Appendix A

COMP1424 Logbook Upload Template

1. Basic Information

1.1 Student name: Usman Basharat 000874782	
1.2 Who did you work with? Note that for logbook exercises, you are allowed to work with one other person as long as you give their name and login id and both contribute to the work.	
1.3 Which Exercise is this? Tick as appropriate.	2 – NameEntry ☐ 3 – Initial GUI for MOBSER ☐ 4 – Initial database for MOBSER ✓ 5 – Initial MOBSER PhoneGap App ☐
1.4 Based on the grading criteria in the logbook specification what mark do you feel you deserve for this piece of work?	0 – little or no attempt ☐ 1 – an attempt but could do less than half properly ☐ 2 – everything done but with significant flaws ☐ 3 – excellent – everything done and only minor flaws ☐ 4 – exemplary – everything done perfectly ✓
1.5 Briefly explain your answer to question 1.4 e.g. what you could have done better?	In this lab, I created a database within this application that stores all the data that has been inserted. Some changes were made from previous GUI. I had used a data picker. However, unable to get the selected date to change from using this. I changed this for a DatePickerDialog to appear. These are the current changes that has been made. However, all of this is subject to change. Also, changes were made to the toolbar too.

2. Exercise answer

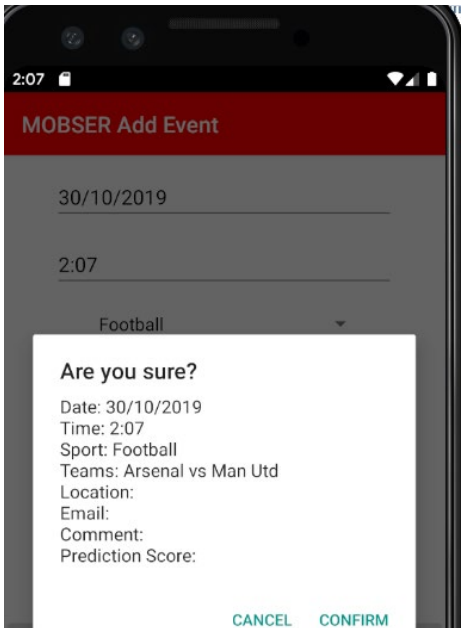
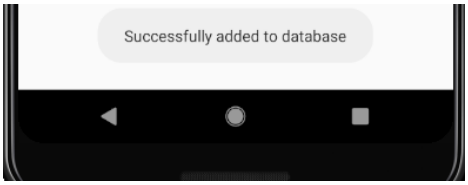


Figure 1 shows the data being entered and successful into the database.

Figure 1 shows the details that has been entered within the database. A confirmation box has been issued alongside if the user wants to confirm or make changes to the data been entered. Once the user selects confirm, this data that has been entered, it is inserted within the database. A message appears for the user to be informed that this has been successfully inserted within the database. Validation and option of the rest of the inputs are all optional. Therefore, as shown, the user only chose to do the first four.



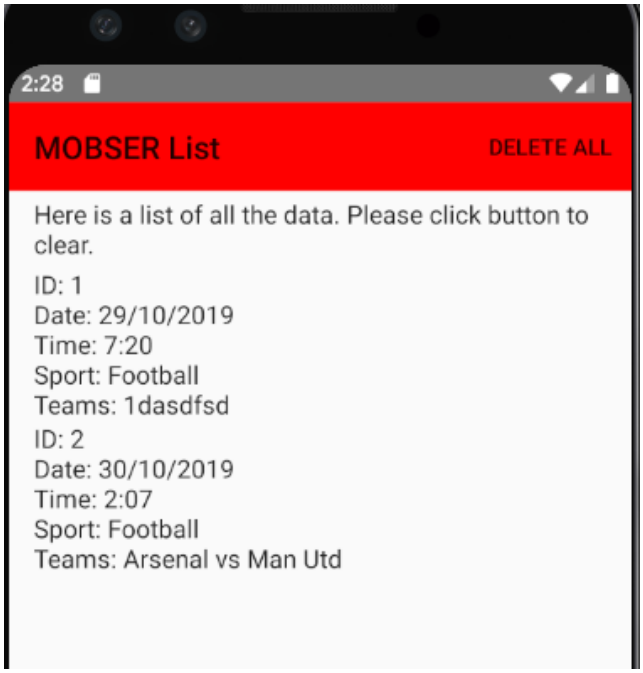


Figure 2 shows the database in a listview.

