STUDENT INFORMATION SYSTEM

## A Project Report Submitted

**In Partial Fulfillment of the Requirements for the Degree of**

MASTER OF COMPUTER APPLICATION

## by

**Rishabh Jain**

**(1900290149082)**

**Under the Supervision of**

## Mr. Ankit Verma

**KIET Group of Institutions, Ghaziabad**



**to the**

## FACULTY OF MCA

**DR. APJ ABDUL KALAM TECHNICAL UNIVERSITY**

**(Formerly Uttar Pradesh Technical University) LUCKNOW**

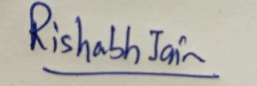
**July 2021**

**DECLARATION**

I hereby declare that the work presented in this report entitled “**Student Information System**", was carried out by me. I have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University or Institute.

I have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, experiments, results, that are not my original contribution. I have used quotation marks to identify verbatim sentences and given credit to the original authors/sources.

I affirm that no portion of my work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, I shall be fully responsible and answerable.



Name : Rishabh Jain Roll. No :1900290149082

Branch : Master of Computer Application

# CERTIFICATE

Certified that **Rishabh Jain** (**1900290149082**) has carried out the project work presented in this report entitled “**Student Information System**” for the award of **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University, Lucknow under my supervision. The report embodies result of original work, and studies are carried out by the student himself and the contents of the report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University.

**Mr. Ankit Verma External Examiner**

Assistant Professor

Dept. of Computer Application

KIET Group of Institutions, Ghaziabad

Date: 10/08/2021

# ABSTRACT

A (**Student** **Information System**) is a formal, sociotechnical, organizational system designed to collect, process, store, and distribute information. From a sociotechnical perspective, information systems are composed by four components: task, people, structure (or roles), and technology. Information systems can be defined as an integration of components for collection, storage and processing of data of which the data is used to provide information, contribute to knowledge as well as digital products that facilitate decision making.

Alter argues for advantages of viewing an information system as a special type of [work system](https://en.wikipedia.org/wiki/Work_system). A work system is a system in which humans or machines perform processes and activities using resources to produce specific products or services for customers. An information system is a work system whose activities are devoted to capturing, transmitting, storing, retrieving, manipulating and displaying information.

# ACKNOWLEDGEMENT

Success in life is never attained single handedly. My deepest gratitude goes to my Project supervisor, **Mr. Ankit Verma, Assistant Professor, Department of Computer Application** for his guidance, help and encouragement throughout my research work. Their enlightening ideas, comments, and suggestions. Words are not enough to express my gratitude for his insightful comments and administrative help at various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

**Rishabh Jain 1900290149082**

**TABLE OF CONTENT**

Page No.

Declaration ii

Certificate iii

Abstract iv

Acknowledgement v

Table of Content vi

**CHAPTER 1: INTRODUCTION**

1.1 Project Description 7

1.2 Project Scope 7

1.3 Hardware/software used in Project 8

**CHAPTER 2: LITERATURE REVIEW**

2.1 Abstract 9

2.2 Introduction 10-11

2.3 Scope 11

**CHAPTER 3: FEATURE & SCOPE**

3.1 Feature 12

3.2 Introduction to Salesforce 13

3.3 Lightning Web Components 14-15

3.4 Aura Components 15-16

3.5 Apex 17

3.6 Images 18-22

**CHAPTER 4: DESIGN**

4.1 DFD  23

**CHAPTER 5: CODING** 24-33

**CHAPTER 6: CONCLUSION**

5.1 Conclusion 34

**REFERENCES** 35-36

# CHAPTER 1

**INTRODUCTION**

## PROJECT DESCRIPTION

The objective of Student information System is to allow the administrator of any organization to edit and find out the personal details of a student and allows the student to keep up to date his profile. It’ll also facilitate keeping all the records of students, such as their id, name, mailing address, phone number, DOB etc. So, all the information about a student will be available in a few seconds.

Overall, it’ll make Student Information Management an easier job for the administrator and the student of any organization. The main purpose of this SRS document is to illustrate the requirements of the project Student information System and is intended to help any organization to maintain and manage its student’s personal data.

Student Information Management System can be used by education institutes to maintain the records of students easily. Achieving this objective is difficult using a manual system as the information is scattered, can be redundant and collecting relevant information may be very time consuming.

Information technology departments in larger organizations tend to strongly influence the development, use, and application of information technology in the business. A series of methodologies and processes can be used to develop and use an information system. Many developers use a system engineering approach such as the [system development life cycle](https://en.wikipedia.org/wiki/System_development_life_cycle) (SDLC), to systematically develop an information system in stages.

## Project Scope

Similar to computer science, other disciplines can be seen as both related and foundation disciplines of IS. The domain of study of IS involves the study of theories and practices related to the work, and social and technological phenomena, which determine the development, use, and effects of discipline information systems in organizations and society.

An additional benefit—though a minor one for most people—is that when you are with other people they are not able to tell what card you have. Users with low credit scores and credit cards with low limits and high APRs might not want, say, an interviewer or date to know these things, and mobile payments offer an additional level of personal privacy.

## 1.3 Hardware/software used in Project

**1.3.1 Technologies / Software Requirements**

* Operating System: Windows
* Programming language: JavaScript, Html, Apex
* IDE: Visual Studio Code

**1.3.2 Hardware requirement / Hardware Used**

* Processor – i3
* Hard Disk – 120 GB
* Memory – 4GB RAM

# CHAPTER 2

**LITERATURE REVIEW**

## Abstract

Internet has become a busy hub for searching information and doing different tasks virtually that used to be done manually before the internet age. There are enormous numbers of mobile and web applications that have made it easier to do different tasks. A big part of our everyday task can be done on the internet at the current age. Faster internet along with fast performing devices demands faster applications. A trend is growing to shift software or applications we used to use in desktop machines to the web. There are plenty of applications that are usable from web and mobile devices. Several JavaScript based frameworks and libraries are used to develop different applications.

JavaScript is widely popular among developer communities because of its simplicity and easy but effective developing process. JavaScript makes it easier to create interactive user interfaces. It efficiently updates through rendering the exact components to the view of each state and makes the data changes in the application. In JS, every component manages their own state and composes them to the user interfaces. This concept of components instead of templates in JavaScript, plenty of data can easily be passed to the app and thus keep the state out of the DOM. Using Node React can also be rendered on the server side. Alongside web apps, to build mobile applications we can use React Native as well. The purpose of the thesis is to carry out a

in-depth research of the ReactJS library based on JavaScript. The fundamental concepts, characteristics, features, development processes, core architecture and market research as well as compatibility will be covered in the thesis.

