**Employee Management System**

[Objective: 3](#_Toc25655)

[Abstract: 3](#_Toc13693)

[Use Case Diagram 3](#_Toc21193)

[Users of Employee Management System 4](#_Toc18240)

[Functionality available to Admin 4](#_Toc17848)

[Functionality available to the Employee 4](#_Toc13289)

[Static Pages and other sections : 5](#_Toc29951)

[Modules in the Employee Management System 6](#_Toc30062)

[Employee Management Module : 6](#_Toc13563)

[Leave Module : 7](#_Toc20473)

[Timesheet Module : 7](#_Toc23937)

[Salary Module : 8](#_Toc5902)

[Login Module : 8](#_Toc2792)

[Search Module : 8](#_Toc14600)

[Key Features : 9](#_Toc4112)

[Employee 9](#_Toc11117)

[Login 9](#_Toc27115)

[Forgot Password 10](#_Toc26337)

[Dashboard 10](#_Toc18460)

[View Profile 11](#_Toc22471)

[Timesheet Entry 11](#_Toc7068)

[Add Timesheet 12](#_Toc4305)

[View Timesheet 12](#_Toc25482)

[Apply Leaves 12](#_Toc14082)

[View Leave Records 13](#_Toc29083)

[Admin 13](#_Toc6912)

[Login 13](#_Toc27317)

[Dashboard 13](#_Toc20892)

[Manage Employee 14](#_Toc11710)

[Add New Employee 14](#_Toc28160)

[View Listing of all Employees 15](#_Toc22963)

[Manage Timesheet 16](#_Toc20963)

[Add/Edit Employee Timesheet 16](#_Toc13679)

[View Timesheets for all Employees 17](#_Toc27815)

[Manage Leave 17](#_Toc3801)

[Add Leave for the Employee 17](#_Toc18532)

[Listing of the Leave for all Employees 18](#_Toc5199)

[Manage Salary 18](#_Toc31770)

[Add Salary for the Employee 18](#_Toc641)

[Listing of the Salaries for all Employees 19](#_Toc5285)

[Search Feature 19](#_Toc4640)

[Classes and their attributes 19](#_Toc21035)

[Classes and their methods 20](#_Toc11274)

[NFRs & SLAs: 20](#_Toc17943)

[Testing: 21](#_Toc24545)

[Other Considerations: 21](#_Toc32229)

[Weekly Milestones 22](#_Toc14369)

[Week 1: 22](#_Toc2374)

[1. Create Architecture of the Project with UML diagrams. 22](#_Toc8250)

[2. Create a Maven project using Eclipse. 22](#_Toc24164)

[3. Implement the Java application using the TDD approach 22](#_Toc29759)

[4. Create the Entity Relationship (ER) Diagram 22](#_Toc30035)

[5. Check code into Bitbucket 23](#_Toc20251)

[Week 2: 24](#_Toc31723)

[1. Refactor for error handling and usage of DAO classes 24](#_Toc4979)

[2. BDD Using Cucumber 24](#_Toc19876)

[3. Update Architecture Diagrams 24](#_Toc19617)

[4. Check code into Git 24](#_Toc30512)

[Week 3: 25](#_Toc12386)

[1. Update the Maven pom 25](#_Toc3034)

[2. Create REST APIs for the CRUD operations. 25](#_Toc3897)

[3. Refactor to use JPA with Hibernate for database operations 25](#_Toc762)

[4. Create a Performance Test Plan in JMeter 25](#_Toc12428)

[5. Update Architecture Diagrams 25](#_Toc17923)

[6. Check code into Git 25](#_Toc16647)

[Week 4: 26](#_Toc7150)

[1. Create a Spring Boot Starter Project with Maven 26](#_Toc27857)

[2. Create the REST Endpoints for the CRUD operations 26](#_Toc17681)

[3. Create a FreeStyle project in Jenkins to automate the workflow 26](#_Toc19593)

[4. Update Architecture Diagrams 26](#_Toc29350)

[5. Check code into Git 27](#_Toc17203)

[Week 5: 27](#_Toc24460)

[1. Create a React Project 27](#_Toc7287)

[2. Use HTML5 semantic elements. Style with CSS3. 28](#_Toc15364)

[3. Create the Dockerfile environments 28](#_Toc30173)

[4. Use Spring Cloud Microservices and Spring Boot Framework to make this application distributed. 28](#_Toc27779)

[5. Architecture Diagrams 28](#_Toc4846)

[6. Check code into Git 28](#_Toc23494)

# Objective:

Develop an end-to-end full-stack web application that is an **Employee Management System** using React, Spring Boot, Spring REST and Spring Data JPA.

