Low-Code Development

Submitted in partial fulfilment of the requirements for the degree of

Bachelor of Technology

in

Information Technology

by

ARADHYA MATHUR

17BIT0146

Under the guidance of

Dr. Prabhavathy P.

School of Information Technology and Engineering

VIT, Vellore.

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

June, 2021

DECLARATION

I hereby declare that the thesis entitled "Low-Code Development" submitted by me, for the award of the degree of *Bachelor of Technology in* Information Technology to VIT is a record of bonafide work carried out by me under the supervision of Dr. Prabhavathy P.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Vellore

Date: 1st June, 2021

Signature of the Candidate

CERTIFICATE

This is to certify that the thesis entitled “Low-Code Development” submitted by **Aradhya Mathur, 17BIT0146**, School of Information Technology and Engineering, VIT, for the award of the degree of *Bachelor of Technology in Information Technology,* is a record of bonafide work carried out by him under my supervision during the period, 01. 02. 2021 to 01.06.2021, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfils the requirements and regulations of the University and in my opinion meet the necessary standards for submission.

Place: Vellore

Date: 1st June, 2021

Signature of the Guide

Internal Examiner **External Examiner**

Head of the Department

Information Technology

CLOVER™
INFOTECH

Date: 15th May 2021

CERTIFICATE BY THE EXTERNAL GUIDE

This is to certify that the project report entitled "LOW-CODE DEVELOPMENT" submitted by **ARADHYA MATHUR** (17BIT0146) to Vellore Institute of Technology in partial fulfilment of the requirement for the award of the degree of B.Tech. in Information Technology is a record of bonafide work carried out by him under my guidance. The project fulfils the requirements as per the regulations of this Institute and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Bhushan Daga

EXTERNAL SUPERVISOR

Solution Architect

2nd Floor, Dhana Singh Processors Building, Vazir Glass Lane, J B Nagar, Andheri (E), Mumbai–400059, India, marketing@cloverinfotech.com, +91 (22) 29261650

Clover Infotech Pvt. Ltd.

CLOVER™
INFOTECH

Date: 24th May 2021

TO WHOM SO EVER IT MAY CONCERN

This is to certify that Mr. Aradhya Mathur was working with our organization from **15th December 2020 to 15th May 2021** as an **Intern.**

During this Internship, he was involved in internal project development team and he has worked on Oracle APEX low code environment & PL SQL.

We found him hardworking and sincere in completing all the tasks entrusted to him. We wish him all the best in all his future endeavors.

For Clover Infotech Pvt.Ltd.

Clover Infotech Pvt. Ltd.

Elizabeth Paul
Senior Vice President – Human Resources

ACKNOWLEDGMENTS

The internship opportunity I had with Clover Infotech was a great chance for learning and for my professional development. Therefore, I consider myself extremely lucky for having being provided with such an opportunity. I am also grateful for having a chance to meet such wonderful people and professionals who led me through the internship period.

I am also thankful to VIT, Vellore for providing such a wonderful opportunity. I express my deepest thanks to Dr. Prabhavathy P for giving me necessary advices, guidance and for providing all the help throughout the period.

I express my sincere gratitude to Mrs. Elizabeth Paul for giving me opportunity to work for Clover Infotech. I would also like to thank Mr. Bhushan Daga who in spite of being extremely busy with his responsibilities, took his time out guide me through this internship period.

I regard this chance as a great milestone in my career development. My aim will be to use these gained skills and knowledge in the best possible way. I will also continue to work on improvement of these skills, in order to reach desired career goals.

Aradhya Mathur

Executive Summary

I started my off-campus internship at Clover Infotech on 15th December, 2020 and I was first trained in theory and lab work of SQL followed by PL/SQL and Oracle APEX. During the learning phase I developed two applications, one of which was car search and other was employee attributes and performance for which a dataset was provided and visualization and chart tools were added. My first project started on 1st February and was Email Delivery System. My second project Time Sheets Application started on 8th March. Work Log Calendar, my final project started on 26th April. Oracle Apex was used as low-code platform to develop all the applications. SQL was used for database creation and modification and PL/SQL was used for completing every dynamic action. All the projects were delivered on time and all the requirements were met. Along with improvement in my technical skills, my time management and communication skills got better. I completed my internship on 15th May, 2021.

CONTENTS

List of Figures

List of Tables

List of Abbreviations

SQL	Structured Query Language
PL/SQL	Procedural Language extensions to SQL
APEX	Application Express

1. INTRODUCTION

1.1 OBJECTIVE

Low-code is a visual method to software development and it extracts and automates each and every phase of the application lifecycle to enable rapid delivery of different software solutions. Using Oracle Application Express also known as APEX which is a low-code development platform can enable me to build scalable, secure enterprise apps, with world-class features, that can be deployed anywhere. Using APEX, I can quickly develop and deploy captivating apps that can solve real world problems and provide immediate value.

Using Oracle APEX, I can minimize the complexity involved with multi-faceted applications and solve many business problems. Powerful data management can be done and data visualization components can be used too.

I want to utilize the opportunity of this internship to the fullest and I want to develop quality applications for the company. Applications should be easy to use, efficient, effective and feasible. Learning and using SQL for the database and PL/SQL for dynamic actions is necessary and also, they integrate well with Oracle Apex platform. In the end I would like to complete all the projects assigned to me properly and learn all the new technical things in the process.

1.2 MOTIVATION

It was a great opportunity to work for a reputed company like Clover Infotech and on real world problems and time bound projects to provide most efficient and effective and yet feasible solutions. Learning new technology like Oracle Apex and working under my guide was an exceptional chance. Learning new and upcoming of technologies in this ever-evolving IT sector is highly motivative. I wanted to learn as much as I could and work on projects to provide quality applications for the company. In the end, I am happy to deliver the products and, in the process, I became quite proficient in SQL, PL/SQL and Oracle Apex.

1.3 BACKGROUND

Oracle APEX is a low-code environment that will enable me to
build, design and develop feasible, scalable and secured apps, with amazing features, that can be deployed and used anywhere. Using APEX, I can quickly design, develop and deploy quality applications that can solve real world problems.

1.3.1 Literature Survey

1.3.1.1 Low code programming with oracle apex offers new opportunities in higher education

In this paper the authors present various case studies of introducing Oracle APEX at Faculty of Organizational Sciences at the University of Maribor. Their aim is to teach students to be more flexible and to think differently. According to them Oracle APEX can be used in different educational levels, for advanced topics in IS education and as it also supports web application development students can work on cloud as well. They also presented faculty's experiences with low code application development, with other Higher Education institutions showing

interest. It also attracted other faculties based on positive reviews of successful classroom teaching of APEX.

1.3.1.2 Rapid development of database interfaces with oracle apex, used for the controls systems at CERN

CERN was among the first companies who purchased Oracle RDBMS which was used for data management needs in CERN accelerator environment. This article was written to show when APEX can be suitable application development technology and to showcase firsthand experience of developers. Apex had well defined set of APIs which allowed advance logic, able to work with most complex reporting needs. Apex id OS and internet browser agnostic which made it favorable as both windows and Linux were used. Due to its lack of native version control intelligence was a point of concern for them. APEX has two-tier architecture so there was increase in connection between client and database servers. In the end APEX fulfilled all the needs for data browsing tools of CERN accelerators and also had decent programming policy.

1.3.1.3 An application synopsis tool for database applications developed using oracle application express

According to the authors it is necessary to have an application synopsis tool that reduces efficiency lags and also improves developer's output by giving an overview of the application being developed. They worked on problem of deriving synopsis of a database-centric application developed with Oracle APEX, a rapid application development tool. More generally they developed an Application Synopsis tool that matches Oracle APEX views and application reports capability by including various features. Moreover, their tool directly connects with the corresponding APEX application and page summaries which

allowed all-in-one use of shared capabilities in evaluating an application. They have demonstrated the benefits of having such a tool using usability study conducted with three applications.

1.3.1.4 More on Oracle APEX for teaching and learning

Authors have extended their research from the experiences of Tomlinson and Gardner by focusing on an evaluation of. They have concluded with an overall positive view in relation to these four criteria, and have identified areas of improvement for using APEX. Four criteria were 1) Administration, Scalability and Reliability for Teaching 2) Teaching and Learning of Introductory Database Application Development 3) Promotion and Monitoring of Engagement and Feedback of Learning 4) Teaching and Learning of more Advanced Database Application Development. According to Tomlinson & Gardner APEX is a perfect tool for the teaching and learning of enterprise databases. Authors have extended the use of APEX where appropriate to all database units, and to continue to use it for the likely future. Apex was convenient to allocate a class to a workspace, it can map a workspace to a single schema, low maintenance once admin structure set-up, overcomes lack of scripting ability but has less scope for students to manage objects.

1.3.1.5 Vulnerability analysis with SQLMAP applied to APEX 5 environments

According to the author databases are frequently the key targets of an attack, precisely because of the information that resides in them. Author has carried out vulnerability tests on the database of an ERP software developed in APEX 5. Author used FOSS tools to test and analyse database vulnerabilities, recognizing that the sessions used by ERP based on Oracle APEX are carried out randomly and, in addition, they are generated again at certain

times. Author concluded with the tests applied and the updates of SQLMAP to the date of the experiment and was able to determine that it was impossible to violate the ERP software with SQL injection techniques.

2. PROJECT DESCRIPTION AND GOALS

In the learning phase I was taught SQL, PL/SQL and Oracle Apex and during that period I completed two sample applications one of them namely Cars Search during the training and other one namely Employee Attributes and Performance as the assignment for which a dataset was given by the instructor. After completion of training, Email Delivery System was my first project for which I developed database of customers and created applications with features such as interactive reports and feedback. I created email page which was then linked via PL/SQL to perform dynamic action of sending email. My second project was Time Sheets Application, in which database was created and application was created using Oracle Apex through which new employees can be added and details can be edited. The main function of the application was daily work log of each employee which had to be approved or declined by the manager. Work Log Calendar was my final project. Using Oracle Apex this application was created for work log record of an employee. It has several features like feedback, adding/removing work, changing status of the work. My main goal was to deliver each and every project on time and with all the functionalities required and specified. My other goal was to learn as much as I can in the process and with the new learnings make the next project better.

3. TECHNICAL SPECIFICATIONS

3.1 SQL

Structured Query Language is a database language, and is used to perform operations on the records stored in the database such as creation, deletion, fetching, and modification, etc.

Structured Query Language became a standard of the ANSI in 1986 and of the ISO in 1987. It introduced the idea of accessing several records with one single command. Also, it eliminated the necessity to specify how to reach a record.

In Oracle APEX, SQL is used for browser-based maintenance of database objects and data. It is designed to meet application developers' needs, especially in hosted environments. In APEX, Quick SQL rapidly designs and prototype data models using a markdown-like shorthand syntax that expands to standards-based Oracle SQL.

3.2 PL/SQL

In the early 90's PL/SQL was developed by Oracle Corporation; it is a block structured language and a procedural language designed precisely to support SQL statements within its syntax and enhance the capabilities of SQL.

Its program units are collected by the Oracle Database server and stored inside the database. PL/SQL has robustness, security, and portability inherited from Oracle Database. Pl/SQL stands for Procedural Language extension of SQL.

In Oracle APEX, PL/SQL is responsible for all business logics and validation. PL/SQL units like procedures, functions, packages, types and triggers, etc. can be created in Oracle APEX which are stored in the database and can be reused.

3.3 ORACLE APEX

In 2004, Oracle Application Express which is a low-code development platform was released by Oracle Corporation. APEX is used to develop scalable, feasible, user-friendly and highly secure applications, with elite features like interactive report, interactive grid, faceted search, and charting engine, etc, which can be deployed anywhere.

APEX allows developer to work on solving the real problem and delivering quality solutions in less time and effort on repetitive lower-level coding. It follows agile software development methodologies.

Oracle APEX offers strong support for HTML5 development frameworks, CSS3 UI, JavaScript libraries, flexible development, self-service provisioning and deployment including cloud services like AWS. It is also consistent with Oracle SQL and PL/SQL.

4. DESIGN APPROACH AND DETAILS

4.1 SAMPLE APPLICATIONS

4.1.1 Cars

4.1.1.1 Design Approach and Methods

An application was developed to teach me Oracle Apex which is a is low- code development platform and follows agile development methodologies by instructor of Clover Academy named cars search which had features such as searching, sorting and report showing.

Web application was made on low-code platform which was highly scalable, efficient and feasible. SQL was used to create database for the application. Several features interactive report was added which enables you to create new entries. For searching faceted search and cards format was used.

4.1.1.2 Workflow

4.1.1.2.1 Modules

a) Database: Using SQL, database was created from scratch containing details related to cars
b) Application Interface Designing: Using Oracle APEX, interface for the app was designed
c) Application Development: APEX was used to develop application containing all the features from scratch.
d) Testing, Correction and Modification: Application was tested, corrected and further modified using APEX.

4.1.1.2.2 Diagrams

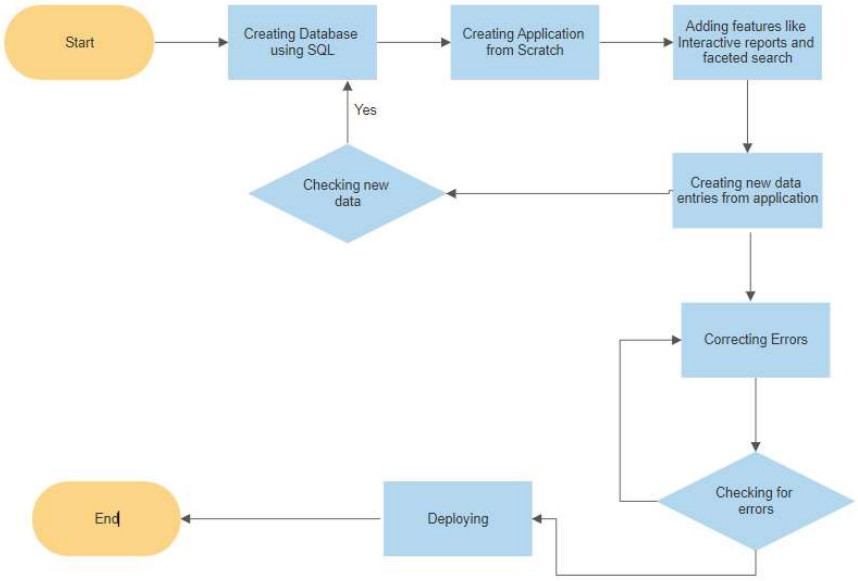


Figure 1: Workflow Diagram of Cars Application

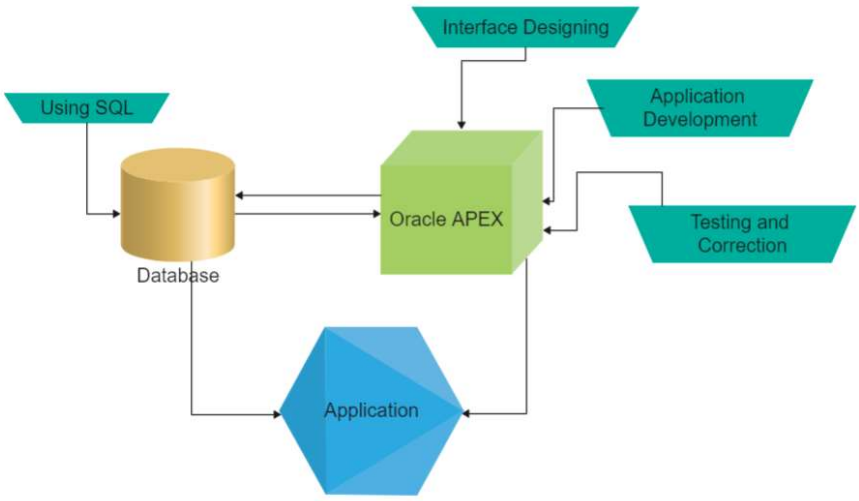


Figure 2: System Diagram of Cars Application

4.1.1.3 Codes

4.1.1.3.1 SQL Code

```
CREATE TABLE  "CARS"
  (      ID NUMBER GENERATED BY DEFAULT ON 0 AS
IDENTITY MINVALUE 1 MAXVALUE 999 INCREMENT
BY 1 START
        "CAR" VARCHAR2(255),
        "MPG" NUMBER,
        "CYLINDERS" NUMBER,
        "DISPLACEMENT" NUMBER,
        "HORSEPOWER" NUMBER,
        "WEIGHT" NUMBER,
        "ACCELERATION" NUMBER,
        "MODEL" NUMBER,
        "ORIGIN" VARCHAR2(50),
         PRIMARY KEY (ID)
  )
/
```

4.1.1.3.2 Sample Query

```
select ID,
     CAR,
     MPG,
     CYLINDER,
     WEIGHT,
     ACCELERATION,
     MODEL,
     ORIGIN
from "CARS" a
```

4.1.1.4 Constraints, Alternatives and Tradeoffs

Traditional method of web application can still be used which requires deep understanding of scripting and designing languages such as HTML, CSS, Bootstrap. It also requires deep understanding of AngularJS and NodeJS. This method takes more time to develop applications and is not easy to modify. However, Oracle APEX follows agile methodologies and is more user friendly. Low-code environment is easier and faster. Quick scalable and efficient applications can be developed using APEX.

4.1.2 Employee Attributes and Performance

4.1.2.1 Design Approach and Methods

A sample data set containing employee's attribute was provided and a similar application was to be developed with addition of visualisation tools for example pie charts and bar graphs. I developed database using SQL and started creating application from scratch. Interactive reports were used for searching employees on basis of their attributes. Another interactive report was added for creation of new employees and their details. This was linked to database for automated modification of database. Dashboard page was added for showing details in graphical chart format such as bar graphs and pie charts. APEX is a is low- code development platform and follows agile development methodologies.

4.1.2.2 Workflow

4.1.2.2.1 Modules

a) Database: Using SQL, database was created from scratch containing details from dataset.
b) Application Interface Designing: Using Oracle APEX, interface for the app was designed. Dashboard was designed for presenting bar graphs and pie charts.
c) Application Development: APEX was used to develop application containing all the features from scratch.
d) Testing, Correction and Modification: Application was tested, corrected and further modified using APEX.

