**Employee Management System – Project Plan**

1. **Requirement Analysis**

The System should be able to answer the following problems.

* 1. Facilitate managing employee details. (able to create, update, delete and view employee detail)
  2. All Employe can be able to calculate the number of working days between two given dates, excluding weekends and holidays.

1. **Design**
   * Design database model.
   * Design project architecture such as design pattern, frameworks and UI.
   * Design the logic for managing employee details.
   * Design the algorithm for calculating working days, considering factors like weekends and holidays.
2. **Project Timeline**

|  |  |  |
| --- | --- | --- |
| Phase | Activities | Duration |
| Planning | Requirement analysis | 1 hr |
| Design | Database design, Project architecture design, UI design, logic and algorithm | 3 hr |
| Development | Development, Integration and Error handling | 2 days |
| Testing | Integration Testing, User acceptance | 1 day |

1. **Implementation**
   * Create database table and create simple store procedure for inserting holidays into database table.
   * Follow the repository design pattern for project creation and setting up database connectivity. (Follow database first approach)
   * Implement CRUD operation for Employee details.
   * Implement the algorithm for calculating working days based on the design.
   * UI implementation and Integration.
   * Perform Integration testing and User acceptance testing.
   * Code optimization. (use cache, delegate, dependency injection)
2. **Pseudo code**
   * 1. **Working Days Calculation Algorithm**

#define a working\_day\_dto class

class working\_day\_dto

attributes :

-no\_of\_workingday

-error

#start function for calculate working day

function calculate\_working\_days(start\_date, end\_date):

working\_day\_dto = working\_day\_dto ()

working\_days = 0

current\_date = start\_date

if is\_weekend\_day(start\_date):

working\_day\_dto .error = “Start Date should be a week day”

else

while current\_date <= end\_date:

if is\_working\_day(current\_date) not true and is\_holiday(current\_date) not true:

working\_days += 1

end if

current\_date = current\_date.addDays(1)

#loop end

working\_day\_dto .no\_of\_workingday= working\_days

return working\_day\_dto

#end function calculate\_working\_days

function is\_weekend\_day(date):

# check if the given date is on a weekend (Saturday or Sunday)

if date.dayofweek== "Saturday" or date.dayof week == "Sunday":

return true

else:

return false

function is\_holiday(date)

#check if the given date is holiday or not

#holiday\_list include all the holidays for current year

#store holiday\_list value in the memory until cache expiration

# check if date is in the holiday\_list

if date in holiday\_list then

return true;

else

return false;

end if