## Introduction

New web projects are commonly built using JavaScript in one form or another, and it is being broadly adopted across the industry. [1] Developers and engineers are choosing JavaScript because it allows spending more time to focus more on the product development and less time spent on fighting and learning to the framework.[2]

An application is a collection of discrete components, each representing a single view. The idea of every individual view component makes it easy to iterate on product development because to make changes on a single view or component, it is not necessary to consider the entire system.[3] When an application is built with React, the code is generally predictable, it is because react wraps the DOM mutative, imperative API with a declarative one, which raises the level of abstraction and simplifies the programming model.[4]

Moreover, it is easier to scale the application built with react Other issues that require copy, for processing. The combination of React and the rapid iteration cycle of the web, has enabled to make some excellent products including many Facebook components.[5] An amazing JavaScript framework called Relay has also been made on top of React, which helps simplifying data fetching on a large scale.[6]

The purpose of the design phase is to develop a clear understanding of what the developer wants people to gain from his/her project. As you the developer works on the project, the test for every design decision should be "Does this feature fulfill the ultimate purpose of the project?".

A purpose statement will affect the design process by explaining what the developer wants the project to do, rather than describing the project itself.[7]

The Design Document will verify that the current design meets all of the explicit requirements contained in the system model as well as the implicit requirements desired by the customer.[8]

This concept of components instead of templates in JavaScript, plenty of data can easily be

passed to the app and thus keep the state out of the DOM. [9]

We are going to develop a secured database. There are different categories of users namely

Administrator, Student who will be viewing either all or some specific information form the

database.[10]

Depending upon the category of user the access rights are decided. It means if the user is an administrator, then he can be able to modify the data, append etc. All other users only have the rights to retrieve the information about database.[11]

When an application is built with React, the code is generally predictable, it is because react wraps the DOM mutative, imperative API with a declarative one, which raises the level of abstraction and simplifies the programming model.[12] An application is a collection of discrete components, each representing a single view. The idea of every individual view component makes it easy to iterate on product development because to make changes on a single view or component, it is not necessary to consider the entire system.[13]

There are plenty of applications that are usable from web and mobile devices. Several JavaScript based frameworks and libraries are used to develop different applications.[14]

Many developers use a system engineering approach such as the [system development life cycle](https://en.wikipedia.org/wiki/System_development_life_cycle) (SDLC), to systematically develop an information system in stages.[15]

## Scope

A **student information system** (SIS) is a software application for educational establishments to manage student data. Student information systems provide capabilities for entering student test and other assessment scores, building student schedules, tracking student attendance, and managing many other student-related data needs in a school, college or university.

**CHAPTER 3**

**FEATURE & IMAGES**

**3.1 Feature**

Our easy-to-use, integrated College administration applications are proven to reduce time spent on administrative tasks so you can concentrate on raising student achievement. Student Information System have to accept, process and generate reports accurately and any point of time any user can get the student information.

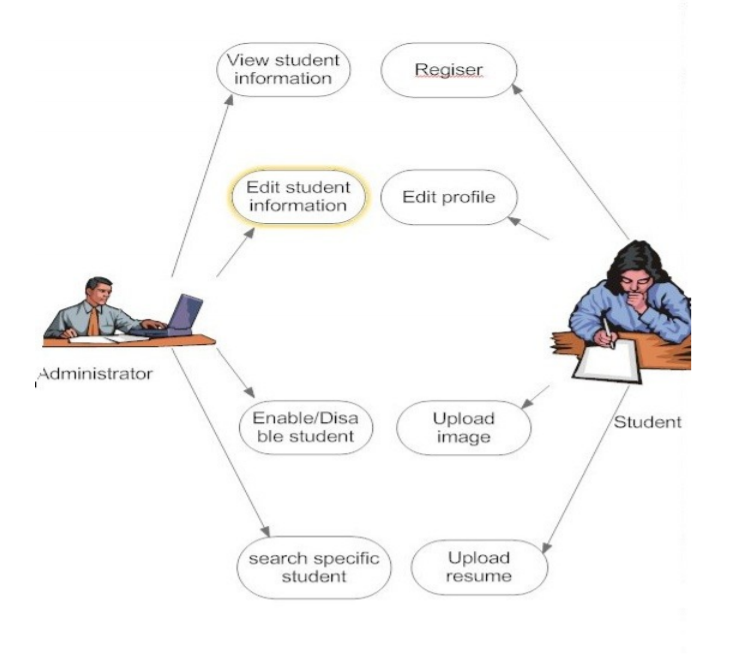


Figure 1

**3.2 INTRODUCTION TO SALESFORCE**

Salesforce.com, Inc. is an American cloud-based software company headquartered in San Francisco, California. It provides customer relationship management service and also provides a complementary suite of enterprise applications focused on customer service, marketing automation, analytics, and application development.

Salesforce is a leading CRM (**C**ustomer **R**elationship **M**anagement) software which is served form cloud. It has more than 800 applications to support various features like generating new leads, acquiring new leads, increasing sales and closing the deals. It is designed to manage the organization's data focused on customer and sales details. It also offers features to customize its inbuilt data structures and GUI to suit the specific needs of a business.