They are numerous ways of viewing data from the database. Referring to Figure 2, one of the ways you can view the database that is inserted is add all information in a list. Therefore, you can view what has been entered within Figure 1.

Figure 3 shows the database entries within the database that match Figure 2. To do this, you have to go on Device File Manager – data > data > the application > database. You will find your database table stored here. Once you do this, you can view your table through an extension provided by Google Chrome called SQLite Manager. Figure 3 shows the outcome.

Export	ID	DATE	TIME	SPORT	TEAMS	LOCATION	EMAIL	COMMENT	PREDICTION	ACTUALSCORE
1	1	29/10/2019	7:20	Football	1dasdfsd	NULL	NULL	NULL	NULL	NULL
2	2	30/10/2019	2:07	Football	Arsenal vs Man Utd		NULL	NULL	NULL	NULL

Figure 3 shows the database entries within the database.

2.2 Code that you wrote

Please note all classes are not included. Only included those that are relevant and important to this lab.

DatabaseSingleton.java

```
package com.example.mobsercoursework;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class DatabaseSingleton extends SQLiteOpenHelper {

    private static final String dbName = "Events.db";
    private static final String table = "addEvent";
    private static final String column1 = "ID";
    private static final String column2 = "DATE";
    private static final String column3 = "TIME";
    private static final String column4 = "SPORT";
    private static final String column5 = "TEAMS";
    private static final String column6 = "LOCATION";
    private static final String column7 = "EMAIL";
    private static final String column8 = "COMMENT";
    private static final String column9 = "PREDICTION";
```

```
    private static final String column10 = "ACTUALSCORE";

    public DatabaseSingleton(Context context) {
        super(context,dbName, null, 1);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
       // String myPath ="C:/Users/user/Desktop/MOBSERCoursework2/Events.db";
       // SQLiteDatabase.openOrCreateDatabase(myPath, null, null);
        db.execSQL(" create table " + table + "(ID INTEGER PRIMARY KEY
AUTOINCREMENT, " +
                "DATE TEXT, TIME TEXT, SPORT TEXT, TEAMS TEXT, " +
                "LOCATION TEXT, EMAIL TEXT, COMMENT TEXT, PREDICTION TEXT,
ACTUALSCORE TEXT)");
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int i, int i1) {
        db.execSQL("DROP TABLE IF EXISTS " + table);
        onCreate(db);
    }

    public Boolean insertData(String date, String time, String sport, String teams,
                              String location, String email, String comment,
                              String prediction) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put( column2 ,date);
        contentValues.put( column3 ,time);
        contentValues.put( column4 ,sport);
        contentValues.put( column5 ,teams);
        contentValues.put( column6 ,location);
        contentValues.put( column7 ,email);
        contentValues.put( column8 ,comment);
        contentValues.put( column9 ,prediction);

        long result = db.insert(table, null, contentValues);
        if(result == -1) {
            return false;
        } else {
            return true;
        }
    }
    public Cursor getAllID() {
        SQLiteDatabase db = this.getWritableDatabase();
        Cursor res = db.rawQuery("SELECT * FROM " + table, null);
        return res;
    }

    public Cursor getSpecificID(int id) {
        SQLiteDatabase db = this.getWritableDatabase();
        Cursor res = db.rawQuery("SELECT * FROM " + table + " WHERE " + column1 + "
= " + id + "", null);
        return res;
    }
    public boolean updateData(int id, String date, String time, String  sport,
String teams,
                              String location, String email, String comment,
                              String prediction) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put(column2, date);
        contentValues.put(column3, time);
        contentValues.put(column4, sport);
        contentValues.put(column5, teams);
        contentValues.put(column6, location);
        contentValues.put(column7, email);
```

```
        contentValues.put(column8, comment);
        contentValues.put(column9, prediction);
        db.update(table, contentValues, "ID = ?", new String[]
{String.valueOf(id)});
        return true;
    }
    public Boolean updateScore(char id, String score) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put(column10, score);
        db.update(table, contentValues, "ID = ?", new String[]
{String.valueOf(id)});
        return true;
    }
    public Boolean deleteData(int id) {
        SQLiteDatabase db = this.getWritableDatabase();
        db.execSQL("delete from "+ table + " WHERE ID =" + id);
        return true;
    }
    public Boolean deleteDataAll() {
        SQLiteDatabase db = this.getWritableDatabase();
        db.execSQL("delete from "+ table);
        return true;
    }

}
```