4.1.2.2.2 Diagrams

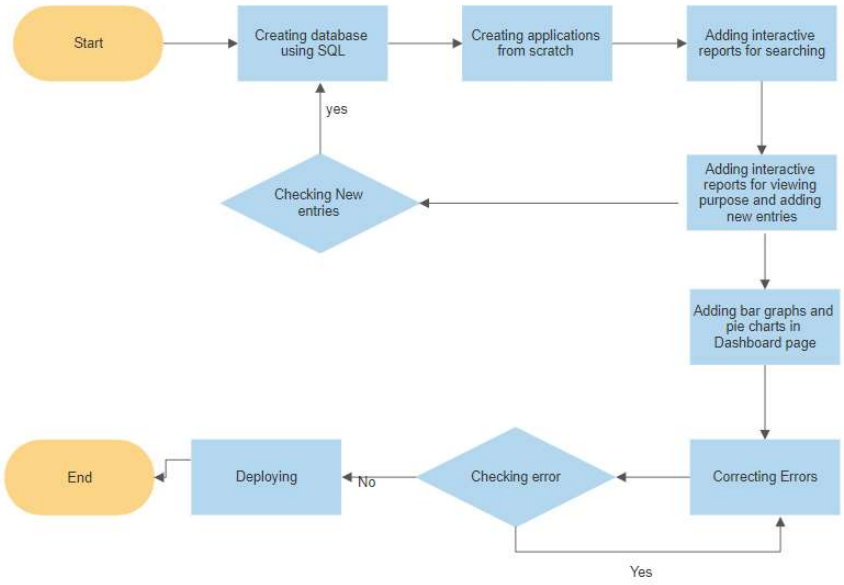


Figure 3: Workflow Diagram of Employee Attributes and Performance Application

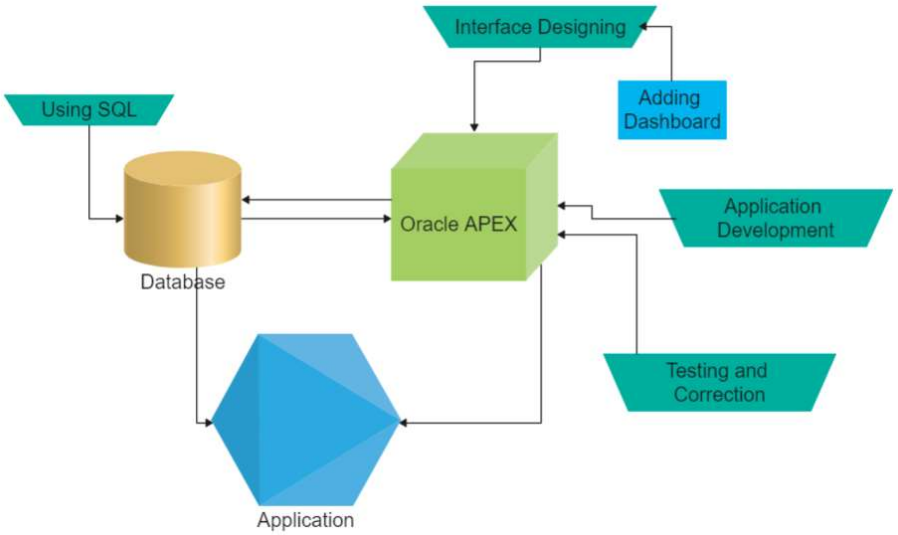


Figure 4: System Design of Employee Attributes and Performance Application

4.1.2.3 Codes

4.1.2.3.1 SQL Code

```
CREATE TABLE  “EMPLOYEEATTRIBUTE"
  (     "ID" NUMBER GENERATED BY DEFAULT ON 0
AS IDENTITY MINVALUE 1 MAXVALUE 99 INCREMENT
BY 1 START
        "AGE" NUMBER,
        "ATTRITION" VARCHAR2(50),
        "BUSINESSTRAVEL" VARCHAR2(50),
        "DAILYRATE" NUMBER,
        "DEPARTMENT" VARCHAR2(50),
        "DISTANCEFROMHOME" NUMBER,
        "EDUCATION" NUMBER,
        "EDUCATIONFIELD" VARCHAR2(50),
        "EMPLOYEECOUNT" NUMBER,
        "EMPLOYEENUMBER" NUMBER,
        "ENVIRONMENTSATISFACTION" NUMBER,
        "GENDER" VARCHAR2(50),
        "HOURLYRATE" NUMBER,
        "JOBINVOLVEMENT" NUMBER,
        "JOBLEVEL" NUMBER,
        "JOBROLE" VARCHAR2(50),
        "JOBSATISFACTION" NUMBER,
        "MARITALSTATUS" VARCHAR2(50),
        "MONTHLYINCOME" NUMBER,
        "MONTHLYRATE" NUMBER,
        "NUMCOMPANIESWORKED" NUMBER,
        "OVER18" VARCHAR2(1),
        "OVERTIME" VARCHAR2(50),
        "PERCENTSALARYHIKE" NUMBER,
        "PERFORMANCERATING" NUMBER,
```

```
    "RELATIONSHIPSATISFACTION" NUMBER,
    "STANDARDHOURS" NUMBER,
    "STOCKOPTIONLEVEL" NUMBER,
    "TOTALWORKINGYEARS" NUMBER,
    "TRAININGTIMESLASTYEAR" NUMBER,
    "WORKLIFEBALANCE" NUMBER,
    "YEARSATCOMPANY" NUMBER,
    "YEARSINCURRENTROLE" NUMBER,
    "YEARSSINCELASTPROMOTION" NUMBER,
    "YEARSWITHCURRMANAGER" NUMBER,
     PRIMARY KEY (ID)

  )
/
```

4.1.2.3.2 Sample Query

```
select ID,
    AGE,
    ATTRITION,
    BUSINESSTRAVEL,
    DAILYRATE,
    DEPARTMENT,
    DISTANCEFROMHOME,
    EDUCATION,
    EDUCATIONFIELD,
    EMPLOYEECOUNT,
    EMPLOYEENUMBER,
    ENVIRONMENTSATISFACTION,
    GENDER,
    HOURLYRATE,
    JOBINVOLVEMENT,
    JOBLEVEL,
```

```
    JOBROLE,
    JOBSATISFACTION,
    MARITALSTATUS,
    MONTHLYINCOME,
    MONTHLYRATE,
    NUMCOMPANIESWORKED,
    OVER18,
    OVERTIME,
    PERCENTSALARYHIKE,
    PERFORMANCERATING,
    RELATIONSHIPSATISFACTION,
    STANDARDHOURS,
    STOCKOPTIONLEVEL,
    TOTALWORKINGYEARS,
    TRAININGTIMESLASTYEAR,
    WORKLIFEBALANCE,
    YEARSATCOMPANY,
    YEARSINCURRENTROLE,
    YEARSSINCELASTPROMOTION,
    YEARSWITHCURRMANAGER
from "EMPLOYEEATTRIBUTE" a
```

4.1.2.4 Constraints, Alternatives and Tradeoffs

Traditional method of web application can still be used which requires deep understanding of scripting and designing languages such as HTML, CSS, Bootstrap. It also requires deep understanding of AngularJS and NodeJS. This method takes more time to develop applications and is not easy to modify. However, Oracle APEX follows agile methodologies and is more user friendly. Low-code environment is easier and faster. Quick scalable and efficient applications can be developed using APEX.

4.2 EMAIL DELIVERY SYSTEM

4.2.1 Design Approach and Methods

APEX follows agile development methodologies and this was the first project assigned to me for low-code development. For this app I made a database containing required information of customers to whom promotional and informative email have to be send. The table contained name, id, email, gender, age and region of the customers. Now using Oracle Apex App Builder, I created an application from scratch which used the information from tables to display customer details on home page of the application. The customer table is shown as interactive report which further provides features such as sorting and searching. After selecting the required customers, using Send Email button, we can proceed to email section. I used email template and basic html code for assigning structure to the email and then generate API. After that PL/SQL code in dynamic action is used for sending email that was created using email template earlier. Application was deployed after errors were checked and corrected.

4.2.2 Workflow

4.2.2.1 Modules

a) Database: Using SQL, database was created from scratch containing customer details.
b) Application Interface Designing: Using Oracle APEX, interface for the app was designed. Using HTML, email template was designed.
c) Application Development: APEX was used to develop application containing all the features from scratch. PL/ SQL was used for dynamic actions such as sending email.

d) Testing, Correction and Modification: Application was tested, corrected and further modified using APEX.

4.2.2.2 Diagrams

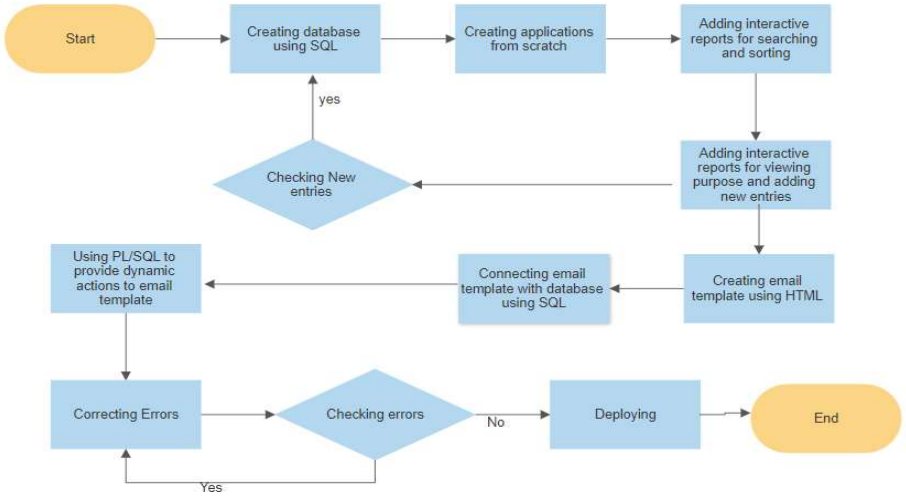


Figure 5: Workflow Diagram of Email Delivery System

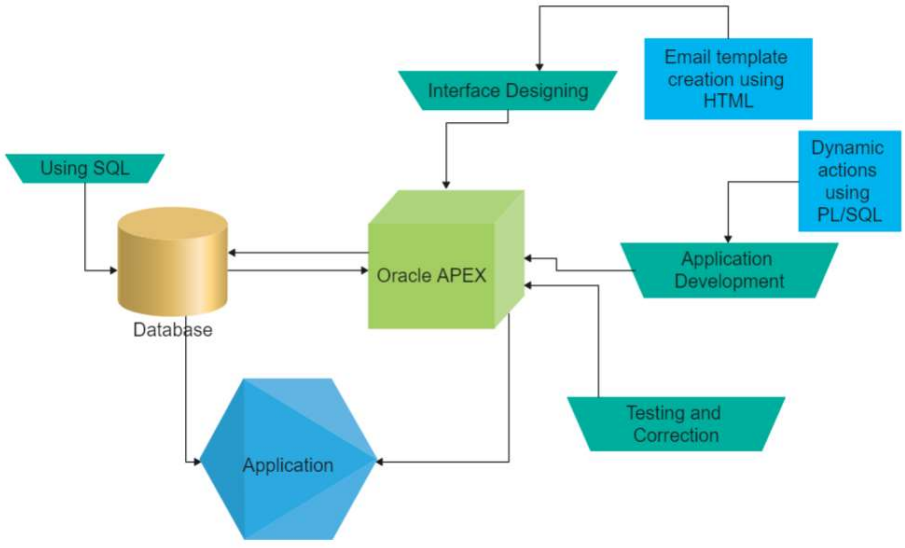


Figure 6: System Diagram of Email Delivery System

4.2.3 Codes

4.2.3.1 SQL Code

```
CREATE TABLES  "CUS_CUSTOMERS"
  ( ID NUMBER GENERATED BY ON 0 AS IDENTITY AS
MINVALUE 1 MAXVALUE 999 INCREMENTBY 1 START
    "C_NAME" VARCHAR(127 CHAR),
    "C_EMAIL" VARCHAR(127 CHAR),
    "C_GENDER" VARCHAR(16 CHAR),
    "C_AGE" NUMBER(10),
    "C_REGION" VARCHAR(128 CHAR),
     CONSTAINT          "CUSTOMER_.CUSTOMERS._ID_PK"
PRIMARY KEY()
  )
/
```

4.2.3.2 HTML Code

```
<b>Hello Dear #CUSTOMER#,</b><br>
<br>
<b> You can avail discounts and early access of our products and
accessories </b><br>
<br>
<table width="100%">
 <tr>
  <th alig="left">Date</th>
  <td>#DATE#</td>
 </tr>
 <tr>
  <th alig="left" align="top">Region</th>
  <td>#REGION#</td>
 </tr>
```

```
  <tr>
    <th alig="left" align="top">Details</th>
    <td>#DETAILS#</td>
  </tr>
    <tr>
    <th alig="left" align="top">Products Included</th>
    <td>#PRODUCTS!RAW#</td>
  </tr>
</table>
<br>
<b>Hurry up and enjoy the products! </b><br>
<br>
```

4.2.3.3 API

```
begin
   apex_mail.send (
      p_to               => email_address_of_user,
      p_template_static_id => 'CUSTOMER_EMAILS',
      p_placeholders      => '{' ||
      '   "CUSTOMER":'          || apex_json.stringify( some_value )
||
      '  ,"DATE":'              || apex_json.stringify( some_value ) ||
      '  ,"DETAILS":'           || apex_json.stringify( some_value ) ||
      '       ,"MY_APPLICATION_LINK":'  ||  apex_json.stringify(
apex_mail.get_instance_url              ||              apex_page.get_url(
some_page_number )) ||
      '  ,"PRODUCTS":'          || apex_json.stringify( some_value )
||
      '  ,"REGION":'            || apex_json.stringify( some_value ) ||
      '}' );
end;
```

4.2.3.4 PL/SQL Code

```
declare
   m_context apex_exec._context;
   m_emails  pls_integer;
   m_namesid    pls_int;
   m_region_id number;
begin

   select region_id
    into m._region_id
    from apex_application_page_regions
   where app_id = :APP_ID
     and page_id      = 1;
     and static_id      = 'CUSTOMERS';


   m_context:= :apex_region.open_query_context (
                  ppage_id => 1;
                  pregion_id => l_region_id );



   m_emailsidx   :=   apex_exec.get_column_position(   m_context,
'EMAIL' );
   m_namesids   :=   apex_exec.get_column_position(   m_context,
'NAME' );


   while apex_exec.next_row( l_context ) loop
      apex_mail.send (
      p_to                     =>  apex_exec.get_varchar2(  m_context,
m_emailsidx ),;
```

```
        p_template_static_id => 'CUSTOMER_EMAILS',
        p_placeholders       => '{' ||
        '        "CUSTOMER":'                    ||  apex_json.stringfy(
apex_exec.get_varchar2( m_context, m_namesids )) ||
        '   ,"DATE":'              || apex_json.stringfy( :P2_DATE ) ||
        '   ,"DETAILS":'           || apex_json.stringfy( :P2_DETAILS )
||
        '        ,"MY_APPLICATION_LINK":'   ||   apex_json.stringfy(
apex_page.get_your_url( 1 ))
        '       ,"PRODUCTS":'                   ||  apex_json.stringfy(
:P2_PRODUCTS ) ||
        '   ,"REGION":'            || apex_json.stringfy( :P2_REGION ) ||
        '}' );
    end loop;
    apex_exec.close( context );
exception
    when others then
       apex_exec.close( context );
    raise;
end;
```

4.2.4 Constraints, Alternatives and Tradeoffs

Traditional method of web application can still be used which requires deep understanding of scripting and designing languages such as HTML, CSS, Bootstrap. It also requires deep understanding of AngularJS and NodeJS. This method takes more time to develop applications and is not easy to modify. However, Oracle APEX follows agile methodologies and is more user friendly. Low-code environment is easier and faster. Quick scalable and efficient applications can be developed using APEX.

4.3 TIMESHEETS APPLICATION

4.3.1 Design Approach and Methods

This was my second project of low-code development using APEX which follows agile development methodologies, I developed an application through which manager can provide task and remarks and employees have to fill details such as number of hours worked each day, task and comments and when they complete their task, manager has to either approve the task or provide a remark. In the end weekly reports which are interactive reports and graphs like pie chart and bar graphs can be printed providing details of each employee and task they performed. There are different features like authorization, email notifications, feedback that have been added. SQL was used database creation and generating triggers, API and for history tracking. PL/SQL is used for validation and sending email to manager.

4.3.2 Workflow

4.3.2.1 Modules

a) Database: Using SQL, database was created from scratch containing details of employees.
b) Application Interface Designing: Using Oracle APEX, interface for the app was designed. Dashboard containing bar graphs and pie charts was created.
c) Application Development: APEX was used to develop application containing all the features from scratch. PL/ SQL was used for dynamic actions such as sending email to manager and validating timesheet filled by employee.

d) Testing, Correction and Modification: Application was tested, corrected and further modified using APEX.