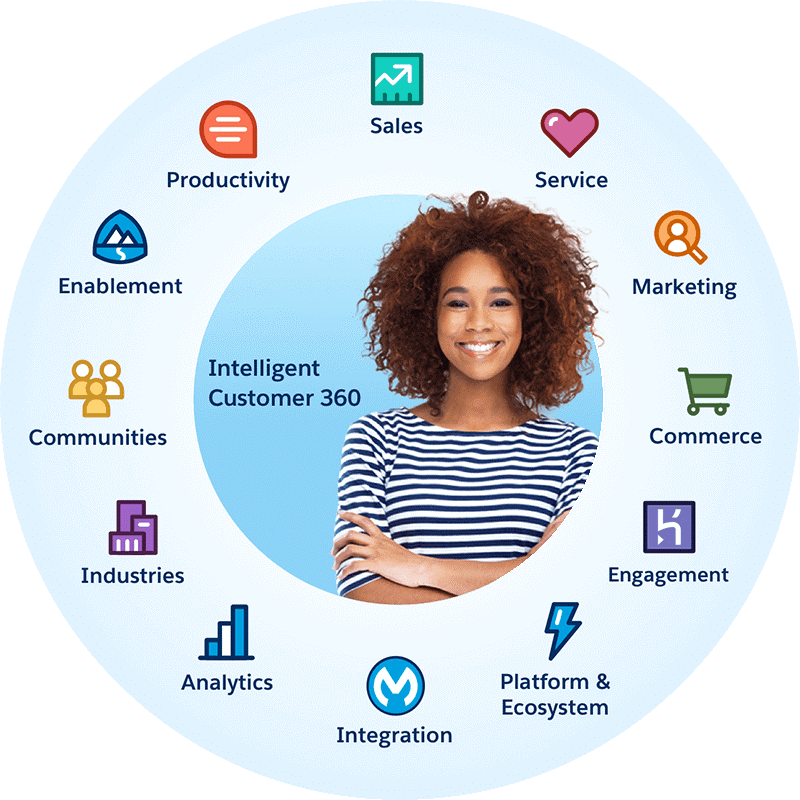


Figure 2

**3.3 LIGHTNING WEB COMPONENTS**

We can build Lightning components using two programming models: Lightning Web Components, and the original model, Aura Components. Lightning web components are custom HTML elements built using HTML and modern JavaScript. Lightning web components and Aura components can coexist and interoperate on a page. To admins and end users, they both appear as Lightning components.

Lightning Web Components uses core [Web Components](https://github.com/w3c/webcomponents/) standards and provides only what’s necessary to perform well in browsers supported by Salesforce. Because it’s built on code that runs natively in browsers, Lightning Web Components is lightweight and delivers exceptional performance. Most of the code you write is standard JavaScript and HTML.

Salesforce is committed to developing open web standards and is a member of the [World Wide Web Consortium (W3C)](https://www.w3.org/Consortium/Member/List).Salesforce developers are contributing members of the Ecma International Technical Committee 39 ([TC39](https://tc39.github.io/ecma262/)), which is the committee that evolves JavaScript.

Base Lightning components are available as Aura components and as Lightning web components. The [Component Reference](https://developer.salesforce.com/docs/component-library/overview/components) includes documentation, specifications, and examples for both. See [Base Components: Aura Vs Lightning Web Components](https://developer.salesforce.com/docs/component-library/documentation/en/migrate_map_aura_lwc_components.html) for differences between them.

A Lightning web component that renders UI must include an HTML file, a JavaScript file, and a metadata configuration file. The files must use the same name so the framework can auto wire them. A service component (library) must include a JavaScript file and a metadata configuration file.

LWC is a new programming model levering the recent web standards. Rather than being a totally custom and development wise rigid framework, It’s quite flexible. It’s mostly the common Web Standards and a Thin Layer of Specialized services to make it a perfect fit for Modern Rich UI Implementations in Salesforce. This thin layer of specialized services contains Base Lightning Components, Lightning Data Service and User Interface API which work behind the curtain for LWC.

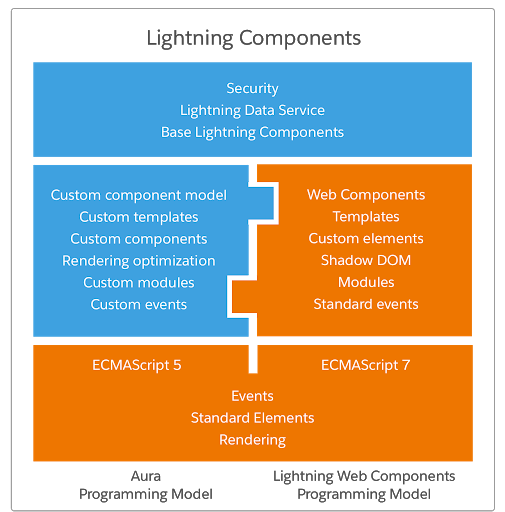


Figure 3

**3.4 AURA COMPONENT**

Aura components are the self-contained and reusable units of an app. They represent a reusable section of the UI, and can range in granularity from a single line of text to an entire app.

The framework includes a set of prebuilt components. For example, components that come with the Lightning Design System styling are available in the lightning namespace. These components are also known as the base Lightning components. You can assemble and configure components to form new components in an app. Components are rendered to produce HTML DOM elements within the browser.

A component can contain other components, as well as HTML, CSS, JavaScript, or any other Web-enabled code. This enables you to build apps with sophisticated UIs.

The details of a component's implementation are encapsulated. This allows the consumer of a component to focus on building their app, while the component author can innovate and make changes without breaking consumers. You configure components by setting the named attributes that they expose in their definition. Components interact with their environment by listening to or publishing events.

**3.5 APEX**

Apex is a strongly typed, object-oriented programming language that allows developers to execute flow and transaction control statements on Salesforce servers in conjunction with calls to the API. Using syntax that looks like Java and acts like database stored procedures, Apex enables developers to add business logic to most system events, including button clicks, related record updates, and Visualforce pages. Apex code can be initiated by Web service requests and from triggers on objects.

Apex is based on familiar Java idioms, such as variable and expression syntax, block and conditional statement syntax, loop syntax, object and array notation. Where Apex introduces new elements, it uses syntax and semantics that are easy to understand and encourage efficient use of the Lightning Platform. Therefore, Apex produces code that is both succinct and easy to write.

Apex is a strongly typed language that uses direct references to schema objects such as object and field names. It fails quickly at compile time if any references are invalid. It stores all custom field, object, and class dependencies in metadata to ensure that they are not deleted while required by active Apex code.

Apex provides built-in support for unit test creation and execution. It includes test results that indicate how much code is covered, and which parts of your code could be more efficient. Salesforce ensures that all custom Apex code works as expected by executing all unit tests prior to any platform upgrades.

Apex is a strongly-typed language, that is, you must declare the data type of a variable when you first refer to it. Apex data types include basic types such as Integer, Date, and Boolean, as well as more advanced types such as lists, maps, objects and sObjects.

Variables are declared with a name and a data type. You can assign a value to a variable when you declare it.