AddEvent.java

```
package com.example.mobsercoursework;

import android.app.AlertDialog;
import android.app.DatePickerDialog;
import android.app.TimePickerDialog;
import android.content.DialogInterface;
import android.os.Build;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TimePicker;
import android.widget.Toast;

import androidx.annotation.RequiresApi;
import androidx.appcompat.app.AppCompatActivity;

import java.util.Calendar;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class AddEvent extends AppCompatActivity {


    private static final String TAG = "AddEvent";
    private DatePicker dateReview;
    private Spinner spinnerSport;
    private DatePickerDialog datePickerDialog;

    private TimePickerDialog.OnTimeSetListener timeSetListener;
    private EditText editTextDate;
```

```
    private EditText editTextTime;
    private EditText editTextTeams;
    private EditText editTextLocation;
    private EditText editTextEmail;
    private EditText editTextComment;
    private EditText editTextPrediction;
    private Button addEvent;
    private String date;
    ArrayAdapter<String> adapterSport;
    DatabaseSingleton db;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.add_event_activity);
        db = new DatabaseSingleton(this);
        editTextDate = (EditText) findViewById(R.id.editTextDate);

        editTextTime = (EditText) findViewById(R.id.editTextTime);
        spinnerSport = (Spinner) findViewById(R.id.spinnerSport);
        editTextTeams = (EditText) findViewById(R.id.editTextPlayed);
        editTextLocation = (EditText) findViewById(R.id.editTextLocation);
        editTextEmail = (EditText) findViewById(R.id.editTextEmail);
        editTextComment = (EditText) findViewById(R.id.editTextComment);
        editTextPrediction = (EditText) findViewById(R.id.editTextPrediction);
        addEvent = (Button) findViewById(R.id.addEvent);

        adapterSport = new ArrayAdapter<String>(AddEvent.this,
android.R.layout.simple_list_item_1,
                getResources().getStringArray(R.array.list));

adapterSport.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)
;
        spinnerSport.setAdapter(adapterSport);
        editTextDate.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View view) {
                Calendar cal = Calendar.getInstance();
                int year = cal.get(Calendar.YEAR);
                int month = cal.get(Calendar.MONTH);
                int day = cal.get(Calendar.DAY_OF_MONTH);

                datePickerDialog = new DatePickerDialog(AddEvent.this,
                        new DatePickerDialog.OnDateSetListener() {
                            @Override
                            public void onDateSet(DatePicker datePicker, int year,
int month, int day) {
                                editTextDate.setText(day + "/" + (month + 1) + "/"
+ year);
                            }
                        }, year, month, day);
                datePickerDialog.show();
                editTextDate.setError(null);
            }
        });

        editTextTime.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                Calendar cal = Calendar.getInstance();
                int hour = cal.get(Calendar.HOUR);
                int minute = cal.get(Calendar.MINUTE);
                Log.d(TAG, "" + hour + minute);

                TimePickerDialog timeDialog = new TimePickerDialog(AddEvent.this,
timeSetListener,
                        hour, minute, true);
```

```
                timeDialog.setTitle("Select Time");
                timeDialog.show();
            }
        });
        timeSetListener = new TimePickerDialog.OnTimeSetListener() {
            @RequiresApi(api = Build.VERSION_CODES.M)
            @Override
            public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
                if (minute == 0 || minute == 1 || minute == 2 ||
                        minute == 3 || minute == 4 || minute == 5 ||
                        minute == 6 || minute == 7 || minute == 8 || minute == 9) {
                    editTextTime.setText(hourOfDay + ":0" + minute);
                    editTextTime.setError(null);
                } else {
                    editTextTime.setText(hourOfDay + ":" + minute);
                    editTextTime.setError(null);
                }
            }
        };
        editTextTeams.addTextChangedListener(new TextChanged() {
            @Override
            public void onTextChanged(CharSequence charSequence, int i, int i1, int
i2) {
                getStringValidationTeams();
            }
        });
        editTextLocation.addTextChangedListener(new TextChanged() {
            @Override
            public void onTextChanged(CharSequence charSequence, int i, int i1, int
i2) {
                getStringValidationLocation();
            }
        });
        editTextComment.addTextChangedListener(new TextChanged() {
            @Override
            public void onTextChanged(CharSequence charSequence, int i, int i1, int
i2) {
                getStringValidationComment();
            }
        });
        editTextEmail.addTextChangedListener(new TextChanged() {
            @Override
            public void onTextChanged(CharSequence charSequence, int i, int i1, int
i2) {
                getValidationEmail();
            }
