deployment / Development

- ## Core Module Coding Samples

The school management system is built with some core functionalities which are the main backbone of the system. I am going to give code examples of those code functions below:

- **Login**

```php
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Foundation\Auth\AuthenticatesUsers;

class LoginController extends Controller
{
    /*
    |--------------------------------------------------------------------------
    | Login Controller
    |--------------------------------------------------------------------------
    |
    | This controller handles authenticating users for the application and
    | redirecting them to your home screen. The controller uses a trait
    | to conveniently provide its functionality to your applications.
    |
    */

    use AuthenticatesUsers;

    /**
     * Where to redirect users after login.
     *
     * @var string
     */
    protected $redirectTo = '/home';

    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('guest')->except('logout');
    }
}
```

Figure 19:Login Module Code

- **User CRUD Module:**

```php
public function index()
{
    return response()->json([

        'users' => User::latest()->paginate(10),
        'roles' => UserRole::get()

    ], Response::HTTP_OK);
}
/**
 * Store a newly created resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    $this->validate($request, [
        'email'=>'required|email|unique:users',
        'password'=>'required|min:8|confirmed',
        'role_id'=>'required'
    ]);

    if($request->photo) {
        $name = substr(base_convert(sha1(uniqid(mt_rand())), 16, 36), 0, 9);
        $imageName = $name . '-' . time() . '.' . explode('/', explode(':', substr($request->photo, 0, strpos($request->photo, ';')))[1])[1];
        Image::make($request->photo)->save(public_path('images/users/').$imageName);
    } else {
        $imageName = 'avatar.jpg';
    }
```

```php
      $create = User::create([
          'email'  => $request->email,
          'password'  => Hash::make($request->password),
          'role_id' => $request->role_id,
          'photo' => $imageName,
          'status' => (boolean)$request->status
      ]);

      return ['message' => 'Created Successfully'];

  }

  /**
   * Display the specified resource.
   *
   * @param  int  $id
   * @return \Illuminate\Http\Response
   */
  public function show($id)
  {
      //
  }

  /**
   * Update the specified resource in storage.
   *
   * @param  \Illuminate\Http\Request  $request
   * @param  int  $id
   * @return \Illuminate\Http\Response
   */
  public function update(Request $request, $id)
  {
      $user = User::findOrFail($id);
      $this->validate($request, [
          'email'=>'required|email|unique:users,email,'.$user->id,
          'password'=>'sometimes|min:8',
          'role_id'=>'required',
      ]);
```

```
        $currentPhoto = $user->photo;
        if($request->photo != $currentPhoto) {
            $name = substr(base_convert(sha1(uniqid(mt_rand())), 16, 36), 0, 9);
            $imageName = $name . '-' . time() . '.' . explode('/', explode(':', substr($request->photo, 0, strpos($request->photo, ';')))[1])[1];
            Image::make($request->photo)->save(public_path('images/users/').$imageName);
            $request->merge(['photo' => $imageName]);

            $oldPhoto = public_path('images/users/').$currentPhoto;
            if(file_exists($oldPhoto)){
                @unlink($oldPhoto);
            }
        }

        if(!empty($request->password)) {
            $request->merge(['password' => Hash::make($request->password)]);
        }

        $user->update($request->all());
        return ['message' => 'Updated Successfully'];
    }
    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
        $user = User::findOrFail($id)->delete();
        return ['message' => 'Deleted Successfully'];
    }

    public function search()
    {
        if($search = \Request::get('q')) {
            $users = User::where(function($query) use ($search) {
                $query->where('name', 'LIKE', "%$search%")->orWhere('email', 'LIKE', "%$search%");
            })->paginate(5);
        } else {
            $users = User::latest()->paginate(10);
        }
        return $users;
    }
```

Figure 20:User CRUD Module code

- **Admission & Sibling management Module:**

```
        if($request->photo) {
            $uniqueStr = substr(base_convert(sha1(uniqid(mt_rand())), 16, 36), 0, 9);
            $imageName = $request->name. '-' . $uniqueStr . '-' . time() . '.' . explode('/', explode(':', substr($request->photo, 0,
            strpos($request->photo, ';')))[1])[1];

            Image::make($request->photo)->save(public_path('images/students/').$imageName);
        } else {
            $imageName = 'avatar.jpg';
        }
        $request->merge(['photo' => $imageName]);

        if($request->sibling_id == ''){
            if($request->father_photo) {
                $uniqueStr = substr(base_convert(sha1(uniqid(mt_rand())), 16, 36), 0, 9);
                $fatherImageName = $request->name . "'sfather-" . $uniqueStr . '-' . time() . '.' . explode('/', explode(':', substr($request-
                >father_photo, 0, strpos($request->father_photo, ';')))[1])[1];

                Image::make($request->father_photo)->save(public_path('images/parents/').$fatherImageName);
            } else {
                $fatherImageName = 'avatar.jpg';
            }
            $request->merge(['father_photo' => $fatherImageName]);

            if($request->mother_photo) {
                $uniqueStr = substr(base_convert(sha1(uniqid(mt_rand())), 16, 36), 0, 9);
                $motherImageName = $request->name . "'smother-" . $uniqueStr . '-' . time() . '.' . explode('/', explode(':', substr($request-
                >mother_photo, 0, strpos($request->mother_photo, ';')))[1])[1];

                Image::make($request->mother_photo)->save(public_path('images/parents/').$motherImageName);
            } else {
                $motherImageName = 'avatar.jpg';
            }
            $request->merge(['mother_photo' => $motherImageName]);

            if($request->guardian_photo) {
                $uniqueStr = substr(base_convert(sha1(uniqid(mt_rand())), 16, 36), 0, 9);
                $guardianImageName = $request->name . "'sguardian-" . $uniqueStr . '-' . time() . '.' . explode('/', explode(':',
                substr($request->guardian_photo, 0, strpos($request->guardian_photo, ';')))[1])[1];

                Image::make($request->guardian_photo)->save(public_path('images/parents/').$guardianImageName);
            } else {
                $guardianImageName = 'avatar.jpg';
            }
```

```
    } else {
        $sibling = Student::findOrFail($request->sibling_id);
        if(isset($sibling->sibling_id)) {
            $request->merge(['sibling_id' => $sibling->sibling_id]);
        } else {
            $siblingId = $sibling->name . '-' . substr(base_convert(sha1(uniqid(mt_rand())), 16, 36), 0, 20);
            $request->merge(['sibling_id' => $siblingId]);
            $sibling->sibling_id = $siblingId;
            $sibling->save();
        }
    }

    $request->merge(['password' => Hash::make($request->password)]);
    $request->merge(['created_at' => Carbon::now()->toDateTimeString()]);
    $request->merge(['role_id' => 5]);

    $createStudent = Student::create($request->all());
    $createUser = User::create($request->all());
    return ['message' => 'Created Successfully'];
}
```

```
    $currentDadPhoto = $student->father_photo;
    if($request->father_photo != $currentDadPhoto) {
        $uniqueStr = substr(base_convert(sha1(uniqid(mt_rand())), 16, 36), 0, 9);
        $fatherImageName = $request->name . "'sfather-" . $uniqueStr . '-' . time() . '.' . explode('/', explode(':', substr($request-
        >father_photo, 0, strpos($request->father_photo, ';')))[1])[1];

        Image::make($request->father_photo)->save(public_path('images/parents/').$fatherImageName);
        $request->merge(['father_photo' => $fatherImageName]);

        $oldDadPhoto = public_path()."/images/parents/".$currentDadPhoto;
        if(file_exists($oldDadPhoto)){
            unlink($oldDadPhoto);
        }
    }

    $currentMomPhoto = $student->mother_photo;
    if($request->mother_photo != $currentMomPhoto) {
        $uniqueStr = substr(base_convert(sha1(uniqid(mt_rand())), 16, 36), 0, 9);
        $motherImageName = $request->name . "'smother-" . $uniqueStr . '-' . time() . '.' . explode('/', explode(':', substr($request-
        >mother_photo, 0, strpos($request->mother_photo, ';')))[1])[1];

        Image::make($request->mother_photo)->save(public_path('images/parents/').$motherImageName);
        $request->merge(['mother_photo' => $motherImageName]);

        $oldMomPhoto = public_path()."/images/parents/".$currentMomPhoto;
        if(file_exists($oldMomPhoto)){
            unlink($oldMomPhoto);
        }
    }

    $currentguardPhoto = $student->guardian_photo;
    if($request->guardian_photo != $currentguardPhoto) {
        $uniqueStr = substr(base_convert(sha1(uniqid(mt_rand())), 16, 36), 0, 9);
        $guardianImageName = $request->name . "'sguardian-" . $uniqueStr . '-' . time() . '.' . explode('/', explode(':', substr($request-
        >guardian_photo, 0, strpos($request->guardian_photo, ';')))[1])[1];

        Image::make($request->guardian_photo)->save(public_path('images/parents/').$guardianImageName);
        $request->merge(['guardian_photo' => $guardianImageName]);

        $oldguardPhoto = public_path()."/images/parents/".$currentguardPhoto;
        if(file_exists($oldguardPhoto)){
            unlink($oldguardPhoto);
        }
```

```
        if(!empty($request->sibling_id)) {
            $siblings = Student::where('sibling_id', $request->sibling_id)->get();
            foreach ($siblings as $sibling) {
                $sibling->father_photo = $request->father_photo;
                $sibling->mother_photo = $request->mother_photo;
                $sibling->guardian_photo = $request->guardian_photo;
                $sibling->yearly_income = $request->yearly_income;
                $sibling->address = $request->address;
                $sibling->father_name = $request->father_name;
                $sibling->father_phone = $request->father_phone;
                $sibling->father_occupation = $request->father_occupation;
                $sibling->mother_name = $request->mother_name;
                $sibling->mother_phone = $request->mother_phone;
                $sibling->mother_occupation = $request->mother_occupation;
                $sibling->guardian_name = $request->guardian_name;
                $sibling->guardian_phone = $request->guardian_phone;
                $sibling->guardian_occupation = $request->guardian_occupation;
                $sibling->guardian_address = $request->guardian_address;
                $sibling->guardian_relation = $request->guardian_relation;
                $sibling->save();
            }
        }


        if(!empty($request->password)) {
            $request->merge(['password' => Hash::make($request->password)]);
        }
        $request->merge(['updated_at' => Carbon::now()->toDateTimeString()]);
        $request->merge(['role_id' => 5]);

        $student->update($request->all());
        $user->update($request->all());
    }
```

```
    public function destroy($id)
    {
        $student = Student::findOrFail($id);
        User::where('email', $student->email)->delete();
        $student->delete();
    }

    public function getRoll($cls, $sec)
    {
        $students = Student::where('class', $cls)->where('section', $sec)->get();
        $count = count($students);
        $roll = str_pad(++$count, 3, '0', STR_PAD_LEFT);
        return response()->json([

            'roll' => $roll,
            'totalAdmitted' => --$count

        ], Response::HTTP_OK);
    }

    public function getStudents($cls, $sec)
    {
        $students = Student::where('class', $cls)->where('section', $sec)->orderBy('roll', 'asc')->get();
        return response()->json([

            'students' => $students

        ], Response::HTTP_OK);
    }
}
```

Figure 21:Admission & Sibling management Module

- **Live Attendance Module**

```
    public function store(Request $request)
    {
        $this->validate($request, [
            'date' => 'required',
            'class' => 'required',
            'section' => 'required'
        ]);

        foreach ($request->attendance as $attend) {
            if(!empty($attend['id'])) {
                $att = new Attendance();
                $att->student_id = $attend['id'];
                $att->class = $request->class;
                $att->section = $request->section;
                $att->attend = $attend['type'];
                $att->note = $attend['note'];
                $att->date = Carbon::create($request->date)->toDateString();
                $att->created_at = Carbon::now()->toDateTimeString();
                $att->save();
            }
        }
    }

    public function update(Request $request)
    {
        $this->validate($request, [
            'date' => 'required',
            'class' => 'required',
            'section' => 'required'
        ]);

        foreach ($request->attendance as $attend) {
            if(!empty($attend['id'])) {
                $att = Attendance::find($attend['id']);
                $att->class = $request->class;
                $att->section = $request->section;
                $att->attend = $attend['type'];
                $att->note = $attend['note'];
                $att->date = $request->date;
                $att->updated_at = Carbon::now()->toDateTimeString();
                $att->save();
            }
        }
```

```
    public function isAlreadyRegistered($cls, $sec, $date)
    {
        $attendance = Attendance::where('class', $cls)->where('section', $sec)->where('date', Carbon::create($date)->toDateString())->get();
        if(count($attendance) == 0){
            $showForm = true;
        } else {
            $showForm = false;
        }
        return response()->json([

            'showForm' => $showForm,
            'prevAttedance' => $attendance

        ], Response::HTTP_OK);
    }
}
```