Dockerize the application.

Transform application by Repackaging and Refactoring into Microservices.

# Abstract:

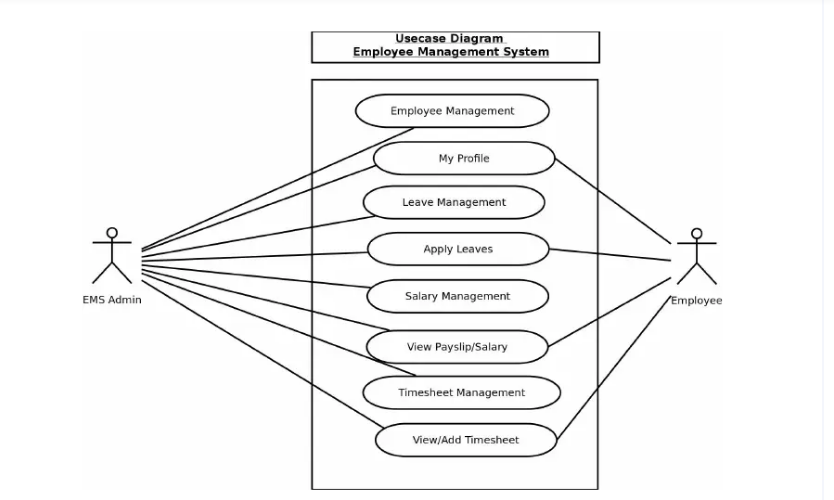
The Employee Management System (EMS) is a distributed information system for managing the staff data within a company or organization. It securely stores and manages personal and other work-related details for the employees.

It’s key features are  time and attendance management (e.g., time tracking by way of employee time sheets), absence and leave management (e.g., time-off requests), an employee database, and an employee self-service portal.

A few features are marked as **Optional**. They need not be implemented if facing time constraints.It is recommended to implement these features for practice and to obtain added recommendations in the performance report.

# Use Case Diagram

**Following are the important uses cases in the system**



# Users of Employee Management System

There are two type of users available in the Employee Management System) :

* Employee : With Limited Access
* Admin : With full access

Access is granted based on the role - either Admin or Employee.

Admins have full access to all features whereas Employees are only allowed to access the information necessary to effectively perform their job duties.

## Functionality available to Admin

****These are the features available to the admin users.****

* Login For Admin
* Forgot/Change password for Admin
* ****Manage Employee****
* Add New Employee
* View Profile of the Employee
* Listing of all Employees
* ****Manage Timesheet****
* Add Timesheet for the Employee
* Edit Timesheet for the Employee
* Listing of the Timesheets

**<Optional>**

* ****Manage Leave****
* Add Leave for the Employee
* Listing of the Leave for all Employees
* Approve/Disapprove Leave

## Functionality available to the Employee

****These are the features available to an Employee****

* Login For Employee
* Forgot /Change password for Employee
* Edit Profile For Employee
* Add Timesheet
* View Timesheet Records

**<Optional>**

* Apply Leaves

**<Optional>**

* View Leave Records

**<Optional>**

* View Payslip/Salary

## Static Pages and other sections :

****These static pages will be available in project****

* **Home Page**
* Well-designed and organized homepage, so it would be easy for linking web pages to follow a logical pattern
* Home Page will contain an animated slider for images banner



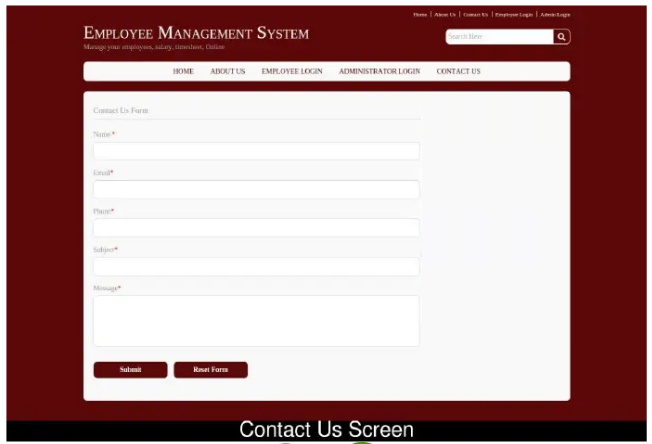


* **About us page**

You need to think of the company and the product as a whole in order to present a unique style imbued with personality that will make your design stand out in a user’s mind.

