Placement Tracker System

1. Introduction

1.1 Objectives

The objective of designing this database was to create a desktop and mobile application for our client that would automate the following task:

Desktop Application:

- Create new Students into the system.
- Create new Company into the system.
- Create new Company Sites into the system.
- Create new Job Vacancy into the system.
- To Browse or search current and past job vacancies
- To View report on status of given student and his or her application(s).
- To provide management reports to show:
- Inactive Students (the name, programme and email address of student without an approved CV)
- Active Students (the name, programme and email address of student with an approved CV and number of applications made)
- Placement Events (the company, site, job title, closing date and number of applications made to date for vacancies with a closing date within the next seven days).
- Students who are placed (The name, programme, email address, company, site, job title and start date of students who have accepted an offer).
- Students who are not placed (the name, programme, email address of active students who have not accepted an offer).
- To display a management dashboard with:
    - A pie chart showing the proportion of placed to unplaced active students.
    - A stacked bar chart showing the number of active students placed and unplaced by programme. ▪ A calendar of job vacancy closing dates.

Student's Mobile web application:

- To view, browse and search current job vacancies
- To allow student to record applications made and maintain his or her application(s) history.
- To allow a student to view a report on the status of his or her application(s).

1.2 Background Information

Our team was given a task to create a desktop application where the University Administrator is able to maintain all data that has been stored in the system such as Student details, Companies Details, Companies Site Details and Job Vacancies. It was also our task to create a mobile app in which Students are able to maintain and check their application status, application history and to search for job vacancies. The development environment we used was Oracle Application Express (also known as APEX) as well as using SQL Developer, which was mainly used for adding some of the triggers with SQL as well as generating our data model. We were given a period of 6 weeks to develop the applications.

1.3 Method of Approach

Firstly, we sat down as a group to establish our intentions, trying to identify each member of the group's strength and weaknesses and draw up the most appropriate action plan in regards to each member's capabilities. Then working together to draw up an ERD map for each member to follow as they proceeded to work on their own individualised assignments.

We endeavoured to split into two sub-groups within our group where one focused on the Desktop Administrator app, and the other focused on the Mobile app for students so as to cover as much ground as possible, and to work as efficiently as we were able to. That way we could all develop experiences in different aspects of development and later work together as a more experienced team when it came to compiling the final builds.

2. Analysis and Design

2.1 Tables, Triggers, and Sequences

1. CREATE TABLE "APEX606"."STUDENTS"

```
(       "STUDENTID" NUMBER,
        "STUDENT_NAME" VARCHAR2(100 BYTE) NOT NULL ENABLE,
        "STUDENT_DOB" DATE NOT NULL ENABLE,
        "TERM_HOME_ADDRESS" VARCHAR2(100 BYTE) NOT NULL ENABLE,
        "TERM_HOME_POSTCODE" VARCHAR2(100 BYTE) NOT NULL ENABLE,
        "HOME_ADDRESS" VARCHAR2(100 BYTE),
        "HOME_ADDRESS_POSTCODE" VARCHAR2(100 BYTE),
        "STUDENT_MOBILE_NO" NUMBER(11,0) NOT NULL ENABLE,
        "STUDENT_EMAIL_ADDRESS" VARCHAR2(255 BYTE) NOT NULL ENABLE,
        "PLACEMENT_NOTE" VARCHAR2(100 BYTE),
        "DATE_OF_CV_SUBMIT" DATE,
        "DATE_OF_CV_APPROVE" DATE,
        "PROGRAMME_ID" NUMBER NOT NULL ENABLE,
        "PASSWORD" VARCHAR2(10 BYTE),
        "USERNAME" VARCHAR2(50 BYTE),
        "STUDENT_LINE_NO" NUMBER(11,0),
         CONSTRAINT "STUDENTS_PK" PRIMARY KEY ("STUDENTID")
              CONSTRAINT "STUDENT_EMAIL_ADDRESS_CHK" CHECK (REGEXP_LIKE(STUDENT_EMAIL_ADDRESS, '[a-zA-Z0-9._%-]+@[a-zA-Z0-
9._%-]+\.[a-zA-Z]{2,4}')) ENABLE,
         CONSTRAINT "USERNAME_NN" CHECK ( "USERNAME" IS NOT NULL) ENABLE,
CONSTRAINT "STUDENTS_CON" UNIQUE ("USERNAME")
 CONSTRAINT "PROGRAMME_ID_FK" FOREIGN KEY ("PROGRAMME_ID")    REFERENCES
"APEX606"."PROGRAMME" ("PROGRAMME_ID") ENABLE

CREATE SEQUENCE   "STUDENTS_SEQ"  MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 621 CACHE
20 NOORDER  NOCYCLE  NOPARTITION
/
```

```
CREATE OR REPLACE EDITIONABLE TRIGGER "APEX606"."BI_STUDENTS"

  before insert or update of STUDENTID, USERNAME on "STUDENTS"
for each row
begin
  if :NEW."STUDENTID" is null then
    select "STUDENTS_SEQ".nextval into :NEW."STUDENTID" from sys.dual;
end if;

  IF INSERTING OR UPDATING THEN
     INSERT INTO USER_ACCOUNT(USERNAME, PASSWORD, STUDENT_NAME, STUDENT_EMAIL_ADDRESS)
      VALUES (:NEW.USERNAME, :NEW.PASSWORD, :NEW.STUDENT_NAME, :NEW.STUDENT_EMAIL_ADDRESS);



  If (:NEW.STUDENT_DOB+NUMTOYMINTERVAL(17,'YEAR') >SYSDATE)
then
  RAISE_APPLICATION_ERROR(-20000,'Student must be at least 17 years old to apply!');
end if;



  end if;


end;
/
ALTER TRIGGER "APEX606"."BI_STUDENTS" ENABLE;
```

2. CREATE TABLE "APEX606"."PROGRAMME"

```
(       "PROGRAMME_ID" NUMBER NOT NULL ENABLE,
        "PROGRAMME_NAME" VARCHAR2(100 BYTE) CONSTRAINT "PROGRAMME_NAME_NN" NOT NULL ENABLE,
        "TYPE_OF_DEGREE" VARCHAR2(10 BYTE) CONSTRAINT "TYPE_DEGREE_NN" NOT NULL ENABLE,
        "MANDATORY_PLACEMENT" VARCHAR2(10 BYTE) CONSTRAINT "MANDATORY_PLACEMENT_NN" NOT NULL ENABLE,
         CONSTRAINT "PROGRAMME_ID_UNQ" UNIQUE ("PROGRAMME_ID")
```