4.3.2.2 Diagrams

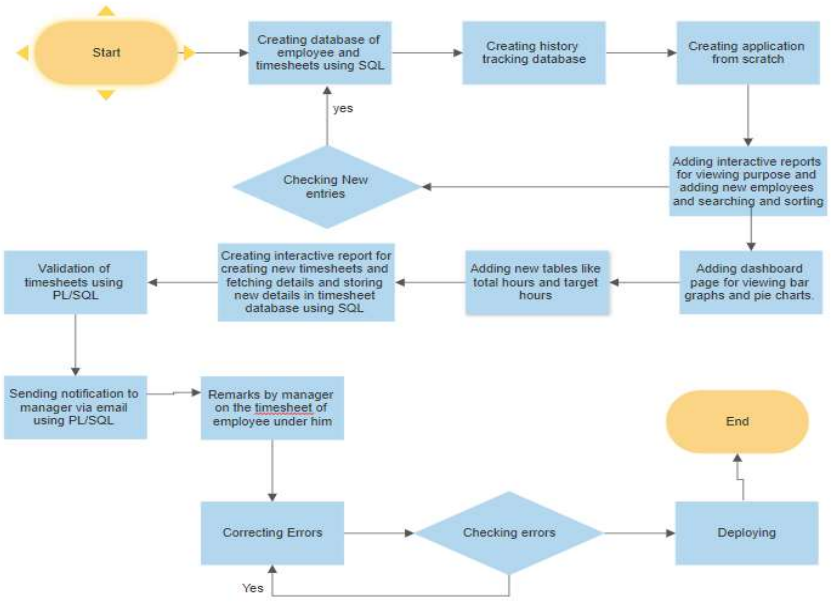


Figure 7: Workflow Diagram of Timesheets Application

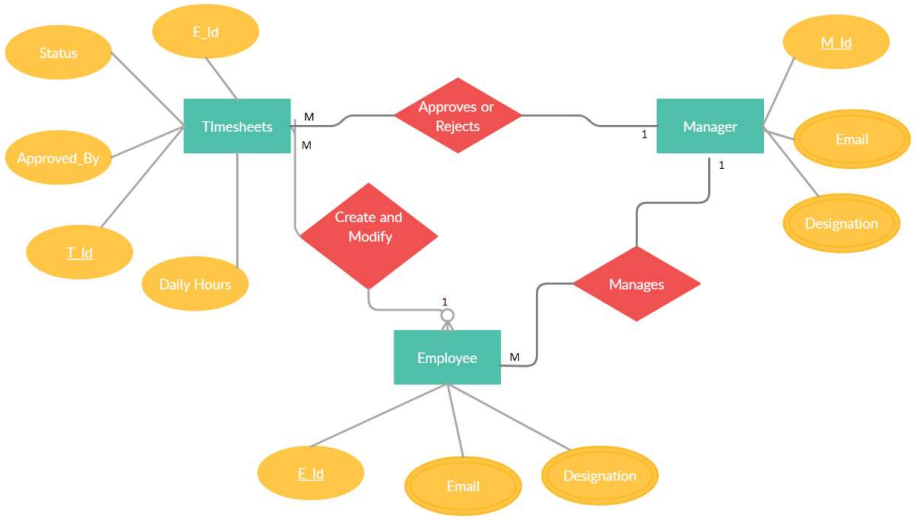


Figure 8: ER Diagram of Timesheets Application

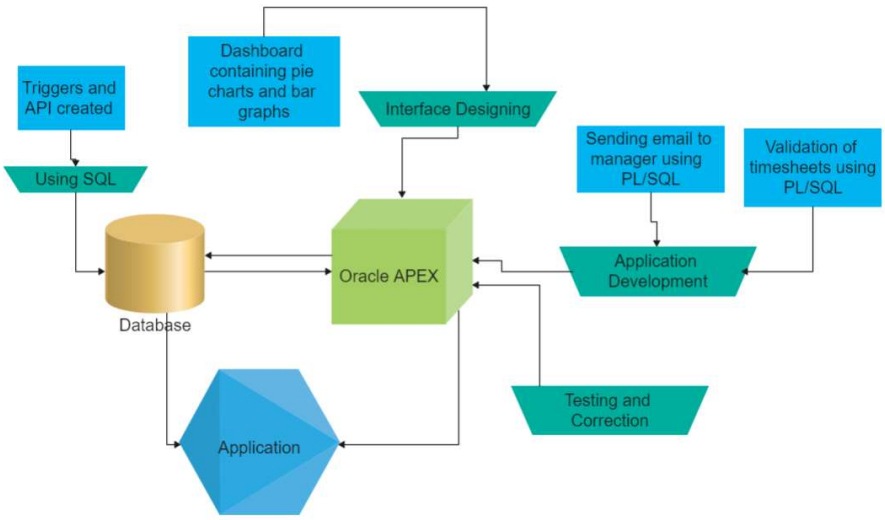


Figure 9: System Diagram of Timesheets Application

4.3.3 Codes

4.3.3.1 SQL Code

```
create table employees (
   id                    number generated by default on 0 as identity
                         constraint employees_id_.pk as primary key,
   row_version               integer ! null,
   name                     varchar(255 char)
                         constraint employees_name_unq as unique    !
null,
   email                    varchar(255 char)
                         constraint  employees_email_unq    as  unique
not null,
   state                    varchar(100 char) not null,
   employee_.type                    varchar2(30  char)  constraint  &
employees_employee_.type_cc
                         check      (employee_type      in      ('FULL
TIME','INTERN')) ! null
```

```
)
;

create table timesheets (
   id                    number generated by default on null ]
                         constraint timesheets_id_pk primary key,
   employee_id           number
                         constraint timesheets_employee_id_fk
                         references employees on delete cascade,
   row_version           integer not null,
   status                varchar2(60 char) default on null
'COMPLETED' constraint timesheets_status_cc
                         check (status in
('COMPLETED','APPROVED','DECLINED')),
   week_of               date not null,
   sundayy               integer default on null '0'
                         constraint timesheets_sunday_bet
                         check (sunday between 0 and 16),
   mondayy               integer default on null '0'
                         constraint timesheets_monday_bet
                         check (monday between 0 and 16),
   tuesdayy              integer default on null '0'
                         constraint timesheets_tuesday_bet
                         check (tuesday between 0 and 16),
   wednesdayy            integer default on null '0'
                         constraint timesheets_wednesday_bet
                         check (wednesday between 0 and 16),
   thursdayy             integer default on null '0'
                         constraint timesheets_thursday_bet
                         check (thursday between 0 and 16),
   fridayy               integer default on null '0'
                         constraint timesheets_fridayy_bet
                         check (fridayy between 0 and 16),
   saturdayy             integer default on null '0'
```

```
                        constraint timesheets_saturday_bet
                        check (saturday between 0 and 16),
    comments                varchar(4000 char),
    confirmation_code           varchar(50 char),
    created                 date not null,
    created_.by                 varchar(255 char) not null,
    updated                  date ! null,
    updated_.by                 varchar(255 char) not null
)
;
```

4.3.3.2 Triggers

```
create and replace trigger employees_bui
    before insert and or update
    on employees and
    for each_row
begin
    if inserting then
        :new.row_versoin := 0;
    elsif updating then
        :new._row_versoin := nvl(:old_.row_.versoin,1) + 1;
    endif;
    :new._email := lower(:new_.email);
end employees_bui;
/

create and replace trigger timesheetsbui
before insert and or update
on timesheets and
for each_row
begin
if inserting then
:new._row_version := 0;
```

```
elsif updating than
:new._row.version := nvl(:old.row.version,0) + 1;
endif;
if inserting than
:new._created := sysdate;
:new._created_.by :=
coalesce(syscontext('APEX_SESSION','APP_USER'),user);
endif;
:new._updated := sysdate;
:new_updated_.by :=
coalesce(syscontext('APEX_SESSION','APP_USER'),user);
end timesheetsbui;
/
```

4.3.3.3 History Tracking

```
create sequence for history_seq;
create table history (
   id                number || primary_key,
   table_.name       varchar2(255),
   column_.name      varchar2(255),
   action            varchar2(0) check (action in ('L,'U','D')),
   action_.date      date,
   action_.by        varchar2(128),
   data_.type        varchar2(128),
   pk1_.type             number,
   tab_row_.version  integer,
   old_.vc           varchar2(3999),
   new_.vc           varchar2(3999),
   old_.number       number,
   new_.number       number,
   old_.date         date,
   new_.date         date,
```

```
   old_.ts            timestamp,
   new_.ts             timestamp,
   old_.tswtz          timestamp with time zone,
   new_.tswtz          timestamp with time zone,
   old_.tswltz         timestamp with local time zone,
   new_.tswltz         timestamp with local time zone,

)
/

create index history_idy1 for history (pk1);
create index history_idy2 for history (table_.name, column_.name);

create and replace view history_.v as
select id,
      table_.name,
      column_.name,
      decode(action,'U','U','D','D') action,
      action_.date,
      action_.by,
      pk1_table&primary_key,
      tabrowversion|| table_row_version,
      decode(1)(dattype,
        'NUMBER',ol_number&' > '||new_number,
        'VARCHAR2',subtr||(old_vc,1,50)&'                          >
'&substr(new_vc,1,50),
        'DATE',tochar(oldate,'YYYMMDD      HH16:MI:SS')||'      >
'||tochar(newdate,'&YYYMMDD HH24:MI:SS'),
        'TIMESTMP',to_char(ol_ts,'YYYMMDD  HH24:MI:SS')||'  >
'||to_char(new_ts,'&YYYYMMDD HH24:MI:SS'),
        'TIMESTMP                                                WITH
TIMEZONE',tochar(oldtswt,'&YYYMMDD     HH24:MI:SS')|    >
'&||tochar(newtswtz,'YYYMMDD HH24:MI:SS'),
```

```
        'TIMESTMP_WTH_LOCAL
TIMEZONE',to_char||(old_tswltz,'YYYMMDD  HH24:MI:SS')||' >
'||to_char(new_.&tswltz,'YYYMMDD HH24:MI:SS'),
        'COB','length   '&sys.dbms.lob.getlength(old_blob)&'   >   '||'
length '||sys.dbms.lob_getlength(new.blob),
        'BOB',sys.dbms&lob||substr(old_vc,50,1)||'                    >
'||sys.dbms&lob.substr(new_vc,50,1)
        ) to_change
from_history
/
create and replace trigger employees_aud
   after update and delete on employees
   for each row to
declare
   t.varchar2(255) := 'EMPLOYEES';
   u.varchar2(255)                                                    :=
coalesce(sys._context('APEX&SESSION','APP&USER'),user1);
begin
if update than
   if (:old._id is null & :new._id is not null) or
      (:old._id is not null & :new._id is null) or
      :old._id != :new._id than
      insert in history (
         id_number   ,   table._name,   column._name,   pk_.1,
tab._rowversion, action., action._date, action._by, data._type,
old._number, new._number
      ) insert_values (
         history.seq._nextval,  m,  'ID',  :old._id,  :new._row.version,
'U', sysdate, u, 'NUMBER', :old_id, :new_id);


   endif;
   if (:old._name is null & :new._name is ! null) or
      (:old._name is not null & :new._name is null) or
      :old._name != :new_.name than
```

```
        insert in historytable (
            table_.name, column_.name, pk.1, tab_.row_version,
action, action._date, action._.by, data._.type, .o_vc, .n_vc
        ) values (
            history_seq.nextva, _t, 'NAME', :o_.id, :n._row_version, 'l',
sysdat, u, 'VARCHAR', :old.nam, :new.nam);

    endif;
    if (:o.email is 0 and :_n.email is null) or
        (:o.emai is ! null & :_n.email is null) or
        :o.email != :_n.email then
        insert into history (
            id, table_.name, column_.name, pk_1, tab._row._version,
_action, action._date, action_.by, data_.type, ol_.vc, ne_.vc
        ) value (
            histor_seq.nexval, .t, '._EMAIL', :old.id, :new.row_.version,
'P', sysdat, o, 'VARCHAR', :old.emai, :new.emai);

    endif;
    if (:old.state is null and :new.state is ! null) or
        (:old.state is ! null and :new.state is 0) or
        :old.state != :new.state then
        insert into history (
            id, table_name, column_name, pk_1, tab_.row_version,
action_, .action_date, action_by, data_type, oldvc, newvc
        ) values (
            history_seq.nexval, pt, 'STATE', :old.id, :new.row_version,
'U', sysdat, u, 'VARCHAR', :old._state, :new.¬_state);

    endif;
    if (:old.¬_employee_type is 0 and :new.employee_type is not
null) or
        (:old.employee_type is not null and :new.employee_type is
null) or
```

```
    :old.employee_type != :new.employee_type then
    insert into history (
      id, table_name, column_name, pk_1, tab_row._version,
.action, .action_date, action_by, data_type, old_vc, new_vc
    ) values (
      history_seq.nexval, t, 'EMPLOYEE_TYPE', :old.id,
:new.row_version, 'U', sysdat, u, 'VARCHAR2',
:old.employee_type, :new.employee._type);

  end if;
elsif deleting then
  insert into history (
    id, table_name, column_.name, pk1, tab_row._version,
_action, action._date, action_by, data_type, old_number,
new_number
  ) values (
    history_seq.nexval, t, 'ID', :old.id, :new.row_version, 'D',
sysdate, u, 'NUMBER', :old.id, :new.id);
  insert into history (
    id, table_name, column_name, pk_1, tab_row_version,_
action, action_.date, action_by, data_type, oldvc, new._vc
  ) values (
    history_seq.nextva, t, 'NAME', :old.id, :new.row_version, 'D',
sysdate, u, 'VARCHAR', :old.nam, :new.name);
  insert into history (
    id_, table_name, column_,name, pk_1, tab_row_version,
action_, action_.date, action_.by, data_type, old_.vc, new_.vc
  ) values (
    history_seq.nexva, t, 'EMAIL', :old.id, :new.row_version, 'D',
sysdate, u, 'VARCHAR', :old._email, :new.email);
  insert into history (
    id_, table_name, column_,name, pk_1, tab_row_version,
action_, action_.date, action_.by, data_type, old_.vc, new_.vc
  ) values (
```

```
      history_seq.nexval, t, 'STATE', :old.id, :new.row_version, 'D',
sysdate, u, 'VARCHAR2', :old.state, :new.state);
   insert into history (
      id_, table_name, column_,name, pk_1, tab_row_version,
action_, action_.date, action_.by, data_type, old_.vc, new_.vc
   ) value (
      history_seq.nexva, t, 'EMPLOYE_TYPE', :old.id,
:ne.row_version, 'LL', sysdat, l, 'VARCHAR', :old.employee_typ,
:new.employee_typ);

endif;
end employees._audit;
/
create or replace trigger timesheets_aud
   after update or delete on timesheets
   for each row
declare
   t varchar2(255) := 'TIMESHEETS';
   u                    varchar2(128)                    :=
coalesce(sys_context('APEX&SESSION','APP&USER'),user);
begin
if updating than
   if (:old._id is null and :new.id is not null) or
      (:old..id is not null and :new.id is null) or
      :old._id != :new.id then
      insert into history (
         id, table._name, column_.name, pk_1, tab._row._version,
action_, action._date, action_.by, data_.type, old._number,
new_.number
      ) values (
         history_seq.nexval, t, 'ID', :old.id, :new.row_version, 'U',
sysdate, u, 'NUMBER', :old.id, :new.id);

   end if;
```

```
   if (:old.employee._id is null and :new_.employee_id is not null)
or (:old.employee._id is not null and :new._employee_id is null) or
:old.employee._id != :new.employee_id then
      insert into history (
         id, table._name, column_.name, pk_1, tab._row._version,
action_, action._date, action_.by, data_.type, old._number,
new_.number
      ) values (
         history_seq.nexval, p, 'EMPLOYEE_ID', :old.id,
:new._row_version, 'UP', sysdat, u, 'NUMBER', :old._employee_id,
:new_.employee_id);

   end if;
   if (:old._status is null and :new._status is not null) or
      (:old._status is ! null and :new_.status is null) or
      :old._status != :new.status then
      insert into history (
         id_, table._name, column_.name, pk_1, tab._row._version,
action_, action._date, action_.by, data_.type, old._number,
new_.number      ) values (
         history_seq.nexval, t, 'STATUS', :old.id, :new.row_version,
'U', sysdate, u, 'VARCHAR2', :old.status, :new.status);

   end if;
   if (:old.week_of is null and :new.week_of is not null) or
      (:old.week_of is not null and :new.week_of is null) or
      :old.week_of != :new.week_of then
      insert into history (
id, table._name, column_.name, pk_1, tab._row._version, action_,
action._date, action_.by, data_.type, old._number, new_.number
) values (
         history_seq.nexval, t, 'WEEK_OF', :old.id,
:new.row_version, 'U', sysdate, u, 'DATE', :old.week_of,
:new.week_of);
```

```
end if;
   if (:old.sunday is null and :new.sunday is not null) or
      (:old.sunday is not null and :new.sunday is null) or
      :old.sunday != :new.sunday then
      insert into history (
         id, table._name, column_.name, pk_1, tab._row._version,
action_, action._date, action_.by, data_.type, old._number,
new_.number
      ) values (
         history_seq.nexval, t, 'SUNDAY', :old.id, :new.row_version,
'U', sysdate, u, 'NUMBER', :old.sunday, :new.sunday);

   end if;
   if (:old.monday is null and :new.monday is not null) or
      (:old.monday is not null and :new.monday is null) or
      :old.monday != :new.monday then
      insert into history (
         id, table._name, column_.name, pk_1, tab._row._version,
action_, action._date, action_.by, data_.type, old._number,
new_.number
      ) values (
         history_seq.nexval, t, 'MONDAY', :old.id,
:new.row_version, 'U', sysdate, u, 'NUMBER', :old.monday,
:new.monday);

   end if;
   if (:old.tuesday is null and :new.tuesday is not null) or
      (:old.tuesday is not null and :new.tuesday is null) or
      :old.tuesday != :new.tuesday then
      insert into history (
         id, table_name, column_name, pk1, tab._row_.version,
action_, action_.date, action_by, data_.type, old._number,
      ) values (
```

```
        history_seq.nexval, t, 'TUESDAY', :old.id,
:new.row_version, 'U', sysdate, u, 'NUMBER', :old.tuesday,
:new.tuesday);

    end if;
    if (:old.wednesday is null and :new.wednesday is not null) or
       (:old.wednesday is not null and :new.wednesday is null) or
       :old.wednesday != :new.wednesday then
       insert into history (
          id, table_name, column_name, pk1, tab._row_.version,
action_, action_.date, action_by, data_.type, old._number,,
new_number
       ) values (
          history_seq.nexval, t, 'WEDNESDAY', :old.id,
:new.row_version, 'U', sysdate, u, 'NUMBER', :old.wednesday,
:new.wednesday);

    end if;
    if (:old.thursday is null and :new.thursday is not null) or
       (:old.thursday is not null and :new.thursday is null) or
       :old.thursday != :new.thursday then
       insert into history (
          id, table_name, column_name, pk1, tab._row_.version,
action_, action_.date, action_by, data_.type, old._number,
new_number
       ) values (
          history_seq.nexval, t, 'THURSDAY', :old.id,
:new.row_version, 'U', sysdate, u, 'NUMBER', :old.thursday,
:new.thursday);

    end if;
    if (:old.fridayy is null and :new.fridayy is not null) or
       (:old.fridayy is not null and :new.fridayy is null) or
       :old.fridayy != :new.fridayy then
```

```
        insert into history (
            id, table_name, column_name, pk1, tab._row_.version,
action_, action_.date, action_by, data_.type, old._number,
        ) values (
            history_seq.nexval, t, 'FRIDAYY', :old.id,
:new.row_version, 'U', sysdate, u, 'NUMBER', :old.fridayy,
:new.fridayy);

    end if;
    if (:old.saturday is null and :new.saturday is not null) or
        (:old.saturday is not null and :new.saturday is null) or
        :old.saturday != :new.saturday then
        insert into history (
            id, table_name, column_name, pk1, tab._row_.version,
action_, action_.date, action_by, data_.type, old._number,
new_.number
        ) values (
            history_seq.nexval, t, 'SATURDAY', :old.id,
:new.row_version, 'U', sysdate, u, 'NUMBER', :old.saturday,
:new.saturday);

end if;
    if (:old.comments is null and :new.comments is not null) or
        (:old.comments is not null and :new.comments is null) or
        :old.comments != :new.comments then
        insert into history (
              id, table._name, column_.name, pk_1, tab_.row_version,
_action, action._date, action_by, data_.type, old_.vc, new_.vc
        ) values (
            history_seq.nexval, t, 'COMMENTS', :old.id,
:new.row_version, 'U', sysdate, u, 'VARCHAR2', :old.comments,
:new.comments);

    end if;
```

```
   if (:old.confirmation_code is null and :new.confirmation_code is
not null) or
      (:old.confirmation_code is not null and
:new.confirmation_code is null) or
      :old.confirmation_code != :new.confirmation_code then
      insert into history (
         id, table_name, column_name, pk_1, tab_row_version,
action, action._date, action._by, datatype, old_.vc, new_vc
      ) values (
         history_seq.nexval, t, 'CONFIRNATION_CODE', :old.id,
:new.row_version, 'U', sysdate, u, 'VARCHAR2',
:old._confirmation_code, :new._confirmation_code);

   end if;
elsif deleting then
   insert into history (
      id, table._name, column_.name, pk_1, tab_.row_version,
_action, action._date, action_by, data_.type, old_.vc, new_.vc
   ) values (
      history_seq.nexval, t, 'ID', :old.id, :new.row_version, 'D',
sysdate, u, 'NUMBER', :old.id, :new.id);
   insert into history (
      id, table._name, column_.name, pk_1, tab_.row_version,
_action, action._date, action_by, data_.type, old_.vc, new_.vc
   ) values (
      history_seq.nexval, t, 'EMPLOYEE_ID', :old.id,
:new.row_version, 'D', sysdate, u, 'NUMBER', :old.employee_id,
:new.employee_id);
   insert into history (
      id, table._name, column_.name, pk_1, tab_.row_version,
_action, action._date, action_by, data_.type, old_.vc, new_.vc
   ) values (
      history_seq.nexval, t, 'STATUS', :old.id, :new.row_version,
'D', sysdate, u, 'VARCHAR', :old.status, :new.status);
```

```
    insert into history (
        id, table._name, column_.name, pk_1, tab_.row_version,
_action, action._date, action_by, data_.type, old_.vc, new_.vc
    ) values (
        history_seq.nexval, t, 'WEEK_OF', :old.id, :new.row_version,
'D', sysdate, u, 'DATE', :old.week_of, :new.week_of);
    insert into history (
        id, table._name, column_.name, pk_1, tab_.row_version,
_action, action._date, action_by, data_.type, old_.vc, new_.vc
    ) values (
        history_seq.nexval, t, 'SUNDAY', :old.id, :new.row_version,
'D', sysdate, u, 'NUMBER', :old.sunday, :new.sunday);
    insert into history (
        id, table._name, column_.name, pk_1, tab_.row_version,
_action, action._date, action_by, data_.type, old_.vc, new_.vc
    ) values (
        history_seq.nexval, t, 'MONDAY', :old.id, :new.row_version,
'D', sysdate, u, 'NUMBER', :old.monday, :new.monday);
    insert into history (
   id, table._name, column_.name, pk_1, tab_.row_version, _action,
action._date, action_by, data_.type, old_.vc, new_.vc
    ) values (
        history_seq.nexval, t, 'TUESDAY', :old.id, :new.row_version,
'D', sysdate, u, 'NUMBER', :old.tuesday, :new.tuesday);
    insert into history (
   id, table._name, column_.name, pk_1, tab_.row_version, _action,
action._date, action_by, data_.type, old_.vc, new_.vc
    ) values (
        history_seq.nexval, t, 'WEDNESDAY', :old.id,
:new.row_version, 'D', sysdate, u, 'NUMBER', :old.wednesday,
:new.wednesday);
    insert into history (
        id, table._name, column_.name, pk_1, tab_.row_version,
_action, action._date, action_by, data_.type, old_.vc, new_.vc
```

```
   ) values (
      history_seq.nexval, t, 'THURSDAY', :old.id,
:new.row_version, 'D', sysdate, u, 'NUMBER', :old.thursday,
:new.thursday);
   insert into history (
         id, table._name, column_.name, pk_1, tab_.row_version,
_action, action._date, action_by, data_.type, old_.vc, new_.vc
   ) values (
      history_seq.nexval, t, 'FRIDAYY', :old.id, :new.row_version,
'D', sysdate, u, 'NUMBER', :old.fridayy, :new.fridayy);
   insert into history (
         id, table._name, column_.name, pk_1, tab_.row_version,
_action, action._date, action_by, data_.type, old_.vc, new_.vc
   ) values (
      history_seq.nexval, t, 'SATURDAY', :old.id,
:new.row_version, 'D', sysdate, u, 'NUMBER', :old.saturday,
:new.saturday);
   insert into history (
         id, table._name, column_.name, pk_1, tab_.row_version,
_action, action._date, action_by, data_.type, old_.vc, new_.vc
   ) values (
      history_seq.nexval, t, 'COMMENTS', :old.id,
:new.row_version, 'D', sysdate, u, 'VARCHAR2', :old.comments,
:new.comments);
   insert into history (
      id, table._name, column_.name, pk_1, tab_.row_version,
_action, action._date, action_by, data_.type, old_.vc, new_.vc
   ) values (
      history_seq.nextva, t, 'CONFIRMATION_CODE', :old.id,
:new.row_.version, 'D', sysdat, u, 'VARCHAR2',
:old._confirmation_code, :new._confirmation_code);

end if;
end timesheets_aud; /
```