* **Contact Us Form**

A Contact Us form to be provided for employees to raise queries or issues regarding EMS with the admin.



# Modules in the Employee Management System

## Employee Management Module :

This module provides all the functionality related to employee management. This employee management module is the main module in the **Employee Management System.** This is a role based module where admin can perform any CRUD operation on data but the employee will be able to view only his/her data, so access level restrictions need to be implemented..

**Features of Employee Module:**

* Admin can add new employee records
* Admin can see the list of employees
* Only admin can edit and update the record of the employee
* Admin will be able to delete the records of the employee
* All employee forms are validated on client side using JavaScript

**<Optional>**

## Leave Module :

The main purpose for developing this module is to manage the leave. All leaves will be managed by admin and employee can easily apply leave and also can see their leave report.

**Features of Leave Module:**

* Admin can manage the leave
* Admin can edit/delete the leave
* Admin can see the list of all leaves
* Employee can view his leave
* Employee can apply leave

## Timesheet Module :

The main purpose of this module is provide all the functionality related to timesheet management.

It includes the following features:

1). Adding a particular employee’s details like name ,ID, contact, etc.

2). Searching an employee from the database of the organization.

3). Editing the details of the employee like contact no, mail-id etc.

4). Deleting employee details from the table records.

5). Displaying the working employees associated with the organization.

**Features of Timesheet Module:**

* Admin can add new timesheet
* Admin can see the list of timesheets for all employees
* Only admin can edit and update the record of the timesheet
* Admin will be able to delete the records of the timesheet
* Employee can add new timesheet
* Employee can view own timesheet records
* All timesheet forms are validated on client side using JavaScript

**<Optional>**

## Salary Module :

The main objective of this module is to manage the salary data . All salaries will be managed by admin. Admin can see the list of all the salaries and filter it according to the employee via the search box provided.

**Features of Salary Module:**

* Admin can manage the salary
* Admin can edit/delete the salary
* Admin can see the list of all salaries
* Employee can view own salary

## Login Module :

Used for managing the login details.

**Features of Login Module:**

* Login
* Logout
* Forgot Password
* Change Password

## Search Module :

Provides context sensitive, role based results for Admin and Employee based on search criteria.

**Features of Search Module:**

* Admin can search for an by Employee by Id, Name, joining date etc.
* Wildcards should be supported
* Employees can search for a particular record with the confines of their own data sets.
* Search should be sensitive to the context in which it is invoked.

Ex;

For an Admin;on the View Timesheet Listing screen , it should return Timesheet records for a particular searched Employee.

For an Employee;on the View Timesheet Listing screen , it should return Timesheet records for a particular searched date.

# Key Features :

## Employee

### Login



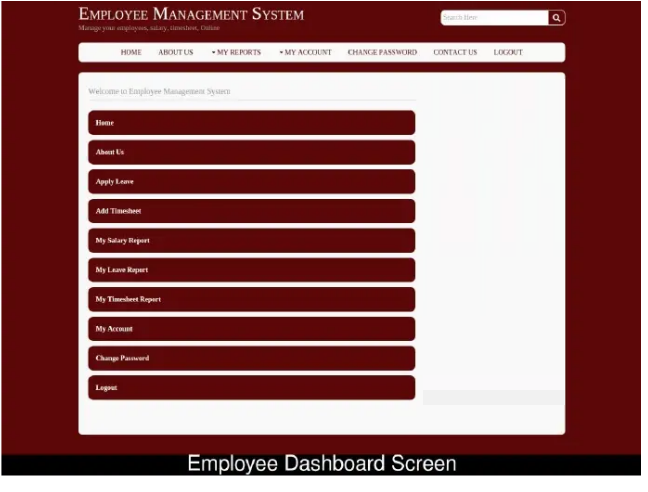
* User must be marked as ‘Active’ in the database
* User must already been added to the system by the admin.
* On Login failure, redirect to Login page.