Figure 22:Live attendance Module

- **Class schedule Module:**

```
    public function store(Request $request)
    {
        $this->validate($request, [
            'class' => 'required',
            'section' => 'required',
            'day_id' => 'required',
            'subject' => 'required',
            'teacher_id' => 'required',
            'start' => 'required',
            'end' => 'required',
            'room' => 'required'
        ]);
        $create = ClassSchedule::create($request->all());
    }

    /**
     * Display the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function show($id)
    {
        //
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, $id)
    {
        $this->validate($request, [
            'class' => 'required',
            'section' => 'required',
            'day_id' => 'required',
            'subject' => 'required',
            'teacher_id' => 'required',
            'start' => 'required',
            'end' => 'required',
        ]);
        $create = ClassSchedule::findOrFail($id)->update($request->all());
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
        $delete = ClassSchedule::findOrFail($id)->delete();
    }

    public function getSubject($cls)
    {
        return response()->json([
            'subjects' => SubjectGroup::where('class', $cls)->first()->subject,
        ], Response::HTTP_OK);
    }

    public function getSchedule($cls, $sec)
    {
        return response()->json([
            'schedules' => ClassSchedule::where('class', $cls)->where('section', $sec)->get()
        ], Response::HTTP_OK);
    }

    public function getScheduleByTeacher($teacher)
    {
        return response()->json([
            'schedules' => ClassSchedule::where('teacher_id', $teacher)->get()
        ], Response::HTTP_OK);
    }
}
```

Figure 23:Class schedule Module code

- **Grading Module:**

```php
public function store(Request $request)
{
    $cate_id = $request->cate_id;
    $this->validate($request, [
        'cate_id' => 'required|integer',
        'grade' => ['required', Rule::unique('grades')->where(function ($query) use ($cate_id) {
            return $query->where('cate_id', $cate_id);
        })],
        'mark_from' => ['required', 'integer', 'max:100', new UniqueMark($cate_id)],
        'mark_to' => ['required', 'integer', 'max:100', 'greater_than_field:mark_from', new UniqueMark($cate_id)],
        'gpa' => ['required', 'numeric', 'between:0.00,5.00', Rule::unique('grades')->where(function ($query) use ($cate_id) {
            return $query->where('cate_id', $cate_id);
        })],
        // 'gpa' => 'required|numeric|between:0.00,5.00'
    ],[
        'gpa.numeric' => 'GPA should be numeric',
        'gpa.between' => 'GPA should be between 0.00 to 5.00',
        'mark_to.greater_than_field' => 'This value must be greater than mark percentage from'
    ]);
    $request->merge(['grade' => Str::upper($request->grade)]);
    $create = Grade::create($request->all());
}
```

```php
public function update(Request $request, $id)
{
    $cate_id = $request->cate_id;
    $this->validate($request, [
        'cate_id' => 'required|integer',
        'grade' => ['required', Rule::unique('grades')->ignore($id)->where(function ($query) use ($cate_id) {
            return $query->where('cate_id', $cate_id);
        })],
        'mark_from' => ['required', 'integer', 'max:100', new UniqueMarkUpdate($cate_id, $id)],
        'mark_to' => ['required', 'integer', 'max:100', 'greater_than_field:mark_from', new UniqueMarkUpdate($cate_id, $id)],
        'gpa' => ['required', 'numeric', 'between:0.00,5.00', Rule::unique('grades')->ignore($id)->where(function ($query) use ($cate_id) {
            return $query->where('cate_id', $cate_id);
        })]
        // 'gpa' => 'required|numeric|between:0.00,5.00'
    ],[
        'gpa.numeric' => 'GPA should be numeric',
        'gpa.between' => 'GPA should be between 0.00 to 5.00',
    ]);

    $update = Grade::findOrFail($id)->update($request->all());
}
```

```php
public function category()
{
    if(\Gate::allows('isSuperAdmin') || \Gate::allows('isAdmin') || \Gate::allows('isTeacher'))  {
        return response()->json([

            'gradingCategory' => GradingCategory::get()

        ], Response::HTTP_OK);
    }
}

public function categoryCreate(Request $request)
{
    $this->validate($request,[
        'category_name' => 'required',
        'opt_gpa_less' => 'required|numeric|between:0.00,5.00'
    ],[
        'opt_gpa_less.between' => 'This value should be between 0.00 and 5.00'
    ]);

    $create = GradingCategory::create($request->all());
}

public function categoryUpdate(Request $request, $id)
{
    $this->validate($request,[
        'category_name' => 'required',
        'opt_gpa_less' => 'required|numeric|between:0.00,5.00'
    ],[
        'opt_gpa_less.between' => 'This value should be between 0.00 and 5.00'
    ]);

    $create = GradingCategory::findOrFail($id)->update($request->all());
}

public function categoryDestroy($id)
{
    $delete = GradingCategory::findOrFail($id)->delete();

    $grades = Grade::where('cate_id', $id)->get();
    foreach ($grades as $grade) {
        $grade->delete();
    }
}
```

Figure 24:Grading Module Code

- **Result Module:**

```
    public function examResult($cls, $sec, $exam_type, $stu_id)
    {
        $year = Carbon::now()->year;
        $exam = Exam::where('class', $cls)->where('section', $sec)->where('exam_type', $exam_type)->where('year', $year)->first();
        if($exam != '') {
            $result = Result::where('stu_id', $stu_id)->where('exam_id', $exam->id)->first();
            if($result) {
                $subjects = SubjectGroup::where('class', $cls)->first();
                $grades = Grade::where('cate_id', $exam->grading)->orderBy('gpa', 'asc')->get();

                $subject_array = explode(',', $subjects->subject);

                $sql = \DB::table('subjects');
                foreach ($subject_array as $select) {
                   $sql->orWhere('subject', 'LIKE', $select);
                }
                $subjects_data = $sql->get();

                $grading = GradingCategory::where('id', $exam->grading)->first();

                return response()->json([
                    'exam' => $exam,
                    'result' => $result,
                    'grades' => $grades,
                    'grading' => $grading,
                    'subjects_data' => $subjects_data,
                ], Response::HTTP_OK);
            } else {
                return response()->json([
                    'exam' => $exam,
                    'result' => $result
                ], Response::HTTP_OK);
            }
        }
        return response()->json([
            'exam' => $exam
        ], Response::HTTP_OK);
    }
}
```

Figure 25:Result Module Code

➢ Possible problem breakdown

I have studied some possible common problems and created breakdown so that I don't have to face them while building my system. The possible problems could be:

- Data redundancy can happen.
- Lack of error handling
- Requirement analysis lack
- Lack of testing
- User acceptance lacking

To avoid these possible problems, I have taken some steps like:

- Created data structure avoiding data redundancy.
- Covered all possible error handling

- Created requirement analysis and verified from real user
- Tested using multiple methods
- User acceptance survey done from real users

All these possible problems are solved.

Chapter 11

Testing

Test Case

➢ Unit Testing

- **Class Creation**

```
public function update(Request $request, $id)
{
    $this->validate($request, [
        'class' => 'required|unique:classes,class,'.$id,
        'section' => 'required'
    ]);
    $sections = implode(',', array_values($request->section));
    $seats = implode(',', array_values($request->seats));

    $request->merge(['section' => $sections]);
    $request->merge(['seats' => $seats]);

    $update = Classes::findOrFail($id)->update($request->all());
    return ['message' => 'Updated Successfully'];

}

/**
 * Remove the specified resource from storage.
 *
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function destroy($id)
{
    $delete = Classes::findOrFail($id)->delete();
    return ['message' => 'Deleted Successfully'];
}
```