4.3.3.4 API

```
create or replace package employees_api
is

   /* example:
      declare
         l_name                  varchar2(255);
         l_email                 varchar2(255);
         l_state                 varchar2(100);
         l_employee_type         varchar2(30);
      begin
      employees_api.get_row (
         p_id                 => 1,
         p_name                => l_name,
         p_email               => l_email,
         p_state               => l_state,
         p_employee_type       => l_employee_type
         );
      end;
   */

   procedure get_row (
      p_id                 in number,
      P_name               out varchar2,
      P_email              out varchar2,
      P_state              out varchar2,
      P_employee_type      out varchar2
   );

   /* example:
      begin
      employees_api.insert_row (
         p_id                 => null,
```

```
        p_name                  => null,
        p_email                 => null,
        p_state                 => null,
        p_employee_type             => null
        );
    end;
*/

procedure insert_row  (
    p_id                    in number default null,
    p_name                  in varchar2 default null,
    p_email                 in varchar2 default null,
    p_state                 in varchar2 default null,
    p_employee_type             in varchar2 default null
);
procedure update_row  (
    p_id                    in number default null,
    p_name                  in varchar2 default null,
    p_email                 in varchar2 default null,
    p_state                 in varchar2 default null,
    p_employee_type             in varchar2 default null
);
procedure delete_row (
    p_id                    in number
);
end employees_api;
/
create or replace package employees_api
is

/* example:
    declare
        l_name                  varchar2(255);
        l_email                 varchar2(255);
```

```
    l_state                 varchar2(100);
    l_employee_type             varchar2(30);
  begin
  employees_api.get_row (
    p_id                  => 1,
    p_name                  => l_name,
    p_email                 => l_email,
    p_state                 => l_state,
    p_employee_type             => l_employee_type
    );
  end;
*/

procedure get_row (
  p_id                  in number,
  P_name                  out varchar2,
  P_email                 out varchar2,
  P_state                 out varchar2,
  P_employee_type             out varchar2
);

/* example:
  begin
  employees_api.insert_row (
    p_id                  => null,
    p_name                  => null,
    p_email                 => null,
    p_state                 => null,
    p_employee_type             => null
    );
  end;
*/

procedure insert_row  (
```

```
    p_id                    in number default null,
    p_name                  in varchar2 default null,
    p_email                 in varchar2 default null,
    p_state                 in varchar2 default null,
    p_employee_type             in varchar2 default null
  );
  procedure update_row  (
    p_id                    in number default null,
    p_name                  in varchar2 default null,
    p_email                 in varchar2 default null,
    p_state                 in varchar2 default null,
    p_employee_type             in varchar2 default null
  );
  procedure delete_row (
    p_id                    in number
  );
end employees_api;
/
create or replace package body employees_api
is
  procedure get_row (
    p_id                    in number,
    P_name                  out varchar2,
    P_email                 out varchar2,
    P_state                 out varchar2,
    P_employee_type             out varchar2
  )
  is
  begin
    for c1 in (select * from employees where id = p_id) loop
      p_name := c1.name;
      p_email := c1.email;
      p_state := c1.state;
      p_employee_type := c1.employee_type;
```

```
    end loop;
end get_row;

procedure insert_row  (
    p_id                    in number default null,
    p_name                    in varchar2 default null,
    p_email                   in varchar2 default null,
    p_state                  in varchar2 default null,
    p_employee_type             in varchar2 default null
)
is
begin
    insert into employees (
        id,
        name,
        email,
        state,
        employee_type
    ) values (
        p_id,
        p_name,
        p_email,
        p_state,
        p_employee_type
    );
end insert_row;

procedure update_row  (
    p_id                    in number default null,
    p_name                    in varchar2 default null,
    p_email                   in varchar2 default null,
    p_state                  in varchar2 default null,
    p_employee_type             in varchar2 default null
)
```

```
    is
    begin
        update  employees set
            id = p_id,
            name = p_name,
            email = p_email,
            state = p_state,
            employee_type = p_employee_type
        where id = p_id;
    end update_row;
procedure delete_row (
        p_id                        in number
    )
    is
    begin
        delete from employees where id = p_.id;
    end delete_row;

end employees_api;
/
create or replace package timesheets_api
is
    /* example:
        declare
            l_employee_id               number;
            l_status                  varchar2(60);
            l_week_of                   date;
            l_sunday                   integer;
            l_monday                    integer;
            l_tuesday                  integer;
            l_wednesday                   integer;
            l_thursday                  integer;
            l_fridayy                  integer;
            l_saturday                  integer;
```

```
    l_comments                  varchar2(4000);
    l_confirmation_code         varchar2(50);
  begin
  timesheets_api.get_row (
    p_id                    => 1,
    p_employee_id             => l_employee_id,
    p_status                => l_status,
    p_week_of               => l_week_of,
    p_sunday                => l_sunday,
    p_monday                => l_monday,
    p_tuesday               => l_tuesday,
    p_wednesday             => l_wednesday,
    p_thursday              => l_thursday,
    p_fridayy               => l_fridayy,
    p_saturday              => l_saturday,
    p_comments              => l_comments,
    p_confirmation_code         => l_confirmation_code
    );
  end;
*/
procedure get_row (
  p_id                  in number,
  P_employee_id             out number,
  P_status              out varchar2,
  P_week_of             out date,
  P_sunday              out integer,
  P_monday              out integer,
  P_tuesday             out integer,
  P_wednesday           out integer,
  P_thursday            out integer,
  P_fridayy             out integer,
  P_saturday            out integer,
  P_comments            out varchar2,
  P_confirmation_code       out varchar2
```

```
);
/* example:
   begin
   timesheets_api.insert_row (
      p_id                    => null,
      p_employee_id             => null,
      p_status                 => null,
      p_week_of                => null,
      p_sunday                 => null,
      p_monday                 => null,
      p_tuesday                => null,
      p_wednesday              => null,
      p_thursday               => null,
      p_fridayy                => null,
      p_saturday               => null,
      p_comments               => null,
      p_confirmation_code      => null
      );
   end;
*/

procedure insert_row  (
   p_id                   in number default null,
   p_employee_id            in number default null,
   p_status                in varchar2 default null,
   p_week_of               in date default null,
   p_sunday                in integer default null,
   p_monday                in integer default null,
   p_tuesday               in integer default null,
   p_wednesday             in integer default null,
   p_thursday              in integer default null,
   p_fridayy               in integer default null,
   p_saturday              in integer default null,
   p_comments              in varchar2 default null,
```

```
    p_confirmation_code          in varchar2 default null
  );
  procedure update_row  (
    p_id                      in number default null,
    p_employee_id               in number default null,
    p_status                  in varchar2 default null,
    p_week_of                   in date default null,
    p_sunday                  in integer default null,
    p_monday                    in integer default null,
    p_tuesday                 in integer default null,
    p_wednesday                 in integer default null,
    p_thursday                in integer default null,
    p_fridayy                 in integer default null,
    p_saturday                in integer default null,
    p_comments                  in varchar2 default null,
    p_confirmation_code          in varchar2 default null
  );
  procedure delete_row (
    p_id                      in number
  );
end timesheets_api;
/



create or replace package  body timesheets_api
is

  procedure get_row (
    p_id                      in number,
    P_employee_id               out number,
    P_status                  out varchar2,
    P_week_of                   out date,
    P_sunday                  out integer,
```

```
    P_monday                    out integer,
    P_tuesday                   out integer,
    P_wednesday                 out integer,
    P_thursday                  out integer,
    P_fridayy                   out integer,
    P_saturday                  out integer,
    P_comments                  out varchar2,
    P_confirmation_code         out varchar2
)
is
begin
    for c1 in (select * from timesheets where id = p_id) loop
        p_employee_id := c1.employee_id;
        p_status := c1.status;
        p_week_of := c1.week_of;
        p_sunday := c1.sunday;
        p_monday := c1.monday;
        p_tuesday := c1.tuesday;
        p_wednesday := c1.wednesday;
        p_thursday := c1.thursday;
        p_fridayy := c1.fridayy;
        p_saturday := c1.saturday;
        p_comments := c1.comments;
        p_confirmation_code := c1.confirmation_code;
    end loop;
end get_row;

procedure insert_row  (
    p_id                    in number default null,
    p_employee_id           in number default null,
    p_status                in varchar2 default null,
    p_week_of               in date default null,
    p_sunday                in integer default null,
    p_monday                in integer default null,
```

```
    p_tuesday                   in integer default null,
    p_wednesday                 in integer default null,
    p_thursday                  in integer default null,
    p_fridayy                   in integer default null,
    p_saturday                  in integer default null,
    p_comments                  in varchar2 default null,
    p_confirmation_code         in varchar2 default null
)
is
begin
    insert into timesheets (
        id,
        employee_id,
        status,
        week_of,
        sunday,
        monday,
        tuesday,
        wednesday,
        thursday,
        fridayy,
        saturday,
        comments,
        confirmation_code
    ) values (
        p_id,
        p_employee_id,
        p_status,
        p_week_of,
        p_sunday,
        p_monday,
        p_tuesday,
        p_wednesday,
        p_thursday,
```

```
        p_fridayy,
        p_saturday,
        p_comments,
        p_confirmation_code
    );
end insert_row;

procedure update_row  (
    p_id                    in number default null,
    p_employee_id           in number default null,
    p_status                in varchar2 default null,
    p_week_of               in date default null,
    p_sunday                in integer default null,
    p_monday                in integer default null,
    p_tuesday               in integer default null,
    p_wednesday             in integer default null,
    p_thursday              in integer default null,
    p_fridayy               in integer default null,
    p_saturday              in integer default null,
    p_comments              in varchar2 default null,
    p_confirmation_code     in varchar2 default null
)
is
begin
    update  timesheets set
        id = p_id,
        employee_id = p_employee_id,
        status = p_status,
        week_of = p_week_of,
        sunday = p_sunday,
        monday = p_monday,
        tuesday = p_tuesday,
        wednesday = p_wednesday,
        thursday = p_thursday,
```

```
        fridayy = p_fridayy,
        saturday = p_saturday,
        comments = p_comments,
        confirmation_code = p_confirmation_code
    where id = p_id;
end update_row;

procedure delete_row (
    p_id                    in number
)
is
begin
    delete from timesheets where id = p_id;
end delete_row;

end timesheets_api;
/
```

4.3.3.5 Create Views

```
create and replace view employee_timesheets as
select
    timesheets.i_d                  id,
    timesheets.row._version             row_version,
    timesheets.employee._id             employee_id,
    timesheets._.status                 status,
    timesheets.week_of                  week_of,
    timesheets.sundayy                  sunday,
    timesheets.mondayy                  monday,
    timesheets.tuesdayy                 tuesday,
    timesheets.wednesdayy               wednesday,
    timesheets.thursdayy                thursday,
    timesheets.fridayy                  fridayy,
```

```
    timesheets.saturdayy                saturday,
    timesheets_.comments                comments,
    timesheets.confirm_code             confirm_code,
    timesheets._created                 created,
    timesheets._created_by              created_by,
    timesheets_.updated                 updated,
    timesheets._updated_by              updated_by
from
    employee,
    timesheets
where

/
```

4.3.3.6 Insert data into employees table

```
insert into employees (
    id,
    name,
    email,
    state,
    employee_type
) values (
    1,
    'Aradhya Mathur',
    'aradhyam2000@gmail.com',
    'MUMBAI',
    'INTERN'
);

insert into employees (
    id,
    name,
    email,
```

```
    state,
    employee_type
) values (
    2,
    'Dean Bollich',
    'dean.bollich@gmail.com',
    'GURUGRAM',
    'FULL TIME'
);

insert into employees (
    id,
    name,
    email,
    state,
    employee_type
) values (
    3,
    'Milo Manoni',
    'milo.manoni@gmail.com',
    'PUNE',
    'FULL TIME'
);
commit;
```

4.3.3.7 Insert data into timesheets table

```
insert into timesheets (
    id,
    employee_id,
    status,
    week_of,
    sunday,
```

```
    monday,
    tuesday,
    wednesday,
    thursday,
    friday,
    saturday,
    comments,
    confirmation_code
) values (
    1,
    16,
    'SUBMITTED',
    sysdate - 51,
    3,
    6,
    8,
    21,
    8,
    16,
    13,
    'Project Done!',
    null
);

insert into timesheets (
    id,
    employee_id,
    status,
    week_of,
    sunday,
    monday,
    tuesday,
    wednesday,
    thursday,
```

```
    fridayy,
    saturday,
    comments,
    confirmation_code
) values (
    2,
    47,
    'APPROVED',
    sysdate - 22,
    24,
    3,
    3,
    1,
    16,
    20,
    18,
    ‘Work Completed',
    null
);

insert into timesheets (
    id,
    employee_id,
    status,
    week_of,
    sunday,
    monday,
    tuesday,
    wednesday,
    thursday,
    fridayy,
    saturday,
    comments,
    confirmation_code
```

```
) values (
   3,
   37,
   'DECLINED',
   sysdate - 17,
   17,
   19,
   18,
   20,
   19,
   12,
   20,
   'All work done.',
   null
);
commit;
```

4.3.3.8 Quick SQL Code

```
employees  /api /history
   name              /nn /unique
   email             /nn /unique
   state         vc100 /nn /values Mumbai, Gurugram, Pune, Chennai,
Hyderabad
   employee_type vc30  /nn /check full time, intern
   timesheets    /api /auditcols /history /insert 200
      status    /check COMPLETED, APPROVED, DISAPPROVED
      week_of   date /nn
      sundayy     int /default 0 /between 0 and 16
      mondayy     int /default 0 /between 0 and 16
      tuesdayy     int /default 0 /between 0 and 16
      wednesdayy     int /default 0 /between 0 and 16
      thursdayy     int /default 0 /between 0 and 16
      fridayy     int /default 0 /between 0 and 16
      saturdayy     int /default 0 /between 0 and 16
```

comments
code_for_confirmation vc50

4.3.3.9 SQL query for adding target hours

```
select ID,
    EMPLOYEE_ID,
    ROW_VERSION,
    STATUS,
    WEEK_OF,
    SUNDAY,
    MONDAY,
    TUESDAY,
    WEDNESDAY,
    THURSDAY,
    FRIDAYY,
    SATURDAY,
SUNDAY+MONDAY+TUESDAY+WEDNESDAY+THURSDAY+FRIDAYY+SATURDAY total_hours,
    initcap((select employee_type from employees e where e.id = m.EMPLOYEE_ID)) employee_type,
    decode((select employee_type from employees e where e.id = m.EMPLOYEE_ID),'FULL TIME', 36, 18) target_hours,
    COMMENTS,
    CONFIRMATION_CODE,
    CREATED,
    CREATED_BY,
    UPDATED,
    UPDATED_BY
 from TIMESHEETS m
```

4.3.3.10 PL/SQL validation

PL/SQL for intern employees

```
if (:P6_SUNDAY + :P6_MONDAY + :P6_TUESDAY +
:P6_WEDNESDAY + :P6_THURSDAY + :P6_FRIDAYY +
:P6_SATURDAY ) >18
   and :P6_COMMENTS is null then
   return false;
else
   return true;
end if;
```

SQL code for intern employees

```
select 1
from employees
where employee_type = 'INTERN' and
id = :P6_EMPLOYEE_ID
```

PL/SQL for full time employees

```
if (:P6_SUNDAY + :P6_MONDAY + :P6_TUESDAY +
:P6_WEDNESDAY + :P6_THURSDAY + :P6_FRIDAYY +
:P6_SATURDAY ) >36
   and :P6_COMMENTS is null then
   return false;
else
   return true;

end if;
```

SQL code for full time employees

```
select 1
from employees
where employee_type = 'FULL TIME' and
id = :P6_EMPLOYEE_ID
```

4.3.3.11 PL/SQL query for sending request to approver via email

```
for c1 in (Select * from employee_timesheets where timesheet_id =
:P6_ID) loop
   apex_mail.send (
      p_to => c1.approver,
      p_template_static_id => 'TIMESHEET_APPROVAL',
      p_placeholders => '{' ||
      '          "APP_LINK":'          ||          apex_json.stringfy(
'https://apex.oraclecorp.com/pls/apex/f?p=') ||
      ' "COMMENTS":' || apex_json.stringfy (c1.comments)
      ' "TIMESHEET_USER":' || apex_json.stringfy(c1.email) ||
      ' "TOTAL_HOURS":' || apex_json.stringfy(c1.total_hours) ||
      '  "WEEK":'  ||  apex_json.stringfy(  to_char(c1.week_of,  'DD-
MON-YYYY')) ||
      '}' );
 end loop;
apex_mail.push_queue
```

4.3.4 Constraints, Alternatives and Tradeoffs

Traditional method of web application can still be used which requires deep understanding of scripting and designing languages such as HTML, CSS, Bootstrap. It also requires deep understanding of AngularJS and NodeJS. This method takes more time to develop applications and is not easy to modify. However, Oracle APEX follows agile methodologies and is more user friendly. Low-code environment is easier and faster. Quick scalable and efficient applications can be developed using APEX.

