TECHNICAL MANUAL

AR-T - AUGMENTED REALITY LEARNING APPLICATION FOR TECHNICAL GRAPHICS

STUDENT NAME: RUSSELL BRADY

STUDENT NUMBER: 15534623

SUBMISSION DATE: 19/05/2019

SUPERVISOR: MONICA WARD

TABLE OF CONTENTS

ABSTRACT

AR-T is an Augmented Reality based learning app for Technical Graphics. AR-T is an android application and is targeted at devices Android 7 or higher (Minimum SDK version 24). The project also has a web app to compliment the Android app.

The android app was built using Java in android studio. The REST Api and Web app was built using Node.js and the database is a MySQL database.

The Android app allows students to interact with lessons, quizzes, augmented reality models they have uploaded as well as viewing and uploading assignments in the classroom. Teachers can create numerous classes, assign work in a class, view submissions as well as view individual students progress overall.

The main aim of the app is to act as an educational tool in aiding the learning of the Junior Cycle subject Technical Graphics (TG). TG is an introductory subject into the area of 3-D object representation and it is therefore important that students have the best learning experience they can have, to set them up for the subject in the future.

OVERVIEW

PROJECT MOTIVATION

The idea for this project, and for this application, stemmed from the unsatisfied need for better and more immersive resources for teachers in teaching such an abstract and difficult subject. It is the aim of this app to help students gain a greater grasp of the basic concepts and a greater understanding of 3-D object representation.

3-D object visualization and representation are very difficult concepts, and this is reiterated by [3]. Many students starting the subject T.G. struggle with the main concepts and this application aims to help solve this problem. The app aims to help students with the basic concepts and make learning the subject and its content easier and more immersive.

Augmented reality is a fantastic way to express the subject content in a new and immersive way [1]. This view was expressed in multiple researched papers which were reviewed as part of the conducted research in the area. This is expanded on in further detail throughout the document.

PLATFORM DEVELOPMENT MOTIVATION

The main motivation to undertake a project involving augmented reality was that it is a very new and emerging field. As of yet, there are no clear educational tools in the subject which implement such technology and as such this project is the first of its kind in the subject.

It was decided that Android would be used as the main platform due to previous experience using it. Having prior experience meant more emphasis could be put on learning and implementing the augmented reality than on the platform it would be delivered on.

Node.js is implemented as the REST Api along with a MySQL database. Before this project the developer had no prior experience developing in Javascript or using Node.js. The reasoning for choosing this technology is further explained in the subsequent section.

RESEARCH CONDUCTED

Research was conducted on the appropriate AR development kit, system components and what had already been done in this area in the past. These are detailed below.

AUGMENTED REALITY SOFTWARE DEVELOPMENT KIT

The primary research at the start of the project was on what development kit would implemented in the project. The main development kits which were possibilities were Wikitude, Vuforia, ARCore, ARKit and EasyAR.

ARKit was appealing as it is a very well-known development kit but is aimed at iOS, so it was not suitable as Android was the chosen platform. Wikitude, Vuforia and EasyAR were all options however they were not open source and so most of the functionality they offered were not free. As well as this, they included watermarks on their software which was not appealing.

ARCore was a suitable option as it is completely open source and was created by Google. The main problem with this development kit is that it was only out of Beta development one year and is still mainly in development. ARCore does have the capability of being used cross platform between iOS and Android so further expansion onto iOS would be possible with this SDK.

ARCore was the development kit that was chosen. This meant that the augmented reality software could be developed in android directly alongside the other android code.

DATABASE & REST API

There was a decision which needed to be made on what database would be most suitable for the project. The main two options were Firebase and MySQL. There were several pros and cons to weigh up on both sides in deciding.

Firebase was going to be the easier to set up due to its simple implementation in android apps. Although this was the case, one of the advantages of a MySQL database was having complete control of the data and the database and not being dependent on a third-party service to manage the apps data. The main downside to going with a MySQL database was that a REST Api would need to be implemented to make calls to the database from the android app, whereas all of this is handled by Firebase itself.

There was also one major problem with developing using Firebase – all the calls made to the database are made on the client side. This means all the server logic would be on the mobile device which is not

advantageous as it leads to tighter coupling between the client and the database. For this reason, it was not good practice for the mobile app to be handling that type of logic.

The decision was made to implement the MySQL database. This meant a REST Api was going to need to be created. This created a layer of abstraction between the client and the database. Node.js was the chosen framework for this as it seemed the perfect fit for what was needed and there was plenty of documentation on linking it up with MySQL and on deployment.

WHAT RESEARCH HAS ALREADY BEEN DONE?

Once the decision was made to pursue the use of augmented reality, the next step was to conduct research on how AR could be best utilized in providing a great learning experience to students. It was important to find out what work had been done previously using augmented reality in the area.

Several research papers dealt directly with the subject and the possible uses of AR in its teaching. These articles detailed the possible benefits of AR and how it could be used to help students grasp the subject, as many students struggle with the concepts and 3-D elements at first.

The papers researched can be found in Appendix 1. These papers were very useful in providing more information on the pedagogy element of teaching of the subject using AR as well as reaffirming that augmented reality can provide a great solution to helping teach concepts of the subject.

The main theme running through the papers was that there are large deficiencies in terms of student's ability to visualize three-dimensional models shown in two dimensions. Also, it was stated that it is difficult to illustrate the relationship between the 3D geometry and 2D projection using only drawings on blackboard and books. Augmented reality was stated as a new way to show these concepts.

Based on studies of visualization skills of entering first year engineering students, some of them have difficulties in dealing with orthographic applications (projections, orthographic to isometric transformations, etc.). They seem to lack enough geometric and/or trigonometric relational skills, both of which are essential when modeling even simple geometric objects. Augmented Reality (AR) technology could provide a solution to this problem [2].

These papers have been extremely useful in helping formulate the ideas implemented in the project and in making the intended outcome of the app clearer.

USER ROLES & OBJECTIVES

The AR-T application divides users of the system into two categories.

STUDENT

Students have various features available to them in the app that are outlined here. There are lessons available on the app which teach sections of the technical graphics course content. These lessons are short and aimed to ensure to keep the student engaged. There are also quizzes based on these lessons to reinforce the learning. Both the lessons and quizzes are conducted through augmented reality. Together, these provide an enjoyable learning experience.

