

COMP-1424-M01-2021-22 Mobile Application Development

Redwan Ahmed ID :001147921

GROUP-38 Jahidul Islam Sany ID:001197465

Venkata Akhilash ID:001151477

CONTENTS

| | |
|--|----|
| ABSTRACT | 2 |
| INTRODUCTION | 2 |
| SECTION 1: CONCISE TABLE. | 2 |
| SECTION 2: Reflection on how it was to work | 3 |
| Individual role in the group | 4 |
| What went well | 5 |
| What did go as anticipated..... | 5 |
| What did not go as required. | 5 |
| 3) SECTION 3: Application evaluation..... | 6 |
| Security..... | 6 |
| Capability of the application to run on different screens | 8 |
| Changes to be made for live deployment | 9 |
| SECTION 4: Demonstrating implemented features. | 10 |
| Employees enter they credentials then log in..... | 10 |
| SECTION 4; CODE LISTING:..... | 17 |
| SECTION 4; CODE LISTING Without Background: | 23 |
| checks the length of the inputs | 24 |
| gets the button id and perform onclick function | 24 |
| Equating the inputs wih the defined user name and password to check if they match | 25 |
| AlertDialog fetch the user iput and display for confirmation | 26 |
| inflates the map to the ui for user interface | 30 |
| Challenges encountered while developing the application:..... | 31 |
| The frequent crashing of the application | 31 |
| Solution..... | 31 |
| We had little knowledge on how to implement some features | 31 |
| Solution..... | 31 |
| The application not being able to display well on higher android versions..... | 31 |
| Solution..... | 31 |
| Conclusion: | 31 |
| References | 32 |

Report

ABSTRACT

M-Expense is based on an Android application in Java, and the platform was used is Android studio. The application can capture the trip details and all the expenses that occurred during that trip.

All the details of that specific trip or multiple trips are kept in the database. Data manipulation such as editing and updates can also be performed. The details of the trip Which the users regarding the expense perform can be seen by the admins. This application is also suited for cross platforms.

INTRODUCTION

As every organization is getting bigger and bigger, it's getting hard to maintain a constant workflow, Cost, employee, projects etc. For example, the employee has to go on various organization-directed trips. An organization needs to keep tabs on employees activity, schedule and a record of the costs.

Cost management is one of the essential parts of running an organization smoothly. The application M-Expense helps an organization to maintain that by storing all the data of the specific trip on a database. The expenses of the trips also saved the panda database. These trip details and expense reports are kept so that the organization can plan a financial estimation for future trips.

Management turn also has the ability to edit the employee's expense and trip details if there are any changes to the plan. Management can also search for trip details within that database, and the entire trip details are shown to them.

This app is easily accessible via phone Android or ISO, can take pictures of specific details the users wanna add, both admin and the user share the trip details.

SECTION 1: CONCISE TABLE.

All the functionalities we able to implement in the m-expense android application are as follows.

| FEATURES | IMPLEMENTATION |
|----------|-------------------|
| A | Fully implemented |

| | |
|---|--|
| B | Fully implemented |
| C | Fully implemented |
| D | Search option implemented no auto suggest |
| E | |
| F | Cordova prototype implemented |
| G | Implemented with additional features |
| H | Additional features ie taking photo adding location help option implemented in android version |

SECTION 2: REFLECTION ON HOW IT WAS TO WORK

The way the application is supposed to work is given here below:

On the opening of the application, one should see the opening page, which includes two options. One states that login as a user. Another is admin. The employee panel will log in as a user, and the management panel will log in as an admin.

When the employee is interested, correct login credential and logs in to the app, a new page with the (+) plus symbol Up the Bolton right corner of the page will be shown to them. Clicking that symbol opens a form that allows employees to fill in trip details. After completing the trip detail, they need to press the submit button.

When employees enter s their correct login credentials, then click login a new page with (+) symbol at the bottom right Conner of the page which on clicking opens a form which allows the employee to fill in the details of the trip they are to go then click the submit button once done filling the required details. The employees can also add photos if necessary into their trip details. There is also a key feature to add location in the app.

When it comes to the admins, they will see a page where older trip details are listed after logging in. After consulting with the employees, the admins can edit this expense list and tribute tales. Die admin can also delete old trips at a time from the database.

There is also a search tool available for both employees and the admin to search different trips and see all the details regarding those trips. The search options can be the name of the date of the trip.

IT is also anticipated that the application could save all the records to the cloud.

