Mostafa El Sayed

EECE457 – Mobile Applications

AUD Bucket List - Final Project

Dr. Hicham El Zabadani

29/04/2018

AUD Bucket List

1. Introduction

The purpose of this mobile application is to allow users to create an account using an email address and a password, which gets stored on an online database, login to their account, and see a list of their bucket-list items which is also stored online. Each item contains a title, description, map location, and due date. The user can create new items or edit existing ones. Additionally, the user can set an item as completed, which shows a checkmark next to that item. The items are sorted by the due date, and the completed items are moved to the bottom of the list. The application data, which consists of user accounts and bucket list items, is stored online using Google Firebase. The application is bug-proofed against common user mistakes, which are going to be listed in the common-mistakes section below.

2. Instructions

When the application is launched, the user is presented with a login screen, as shown in figure 1. The user is asked to input an email-address and a password and tap login or register, depending on whether they have an account or not. The input credentials are added to the online firebase database or matched against it. The following credentials can be used for testing: *mostafa.elsayed@mymail.aud.edu* *123456*

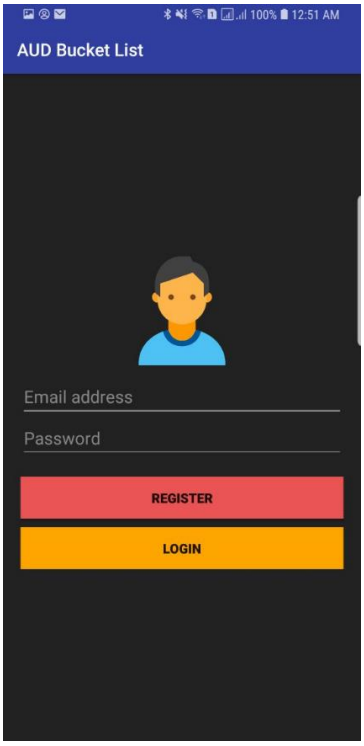

Figure 1 – Main screen

Registering a new account or logging in to an existing one navigates the user to the list of bucket items, which is fetched from the online database. The user is then presented with the list of existing bucket-list items, as shown in figure 2.

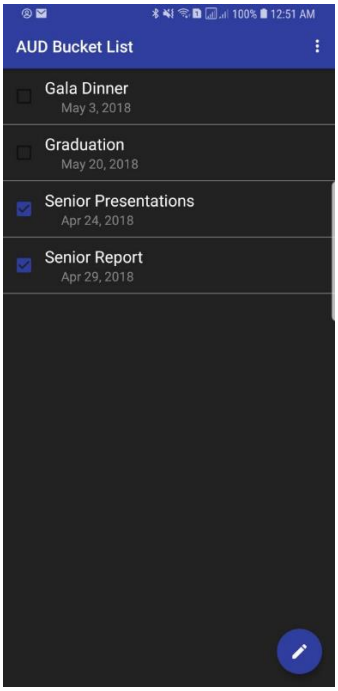

Figure 2 – Bucket-list screen

Tapping on the pencil icon at the bottom right-hand corner navigates the user to the new item page, which allows them to fill information for a new bucket-list item, shown in figure 3. The map location is chosen by simply tapping on the map.

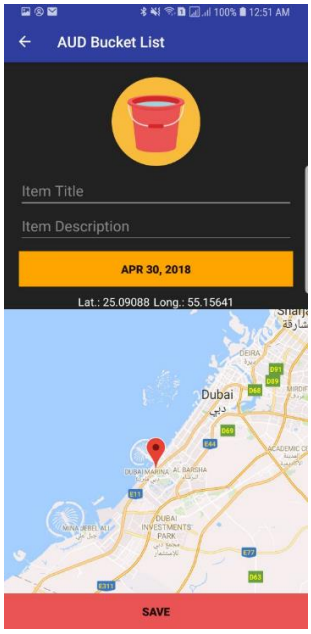

Figure 3 – New item screen

Tapping on an item from the list of bucket-list items navigates the user to the same screen of creating a new item, but it also populates the fields with the values corresponding to the tapped item, as shown in figure 4, allowing the user to edit an existing item.