### Forgot Password

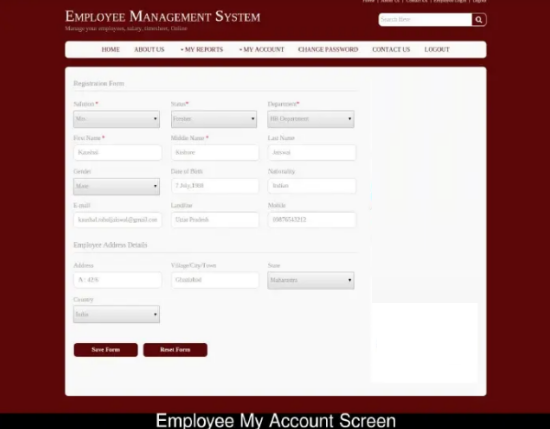


* Support email verification
* Enforce password policies
* Password Expiry (every 90 days)

### Dashboard

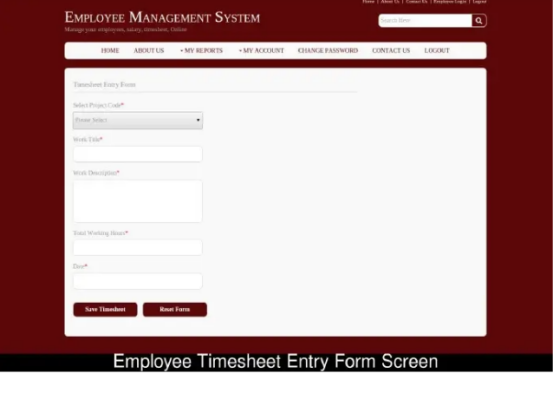


### View Profile



* Modify profile details like add/remove addresses
* Client side validation on the form
* Appropriate error messages to be displayed for invalid input.

### Timesheet Entry



### Add Timesheet

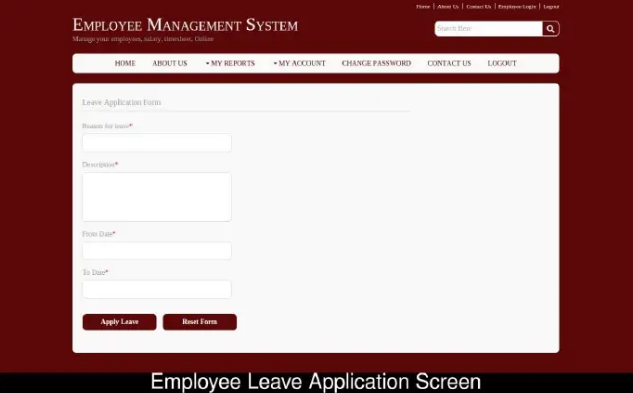
* Add a new Timesheet

### View Timesheet

* Listing of Timesheets for the logged in employee

**<Optional>**

### Apply Leaves



* Leaves to be categorized as :
  + Company Holidays
  + Restricted Leave
  + Casual Leave
* System will check appropriate leave balance for casual leave and allow/disallow submission of the form.
* For other types of leave, options will be enabled for dates that are company sanctioned. This may be checked from a company calendar maintained within the application.
* Leave category can be selected from a dropdown menu.
* Apply Leave button displays as disabled till all fields are entered with valid data.
* On successful save, the user will be navigated to the View Leaves Records screen.
* Attempting to save with invalid data, will reload the Leave Application screen with the appropriate error messages.

**<Optional>**

### View Leave Records

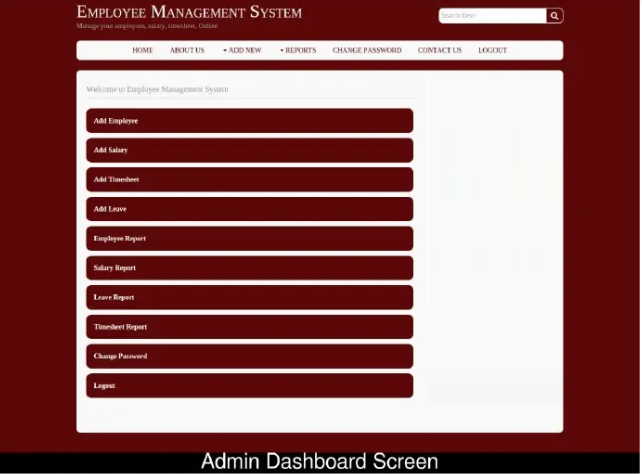
* Search should be enabled.
* Status will display Approved, Rejected or Pending (by the Admin).