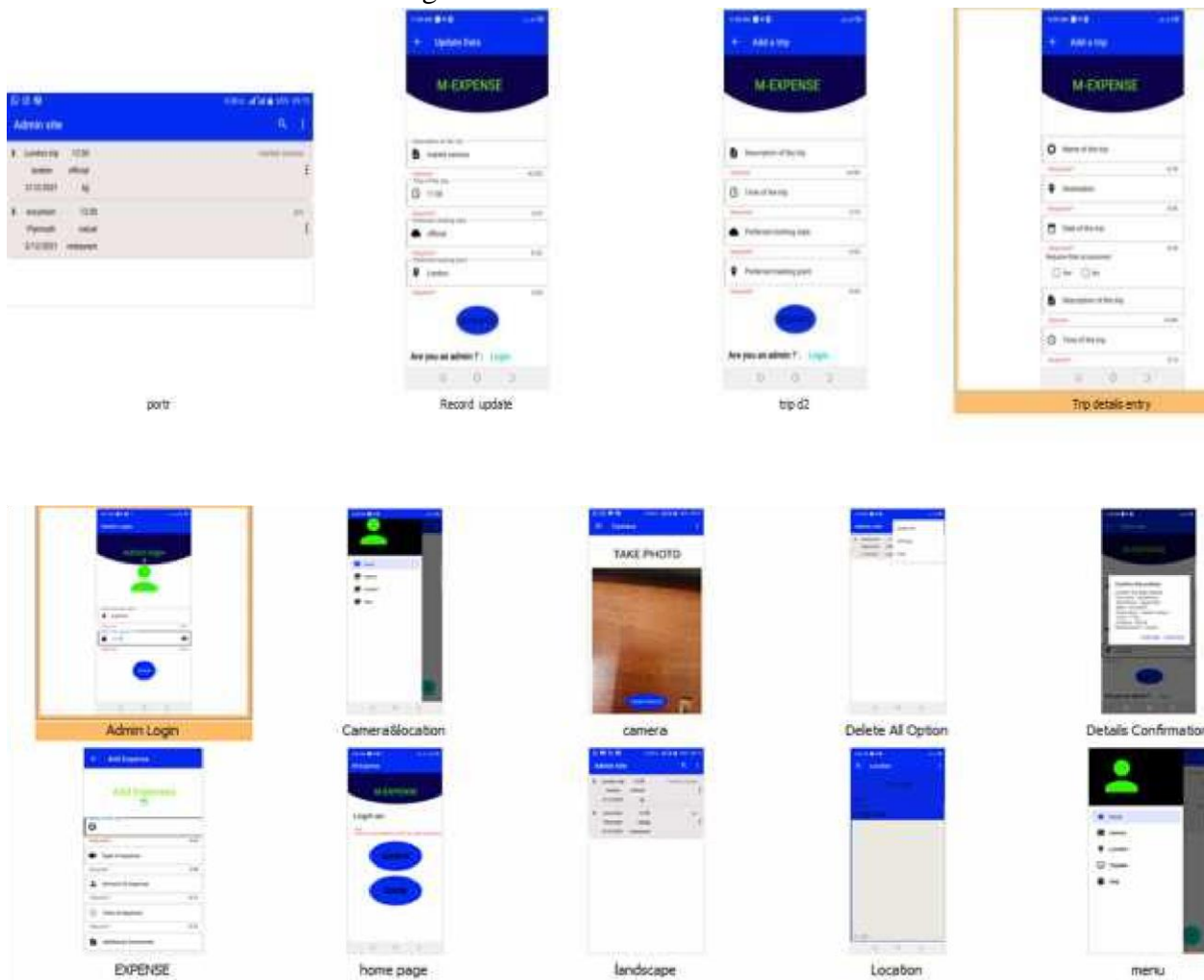
INDIVIDUAL ROLE IN THE GROUP

In this entire project, all tasks were divided by the teammates equally. But the part that I contributed most was the designing and the implementation of the feature (A), which is (entering details of the trip) and feature (B) containing (storage viewing, deleting trip details and

clearing

the

database.



Another feature that I helped implement was the camera functionality and Android application. This task was quite tricky, and I had several problems solving it, but with the help of YouTube and you Udemy, I was finally able to implement (YouTube) (Udemy).

WHAT WENT WELL

The employees can enter trip details without difficulty. A preview of the

Details entered are displayed for the employee to confirm the details before being saved to the database. Changes can also be made to the details saved where necessary. The admin can add expenses to the trip and also delete records.

WHAT DID GO AS ANTICIPATED

The implementation of all features (A) went smoothly, and capturing its details was also went smoothly.

For feature (B), storing viewing, and deleting Add, not to mention resetting the database, are also implemented properly.

Feature (C) the trip expense feature was also added successfully.

WHAT DID NOT GO AS REQUIRED.

Additional features such as location taking pictures, help option feature in a cross-platform prototype of this application could not be implemented. Another feature, autosuggest, while the user typing in the search bar could not be implemented.

Another key step backwards securing a cross-platform application with a password. Other data manipulation like delete, search records could not be done, and the prototype of this application is cross-platform. Another key failure w

as attempting to applaud details to cloud the application throws a connection error.

M-expense app restricts unauthorized personnel from using the application

Application

The application access is only restricted to two types of users

The admin (management)

User (Employees)

3) SECTION 3: APPLICATION EVALUATION.

SECURITY

M-expense app restricts unauthorized personnel from using the application

Application

The application access is only restricted to two types of users

The admin (management)

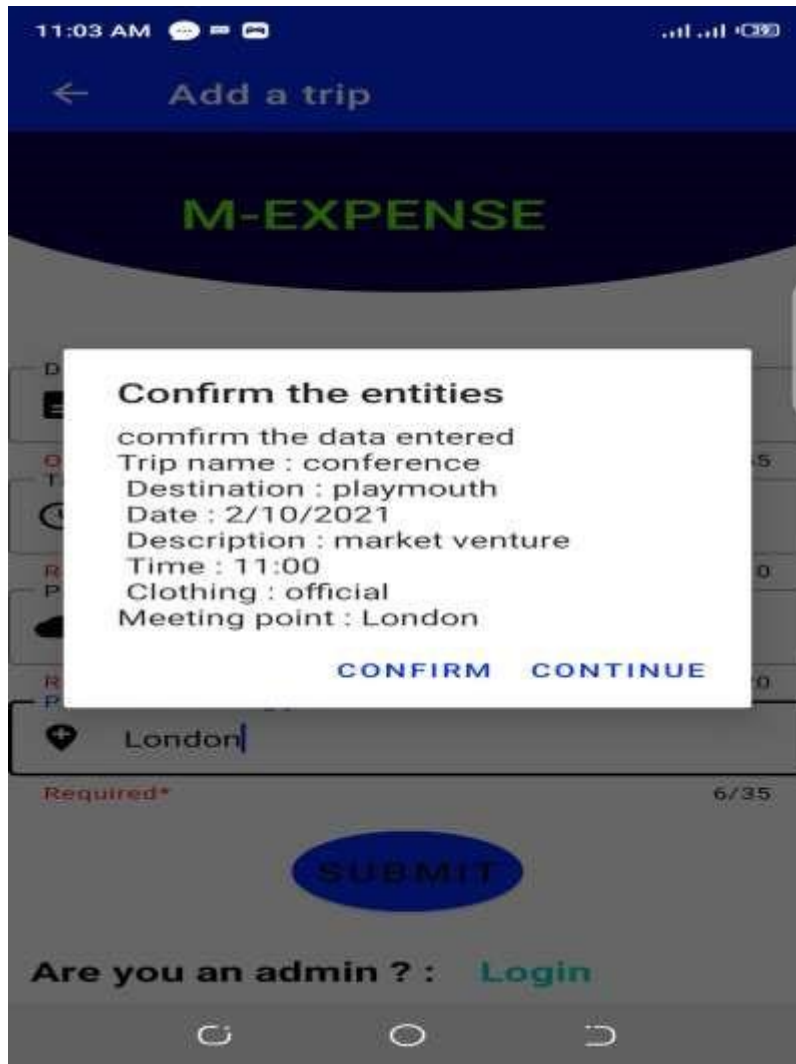
User (Employees)



The users (employees) log in credentials are different from the admin login credentials.