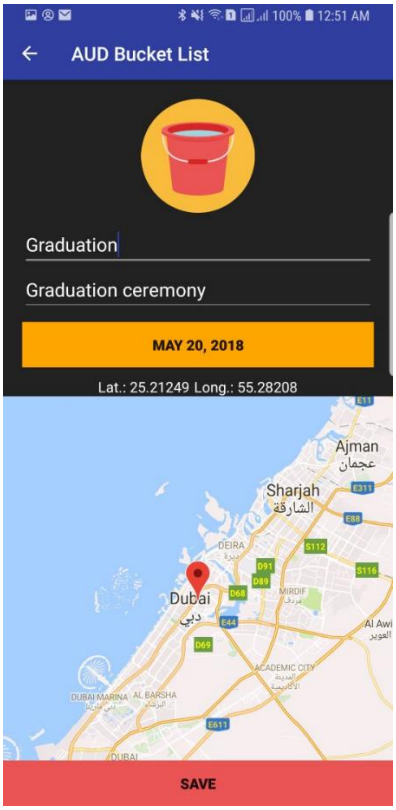

Figure 4 – Edit item screen

3. Bug-proofing

The application is proofed against several bugs. This is done by preventing it from crashing and displaying meaningful error messages, as shown in figure 5. The following bugs have been considered:

- Leaving email or password fields empty while registering or logging in
- Inputting an email in an incorrect format (without “@” or “.”)
- Inputting a password that is shorter than 6 characters, as this is not allowed by Firebase
- Logging in with unknown or wrong credentials
- Registering more than once with the same email-address
- Creating a new item or editing an existing one without inputting a title or description

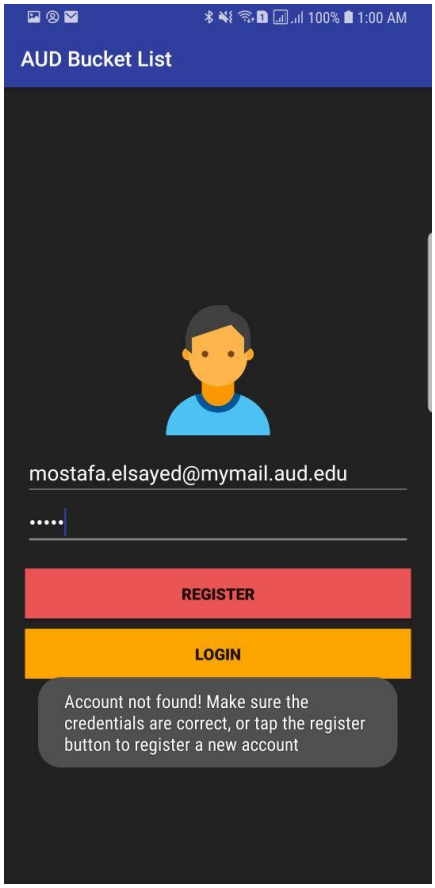

Figure 5 – An example of an error message

4. Lessons Learned

As briefly mentioned above, this mobile application covers a wide range of aspects and features. Those features have to be carefully implemented; meaning that potential mistakes that could be done by the user have to be considered, in order to prevent crashes or unexpected errors as well as to guide the user on what to do, and the implementation code should be as efficient as possible.

The first aspect that had to be considered while designing this application is the design of the user interface. The user interface is a critical element in the design process of any mobile application. It is what the user develops their first impression from, and it should be intuitive and simple to use, in order to avoid confusing the user.