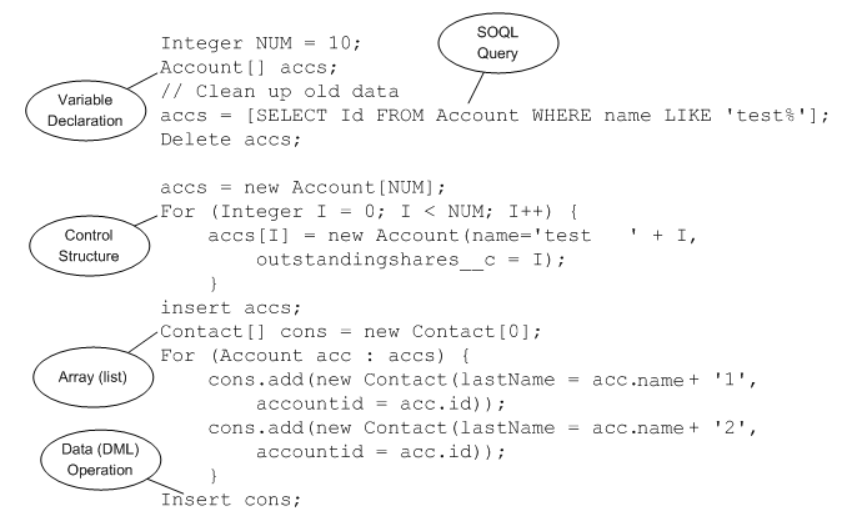


Figure 4

You can add, edit, or delete Apex using the Salesforce user interface only in a Developer Edition organization, a Salesforce Enterprise Edition trial organization, or sandbox organization. In a Salesforce production organization, you can change Apex only by using the Metadata API deploy call, the Salesforce Extensions for Visual Studio Code, or the Ant Migration Tool.

**3.6 IMAGES**

Students can edit their profiles by using the edit option. The above page is used for editing student information.

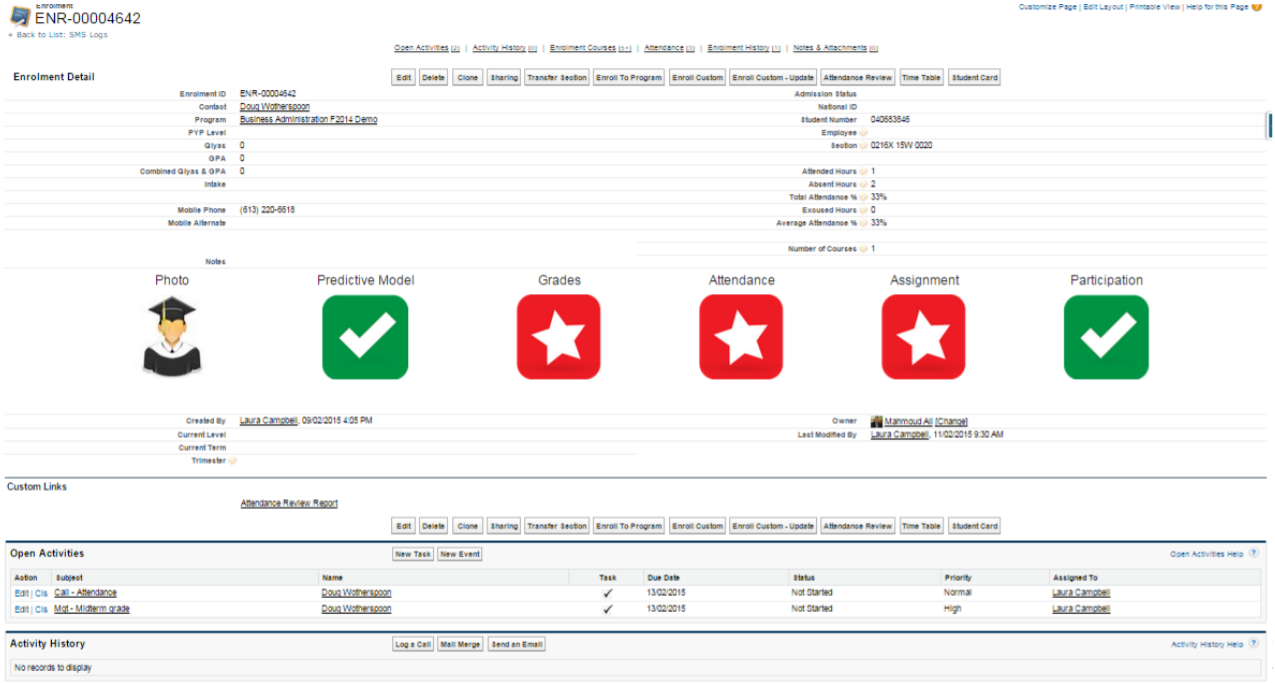


Figure 5

Students can edit their profiles by using the edit option. The above page is used for editing student information.

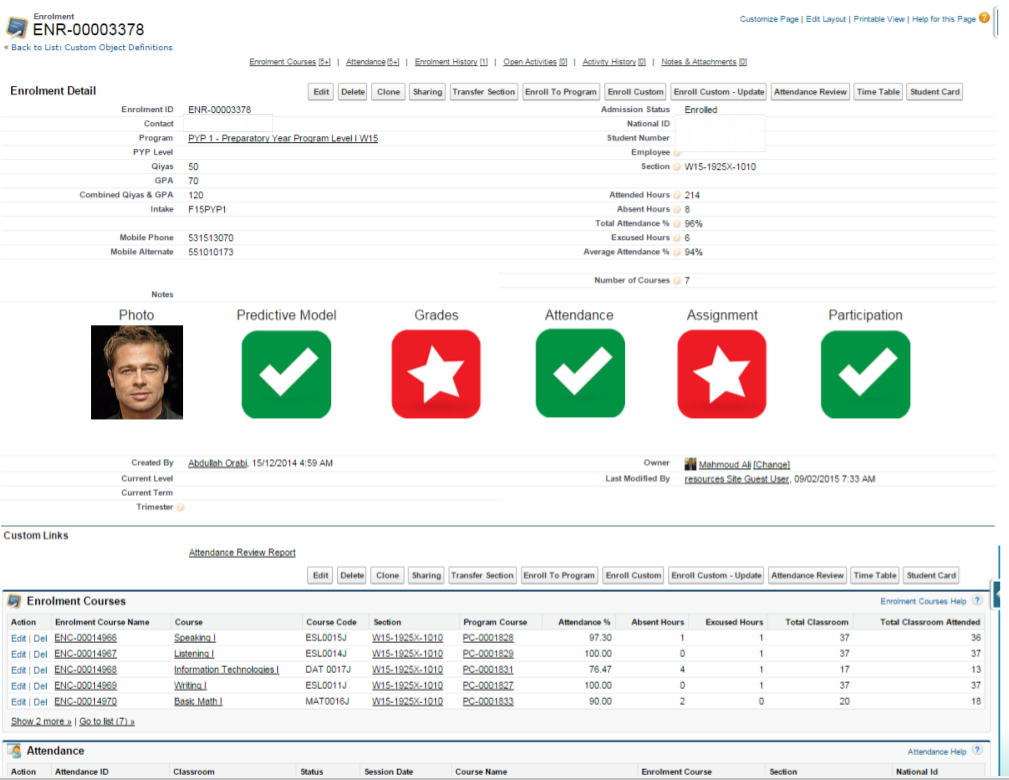


Figure 6

This is student’s Dashboard.