The user can only enter trip details and confirm them before saving them to the database.

The screenshot below illustrates:



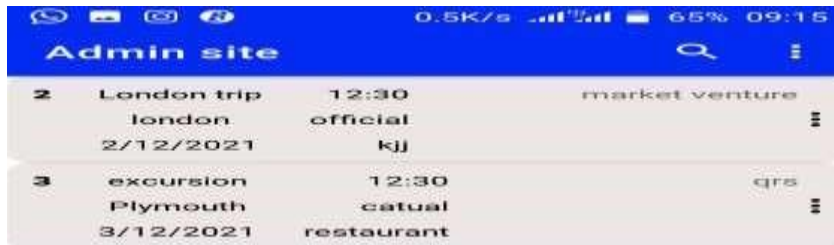
The application is divided into two parts one is admin, and another is the normal user section, To ensure there are some degrees of security for the date of application. As long as The employees Of the admin panel keeps their login credentials secret, all the details of this application, as well as the data, are safe.

The main security issue that is needed to take concern is the prototype cross-platform application, which is created by using Cordova, does not have any password restrictions. So anyone can just go in and a different kind of trip and its expenses.

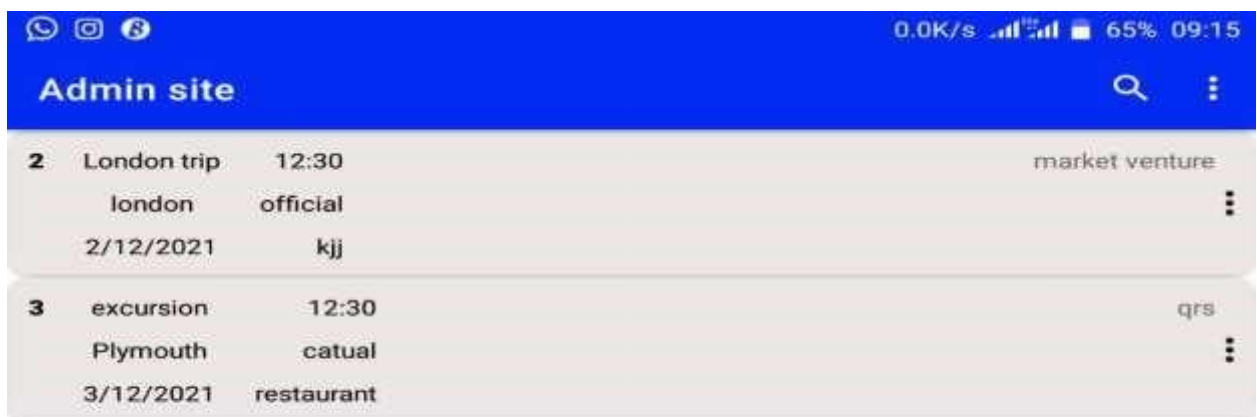
CAPABILITY OF THE APPLICATION TO RUN ON DIFFERENT SCREENS

M-Expense Can adjust to various screen sizes depending on the device as well as it is completely fit for rotated screens. The Only exception is the homepage which fit Well on a rotated screen.

Screenshot



Display on small screen size



Display on a widescreen size

CHANGES TO BE MADE FOR LIVE DEPLOYMENT

The M-Expense Application runs all versions of Android except for Android version 11.

There is some limitations as well with the previous version of Android except Android version 10. The version below Android 8 does not display all the features that are designed in this application. All the features of how to do simplification work perfectly well in Android 10.

Guidelines need to be made on how to use the application, which also includes the and FAQ section and help section so that users access it easily.

- Settings options enable the user to customize the appearance of the application whenever they feel uncomfortable with the current appearance

SECTION 4: DEMONSTRATING IMPLEMENTED FEATURES.

EMPLOYEES ENTER THEY CREDENTIALS THEN LOG IN.



The screenshot shows a mobile application interface for adding a trip. At the top, the status bar displays the time 10:59 AM, signal strength, and battery level at 46%. The app's header is blue with a back arrow and the text 'Add a trip'. Below the header is a dark blue curved banner with the text 'M-EXPENSE' in green. The form consists of several input fields, each with a red 'Required*' label and a character count. The fields are: 'Name of the trip' (0/35), 'Destination' (0/35), 'Date of the trip' (0/35) which includes a 'Requires Risk Assessment' section with 'Yes' and 'No' radio buttons, 'Description of the trip' (0/255) labeled as 'Optional', and 'Time of the trip' (0/10) labeled as 'Required*'. The bottom of the screen shows a grey bar with three navigation icons: a home icon, a circle icon, and a list icon.