One of the apps features allows students to upload 3D models they have created using CAD software onto the web app and then view the model in augmented reality on the android app. Students are spending more time using CAD software in the subject with the introduction of a new curriculum, so this is a great feature to have available to them.

The augmented images feature in the app is intended to act as a supplementary resource to a textbook for the subject. This feature detects images which have been predefined and renders augmented reality models / video in augmented reality over the image. This means that if the app was linked with a book in the subject, images in the book could be defined in the app to show content in augmented reality. The aim of this feature is to make the teaching of the subject more engaging and interesting than reading straight from a book.

These features can be seen in the user manual which includes screenshots.

The objective from the students' point of view is to have an app which is fun to use and helps them to understand course content in an interactive manner. It was important that the app utilize learning styles which are not attainable by a book or normal online resource, otherwise it holds no advantage.

TEACHER

As mentioned in the abstract, teachers can create numerous classes, assign work in a class, view submissions as well as view individual students progress overall.

The teacher can create a class with a class name and ID. Once they have created the class, students can register in the app using the class ID provided by the teacher. The teacher can also create assignments in the class for the students to complete. These assignments have a start and end date and include a description of the work to be completed. This work can be seen on the student's side. When the students upload submissions, they can be viewed by the teacher and graded.

The teacher may also view the individual progress of a student, looking at the lessons and quizzes completed by the student as well as all the students completed assignments. These metrics give a good indication as to how the student is progressing in the class.

For greater details on these user stories check the user manual.

The objectives of the system from the teachers' point of view is to have a resource which can be used to keep track of students' progress in terms of learning course content and being able to assign work but also to partake in the students learning through augmented reality. The application caters for this.

GLOSSARY

Augmented Reality (AR): A technology that superimposes a computer-generated image on a user's view of the real world, thus providing a composite view.

ARCore: ARCore is a software development kit developed by Google that allows for augmented reality applications to be built.

App: An application, especially as downloaded by a user to a mobile device. The app will refer to AR-T in this case.

Technical Graphics: Subject where you will learn how to represent 3-D objects on paper and on computer, helping you to develop problem solving and creative thinking skills through the solution of graphical problems.

Cloud Anchor: Cloud anchors are a feature provided by Google to AR apps where you can host your AR object. Somebody on a different device can then also look at your hosted AR object by using a provided Api Key. This will be utilized and implemented in AR-T.

Gradle: Gradle is an open-source build automation system that builds upon the concepts of Apache Ant and Apache Maven and introduces a Groovy-based domain-specific language instead of the XML form used by Apache Maven for declaring the project configuration.

REST Api: A REST API defines a set of functions which developers can perform requests and receive responses via HTTP protocol such as GET and POST.

Continuous Integration (CI): Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day.

CI/CD: In software engineering, CI/CD or CICD may refer to the combined practices of continuous integration and continuous delivery.

CAD: Computer aided design. It is the use of computers to aid in the creation, modification, analysis, or optimization of a design.

Routing: Routing refers to how an application's endpoints (URIs) respond to client requests.

DESIGN

The application contains the requirements outlined in the functional specification however the requirements have evolved and adapted which is natural in any software development lifecycle process.

SYSTEM ARCHITECTURE

The system architecture involves a few key components. These components can be seen in the Component Diagram below. They consist of the following:

ANDROID APP

The android app is the primary component in the project. The app was built using Java in Android Studio and links to the REST Api via Retrofit HTTP library. Retrofit is a REST Client for Android and Java

by Square. It makes it relatively easy to retrieve and upload JSON (or other structured data) via a REST based webservice. In Retrofit you configure which converter is used for the data serialization. The android app uses ARCore as it's AR development kit.

NODE.JS SERVER

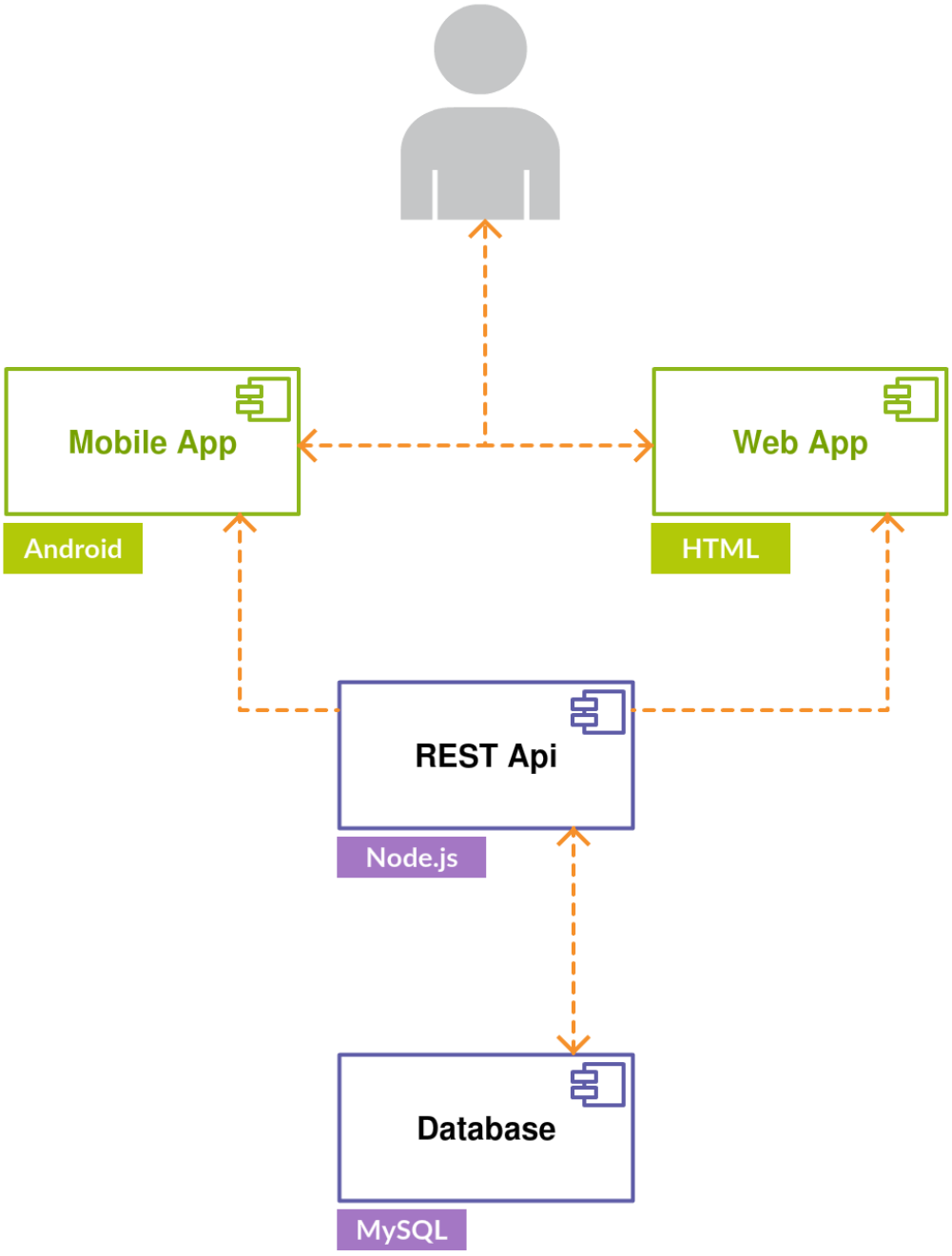
The Node.js server has two main functions. The first is to provide a REST Api to the android app so that it can access the database. The second is that it serves the frontend UI to the web application. This component acts as an endpoint to access data from the database.