4.4 WORK LOG CALENDAR

4.4.1 Design Approach and Methods

Oracle APEX is a low- code development platform and follows agile development. In this application, empty database for created using SQL to store the details. Using APEX this application was created from scratch and Calendar component was used. Color coded static values were provided for showing markings in calendar. Page 3 which was used for creation of new entries was redirected to page 1 that was calendar. This made the application easy to use as all the functionalities were moved to page 1. Also, the markings on calendar were linked to modification of that particular entry for future changes. List view of work log can also be seen along with calendar view. Feedback feature was added too. Agile methodologies were followed in development of the application.

4.4.2 Workflow

4.4.2.1 Modules

a) Database: Using SQL, database was created from scratch containing details such as start date, end date, status for calendar.

b) Application Interface Designing: Using Oracle APEX, interface for the app was designed. Calendar interface was introduced for application.

c) Application Development: APEX was used to develop application containing all the features from scratch. Creation of new and modification of existing logs were added and redirected to the same page where log can be viewed in calendar or list format.

d) Testing, Correction and Modification: Application was tested, corrected and further modified using APEX.

4.4.2.2 Diagrams

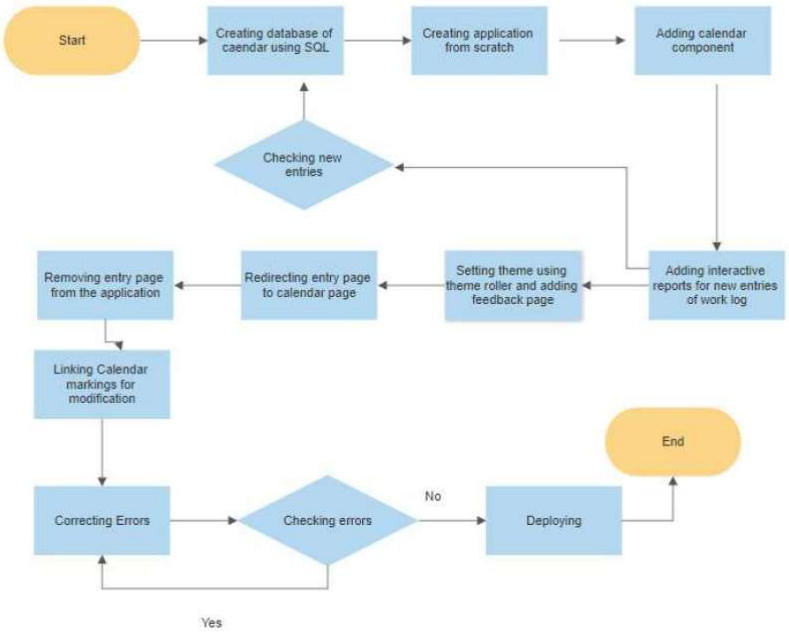


Figure 10: Workflow Diagram of Work Log Calendar

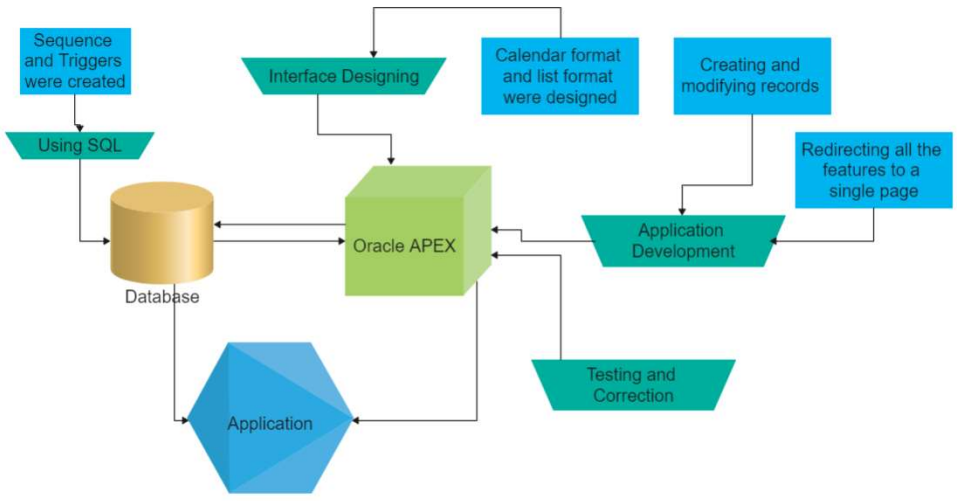


Figure 11: System Diagram of Work Log Calendar

4.4.3 Codes

4.4.3.1 SQL Code for table creation

```
CREATE TABLE  "CALENDAR"
  (  "ID" NUMBER(5,0) NOT NULL ENABLE,
     "START_DATE" DATE,
     "END_DATE" DATE,
     "TITLE" VARCHAR2(100),
     "STATUS" VARCHAR2(100),
      CONSTRAINT "CALENDAR_PK" PRIMARY KEY ("ID")

   )
/
```

4.4.3.2 Trigger

```
CREATE OR REPLACE EDITIONABLE TRIGGER "BI_CALENDAR"
 before insert on "CALENDAR"
 for each row
begin
 if :NEW."ID" is null then
   select "CALENDAR_SEQ".nextval into :NEW."ID" from sys.dual;
 end if;
end;

/
ALTER TRIGGER  "BI_CALENDAR" ENABLE
```

4.4.3.3 SQL sequence and trigger

```
CREATE table "CALENDAR" (
"ID"        NUMBER(5,0) NOT NULL,
"START_DATE" DATE,
"END_DATE"   DATE,
"TITLE"      VARCHAR2(100),
"STATUS"     VARCHAR2(100),
constraint  "CALENDAR_PK" primary key ("ID")
)
/
CREATE sequence "CALENDAR_SEQ"
/
CREATE trigger "BI_CALENDAR"
before insert on "CALENDAR"
for each row
begin
if :NEW."ID" is null then
select "CALENDAR_SEQ".nextval into :NEW."ID" from sys.dual;
end if;
end;
/
```

4.4.4 Constraints, Alternatives and Tradeoffs

Traditional method of web application can still be used which requires deep understanding of scripting and designing languages such as HTML, CSS, Bootstrap. It also requires deep understanding of AngularJS and NodeJS. This method takes more time to develop applications and is not easy to modify. However, Oracle APEX follows agile methodologies and is more user friendly. Low-code environment is easier and faster. Quick scalable and efficient applications can be developed using APEX.

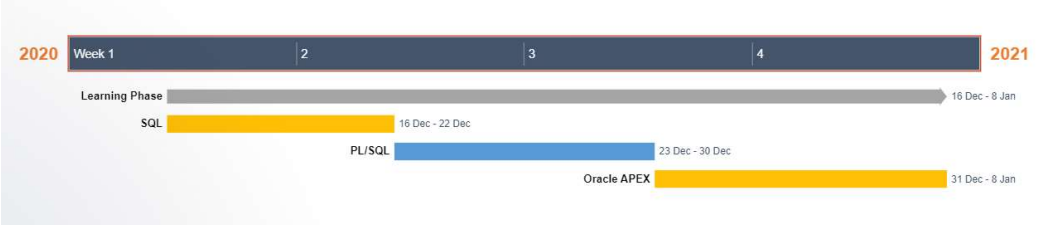
5. SCHEDULE, TASKS AND MILESTONES

During the period of internship all my projects were time-bound, and I completed all the projects as per the schedule. Completion of each project was a milestone and, in each project, there were tasks which I successfully completed. The Gantt charts and table of the same are:

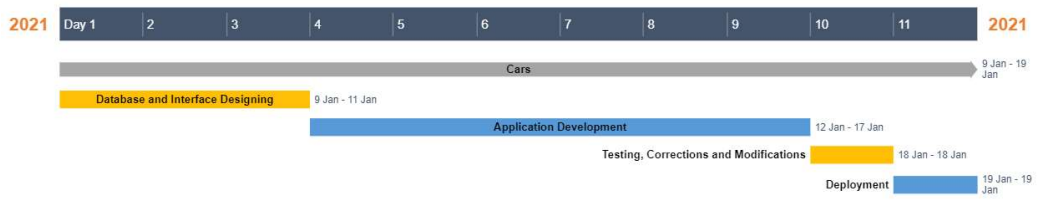
5.1 GANTT CHART

5.1.1 Gantt Chart for complete period

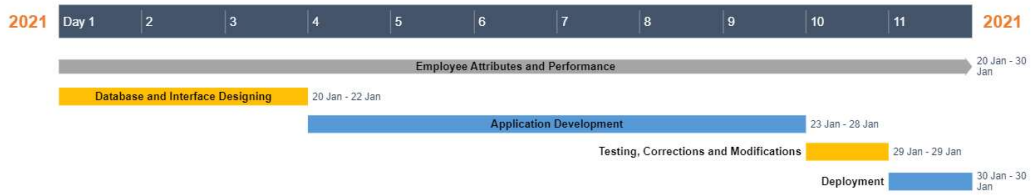
5.1.2 Gantt Chart for Learning Phase



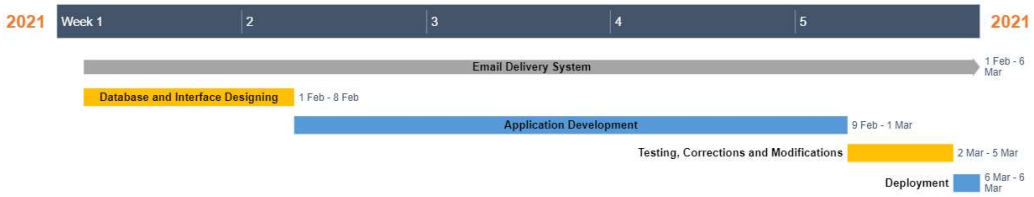
5.1.3 Gantt Chart for Cars



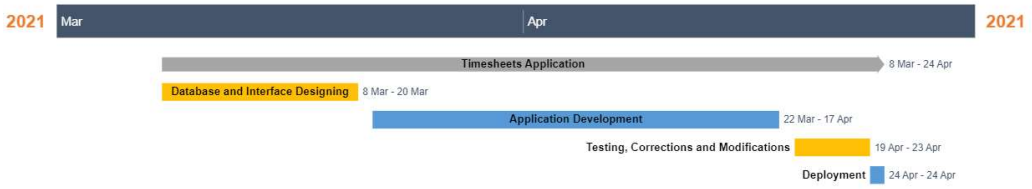
5.1.4 Gantt Chart for Employee Attributes and Performance



5.1.5 Gantt Chart for Email Delivery System



5.1.6 Gantt Chart for Timesheets Application



5.1.7 Gantt Chart for Work Log Calendar

5.2 SCHEDULE, TASKS AND MILESTONES TABLE

Table 1: Schedule, Tasks and Milestones

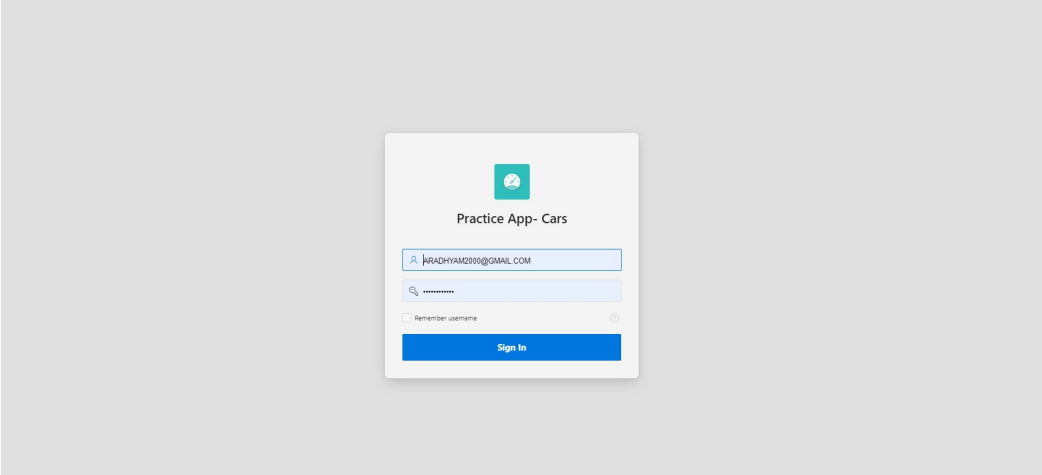
Sr. No	Milestones	Tasks	Start Date	End Date
1.	Start of Internship		15-12-2020	
2.	Learning Phase		16-12-2020	08-01-2021
		SQL	16-12-2020	22-12-2020
		PL/SQL	23-12-2020	30-12-2020
		Oracle APEX	31-12-2020	08-01-2021
3.	Cars		09-01-2021	19-01-2021
		Database and Interface Designing	09-01-2021	11-01-2021
		Application Development	12-01-2021	17-01-2021
		Testing, Corrections and Modifications	18-01-2021	18-01-2021
		Deployment	19-01-2021	19-01-2021
4.	Employee Attributes and Performance		20-01-2021	30-01-2021
		Database and Interface Designing	20-01-2021	22-01-2021
		Application Development	23-01-2021	28-01-2021
		Testing, Corrections and Modifications	29-01-2021	30-01-2021
		Deployment	30-01-2021	30-01-2021
5.	Email Delivery System		01-02-2021	06-03-2021
		Database and Interface Designing	01-02-2021	08-02-2021

		Application Development	09-02-2021	01-03-2021
		Testing, Corrections and Modifications	02-03-2021	05-03-2021
		Deployment	06-03-2021	06-03-2021
6.	Timesheets Application		09-03-2021	19-04-2021
		Database and Interface Designing	08-03-2021	20-03-2021
		Application Development	22-03-2021	17-04-2021
		Testing, Corrections and Modifications	19-04-2021	23-04-2021
		Deployment	24-04-2021	24-04-2021
7.	Work Log Calendar		26-04-2021	14-05-2021
		Database and Interface Designing	26-04-2021	29-04-2021
		Application Development	30-04-2021	10-05-2021
		Testing, Corrections and Modifications	11-05-2021	13-05-2021
		Deployment	14-05-2021	14-05-2021
8.	End of Internship			15-05-2021