## Admin

### Login

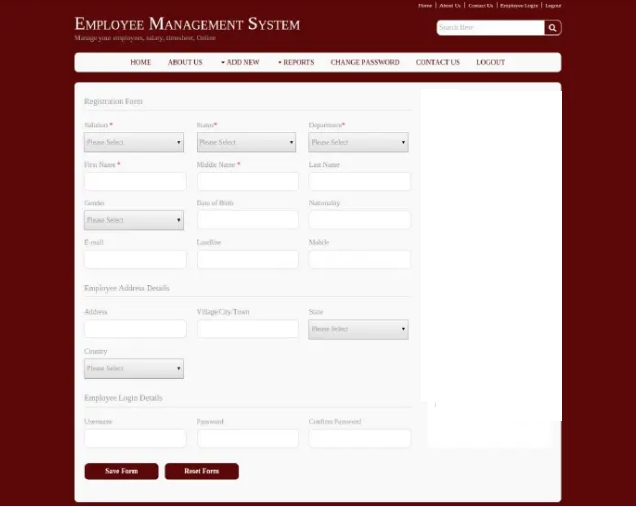
* One super user needs to be created in the system.
* Admin must be marked as ‘Active’ in the database
* Admin must already been added to the system by the super user .

### Dashboard



## Manage Employee

### Add New Employee



* Add a new Employee
* Client side validation on the form
* Appropriate error messages to be displayed for invalid input.
* Admin should have two options to add a(n) employee(s):
  + - Manual entry in a form
    - Upload a single new Employee’s details in CSV format
    - Bulk upload up to 50,000 Employee details in CSV format
* All required fields must be entered before submission.
* Save Form button displays as disabled till all fields are entered with valid data.
* On successful save, the user will be navigated to the Employee Listing screen.
* Attempting to save with invalid data, will reload the Add Employee screen with the appropriate error messages.

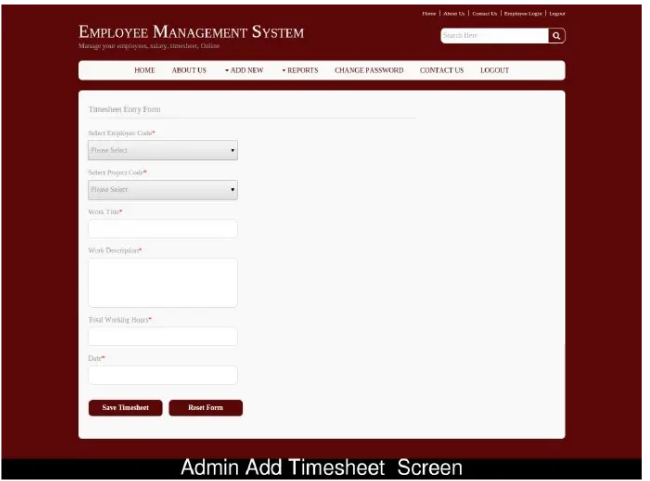
### View Listing of all Employees



* Edit an Employee’s details from this screen.
* Pagination should be supported. A choice of 50 or 100 records to be displayed on the screen with a dropdown list or UI toggle button.
* Search should be enabled

## Manage Timesheet

### Add/Edit Employee Timesheet



* Save Timesheet button displays as disabled till all fields are entered with valid data.
* On successful save, the Admin will be navigated to the Timesheet Listing screen.

### View Timesheets for all Employees

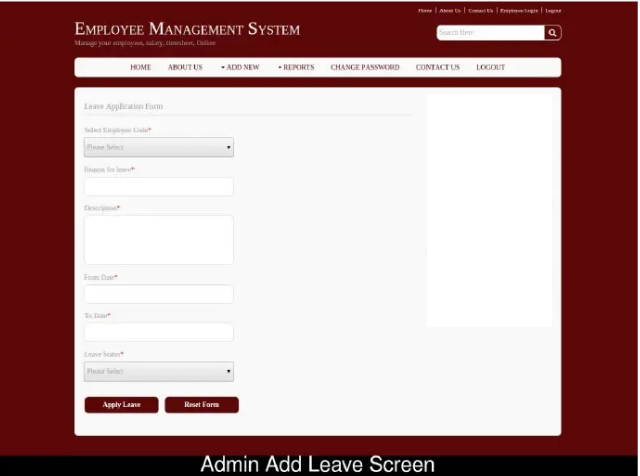


* Admin can choose view/edit an Employee’s Timesheet from this screen.
* Pagination should be supported. A choice of 50 or 100 records to be displayed on the screen with a dropdown list or UI toggle button.
* Search should be enabled

**<Optional>**

## Manage Leave

### Add Leave for the Employee



* Apply Leave button displays as disabled till all fields are entered with valid data.
* On successful save, the user will be navigated to the Leave Listing screen.