WEB APP

The web app is a Node.js app which runs along with the REST Api. This is only a small component as it provides some very specific functionality which the Android app can use.

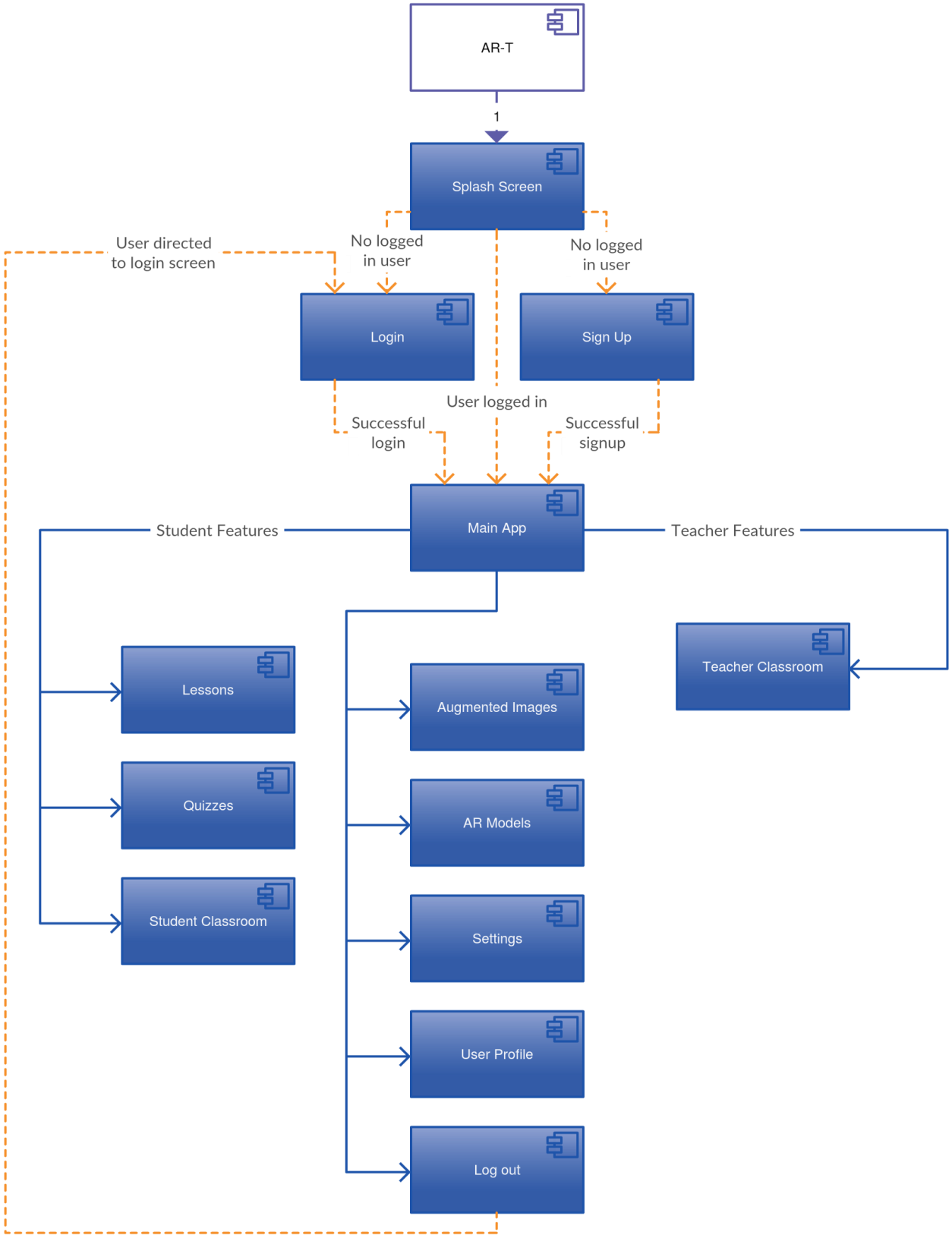
MYSQL DATABASE

The MySQL database is the projects primary store of data. All the app and user's data are stored here and can be accessed via the REST Api. The database is in 3rd Normal Form.



HIGH LEVEL DESIGN

The diagram below illustrates the high-level design of the android applications functionality. On entering the app, a splash screen is displayed. If there is a user logged in then they are directed to the main app, otherwise the app directs to the login/signup screen. Inside the main app the user will have access to different features depending on whether the user is a student or teacher, however there are features which are common to both. On logging out, the app directs back to the login screen.