Figure 26:Class Module code

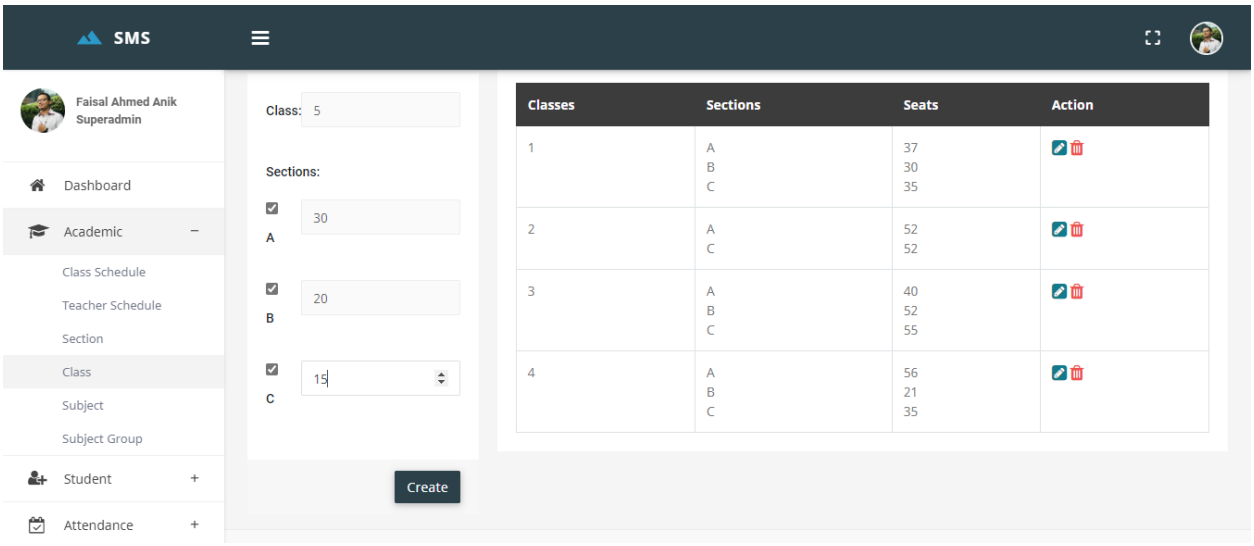


Figure 27:Given Class details in fields

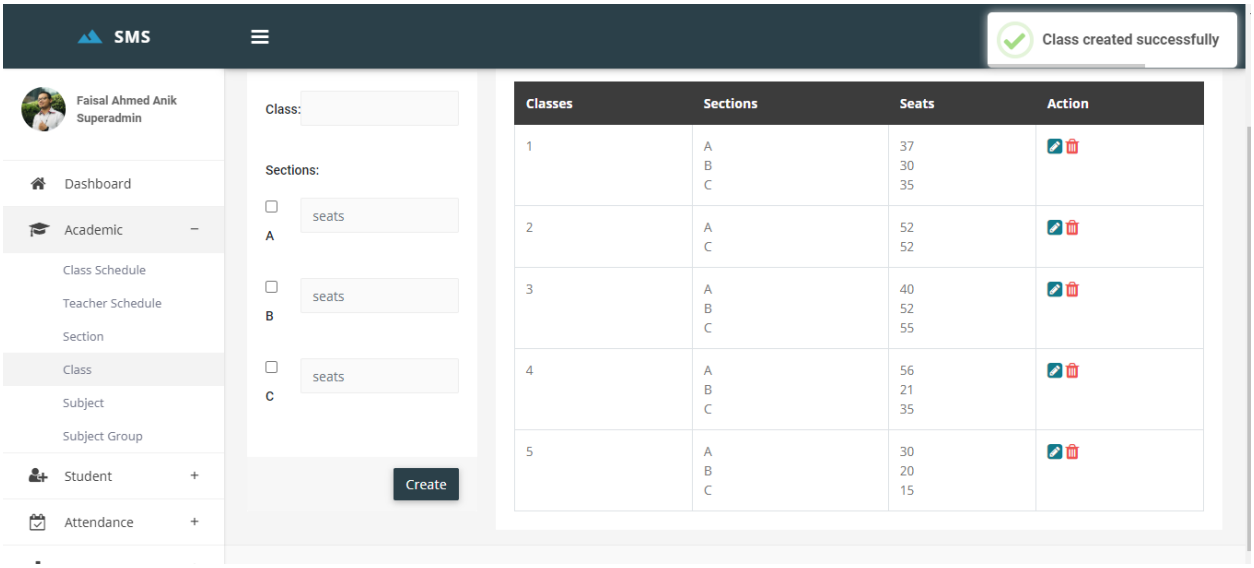


Figure 28:Class module working properly

- **Admission**

```php
if($request->photo) {
    $uniqueStr = substr(base_convert(sha1(uniqid(mt_rand())), 16, 36), 0, 9);
    $imageName = $request->name. '-' . $uniqueStr . '-' . time() . '.' . explode('/', explode(':', substr($request->photo, 0,
    strpos($request->photo, ';')))[1])[1];

    Image::make($request->photo)->save(public_path('images/students/').$imageName);
} else {
    $imageName = 'avatar.jpg';
}
$request->merge(['photo' => $imageName]);

if($request->sibling_id == ''){
    if($request->father_photo) {
        $uniqueStr = substr(base_convert(sha1(uniqid(mt_rand())), 16, 36), 0, 9);
        $fatherImageName = $request->name . "'sfather-" . $uniqueStr . '-' . time() . '.' . explode('/', explode(':', substr($request-
        >father_photo, 0, strpos($request->father_photo, ';')))[1])[1];

        Image::make($request->father_photo)->save(public_path('images/parents/').$fatherImageName);
    } else {
        $fatherImageName = 'avatar.jpg';
    }
    $request->merge(['father_photo' => $fatherImageName]);

    if($request->mother_photo) {
        $uniqueStr = substr(base_convert(sha1(uniqid(mt_rand())), 16, 36), 0, 9);
        $motherImageName = $request->name . "'smother-" . $uniqueStr . '-' . time() . '.' . explode('/', explode(':', substr($request-
        >mother_photo, 0, strpos($request->mother_photo, ';')))[1])[1];

        Image::make($request->mother_photo)->save(public_path('images/parents/').$motherImageName);
    } else {
        $motherImageName = 'avatar.jpg';
    }
    $request->merge(['mother_photo' => $motherImageName]);

    if($request->guardian_photo) {
        $uniqueStr = substr(base_convert(sha1(uniqid(mt_rand())), 16, 36), 0, 9);
        $guardianImageName = $request->name . "'sguardian-" . $uniqueStr . '-' . time() . '.' . explode('/', explode(':',
        substr($request->guardian_photo, 0, strpos($request->guardian_photo, ';')))[1])[1];

        Image::make($request->guardian_photo)->save(public_path('images/parents/').$guardianImageName);
    } else {
        $guardianImageName = 'avatar.jpg';
```

Figure 29:Admission code

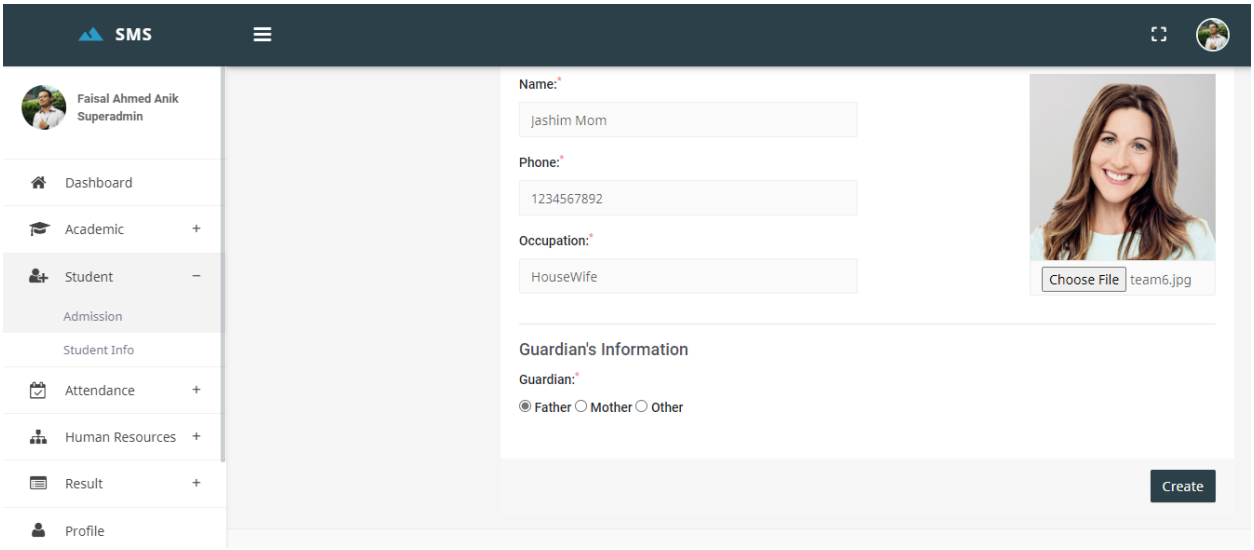


Figure 30:Admission form filled

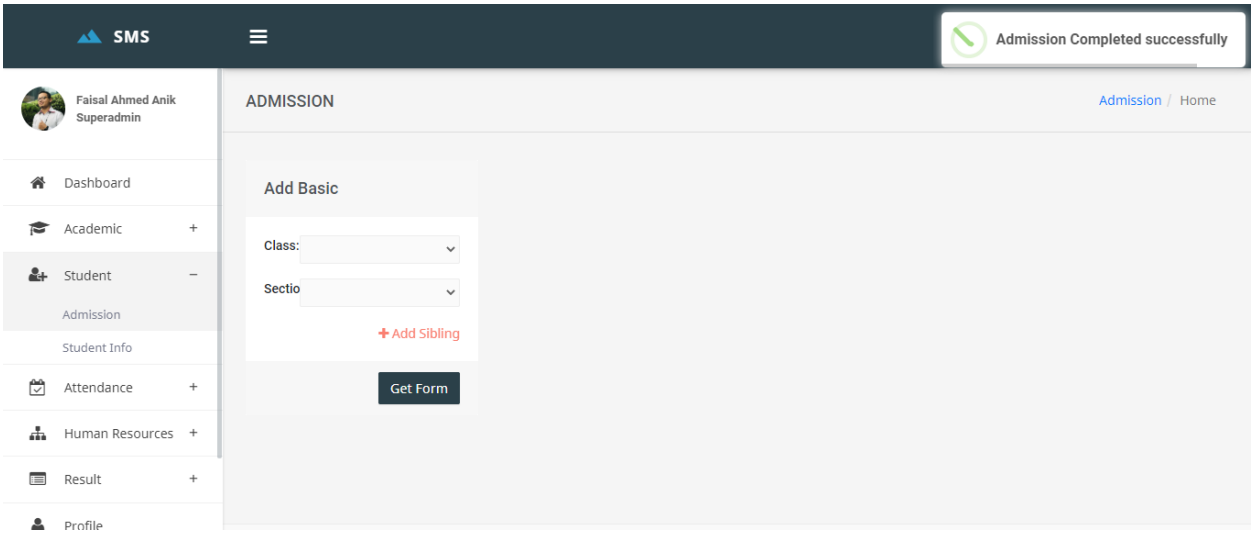


Figure 31:Admission Working properly

➢ Module Testing

Test Case No.		Module testing 1		
Test Class		Login		
Objectives		Login with empty fields		
Data Source	Task Description	Task Steps	Expected Result	Actual Result
User Entry	Login with blank field.	Keep username and password field empty and press login button.	Show Error	As expected.

Actual Result

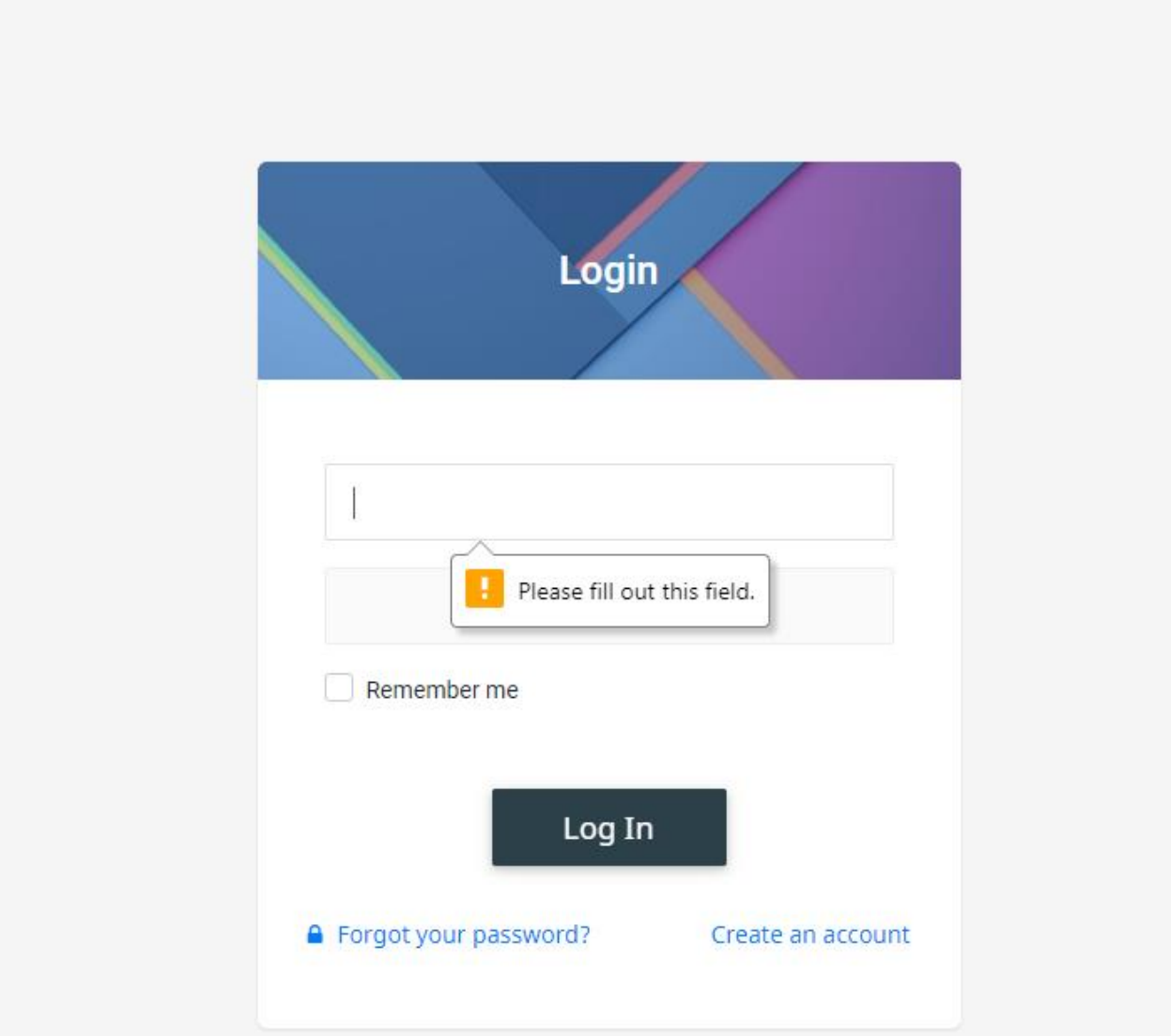


Figure 32:Blank Login attempt

Test Case No.		**Module testing 2**		
Test Class		Login		
Objectives		Login with correct data		
Data Source	Task Description	Task Steps	Expected Result	Actual Result

User Entry	Login with correct data.	Data: Username: faisalanik2@gmail.com Password: 12345678	Login successful	As expected.