CREATE SEQUENCE "PROGRAMMES_SEQ" MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 61 CACHE 20 NOORDER NOCYCLE NOPARTITION
/

CREATE OR REPLACE EDITIONABLE TRIGGER "APEX606"."BI_PROGRAMME"

```
before insert on "PROGRAMME"                for each row   begin
  if :NEW."PROGRAMME_ID" is null then
    select "PROGRAMME_SEQ1".nextval into :NEW."PROGRAMME_ID" from sys.dual;
end if;  end;

/
ALTER TRIGGER "APEX606"."BI_PROGRAMME" ENABLE;
```

3. CREATE TABLE "APEX606"."JOBS"

```
(       "JOB_ID" NUMBER NOT NULL ENABLE,
        "JOB_TITLE" VARCHAR2(150 BYTE) NOT NULL ENABLE,
        "JOB_DESCRIPTION" VARCHAR2(500 BYTE) NOT NULL ENABLE,
        "SITE_ID" NUMBER NOT NULL ENABLE,
        "JOB_EMAIL_ADDRESS" VARCHAR2(150 BYTE),
        "CLOSING_DATE_APPLICATION" DATE NOT NULL ENABLE,
```

```
"JOB_VACANCY" VARCHAR2(10 BYTE) NOT NULL ENABLE,
"PLACEMENT_START_DATE" DATE NOT NULL ENABLE,
"PLACEMENT_END_DATE" DATE NOT NULL ENABLE,
"SALARY" NUMBER NOT NULL ENABLE,
"Application_Method" VARCHAR2(150 BYTE) NOT NULL ENABLE,
"CONTACT_ENQUIRY" NUMBER(11,0) NOT NULL ENABLE,
CONSTRAINT "JOBS_PK" PRIMARY KEY ("JOB_ID"),
 CONSTRAINT "JOB_ADDRESS_CHK" CHECK (REGEXP_LIKE(JOB_EMAIL_ADDRESS, '[a-zA-Z0-9._%-]+@[a-zA-Z0-9._%-]+\.[a-zA-Z]{2,4}'))
ENABLE,
 CONSTRAINT "JOBS_SITE_ID" FOREIGN KEY ("SITE_ID")
  REFERENCES "APEX606"."COMPANY_SITES" ("SITE_ID") ENABLE
CREATE SEQUENCE   "JOBS_SEQ"  MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 21 CACHE 20
NOORDER  NOCYCLE  NOPARTITION
/

 CREATE OR REPLACE EDITIONABLE TRIGGER "APEX606"."BI_JOBS"
before insert on "JOBS"
 for each row   begin
 if :NEW."JOB_ID" is null then
   select "JOBS_SEQ4".nextval into :NEW."JOB_ID" from sys.dual;
end if;
end;

/
ALTER TRIGGER "APEX606"."BI_JOBS" ENABLE;
```

4. **CREATE TABLE "APEX606"."COMPANY_SITES"**

```
  (       "SITE_ID" NUMBER NOT NULL ENABLE,
          "SITE_NAME" VARCHAR2(150 BYTE) NOT NULL ENABLE,
          "COMPANY_NAME" VARCHAR2(150 BYTE) NOT NULL ENABLE,
          "SITE_EMAIL_ADDRESS" VARCHAR2(100 BYTE) NOT NULL ENABLE,
          "SITE_TEL_NO" VARCHAR2(50 BYTE) NOT NULL ENABLE,
```

```
"SITE_POSTCODE" VARCHAR2(25 BYTE) NOT NULL ENABLE,
 CONSTRAINT "COMPANY_SITES_PK" PRIMARY KEY ("SITE_ID")
,
 CONSTRAINT "SITE_EMAIL_ADDRESS_CHK" CHECK (REGEXP_LIKE(SITE_EMAIL_ADDRESS, '[a-zA-Z0-9._%-]+@[a-zA-Z0-9._%-]+\.[a-zA-Z]{2,4}')) ENABLE,
 CONSTRAINT "COMPANY_NAME-FK" FOREIGN KEY ("COMPANY_NAME")
REFERENCES "APEX606"."COMPANY" ("COMPANY_NAME") ENABLE

CREATE SEQUENCE  "COMPANY_SITES_SEQ"  MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 1
CACHE 20 NOORDER  NOCYCLE  NOPARTITION
/
 CREATE OR REPLACE EDITIONABLE TRIGGER "APEX606"."BI_COMPANY_SITES"
before insert on "COMPANY_SITES"          for each row   begin
  if :NEW."SITE_ID" is null then
    select "COMPANY_SITES_NEW_SEQ".nextval into :NEW."SITE_ID" from sys.dual;
end if;
end;

/
ALTER TRIGGER "APEX606"."BI_COMPANY_SITES" ENABLE;
```

5. **CREATE TABLE "APEX606"."COMPANY"**

```
(       "COMPANY_NAME" VARCHAR2(150 BYTE) NOT NULL ENABLE,
        "COMPANY_EMAIL_ADDRESS" VARCHAR2(100 BYTE) NOT NULL ENABLE,
        "COMPANY_TEL_NO" VARCHAR2(50 BYTE) NOT NULL ENABLE,
        "COMPANY_POSTCODE" VARCHAR2(10 BYTE) NOT NULL ENABLE,
          CONSTRAINT "COMPANY_EMAIL_ADDRESS_CHK" CHECK (REGEXP_LIKE(COMPANY_EMAIL_ADDRESS, '[a-zA-Z0-9._%-]+@[a-zA-Z0-
```

```
9._%-]+\.[a-zA-Z]{2,4}')) ENABLE,
        CONSTRAINT "COMPANY_NAME_UQ" UNIQUE ("COMPANY_NAME")

CREATE SEQUENCE  "COMPANIES_SEQ"  MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 21 CACHE
20 NOORDER  NOCYCLE  NOPARTITION
/
 CREATE OR REPLACE EDITIONABLE TRIGGER "APEX606"."BI_COMPANY"
before insert on "COMPANY"          for each row   begin
 if :NEW."COMPANY_NAME" is null then
   select "COMPANY_SEQ".nextval into :NEW."COMPANY_NAME" from sys.dual;
end if;
end;

/
ALTER TRIGGER "APEX606"."BI_COMPANY" ENABLE;
```