AR-T
1
Splash Screen
No logged in user
No logged in user
User directed to login screen
Login
Sign Up
User logged in
Successful login
Successful signup
Main App
Student Features
Teacher Features
Lessons
Quizzes
Student Classroom
Augmented Images
AR Models
Settings
User Profile
Log out
Teacher Classroom

MVVM DESIGN PATTERN

AR-T Android app was designed with the MVVM design pattern as this is currently very popular and widely adopted among the Android community.

The MVVM architecture is described in the function specification but as a quick recap here are the main components:

Model	The model represents the data of the app. Local models will 'model' the data in the database.
View	It is the presentation layer for the model and is essentially what the user will see when using the device (Activities / Fragments).
ViewModel	The ViewModel exposes streams of data relevant to the View. The ViewModel retrieves the necessary data from the Model, applies the UI logic and then exposes it to the View for it to consume.

This pattern was followed throughout the development of the app and it led to very clean and coherent design.

Java Objects are created which model the data in the database. The android app interacts with the database using Retrofit HTTP Library and when REST calls are made, the responses which are returned are cast into these Java 'Models' which have been created on the client side.

Once this response is returned, if there is UI logic which needs to be carried out before it is displayed then this is carried out by the ViewModel. The idea is to try and extract as much business logic away from the views as possible. Otherwise if there was no logic needed, the data was passed straight to the views where it was handled and displayed.

In practice it was found that in some cases the ViewModel could not be used. There were two main reasons for this.

- There was no logic needed – In many cases in the app, calls made to the REST Api simply returned the data which needed to be displayed on the screen without carrying out any prior UI logic.
- Logic had to be in the Views – This was a problem when creating the Views which were dealing with the augmented reality software. All the logic for creating the augmented reality had to be in the view and could not be extracted to the ViewModel. This was not a problem as even if it could have been extracted to the ViewModel it would not have been unit testable anyway.

DATABASE DESIGN

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties.

By defining the entities, their attributes, and showing the relationships between them, the ER diagram illustrates the logical structure of the database. The database was designed to follow third normal form (3NF). Third normal form (3NF) is a database principle that supports the integrity of data by building upon the database normalization principles provided by first normal form (1NF) and second normal form (2NF). The purpose of 3NF is to improve database processing while also minimizing storage costs.

The database followed first normal form by complying with these rules:

- No duplicative columns in the same table.
- Separate tables for each group of related data and identify each row with a unique column (the primary key).

The database followed second normal form by complying with these rules:

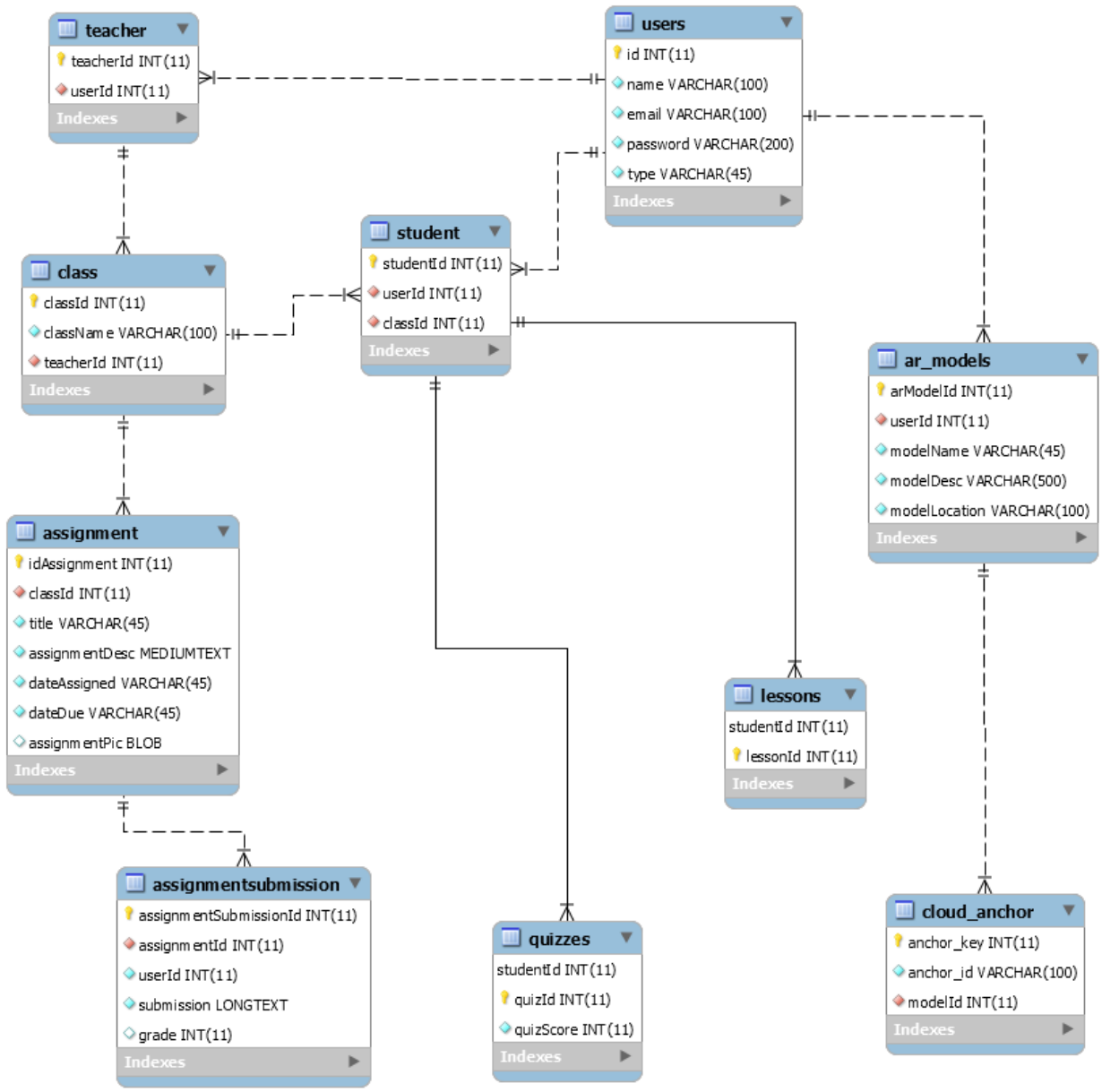
- Remove subsets of data that apply to multiple rows of a table and place them in separate tables.
- Create relationships between these new tables and their predecessors using foreign keys.

