Project Report

on

E-Training Management System

Submitted in partial fulfillment for the award of **Post Graduate Diploma in Advance Computing (PG-DAC)** from **C-DAC ACTS (Pune).**

Guided by: Mr. Vishwas Srivastava

Submitted by:

Joshua M Singh	180840120077
Nitin Naresh Singh	180840120114
Shrilekha Bhau Pawar	180840120179
Surabhi Bhutada	180840120197
Vijay Singh	180840120216

Centre of Development of Advanced Computing (C-DAC), Pune

ACKNOWLEDGEMENT

This project "**E-Training Management System**" was a great learning experience for us and we are submitting this work to Advanced Computing Training School (CDAC ACTS).

We all are very glad to mention the name of *Mr. Vishwas Srivastava* for his valuable guidance to work on this project. His guidance and support helped us to overcome various obstacles and intricacies during the course of project work.

We are highly grateful to Ms. Risha P.R. Manager (ACTS training Centre), C-DAC, for her guidance and support whenever necessary while doing this course Post Graduate Diploma in *Advanced Computing (PG-DAC)* through C-DAC ACTS, Pune.

Our most heartfelt thanks goes to *Ms. Namrata Ailawar* (Course Coordinator, PG-*DAC*) who gave all the required support and kind coordination to provide all the necessities like required hardware, internet facility and extra Lab hours to complete the project and throughout the course up to the last day here in C-DAC ACTS, Pune.

From:

Joshua M Singh	180840120077
Nitin Naresh Singh	180840120114
Shrilekha Bhau Pawar	180840120179
Surabhi Bhutada	180840120197
Vijay Singh	180840120216

Centre of Development of Advanced Computing (C-DAC), Pune

TABLE OF CONTENTS

1. **Abstract**
2. **Introduction and Overview of Project**
   - 2.1 Purpose
   - 2.2 Aims and Objectives
   - 2.3 Scope
3. **Overall Description**
   - 3.1 Product Perspective
   - 3.2 Product Features
   - 3.3 User Classes and Characteristics
4. **Requirement Analysis**
   - 4.1 Feasibility Study
   - 4.2 Technical Feasibility
   - 4.3 Economic Feasibility
   - 4.4 Operational Feasibility
   - 4.5 Social Feasibility
5. **Selection of Software Development Lifecycle**
   - 5.1 Requirements Analysis
   - 5.2 Design Phase
   - 5.3 Implementation Phase
   - 5.4 Testing Phase
   - 5.5 Maintenance Phase
6. **System Requirements**
   - 6.1 Software Requirements

6.2 Hardware Requirements

7. Software Model Adopted

8. UML Diagrams

9. Database Tables

10. Snapshots

11. Testing

11.1 Introduction

11.2 Test Plan

11.3 Test Report

12. Future Enhancements

13. Bibliography

1. ABSTRACT

E-Training Management System is an online training management system, inspired from cdac's training system. The application will have options to view schedule, register student and faculty, attendance management, set class schedule etc.

The application usage will be widely shared among three clients, that are: Admin(Co-ordinator),user(student) and teaching faculty.

The Front end of the application is created using Angular framework, and the backend is powered by SpringBoot .The server security has been given much importance and is implemented using spring security and JSON Web Token (JWT). JWT is also used to maintain session for a particular logged in client and is a secure way to transmit vital information over a network through particular defined encryption, which ultimately provides network security. MySQL is used as the database server and vast array of methods provided by JPA Repository are used to interact with the database.

2. Introduction and Overview of the Project

E-Training Management System is a software developed for training institutes, where there are three client i.e., admin (co-ordinator), user (student) and faculty. The Admin plays a vital role. The admin controls the entire database, add new user (admin, faculty or student), add new courses as well as their corresponding modules, also view/add schedules, view attendance details and get list of users. The Student can view his/her schedule and give the attendance. The Faculty can view his / her Schedule and cancel class.

2.1 Purpose:

The purpose of developing training Management System is to change the primitive pen to paper design of managing different components of a training to a computerized system. This would enhance and ease the operation done by different client on the components related to training management.

2.2 Aim & Objective:

The main aim of **E-Training Management System** is to organise training delivery system which keeps students and teachers up to date about their upcoming schedules, Manage faculty and student details, manage student attendance and much more. The database will maintain the users' details, course details, Schedule details, room details etc which is managed by administrator/Co-ordinator. This Online Management system involves three types of users:

- **USER (STUDENT)**
- **COORDINATOR (ADMIN)**
- **FACULTY**

USER ROLE:

The Student can login/logout to the System. He / She can view his/her schedule and give their attendance. The student can just view his details but can not make changes in the database.

COORDINATOR (ADMIN) ROLE:

The Admin plays a vital role in this **E-Training Management System**. The admin controls the entire database. The main role of the admin is to add new user (admin, faculty or student), add courses as well as their corresponding modules, also add/view schedules and get list of all the users and to also add room details of the courses.

FACULTY ROLE:

The Faculty can login/logout to the System. He /She can view his / her schedule and cancel class by notifying through email to coordinator.

2.3 Scope:

This project provides secure registration and profile management facility for users in secure manner. It allows multiple users to be logged in at the same time. The administrator controls the entire database. The main role of the administrator is to safeguard the database and can add, update or delete the data from the database.

3. Overall description:

3.1 Product Perspective:

The product is developed as a online Training management website. Students can view the Schedule, give Attendance but before that he must be registered by Admin. This application stores all the information in the database which can be retrieved whenever needed and all the validations are performed during the entry of the data by the Admin thus providing authentication and authorization of the user (Customer and Faculty).

3.2 Product Features:

Student have to login to give attendance and to view the Schedule of their upcoming lecture. Faculty can also login to view the schedule and cancel the classes. Moreover Admin has the authority to register the student and faculty

After providing and validating their credentials.

3.3 User Classes and Characteristics:

This Online Training Management system involves with three types of users:

1. STUDENTS
2.ADMIN

3.FACULTY

1.The Student can login/logout to the System. He / She can view his/her schedule and give the attendance. The customer can just view their details whereas he/she could not make changes in the database.

2.The main role of the admin is to add new user (admin, faculty or student),add courses as well as their corresponding modules also view schedules and list of users.The administrator controls the entire database.

3.The Faculty can also login/logout to the System. He /She can view his / her

Schedule and cancel class .

4. Requirement Analysis

4.1 Feasibility Study:

From the inception of ideas for software system, until it is implemented and delivered to customer and even after that the system undergoes gradual developments and evaluations.

Feasibility is the determination of whether or not a project is worth doing. A feasibility study is carried out select a best system that mate performance requirements. The data collected during primary investigation examines system feasibilities that is likelihood that the system will be beneficial to the organization. Four tests for feasibility study are as follows:-

4.2 Technical Feasibility:

This is concerned with specifying equipment and software that will successfully satisfy the use considerably, but might include

- The feasibility to produce output in a given time because system is fast enough to handle multiple users.
- Response time under certain circumstances and ability to process a certain volume of transaction of a particular speed.
- Feasibility to communicate data to distant location.

4.3 Economical Feasibility:

Economic analysis is the most frequently used technique used for evaluating the effectiveness of a proposed system. More commonly known as cost/benefit analysis the procedure is to determine the benefits and savings that are expected from a proposed system and compared them with cost. Though the cost of installing the system may appear high, it is one time investment. The resulting benefits is that automation results in turnaround time. The resulting cost/benefit ratio is favorable.

In this type of feasibility study, the benefits of the system to the organization are considered by taking into consideration the cost-benefit analysis. The basic software, which is required for the implementation of the system, is Java which easily available. Also with the basic training user can use this software thus reducing the training cost to the organization. Thus, using this system is feasible for the organization and loading Java and the proposed system is economically feasible for the organization. As our system goes online we will have a lot of customers adding to our publicity. This in turn will increase our profit.

4.4 **Operational Feasibility:**

It is mainly related to human organizational as social aspects. The points to be considered are - The system interface is standard, user friendly and provides extensive help. Hence no special training is not required.

In this type of feasibility study the operation implementation of the system is considered. Checking is done regarding whether it is feasible for the user department to use the software or will there be any inertial resistance from the users. Thus the proposed system is said to be operationally feasible only of the end users are able to understand the system clearly and correctly and can use the system with ease and with the minimum training.

We need to train our staff so that system will be handled efficiently. As the system developed is very user-friendly and easy to operate for any person with minimum computer knowledge of computer is also able to handle our system. It is also easy to operate due to the user-friendly interface developed using Java.

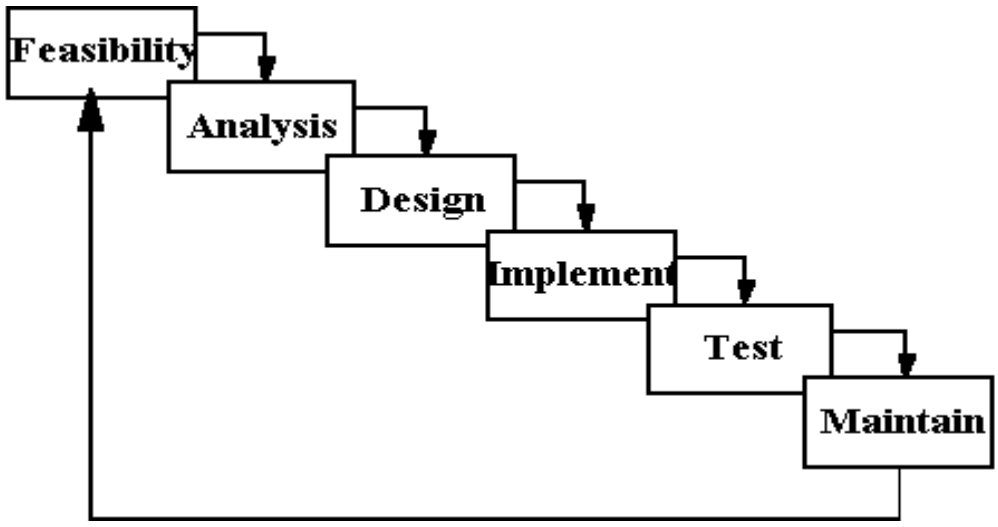
4.5 **Social Feasibility:**

Social feasibility is determination of whether a proposed project will be acceptable to people or not.