6. **CREATE TABLE "APEX606"."APPLICATION_RECEIVED"**

```
  (       "APPLICATION_ID" NUMBER CONSTRAINT "APPLICATION_ID_NN" NOT NULL ENABLE,
          "STUDENT_ID" NUMBER CONSTRAINT "STUDENT_ID_NN" NOT NULL ENABLE,
          "JOB_ID" NUMBER CONSTRAINT "JOB_ID_NN" NOT NULL ENABLE,
          "APPLICATION_STATUS" VARCHAR2(100 BYTE),
          "APPLICATION_ACCEPTED_DATE" DATE,
          "APPLICATION_DATE" DATE,
           PRIMARY KEY ("APPLICATION_ID")

CREATE SEQUENCE  "SEQ_APPLICATION_RECEIVED"  MINVALUE 1 MAXVALUE 99999 INCREMENT BY 1 START WITH 49 NOCACHE  ORDER
NOCYCLE  NOPARTITION
/
 CREATE OR REPLACE EDITIONABLE TRIGGER "APEX606"."TRG_APPLICATION_RECEIVED"
BEFORE INSERT OR UPDATE OF APPLICATION_ID,STUDENT_ID,JOB_ID,APPLICATION_ACCEPTED_DATE ON APPLICATION_RECEIVED
FOR EACH ROW BEGIN
IF INSERTING THEN
```

```
  :NEW.APPLICATION_ID := SEQ_APPLICATION_RECEIVED.nextval;
 END IF;
 IF:NEW.STUDENT_ID != :OLD.STUDENT_ID OR :NEW.JOB_ID != :OLD.JOB_ID THEN

  :NEW.APPLICATION_ACCEPTED_DATE := SYSDATE;

  IF INSERTING or UPDATING THEN
 INSERT INTO APPLICATION_HISTORY_NEW(STUDENT_ID, JOB_ID, APPLICATION_STATUS, APPLICATION_DATE)
VALUES (:OLD.STUDENT_ID, :OLD.JOB_ID, :OLD.APPLICATION_STATUS, :OLD.APPLICATION_DATE);
 END IF;

  --ELSE
  --RAISE_APPLICATION_ERROR(-20000,'Warnig Duplicating Data');



 END IF;

END;
/
ALTER TRIGGER "APEX606"."TRG_APPLICATION_RECEIVED" ENABLE;
```

7. CREATE TABLE "APEX606"."APPLICATION_HISTORY_NEW"

```
(       "APPHISTORY_ID" NUMBER NOT NULL ENABLE,
        "STUDENT_ID" NUMBER,
        "JOB_ID" NUMBER,
        "APPLICATION_STATUS" VARCHAR2(100 BYTE),
        "APPLICATION_METHOD" VARCHAR2(100 BYTE),
        "UPDATE_TIME" DATE,
        "APPLICATION_DATE" DATE,
         CONSTRAINT "SYS_C00147762" PRIMARY KEY ("APPHISTORY_ID")
         CONSTRAINT "APPLICATION_HISTORY_STUDENT" FOREIGN KEY ("STUDENT_ID")
```

```
REFERENCES "APEX606"."STUDENTS" ("STUDENTID") ENABLE,
CONSTRAINT "APPLICATION_HISTORY_JOB" FOREIGN KEY ("JOB_ID")
REFERENCES "APEX606"."JOBS" ("JOB_ID") ENABLE
;
CREATE SEQUENCE   "APPLICATION_HISTORY_NEW_SEQ"  MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1
START
WITH 1 CACHE 20 NOORDER  NOCYCLE  NOPARTITION
/


  CREATE OR REPLACE EDITIONABLE TRIGGER "APEX606"."TRG_APPLICATION_HISTORY_NEW"
BEFORE INSERT OR UPDATE OF APPHISTORY_ID,UPDATE_TIME ON APPLICATION_HISTORY_NEW  FOR
EACH ROW
BEGIN
 IF INSERTING THEN
 :NEW.APPHISTORY_ID := SEQ_APPLICATION_HISTORY.nextval;
end if;

  IF:NEW.STUDENT_ID = :OLD.STUDENT_ID AND
    :NEW.JOB_ID = :OLD.JOB_ID AND
    :NEW.APPLICATION_STATUS = :OLD.APPLICATION_STATUS THEN
      :NEW.UPDATE_TIME := SYSDATE;




    IF INSERTING OR UPDATING THEN
     INSERT INTO APPLICATION_HISTORY_NEW(STUDENT_ID, JOB_ID, APPLICATION_STATUS, APPLICATION_DATE)
VALUES (:NEW.STUDENT_ID, :NEW.JOB_ID, :NEW.APPLICATION_STATUS, :NEW.APPLICATION_DATE);
  END IF;

END IF;
```

```
END;
/
ALTER TRIGGER "APEX606"."TRG_APPLICATION_HISTORY_NEW" ENABLE;
```

8. CREATE TABLE "APEX606"."APPLICATION"

```
(       "STUDENT_ID" NUMBER,
        "JOB_ID" NUMBER,
        "CV_SUBMITTED" VARCHAR2(10 BYTE) CONSTRAINT "CV_SUBMITTED_NN" NOT NULL ENABLE,
        "APPLICATION_STATUS" VARCHAR2(100 BYTE) CONSTRAINT "APPLICATION_STATUS_NN" NOT NULL ENABLE,
        "DATE_OF_ACCEPTED" DATE,
        "APPLICATION_DATE" DATE NOT NULL ENABLE,
         PRIMARY KEY ("STUDENT_ID", "JOB_ID"),
         CONSTRAINT "APPLICATION_STUDENT_ID_FK" FOREIGN KEY ("STUDENT_ID")
          REFERENCES "APEX606"."STUDENTS" ("STUDENTID") ENABLE,
         CONSTRAINT "APPLICATION_JOB_ID_FK" FOREIGN KEY ("JOB_ID")
          REFERENCES "APEX606"."JOBS" ("JOB_ID") ENABLE
```

CREATE SEQUENCE "APPLICATION_SEQ" MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 1 CACHE 20 NOORDER NOCYCLE NOPARTITION

```
/
```