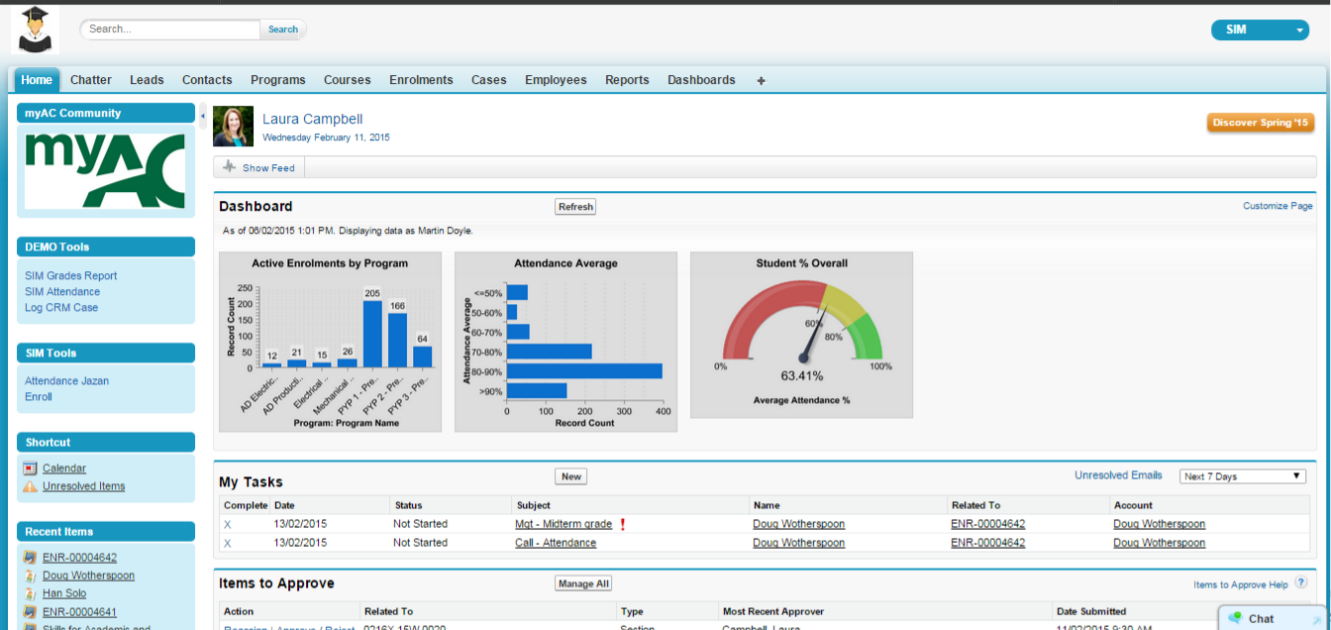


Figure 7

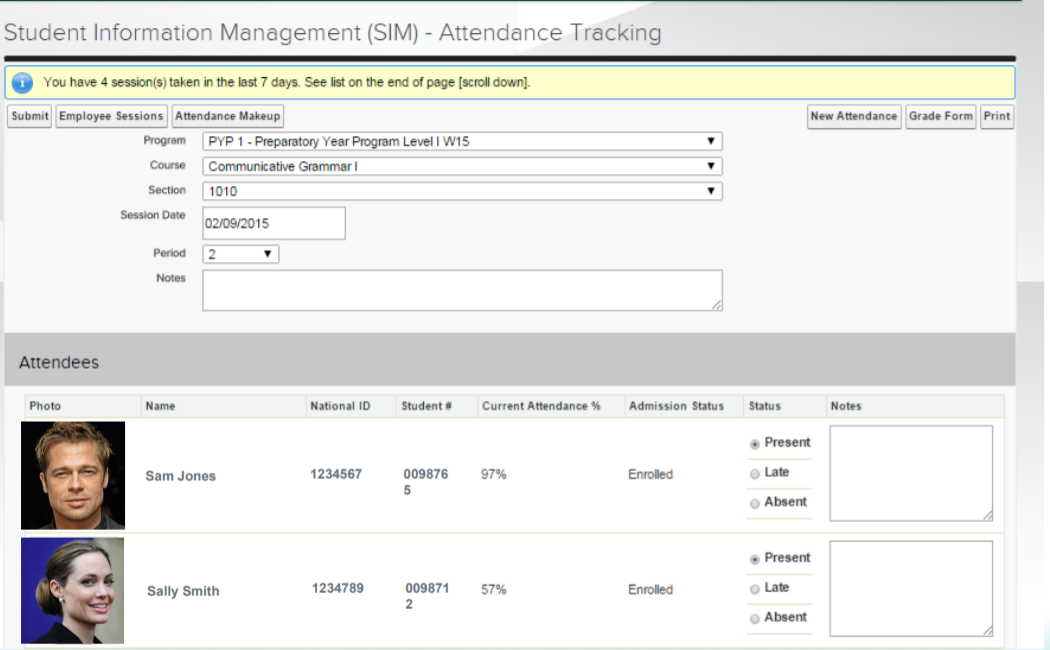


Figure 8

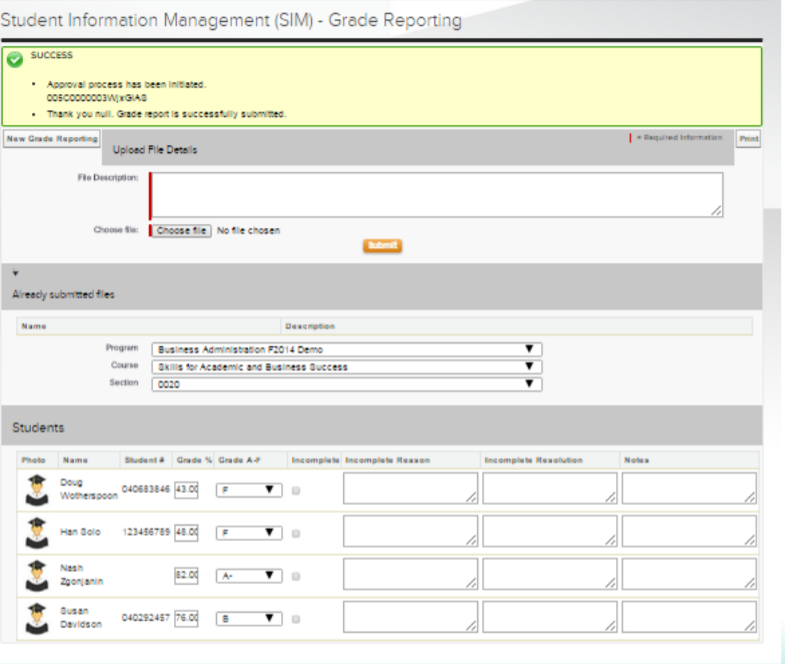


Figure 9

# CHAPTER 4

**DESIGN**

**4.1 DFD**

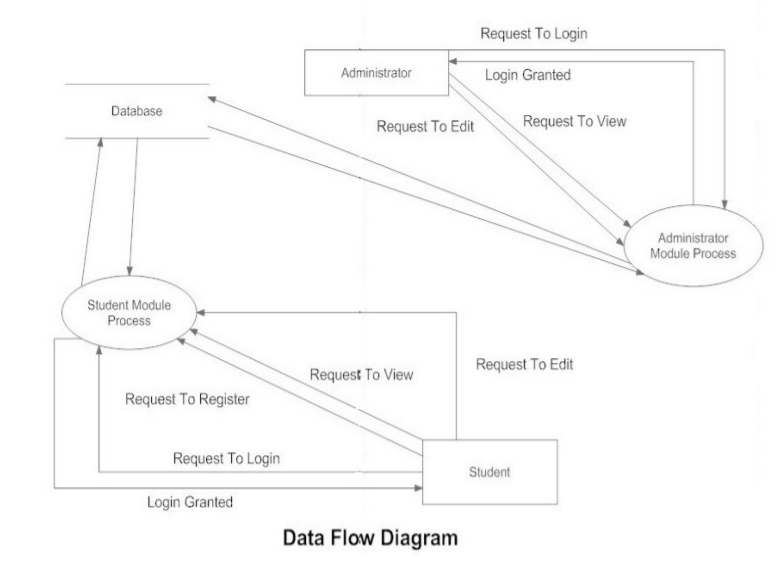


Figure 10

**CHAPTER 5**

**CODING**

**5.1 MaintainanceRequirement.trigger**

trigger MaintenanceRequest on Case (before update, after update) {

Map<Id,Case> validCaseMap = new Map<Id,Case>();

if(Trigger.isUpdate && Trigger.isAfter){

for(Case caseHere: Trigger.new){

if (caseHere.IsClosed && (caseHere.Type.equals('Repair') || caseHere.Type.equals('Routine Maintenance'))){

validCaseMap.put(caseHere.Id, caseHere);

}

}

if(!validCaseMap.values().isEmpty()){

MaintenanceRequestHelper.createNewRequest(validCaseMap);

}

}

}

**5.2 MaintainanceRequestHelper.cls**

public class MaintenanceRequestHelper {

public static void createNewRequest(Map<Id, Case> validCaseMap){

List<Case> newCases = new List<Case>();

Map<Id, Integer> productMaintenanceCycleMap = new Map<Id, Integer>();

Map<Id, Integer> workPartMaintenanceCycleMap = new Map<Id, Integer>();

for (Product2 productHere : [select Id, Maintenance\_Cycle\_\_c from Product2]) {

if (productHere.Maintenance\_Cycle\_\_c != null) {

productMaintenanceCycleMap.put(productHere.Id, Integer.valueOf(productHere.Maintenance\_Cycle\_\_c));

}

}

for (Work\_Part\_\_c workPart : [select Id, Equipment\_\_c, Maintenance\_Request\_\_c from Work\_Part\_\_c where Maintenance\_Request\_\_c in :validCaseMap.keySet()]) {

if (workPart.Equipment\_\_c != null) {

if(!workPartMaintenanceCycleMap.containsKey(workPart.Maintenance\_Request\_\_c)){