Actual Result:

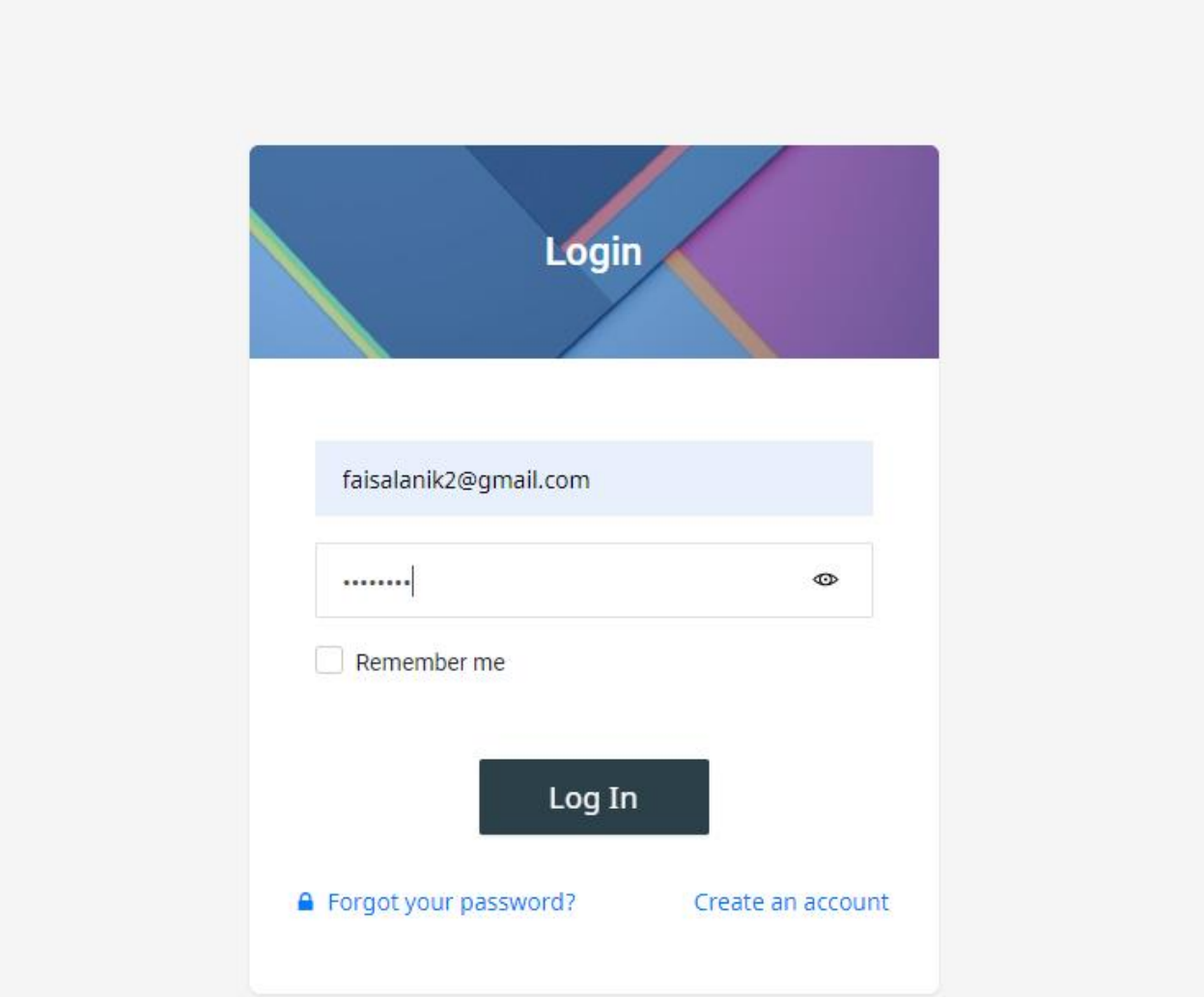


Figure 33:Fill login form with correct data

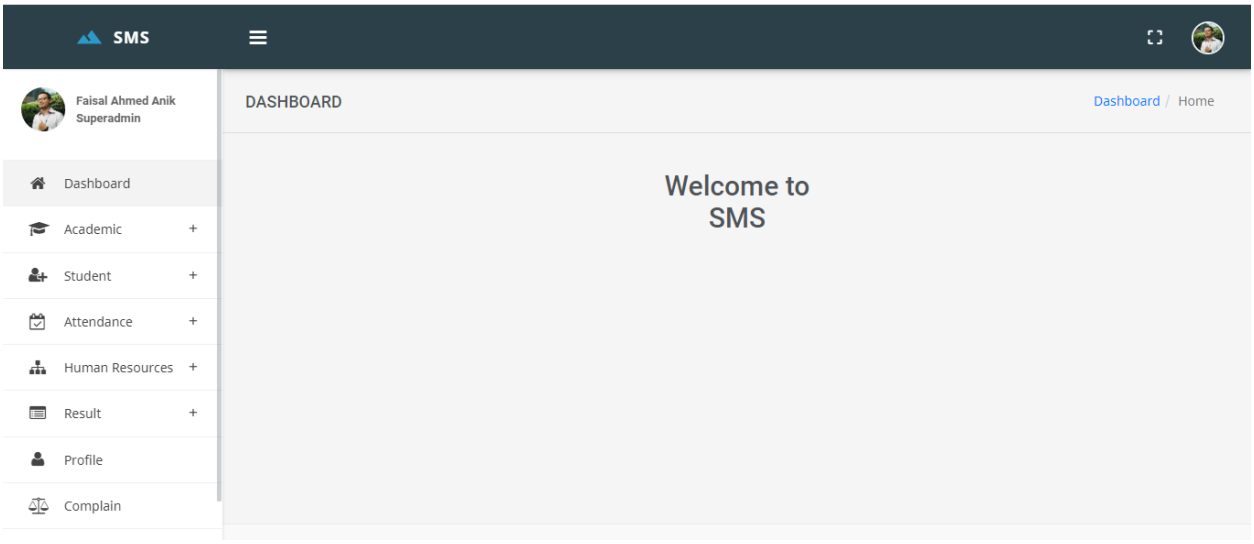


Figure 34:Login Successful

➢ Integration Testing

- **Login**

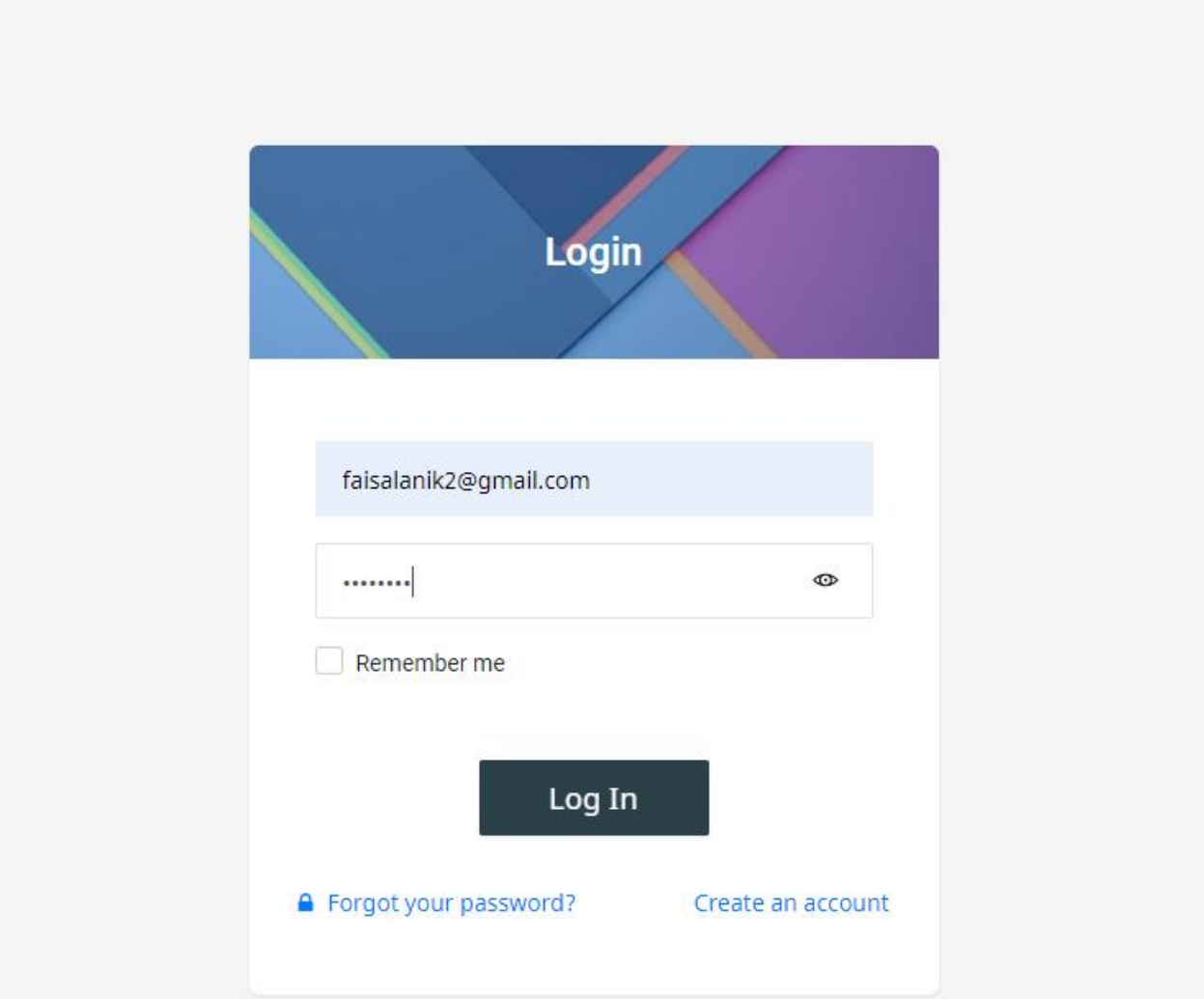


Figure 35:Login test with correct data

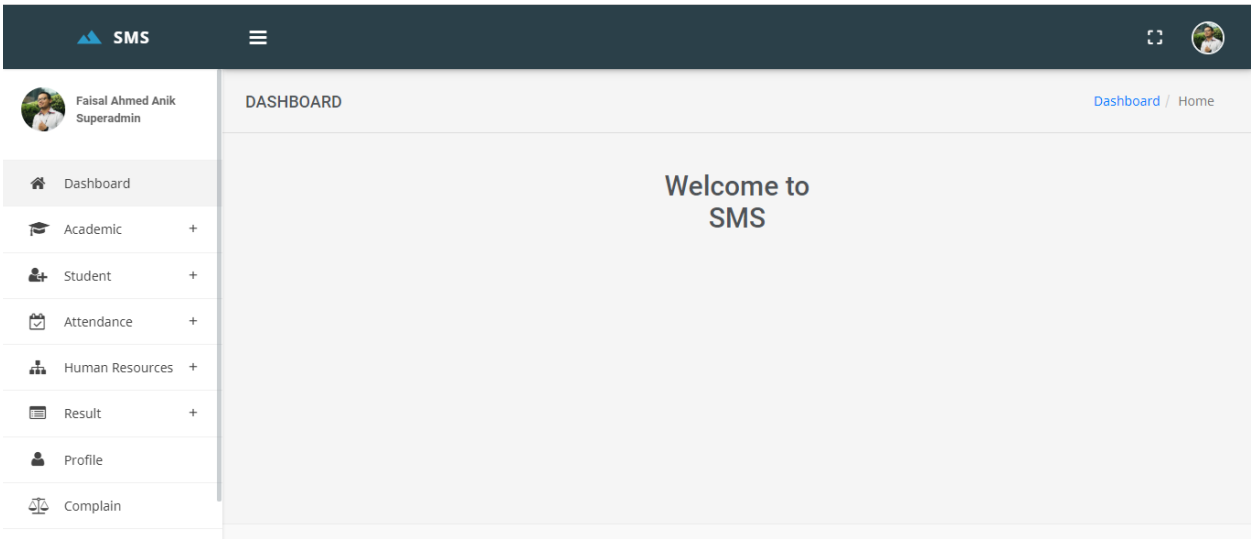


Figure 36:Login successful with correct data

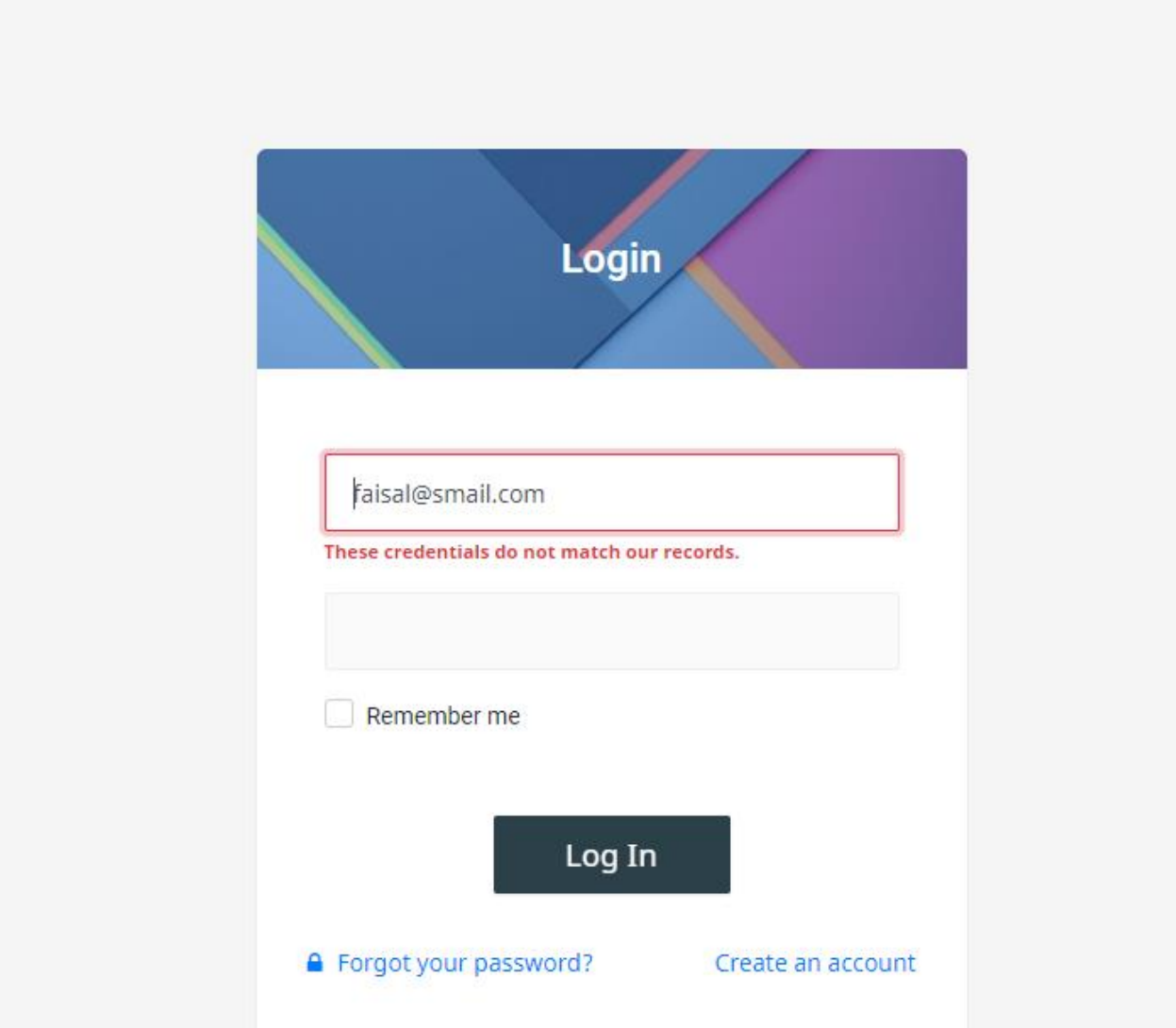


Figure 37:Login failed with incorrect data

- **User Create:**

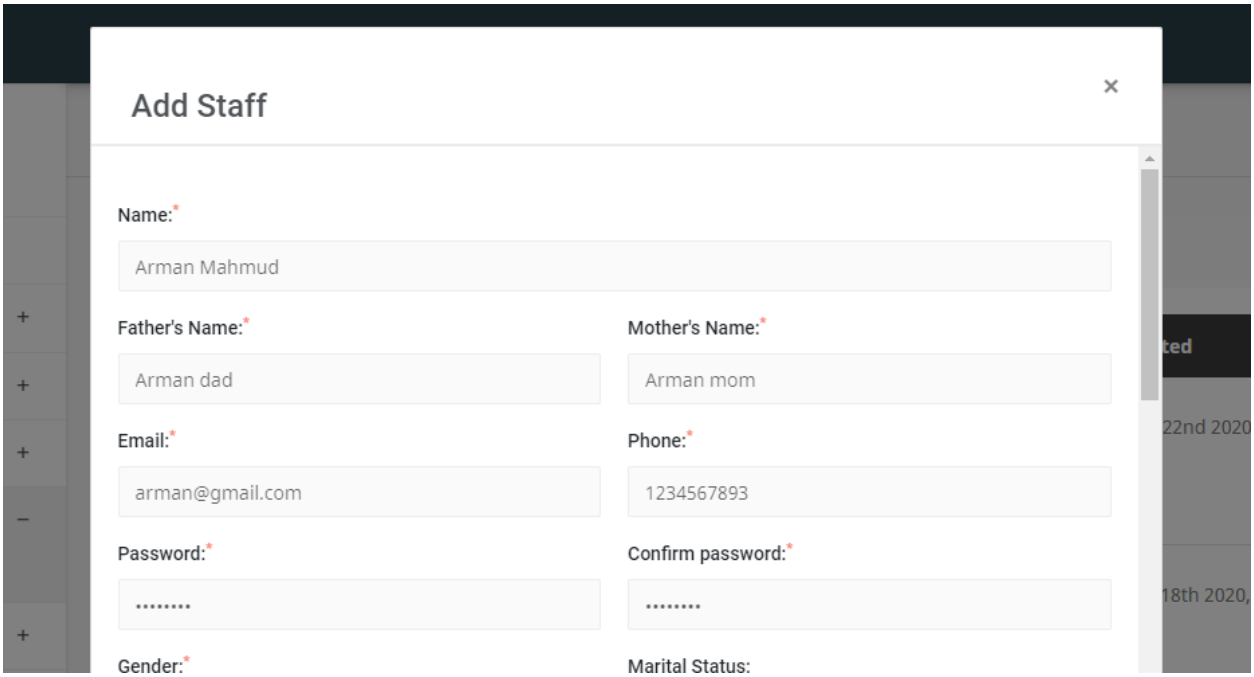


Figure 38:User create test with data

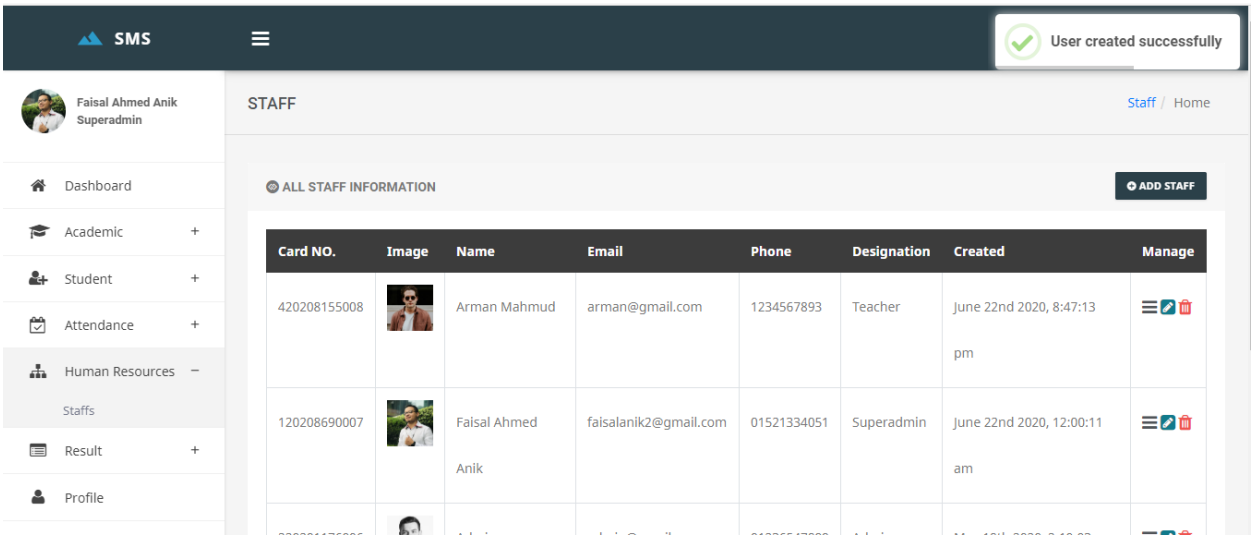


Figure 39:User created successfully with data

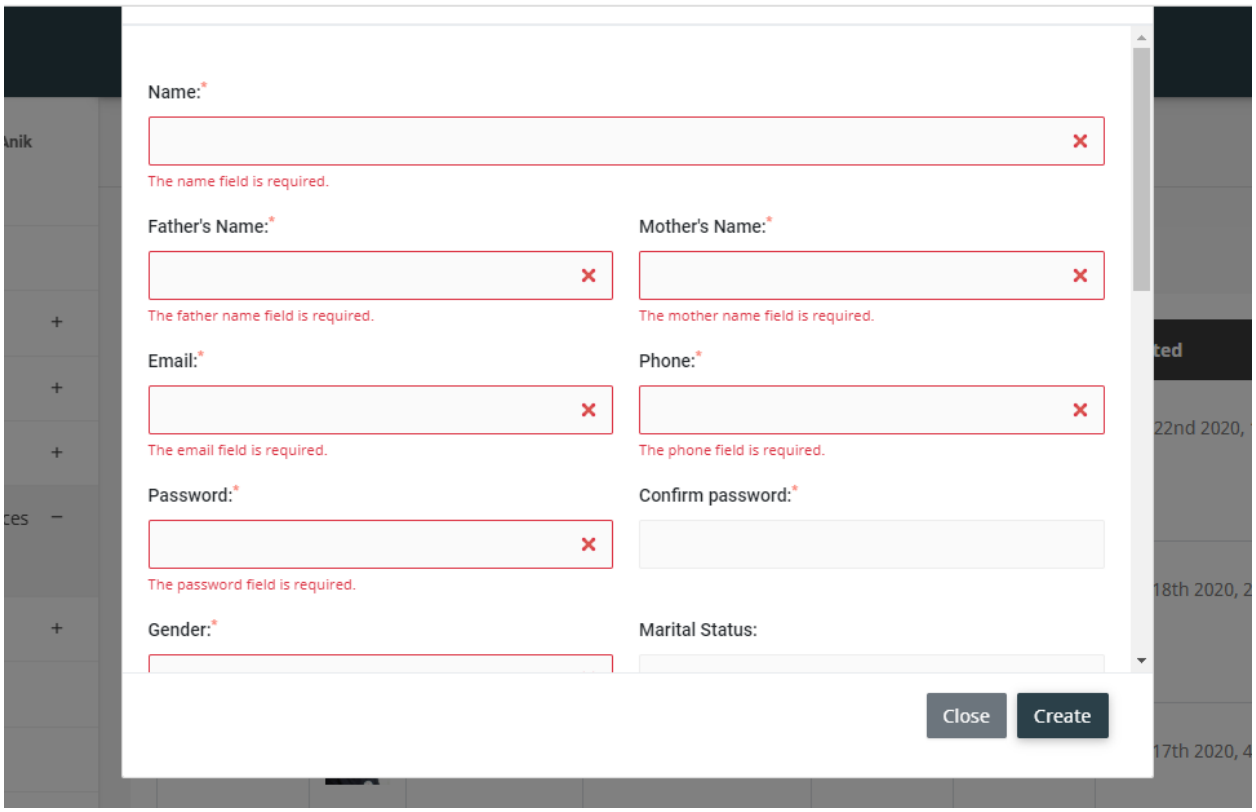


Figure 40:User creation failed with incorrect data

Chapter 12

Implementation

➢ Training

Software's are made to reduce pain of working, automate maximum things. A good system must be user friendly so that users of that system can easily run the system. But whether being user friendly, because of lack of knowledge in IT sector some users may get trouble using the system. That's why there should be a training phase of any big system.

My system is fully user friendly. But I will also run training phase in the school where I will sell it.

I need to train a bunch of representatives of school who can train others in the school. I will demonstrate whole system functionality and workability to them practically in a demo server area. After my demonstration I will see if they can do that perfectly or not. After teaching them the training phase can be ended successfully.

- **Big Bang (no pilot, parallel implementation scheme)**

The Big Bang model of software development is designed around the notion that, beginning with nothing, a rapid growth and expansion of code will quickly emerge, thus producing a finished product in a mere instant. (airbrake, 2020)