The second major aspect of this project is the usage of Google firebase database. With a few simple steps that consist of downloading a JSON file and importing a few libraries in the application's gradle file, firebase allows us to create a registration and login system with ease, as well as store data online and retrieve it. Firebase uses a tree data structure which requires no SQL knowledge, unlike other database services. This tree data structure allows developers to access and store data in a hierarchical fashion, by accessing children elements of nodes.

The third and final major aspect of this project is the usage of a Google Maps GUI object. A Google Maps object, called fragment, allows developers to display a fully functional map that has all the basic map features that the average user is accustomed to. Those features include smooth scrolling, pinch-zooming, and insertion of pins. The only requirement for having a Google Maps object is an API key which can be obtained for free from the online Google Developer Console. The map object comes with much more sophisticated features that are not utilized in this simple application, such as the display of routes, for instance.

5. Java Code

MainActivity

```
public class MainActivity extends AppCompatActivity
{
    // global variables
    public static FirebaseAuth firebaseAuth;
    public static DatabaseReference databaseReference;

    // GUI elements
    EditText emailEditText, passwordEditText;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // initialize global variables
        firebaseAuth = FirebaseAuth.getInstance();
        databaseReference = (FirebaseDatabase.getInstance()).getReference();
        emailEditText = (EditText) findViewById(R.id.emailEditText);
        passwordEditText = (EditText) findViewById(R.id.passwordEditText);

        // if user is remembered, redirect them to bucket list
        if(firebaseAuth.getCurrentUser() != null)
        {
            startBucketListActivity();
        }
    }

    // GUI functions
    public void registerClicked(View view)
    {
        // ensure fields are not empty
        if (emailEditText.getText().toString().length() == 0)
        {
            Toast.makeText(this, "Please input an email address",
Toast.LENGTH_LONG).show();
            return;
        }
        if (passwordEditText.getText().toString().length() == 0)
        {
            Toast.makeText(this, "Please input a password", Toast.LENGTH_LONG).show();
            return;
        }
```

```
        // ensure fields are correct
        if (!emailEditText.getText().toString().contains("@") ||
!emailEditText.getText().toString().contains("."))
        {
            Toast.makeText(this, "Email address is in an incorrect format",
Toast.LENGTH_LONG).show();
            return;
        }
        if (passwordEditText.getText().toString().length() < 6)
        {
            Toast.makeText(this, "Password must be at least 6 characters long",
Toast.LENGTH_LONG).show();
            return;
        }

        // register
        try
        {
            // attempt to register

firebaseAuth.createUserWithEmailAndPassword(emailEditText.getText().toString(),
passwordEditText.getText().toString())
                    .addOnCompleteListener(this, new OnCompleteListener<AuthResult>()
{
                        @Override
                        public void onComplete(@NonNull Task<AuthResult> task) {
                            if (task.isSuccessful())
                            {
                                startBucketListActivity(); // redirect to bucket list
activity
                            }
                            else
                            {
                                Toast.makeText(MainActivity.this, "An error has
occured. Perhaps an account already exists?", Toast.LENGTH_SHORT).show();
                            }
                        }
                    });
        }
        catch(Exception exc)
        {
            Toast.makeText(this, "An error has occured. Make sure Google Play Services
are updated.", Toast.LENGTH_SHORT).show();
        }
    }
    public void loginClicked(View view)
    {
        // ensure fields are not empty
        if (emailEditText.getText().toString().length() == 0)
        {
            Toast.makeText(this, "Please input an email address",
Toast.LENGTH_LONG).show();
            return;
        }
        if (passwordEditText.getText().toString().length() == 0)
        {
            Toast.makeText(this, "Please input a password", Toast.LENGTH_LONG).show();
            return;
        }


        try
        {
```

```
            // attempt to login

firebaseAuth.signInWithEmailAndPassword(emailEditText.getText().toString(),
passwordEditText.getText().toString()).addOnCompleteListener(this, new
OnCompleteListener<AuthResult>()
            {
                @Override
                public void onComplete(@NonNull Task<AuthResult> task)
                {
                    if (task.isSuccessful())
                    {
                        FirebaseUser user = firebaseAuth.getCurrentUser();
                        startBucketListActivity(); // start bucket list activity
                    }
                    else
                    {
                        Toast.makeText(MainActivity.this, "Account not found! Make
sure the credentials are correct, or tap the register button to register a new
account", Toast.LENGTH_SHORT).show();
                    }
                }
            });
        }
        catch(Exception exc)
        {
            Toast.makeText(this, "An error has occured. Make sure Google Play Services
are updated.", Toast.LENGTH_SHORT).show();
        }
    }

    // Functions
    private  void startBucketListActivity()
    {
        startActivity(new Intent(this, BucketListActivity.class));
        finish();
    }
}
```