The database followed third normal form by complying with this rule:

- All database columns must depend on the primary key, meaning that any column's value can be derived from the primary key only.

Below is the ER Diagram for the AR-T application. This diagram was generated directly from the MySQL database using MySQL Workbench.

Taking the User's table as an example. This table can have many teachers and many students. Each User can have many augmented reality models. A Teacher can have many classes and each class can have many students. The relationships between these entities can be seen to be managed by foreign keys in the tables.



UI DESIGN

Android users expect their apps to look and behave in a way that is consistent with the platform. Not only were material design guidelines followed for visual and navigation patterns, but quality guidelines were also followed for compatibility and more.

In designing the application, Google's material design guidelines were implemented which is the standard for Android mobile development. This helped in designing the application theme colors as well as the layout, views and animations. Following these guidelines and principles helped to create an app which followed standard Android UI design.

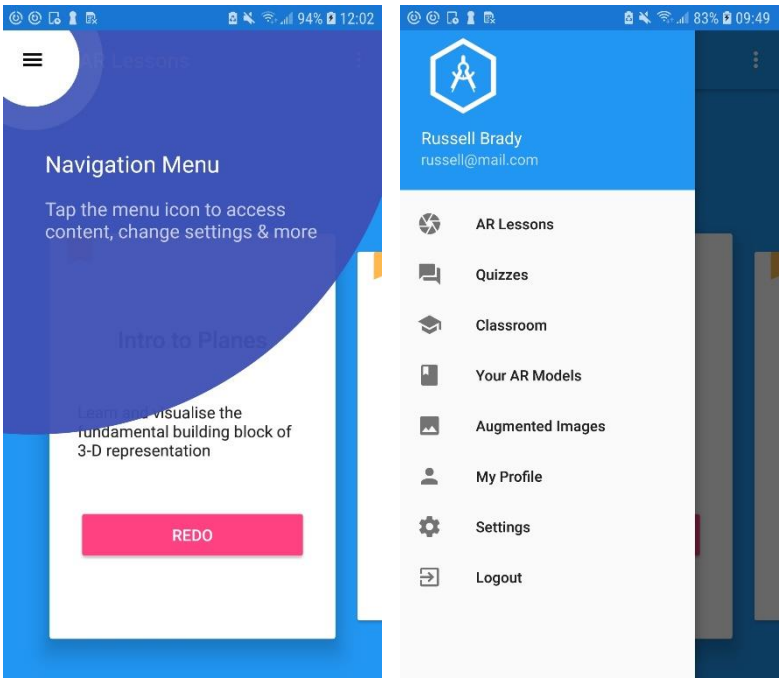
Below are three high-level principles which form part of Google's design philosophy. These were followed these over the course of the project to direct me in designing the app:

1. **Enchant Me** – Generally concerns appearance
2. **Simplify My Life** – Means making apps that are easy and intuitive to use
3. **Make Me Amazing** – Is mostly for giving people ownership and making them feel capable of using all the app's functions

These are high-level objectives, but there are some key design principles within them that every Android developer should implement. Here are some that were followed in the project:

ONLY SHOW WHAT I NEED WHEN I NEED IT

This involves creating apps that are clean and uncluttered. One way to do this is to avoid crowding the screen with lots of options, links and menu items. These options and features should all be available to the user, but only when the user decides they want them. An example of this is in the app is putting three vertical dots in the corner of the screen to show the user that there is a menu. When the user wants to access the settings screen, they can do so through the menu when they tap on it. When they do not want to use it, the menu stays out of their way. Similarly, there is a navigation drawer which can be accessed in the top left of the app, which can navigate the user to all other places in the app. The drawer only opens when it is requested to do so. The navigation drawer is highlighted in the left screenshot below by the showcase view. On clicking this hamburger button, the navigation drawer is opened.

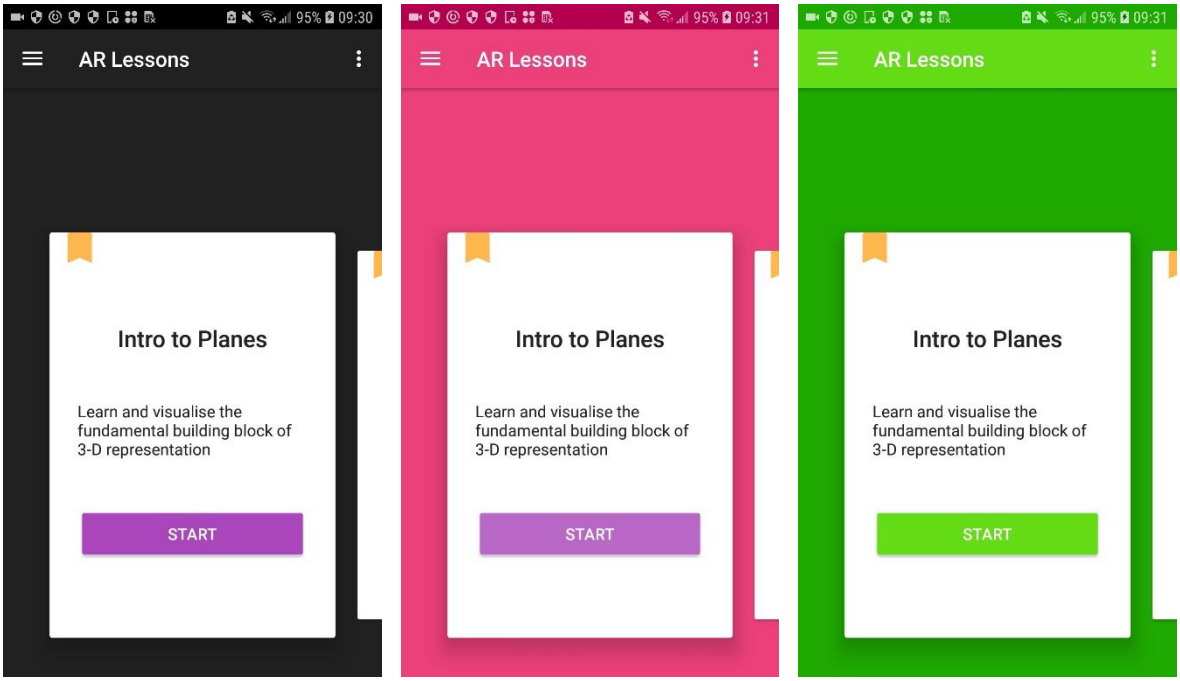


MAKE IMPORTANT THINGS FAST

In an app that does several things (For two different user types) the most important function on the app should be made available to the users the quickest. For example, in the case of the student, the augmented reality lessons are the most important feature and therefore, this should be prominent so that it is easy to find and easy to use. This feature is the first screen that appears on opening the app as a student. Similarly, for the teacher, the classroom is most important, so this is the screen that opens on launching the app.