Big bang is a method where a system can adopt previous system data automatically. It is like updating versions where data doesn't loose but the system gets updated. This feature making is a time-consuming fact. I did not include this feature in my system. User must input old system data manually. But I have plans to bring this feature inside my system in future.

Chapter 13

Critical Appraisal and Evaluation

➢ Objective that could be met

No system in the world is fully complete. There is always a scope of updating. I have tried to fulfill every basic and needed objectives so that a school can run this system and use it for long. But there are some scopes which could be met. Those are:

- User interface for school frontend for global view.
- CMS for the user interface.
- Different statistical representation of data.
- Staff salary management.

These are some scopes which could be met but without them the system can be used properly because all major objectives are fulfilled.

• Success rate against each objective

Success rate for the whole system and all objectives are good. All related data are taken and showed in a better way so that user can easily operate. Student, parent and teacher portals contains all information regarding them and all errors are properly handled.

• How much better could have been done

The system is already built using Laravel and Vue.js which made the system updated and faster than any other similar systems. But it would be better if I could include library management feature in the system.

- **How better is the features of the solution**

I created the system to manage schools. Nowadays all school management system are built with old technology and slow. I used Laravel and vue.js that's why it became faster and secure than any other competitor in this sector. All features are well built and handled.

- **Which features could not be touched**

This system has all the required major features. All the features are well handled, secure, user friendly and faster. The system is easy and simple. But Library management system, multiple statistical data view and online payment system could not be touched in the system.