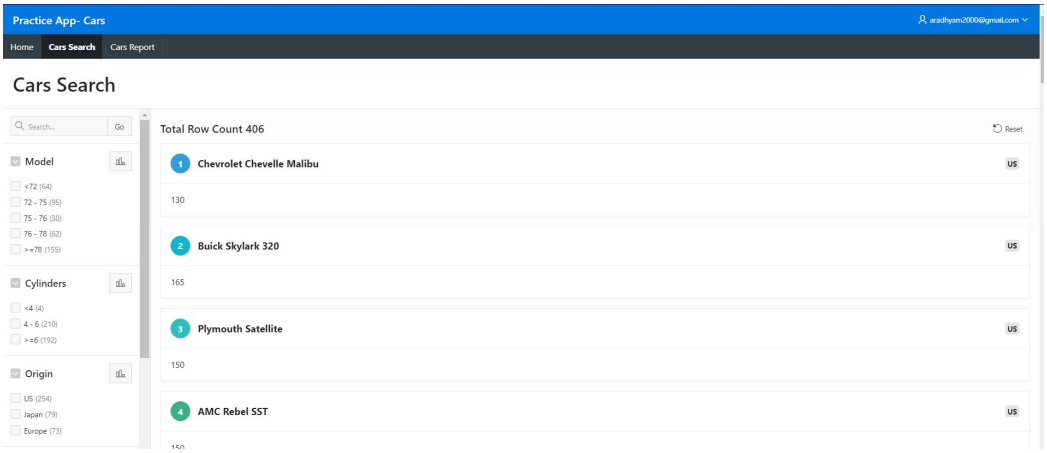
6. PROJECT DEMONSTRATION

6.1 SAMPLE APPLICATIONS

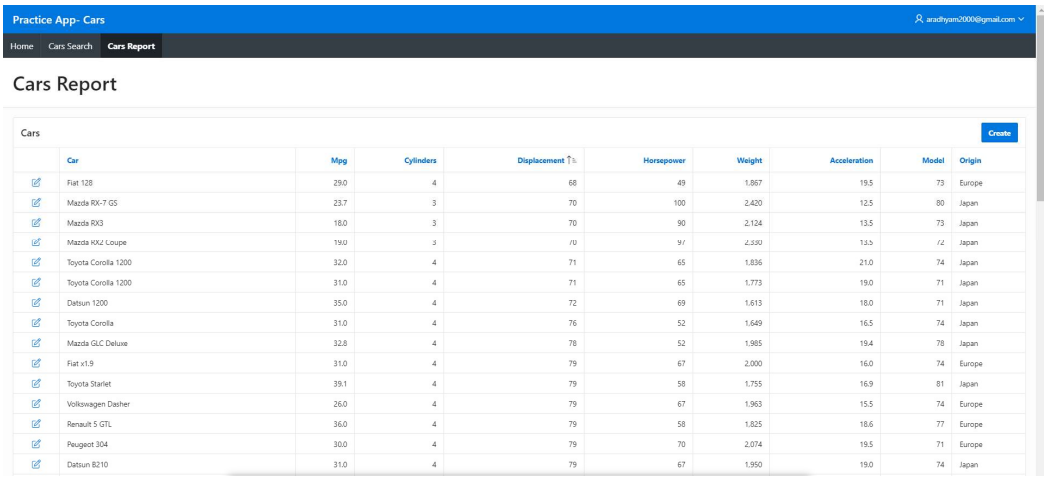
6.1.1 Cars Application



Login

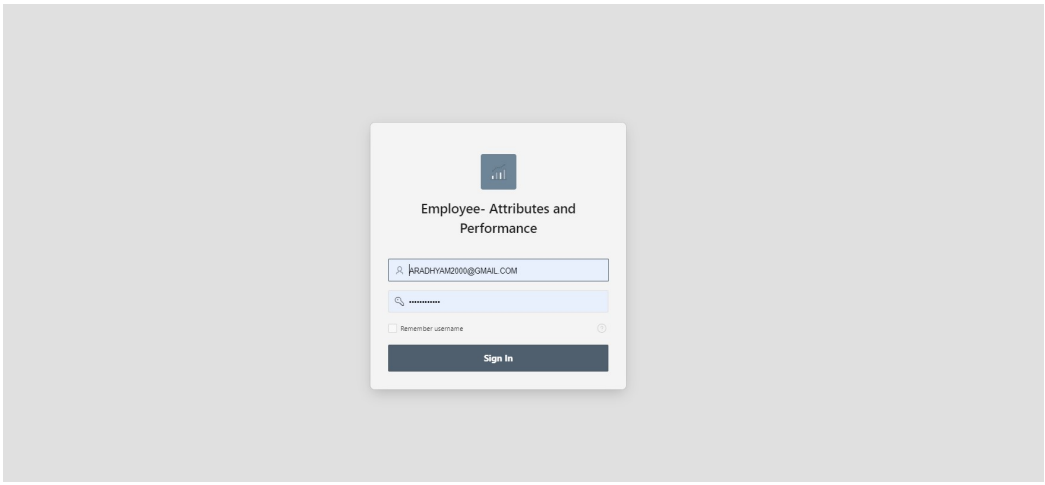


Cards format for searching

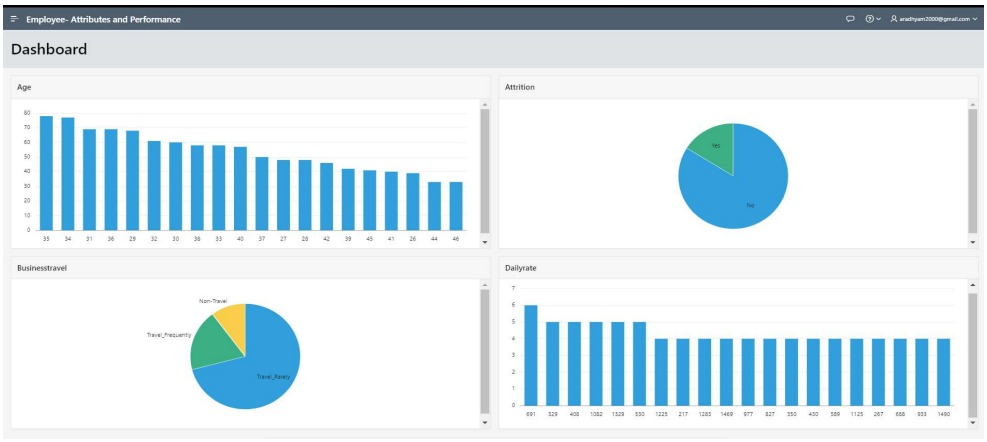


Report can be seen and new entries can be made

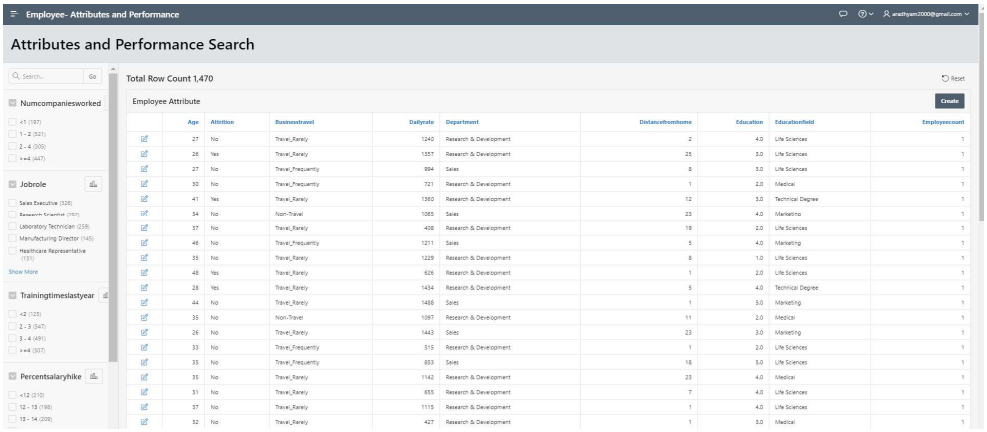
6.1.2 Employee Attributes and Performance Application



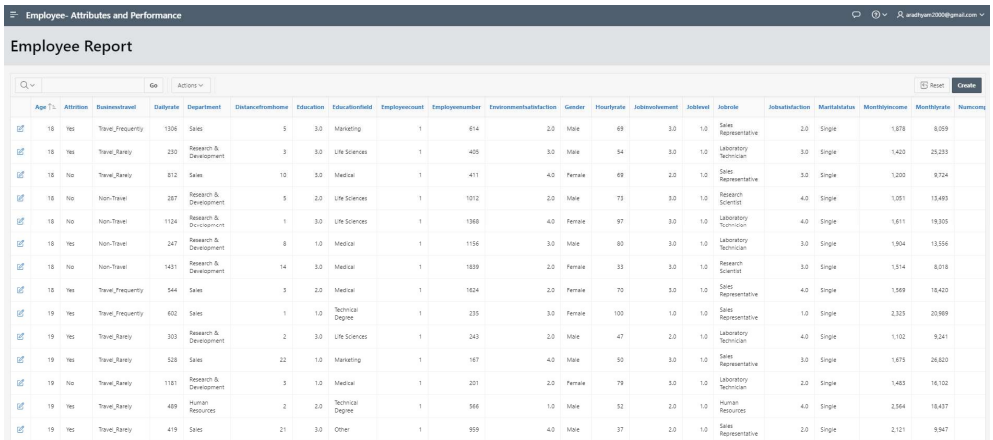
Login



Visualization

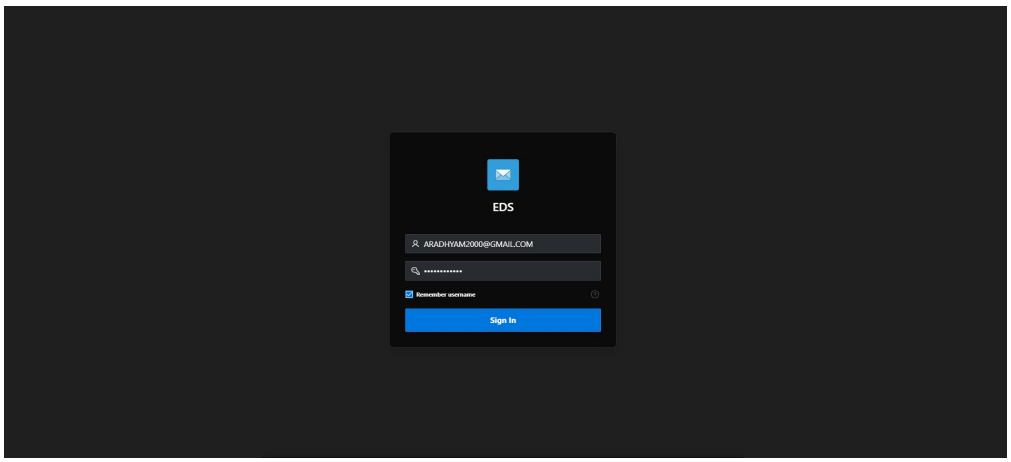


Form format for searching

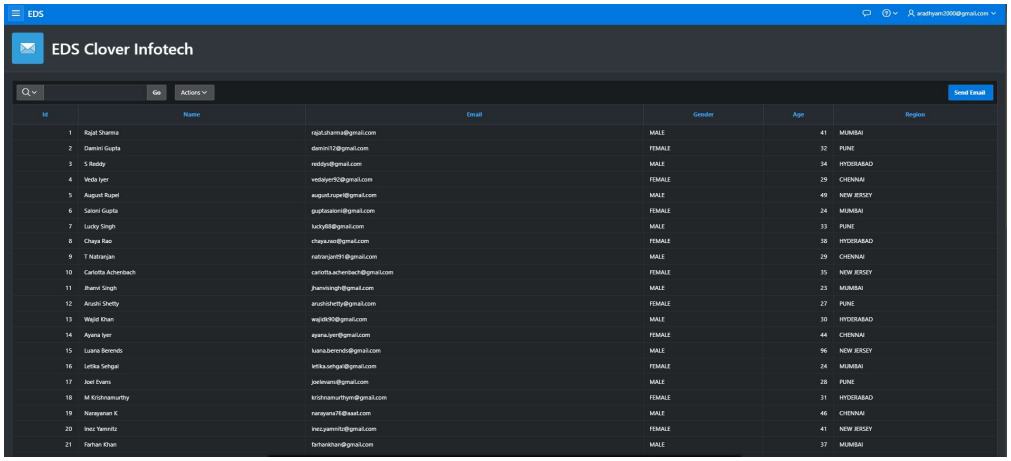


Option to create entry and reset report

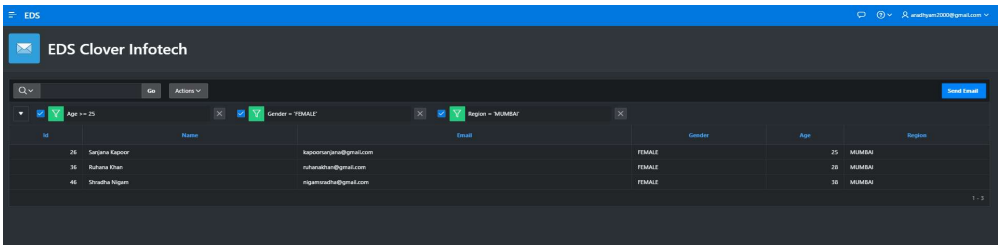
6.2 EMAIL DELIVERY SYSTEM



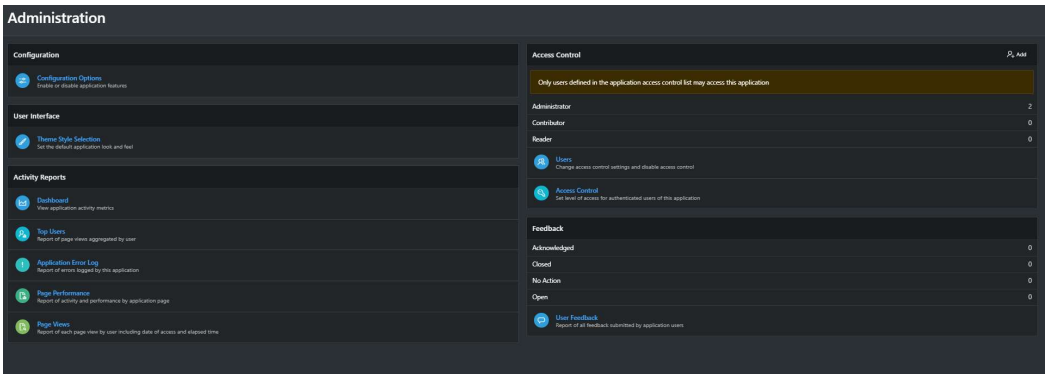
Login Page



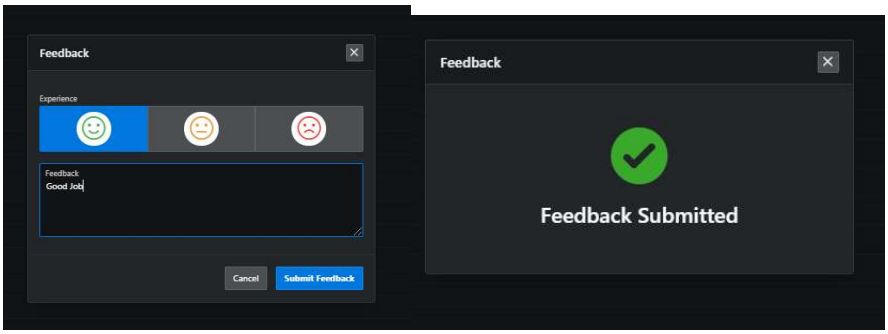
Customer Information



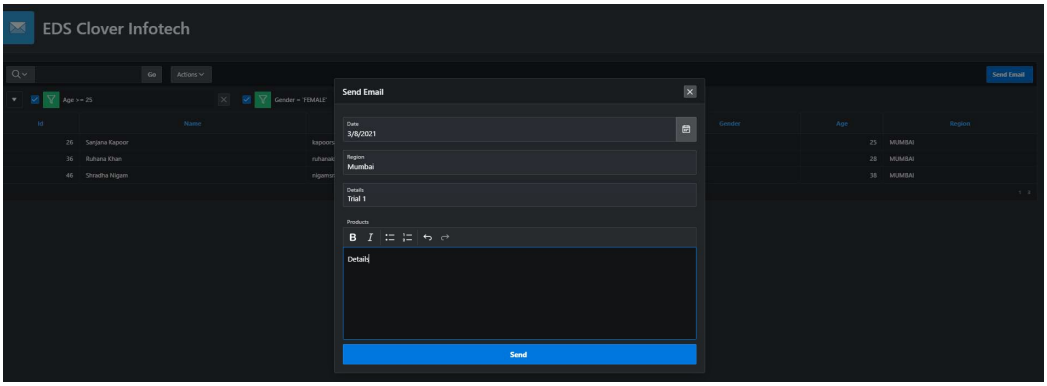
Filter to find required customers



Administration page

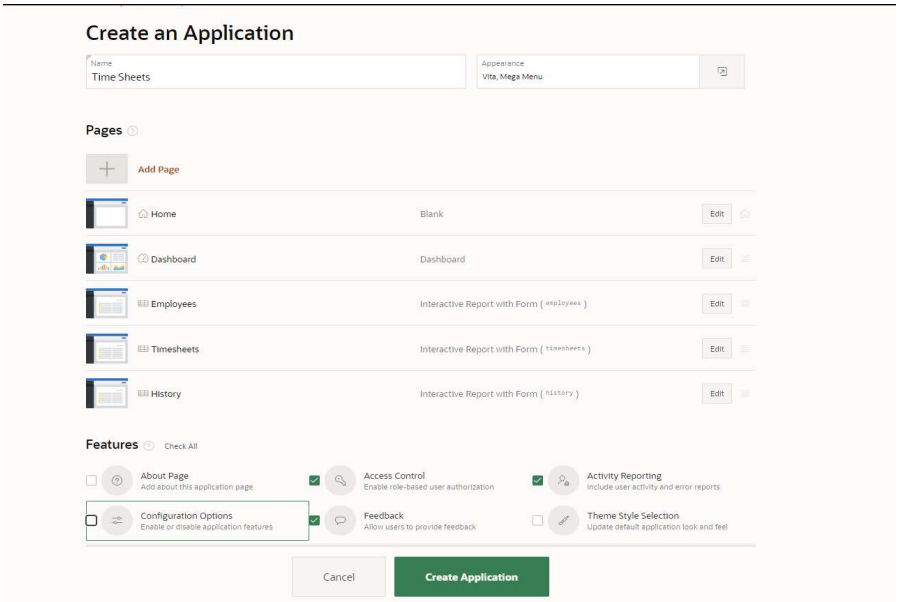


Feedback Page

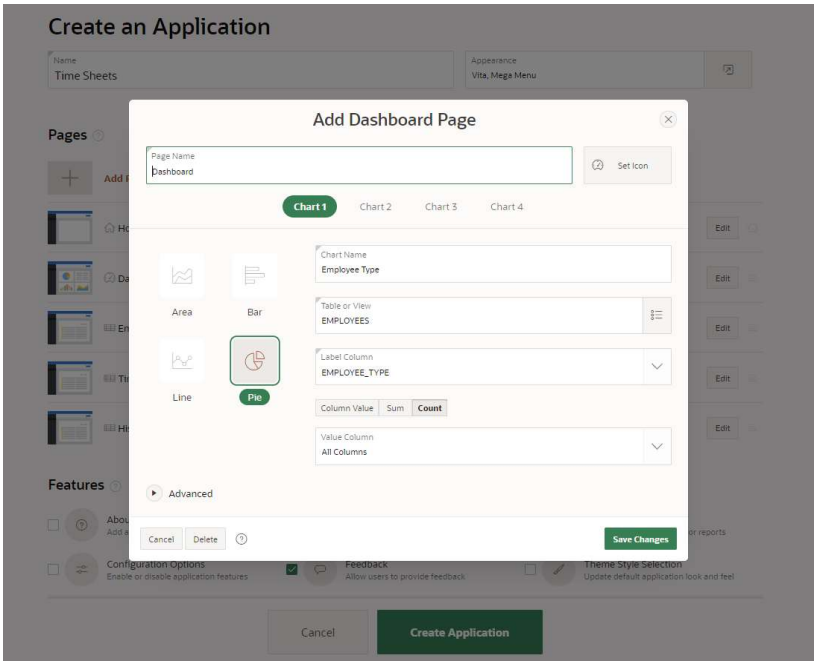


Send Email Option

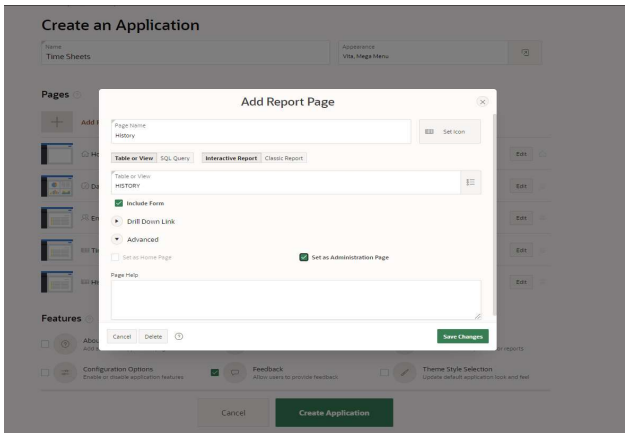
6.3 TIMESHEETS APPLICATION



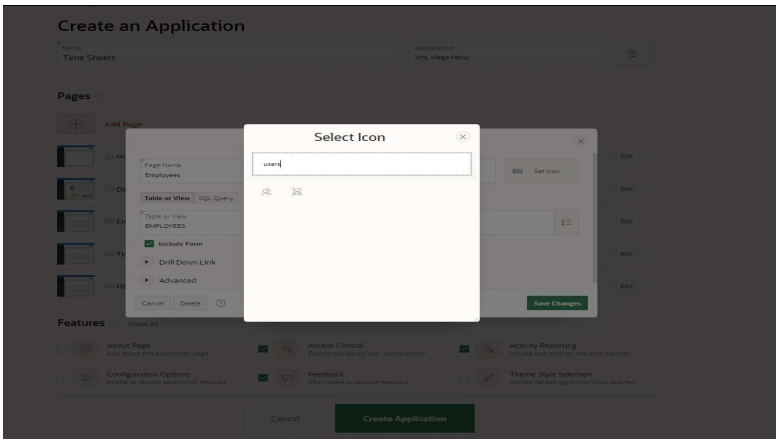
Creating Application



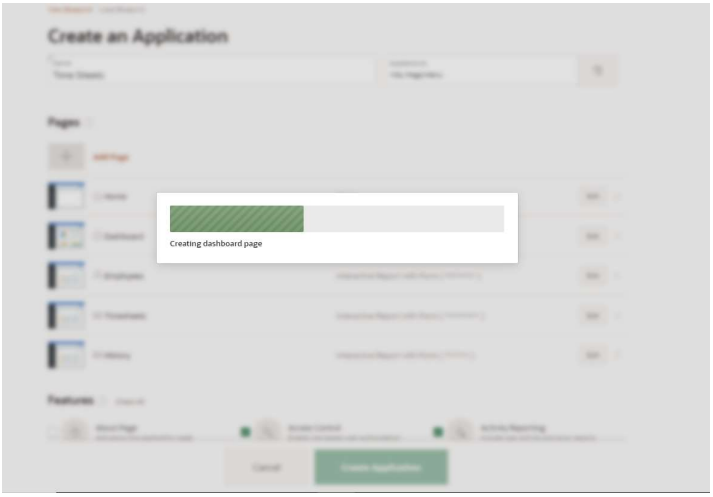
Adding Dashboard Page



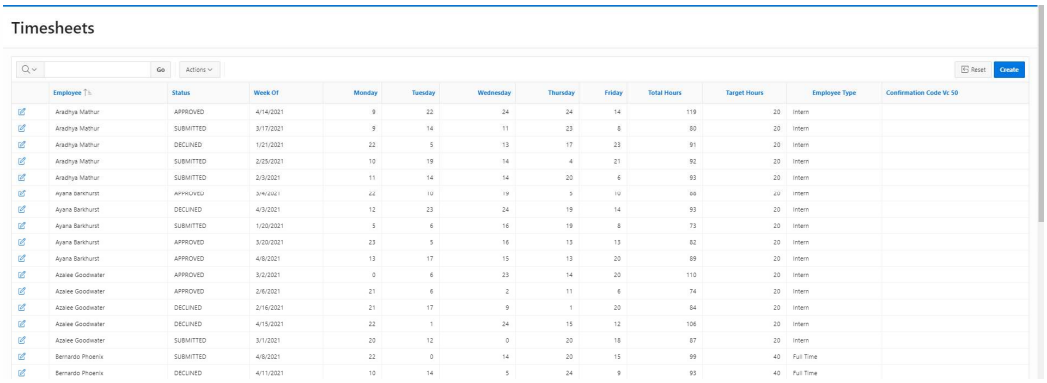
History Page is set as Administration page