CREATE OR REPLACE EDITIONABLE TRIGGER "APEX606"."TRG_APPLICATION"

```
BEFORE INSERT OR UPDATE OF DATE_OF_ACCEPTED ON APPLICATION
FOR EACH ROW
BEGIN
    IF(:NEW.DATE_OF_ACCEPTED > SYSDATE) THEN
   RAISE_APPLICATION_ERROR(-20000, 'Applicant cannot be accepted before application is submitted');
END IF;

  IF INSERTING OR UPDATING THEN
```

```
    INSERT INTO APPLICATION_HISTORY_NEW(STUDENT_ID, JOB_ID, APPLICATION_STATUS, APPLICATION_DATE)
     VALUES (:NEW.STUDENT_ID, :NEW.JOB_ID, :NEW.APPLICATION_STATUS, :NEW.APPLICATION_DATE);

     IF INSERTING OR UPDATING THEN
    INSERT INTO APPLICATION_RECEIVED(STUDENT_ID, JOB_ID, APPLICATION_STATUS, APPLICATION_DATE)
     VALUES (:NEW.STUDENT_ID, :NEW.JOB_ID, :NEW.APPLICATION_STATUS, :NEW.APPLICATION_DATE);

   IF(:NEW.APPLICATION_STATUS = 'Offer Accepted') THEN
     INSERT INTO APPLICATION_RECEIVED(STUDENT_ID, JOB_ID, APPLICATION_ID)
VALUES (:NEW.STUDENT_ID, :NEW.JOB_ID,1);
     DELETE FROM APPLICATION WHERE (:NEW.APPLICATION_STATUS = 'Offer Accepted');
   END IF;

   DELETE FROM APPLICATION WHERE (:NEW.APPLICATION_STATUS = 'Applicant Withdraw');
   DELETE FROM APPLICATION WHERE (:NEW.APPLICATION_STATUS = 'Applicant Rejected');

   END IF;
   END IF;

END;
/
ALTER TRIGGER "APEX606"."TRG_APPLICATION" ENABLE;
```

3. Development

3.1 Assumptions

We used a variety of Tables, Triggers and Sequences to construct our Placement Tracker System. Our core tables consist of the Students table, Company Table, Company Site Table, Jobs Table, Applications Table, Applications History Table. We also made assumptions when we developed our database; those assumptions are Application Received where this table was constructed to enable us to update all columns in the Application table, see figure 3 for illustration.

Our core tables have different functionality; Student table which stores a list of all the students in University, enabling the administrator to easily maintain the record of students in the system; Company table which stores a list of all the companies where the Company name is a unique variable, enabling the administrator to easily maintain the record of Company in the system; Company Site table which stores a list of all the Company sites where Company Sites has a relationship with Company, see figure 3.2 for illustration; Job Table which stores a list of all the jobs that available, it is also associated with Company Site table; Application table which to be used to create an application for all students who wanted to take a placement; Application History table which stores all the applications that have been made by students.

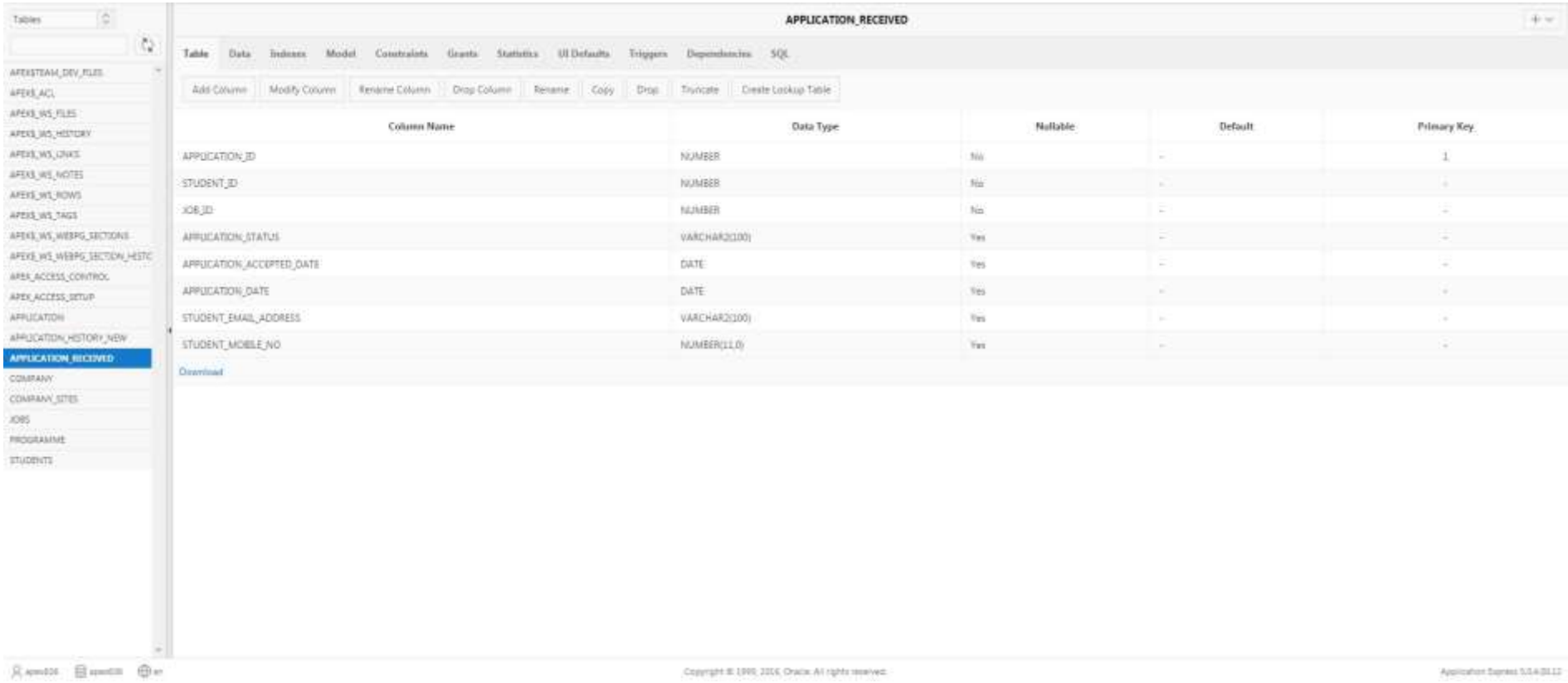
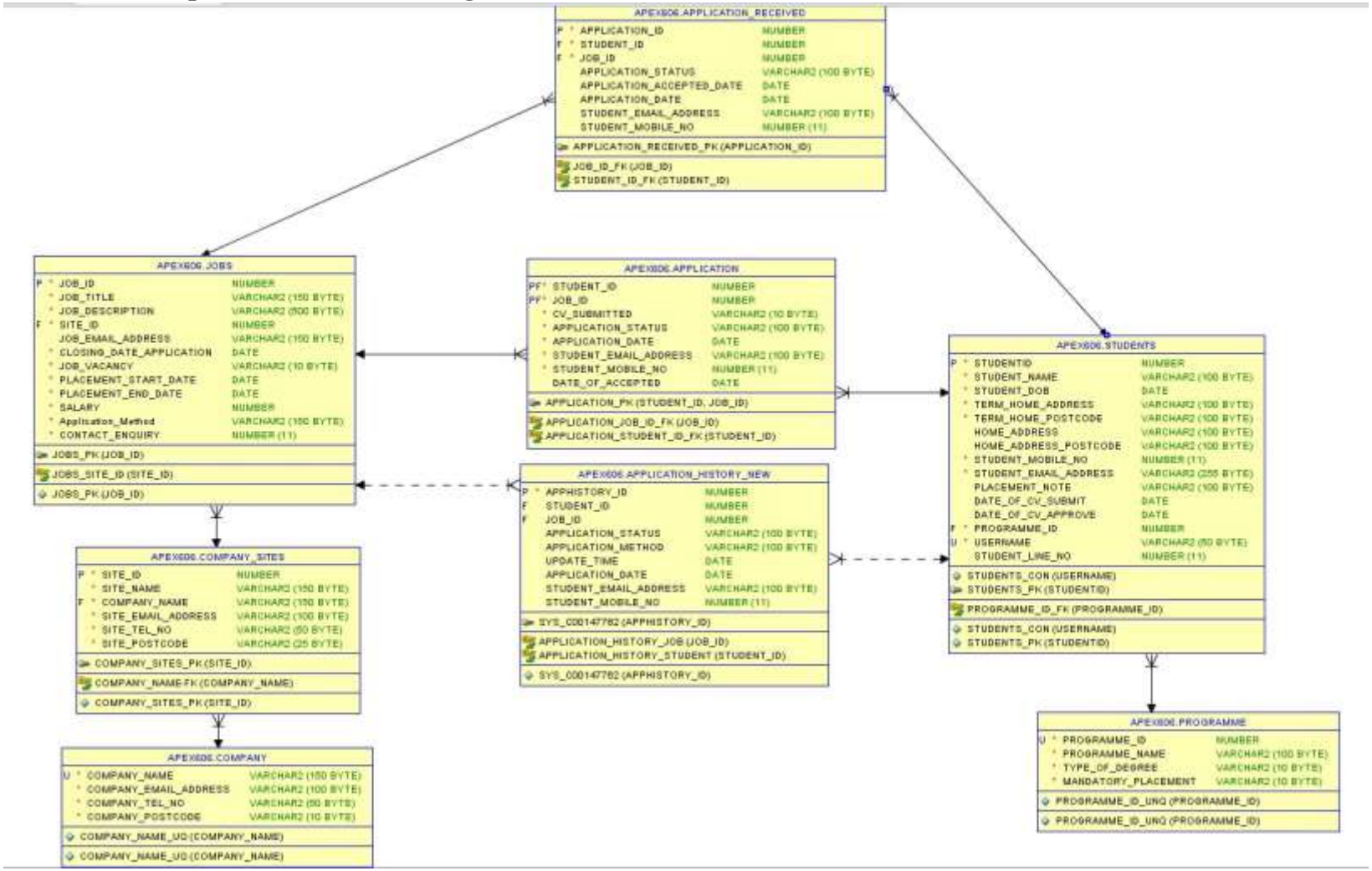