LET ME MAKE IT MINE

People love to add personal touches because it helps them feel at home and in control. In the augmented reality features of the app it was important to give users as much freedom as possible to explore as opposed to giving constant direction as to what to do. Features like the ability to customize the app theme colors also allow for a personal touch. Individual users have different preferences, so it is important to cater for this.

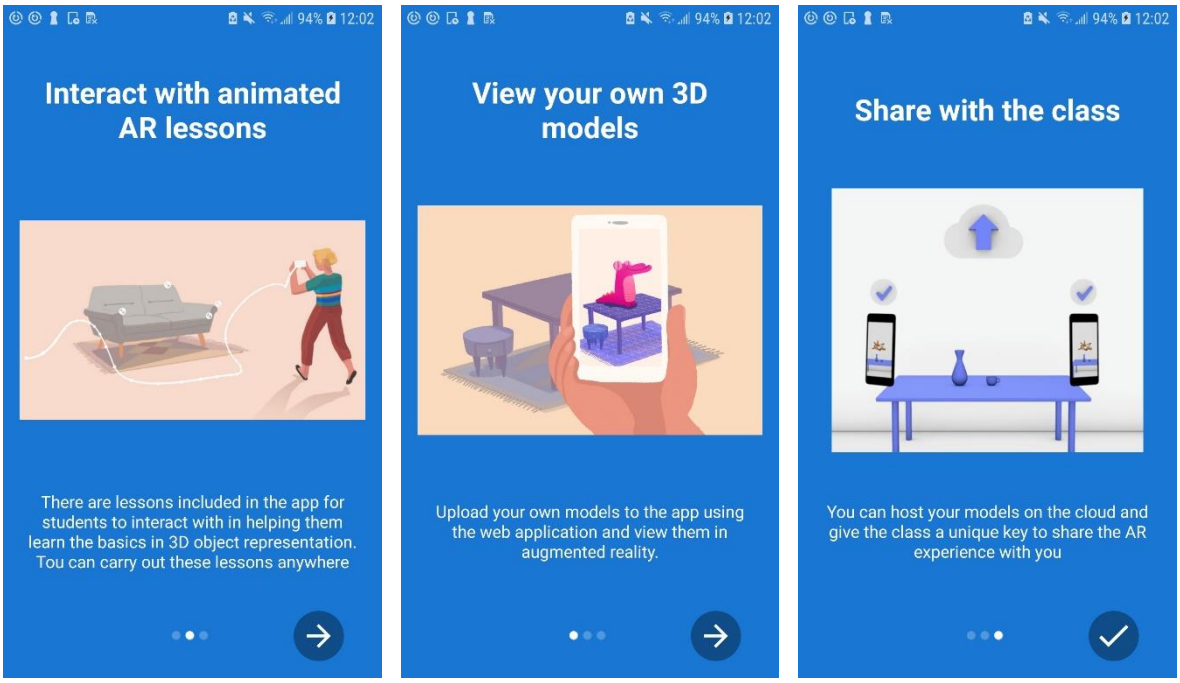


I SHOULD ALWAYS KNOW WHERE I AM

Give people confidence that they know their way around. The idea here is to make places in your app look distinct and use transitions to show relationships among screens. Provide feedback on tasks in progress. This was implemented in the app in several ways. A Toolbar at the top of the screen which displays the title of the current screen informs the user as to where they are in the app. The app implements a back stack which allows for proper back navigation through all the previous screens which were visited. When information is being downloaded / uploaded from the internet, a Content Loading Progress Bar is shown to provide users with feedback that a task is in progress.

PICTURES ARE FASTER THAN WORDS

Using pictures to explain ideas. They get people's attention and can be much more efficient than words. An example of this is in the Guided Tour, where pictures are used to express the different features which are available to the user in the app.



IMPLEMENTATION

The implementation of this platform's functionality is spread across three main modules, the android app, node.js REST Api and the web app. This section will detail some of the main functionality from all three modules.

ANDROID APPLICATION

There are several main features in the android app which will be explained below using code snippets.

SHOW AUGMENTED REALITY MODEL

This feature allows students and teachers to view 3D models they have uploaded to the web in augmented reality and share them with the rest of the class.

The piece of code below is a function in the activity which runs when the activity is created. The location of the model on the server to be shown in augmented reality is passed into the activity and saved as the private variable modelLocation. In this function a 'Renderable' is being created which is a model that can be shown in augmented reality. This model is setting its source to be the model at modelLocation. If it successfully creates the renderable, then it sets the result to the private variable renderable.

```
private void buildModel() {
    if (modelLocation != null) {
        ModelRenderable.builder()
                .setSource( context: this, RenderableSource.builder().setSource(
                        context: this,
                        Uri.parse(modelLocation),
                        RenderableSource.SourceType.GLB)
                        .build())
                .setRegistryId(modelLocation)
                .build()
                .thenAccept(renderable -> this.renderable = renderable)
                .exceptionally(
                        throwable -> {
                            showToast("Unable to load renderable" + modelLocation);
                            return null;
                        });
    }
}
```

When the activity is created a listener is also created on arFragment, which is the augmented reality scene view which appears on the screen, so that when the user taps on a point on the screen an anchor is created to hold the augmented reality model at that point. This can be seen in the snippet below where an anchor is created at the hitResult.

```
arFragment.setOnTapArPlaneListener(
    (HitResult hitResult, Plane plane, MotionEvent motionEvent) -> {
        if (renderable == null) {
            showToast(getString(R.string.please_resolve));
            return;
        } else if (modelLocation == null){
            showToast(getString(R.string.resolve_anchor_toast));
        }
        isPlaced = true;
        arFragment.getPlaneDiscoveryController().hide();
        // Create the Anchor.
        Anchor anchor = hitResult.createAnchor();
        setCloudAnchor(anchor);
        createAnchor();
    });
```