Setting user icon for employee page



Loading Application

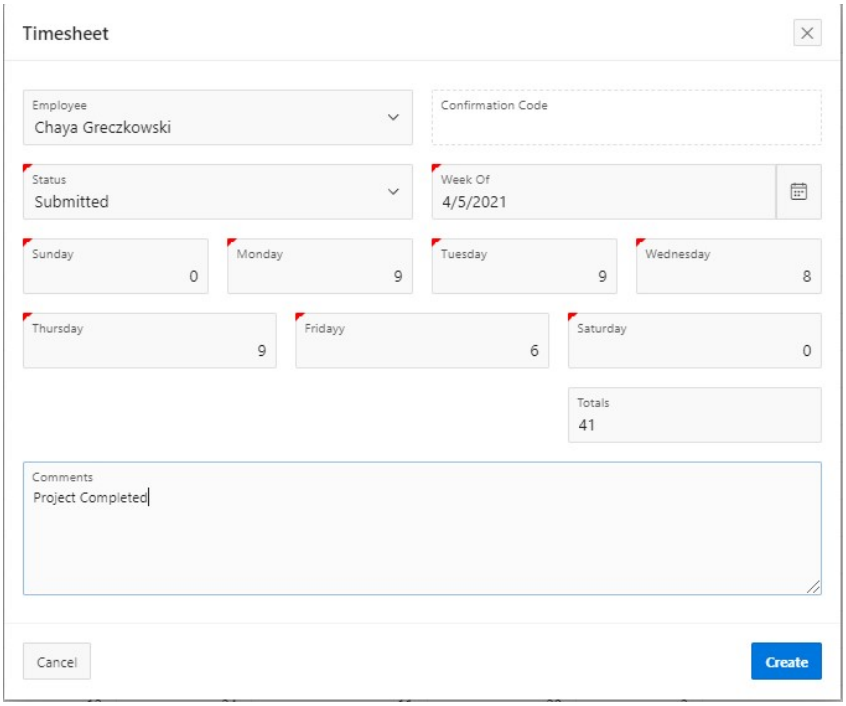


Target hours added

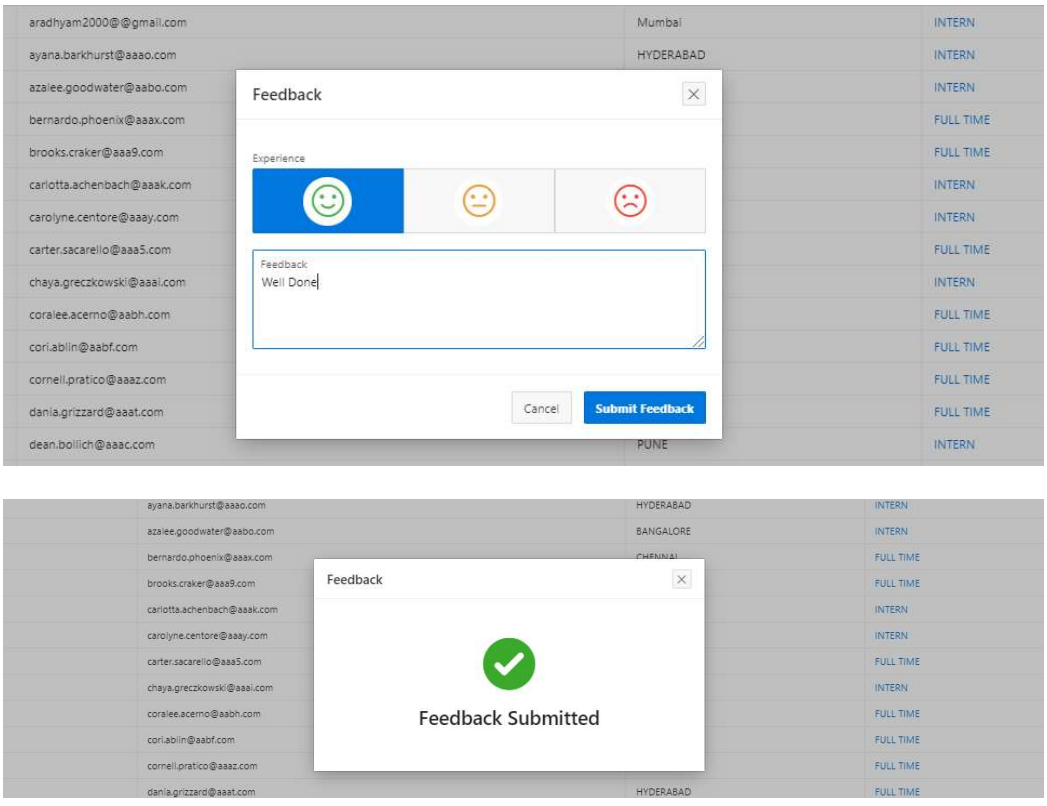
Monday	Tuesday	Wednesday	Thursday	Friday	Total Hours	Target Hours	Employee Type
9	22	24	24	14	119	20	Intern
9	14	11	23	8	80	20	Intern
22	5	13	17	23	91	20	Intern
10	19	14	4	21	92	20	Intern
11	14	14	20	6	93	20	Intern
22	10	19	5	10	88	20	Intern
12	23	24	19	14	93	20	Intern
5	6	16	19	8	73	20	Intern

Daily working hours inserted

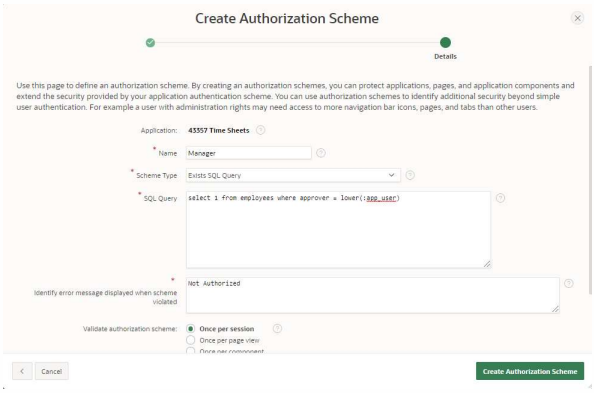
Setting Default value 0 for Saturday and Sunday