10:59 AM

← Add a trip

M-EXPENSE

Name of the trip
Required* 0/35

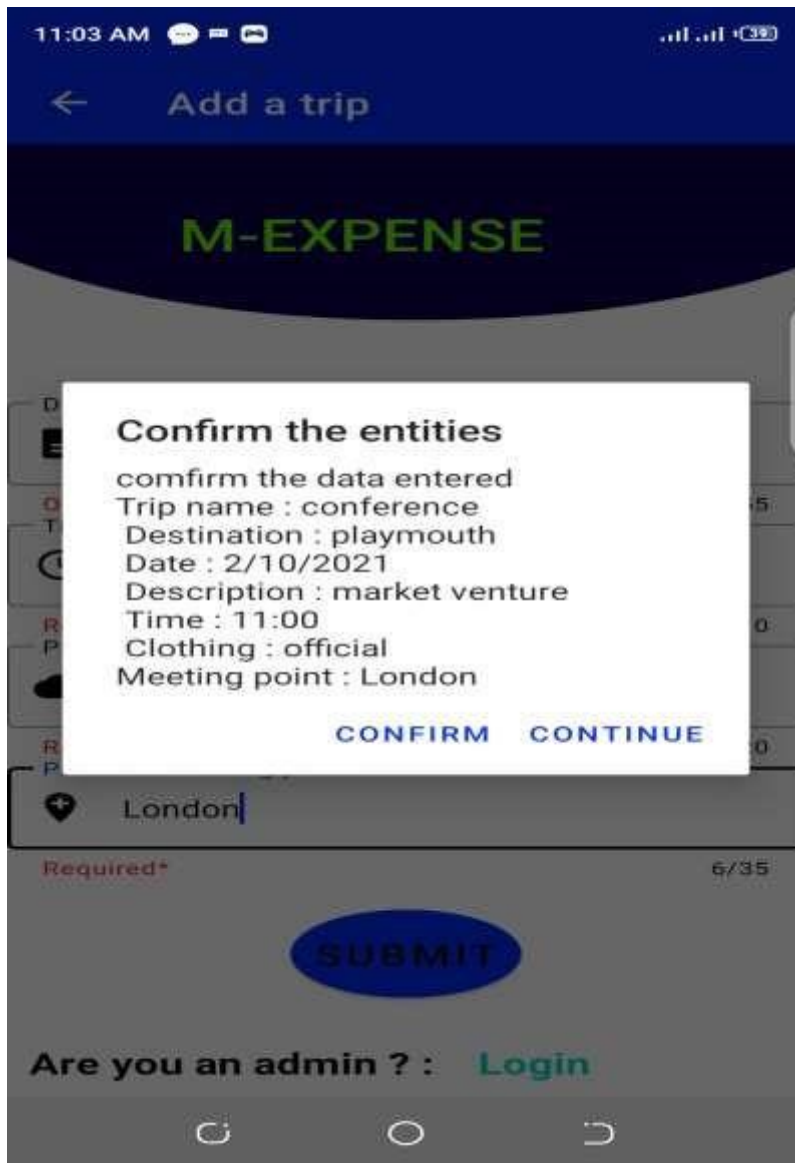
Destination
Required* 0/35

Date of the trip
Required* 0/35
Requires Risk Assessment
☐ Yes ☐ No

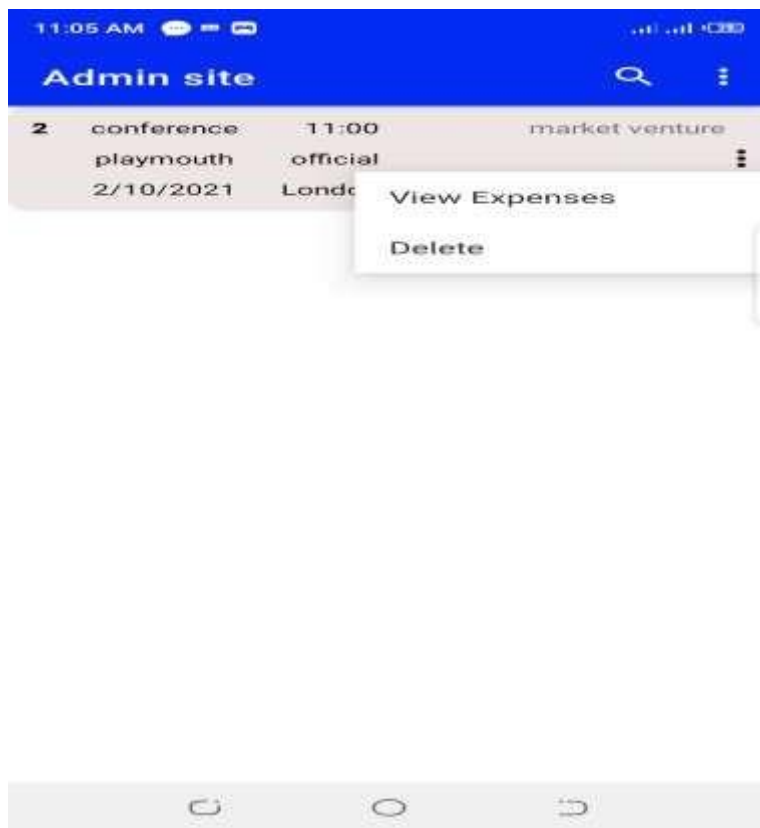
Description of the trip
Optional 0/255

Time of the trip
Required* 0/10


The screenshot above show the implementation of feature “a” It allows the user to enter trip details.



Once the user finishes entering the required field the details are displayed for confirmation before saving as shown above.





As shown in the screenshot above the admin can view details from the database he can also delete the details. Feature "b"