Figure 3 Application Received table.

3.2 SQL Developer Data Modeller diagram



The screenshot above shows our Developer data modeller diagram, complete with the relationships.

4. Testing

We used triggers, sequences and SQL statements in our testing, many of which we have discarded or rewritten throughout the testing process to keep them as simple and efficient as we possibly could. We have now arrived at what we think is the simplest version possible, at least per what we have intended to do. Since we have finished the database we have tested every part of it robustly to ensure there are no system breaking bugs to crash the record, or to render it unusable, and we are confident that if any bugs remain they are minor and should not cause any real issues in the day-to-day usage of the system.

5. Evaluation

We feel that the system is complete enough to be functional for the University administrator and students to use our system. The regular creating, updating and checking the data that has been stored in the system can be easily managed and followed per the whims of the user without too much hassle or technical knowledge. If, however we were to start to it from scratch we may have decided, now that we are more experienced with the design process to use more SQL statement than triggers and sequences for navigating the database and forms. It would just tidy up the whole thing and keep all of data much more manageable for editing purposes. If we were to carry on maintaining the system throughout its life cycle, we would likely release an update to fix the issue with the pie chart so the user would be able to see the proportion of students unplaced to students placed properly.

6. Critical Appraisal

Our team tried to create variety of triggers and SQL statements to develop our database system. We had a struggle with coding the login page for the student mobile application, to pass data from Application to Application History, and to create the pie chart. After much researching over the internet and revising Oracle SQL code, we eventually found our solution. The first solution that we found was the following:

- When on home page of APEX, click dropdown menu and select “Administration”
- Add the email addresses of all the students that we wanted to add (Exclude @ domain as part of the username)
- We then set the default password for these students, clicked next
- Confirmation screen appears, and now we have created our users.

For our second solution, we found how to be able to get data from the associated username. The SQL for this follows:

SELECT APPLICATION_HISTORY_NEW.*, STUDENTS.USERNAME

FROM APPLICATION_HISTORY_NEW

INNER JOIN STUDENTS

ON APPLICATION_HISTORY_NEW.STUDENT_ID = STUDENTS.STUDENTID

INNER JOIN STUDENTS

ON APPLICATION_HISTORY_NEW.STUDENT_EMAIL_ADDRESS = STUDENTS.STUDENT_EMAIL_ADDRESS
WHERE STUDENTS.USERNAME = LOWER (:P4_USERNAME)

And for our third solution we used triggers to pass data from Application to Application History. These are the triggers that we have used:

TRG_APPLICATION

```
CREATE OR REPLACE EDITIONABLE TRIGGER  "TRG_APPLICATION"  BEFORE INSERT OR UPDATE OF DATE_OF_ACCEPTED ON APPLICATION FOR EACH
ROW BEGIN

    IF(:NEW.DATE_OF_ACCEPTED > SYSDATE) THEN       RAISE_APPLICATION_ERROR(-20000, 'Applicant cannot be accepted before
application is submitted');     END IF;
```

```
    IF INSERTING OR UPDATING THEN       INSERT INTO APPLICATION_HISTORY_NEW(STUDENT_ID, JOB_ID, APPLICATION_STATUS,
APPLICATION_DATE, STUDENT_EMAIL_ADDRESS, STUDENT_MOBILE_NO)       VALUES (:NEW.STUDENT_ID, :NEW.JOB_ID,
:NEW.APPLICATION_STATUS, :NEW.APPLICATION_DATE, :NEW.STUDENT_EMAIL_ADDRESS, :NEW.STUDENT_MOBILE_NO);

      IF INSERTING OR UPDATING THEN       INSERT INTO APPLICATION_RECEIVED(STUDENT_ID, JOB_ID, APPLICATION_STATUS,
APPLICATION_DATE, STUDENT_EMAIL_ADDRESS, STUDENT_MOBILE_NO, CV_SUBMITTED)       VALUES (:NEW.STUDENT_ID, :NEW.JOB_ID,
:NEW.APPLICATION_STATUS, :NEW.APPLICATION_DATE, :NEW.STUDENT_EMAIL_ADDRESS, :NEW.STUDENT_MOBILE_NO, :NEW.CV_SUBMITTED);

    IF(:NEW.APPLICATION_STATUS = 'Offer Accepted') THEN       INSERT INTO APPLICATION_RECEIVED(STUDENT_ID, JOB_ID,
APPLICATION_ID)       VALUES (:NEW.STUDENT_ID, :NEW.JOB_ID,1);       DELETE FROM APPLICATION WHERE (:NEW.APPLICATION_STATUS =
'Offer Accepted');     END IF;

    DELETE FROM APPLICATION WHERE (:NEW.APPLICATION_STATUS = 'Applicant Withdraw');
    DELETE FROM APPLICATION WHERE (:NEW.APPLICATION_STATUS = 'Applicant Rejected');

    END IF;     END IF;

END;
/
ALTER TRIGGER  "TRG_APPLICATION" ENABLE
/
```