workPartMaintenanceCycleMap.put(workPart.Maintenance\_Request\_\_c, productMaintenanceCycleMap.get(workPart.Equipment\_\_c));

}

else if(productMaintenanceCycleMap.get(workPart.Equipment\_\_c) < workPartMaintenanceCycleMap.get(workPart.Maintenance\_Request\_\_c)){

workPartMaintenanceCycleMap.put(workPart.Maintenance\_Request\_\_c, productMaintenanceCycleMap.get(workPart.Equipment\_\_c));

}

}

}

for(Case caseHere: validCaseMap.values()){

Case newCase = new Case();

newCase.Vehicle\_\_c = caseHere.Vehicle\_\_c;

newCase.Equipment\_\_c = caseHere.Equipment\_\_c;

newCase.Type = 'Routine Maintenance';

newCase.Subject = String.isBlank(caseHere.Subject) ? 'Routine Maintenance Request' : caseHere.Subject + ' New';

newCase.Date\_Reported\_\_c = Date.today();

newCase.Date\_Due\_\_c = workPartMaintenanceCycleMap.containsKey(caseHere.Product\_\_c) ? Date.today().addDays(workPartMaintenanceCycleMap.get(caseHere.Product\_\_c)) : Date.today();

newCase.Status = 'New';

newCase.Product\_\_c = caseHere.Product\_\_c;

newCase.AccountId = caseHere.AccountId;

newCase.ContactId = caseHere.ContactId;

newCase.AssetId = caseHere.AssetId;

newCase.Origin = caseHere.Origin;

newCase.Reason = caseHere.Reason;

newCases.add(newCase);

}

if(newCases.size() > 0){

insert newCases;

}

}

}

**5.3 CalloutService.cls**

public with sharing class WarehouseCalloutService {

private static final String WAREHOUSE\_URL = 'https://th-superbadge-apex.herokuapp.com/equipment';

// complete this method to make the callout (using @future) to the

// REST endpoint and update equipment on hand.