Add Expense


Add Expenses






Name of the trip

Required*0/35




Type of expense

Required*0/80




Amount of expense

Required*0/10



Time of expense

Required*0/10



Additional Comments

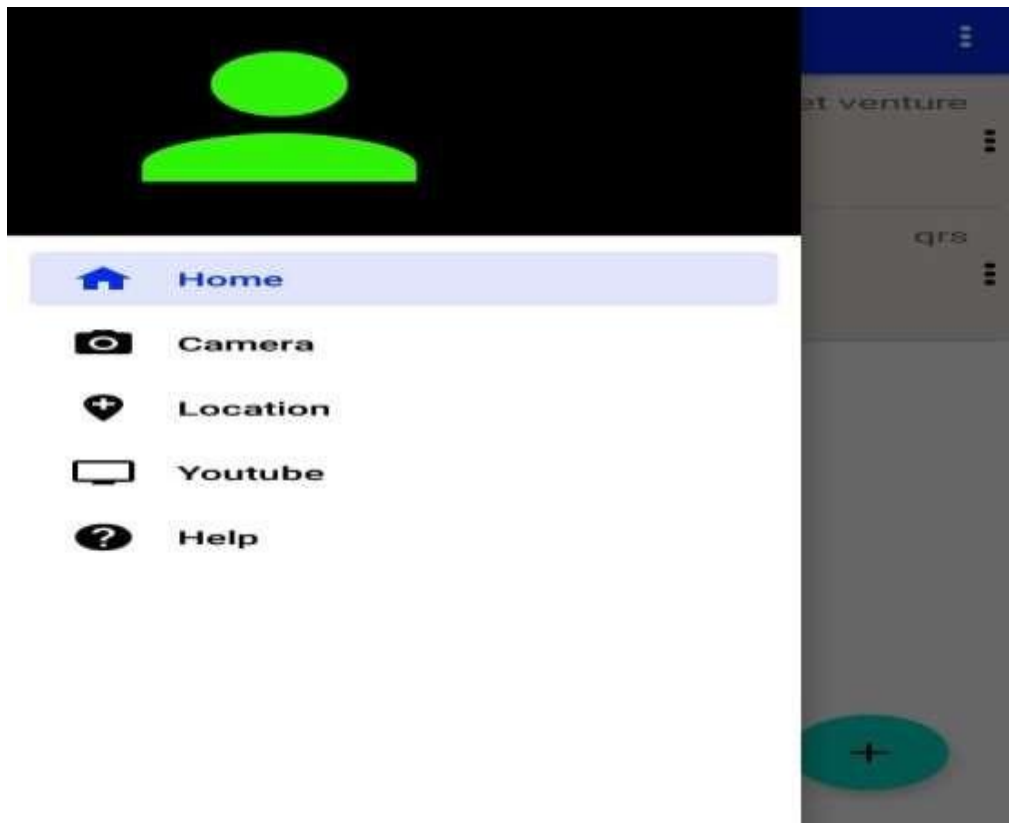
Adding Expenses Feature “c”



The employee can take a picture or choose from gallery Feature h



Details can be edited by clicking the edit option show above.



There is also an option for camera and location

SECTION 4; CODE LISTING:

```
@Override
//Checking user log in credentials and validation
public void onClick(View v) {
    String user = text.getText().toString();
    String pass = text2.getText().toString();    if (user.isEmpty() || pass.isEmpty()){
text.setError("User name required!!");    text.requestFocus();
        text2.setError("Password is required !!");    text2.requestFocus();
    }
    else if(pass.length() < 6 || user.length() < 4){    text.setError("Username less than 4
characters");    text.requestFocus();
        text2.setError("Password is length should be more 6 characters!!");    text2.requestFocus();
    }
    else if (user.equals("other") && !pass.equals("1234567")){    text2.setError("Check Your
Password");    text2.requestFocus();
    }
    else if(!user.equals("other") && pass.equals("1234567")){    text.setError("Check Your
Username");    text.requestFocus();
    }
    else if(!user.equals("other") && !pass.equals("1234567")){    text.setError("Check Your
Username");    text.requestFocus();
        text2.setError("Check Your Password");    text2.requestFocus();
    }
    else if(user.equals("other") && pass.equals("1234567")){    text.setError(null);
text.requestFocus();    text2.setError(null);    text2.requestFocus();    text.setText(null);
text2.setText(null);
        Toast.makeText(LoginUser.this, "Login successfull !!",
Toast.LENGTH_LONG).show();
        Intent intent = new Intent(LoginUser.this, MainActivity2.class);    startActivity(intent);
    }
}
```

```

//Admin login and validation
public void onClick(View v) {
    String user = text.getText().toString();
    String pass = text2.getText().toString();    if (user.isEmpty() || pass.isEmpty()){
text.setError("User name required!!");    text.requestFocus();
        text2.setError("Password is required !!");    text2.requestFocus();    }
    else if(pass.length() < 6 || user.length() < 4){    text.setError("Username less than 4
characters");    text.requestFocus();
        text2.setError("Password is length should be more 6 characters!!");    text2.requestFocus();
    }
    else if(user.equals("expense") && !pass.equals("1234567")){    text2.setError("Check Your
Password");    text2.requestFocus();
    }
    else if(!user.equals("expense") && pass.equals("1234567")){    text.setError("Check Your
Username");    text.requestFocus();
    }
    else if(!user.equals("expense") && !pass.equals("1234567")){    text.setError("Check Your
Username");    text.requestFocus();
        text2.setError("Check Your Password");    text2.requestFocus();
    }
    else if(user.equals("expense") && pass.equals("1234567")){    text.setError(null);
text.requestFocus();    text2.setError(null);    text2.requestFocus();    text.setText("");
text2.setText("");
        Toast.makeText(EmployeeLogin.this, "Login successfull !!",
Toast.LENGTH_LONG).show();
        Intent intent = new Intent(EmployeeLogin.this, Employee.class);    startActivity(intent);

```

```

// Getting trip details from user and notifying user for empty entries
DB = new DBHelper(AddDatabase.this);
String name = editText.getText().toString();
String namedesc = editText1.getText().toString();
String namedate = editText2.getText().toString();
String nameDescription = editText3.getText().toString();
String nameTime = editText4.getText().toString();
String nameCloth = editText5.getText().toString();
String nameMeet = editText6.getText().toString();
if(name.isEmpty() || namedesc.isEmpty() || namedate.isEmpty() || nameTime.isEmpty()
|| nameCloth.isEmpty() || nameMeet.isEmpty()){    editText.setError("Trip name Required!!");
editText.requestFocus();

    editText1.setError("Destination Required!!");    editText1.requestFocus();
    editText2.setError("Date of the trip Required!!");    editText2.requestFocus();
    editText4.setError("Time Required!!");    editText4.requestFocus();
    editText5.setError("Clothing type Required!!");    editText5.requestFocus();
    editText6.setError("Meeting point Required!!");    editText6.requestFocus();