And finally, for our critical appraisal, we found that our pie chart did not work properly because even though we have our data store in our system all the data that counts as “application accepted” still counts as an unplaced student. We have been trying to fix our SQL statement but we could not find any other way to fix our solution. Here is the SQL that we used to create our pie chart:

```
select null as link, 'Placed Student' as label, count(*) as value

from APPLICATION_HISTORY_NEW

WHERE APPLICATION_STATUS ='Offer Accepted'  UNION
```

```
select null as link, 'Unplaced Student' as label, count ( distinct STUDENTS.STUDENTID) as value from
APPLICATION_HISTORY_NEW, STUDENTS
WHERE STUDENTS.STUDENTID = APPLICATION_HISTORY_NEW.STUDENT_ID and APPLICATION_STATUS = 'Application Submitted';
```

If we still had time we could fix it but because we only have a limited amount of time we are not able to find the solution.

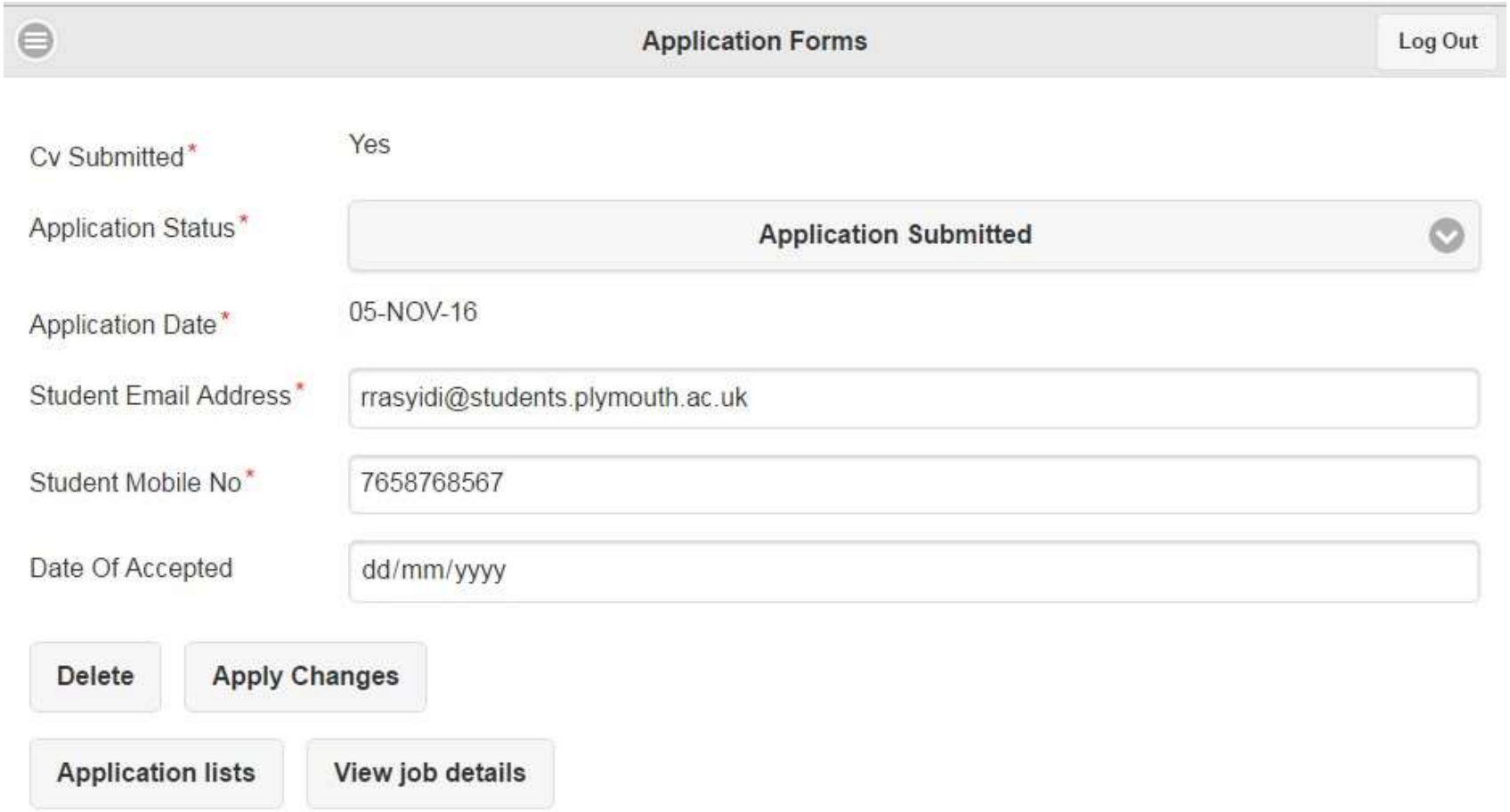
7. Sets of screenshots

Student Mobile Application

7.1 Login Page