When the anchor has been created, it is set to the private variable cloudAnchor in the setCloudAnchor() function. Once this is done the scene is finally created in the function below. An 'AnchorNode' is created at the point in the scene that the user tapped on. A new 'TransformableNode' is created as a child of the anchorNode. This node is a node which can be rotated and scaled. The renderable is set to this node and following this, the augmented reality object appears in the scene.

```
private void createAnchor() {
    AnchorNode anchorNode = new AnchorNode(cloudAnchor);
    arFragment.getArSceneView().getScene().addChild(anchorNode);
    transformableNode = new TransformableNode(arFragment.getTransformationSystem());
    transformableNode.setRenderable(renderable);
    // Set the min and max scales of the ScaleController.
    // Default min is 0.75, default max is 1.75.
    transformableNode.getScaleController().setMinScale(0.4f);
    transformableNode.getScaleController().setMaxScale(2.0f);
    // Set the local scale of the transformableNode BEFORE setting its parent
    transformableNode.setLocalScale(new Vector3( v: 0.55f,  v1: 0.55f,  v2: 0.55f));
    transformableNode.setParent(anchorNode);

    CustomNode titleView = new CustomNode();
    titleView.setParent(transformableNode);
    titleView.setRenderable(solarControlsRenderable);
    titleView.setLocalPosition(new Vector3( v: 0.0f,  v1: 1.5f,  v2: 0.0f));
```

As an example, in the image below, the user taps on the screen and the steps outlined above occur causing the cube to be rendered at the point where the user tapped on the screen.

For a student to share their model with the class, they need to click on the upload button which can be seen in the screenshot above. Once the upload is completed, an anchor id for the model will be returned. This anchor id is a reference to the model and its location in 3-D space.

Other classmates can resolve this anchor id given to them and once it is resolved, the object will be placed on the screen for them to view.

AUGMENTED IMAGES FEATURE

This feature allows predefined 3D models and video to be displayed in augmented reality when the camera detects a certain image in the scene. The following code describes how some of this feature was implemented.

The first thing which is done is that in the AugmentedImagesFragment, the image database is set up with the images that need to be detected in the scene. These images are defined as the keys in augmentedImageItemMap. Each image is added to the augmentedImageDatabase and this database is then set to the scene. In each frame, the fragment checks for images which are contained in the database in the scene.

```
private boolean setupAugmentedImageDatabase(Config config, Session session) {
    AugmentedImageDatabase augmentedImageDatabase;

    AssetManager assetManager = getContext() != null ? getContext().getAssets() : null;
    if (assetManager == null) {
        Log.e(TAG, msg: "Context is null, cannot initialize image database.");
        return false;
    }

    augmentedImageDatabase = new AugmentedImageDatabase(session);

    for (String item : augmentedImageItemMap.keySet()) {
        Bitmap augmentedImageBitmap = loadAugmentedImage(item);
        if (augmentedImageBitmap == null) {
            return false;
        }
        augmentedImageDatabase.addImage(item,augmentedImageBitmap);
    }

    config.setAugmentedImageDatabase(augmentedImageDatabase);
    config.setUpdateMode(Config.UpdateMode.LATEST_CAMERA_IMAGE);
    return true;
}
```

The augmentedImagesFragment is created in the AugmentedImagesActivity. In this activity code is implemented to create video and 3D models in augmented reality when an image in the scene has been detected. A small part of the code used to implement this is shown below. This code is run every frame to detect new images.

```
if (!augmentedImageModelNodeHashMap.containsKey(augmentedImage) && !augmentedImageVideoNodeMap.containsKey(augmentedImage)) {
    AugmentedImageItem item = AugmentedImagesFragment.augmentedImageItemMap.get(augmentedImage.getName());
    if (item.getType() == AugmentedImagesFragment.CONTENT_TYPE.MODEL) {
        AugmentedImageModelNode node = new AugmentedImageModelNode(getApplicationContext(), item, infoCard.makeCopy());
        node.setImage(augmentedImage);
        augmentedImageModelNodeHashMap.put(augmentedImage, node);
        arFragment.getArSceneView().getScene().addChild(node);
    } else if (item.getType() == AugmentedImagesFragment.CONTENT_TYPE.VIDEO) {
        AugmentedImageVideoNode node = new AugmentedImageVideoNode(getApplicationContext(), item, augmentedImage);
        augmentedImageVideoNodeMap.put(augmentedImage, node);
        arFragment.getArSceneView().getScene().addChild(node);
    }
    fab.setVisibility(View.VISIBLE);
    Toast.makeText( context: this, text: "Image detected", Toast.LENGTH_LONG).show();
}
```

In the activity two different hash maps (one for video and one for models) are used to keep track of all the images which have already been detected in the scene. If the detected 'augmented image' is already in either hashmap then the image is not re-added, otherwise the item associated with that

image is retrieved and an augmented reality node is created to hold the item, whether the item type is a model or video. This node is added to the scene and the model/video appears in augmented reality.

RETROFIT REST CLIENT

Retrofit is a REST Client for Android and Java by Square. It makes it relatively easy to retrieve and upload JSON (or other structured data) via a REST based webservice. In Retrofit the app configures which converter is used for the data serialization. The app has implemented Retrofit in the application through two main classes (ApiClient & ApiUtils) and one main interface (ApiInterface).

ApiClient sets up the Retrofit client by providing the BASE_URL of the REST Api. The HTTP client is set up with this URL in a static method which returns the retrofit instance.

ApiInterface defines all the POST and GET routes of the HTTP client. These can be seen below. All these routes map to routes on the REST Api side. These functions return a call which includes some response. Each function defines the expected response as a Java object where the fields of the java object map to the fields in the response.