5. Selection Of Software Development Lifecycle

SYSTEM DEVELPOMENT LIFE CYCLE (SDLC)

The Systems Development Life Cycle **(SDLC)** is a conceptual model used in project management that describes the stages involved in an information system development project from an initial feasibility study through maintenance of the completed application. Various SDLC methodologies have been developed to guide the processes involved including the waterfall model (the original SDLC method). Documentation is crucial regardless of the type of model chosen or devised for any application, and is usually done in parallel with the development process. Some methods work better for specific types of projects, but in the final analysis, the most important factor for the success of a project may be how closely
particular plan was followed.



5.1 Requirement Analysis:

Analysis is the focus of system developing and is the stage when system designers have to work at two levels of definition regarding the study of situational issues and possible solutions in terms of "what to do" and "how to do".

5.1.1. Modeling the Requirements

Module Description

Login as well as Register new Users

View Student and faculty Details

Attendance Management

Add Courses and their corresponding Modules, Create and Edit Schedule

5.2 Design Phase:

The design phase involves converting the informational, functional, and network requirements identified during the initiation and planning phases into unified design specifications that developers use to script programs during the development phase. Program designs are constructed in various ways. Using a top down approach, designers first identify and link major program components and interfaces, then expand design layouts as they identify and link smaller subsystems and connections. Using a bottom-up approach, designers first identify and link minor program components and interfaces, then expand design layouts as they identify and link larger systems and connections.

Contemporary design techniques often use prototyping tools that build mock-up designs of items such as application screens, database layouts, and system architectures. End users, designers, developers, database managers, and network administrators should review and refine the prototyped designs in an iterative process until they agree on an acceptable design.

Designers should carefully document completed designs. Detailed documentation enhances a programmer's ability to develop programs and modify them after they are placed in production. The documentation also helps management ensure final programs are consistent with original goals and specifications. Organizations should create initial testing, conversion, implementation, and training plans during the design phase. Additionally, they should draft user, operator, and maintenance manuals.

For design of the website project:

1. First Database has to be designed which can be used to handle all the requirements of the users.

2. The basic structure of the website has to be designed.
3. The main template to be used for the website is designed.

5.3 Implementation Phase:

The implementation phase involves installing approved applications into production environments. Primary tasks include announcing the implementation schedule, training end users, and installing the product. Additionally, organizations should input and verify data, configure and test system and security parameters, and conduct post-implementation reviews. Management should circulate implementation schedules to all affected parties and should notify users of any implementation responsibilities.

After organizations install a product, pre-existing data is manually input or electronically transferred to a new system. Verifying the accuracy of the input data and security configurations is a critical part of the implementation process.
Organizations often run a new system in parallel with an old system until they verify the accuracy and reliability of the new system. Employees should document any programming, procedural, or configuration changes made during the verification process.

For implementation of the website project:

1. The website can be installed on a computer or a server which supports Java and MYSQL installed in it.
2. The owners of the website are to be properly trained to use all the features of the website, giving details of each features of the website.
3. To show the accuracy of the website and conformance of the website to the requirements of the owners or users of the website.

5.4 Testing Phase:

Software testing

Software testing is the process used to measure the quality of developed computer software. Usually, quality is constrained to such topics as correctness, completeness, security, but can also include more technical requirements as described under the ISO standard ISO 9126, such as capability, reliability, efficiency, portability, maintainability, compatibility, and usability. Testing is a process of technical investigation, performed on behalf of stakeholders, that is intended to reveal quality-related information about the product with respect to the context in which it is intended to operate.

White box, black box, and grey box testing

White box and black box testing are terms used to describe the point of view that a test engineer takes when designing test cases. **Black box** testing treats the software as a black-box without any understanding as to how the internals behave. Thus, the tester inputs data and only sees the output from the test object. This level of testing usually requires thorough test cases to be provided to the tester who then can simply verify that for a given input, the output value (or behavior), is the same as the expected value specified in the test case.

White box testing, however, is when the tester has access to the internal data structures, code, and algorithms. For this reason, unit testing and debugging can be classified as white-box testing and it usually requires writing code, or at a minimum, stepping through it, and thus requires more skill than the black-box tester. If the software in test is an interface or API of any sort, white-box testing is almost always required.

In recent years the term grey box testing has come into common usage. This involves having access to internal data structures and algorithms for purposes of designing the test cases, but testing at the user, or black-box level. Manipulating input data and formatting output do not qualify as grey-box because the input and output are clearly outside of the black-box we are calling the software under test.

This is particularly important when conducting integration testing between two modules of code written by two different developers, where only the interfaces are exposed for test.

Grey box testing could be used in the context of testing a client-server environment when the tester has control over the input, inspects the value in a SQL database, and the output value, and then compares all three (the input, sql value, and output), to determine if the data got corrupt on the database insertion or retrieval.

Verification and Validation

Software testing is used in association with verification and validation (V&V). *Verification* is the checking of or testing of items, including software, for conformance and consistency with an associated specification. Software testing is just one kind of verification, which also uses techniques such as reviews, inspections, and walkthroughs. *Validation* is the process of checking what has been specified is what the user actually wanted.

· Verification: Have we built the software right? (i.e. does it match the specification).
· Validation: Have we built the right software? (i.e. Is this what the customer wants?)

Level of testing:

- **Unit testing** tests the minimal software component, or module. Each unit (basic component) of the software is tested to verify that the detailed design for the unit has been correctly implemented. In an Object-oriented

environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors.

- **Integration testing** exposes defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.

- **Functional testing** tests at any level (class, module, interface, or system) for proper functionality as defined in the specification.

- **System testing** tests a completely integrated system to verify that it meets its requirements.

- **System integration testing** verifies that a system is integrated to any external or third party systems defined in the system requirements.

- **Acceptance testing** can be conducted by the end-user, customer, or client to validate whether or not to accept the product. Acceptance testing may be performed as part of the hand-off process between any two phases of development.

- ***Alpha testing*** is simulated or actual operational testing by potential users/customers or an independent test team at the developers' site. Alpha testing is often employed for off-the-shelf software as a form of internal acceptance testing, before the software goes to beta testing.
- ***Beta testing*** comes after alpha testing. Versions of the software, known as beta versions, are released to a limited audience outside of the company. The software is released to groups of people so that further testing can ensure the product has few faults or bugs. Sometimes, beta versions are made available to the open public to increase the feedback field to a maximal number of future users.

It should be noted that although both Alpha and Beta are referred to as testing it is in fact use immersion. The rigors that are applied are often unsystematic and many of the basic tenets of testing process are not used. The Alpha and Beta period provides insight into environmental and utilization conditions that can impact the software.

After modifying software, either for a change in functionality or to fix defects, a regression test re-runs previously passing tests on the modified software to ensure that the modifications haven't unintentionally caused a ***regression*** of previous functionality. Regression testing can be performed at any or all of the above test levels. These regression tests are often automated.

- **Smoke Testing**

Smoke testing is a term used in plumbing, woodwind repair, electronics, and computer software development. It refers to the first test made after repairs or first assembly to provide some assurance that the system under test will not catastrophically fail. After a *smoke test* proves that the pipes will not leak, the keys seal properly, the circuit will not burn, or the software will not crash outright, the assembly is ready for more stressful testing.

· In plumbing, a *smoke test* forces actual smoke through newly plumbed pipes to find leaks, before water is allowed to flow through the pipes.

· In woodwind instrument repair, a smoke test involves plugging one end of an instrument and blowing smoke into the other to test for leaks. (This test is no longer in common use)

· In electronics, a *smoke testing* is the first time a circuit is attached to power, which will sometimes produce actual smoke if a design or wiring mistake has been made.

· In computer programming and software testing, *smoke testing* is a preliminary to further testing, which should reveal simple failures severe enough to reject a prospective software release. In this case, the smoke is metaphorical.

5.5 Maintenance:

The maintenance phase involves making changes to hardware, software, and documentation to support its operational effectiveness. It includes making changes to improve a system's performance, correct problems, enhance security, or address user requirements. To ensure modifications do not disrupt operations or degrade a system's performance or security, organizations should establish appropriate change management standards and procedures.

Routine changes are not as complex as major modifications and can usually be implemented in the normal course of business. Routine change controls should include procedures for requesting, evaluating, approving, testing, installing, and documenting software modifications. Maintaining accurate, up-to-date hardware and software inventories is a critical part of all change management processes.

Management should carefully document all modifications to ensure accurate system inventories. Management should coordinate all technology related changes through an oversight committee and assign an appropriate party responsibility for administering software patch management programs. Quality assurance, security, audit, regulatory compliance, network, and end-user personnel should be appropriately included in change management processes.

Risk and security review should be done whenever a system modification is implemented to ensure controls remain in place.

For maintenance of the website:

1. The database has to be updated regularly according to new available information.
2. Redundant and false information must be removed from the database.

6. System Requirements:

A. Hardware Requirements:

Hard Disk Space	10GB (minimum).
Processor	Pentium IV
Processor Speed	1.13Ghz

Ram		1 GB (minimum)

B. Software Requirements:

Operating System	:	Windows 10 and above
Technologies	**:**	SpringBoot,Hibernate,Javascript, BootStrap, JWT, Spring Security
Client Side Scripting Language	:	Angular 6, Material UI design
Markup Language	**:**	HTML,CSS
Database Server	:	MySQL
Web Server	:	Tomcat v8.0
IDE	**:**	Eclipse, VS Code

7. Software Model adopted:

Incremental Model

The goal of a software development process is production of software that works, is on time and within budget, and can be maintained and reused.

The incremental software development approach is a combination of Rational Unified Process (RUP) and eXtreme Programming (XP). The focus of our approach is to get the "right level" of process. Understanding the challenges faced by the development team and the business environment in which it operates, determines the right level of process formality. Once we understand these challenges, we apply just enough process to mitigate the risks. Although there is no one-size-fits-all process, lightweight or otherwise, our

approach does emphasize certain values: Communication, Simplicity, Feedback and Iteration.

(a) Communication

A good process *facilitates* communications. It provides the channels between the parties that need to communicate, and indicates the form, purpose and goal of that communication.

Documents may be needed to achieve this facilitation; however a good process does not reduce communication to production of documents. Communication takes place between *people*; documents are secondary.

(b) Simplicity

A process that is too complex will fail. Simplicity is a value to be intensely defended, both in our software and in our process. We do add activities, documents, or artifacts to our processes unless the need for them is critical. We regularly sweep through our processes and remove accumulated complexity. Anything that cannot be completely justified is eliminated. A process description should always look too small.