**<Optional>**

### Listing of the Leave for all Employees

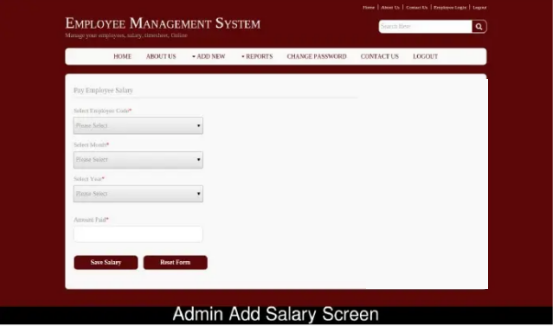


* Pending leave requests can be approved/disapproved by the Admin.
* Search should be enabled.
* Pagination should be supported. A choice of 50 or 100 records to be displayed on the screen with a dropdown list or UI toggle button.

## Manage Salary

**<Optional>**

### Add Salary for the Employee



* The System shall send a notification to the employee for whom a salary is added.
* Save Salary button displays as disabled till all fields are entered with valid data.
* On successful save, the user will be navigated to the Salaries Listing screen.

**<Optional>**

### Listing of the Salaries for all Employees



* Option to edit Salary details.
* Search should be enabled.
* Pagination should be supported. A choice of 50 or 100 records to be displayed on the screen with a dropdown list or UI toggle button.

### Search Feature



* The Search Feature should be displayed on every page.

# Classes and their attributes

* ****Employee Attributes**** : empID, empName, empContact,empDepartment, empEmail, empAddress,empRole
* ****Timesheet Attributes**** : empID, empName, timeTotalHours,timeDate, timeTask, timeDescription
* ****Salary Attributes**** : salID, empID, salMonth, salDate, salAmount
* ****Leaves Attributes**** : leaveID, empID, leaveTitle, leaveApplyDate, leaveStartDate, leaveEndDate, leaveStatus
* ****Login Attributes**** : loginID, empID, loginUsername, loginPassword, loginStatus

# Classes and their methods

* ****Employee Methods**** : addEmployee(), updateEmployee(), getEmployee(),deleteEmployee(),listEmployees(), searchEmployee(), getRole()
* ****Timesheet Methods**** : addTimesheet(), updateTimesheet(), getTimesheet(), listTimesheets()
* ****Salary Methods**** : addSalary(), updateSalary(), deleteSalary(), saveSalary(), listSalariess(),searchSalary()
* ****Leaves Methods**** : applyLeave(), updateLeave(), getLeave(), deleteLeave(), listLeaves(), searchLeave(), getStatus()
* ****Login Methods**** : checkLogin(), getLoginDetails(), getLoginStatus(), createLogin(), changePassword()

# NFRs & SLAs:

* The system must support 50,100, 500,5000,50,000 concurrent users
* The system must support 1-page/5-page requests per second
* The  system will need to support 15 user logins per second
* The system will need to support 150 transactions per second.
* System eventually needs secure access over HTTPS/TLS
* The system should be extensible ie; ability to add more functions/features in the future.
* 99% uptime.
* Each page must load within 2 seconds.
* Different employees should not be able to access each other’s data. Only an admin has the authority to edit customer data and enter new customers to the system. The admin would also be the only one to see all employees’ data.
* Web forms would need to be as simple as possible, and in case there was data that needed to be entered in a certain way, it would need to be explained on the page. Every link and button needed to be logical and lead to a page they clearly indicated they would lead to. Everything in the system would be designed with the end user in mind.
* Secure coding standards to be followed.
* Security - once the Admin approves a timesheet, they can secure it to make sure that what arrives at payroll is the same as what they finalized.No one can come in and fudge the numbers. The database record must be locked.
* Search results should return in <3 seconds.
* Design of the website should be responsive to viewports of mobile, tablets and desktop PCs.
* Allow the database to only be accessed from validated IP addresses
* User friendly error pages should be displayed to the end users
* Automate Performance Testing with JMeter
* DB failover and disaster recovery strategy to be documented
* **Only one session per user** at a time or **no concurrent session per user**. If the user tries to open a new session then either an alert is shown or his previous session is closed.

# Testing:

* Smoke and Sanity Testing
* Unit Testing
* System Testing
* [JaCoCo](https://www.eclemma.org/jacoco/" \t "https://medium.com/capital-one-tech/_blank) - Code coverage - 80%
* [JaCoCo](https://www.eclemma.org/jacoco/" \t "https://medium.com/capital-one-tech/_blank) - Cyclomatic complexity < 4

# Other Considerations:

* System should support upload of raw data in CSV format .
* Up to 50,000 employee records need to be stored
* The below link allows you to download sample csv files ranging from 100 records to 5000000 records :-

<https://eforexcel.com/wp/downloads-16-sample-csv-files-data-sets-for-testing/>

* Other links that may be helpful for obtaining raw data:-

<https://gist.github.com/kevin336/acbb2271e66c10a5b73aacf82ca82784>

<https://sample-videos.com/download-sample-csv.php>

<https://wsform.com/knowledgebase/sample-csv-files/>

* If no one csv suits your requirement, you can compose your datasets from multiple CSVs.
* You may import only selected columns from a CSV into tables in the database.