@future(callout=true)

public static void runWarehouseEquipmentSync(){

Http http = new Http();

HttpRequest request = new HttpRequest();

request.setEndpoint(WAREHOUSE\_URL);

request.setMethod('GET');

HttpResponse response = http.send(request);

if (response.getStatusCode() == 200) {

List<Object> results = (List<Object>) JSON.deserializeUntyped(response.getBody());

List<Product2> equipmentList = new List<Product2>();

for (Object record: results) {

Map<String, Object> recordMap = (Map<String, Object>)record;

Product2 equipment = new Product2();

equipment.Name = (String)recordMap.get('name');

equipment.Cost\_\_c = (Decimal)recordMap.get('cost');

equipment.ProductCode = (String)recordMap.get('\_id');

equipment.Current\_Inventory\_\_c = (Integer)recordMap.get('quantity');

equipment.Maintenance\_Cycle\_\_c = (Integer)recordMap.get('maintenanceperiod');

equipment.Replacement\_Part\_\_c = (Boolean)recordMap.get('replacement');

equipment.Lifespan\_Months\_\_c = (Integer)recordMap.get('lifespan');

equipment.Warehouse\_SKU\_\_c = (String)recordMap.get('sku');

equipmentList.add(equipment);

}

if(equipmentList.size() > 0){

upsert equipmentList;

}

}

}

}

**5.4 SyncShedule.cls**

public class WarehouseSyncSchedule implements Schedulable{

// implement scheduled code here

public void execute(System.SchedulableContext context){

WarehouseCalloutService.runWarehouseEquipmentSync();

}

}

**5.5 DataService.cls**

public with sharing class BoatDataService {

public static final String LENGTH\_TYPE = 'Length';

public static final String PRICE\_TYPE = 'Price';

public static final String TYPE\_TYPE = 'Type';

@AuraEnabled(cacheable=true)

public static List<Boat\_\_c> getBoats(String boatTypeId) {

// Without an explicit boatTypeId, the full list is desired

String query = 'SELECT '

+ 'Name, Description\_\_c, Geolocation\_\_Latitude\_\_s, '

+ 'Geolocation\_\_Longitude\_\_s, Picture\_\_c, Contact\_\_r.Name, '

+ 'BoatType\_\_c, BoatType\_\_r.Name, Length\_\_c, Price\_\_c '

+ 'FROM Boat\_\_c';

if (String.isNotBlank(boatTypeId)) {

query += ' WHERE BoatType\_\_c = :boatTypeId';

}

query += ' WITH SECURITY\_ENFORCED ';

return Database.query(query);

}

@AuraEnabled(cacheable=true)

public static List<Boat\_\_c> getSimilarBoats(Id boatId, String similarBy) {

List<Boat\_\_c> similarBoats = new List<Boat\_\_c>();

List<Boat\_\_c> parentBoat = [SELECT Id, Length\_\_c, Price\_\_c, BoatType\_\_c, BoatType\_\_r.Name

FROM Boat\_\_c

WHERE Id = :boatId

WITH SECURITY\_ENFORCED];

if (parentBoat.isEmpty()) {

return similarBoats;

}

if (similarBy == LENGTH\_TYPE) {

similarBoats = [

SELECT Id, Contact\_\_r.Name, Name, BoatType\_\_c, BoatType\_\_r.Name, Length\_\_c, Picture\_\_c, Price\_\_c, Year\_Built\_\_c

FROM Boat\_\_c

WHERE Id != :parentBoat.get(0).Id

AND (Length\_\_c >= :parentBoat.get(0).Length\_\_c / 1.2)

AND (Length\_\_c <= :parentBoat.get(0).Length\_\_c \* 1.2)

WITH SECURITY\_ENFORCED

ORDER BY Length\_\_c, Price\_\_c, Year\_Built\_\_c

];

} else if (similarBy == PRICE\_TYPE) {

similarBoats = [

SELECT Id, Contact\_\_r.Name, Name, BoatType\_\_c, BoatType\_\_r.Name, Length\_\_c, Picture\_\_c, Price\_\_c, Year\_Built\_\_c

FROM Boat\_\_c

WHERE Id != :parentBoat.get(0).Id

AND (Price\_\_c >= :parentBoat.get(0).Price\_\_c / 1.2)

AND (Price\_\_c <= :parentBoat.get(0).Price\_\_c \* 1.2)

WITH SECURITY\_ENFORCED

ORDER BY Price\_\_c, Length\_\_c, Year\_Built\_\_c

];

} else if (similarBy == TYPE\_TYPE) {

similarBoats = [

SELECT Id, Contact\_\_r.Name, Name, BoatType\_\_c, BoatType\_\_r.Name, Length\_\_c, Picture\_\_c, Price\_\_c, Year\_Built\_\_c

FROM Boat\_\_c

WHERE Id != :parentBoat.get(0).Id

AND (BoatType\_\_c = :parentBoat.get(0).BoatType\_\_c)

WITH SECURITY\_ENFORCED

ORDER BY Price\_\_c, Length\_\_c, Year\_Built\_\_c

];

}

return similarBoats;

}

@AuraEnabled(cacheable=true)

public static List<BoatType\_\_c> getBoatTypes() {

return [SELECT Name, Id FROM BoatType\_\_c WITH SECURITY\_ENFORCED ORDER BY Name];

}

@AuraEnabled

public static List<BoatReview\_\_c> getAllReviews(Id boatId) {

return [

SELECT

Id,

Name,

Comment\_\_c,

Rating\_\_c,

LastModifiedDate,

CreatedDate,

CreatedBy.Name,

CreatedBy.SmallPhotoUrl,

CreatedBy.CompanyName

FROM

BoatReview\_\_c

WHERE

Boat\_\_c =:boatId

WITH SECURITY\_ENFORCED

ORDER BY

CreatedDate DESC

];

}

// Assume this may be an API that return this data, not a SOQL query

@AuraEnabled(cacheable=true)

public static String getBoatsByLocation(Decimal latitude, Decimal longitude, String boatTypeId) {