(c) Feedback

The only way to truly control a software project is to continuously measure its progress, compare that progress against the plan, and then adjust the development parameters to correct any deviation from the plan. This is the foundational motivation for all iterative methods.

(d) Increment

In our approach, the software development process moves forward in increments called iterations. The goal of each iteration is to develop some working software that can be demonstrated to all the stakeholders, and that the stakeholders will find meaningful.

The software developed by iteration should cut through all or most of the major subsystems of the project. It should not be concentrated into a single subsystem. Each iteration represents an effort by each member of the team to build a small part of their part of the project and integrate those parts together. The length of iteration depends upon the kind of project we are working with. However, short iterations are to be desired over long ones. The shorter the iteration, the less time passes before the team gets feedback. Iteration lengths of one or two weeks are not too short for most projects.

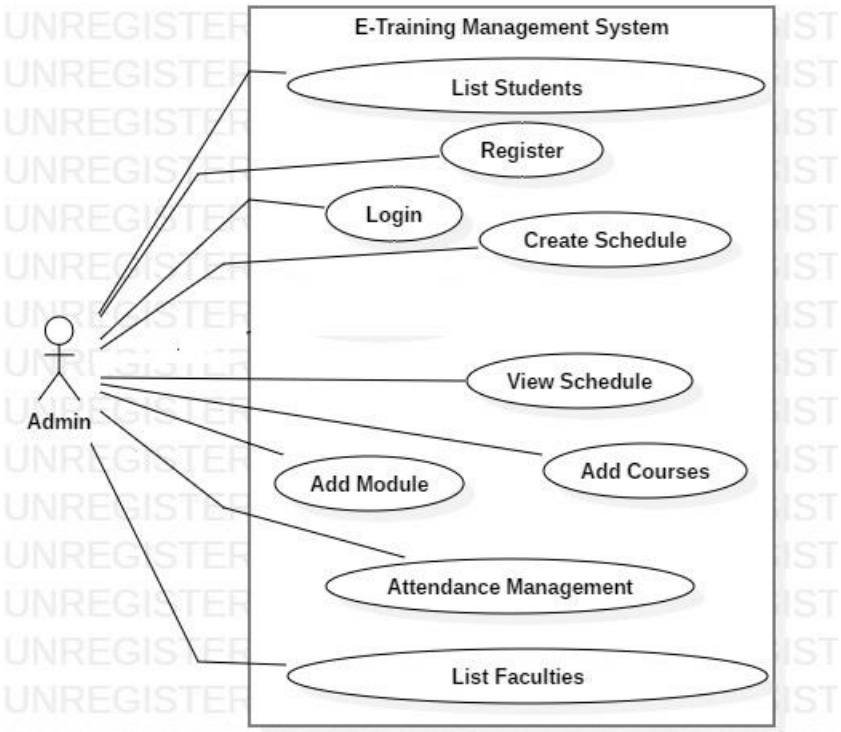
In iteration, all of the software development activities are executed. There is some business modeling in order to understand the needs of the business fulfilled by that iteration. There is some requirements analysis, to ensure we understand what behaviors the iteration must

have. There is some analysis, design, implementation, testing, and deployment planning. No activity is ever omitted from iteration.

8. UML Modeling:

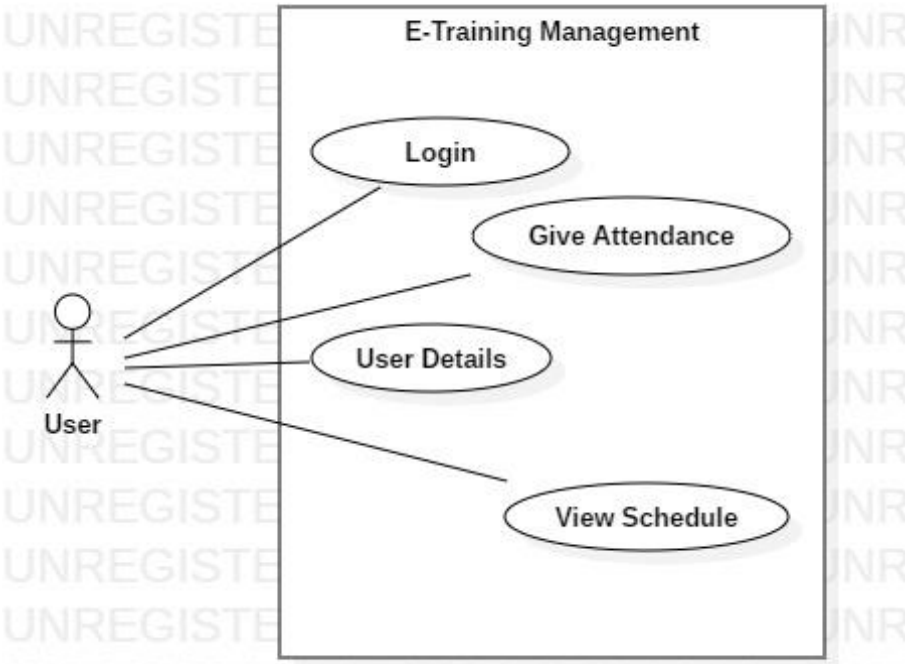
Use-Case:

USE CASE 1: Admin

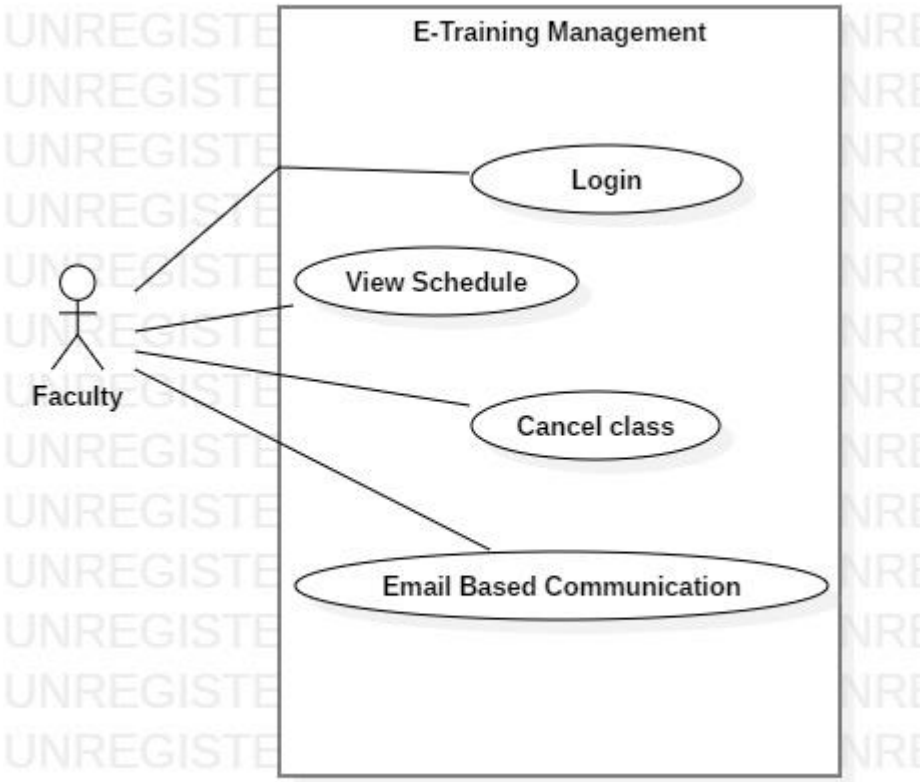


USE CASE :USER

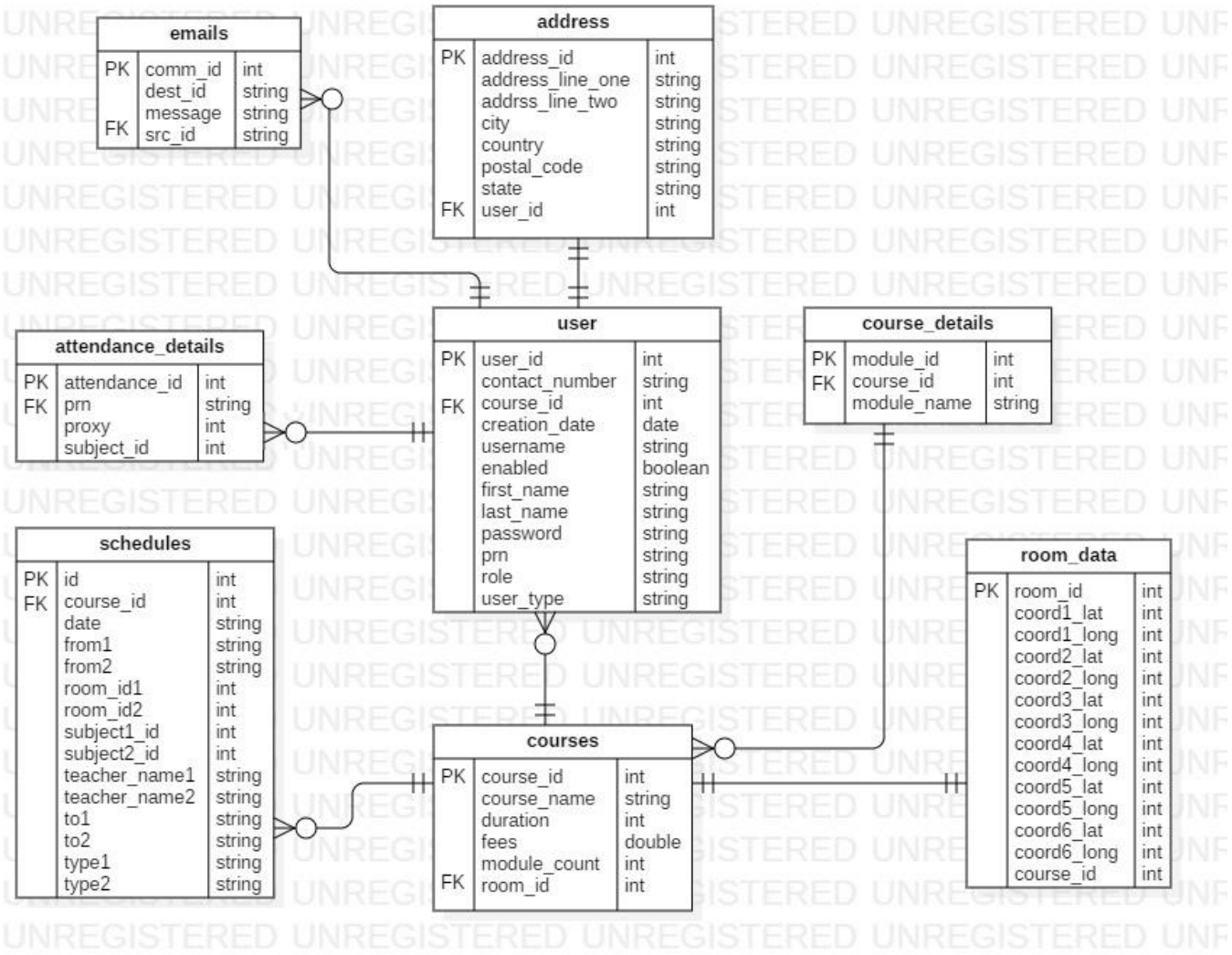
Admin UseCase.mdj



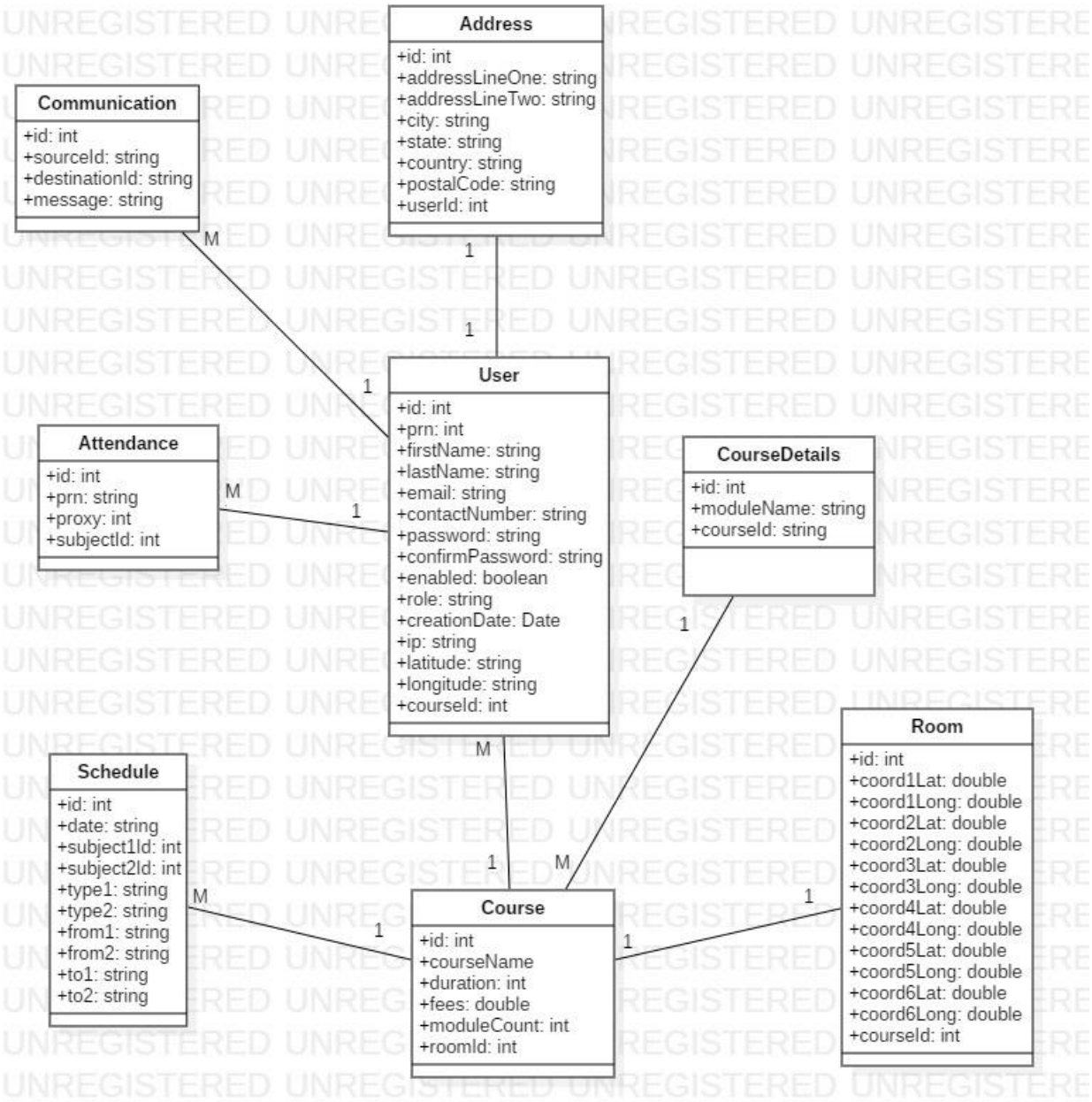
USE CASE:Faculty


E-Training Management
Login
View Schedule
Cancel class
Email Based Communication
Faculty

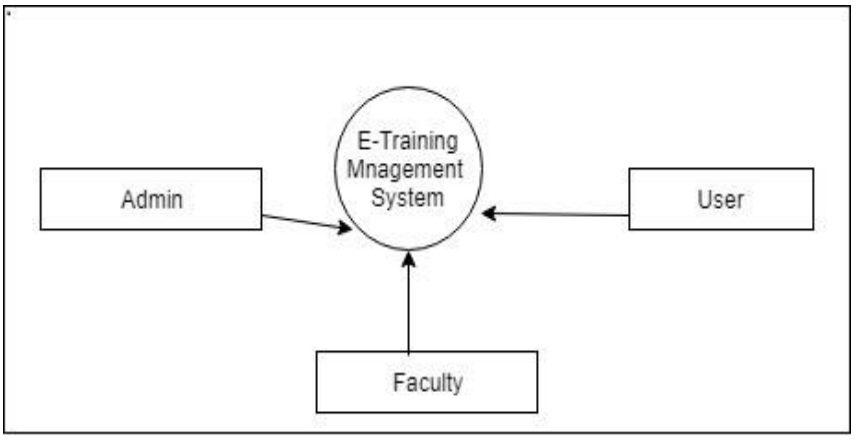
E-R Diagram


emails
PK comm_id int
dest_id string
message string
FK src_id string
address
PK address_id int
address_line_one string
addrss_line_two string
city string
country string
postal_code string
state string
FK user_id int
user
PK user_id int
contact_number string
FK course_id int
creation_date date
username string
enabled boolean
first_name string
last_name string
password string
prn string
role string
user_type string
course_details
PK module_id int
FK course_id int
module_name string
attendance_details
PK attendance_id int
FK prn string
proxy int
subject_id int
schedules
PK id int
FK course_id int
date string
from1 string
from2 string
room_id1 int
room_id2 int
subject1_id int
subject2_id int
teacher_name1 string
teacher_name2 string
to1 string
to2 string
type1 string
type2 string
courses
PK course_id int
course_name string
duration int
fees double
module_count int
FK room_id int
room_data
PK room_id int
coord1_lat int
coord1_long int
coord2_lat int
coord2_long int
coord3_lat int
coord3_long int
coord4_lat int
coord4_long int
coord5_lat int
coord5_long int
coord6_lat int
coord6_long int
course_id int