- **Why these features could not be touched**

I could not build those features because I was unable to research and gather knowledge about those because of COVID-19 pandemic situations. My knowledge was limited in these fields and for the situation I couldn't touch those features.

➢ Objectives totally not met / touched

The system contains all the needed features for a school management system. But online transaction for admission is not touched in this system.

- **Why it could not be touched**

I did some field work about this feature. I found that our country guardians and schools are not so familiar with online transactions. They think this is a

less secure and more hassle type of thing. Many guardians are not well educated so that this feature will be unused in maximum cases. That's why I didn't create this feature.

- **What could have been done**

In the system mobile banking system could be introduced. So that guardians can make payment of tuition fees using their respective mobile banking system.

Chapter 14

Conclusion

➢ Summary of the project

I developed a system which is called "School Management System". This is a project which is for schools. In our country maximum of schools are not digitalized. The schools are still maintaining old method of keeping data manually. That's why I tried to figure out what need to be done for making the school management system which will be helpful for all the related users of any school. I studied on field and also with the help of internet I found out many features which need to be implemented to build this project. To make the schools digital and advanced this system has so many features for all the users who are related to this system.

For Students, they can view their subject's grades, contact with the teachers, attendance report or present percentage, and they also up to date with all school's news or posts that publish by the other users.

For Admin, they have a full control on the system, like they can add a new, teachers and students with their subjects, create sections, courses, assign subjects to the teachers, send email anyone regarding notices.

For Teachers, they can add student's grades or edit it for their own subjects only, and they can take attendance, see attendance reports, upload and download assignments of the related students

➢ Goal of the project

The main goal of making this project is to make the school management easier and faster. The school system has some users and work lists. My aim was to make sure all aspects are touched and made correctly inside the system. My system can manage maximum administration works, teacher and student works. It will stablish connection between student and teacher after the school also. All department interaction will be improved. Easy user interface and can be accessed from anywhere in the globe with connection of internet. The system is made responsive so that users can use it from any size of devices. All in all the concentration was fully on security, usability and efficiency. All these made the system unique of its kind.

➢ Success of the project

Project success has been historically defined as a project that meets its objectives under budget and under schedule. This evaluation criterion has

remained as the most common measure in many industries. (pm4dev, 2020)

I have created a system which contains all the required and needed features of schools. It can be successfully used in the schools respectively. This is the major success that I can provide this management system to the schools who needs it. I made a full management system for school successfully. This is the major success of the project.

➢ Value of the project

The term ‘project value’ represents the total project cost, which includes all costs such as construction, design, land acquisition etc. (breeam, 2020)

I have created a system which is built using Laravel and vue.js which are current technology. My system contains all major features which a school management system must have. The system works in Realtime so that it is faster than any other system at this sector. This is fully secure and errors are highly handled in this system. All these makes the system more valuable than other competitor systems of this kind.

➢ My experience

I gained huge knowledge while making this system. I learned about a whole school management system. The working procedure and functions. I gathered huge knowledge about Laravel and vue.js as the system is built upon these. I learned a lot from my teachers and supervisor. They have guided me to the right path. All in all this was a great learning experience for me.

References

agilebusiness, 2020. *agilebusiness.* [Online]
Available at: https://www.agilebusiness.org/page/whatisdsdm

airbrake, 2020. *airbrake.* [Online]
Available at: https://airbrake.io/blog/sdlc/big-bang-model

breeam, 2020. *breeam.* [Online]
Available at: https://kb.breeam.com/knowledgebase/project-value-definition/#:~:text=The%20term%20%27project%20value%27%20represents,%2C%20design%2C%20land%20acquisition%20etc.

pm4dev, 2020. *pm4dev.* [Online]
Available at: https://www.pm4dev.com/pm4dev-blog/entry/definition-of-project-success.html#:~:text=Project%20success%20has%20been%20historically,under%20budget%20and%20under%20schedule.&text=Efficiency%20is%20related%20to%20how,with%20internal%20and%20external%20stakeho

projectmanager, 2020. *projectmanager.* [Online]
Available at: https://www.projectmanager.com/blog/cost-benefit-analysis-for-projects-a-step-by-step-guide

Appendices

Test Scripts

Test Case No.		**Test script 1**		
Test Class		Login		
Objectives		Login with empty fields		
Data Source	Task Description	Task Steps	Expected Result	Actual Result
User Entry	Login with blank field.	Keep username and password field empty and press login button.	Show Error	As expected.

Test Case No.		**Test script 2**		
Test Class		Login		
Objectives		Login with correct data		
Data Source	Task Description	Task Steps	Expected Result	Actual Result
User Entry	Login with correct data.	Data: Username: faisalanik2@gmail.com Password: 12345678	Login successful	As expected.

User Guide

This system is built using Laravel and vue.js which has some prerequisites before running the system. There are 2 scenarios. One is live domain

server and another is local server. If the system is uploaded in the root of live domain server then it will automatically work after connecting with it's database. But while running in localhost, user need to run composer inside the folder of the system and write "php artisan serve" then press enter. A link will generate automatically which will perfectly access the system in browser.

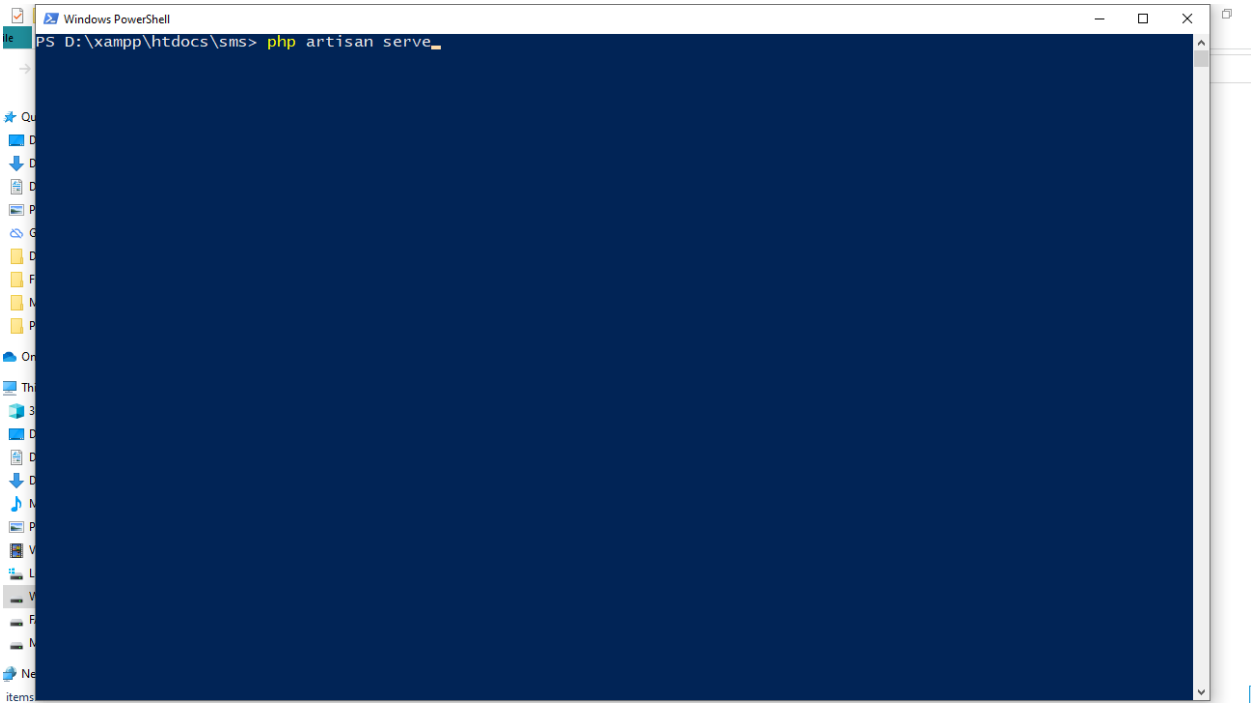


Figure 41:System link generation code for localhost

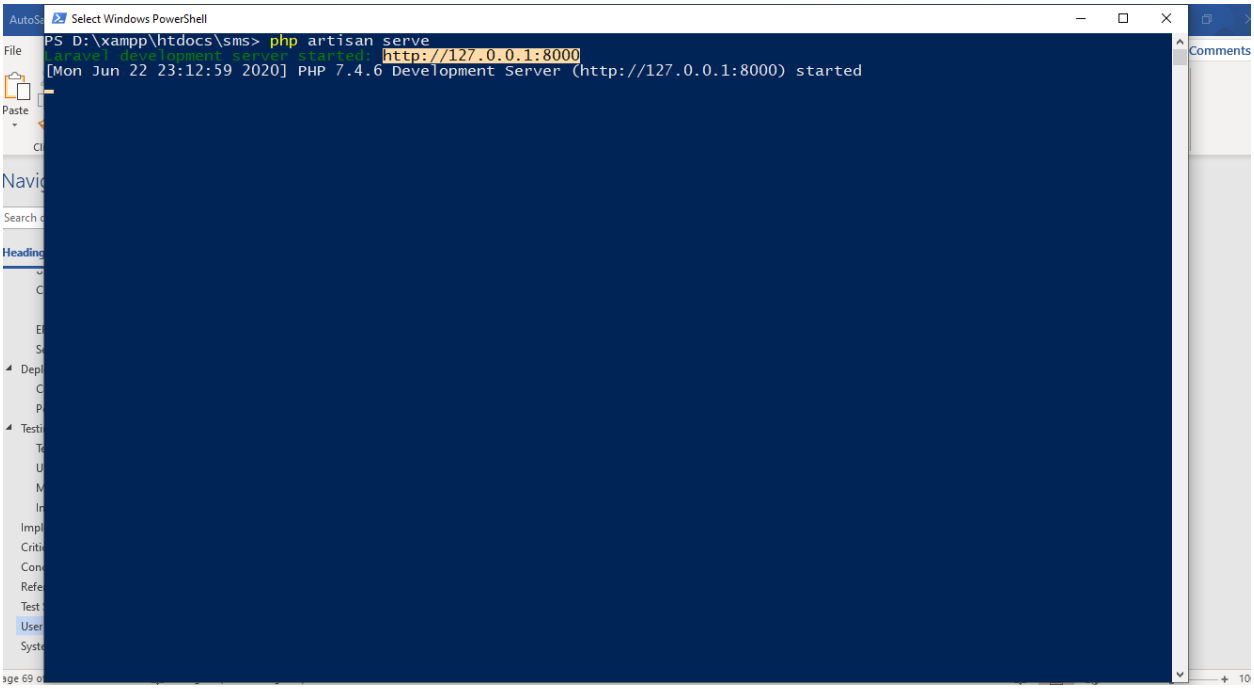


Figure 42:Successfully generated code

Admin Panel guide:

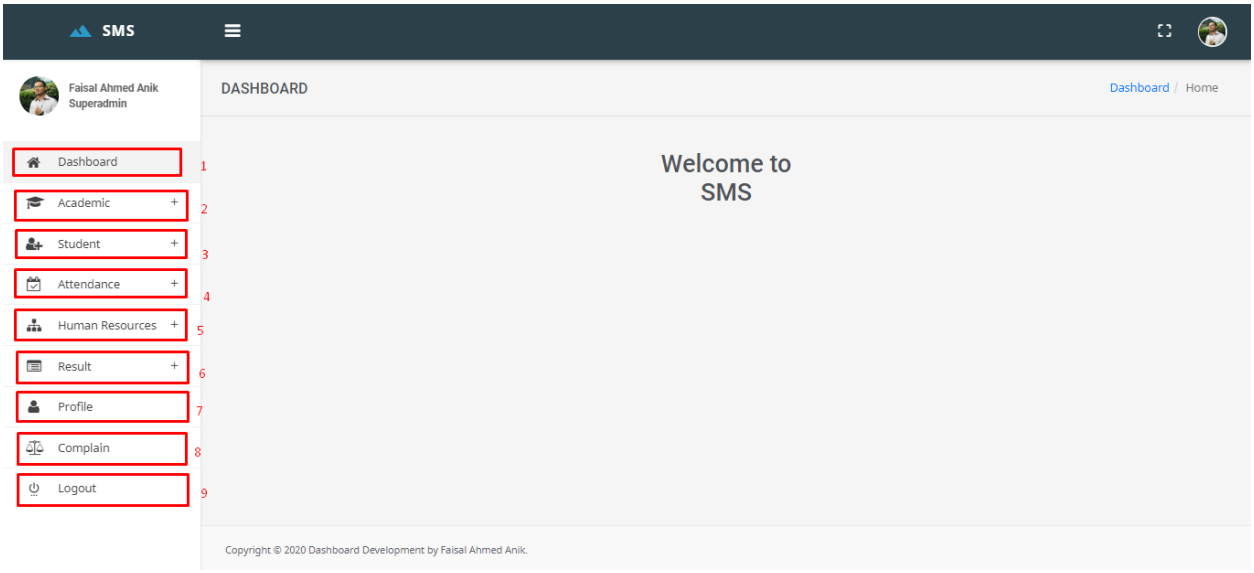


Figure 43:Admin Panel guide

1. This is the dashboard link which will take user directly to the home of portal.
2. Academic option contains all type of academic features like routine, class, section, subject management.