        });

        editTextPrediction.addTextChangedListener(new TextChanged() {
            @Override
            public void onTextChanged(CharSequence charSequence, int i, int i1, int
i2) {
                getStringValidationPrediction();
            }
        });

        addEvent.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View view) {
                if (getValidation() == false) {
                    Toast toast = Toast.makeText(AddEvent.this, "Please confirm the
review", Toast.LENGTH_SHORT);
                    toast.show();
                    getMessage();
                }
            }
        });
    }
```

```
    private Boolean getValidation() {
        if (TextUtils.isEmpty(editTextDate.getText())) {
            editTextDate.setError("Required!");
            return true;
        } else if (TextUtils.isEmpty(editTextTime.getText())) {
            editTextTime.setError("Required!");
            return true;
        }
        else if (TextUtils.isEmpty(editTextTeams.getText())) {
            editTextTeams.setError("Required!");
            return true;
        } else if (getStringValidationTeams() == false) {
            return true;
        }
        return false;
    }

    private boolean getStringValidationTeams() {
        Matcher matchTeams =
getPatternString().matcher(editTextTeams.getText().toString());
        if (!matchTeams.matches()) {
            editTextTeams.setError("Please enter valid information");
            return false;
        }
        return true;
    }

    private boolean getStringValidationLocation() {
        Matcher matchLocation =
getPatternString().matcher(editTextLocation.getText().toString());
        if (!matchLocation.matches()) {
            editTextLocation.setError("Please enter valid information");
            return false;
        }
        return true;
    }

    private boolean getStringValidationComment() {
        Matcher matchTeams =
getPatternString().matcher(editTextComment.getText().toString());
        if (!matchTeams.matches()) {
            editTextComment.setError("Please enter valid information");
            return false;
        }
        return true;
    }

    private boolean getStringValidationPrediction() {
        Matcher matchTeams =
getPatternPrediction().matcher(editTextPrediction.getText().toString());
        if (!matchTeams.matches()) {
            editTextPrediction.setError("Please enter valid information");
            return false;
        }
        return true;
    }

    private boolean getValidationEmail() {
        Matcher matchTeams =
getPatternEmail().matcher(editTextEmail.getText().toString());
        if (!matchTeams.matches()) {
            editTextEmail.setError("Please enter valid email");
            return false;
        }
        return true;
    }
```

```
    private Pattern getPatternString() {
        String patternString = "^[-_a-zA-Z0-9.\\s*]+$";
        Pattern pattern = Pattern.compile(patternString);
        return pattern;
    }

    private Pattern getPatternPrediction() {
        String patternString = "^[-0-9]+$";
        Pattern pattern = Pattern.compile(patternString);
        return pattern;
    }

    private Pattern getPatternEmail() {
        String patternString = "^[A-Z0-9._%+-]+@[A-Z0-9.-]+\\.[A-Z]{2,6}$";
        Pattern pattern = Pattern.compile(patternString, Pattern.CASE_INSENSITIVE);
        return pattern;
    }

    private String getConfirmationMessage() {
        return "Date: " + editTextDate.getText() + "\n" +
                "Time: " + editTextTime.getText() + "\n" +
                "Sport: " + spinnerSport.getSelectedItem() + "\n" +
                "Teams: " + editTextTeams.getText() + "\n" +
                "Location: " + editTextLocation.getText() + "\n" +
                "Email: " + editTextEmail.getText() + "\n" +
                "Comment: " + editTextComment.getText() + "\n" +
                "Prediction Score: " + editTextPrediction.getText() + "\n";
        //Give a rating out of 5 of your enjoyment of the game
    }

    private AlertDialog.Builder getMessage() {
        AlertDialog.Builder alert = new AlertDialog.Builder(AddEvent.this);
        alert.setTitle("Are you sure?");
        alert.setMessage(getConfirmationMessage());
        alert.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {

            }
        }).setPositiveButton("Confirm", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                try {
                   addData();
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        }).show();
        return alert;
    }
    public void addData() {
        boolean isInserted =  db.insertData(editTextDate.getText().toString(),
editTextTime.getText().toString(),
                spinnerSport.getSelectedItem().toString(),
                editTextTeams.getText().toString(),
editTextLocation.getText().toString(),
                editTextEmail.getText().toString(),
                editTextComment.getText().toString(),
editTextPrediction.getText().toString());
        if(isInserted == true) {
            Toast.makeText(AddEvent.this, "Successfully added to database",
Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(AddEvent.this, "Unsuccessfully added to database",
Toast.LENGTH_SHORT).show();
        }
    }
```

```
}
```