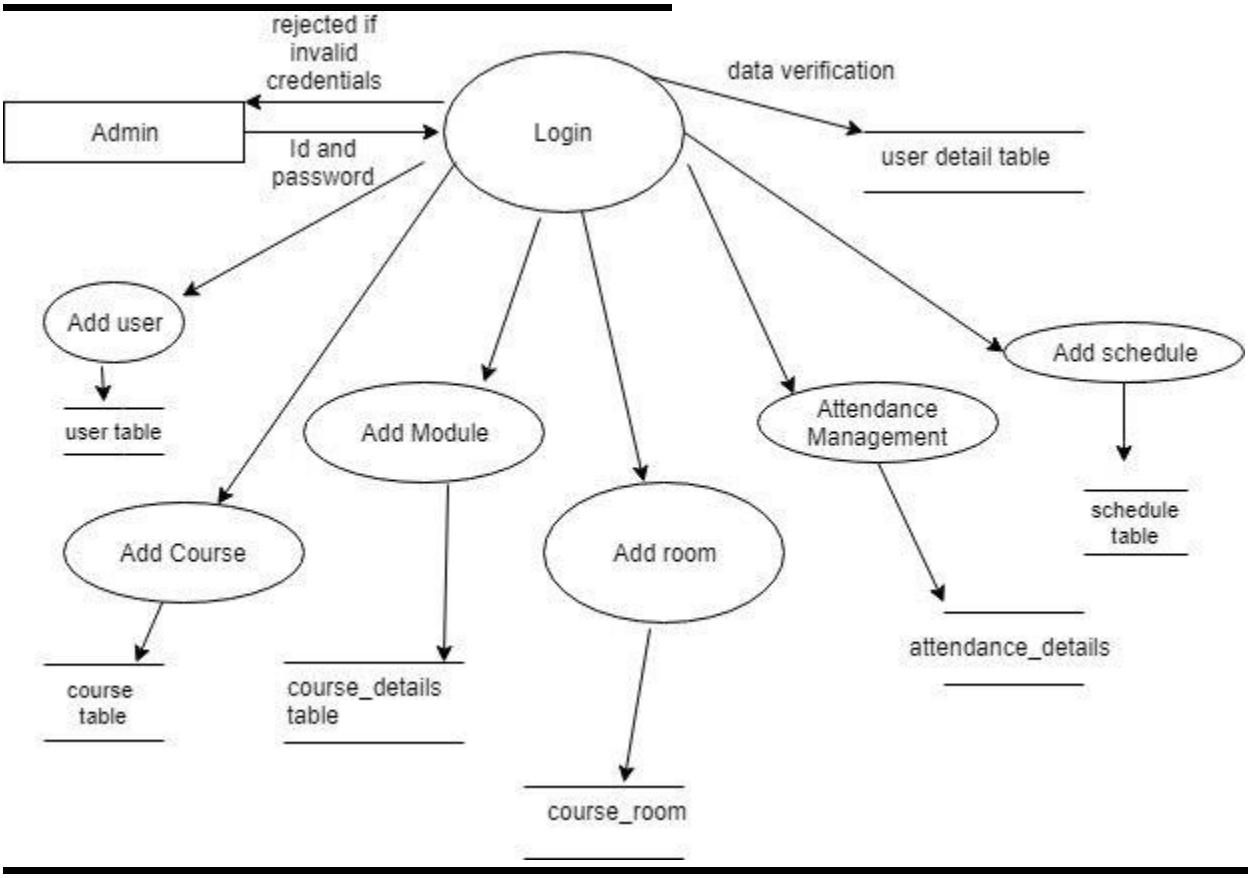
Class Diagram



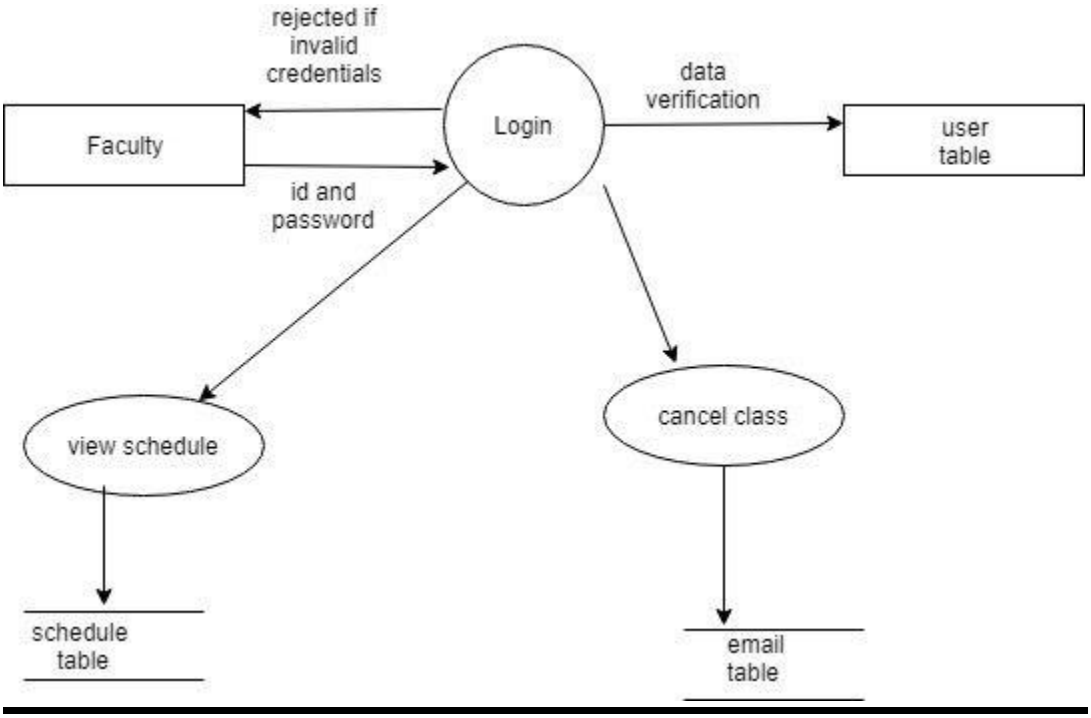
Data-Flow Diagram
Context-Level DFD:



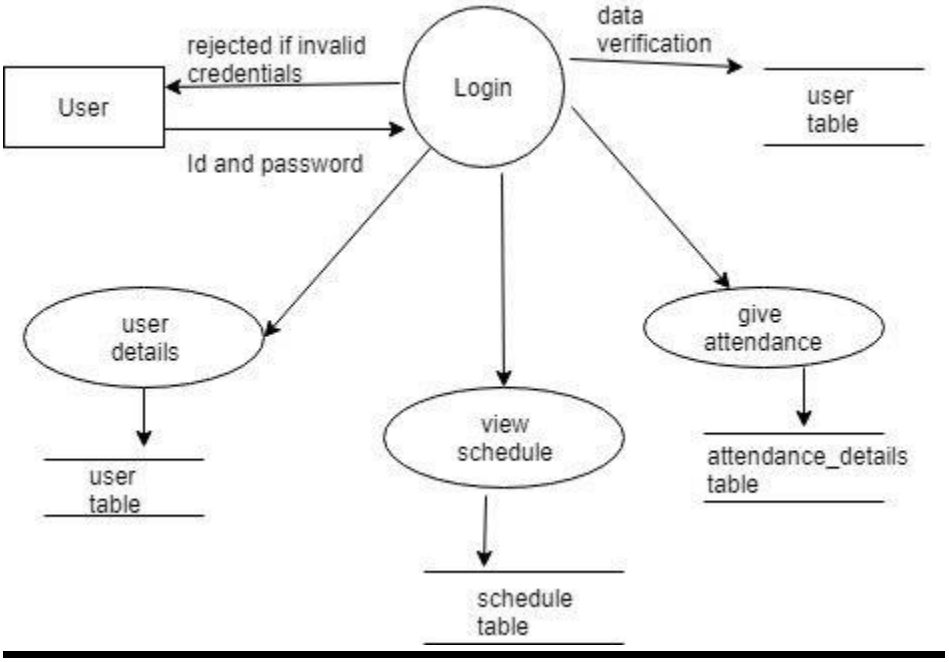
Level 1 DFD for Admin:



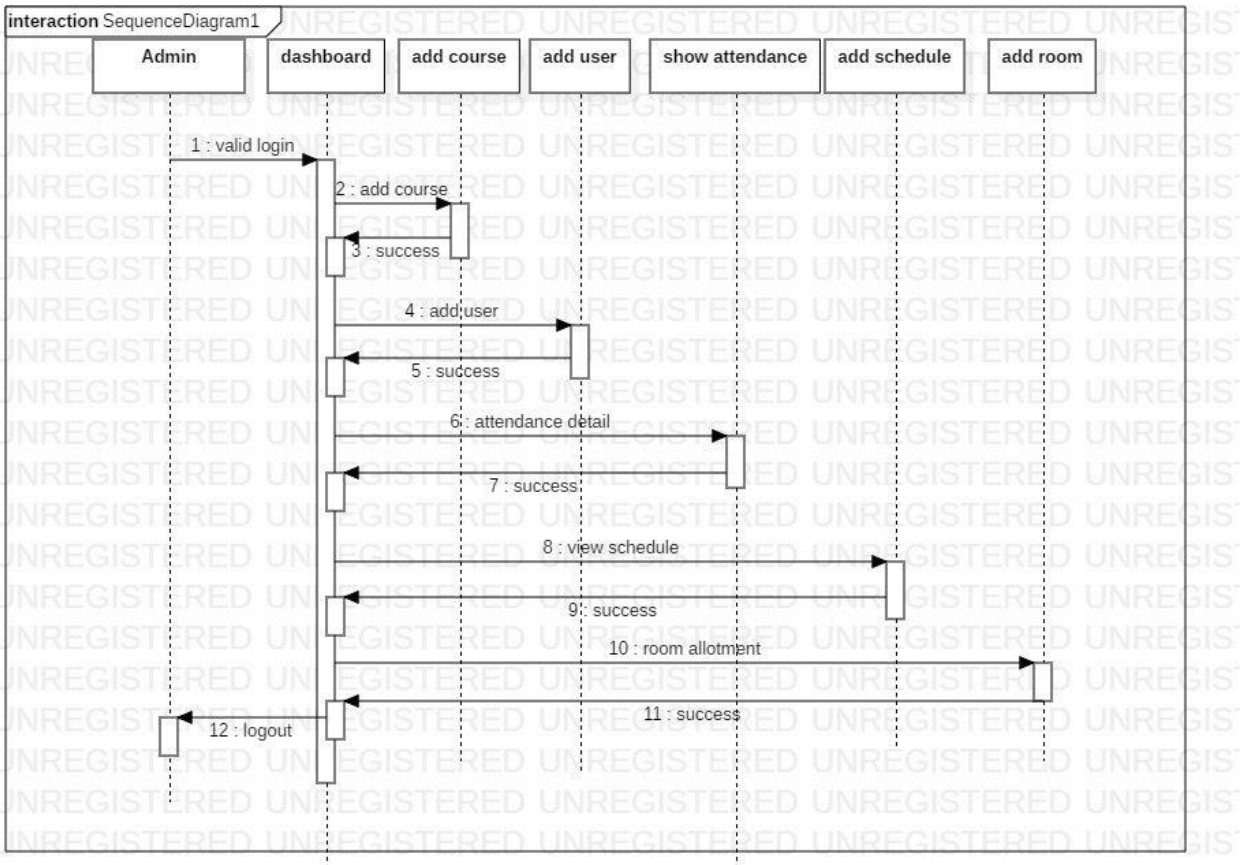
Level 1 DFD for Faculty:



Level 1 DFD for Student:



Sequence Diagram:



9. Database Tables:

A database design is a collection of stored data organized in such a way that the data requirements are satisfied by the database. The general objective is to make information access easy, quick, inexpensive and flexible for the user. There are also some specific objectives like controlled redundancy from failure, privacy, security and performance.

A collection of relative records make up a table. To design and store data to the needed forms database tables are prepared. Two essential settings for a database are:

- Primary key: - The field that is unique for all the record occurrences.
- Foreign key: - The field used to set relation between tables. Normalization is a technique to avoid redundancy in the tables.