3. Student option contains admission and general information of students and siblings.
4. Attendance contains live attendance system and attendance history.
5. Human resource contains staff management, adding new roles.
6. Result contains multiple grading system and result.
7. Profile gives general view and edit option for user.
8. Complain contains all complains of students.
9. Logout will simply logout the user.

Teacher Panel guide:

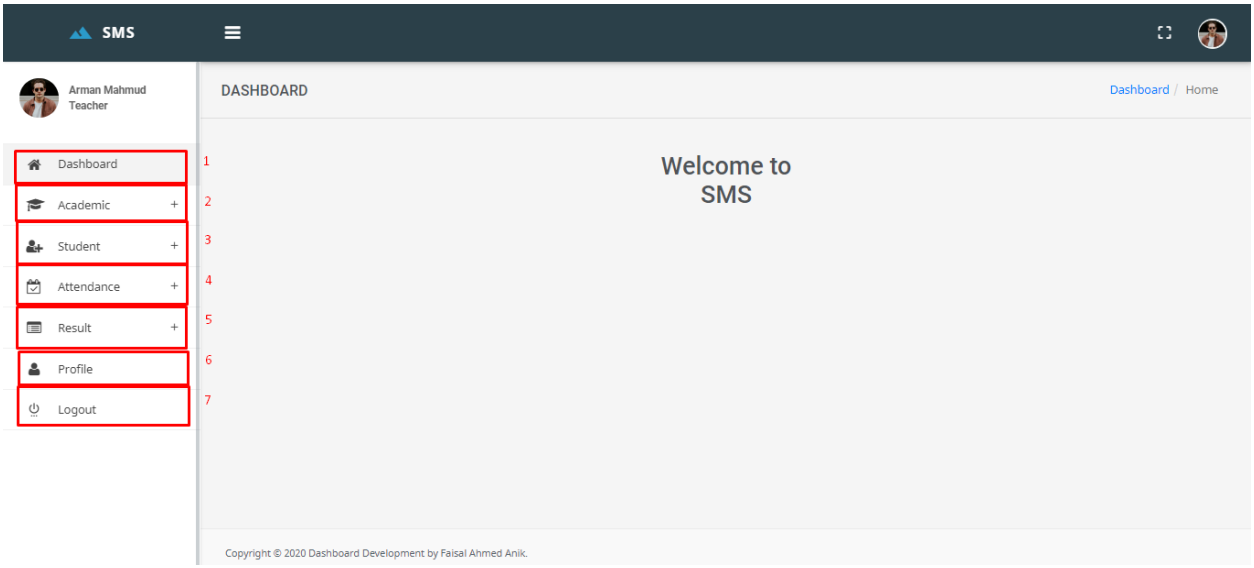


Figure 44:Teacher panel Guide

1. This is the dashboard link which will take user directly to the home of portal.
2. Academic option contains all type of academic features like routine, class, section, subject management.
3. Student option contains admission and general information of students and siblings. Here teacher also can generate complain of any student.
4. Attendance contains live attendance system and attendance history.
5. Result contains multiple grading system and result.
6. Profile gives general view and edit option for user.
7. Logout will simply logout the user.

Student & Parent Panel Guide:

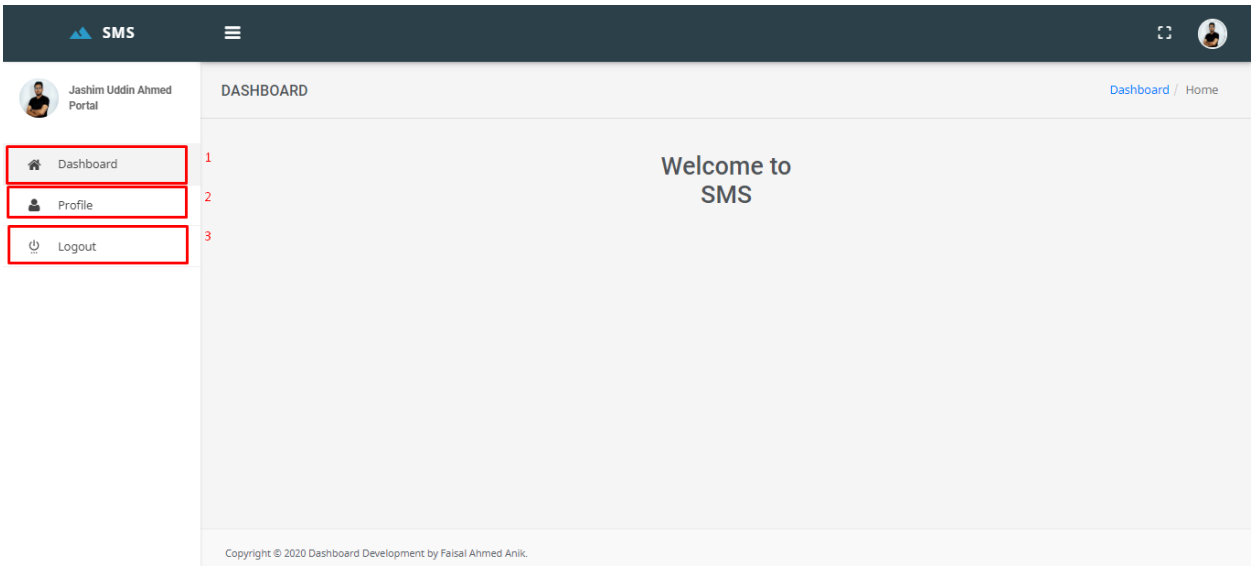


Figure 45:Student & Parent panel Guide

1. This is the dashboard link which will take user directly to the home of portal.
2. Profile option contains all features of student portal like attendance history, result, complain history, Id card etc.
3. Logout will simply logout the user.

System Code

```
'connections' => [

    'sqlite' => [
        'driver' => 'sqlite',
        'url' => env('DATABASE_URL'),
        'database' => env('DB_DATABASE', database_path('database.sqlite')),
        'prefix' => '',
        'foreign_key_constraints' => env('DB_FOREIGN_KEYS', true),
    ],

    'mysql' => [
        'driver' => 'mysql',
        'url' => env('DATABASE_URL'),
        'host' => env('DB_HOST', '127.0.0.1'),
        'port' => env('DB_PORT', '3306'),
        'database' => env('DB_DATABASE', 'forge'),
        'username' => env('DB_USERNAME', 'forge'),
        'password' => env('DB_PASSWORD', ''),
        'unix_socket' => env('DB_SOCKET', ''),
        'charset' => 'utf8mb4',
        'collation' => 'utf8mb4_unicode_ci',
        'prefix' => '',
        'prefix_indexes' => true,
        'strict' => true,
        'engine' => null,
        'options' => extension_loaded('pdo_mysql') ? array_filter([
            PDO::MYSQL_ATTR_SSL_CA => env('MYSQL_ATTR_SSL_CA'),
        ]) : [],
    ],

    'pgsql' => [
        'driver' => 'pgsql',
        'url' => env('DATABASE_URL'),
        'host' => env('DB_HOST', '127.0.0.1'),
        'port' => env('DB_PORT', '5432'),
        'database' => env('DB_DATABASE', 'forge'),
        'username' => env('DB_USERNAME', 'forge'),
        'password' => env('DB_PASSWORD', ''),
        'charset' => 'utf8',
        'prefix' => '',
        'prefix_indexes' => true,
        'schema' => 'public',
        'sslmode' => 'prefer',
```

Figure 46:Database connection code

```php
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Foundation\Auth\AuthenticatesUsers;

class LoginController extends Controller
{
    /*
    |--------------------------------------------------------------------------
    | Login Controller
    |--------------------------------------------------------------------------
    |
    | This controller handles authenticating users for the application and
    | redirecting them to your home screen. The controller uses a trait
    | to conveniently provide its functionality to your applications.
    |
    */

    use AuthenticatesUsers;

    /**
     * Where to redirect users after login.
     *
     * @var string
     */
    protected $redirectTo = '/home';

    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('guest')->except('logout');
    }
}
```

Figure 47:Login Code

```
    if($request->photo) {
        $uniqueStr = substr(base_convert(sha1(uniqid(mt_rand())), 16, 36), 0, 9);
        $imageName = $request->name. '-' . $uniqueStr . '-' . time() . '.' . explode('/', explode(':', substr($request->photo, 0,
        strpos($request->photo, ';')))[1])[1];

        Image::make($request->photo)->save(public_path('images/students/').$imageName);
    } else {
        $imageName = 'avatar.jpg';
    }
    $request->merge(['photo' => $imageName]);

    if($request->sibling_id == ''){
        if($request->father_photo) {
            $uniqueStr = substr(base_convert(sha1(uniqid(mt_rand())), 16, 36), 0, 9);
            $fatherImageName = $request->name . "'sfather-" . $uniqueStr . '-' . time() . '.' . explode('/', explode(':', substr($request-
            >father_photo, 0, strpos($request->father_photo, ';')))[1])[1];

            Image::make($request->father_photo)->save(public_path('images/parents/').$fatherImageName);
        } else {
            $fatherImageName = 'avatar.jpg';
        }
        $request->merge(['father_photo' => $fatherImageName]);

        if($request->mother_photo) {
            $uniqueStr = substr(base_convert(sha1(uniqid(mt_rand())), 16, 36), 0, 9);
            $motherImageName = $request->name . "'smother-" . $uniqueStr . '-' . time() . '.' . explode('/', explode(':', substr($request-
            >mother_photo, 0, strpos($request->mother_photo, ';')))[1])[1];

            Image::make($request->mother_photo)->save(public_path('images/parents/').$motherImageName);
        } else {
            $motherImageName = 'avatar.jpg';
        }
        $request->merge(['mother_photo' => $motherImageName]);

        if($request->guardian_photo) {
            $uniqueStr = substr(base_convert(sha1(uniqid(mt_rand())), 16, 36), 0, 9);
            $guardianImageName = $request->name . "'sguardian-" . $uniqueStr . '-' . time() . '.' . explode('/', explode(':',
            substr($request->guardian_photo, 0, strpos($request->guardian_photo, ';')))[1])[1];

            Image::make($request->guardian_photo)->save(public_path('images/parents/').$guardianImageName);
        } else {
            $guardianImageName = 'avatar.jpg';
        }
```

Figure 48:Admission Code

```
        foreach ($request->attendance as $attend) {
            if(!empty($attend['id'])) {
                $att = new Attendance();
                $att->student_id = $attend['id'];
                $att->class = $request->class;
                $att->section = $request->section;
                $att->attend = $attend['type'];
                $att->note = $attend['note'];
                $att->date = Carbon::create($request->date)->toDateString();
                $att->created_at = Carbon::now()->toDateTimeString();
                $att->save();
            }
        }
    }

    public function update(Request $request)
    {
        $this->validate($request, [
            'date' => 'required',
            'class' => 'required',
            'section' => 'required'
        ]);

        foreach ($request->attendance as $attend) {
            if(!empty($attend['id'])) {
                $att = Attendance::find($attend['id']);
                $att->class = $request->class;
                $att->section = $request->section;
                $att->attend = $attend['type'];
                $att->note = $attend['note'];
                $att->date = $request->date;
                $att->updated_at = Carbon::now()->toDateTimeString();
                $att->save();
            }
        }
    }

    public function isAlreadyRegistered($cls, $sec, $date)
    {
        $attendance = Attendance::where('class', $cls)->where('section', $sec)->where('date', Carbon::create($date)->toDateString())->get();
        if(count($attendance) == 0){
            $showForm = true;
        } else {
```

Figure 49:Attendance code

```
     */
    public function update(Request $request, $id)
    {
        $this->validate($request, [
            'class' => 'required',
            'section' => 'required',
            'day_id' => 'required',
            'subject' => 'required',
            'teacher_id' => 'required',
            'start' => 'required',
            'end' => 'required',
            'room' => 'required'
        ]);
        $create = ClassSchedule::findOrFail($id)->update($request->all());
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
        $delete = ClassSchedule::findOrFail($id)->delete();
    }

    public function getSubject($cls)
    {
        return response()->json([
            'subjects' => SubjectGroup::where('class', $cls)->first()->subject,
        ], Response::HTTP_OK);
    }

    public function getSchedule($cls, $sec)
    {
        return response()->json([
            'schedules' => ClassSchedule::where('class', $cls)->where('section', $sec)->get()
        ], Response::HTTP_OK);
    }

    public function getScheduleByTeacher($teacher)
    {
```