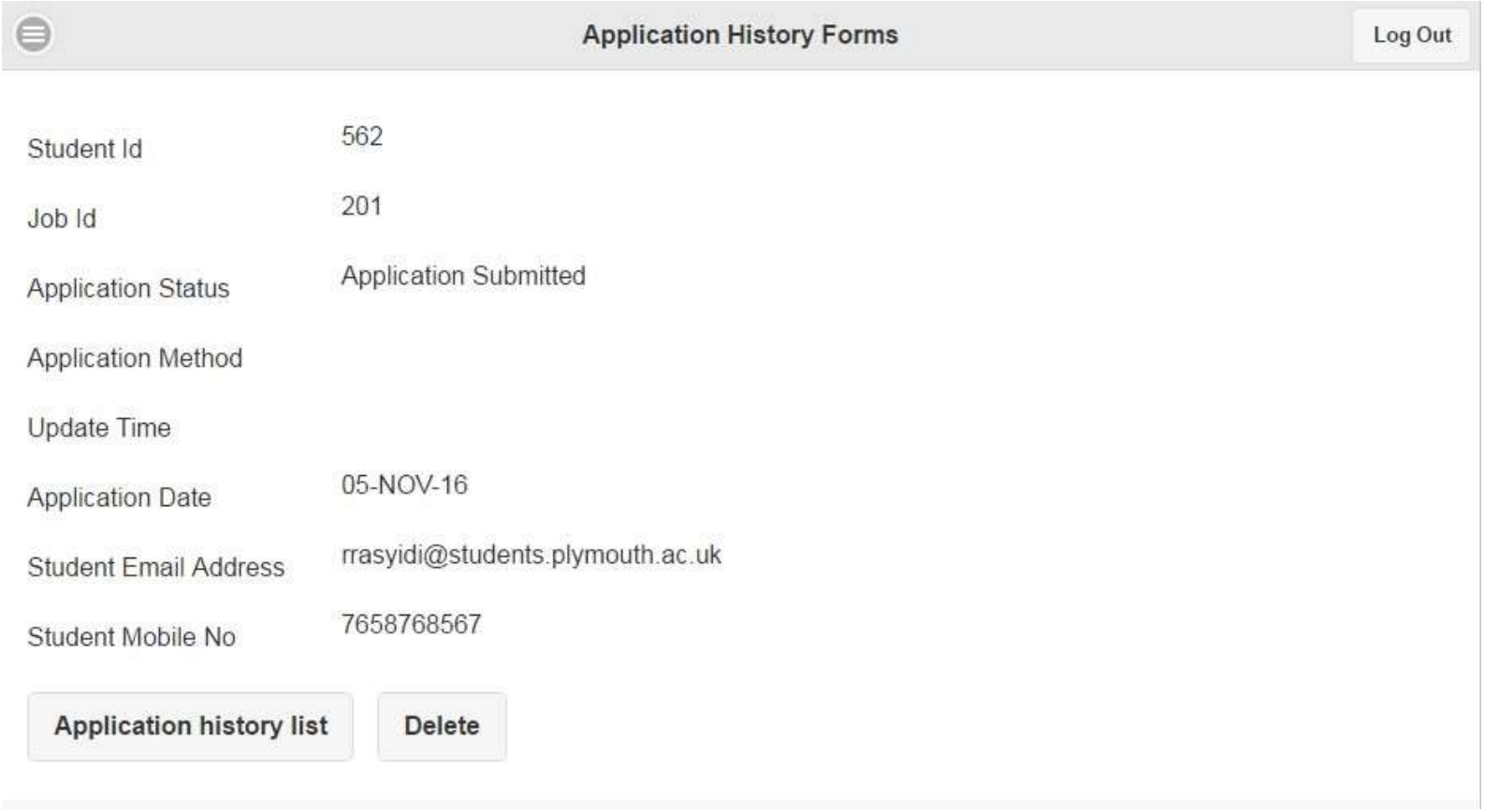
Login page

7.2 Application Details



Application Details page

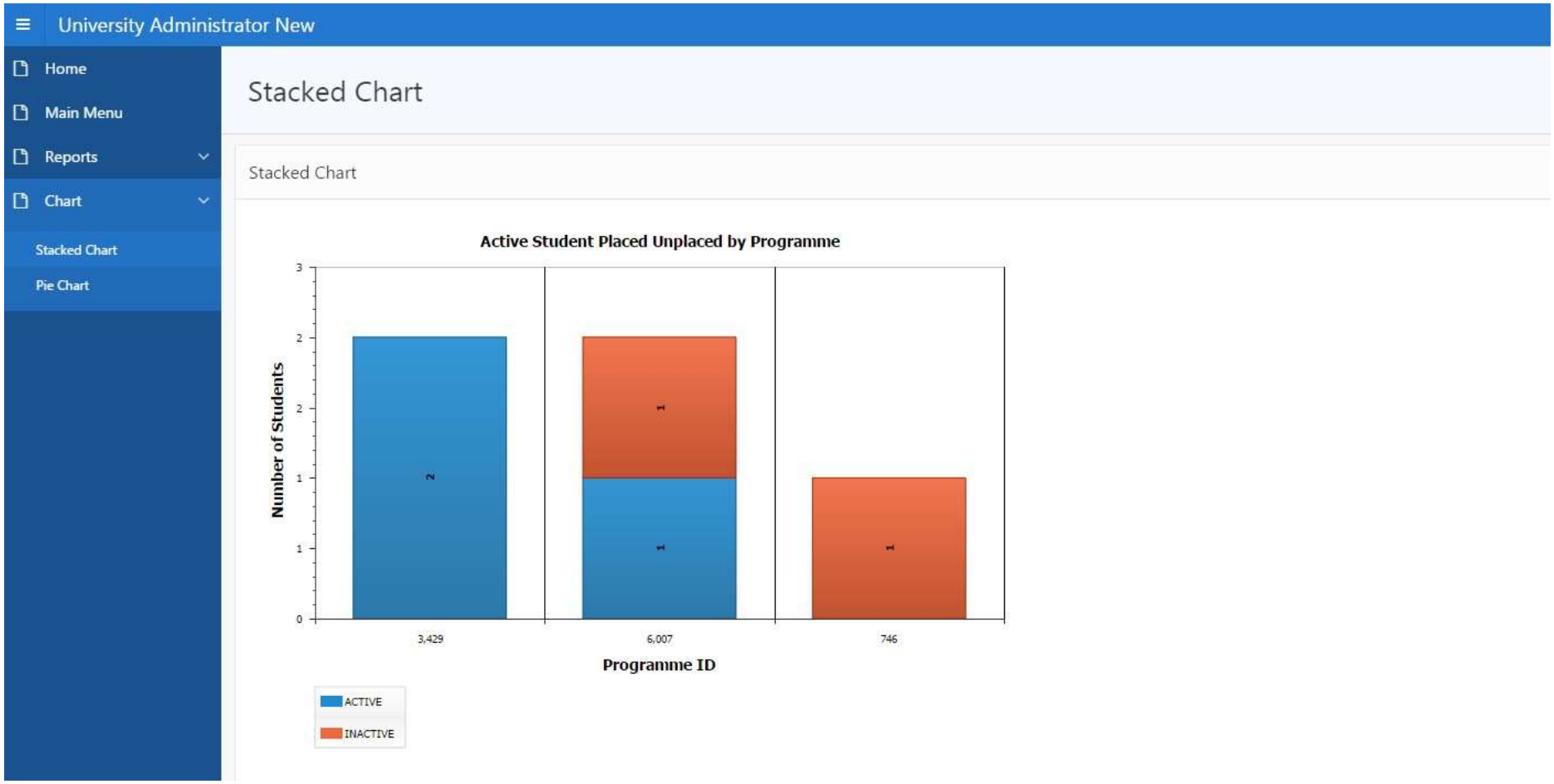
7.3 Application History Form



Application History form

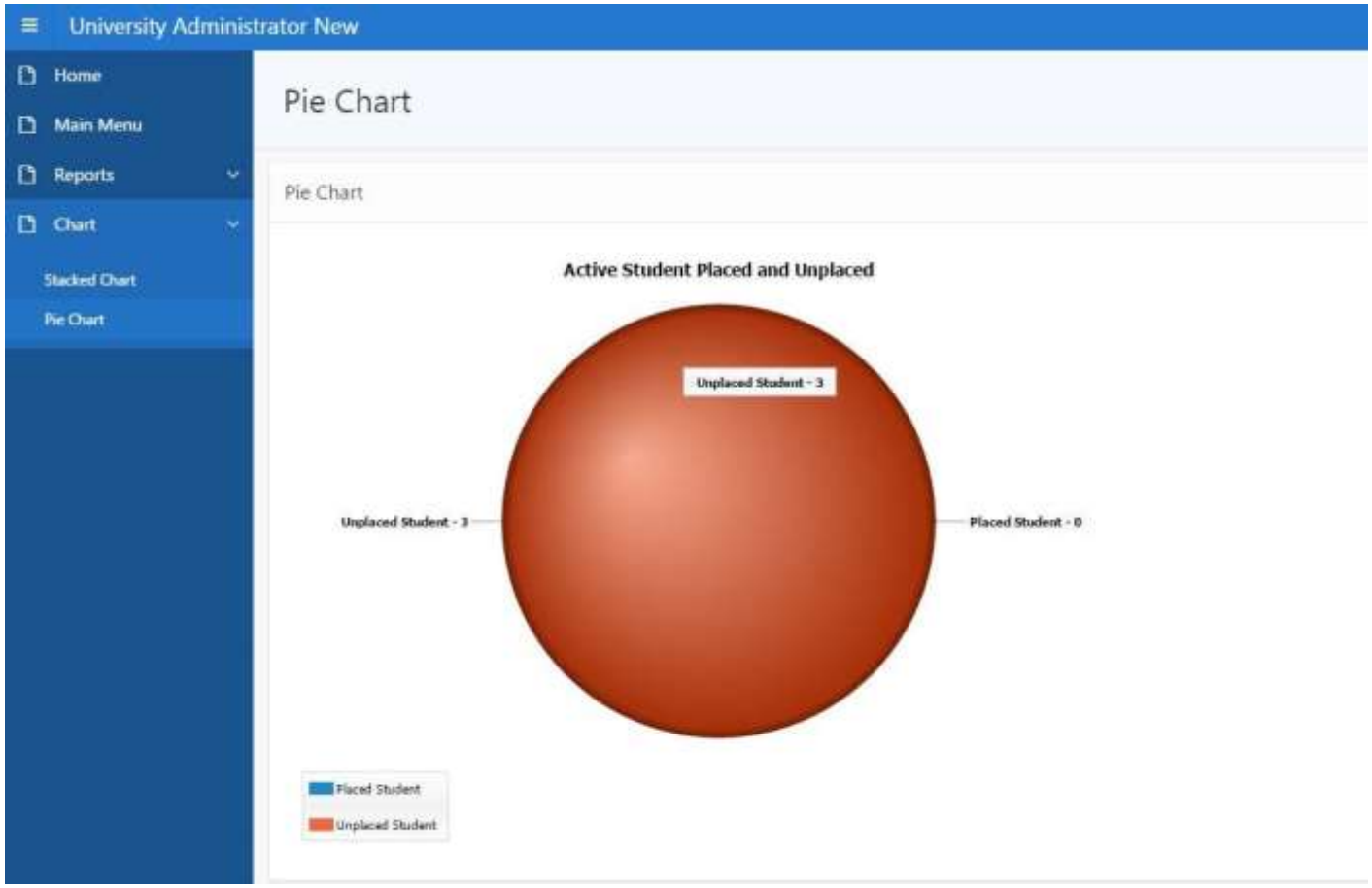
Desktop Application

7.4 Stack bar chart



Stack bar chart form

7.5 Pie Chart



Pie chart form

8. Project Plan

	Placement Application Tracking System				
Week beginning	Task	Member Assigned	To be Finished by	Progress	Status
				Pending/Success	(L/NL)
10/7/2016	Normalisation table(s) and ERD	Everyone	10/10/2016; 15PM	Success	Live
	Students				
	Company sites				
	Programme				
	Jobs				
	Companies				
	Application				
	Application Method				
	Application Status				
	Application history				
10/7/2016	Oracle SQL (optional)	Everyone	Complete	Complete	Live
	Oracle Application express (Mandatory)	Everyone	2/11/2016; 23:59 PM (EXPECTED)	Complete	Live
10/10/2016	Implementation Oracle Application Express:				
	Admnistrator's desktop application:				Live

	To enter and maintain data relating to students,	Rafi	01/11/2016; 23:59 PM (EXPECTED)	Success	Live
	companies, company sites and job vacancies				
	To browse/search current and past job vacancies	Rafi	01/11/2016; 23:59 PM	Success	Live
	To view a report on status student application(s)	Rafi	01/11/2016; 23:59 PM	Success	Live
	To provide management report to show	Rafi	01/11/2016; 23:59 PM	Success	Live
	To display a management dasboard with a chart	Rafi	01/11/2016; 23:59 PM	Success	Live
	Student's mobile web application:		01/11/2016; 23:59 PM	Success	Live
	To view/browse/search current job vacancies	Cliff, Sandra, Tom		Success	Live
	To allow a student to record applications made and maintain her	Cliff, Sandra, Tom	05/11/2016; 23:59 PM	Success	Live
	application(s) history				
30/10/2016					

	To allow a student to view a report on the status of his/her application(s)	**Cliff, Sandra, Tom**	**05/11/2016; 23:59 PM**	**Success**	**Live**
	Final test and revision Oracle Application	**Everyone**	**05/11/2016; 23:59 PM**	**Success**	**Live**

The above is our project plan, with displays our progress over the past 6 weeks.