Table 1- emails:

Field	Type	Null	Key	Default	Extra
comm_id	int(11)	No	PRI	NULL	auto_increment
dest_id	varchar(255)	Yes		NULL	
message	varchar(255)	Yes		NULL	
src_id	varchar(255)	Yes	MUL	NULL	

Table 2- address:

Field	Type	Null	Key	Default	Extra
address_id	int(11)	No	PRI	NULL	Auto_increment
address_line_one	varchar(255)	Yes		NULL	
address_line_two	varchar(255)	Yes		NULL	
city	varchar(255)	Yes		NULL	
state	varchar(255)	Yes		NULL	
country	varchar(255)	Yes		NULL	
postal_code	varchar(255)	Yes		NULL	
User_id	int(11)	Yes	MUL	NULL	

Table 3- user:

Field	Type	Null	Key	Default	Extra
User_id	int(11)	No	PRI	NULL	Auto_increment
Contact_number	varchar(255)	Yes		NULL	
Course_id	int(11)	No	MUL	NULL	
Creation_date	date	Yes		NULL	
username	varchar(255)	Yes		NULL	
enabled	Tinyint(1)	No		NULL	
First_name	varchar(255)	Yes		NULL	
Last_name	varchar(255)	Yes		NULL	
password	varchar(255)	Yes		NULL	
prn	varchar(255)	Yes		NULL	
role	varchar(255)	Yes		NULL	
User_type	varchar(255)	Yes		NULL	

Table 4- attendance_details:

Field	Type	Type	Key	Default	Extra
Attendance id	int(11)	No	PRI	NULL	auto_increment
prn	varchar(255)	Yes	MUL	NULL	
proxy	int(11)	Yes		NULL	
Subject_id	int(11)	Yes		NULL	

Table 5- courses:

Field	Type	Null	Key	Default	Extra
course_id	int(11)	No	PRI	NULL	auto_increment
Course_name	varchar(255)	Yes		NULL	
duration	int(11)	Yes		NULL	
fees	double	Yes		NULL	
room_id	int(11)		MUL	NULL	
module_count	int(11)	Yes		NULL	

Table 6- room_data:

Field	Type	Null	Key	Default	Extra
room_id	int(11)	No	PRI	NULL	auto_increment
coord1_lat	double	Yes		NULL	
coord1_long	double	Yes		NULL	
Coord2_lat	double	Yes		NULL	
Coord2_long	double	Yes		NULL	
Coord3_lat	double	Yes		NULL	
Coord3_long	double	Yes		NULL	
Coord4_lat	double	Yes		NULL	
Coord4_long	double	Yes		NULL	
Coord5_lat	double	Yes		NULL	
Coord5_long	double	Yes		NULL	
Coord6_lat	double	Yes		NULL	
Coord6_long	double	Yes		NULL	
Course_id	double	Yes		NULL	

Table 7- course_detail:

Field	Type	Null	Key	Default	Extra
module_id	int(11)	No	PRI	NULL	auto_increment
course_id	int(11)	No	MUL	NULL	
module_name	varchar(255)	Yes		NULL	

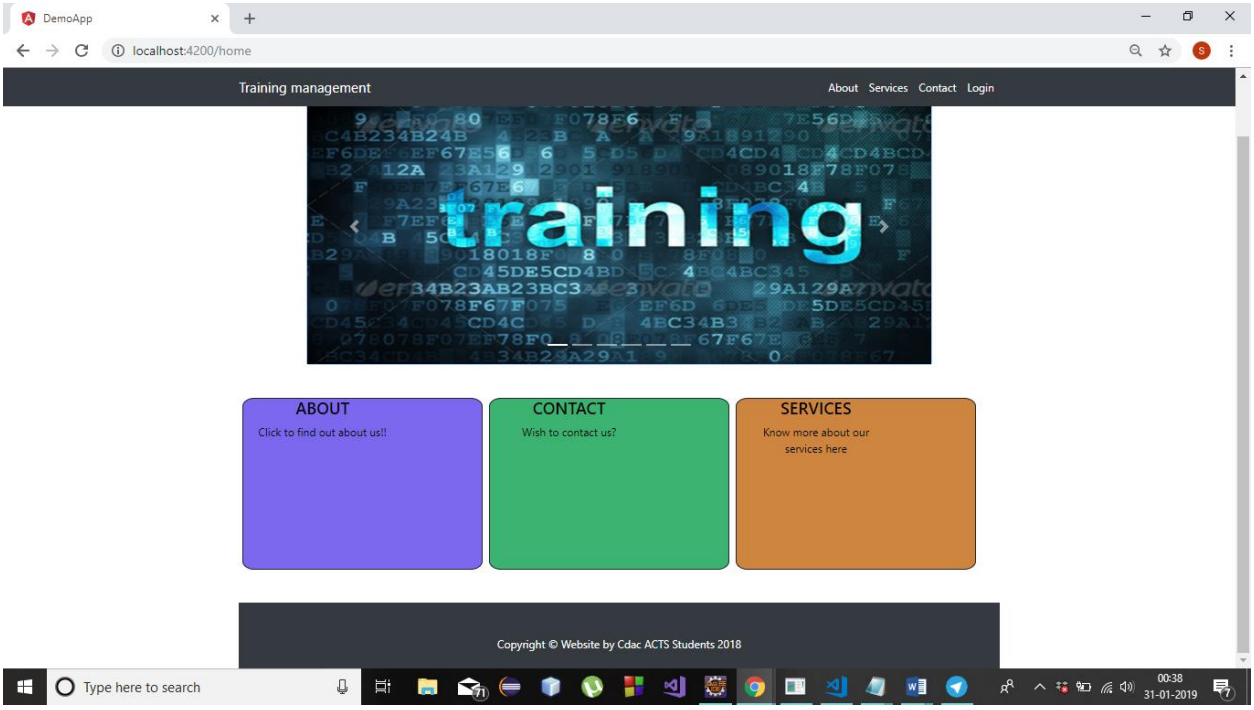
Table 8 -schedules:

Field	Type	Null	Key	Default	Extra
Id	int(11)	No	PRI	NULL	auto_increment
course_id	int(11)	Yes	MUL	NULL	
date	varchar(255)	Yes		NULL	
from1	varchar(255)	Yes		NULL	
from2	varchar(255)	Yes		NULL	
room_id1	int(11)	Yes		NULL	
room_id2	int(11)	Yes		NULL	

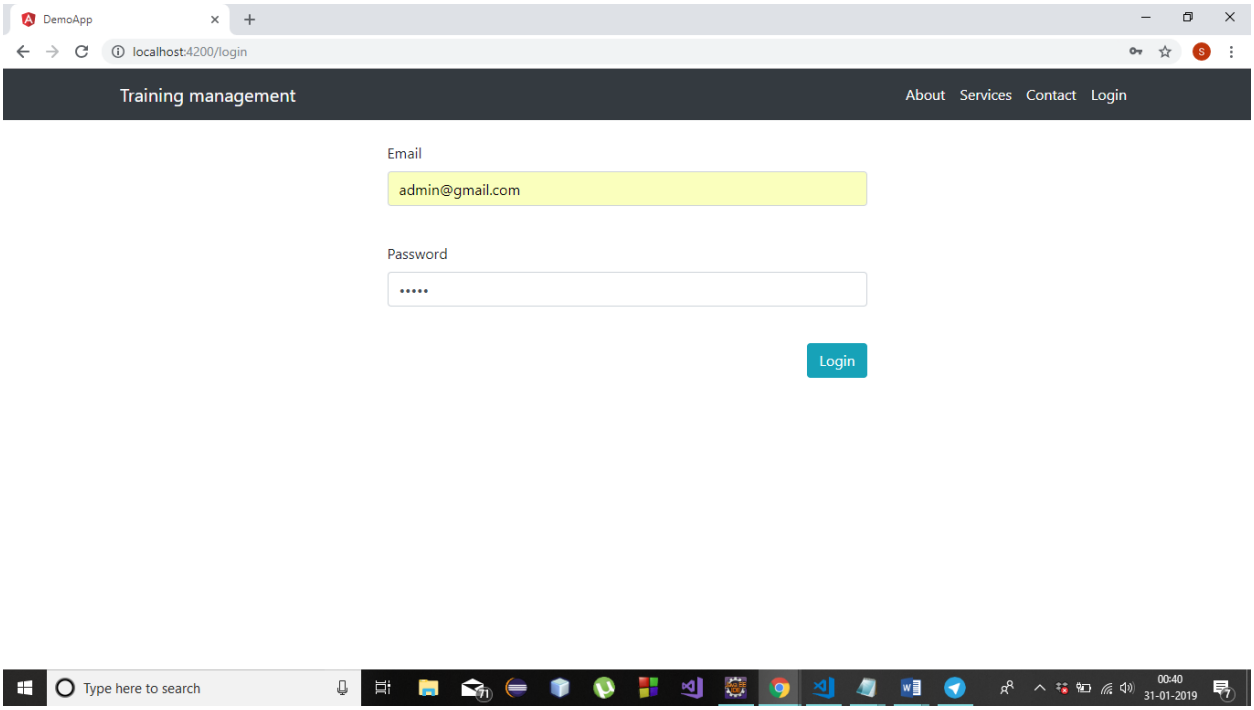
subject1_id	int(11)	Yes		NULL	
subject2_id	int(11)	Yes		NULL	
teacher_name1	varchar(255)	Yes		NULL	
teacher_name2	varchar(255)	Yes		NULL	
to1	varchar(255)	Yes		NULL	
to2	varchar(255)	Yes		NULL	
type1	varchar(255)	Yes		NULL	
type2	varchar(255)	Yes		NULL	

10. SNAPSHOTS

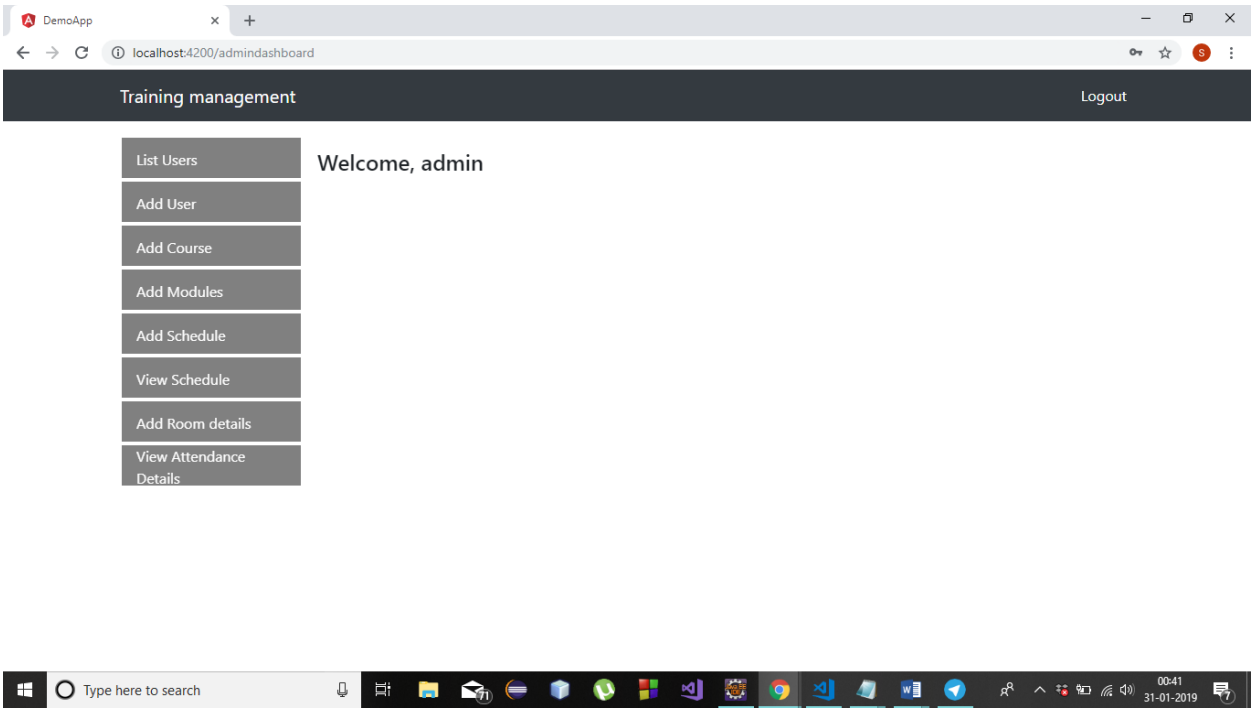
Homepage:



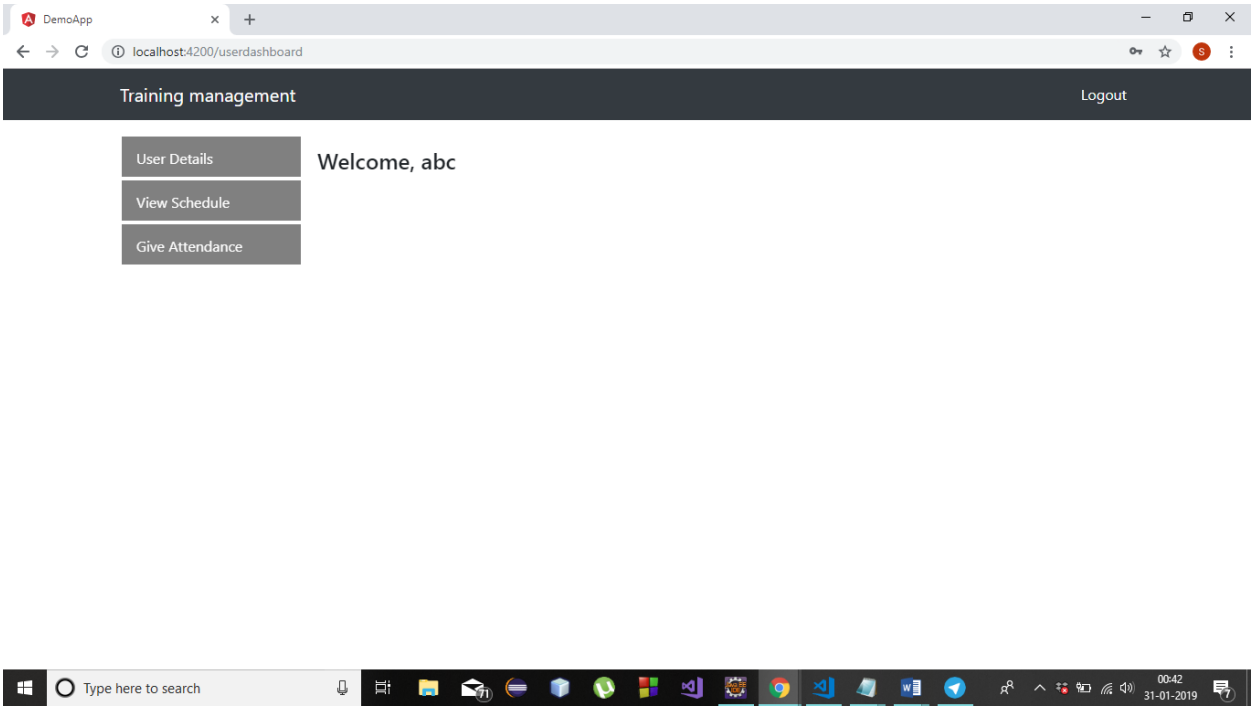
Login Page:



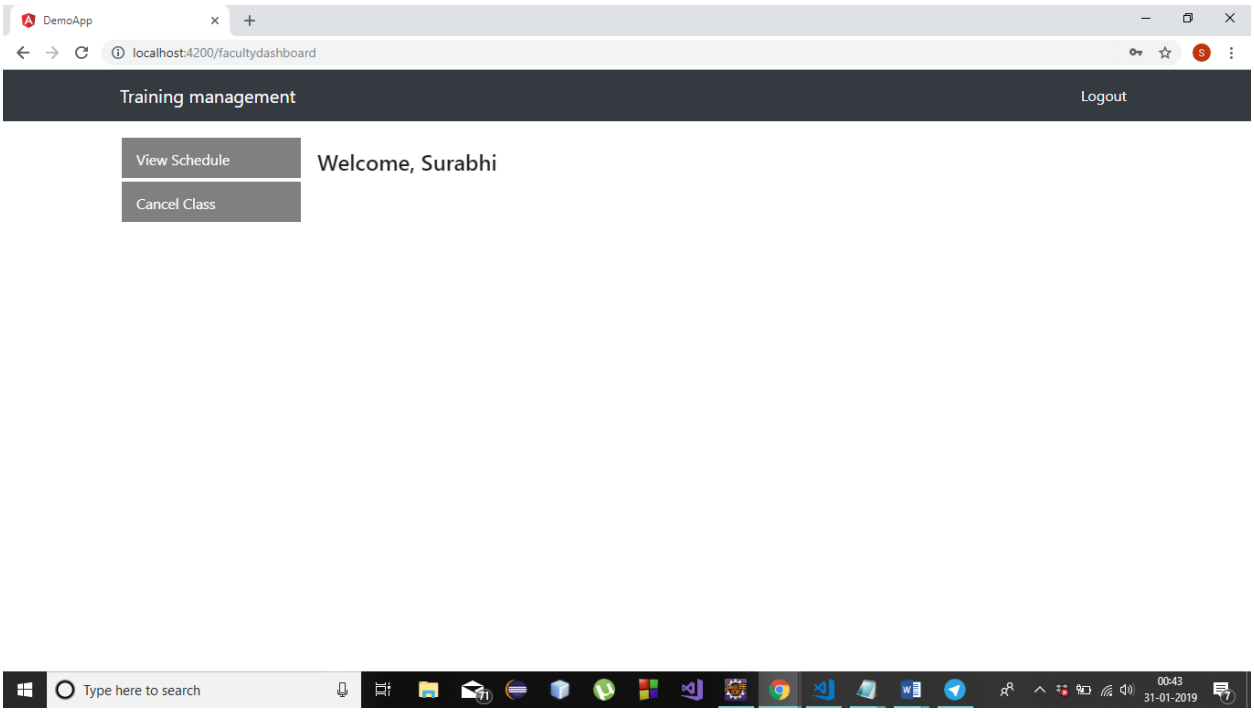
Admin Dashboard:



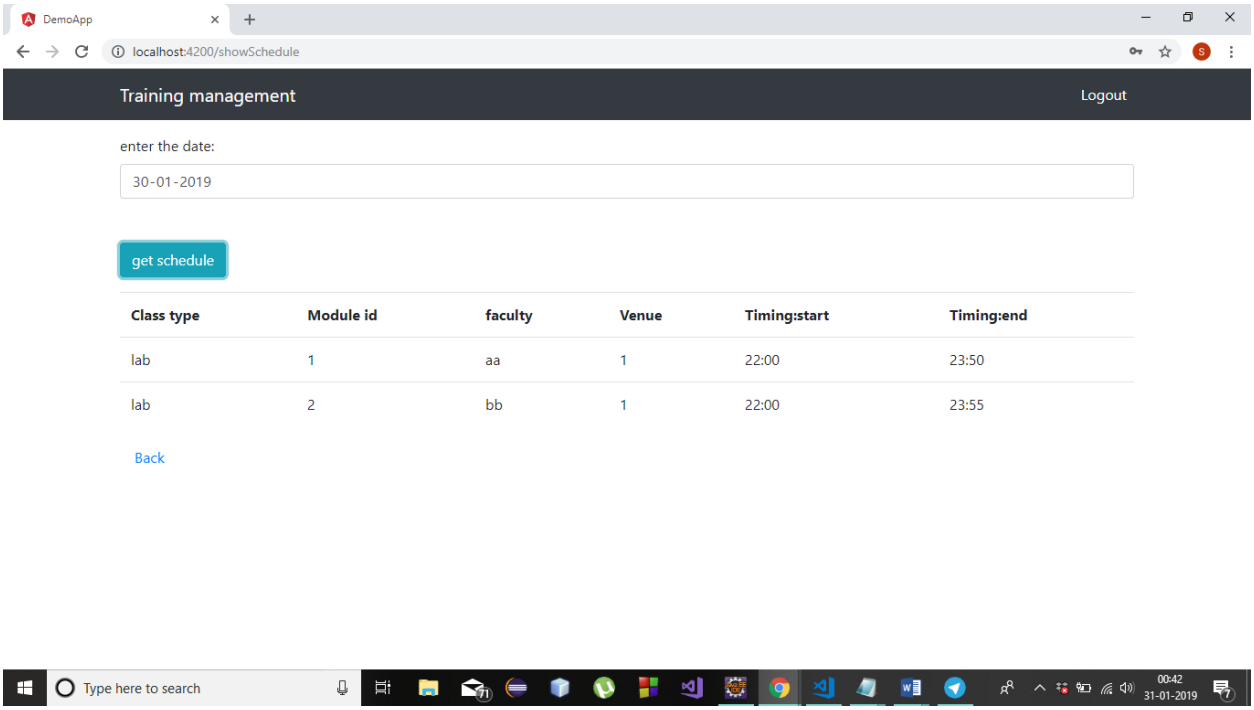
Student Dashboard:



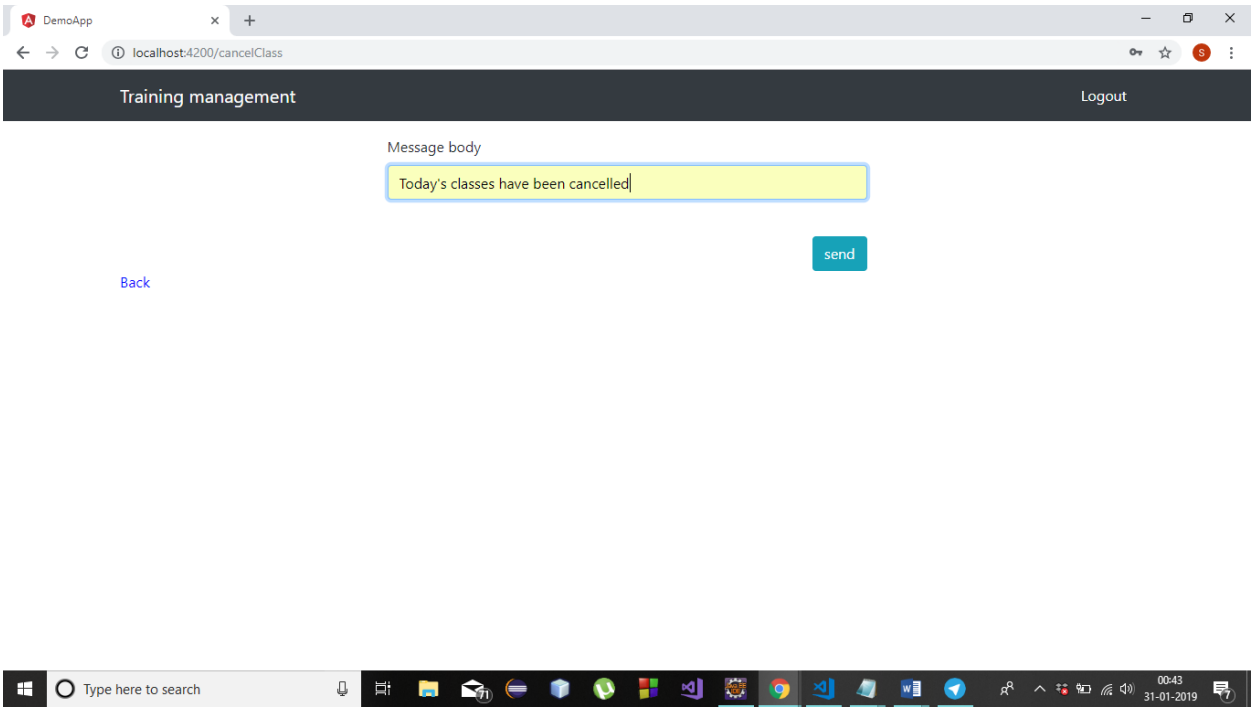
Faculty Dashboard:



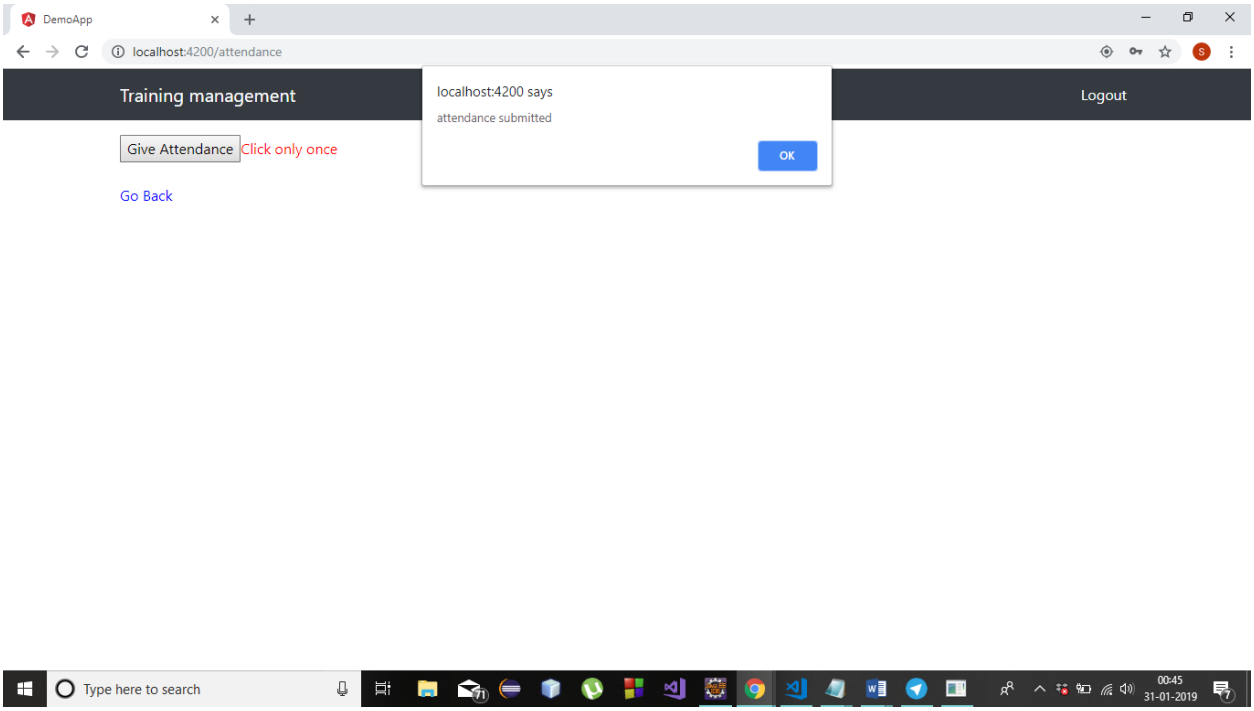
View Schedule:

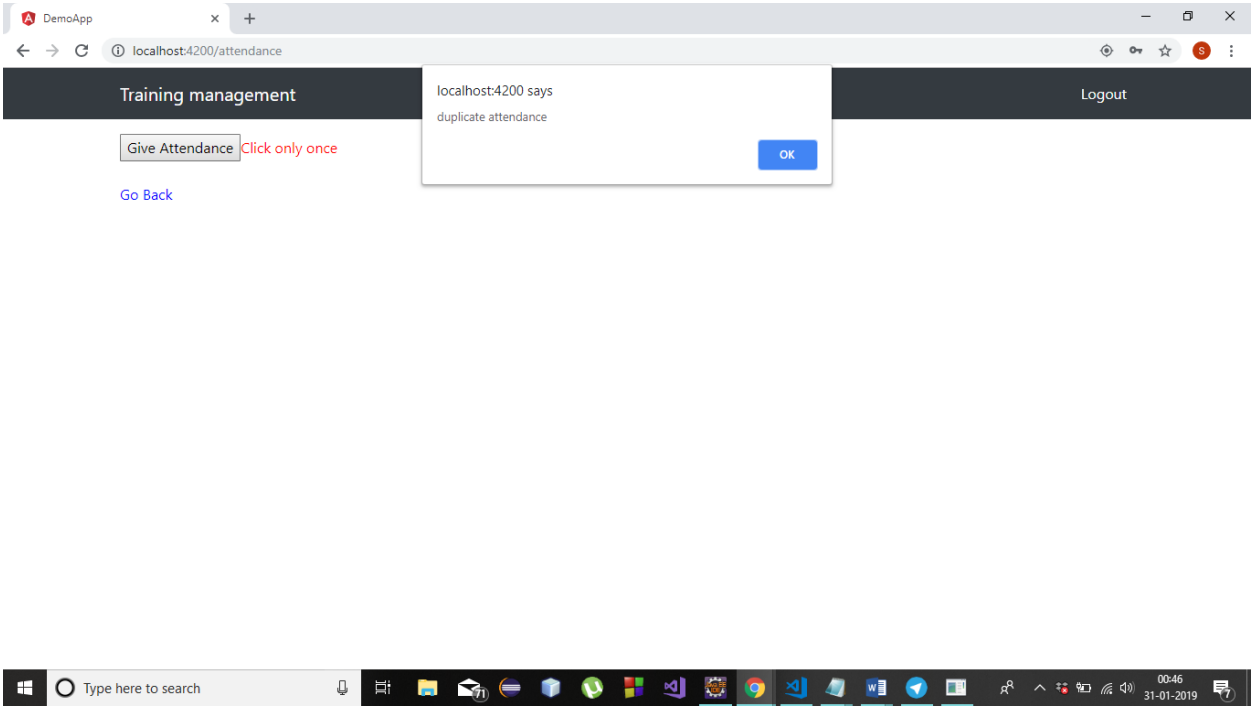


Cancel Class Page:



Attendance Page:





11. Testing

11.1 Introduction:

Software testing is the process used to measure the quality of developed computer software. Usually, quality is constrained to such topics as correctness, completeness, security, but can also include more technical requirements as described under the ISO standard ISO 9126, such as capability, reliability, efficiency, portability, maintainability, compatibility, and usability. Testing is a process of technical investigation, performed on behalf of stakeholders, that is intended to reveal quality-related information about the product with respect to the context in which it is intended to operate.

11.2 Test Plan:

The testing phase requires organizations to complete various tests to ensure the accuracy of programmed code, the inclusion of expected functionality, and the interoperability of applications and other network components. Thorough testing is critical to ensuring systems meet organizational and end-user requirements. Test plans created during initial project phases enhance an organization's ability to create detailed tests.

A bottom-up approach tests smaller components first and progressively adds and tests additional components and systems. A top-down approach first tests major components and connections and progressively tests smaller components and connections.

Bottom-up tests often begin with functional (requirements based) testing. Functional tests should ensure that expected functional, security, and internal control features are present and operating properly. Testers then complete integration and end-to-end testing to ensure application and system components interact properly. Users then conduct acceptance tests to ensure systems meet defined acceptance criteria. Organizations should review and complete user, operator, and maintenance manuals during the testing phase. Additionally, they should finalize conversion, implementation, and training plans.

For testing of the website:

1. All the features of the website are tested by running each function available in the website.
2. The results of the tests conducted on the website are analyzed properly. Only after getting satisfactory results of testing the website can be uploaded on the network i.e. internet.

11.3 Test Report:

Sr. No	Test Case Title	Description	Expected Outcome	Error Message	Result
1	Login Page-Admin	If User Id=Admin ID, Password= Admin Password	If Validated goto admion dashboard If not redirect to same page	Validation Error message	Passed
2	Login Page –User	If User Id=User ID, Password= User Password	If Validated go to user dashboard If not redirect to same page	Validation Error message	Passed
3	Home page Displayed	Homepage displayed for everyone	Home Page Displayed	No Error	Passed
4	Add Course	Only Admin can access this to enter new details with his Admin login	course details updated	Validation Error message	Passed
5	Add user	Only Admin can access this to enter new new user details with his Admin login	New user details entered	Validation Error message	Passed
6	View users	displayed for admin	List of users displayed.	No Error	Passed

12. Future Enhancements

1. Attendance functionality can be implemented using Google's Geolocation API.
2. Multi-Lingual support could be provided.
3. Faculty could upload study material.
4. Feedback system can be added.

13. Bibliography

http://www.google.com ,

http://www.github.com
http://www.youtube.com