BucketListActivity

```
public class BucketListActivity extends AppCompatActivity
{
    // global variables
    ArrayList<BucketItem> listOfBucketItems; // contains bucket items
    CustomAdapter listViewAdapter; // adapter for listview
    public static String userID = "";
    public static BucketItem chosenItem;
    public static boolean isEditing = false;
    // GUI elements
    ListView listView;

    // constructor
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_bucket_list);

        // initialize global variables
        listOfBucketItems = new ArrayList<>();
        listViewAdapter = new CustomAdapter(listOfBucketItems);
```

```
        userID = firebaseAuth.getCurrentUser().getUid();
        listView = (ListView) findViewById(R.id.listView);
        listView.setAdapter(listViewAdapter);

        // create new item button
        FloatingActionButton newItemButton = (FloatingActionButton)
findViewById(R.id.newItemButton);
        newItemButton.setOnClickListener(new View.OnClickListener()
        {
            @Override
            public void onClick(View view)
            {
                isEditing = false;
                startActivity(new Intent(BucketListActivity.this,
AddAndEditItemActivity.class));
            }
        });

        // edit item (when its clicked)
        listView.setOnItemClickListener(new AdapterView.OnItemClickListener()
        {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position,
long id)
            {
                chosenItem = listOfBucketItems.get(position);
                isEditing = true;
                startActivity(new Intent(BucketListActivity.this,
AddAndEditItemActivity.class));
            }
        });

        // set listener for when data changes on firebase
        setDataListener();
    }
    // menu
    @Override
    public boolean onCreateOptionsMenu(Menu menu)
    {
        getMenuInflater().inflate(R.menu.menu, menu);
        return true;
    }
    // menu option selected
    @Override
    public boolean onOptionsItemSelected(MenuItem item)
    {
        if(item.getItemId() == R.id.logout)
        {
            // signout
            FirebaseAuth.getInstance().signOut();
            // navigate to main login page
            startActivity(new Intent(this, MainActivity.class));
            finish();
        }
        return true;
    }


    // FUNCTIONS
    void setDataListener()
    {
        // fires when database items change
        databaseReference.child(userID).child("items").addValueEventListener(new
```

```
ValueEventListener()
        {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot)
            {
                // clear the list
                listOfBucketItems.clear();
                for (DataSnapshot imageSnapshot: dataSnapshot.getChildren()) // loop
over items
                {
                    // get the bucket item
                    BucketItem tempBucketItem =
imageSnapshot.getValue(BucketItem.class);
                    // set key unique key of item
                    tempBucketItem.itemID = imageSnapshot.getKey();
                    // add it to list
                    listOfBucketItems.add(tempBucketItem);
                }
                // sort the list according to date and check-state
                sortList();
                // update listview
                listViewAdapter.notifyDataSetChanged();