```

@Override

//Adding expenses to trip details and indicating

required fields

```
public void onClick(View v) {
    DB = new DBexpense(AddExpenses.this);
    String name = editText.getText().toString();
    String namedesc = editText1.getText().toString();      String namedate =
editText2.getText().toString();
    if(name.isEmpty() || namedesc.isEmpty() || namedate.isEmpty()){
        editText.setError("Expense type required");      editText.requestFocus();
        editText1.setError("Amount spent required");      editText1.requestFocus();
        editText2.setError("Time spent required");      editText2.requestFocus();
    }
    else {
        editText.setError(null);      editText.requestFocus();      editText1.setError(null);
        editText1.requestFocus();      editText2.setError(null);      editText2.requestFocus();
        AlertDialog.Builder builder = new AlertDialog.Builder(AddExpenses.this);
        builder.setTitle("Confirm the entities");
        builder.setMessage("confirm the data entered\n" +
            "Type : " + editText.getText().toString().trim() + "\n " + "Amount : "
+ editText1.getText().toString().trim() + "\n " + "Time" + editText2.getText().toString().trim() +
            "\n " + "Comment : " + editText3.getText().toString().trim() + "\n
");
        builder.setPositiveButton("Continue", new DialogInterface.OnClickListener() {
            @Override
```

```
@Override
```

//creating sqlite table to store trip details

```
public void onCreate(SQLiteDatabase db) {    String query = "CREATE TABLE " + TABLE1 +
    "(" + COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
    COLUMN_NAME + " TEXT, " + COLUMN_DESTINATION + " TEXT, " +
    COLUMN_DATE + " TEXT, " +
    COLUMN_DESCRIPTION + " TEXT, " + COLUMN_TIME + " TEXT, " +
    COLUMN_CLOTH + " TEXT, " +
    COLUMN_MEET + " TEXT );";    db.execSQL(query);
}
```

I

```
// insert details to Database table
public void insertdaatabase(String Name, String Desc, String Date, String Descr,
String Time, String Cloth, String meet){
    SQLiteDatabase db = this.getWritableDatabase();    ContentValues contentValues = new
ContentValues();
    contentValues.put(COLUMN_NAME, Name);
contentValues.put(COLUMN_DESTINATION, Desc);    contentValues.put(COLUMN_DATE,
Date);
    contentValues.put(COLUMN_DESCRIPTION, Descr);
contentValues.put(COLUMN_TIME, Time);

    contentValues.put(COLUMN_CLOTH, Cloth);    contentValues.put(COLUMN_MEET,
meet);

    long result = db.insert(TABLE1, null, contentValues);    if (result == -1){
        Toast.makeText(context, "Failed", Toast.LENGTH_SHORT).show();    }
}
```

```
// getting data from database table to display
public Cursor getData(){
    String query = "SELECT * FROM " + TABLE1;
    SQLiteDatabase db = this.getReadableDatabase();

    Cursor cursor = null;    if (db != null){
        cursor = db.rawQuery(query, null);
    }

    return cursor;
}
```

// clearing data in the database

```
public void deleteAll(){
    SQLiteDatabase db = this.getWritableDatabase();    db.execSQL("DELETE FROM " +
TABLE1);
}
```

// searching for details from database

```
public Cursor getSearch(String text){
    SQLiteDatabase sqLiteDatabase = this.getReadableDatabase();

    String query = "SELECT * FROM " + TABLE1 + " WHERE " + COLUMN_NAME + " LIKE
'" + text + "%'" + " OR " + COLUMN_DESTINATION + " LIKE '" + text + "%'";

    Cursor cursor = sqLiteDatabase.rawQuery(query, null);    return cursor;
}
```

```

@Override
//Camera image display
public ImageDisplay.ImagesAdapter.ImagesView onCreateView(@NonNull ViewGroup
parent, int viewType) {
    LayoutInflater inflater = LayoutInflater.from(context);
    View view = inflater.inflate(R.layout.imagedisplay, parent, false);
    ImageDisplay.ImagesAdapter.ImagesView(view);
}

@Override
// image display by camera
public void onBindViewHolder(@NonNull ImageDisplay.ImagesAdapter.ImagesView holder, int
position) {
    int images = cursor.getColumnIndex(ImageDatabase.COLUMN_Image);
    cursor.moveToPosition(position); byte[] image1 = cursor.getBlob(images);
    Bitmap bitmap = BitmapFactory.decodeByteArray(image1, 0, image1.length);
    holder.image.setImageBitmap(Bitmap.createScaledBitmap(bitmap, 2000, 2000, false));
}

@Override
public int getItemCount() { if (cursor == null) { return 0; } return cursor.getCount();
}

```

location

```

public class MapsFragment extends Fragment {
// displaying location on a map
    private OnMapReadyCallback callback = new OnMapReadyCallback() {

        @Override
        public void onMapReady(GoogleMap googleMap) {
            LatLng sydney = new LatLng(-
34, 151);
            googleMap.addMarker(new MarkerOptions().position(sydney).title("Marker in Sydney"));
            googleMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
        }
    };

    @Nullable @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
        @Nullable ViewGroup container,
        @Nullable Bundle savedInstanceState) {
inflater.inflate(R.layout.fragment_maps, container, false);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState)
    {
        super.onViewCreated(view, savedInstanceState);
        SupportMapFragment mapFragment = (SupportMapFragment)
getChildFragmentManager().findFragmentById(R.id.map);
        if (mapFragment != null) {
            mapFragment.getMapAsync(callback);
        }
    }
}

```

SECTION 4; CODE LISTING WITHOUT BACKGROUND:

User Log in

```
@Override
```


Checking user log in credentials and validation gets the button id and perform onclick function under view module
public void
onClick(View v) { gets the user inputs

```
String user = text.getText().toString(); String pass =  
text2.getText().toString(); if (user.isEmpty() || pass.isEmpty()){  
text.setError("User name required!!"); text.requestFocus();  
  
text2.setError("Password is required !!"); text2.requestFocus();  
  
}
```

checks the length of the inputs

```
else if (pass.length() < 6 || user.length() < 4){ text.setError("Username less than 4 characters");  
text.requestFocus();  
  
text2.setError("Password is length should be more 6 characters!!"); text2.requestFocus();  
  
}
```

equates the inputs with the defined inputs displays an error incase of amismatch

```
else if (user.equals("other") && !pass.equals("1234567")){ text.setError("Check Your  
Password"); text.requestFocus();  
  
}  
  
else if(!user.equals("other") && pass.equals("1234567")){ text.setError("Check Your  
Username"); text.requestFocus();  
  
}  
  
else if(!user.equals("other") && !pass.equals("1234567")){ text.setError("Check Your  
Username"); text.requestFocus();  
  
text2.setError("Check Your Password"); text2.requestFocus();  
  
}  
  
else if (user.equals("other") && pass.equals("1234567")){ text.setError(null);  
text.requestFocus(); text2.setError(null); text2.requestFocus(); text.setText(null);  
text2.setText(null);
```

displays success in login message when the details match

```
Toast.makeText(LoginUser.this, "Login successfull !!",  
Toast.LENGTH_LONG).show();  
  
Intent intent = new Intent(LoginUser.this, MainActivity2.class); startActivity(intent);  
  
}
```