Figure 50:Routine code

```
    public function index()
    {
        if(\Gate::allows('isSuperAdmin') || \Gate::allows('isAdmin'))  {
            return response()->json([

                'complains' => Complain::latest()->paginate(20),
                'students' => Student::select('id','class', 'section', 'name', 'roll')->get(),
                'staffs' => Staff::select('email','name')->get()

            ], Response::HTTP_OK);
        }
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        if(\Gate::allows('isSuperAdmin') || \Gate::allows('isAdmin') || \Gate::allows('isTeacher')) {
            $this->validate($request,[
                'complain' => 'required|string'
            ]);

            $complain = new Complain;
            $complain->complain = $request->complain;
            $complain->stu_id = $request->id;
            $complain->complainer_email = Auth::user()->email;
            $role = Auth::user()->role_id;
            if($role == 1 || $role == 2) {
                $complain->status = 1;
            } else {
                $complain->status = 0;
            }
            $complain->created_at = Carbon::now()->toDateTimeString();
            $complain->save();
        }
```

Figure 51:Complain box code

```php
    public function index()
    {
        if(\Gate::allows('isSuperAdmin') || \Gate::allows('isAdmin') || \Gate::allows('isTeacher'))  {
            return response()->json([

                'gradingCategory' => GradingCategory::get(),
                'grades' => Grade::orderBy('gpa', 'asc')->get()

            ], Response::HTTP_OK);
        }
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        $cate_id = $request->cate_id;
        $this->validate($request, [
            'cate_id' => 'required|integer',
            'grade' => ['required', Rule::unique('grades')->where(function ($query) use ($cate_id) {
                return $query->where('cate_id', $cate_id);
            })],
            'mark_from' => ['required', 'integer', 'max:100', new UniqueMark($cate_id)],
            'mark_to' => ['required', 'integer', 'max:100', 'greater_than_field:mark_from', new UniqueMark($cate_id)],
            'gpa' => ['required', 'numeric', 'between:0.00,5.00', Rule::unique('grades')->where(function ($query) use ($cate_id) {
                return $query->where('cate_id', $cate_id);
            })],
            // 'gpa' => 'required|numeric|between:0.00,5.00'
        ],[
            'gpa.numeric' => 'GPA should be numeric',
            'gpa.between' => 'GPA should be between 0.00 to 5.00',
            'mark_to.greater_than_field' => 'This value must be greater than mark percentage from'
        ]);
        $request->merge(['grade' => Str::upper($request->grade)]);
        $create = Grade::create($request->all());
    }
```

Figure 52:Grading code

```php
    public function profile()
    {
        $user = User::findOrFail(Auth::user()->id);
        if($user->role_id == 1 || $user->role_id == 2 || $user->role_id == 3 || $user->role_id == 4 ) {
            $personal = Staff::where('email', $user->email)->first();
        } else {
            $personal = 'no-data';
        }
        return response()->json([

            'user' => $user,
            'personal' => $personal,
            'roles' => UserRole::get()

        ], Response::HTTP_OK);
    }

    public function settings(Request $request)
    {
        $user = User::findOrFail(auth()->user()->id);
        $staff = Staff::where('email', $user->email)->first();
        $this->validate($request, [
            'email'=>'required|email|unique:users,email,'.$user->id,
            'oldPassword' => ['required', new MatchOldPassword],
            'newPassword' => 'nullable|min:8',
            'newPassword_confirmation' => 'required_with:newPassword|same:newPassword'
        ],[
            'newPassword_confirmation.required_with' => 'Write your new password again'
        ]);

        if($request->newPassword) {
            $request->request->add(['password' => Hash::make($request->newPassword)]);
        }

        $user->update($request->all());
        $staff->update($request->all());
    }
}
```

Figure 53:Profile code

```php
public function index()
{
    if(\Gate::allows('isSuperAdmin') || \Gate::allows('isAdmin') || \Gate::allows('isTeacher'))  {
        return response()->json([

            'classes' => Classes::get(),
            'exams' => ExamCategory::get(),
            'gradings' => GradingCategory::get()


        ], Response::HTTP_OK);
    }
}

/**
 * Store a newly created resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    $this->validate($request, [
        'class' => 'required',
        'section' => 'required',
        'stu_id' => 'required',
        'exam_type' => 'required',
        'grading' => 'required',
        'year' => 'required',
        'marks' => 'required|array|min:1',
        'marks.*' => 'nullable|integer|between:0,100'
    ],[
        'marks.*.integer' => 'This should be an integer',
        'marks.*.between' => 'Marks should be between 0 and 100',
        'marks.required' => 'Marks are required'
    ]);
    $request->merge(['marks' => implode(',', array_values($request->marks))]);


    $create = Result::create($request->all());
}
```

Figure 54:Result code

```php
public function index()
{

    if(\Gate::allows('isSuperAdmin') || \Gate::allows('isAdmin'))  {
        return response()->json([

            'sections' => Section::orderBy('section', 'ASC')->get()

        ], Response::HTTP_OK);
    }
}

/**
 * Store a newly created resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    $this->validate($request, [
        'section' => 'required|unique:sections'
    ]);

    $section = new Section;
    $section->section = Str::ucfirst($request->section);
    $section->save();

    return ['message' => 'Created Successfully'];
}
```

Figure 55:Section code

```php
public function store(Request $request)
{
    $this->validate($request, [
        'email'=>'required|email|unique:users',
        'password'=>'required|min:8|confirmed',
        'role_id'=>'required'
    ]);

    if($request->photo) {
        $name = substr(base_convert(sha1(uniqid(mt_rand())), 16, 36), 0, 9);
        $imageName = $name . '-' . time() . '.' . explode('/', explode(':', substr($request->photo, 0, strpos($request->photo, ';')))[1])[1];
        Image::make($request->photo)->save(public_path('images/users/').$imageName);
    } else {
        $imageName = 'avatar.jpg';
    }

    $create = User::create([
        'email'  => $request->email,
        'password'  => Hash::make($request->password),
        'role_id' => $request->role_id,
        'photo' => $imageName,
        'status' => (boolean)$request->status
    ]);

    return ['message' => 'Created Successfully'];

}
```

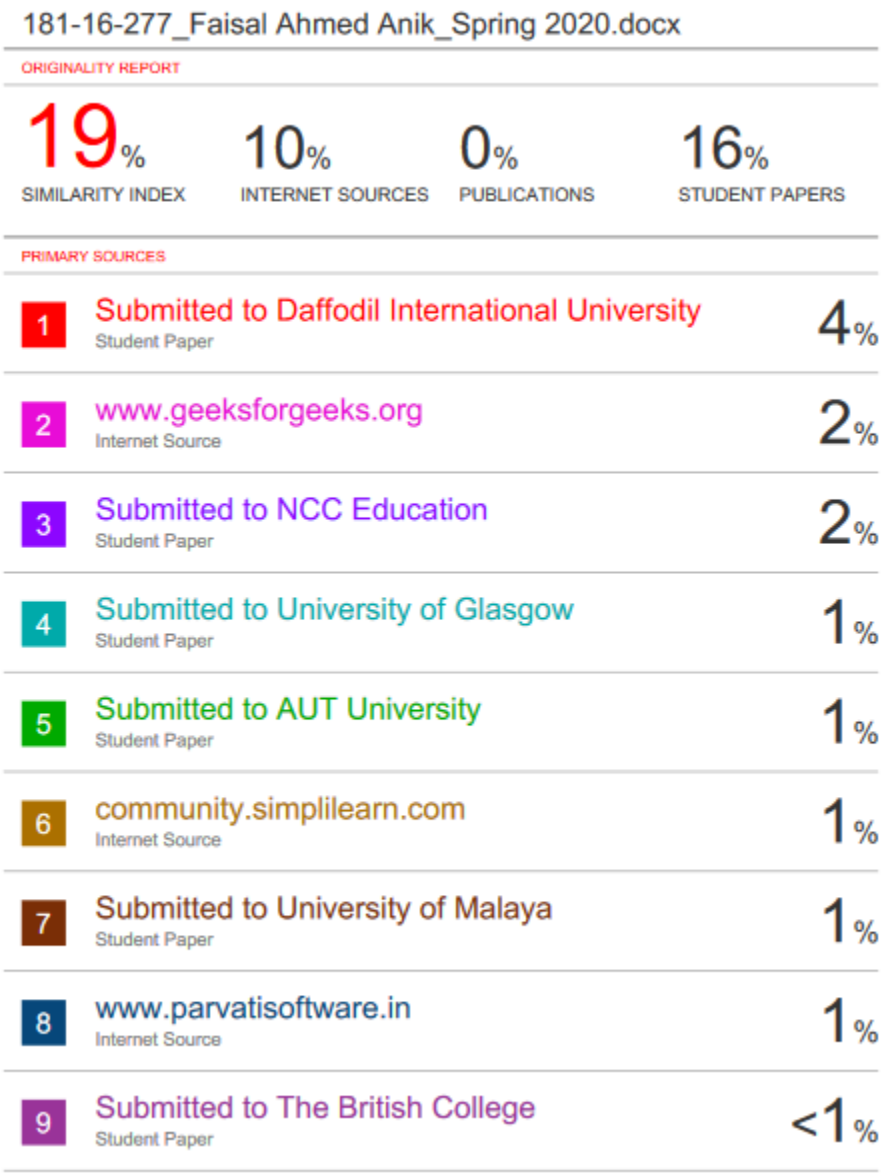
Figure 56:User manage code

```blade
<div id="sidebar-menu">
    <ul>
        <li><router-link to="/dashboard" class="waves-effect"><i class="fa fa-home"></i><span> Dashboard </span>
        </router-link></li>
        @cannot('isPortal')
        <li class="has_sub">
            <a href="#" class="waves-effect"><i class="fa fa-mortar-board"></i><span> Academic </span><span
            class="pull-right"><i class="md md-add"></i></span></a>
            <ul class="list-unstyled">
                <li><router-link to="/class_schedule">Class Schedule</router-link></li>
                <li><router-link to="/teacher_schedule">Teacher Schedule</router-link></li>
                @if(Gate::check('isSuperAdmin') || Gate::check('isAdmin'))
                <li><router-link to="/section">Section</router-link></li>
                <li><router-link to="/class">Class</router-link></li>
                <li><router-link to="/subject">Subject</router-link></li>
                <li><router-link to="/subject_group">Subject Group</router-link></li>
                @endif

            </ul>
        </li>
        <li class="has_sub">
            <a href="#" class="waves-effect"><i class="fa fa-user-plus"></i><span> Student</span><span class="pull-
            right"><i class="md md-add"></i></span></a>
            <ul class="list-unstyled">
                <li><router-link to="/admission">Admission</router-link></li>
                <li><router-link to="/student">Student Info</router-link></li>
            </ul>
        </li>
        <li class="has_sub">
            <a href="#" class="waves-effect"><i class="fa fa-calendar-check-o"></i><span> Attendance</span><span
            class="pull-right"><i class="md md-add"></i></span></a>
            <ul class="list-unstyled">
                <li><router-link to="/attendance">Daily Attendance</router-link></li>
                <li><router-link to="/all_attendance">All Attendance</router-link></li>
            </ul>
        </li>
        @can('isSuperAdmin')
        <li class="has_sub">
            <a class="waves-effect"><i class="fa fa-sitemap"></i><span> Human Resources </span><span class="pull-
            right"><i class="md md-add"></i></span></a>
            <ul class="list-unstyled">
                <li><router-link to="/staff">Staffs</router-link></li>
```

Figure 57:Sidebar code

Plagiarism Report



181-16-277_Faisal Ahmed Anik_Spring 2020.docx

ORIGINALITY REPORT

19%	10%	0%	16%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Daffodil International University Student Paper	4%
2	www.geeksforgeeks.org Internet Source	2%
3	Submitted to NCC Education Student Paper	2%
4	Submitted to University of Glasgow Student Paper	1%
5	Submitted to AUT University Student Paper	1%
6	community.simplilearn.com Internet Source	1%
7	Submitted to University of Malaya Student Paper	1%
8	www.parvatisoftware.in Internet Source	1%
9	Submitted to The British College Student Paper	<1%

Figure 58:Plagiarism Report