                if (listOfBucketItems.size() == 0)
                    Toast.makeText(BucketListActivity.this, "Empty list!",
Toast.LENGTH_LONG).show();
            }
            @Override
            public void onCancelled(DatabaseError databaseError)
            {
                // nothing
            }
        });
    }
    public void sortList()
    {
        // sort list by items due date
        Collections.sort(listOfBucketItems, new Comparator<BucketItem>()
        {
            public int compare(BucketItem itemA, BucketItem itemB)
            {
                if (itemA.dueDate == null || itemB.dueDate == null)
                {
                    return 0;
                }
                else
                {
                    return itemA.dueDate.compareTo(itemB.dueDate);
                }
            }
        });

        // move checked items to the bottom
        for (int i = 0; i < listOfBucketItems.size(); i++)
        {
            for (int j = listOfBucketItems.size()-1; j >= 0; j--)
            {
                if(listOfBucketItems.get(j).checked == true)
                {
                    listOfBucketItems.add(listOfBucketItems.get(j));
                    listOfBucketItems.remove(j);
                }
            }
```

```
        }
    }

    // ListView adapter
    public class CustomAdapter extends BaseAdapter
    {
        private ArrayList<BucketItem> bucketItems;
        public CustomAdapter(ArrayList<BucketItem> bucketItems)
        {
            this.bucketItems = bucketItems;
        }
        @Override
        public int getCount()
        {
            return bucketItems.size();
        }
        @Override
        public Object getItem(int i)
        {
            return null;
        }
        @Override
        public long getItemId(int i)
        {
            return 0;
        }

        @Override
        public View getView(final int i, View view, ViewGroup viewGroup)
        {
            view = getLayoutInflater().inflate(R.layout.mylist, null);
            final CheckBox checkBox = (CheckBox) view.findViewById(R.id.checkBox);
            TextView titleTextView = (TextView) view.findViewById(R.id.titleTextView);
            TextView dateTextView = (TextView) view.findViewById(R.id.dateTextView);

            // detect clicks on checkboxes
            checkBox.setOnClickListener(new View.OnClickListener()
            {
                @Override
                public void onClick(View v)
                {
                    BucketItem item = listOfBucketItems.get(i);
                    // update local bucket items
                    item.checked = checkBox.isChecked();
                    // update on database

databaseReference.child(userID).child("items").child(item.itemID).setValue(item);
                }
            });

            // set GUI attributes
            // Checkbox
            if(bucketItems.get(i).checked)
            {
                checkBox.setChecked(true);
            }
            else
            {
                checkBox.setChecked(false);
            }
            // date
            DateFormat dateFormat = DateFormat.getDateInstance();
            dateTextView.setText(dateFormat.format(bucketItems.get(i).dueDate));
```

```
            // title
            titleTextView.setText(bucketItems.get(i).title);
            return view;
        }
    }

}
```