# Weekly Milestones

## Week 1:

### Create Architecture of the Project with UML diagrams.

1. Use Case Diagram
   1. Use Case Diagram of EMS
2. Activity Diagrams
   1. Employee Login
   2. Admin Add Employee
   3. Employee Timesheet Entry
3. Class Diagrams
   1. EMS Class Diagram
4. Sequence Diagrams
   1. Employee Login
   2. Admin Add Employee
   3. Employee Timesheet Entry

### Create a Maven project using Eclipse.

1. Write up the dependencies in the pom.xml for a Java application.
2. Include the JUnit5 dependency.
3. Add JaCoCo Maven Plug-in for code coverage.

### Implement the Java application using the TDD approach

1. Write the initial test cases using JUnit5.
2. Use Mockito to mock the database.
3. Create the java classes according to the UML model defined. Refer to the classes, methods and attributes described [here](#Classes).
4. Method implementations should initially be empty in the classes.

### Create the Entity Relationship (ER) Diagram

1. Create the database tables, constraints and relationships between tables.
2. Load sample data in to the database.
3. Use JDBC for connecting to, retrieving and inserting/updating records in the database.
4. The user should be able to interact with the application from the command prompt.
5. Based on user input, correctly formatted output should be displayed in the command prompt.
6. Create the Command Line interface with the Scanner class to retrieve user input and display the results.

### Check code into Bitbucket

Push the code base to the appropriate Bitbucket repository

## Week 2:

### Refactor for error handling and usage of DAO classes

1. Follow the TDD approach to refactoring the application to include exception handling for each CRUD operation.
2. Create a custom exception class to handle business logic errors.Write the JUnit test cases for the refactored application.
3. Use Mockito with @Spy annotation to mock the database.
4. Create a DAO class and interface for each module - Employee, Timesheet, Login and Search.
5. The interface presented to the user in the command prompt remains the same. Exceptions should be handled with an appropriate message to the end user.
6. Implement Employee, Timesheet, Login and Search classes to use the corresponding DAO to perform CRUD operations

### BDD Using Cucumber

1. Use behavior-driven development (BDD) to define clear, unambiguous requirements.
2. Turn these requirements into automated acceptance tests.

### Update Architecture Diagrams

Changes to the code base needs to be reflected in the UML class and sequence diagrams.

### Check code into Git

Check the code into the Bitbucket repository over the course of the Sprint for review.

## Week 3:

### Update the Maven pom

1. Add the dependencies for Web Application development in the pom.xml .
2. Add the JPA and Hibernate dependencies.
3. Remove the JDBC dependencies.

### Create REST APIs for the CRUD operations.

1. Use JAX-RS to create the REST APIs for CRUD operations.
2. Test the APIs in Postman.

### Refactor to use JPA with Hibernate for database operations

1. Follow the TDD approach to refactoring the application to use JPA with Hibernate instead of JDBC.
2. Write the JUnit test cases for the refactored Java classes.
3. Use @Spy to mock the database implementation class.
4. Implement the refactored classes according to specifications.

### Create a Performance Test Plan in JMeter

1. Do a performance analysis of the web application for 1000 users.
2. JMeter should show the test result in Graph format.

### Update Architecture Diagrams

Changes to the code base needs to be reflected in the UML class and sequence diagrams.

### Check code into Git

Check the code into the Bitbucket repository over the course of the Sprint for review.

## Week 4:

### Create a Spring Boot Starter Project with Maven

1. Add the dependencies for Spring MVC, Data JPA ,Bean Validator Reference Implementation and Database Connector.
2. Remove the JPA and Hibernate dependencies.
3. **(Optional)** You may add other dependencies like Lombok to ease the burden of writing boilerplate code.
4. Spring Test is included by default.
5. The bean validator reference implementation is actually an instance of hibernate‑validator .Add the hibernate‑validator dependency to the project.