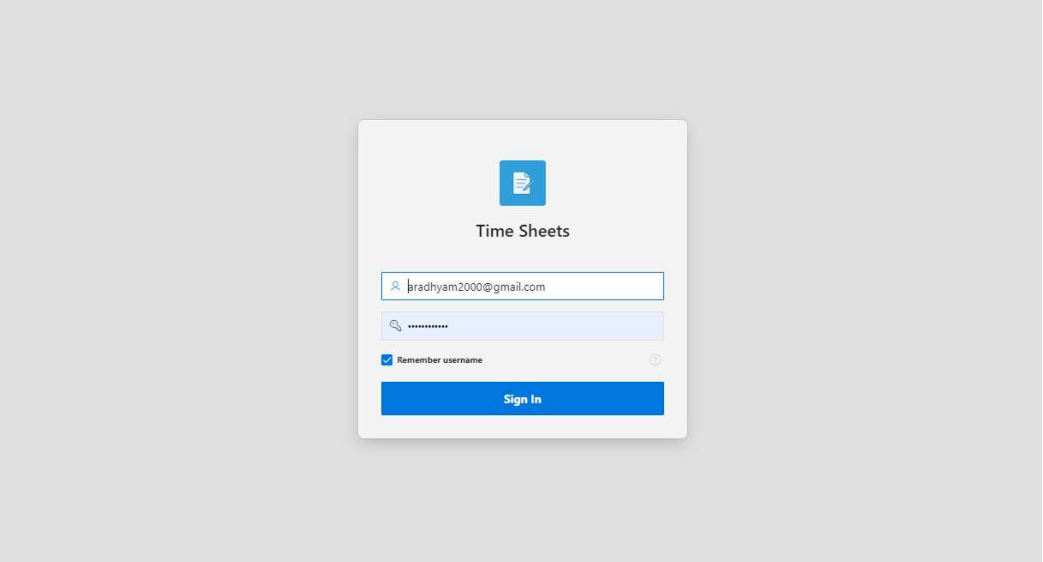
Timesheet



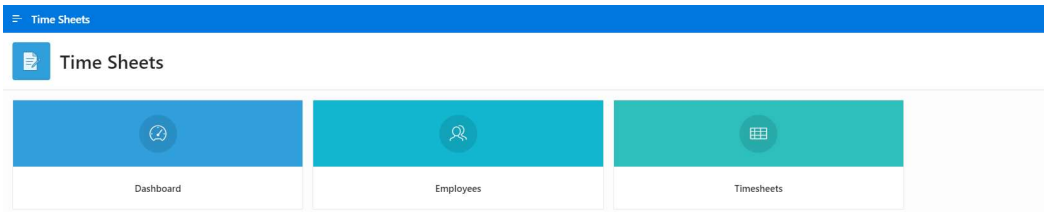
Feedback Page



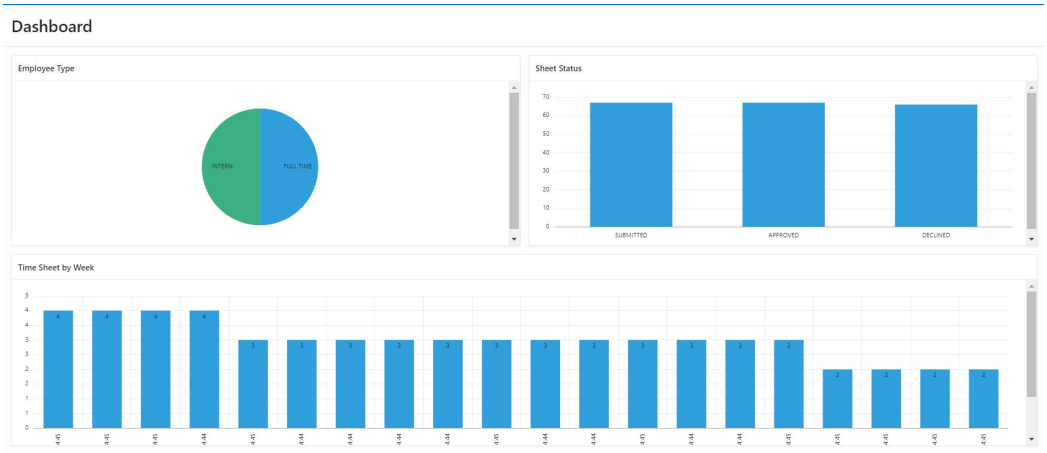
Creating Authorization for Manager



Login Page

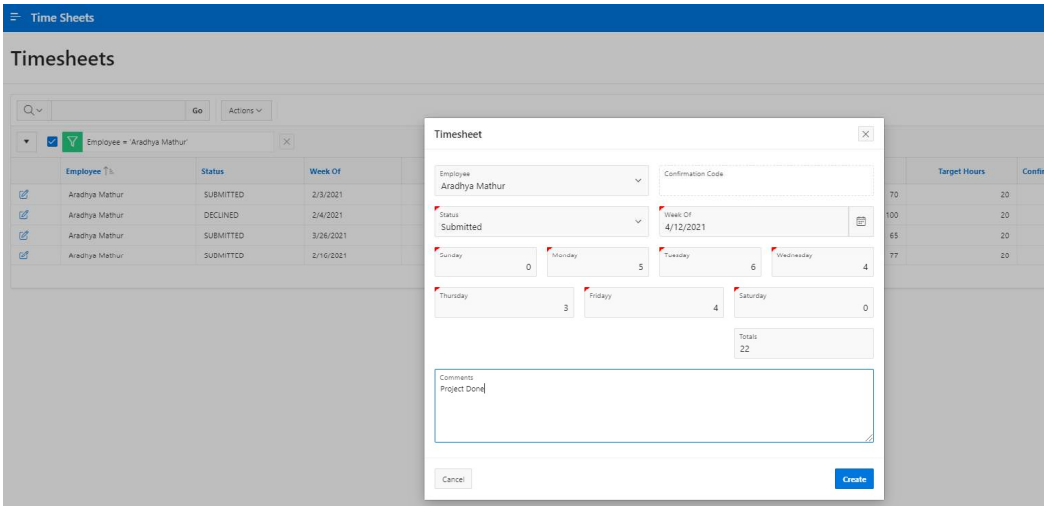


Homepage



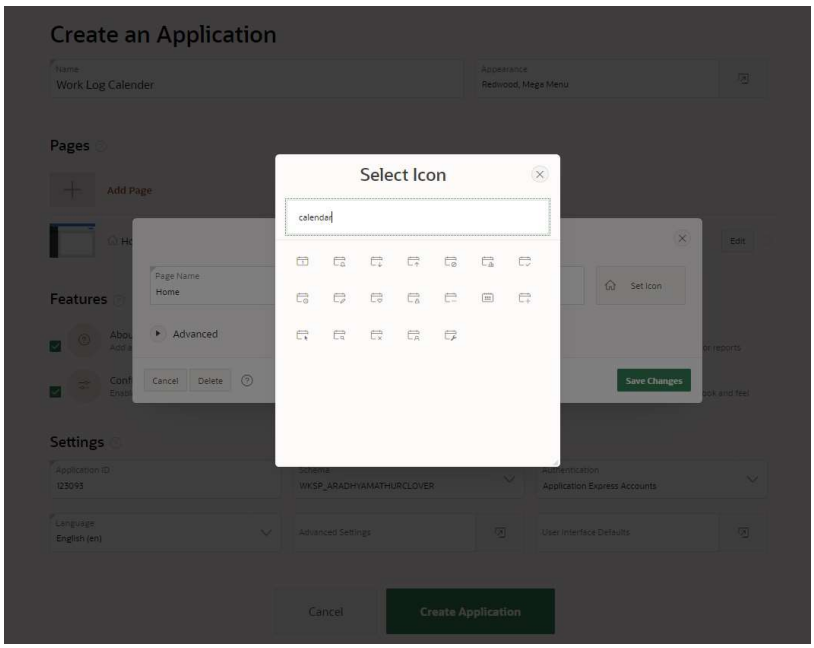
Dashboard

Employee Page

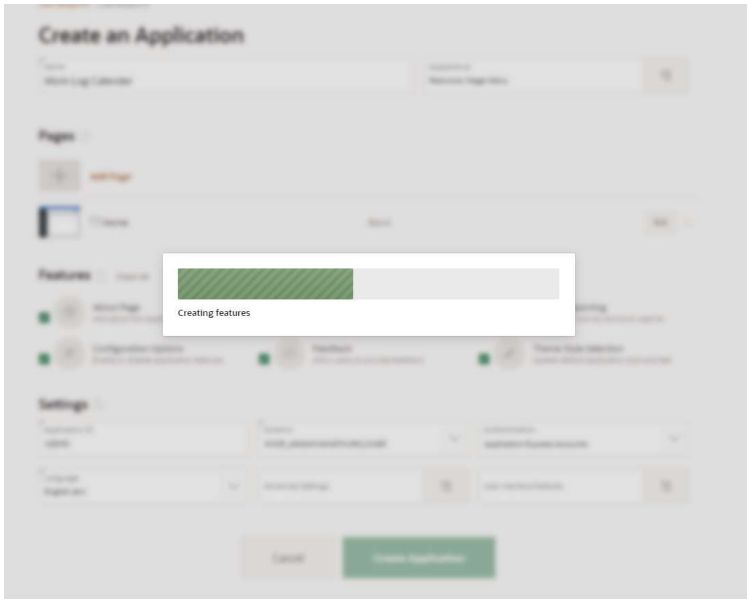


Filling Time Sheet

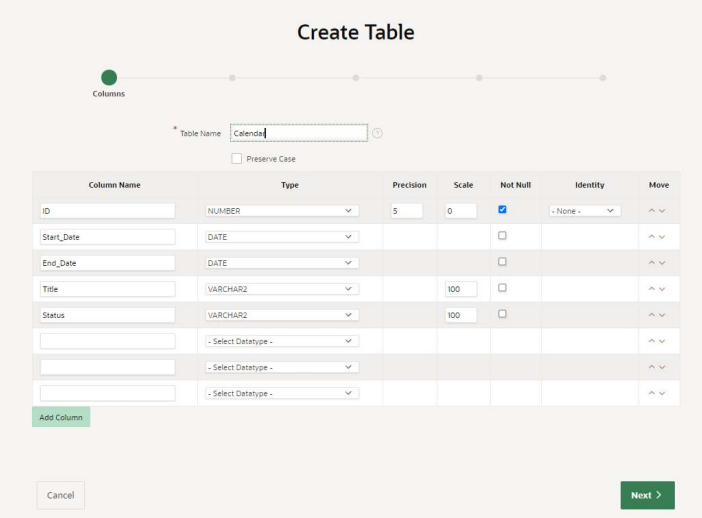
6.4 WORK LOG CALENDAR



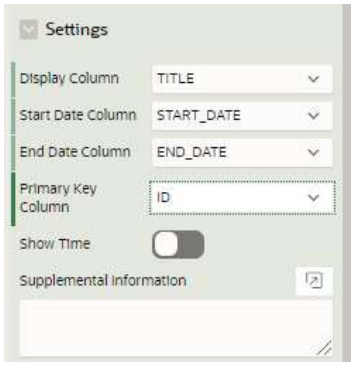
Setting Icon



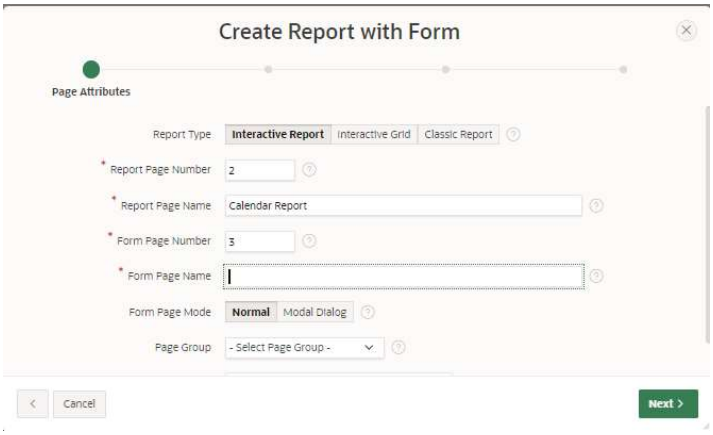
Creating Application



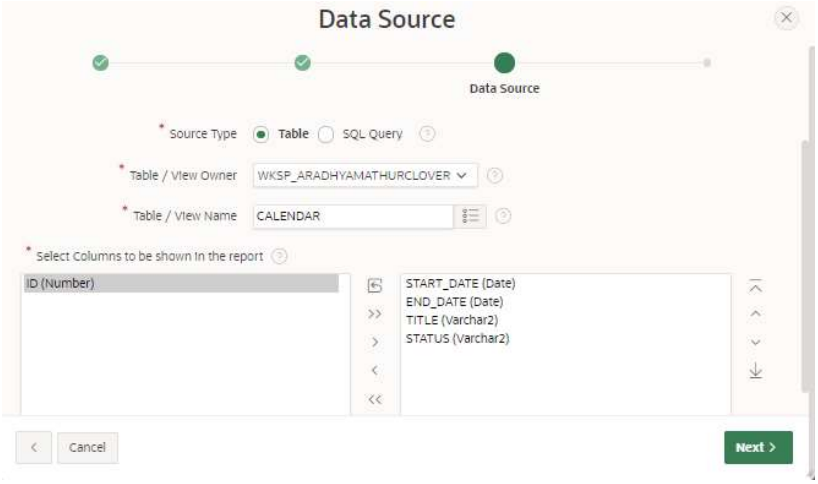
Creating Table



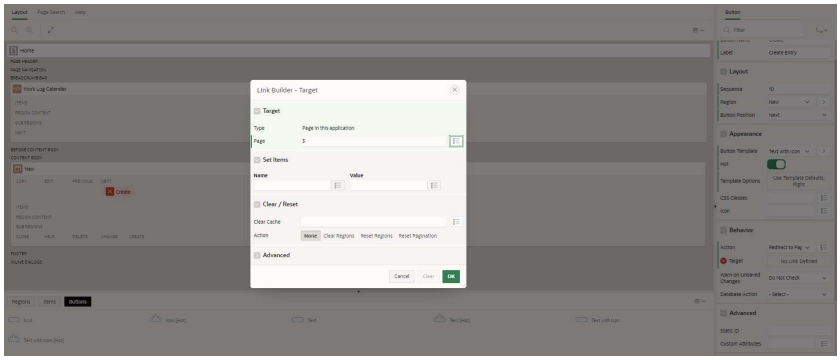
Setting attributes



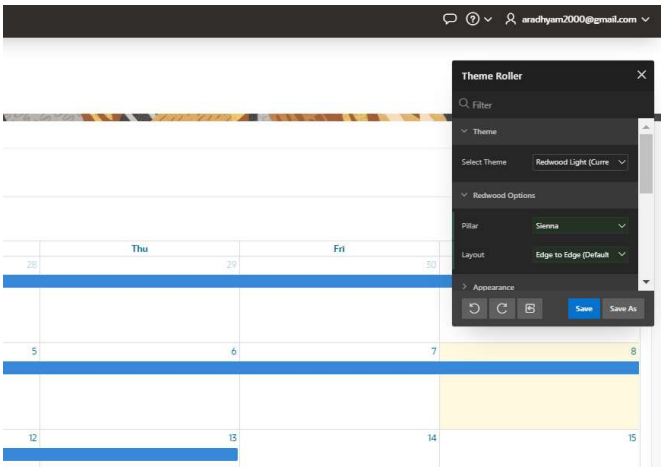
Creating interacting report



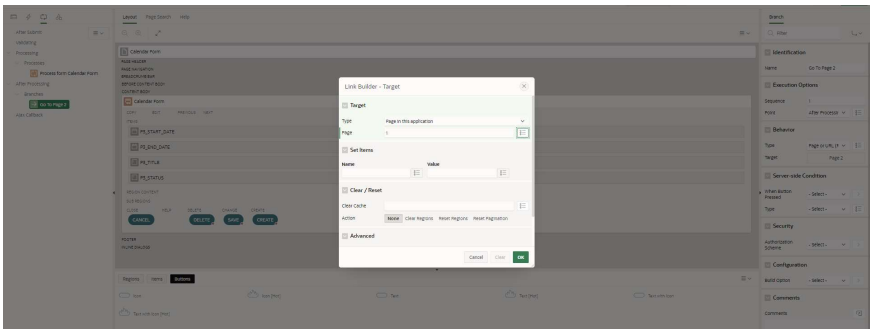
Data source for redirection



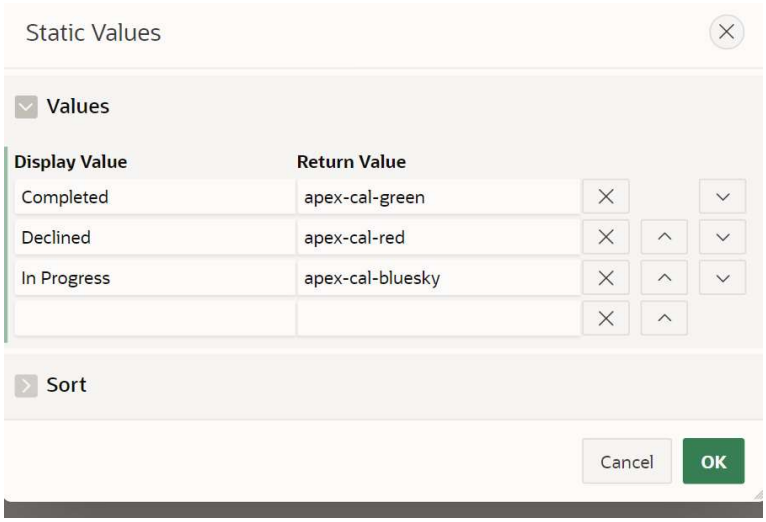
Redirecting entry page



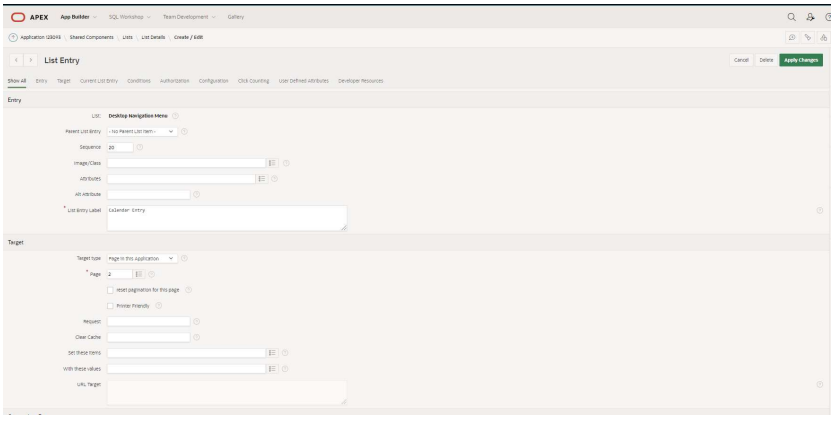
Theme roller



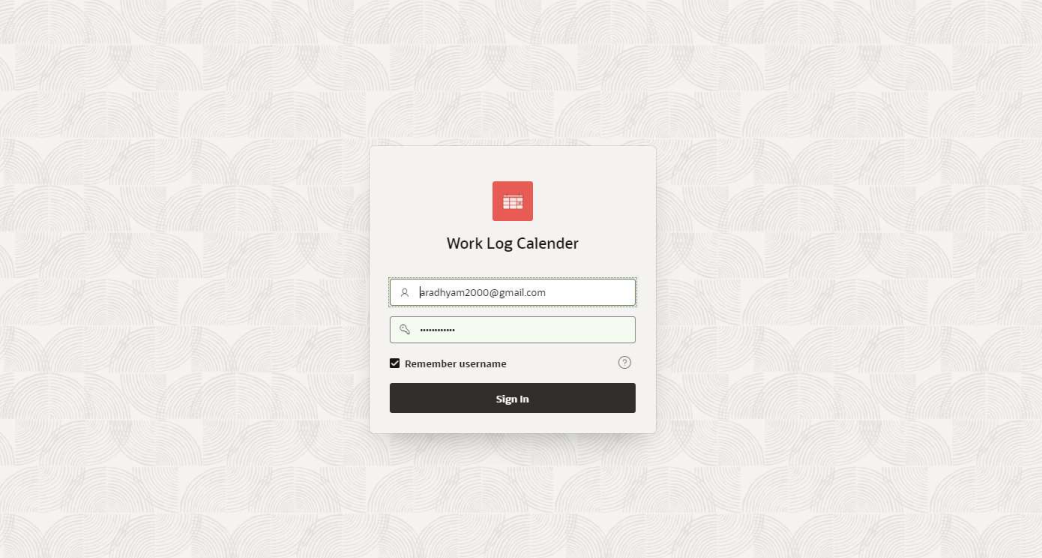
Redirecting to calendar page



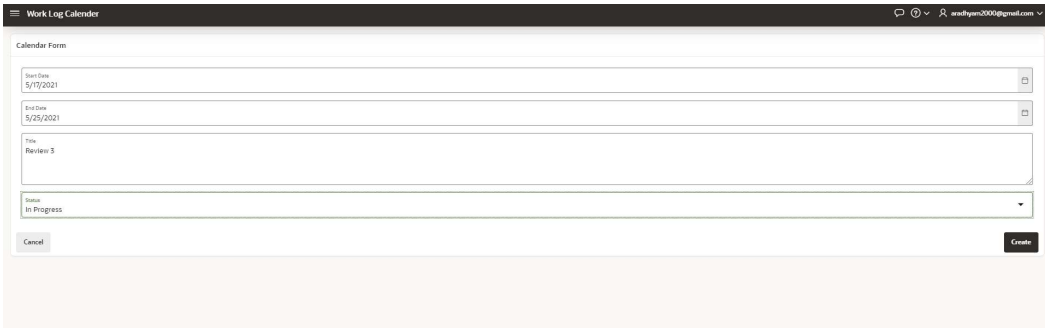
Changing static values for calendar markings



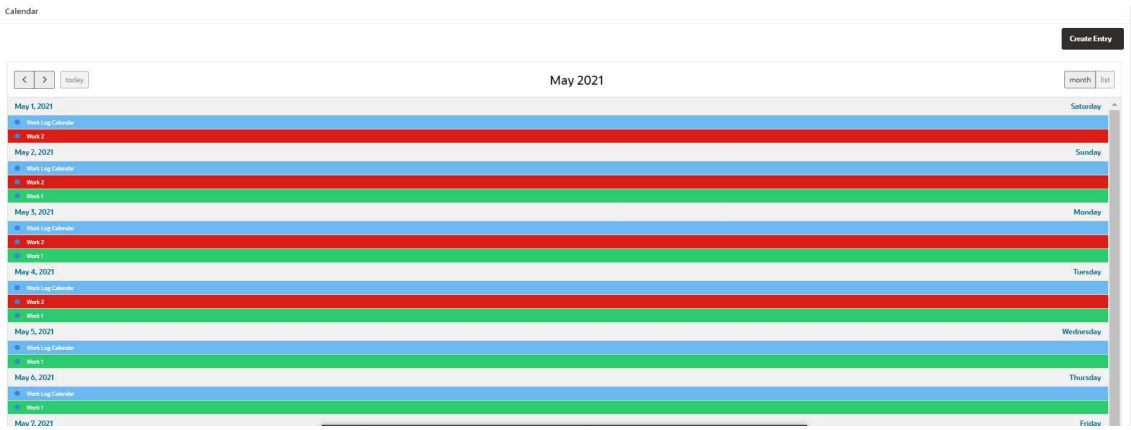
Removing calendar entry page



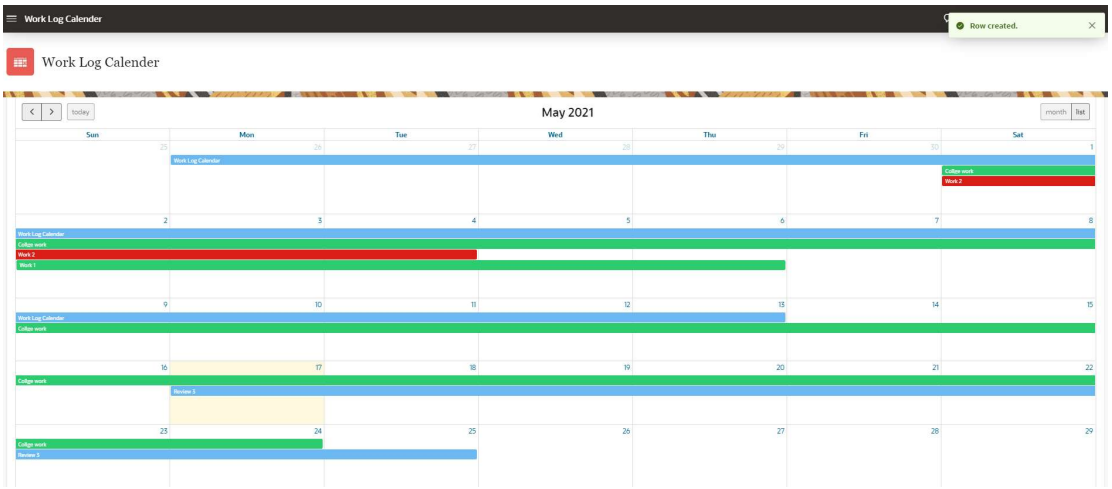
Login Page



Creating Work Log



List Format



Calendar Format

7. RESULT AND DISCUSSIONS

5 months of internship was a great opportunity to learn new things which has resulted in vast improvement of my technical knowledge and skill set. All my projects were completed and submitted to my external guide on time successfully. In each project I learned new things and methods to use SQL, PL/SQL and Oracle APEX for development of an application.

I will be forever grateful to VIT University for giving me a perfect opportunity to gain work experience and learn new things at very reputed organization Clover Infotech.

Considering the assigned projects, I had, it was interesting to implement something else that I was taught in my four years of bachelor program. It was interesting to me in different points of views. It was a real pleasure to work in corporate culture. Moreover, working under a team leader made me learn different things.

The learnings of this period such as time management, regularity, communication skills, analytical abilities will favor me in my future carrier a lot.

8. SUMMARY

My internship started on 15th December, 2020 at Clover Infotech which is a leading multinational company. During my time I was introduced and trained in theory and lab work of SQL followed by PL/SQL and Oracle APEX. Together these are used for low-code development. Learning these things improved my technical skill sets immensely. I developed two sample applications, first one was taught by the clover academy instructor which was Cars. I learned to use SQL for creating database for the application which was further developed using Oracle APEX. After that, I was given a task for which I was handed an excel file containing data related to employees, their performance and their attributes. I used SQL for creation of database and APEX for application development. In this application I used the charting tools for creation of bar graphs and pie charts. Searching, sorting, creation and modification of data were basic functionalities of the application. My first project assigned was Email Delivery System which I started on 1st February, 2021. In this application too, SQL was used to create database for the application. Using the APEX, interface of the application was designed. PL/SQL was used for dynamic actions such as sending email. My second project, Timesheets application started on 8th March, 2021 had several functionalities. SQL was used for creation of database and connecting it with Oracle APEX. Interface had 3 pages, dashboard page showed bar graphs and pie charts which were created using chart tools. In the second page, details of the employees were stored and features of adding new employees or modification of details of them were added. Using the third page new timesheets were to be created and also the old timesheets could be seen. After creation of timesheet, functionality to send email to the manager was created using PL/SQL. PL/SQL also was used for all the validations and dynamic actions in the application. Another functionality was to give manager facility to either approve or reject timesheet. My third and final project started on 26th April, 2021. Work Log Calendar was an application through which daily work and project can be logged on a calendar. Database was created using SQL and using APEX application was created. This application had features to create new logs

and modifying previous logs. All the features were redirected to a single page making the application easy to use. The work log can be viewed in a list format or a calendar format.

Besides all the technical work I also learnt how an IT company works. I also made few new contacts in the IT industry and learnt to work in a team of professionals who were highly skilled.

Time management was one of the most important things I learned. Since, I performed the projects for clients, managing time and settings targets were inevitably important. I understood that without setting targets and going in an open-ended fashion, I could never really produce the desired results. My communication skills also got better. Overall, it was a delightful learning experience for me.

9. REFERENCES

[1] Baggia, A., Leskovar, R., Rodič, B.(2019, January). Low code programming with oracle apex offers new opportunities in higher education. Third International Scientific Conference ITEMA Recent Advances in Information Technology, Tourism, Economics, Management and Agriculture. (pp. 91-97).

[2] Zaharieva, Z., Billen, R. (2009). Rapid development of database interfaces with oracle apex, used for the controls systems at CERN. Proceedings of ICALEPCS2009, Kobe, Japan THP108. (pp. 883-885).

[3] Srinivas, S., Biswas, A., Srinivasan, J. (2010, February). An application synopsis tool for database applications developed using oracle application express. Proceeding of the 3rd Annual India Software Engineering Conference (ISEC). (pp. 113-118).

[4] Monger, A., Baron, S., Lu, J. (2009, July) More on Oracle APEX for teaching and learning. 7th International Workshop on Teaching, learning and Assessment of Databases 2009. (pp. 409-419).

[5] Crespo, E. (2020, December) Vulnerability analysis with SQLMAP applied to APEX 5 environments. The Universidad Politécnica Salesiana of Ecuador. Creative commons. (pp. 104-113.)

[6] Godfrey, J. (2020). Oracle Application Express Tutorial: Building an Application, California. (pp. 7-89).

[7] McDonald, C., Katz, C., Beck, C., Kalman, J., Knox, D. (2004). Mastering Oracle PL/SQL: Practical Solutions, California. (pp. 307-349).

[8] Jennings, T. (2020). Oracle Application Express App Builder User's Guide, California. (pp. 2-26).

[9] Gault, D., Peake, D. (2020). Oracle Application Express SQL Workshop Guide, California. (pp. 2-43).

[10] Jennings, T. (2020). Oracle Application Express API Reference, California. (pp. 29-38).

Appendix A

Oracle Application Express

Oracle Application Express which is a low-code development platform was released by Oracle Corporation. APEX is used to develop scalable, feasible, user-friendly and highly secure applications, with elite features like interactive report, interactive grid, faceted search, and charting engine, etc., which can be deployed anywhere. This appendix contains all the features, details and properties of Oracle Apex can be found in the given link https://docs.oracle.com/en/database/oracle/application-express/20.1/htmrn/index.html#HTMRN-GUID-540B73CB-08A7-4422-B6BF-CC785EC47694

Appendix B

Oracle PL/SQL

PL/SQL is a block structured language and a procedural language designed precisely to support SQL statements within its syntax and enhance the capabilities of SQL. This appendix contains complete guide for learning and implementing Oracle PL/SQL and can be found in the given link https://www.oracle.com/in/database/technologies/appdev/plsql.html

Appendix C

Oracle SQL

Structured Query Language is a database language, and is used to perform operations on the records stored in the database such as creation, deletion, fetching, and modification, etc. This appendix contains guide for accessing, defining and maintaining data using Oracle SQL and can be found in the link https://www.oracle.com/in/database/technologies/appdev/sql.html