// Without an explicit boatTypeId, the full list is desired

String query = 'SELECT Name, Geolocation\_\_Latitude\_\_s, Geolocation\_\_Longitude\_\_s FROM Boat\_\_c ';

if (String.isNotBlank(boatTypeId)) {

query += 'WHERE BoatType\_\_c = :boatTypeId ';

}

query += ' WITH SECURITY\_ENFORCED ORDER BY DISTANCE(Geolocation\_\_c, GEOLOCATION(:latitude, :longitude), \'mi\') LIMIT 10';

return JSON.serialize(Database.query(query));

}

}

**5.6 ContactController.cls**

public with sharing class ContactController {

public ContactController() {

}

@AuraEnabled(cacheable=true)

public static List<contact> getContacts() {

throw new AuraHandledException('Forced error');

/\* return [

SELECT FirstName, LastName, Email

FROM Contact

WITH SECURITY\_ENFORCED

ORDER BY LastName

];\*/

}

}

**5.7 GenerateData.cls**

@isTest

public class GenerateDataTests {

@isTest

static void test\_initialization\_works() {

Test.startTest();

GenerateData.init();

Test.stopTest();

// The following simple counts are fine

System.assert([SELECT count() from Boat\_\_c] > 0, 'No Boat\_\_c were found');

System.assert([SELECT count() from BoatType\_\_c] > 0, 'No BoatType\_\_c were found');

System.assert([SELECT count() from Account] == 1, 'No Account was found');

System.assert([SELECT count() from Contact] > 0, 'No Contacts found');

}

@isTest

static void test\_getLengthForBoatType\_NoType(){

Integer boatLength = GenerateData.getLengthForBoatType('');

System.assertEquals(20, boatLength, 'The boat length was calculated wrongly.');

}

@isTest

static void test\_getPriceForBoatType\_NoType(){

Integer boatPrice = GenerateData.getPriceForBoatType('');

System.assertEquals(15000, boatPrice, 'The boat price was calculated wrongly.');

}

}

**5.8 ContactCreator.html**

<template>

<lightning-card>

<lightning-record-form

object-api-name={objectApiName}

fields={fields}

onsuccess={handleSuccess}>

</lightning-record-form>

</lightning-card>

</template>

**5.9 ContactCreator.js**

import { LightningElement } from 'lwc';

import { ShowToastEvent } from 'lightning/platformShowToastEvent';

import CONTACT\_OBJECT from '@salesforce/schema/Contact';

import FIRSTNAME\_FIELD from '@salesforce/schema/Contact.FirstName';

import LASTNAME\_FIELD from '@salesforce/schema/Contact.LastName';

import EMAIL\_FIELD from '@salesforce/schema/Contact.Email';

export default class ContactCreator extends LightningElement {

objectApiName = CONTACT\_OBJECT;

fields = [FIRSTNAME\_FIELD, LASTNAME\_FIELD, EMAIL\_FIELD];

handleSuccess(event) {

const toastEvent = new ShowToastEvent({

title: "Contact created",

message: "Record ID: " + event.detail.id,

variant: "success"

});

this.dispatchEvent(toastEvent);

}

}

**5.10 ContactCretor.js-meta.xml**

<?xml version="1.0" encoding="UTF-8"?>

<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">

<apiVersion>48.0</apiVersion>

<isExposed>true</isExposed>

<targets>

<target>lightning\_\_AppPage</target>

<target>lightning\_\_RecordPage</target>

<target>lightning\_\_HomePage</target>

</targets>

</LightningComponentBundle>

**CHAPTER 6**

**CONCLUSION**

Student information management system led to a better organization structure since the

information management of the students is well structured and also lead to better as well

as efficient utilization of resources.

Student Information Management System can be used by education institutes to maintain

the records of students easily. Achieving this objective is difficult using a manual system

as the information is scattered, can be redundant and collecting relevant information may

be very time consuming. All these problems are solved using this project.

**REFERENCES**

1. Basu, Amiya K., Rajiv Lal, Venkataraman Srinivasan, and Richard Staelin. "Salesforce compensation plans: An agency theoretic perspective." *Marketing science* 4, no. 4 (1985): 267-291.
2. Yu, D., Chander, A., Islam, N., & Serikov, I. (2007). JavaScript instrumentation for browser security. ACM SIGPLAN Notices, *42*(1), 237-249. Yue, Chuan, and Haining Wang. "Characterizing insecure JavaScript practices on the web." In Proceedings of the 18th international conference on World wide web, pp. 961-970. 2009.
3. Patel, J. and Chouhan, A., 2016, October. An approach to introduce basics of Salesforce. com: A cloud service provider. In *2016 International Conference on Communication and Electronics Systems (ICCES)* (pp. 1-8). IEEE.
4. Patel, Saurabh, Sankalp Sharma, and Rohan Prasad. "MULTITENANT EFFECTIVE CRM APPLICATION USING SALESFORCE.".
5. Keel, Jonathan. "Salesforce. com Lightning Process Builder and Visual Workflow."
6. Robbins, Jennifer Niederst. Learning web design: A beginner's guide to HTML, CSS, JavaScript, and web graphics. " O'Reilly Media, Inc.", 2012.
7. Duckett, J., 2011. *Beginning html, xhtml, css, and javascript*. John Wiley & Sons
8. Meloni, Julie C. Sams teach yourself HTML, CSS, and JavaScript all in one. Pearson Education India, 2016.
9. Yin, Junjie. "Salesforce-Usability of Lightning Web Components." (2019).
10. Chowdhury, R.A. and Chowdhury, R.A., 2021. Salesforce Platform. *Building a Salesforce-Powered Front Office: A Quick-Start Guide*, pp.229-257.
11. Horeman, T., Rodrigues, S.P., van den Dobbelsteen, J.J., Jansen, F.W. and Dankelman, J., 2012. Visual force feedback in laparoscopic training. *Surgical endoscopy*, *26*(1), pp.242-248.
12. Churchill, G.A., Ford, N.M., Walker, O.C., Johnston, M.W. and Tanner, J.F., 1993. *Sales force management*. Homewood, IL: Irwin.
13. Edwin. "Evaluate corporate social responsibility disclosure at Annual Report Companies in multifarious group of industry members of Jakarta Stock Exchange (JSX), Indonesia." Social Responsibility Journal (2008).
14. Zakas NC. Professional JavaScript for web developers. John Wiley & Sons; 2009 Feb 9.
15. Pollock, J. (2013). JavaScript: a beginner's guide. McGraw-Hill Inc.