Admin log in

//Admin login and validation

gets the button id and perform onclick function

```
public void onClick(View v) {  
  
String user = text.getText().toString();  
String pass = text2.getText().toString();
```

Equating the inputs with the defined user name and password to check if they match

```
if (user.isEmpty() || pass.isEmpty()){ text.setError("User name required!!");
text.requestFocus();

text2.setError("Password is required !!"); text2.requestFocus();

}

else if(pass.length() < 6 || user.length() < 4){ text.setError("Username less than 4 characters");
text.requestFocus();

text2.setError("Password is length should be more 6 characters!!"); text2.requestFocus();

}

else if (user.equals("expense") && !pass.equals("1234567")){ text2.setError("Check Your Password"); text2.requestFocus();

}

else if(!user.equals("expense") && pass.equals("1234567")){ text.setError("Check Your Username"); text.requestFocus();

}

else if(!user.equals("expense") && !pass.equals("1234567")){ text.setError("Check Your Username"); text.requestFocus();

text2.setError("Check Your Password"); text2.requestFocus();

}

else if(user.equals("expense") && pass.equals("1234567")){ text.setError(null);
text.requestFocus(); text2.setError(null); text2.requestFocus(); text.setText("");
text2.setText("");

Toast.makeText(EmployeeLogin.this, "Login successfull !!",

Toast.LENGTH_LONG).show();

Intent intent = new Intent(EmployeeLogin.this, Employee.class); startActivity(intent);
```

Trip details entry

//Calls the dbhelper class and set an object DB.

```
= new DBHelper(AddDatabase.this);

//gets the strings entered by the user in addexpensess
```

Gets the strings entered by the user

```
String name = editText.getText().toString();
String namedesc = editText1.getText().toString();
String namedate = editText2.getText().toString();
String nameDescription = editText3.getText().toString();
String nameTime = editText4.getText().toString();
String nameCloth = editText5.getText().toString(); String nameMeet =
editText6.getText().toString();
```

checks the details for verification if there are null fields

the user is notified before uploading the database

```
if (name.isEmpty() || namedesc.isEmpty() || namedate.isEmpty() || nameTime.isEmpty()
|| nameCloth.isEmpty() || nameMeet.isEmpty()){ editText.setError("Trip name
Required!!"); editText.requestFocus();

editText1.setError("Destination Required!!"); editText1.requestFocus();
```

```

editText2.setError("Date of the trip Required!!"); editText2.requestFocus();
editText4.setError("Time Required!!"); editText4.requestFocus();
editText5.setError("Clothing type Required!!"); editText5.requestFocus();
editText6.setError("Meeting point Required!!"); editText6.requestFocus();
empty entr

```

Adding expenses to trip

@Override

Adding expenses to trip details saving them to the database and if any of the fields are empty the user is notified

```

public void onClick(View v) {

    DB = new DBexpense(AddExpenses.this);
    String name = editText.getText().toString();
    String namedesc = editText1.getText().toString();
    String namedate = editText2.getText().toString();

    //checks the inputs fields are empty if empty displays an error if(name.isEmpty() || namedesc.isEmpty() ||
    namedate.isEmpty()){ editText.setError("Expense type required"); editText.requestFocus();

    editText1.setError("Amount spent required"); editText1.requestFocus();
    editText2.setError("Time spent required"); editText2.requestFocus();
} after loading the data to database it sets the input fields
empty

else {
    editText.setError(null);
    editText.requestFocus();
    editText1.setError(null);
    editText1.requestFocus();
    editText2.setError(null);
    editText2.requestFocus();
}

```

AlertDialog fetch the user input and display for confirmation

```

AlertDialog.Builder builder = new AlertDialog.Builder(AddExpenses.this); builder.setTitle("Confirm the entities");

"Type : " + editText.getText().toString().trim() + "\n " + "Amount : "
+ editText1.getText().toString().trim() + " \n " + "Time" +
editText2.getText().toString().trim() +
"\n " + "Comment : " + editText3.getText().toString().trim() + "\n
"); builder.setPositiveButton("Continue", new DialogInterface.OnClickListener() { @Override

```

Creating table for trip details

creating sqlite table to store trip details

```

public void onCreate(SQLiteDatabase db) {
    String query = "CREATE TABLE " + TABLE1 +
    "(" + COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
    COLUMN_NAME + " TEXT, " + COLUMN_DESTINATION + " TEXT, " + COLUMN_DATE + "
    TEXT, " +

```

```

COLUMN_DESCRIPTION + " TEXT, " + COLUMN_TIME + " TEXT, " + COLUMN_CLOTH +
" TEXT, " +
COLUMN_MEET + " TEXT ); "; db.execSQL(query);

}

```

@Override

Inserting details to Table in database

insert details to Database table

```

public void insertdaatabase(String Name, String Desc, String Date, String Descr,
String Time, String Cloth, String meet){ SQLiteDatabase db =
this.getWritableDatabase(); ContentValues contentValues = new
ContentValues();

// contentValues fetch data input from user and add it to given column contentValues.put(COLUMN_NAME, Name);
contentValues.put(COLUMN_DESTINATION, Desc); contentValues.put(COLUMN_DATE, Date);
contentValues.put(COLUMN_DESCRIPTION, Descr); contentValues.put(COLUMN_TIME, Time);

contentValues.put(COLUMN_CLOTH, Cloth);
contentValues.put(COLUMN_MEET, meet);

long result = db.insert(TABLE1, null, contentValues); if (result == -1){
Toast.makeText(context, "Failed", Toast.LENGTH_SHORT).show();
}
}