AddAndEditItemActivity

```
public class AddAndEditItemActivity extends AppCompatActivity implements
OnMapReadyCallback
{
    // global variables
    private GoogleMap mMap;
    private Marker chosenPoint;
    private Date chosenDate = new Date();
    // GUI variables
    private EditText titleEditText, descriptionEditText;
    private TextView latitudeTextView, longitudeTextView;
    private Button dateButton, saveButton;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_and_edit_item);
        // setup toolbar
        Toolbar myToolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(myToolbar);
        getSupportActionBar().setDisplayShowHomeEnabled(true);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        // back clicked
        myToolbar.setNavigationOnClickListener(new View.OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
                finish();
            }
        });

        // initialize global variables
        SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
        titleEditText = (EditText)findViewById(R.id.titleEditText);
        descriptionEditText = (EditText)findViewById(R.id.descriptionEditText);
        latitudeTextView = (TextView)findViewById(R.id.latitudeTextView);
        longitudeTextView = (TextView)findViewById(R.id.longitudeTextView);
        dateButton = (Button) findViewById(R.id.dateButton);
        saveButton = (Button) findViewById(R.id.saveButton);

        // set date button to today's date
        DateFormat dateFormat = DateFormat.getDateInstance();
        dateButton.setText(dateFormat.format(new Date()));

        // populate EditTexts, only if editing
        if(isEditing)
        {
            // title
```

```
            titleEditText.setText(chosenItem.title);
            // description
            descriptionEditText.setText(chosenItem.description);
            // date
            chosenDate = chosenItem.dueDate;
            dateButton.setText(dateFormat.format(chosenDate));
        }
    }
    // fires when Google Maps is ready
    @Override
    public void onMapReady(GoogleMap googleMap)
    {
        // initialize mao instance
        mMap = googleMap;

        // show item location
        LatLng pos;
        if (isEditing)
        {
            pos = new LatLng(chosenItem.latitude, chosenItem.longitude);
        }
        else
        {
            pos = new LatLng(25.090877, 55.156412);
        }
        chosenPoint = mMap.addMarker(new
MarkerOptions().position(pos).title("point"));
        mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(pos, 10));
        // set gui
        longitudeTextView.setText("Long.: " + String.format("%.5f",
chosenPoint.getPosition().longitude));
        latitudeTextView.setText("Lat.: " + String.format("%.5f",
chosenPoint.getPosition().latitude));

        initializeMapClicks();
    }


    // GUI
    public void dateClicked(View view)
    {
        final Calendar calendar = Calendar.getInstance();
        final DateFormat dateFormat = DateFormat.getDateInstance();

        // fires when date is chosen
        final DatePickerDialog.OnDateSetListener datePicker = new
DatePickerDialog.OnDateSetListener()
        {
            @Override
            public void onDateSet(DatePicker view, int year, int month, int
dayOfMonth) // date set
            {
                // set date value
                calendar.set(year, month, dayOfMonth);
                chosenDate = calendar.getTime();
                // update GUI
                dateButton.setText(dateFormat.format(calendar.getTime()));
            }
        };

        // show calendar
        if (isEditing)
        {
```

```
            calendar.setTime(chosenItem.dueDate);
        }

        new DatePickerDialog(AddAndEditItemActivity.this, datePicker,
calendar.get(Calendar.YEAR), calendar.get(Calendar.MONTH),
calendar.get(Calendar.DAY_OF_MONTH)).show();
    }

    public void saveClicked(View view)
    {
        // ensure fields are not empty
        if (titleEditText.getText().toString().length() == 0)
        {
            Toast.makeText(this, "Please input a title", Toast.LENGTH_LONG).show();
            return;
        }
        if (descriptionEditText.getText().toString().length() == 0)
        {
            Toast.makeText(this, "Please input a description",
Toast.LENGTH_LONG).show();
            return;
        }

        // create a new bucket list item
        BucketItem bucketItem = new BucketItem();
        bucketItem.title = titleEditText.getText().toString();
        bucketItem.description = descriptionEditText.getText().toString();
        bucketItem.dueDate = chosenDate;
        bucketItem.checked = false; // default state is false
        bucketItem.latitude = chosenPoint.getPosition().latitude;
        bucketItem.longitude = chosenPoint.getPosition().longitude;

        // create new item or edit existing one
        if(isEditing)
        {
            // edit

databaseReference.child(userID).child("items").child(chosenItem.itemID).setValue(bucke
tItem);
        }
        else
        {
            // create new

databaseReference.child(userID).child("items").push().setValue(bucketItem);
        }
        Toast.makeText(this, "Item saved successfully", Toast.LENGTH_LONG).show();
        finish();
    }

    // FUNCTIONS
    void initializeMapClicks()
    {
        mMap.setOnMapClickListener(new GoogleMap.OnMapClickListener()
        {
            @Override
            public void onMapClick(LatLng point)
            {
                chosenPoint.remove(); // remove previous point
                chosenPoint = mMap.addMarker(new
MarkerOptions().position(point).title("point")); // update point
                // set gui
                longitudeTextView.setText("Long.: " + String.format("%.5f",
```

```
chosenPoint.getPosition().longitude));
                latitudeTextView.setText("Lat.: " + String.format("%.5f",
chosenPoint.getPosition().latitude));
            }
        });
    }
}
```