### Create the REST Endpoints for the CRUD operations

1. Follow the TDD approach to creating the back end tier.
2. You will still be writing the test cases using JUnit but using the framework support in Spring Boot. Spring Test is the Spring Boot test support.You need to cover unit tests that can run in isolation as well as integration tests that will bootstrap Spring context before executing tests.
3. The following classes of tests need to be implemented:-

1. **Smoke and Sanity test:-.** Check if the Spring Context is running.
2. **System test:-** It will test the complete application.
3. **Integration test:-** Test the Controller tier together with the Service layer.
4. **JPA test:-** Test the Service tier and the database.
5. Create the Controller, Service and Repository tiers.
6. Create RESTful Web Service endpoints corresponding to each CRUD operation.
7. Use Status Codes in Error Handling
8. Implement global exception handling with @ControllerAdvice for Controller classes.
9. Replace the previous JPA with Hibernate database implementation with Spring Data JPA.
10. Use the JSR 380 and implement Validation on the model objects.
11. Test the REST API endpoints with Postman.

### Create a FreeStyle project in Jenkins to automate the workflow

1. Create a Jenkins FreeStyle project
2. Checkout/clone the code base from Git repository
3. Compile and package the application
4. Run the JUnit tests and publish the results
5. Build should complete successfully in Jenkins with passing test cases

### Update Architecture Diagrams

Changes to the code base needs to be reflected in the UML class and sequence diagrams.

### Check code into Git

Check the code into the Bitbucket repository over the course of the Sprint for review.

## Week 5:

### Create a React Project

1. Follow TDD with Jest to build the complete front end using React, HTML5 and CSS3.

The following screens need to be developed:

1. Login Page(may be a single page for both Admins and Employees or separate)
2. About Us (Optional)
3. Contact Us (Optional)
4. Forgot Password (may be a single page for both Admins and Employees or separate)
5. Admin Add Employee

* The Admin Add Employee screen should have an option to select a CSV file for bulk upload of employees’ details into the database.

1. Admin List Employees
2. Employee Timesheet Entry
3. Employee View Timesheets
4. Search Feature

1. (Optional) Implement Salary and Leave modules for extra credit.
2. Introduce a 3 second delay into the application when the list of records is being retrieved, or when a record is being updated or added.
3. The pages displaying the list of records and add/update a record should show a Spinner or Progress Bar while the page is being displayed or the records are being retrieved. The Spinner or Progress Bar should disappear when the page is rendered.
4. The page(s) where a record is added or updated should have the Save/Update button disabled while a new record is being saved or updated. The label on the button should change from 'Save' to 'Saving..'.
5. On successful save or update, the user should only then be navigated to the list of records.
6. On unsuccessful save or update, the user should remain on the page. An notification should be displayed with a user friendly error message.
7. On Delete, the user must immediately be navigated to the list of records(Optimistic Deletes) unless the Delete buttons are on the same page; in which case the page simply needs to be reloaded to display the updated list.
8. The notification for an unsuccessful Create,Update or Delete action will not have a display interval. It will need to be manually closed by the end user.
9. Use Toast Notifications or SweetAlert or any library of your choosing to display alerts and messages to a user. It can be a success message, warning message, or custom message. You may set the display interval to 2-3 seconds.
10. Consume the REST APIs in the Spring Boot-REST-Data JPA back end from your React application using AXIOS.
11. Develop the production server of your react app with Node.js

### Use HTML5 semantic elements. Style with CSS3.

1. Use your creativity and knowledge of HTML5 and CSS3 to design and present an appealing website.
2. Use CSS3 Media Queries so that the application renders properly on small to medium sized devices as well as desktop PCs.
3. Setup up linting with ESLint in the package.json file with appropriate rules.
4. You may deploy the application using Node.js.

### Create the Dockerfile environments

1. Define the Docker environments using Dockerfile.
2. Create the  Dockerfile for the Spring Boot app.
3. Deploy the React app behind an nginx server.
4. Create the docker-compose.yml configuration
5. Run the complete stack with docker compose

### Use Spring Cloud Microservices and Spring Boot Framework to make this application distributed.

You may come up with your own architecture and deployment strategy. A few suggestions are:-

1. Register the spring boot application with the eureka discovery server.
2. Front End React App makes requests to the NGINX server which acts as a reverse proxy.
3. NGINX server redirects the requests to Zuul API Gateway.
4. Zuul will route the requests to microservice based on the URL route.
5. Zuul also registers with eureka and gets the IP/domain from eureka for the microservice while routing the request.
6. Ribbon strategy

### Architecture Diagrams

1. Changes to the code base needs to be reflected in the UML class and sequence diagrams.
2. Create the Deployment diagram
3. Create the architecture diagram for the microservices project.

### Check code into Git

Check the code into the Bitbucket repository over the course of the Sprint for review.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*