```

Getting data from data base to display

The display the data

```

public Cursor getData(){
String query = "SELECT * FROM " + TABLE1;
SQLiteDatabase db = this.getReadableDatabase();

Cursor cursor = null; if (db != null){ cursor =
db.rawQuery(query, null);
}

return cursor; }

```

Clearing data from database(deleting)

delete query to delete data from table

```
public void deleteAll(){  
    SQLiteDatabase db = this.getWritableDatabase(); db.execSQL("DELETE  
    FROM " + TABLE1); }
```

// clearing data in the database

Searching data in the database Returns a given row where agiven name exists in the table

```
public Cursor getSearch(String text){  
    SQLiteDatabase sqLiteDatabase = this.getReadableDatabase();  
    String query = "SELECT * FROM " + TABLE1 + " WHERE " + COLUMN_NAME + " LIKE '%" + text + "%'" + " OR " +  
    COLUMN_DESTINATION + " LIKE '%" + text + "%'";  
    Cursor cursor = sqLiteDatabase.rawQuery(query,null); return cursor; }
```

// searching for details from database

Camera image display

@Override

An adapter for fetching images from imagedatabase and display them using recyclerview

```
public ImageDisplay.ImagesAdapter.ImagesView onCreateView(@NonNull ViewGroup  
parent, int viewType) {
```

inflates the imagedisplay with images fetched from the database

```
    LayoutInflater inflater = LayoutInflater.from(context);  
    View view = inflater.inflate(R.layout.imagedisplay, parent, false);  
    return new ImageDisplay.ImagesAdapter.ImagesView(view);  
}
```

@Override

binds the images fetched for display.

```
public void onBindViewHolder(@NonNull ImageDisplay.ImagesAdapter.ImagesView holder,  
int position) {  
    int images = cursor.getColumnIndex(ImageDatabase.COLUMN_Image);  
    cursor.moveToPosition(position);  
    byte[] image1 = cursor.getBlob(images);  
    Bitmap bitmap = BitmapFactory.decodeByteArray(image1, 0, image1.length);  
    holder.image.setImageBitmap(Bitmap.createScaledBitmap(bitmap, 2000, 2000, false));  
}
```

returns the total number of images in the database

@Override

```
public int getItemCount()  
    if (cursor == null) {  
  
        return 0;  
    }  
    return cursor.getCount();  
}
```

Showing location fragment for

displaying maps

```
public class MapsFragment extends Fragment {
```

```
// displaying a map after checking all the permissions private OnMapReadyCallback callback =  
new OnMapReadyCallback() {
```

@Override

displays the map and add the marker to the location

```
public void onMapReady(GoogleMap googleMap) { LatLng sydney = new LatLng(-34, 151);  
googleMap.addMarker(new MarkerOptions().position(sydney).title("Marker in Sydney")); googleMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));  
}  
};
```

@Nullable

@Override

inflates the map to the ui for user interface

```
public View onCreateView(@NonNull LayoutInflater inflater,  
@Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {  
return inflater.inflate(R.layout.fragment_map, container, false);  
}
```

@Override

```
public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState)  
{ super.onViewCreated(view, savedInstanceState); SupportMapFragment  
mapFragment =  
(SupportMapFragment)  
getChildFragmentManager().findFragmentById(R.id.map); if (mapFragment != null) { mapFragment.getMapAsync(callback
```

CHALLENGES ENCOUNTERED WHILE DEVELOPING THE APPLICATION:

During the whole process of developing the android application and the cross prototype version, we faced the following challenges.

THE FREQUENT CRASHING OF THE APPLICATION

As we began developing the application and attempted to run the application after building it and successfully launching it in our target phone, the application could continuously crash as you attempt to open it. At first, trying to figure out what part of our code was causing the application to crash was a problem .as we could check the code, but still, we couldn't see any error.

SOLUTION

We realized that we are trying to implement a lot of features at a go, So we decided to implement one feature at a time so that when the program crashes, we have a small area to inspect for errors, and this really worked out well.

WE HAD LITTLE KNOWLEDGE ON HOW TO IMPLEMENT SOME FEATURES

As we came to the part where we had to implement features such as camera location settings and other data manipulation operations such as clearing the database searching for a record in the database, what we knew at that time was not enough to implement these features to completion.

SOLUTION

We had to check on a few tutorials from youtube and some documentation on SQLite queries. The documentation on SQLite queries helped us manage to implement the search function delete function and clear database function.

THE APPLICATION NOT BEING ABLE TO DISPLAY WELL ON HIGHER ANDROID VERSIONS

In the beginning, we were using android version 7 to test our application, and it was working just fine, but on running the application on android 8,9 and 10. Our text fields and some element could not display at all

SOLUTION

We decided to use android version 10 to be our test device, and we realized that the app could run on lower android versions with small variations.

The cross-platform application built using Cordova took a lot of time to build

We decided to implement one feature at a time to slow.

CONCLUSION:

Developing the Android version of the application Above averagely a success gave more time, we would have implemented more features on the application. The cross-platform version was

slightly below average, But I have gained a lot from this course work. There is a lot of chances to improve the application such as updating the trip via GPS location, pay and update, suggestion of better trip plan and so on.

REFERENCES

1. Android documentation (Accessed 4 December 2021)
2. Cordova documentation google(cordova.org) (Accessed 4 December 2021)
3. Anroid notes for professionals (Accessed 4 December 2021)
4. <https://cordova.apache.org/docs/en/10.x/guide/cli/> (Accessed 4 December 2021)
5. <https://www.toptal.com/mobile/developing-mobileapplications-with-apache-cordova> (Accessed 1 December 2021)
6. [Android Development for Beginners - Full Course - YouTube](#) (Accessed 1 December 2021)
7. [Android App Development in Java All-in-One Tutorial Series \(4 HOURS!\) - YouTube](#) (Accessed 1 December 2021)
8. [Online Courses - Learn Anything, On Your Schedule | Udemy](#) (Accessed 1 December 2021)
9. [Intro to Mobile App Development with Cordova 7 - YouTube](#) (Accessed 1 December 2021)