add_event_activity.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/scrollView4"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".AddEvent">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <EditText
            android:id="@+id/editTextDate"
            android:layout_width="303dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginTop="15dp"
            android:autofillHints=""
            android:ems="10"
            android:focusable="false"
            android:hint="@string/clickMe1"
            android:inputType="time" />


        <EditText
            android:id="@+id/editTextTime"
            android:layout_width="303dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginTop="15dp"
            android:autofillHints=""
            android:ems="10"
            android:focusable="false"
            android:hint="@string/clickMe"
            android:inputType="time" />


        <Spinner
            android:id="@+id/spinnerSport"
            android:layout_width="255dp"
            android:layout_height="44dp"
            android:layout_gravity="center"
            android:layout_marginTop="8dp"
            android:layout_weight="1" />

        <EditText
            android:id="@+id/editTextPlayed"
            android:layout_width="303dp"
            android:layout_height="46dp"
            android:layout_gravity="center"
            android:layout_marginBottom="8dp"
            android:ems="10"
            android:hint="Teams"
            android:inputType="textPersonName" />

        <EditText
            android:id="@+id/editTextLocation"
            android:layout_width="303dp"
            android:layout_height="46dp"
            android:layout_gravity="center"
```

```
            android:ems="10"
            android:hint="@string/location"
            android:inputType="textPersonName" />

        <EditText
            android:id="@+id/editTextEmail"
            android:layout_width="303dp"
            android:layout_height="46dp"
            android:layout_gravity="center"
            android:ems="10"
            android:hint="@string/email"
            android:inputType="textEmailAddress" />


        <EditText
            android:id="@+id/editTextComment"
            android:layout_width="303dp"
            android:layout_height="46dp"
            android:layout_gravity="center"
            android:ems="10"
            android:hint="@string/comment"
            android:inputType="text" />


        <EditText
            android:id="@+id/editTextPrediction"
            android:layout_width="303dp"
            android:layout_height="46dp"
            android:ems="10"
            android:layout_gravity="center"
            android:hint="Prediction"
            android:inputType="phone" />

        <Button
            android:id="@+id/addEvent"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:text="@string/addEvent" />

    </LinearLayout>

</ScrollView>
```