ApiUtils has a static method getApiService() which will return the ApiInterface created through the Retrofit Client. This allows the ApiInterface to be implemented throughout the app to make calls make calls to the REST Api.

```
public class ApiClient {

    public static final String BASE_URL = "http://6ad130f2.ngrok.io";
    private static Retrofit retrofit;

    public static Retrofit retrofit() {

        if (retrofit == null) {
            OkHttpClient okHttpClient = new OkHttpClient().newBuilder()
                    .build();

            GsonBuilder gsonBuilder = new GsonBuilder();

            Gson customGson = gsonBuilder.create();

            retrofit = new Retrofit.Builder()
                    .baseUrl(BASE_URL).client(okHttpClient)
                    .addConverterFactory(GsonConverterFactory.create(customGson))
                    //.addCallAdapterFactory(RxJava2CallAdapterFactory.create())
                    .build();
        }
        return retrofit;
    }
}
```

```
public class ApiUtils {

    public static ApiInterface getApiService() {

        return ApiClient.retrofit().create(ApiInterface.class);
    }
}
```

```
@POST("addCloudAnchorId")
@FormUrlEncoded
Call<CloudAnchorResponse> addCloudAnchor(@Field("anchor_id") String anchor_id, @Field("model_id") int model_id);

@POST("getCloudAnchorId")
@FormUrlEncoded
Call<CloudAnchorResponse> getCloudAnchor(@Field("anchor_key") int anchor_key);

@GET("getModels/{userId}")
Call<ARResponse> getARModels(@Path("userId") int userId);

@GET("getLessonsCompleted/{userId}")
Call<LessonResponse> getLessonsCompleted(@Path("userId") int userId);

@POST("setLessonsCompleted")
@FormUrlEncoded
Call<ApiResponse> setLessonsCompleted(@Field("userId") int userId, @Field("lessonId") int lessonId);

@GET("getStudentSubmissions/{id}")
Call<AssignmentSubmissionResponse> getStudentSubmissions(@Path("id") int id);

@POST("setQuizCompleted")
@FormUrlEncoded
Call<ApiResponse> setQuizCompleted(@Field("userId") int userId, @Field("quizId") int quizId, @Field("quizScore") int quizScore);

@GET("getQuizzesCompleted/{userId}")
Call<QuizResponse> getQuizzesCompleted(@Path("userId") int userId);
```

MAKING CALLS TO REST API FROM VIEWS

Making calls to the REST Api from the views was carried out by implementing the ApiInterface which had been set up. In the code snippet below, there is a function which is called to create an assignment in a class. The correct method from the ApiInterface is called and is given the arguments to send as part of the body. A callback is added to the method so that the response can be caught. In this case the response is modelled by the java object 'ApiResponse'. Several checks are performed to ensure the response is not null and is the correct response type. This is an asynchronous request and it is carried out on a background thread. However, when the response is returned, it jumps back onto the main thread so that any UI changes can be made.

```
private void createAssignment(String title, String desc, String dateDue) {
    progressBar.setVisibility(View.VISIBLE);
    String dateAssigned = new SimpleDateFormat( pattern: "yyyy-MM-dd", Locale.getDefault()).format(new Date());
    apiInterface.createAssignment(classNumber, title, desc, dateAssigned, dateDue).enqueue(new Callback<ApiResponse>() {
        @Override
        public void onResponse(Call<ApiResponse> call, Response<ApiResponse> response) {
            if (response.body() != null && response.body().getCode() == 200) {
                Toast.makeText(getContext(), R.string.create_assignments_success, Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText(getContext(), R.string.create_assignments_failed, Toast.LENGTH_SHORT).show();
            }
            progressBar.setVisibility(View.GONE);
            swipeRefreshLayout.setRefreshing(false);
        }

        @Override
        public void onFailure(Call<ApiResponse> call, Throwable t) {
            Toast.makeText(getContext(), R.string.create_assignments_failed, Toast.LENGTH_SHORT).show();
            progressBar.setVisibility(View.GONE);
            swipeRefreshLayout.setRefreshing(false);
        }
    });
}
```

NODE.JS REST API

The Node.js REST Api has many routes which act as an endpoint between the Android app and the database. The main implementation of the REST Api and some of these routes are explained below.

MAIN APP.JS FILE

In implementing the REST Api, a router is used to handle all of routes. The router allows the routes of the application to be distributed into appropriate files. These different files are all managed from the app.js file. One of these sample files can be seen in a subsequent section. The app is defined as an Express app. This app collects all the routes defined by the routers in the different files and uses them. The app then listens on the port defined by the environment or port 3003.

At the bottom of the file it can also be seen that the app is serving static files. These static files are AR models, assignment submissions and the files needed for the web app. For assignment submissions and AR models, they are uploaded and saved in the file system and a reference to the file is stored in the database.

```
// load our app server using express....
const express = require('express')
const app = express()
const morgan = require('morgan')
const bodyParser = require('body-parser')
var session = require('express-session');

app.engine('html', require('ejs').renderFile);
app.set('view engine', 'html');
app.set('views', __dirname + '/public');

app.use(bodyParser.urlencoded({extended:true, limit:'50mb'}))
app.use(session({
    secret: "Shh, its a secret!",
    resave: false,
    saveUninitialized: true
}));
app.use(morgan('short'))

const authentication = require('./routes/authentication.js')
const classes = require('./routes/class.js')
const cloudAnchors = require('./routes/cloudAnchors.js')
const assignments = require('./routes/assignments.js')
const assignmentSubmission = require('./routes/assignmentSubmission.js')
const models = require('./routes/models.js')
const webAppAuth = require('./routes/webAppAuth.js')
const lessons = require('./routes/lessons')
const quizzes = require('./routes/quizzes')

app.use(authentication)
app.use(classes)
app.use(cloudAnchors)
app.use(assignments)
app.use(assignmentSubmission)
app.use(models)
app.use(webAppAuth)
app.use(lessons)
app.use(quizzes)

app.use(express.static('./public'))
app.use('/uploads', express.static('./uploads'))
app.use('/assignmentSubmissions', express.static('./assignmentSubmissions'))

app.get("/", (req, res) => {
    console.log("Responding to root route")
    res.send("This is the root route...")
})

const PORT = process.env.PORT || 3003
app.listen(PORT, () => {
    console.log("Server is up and running...")
})

module.exports = app;
```

SAMPLE ROUTER FILE

```
const express = require('express')
const server = require('./server')
const router = express.Router()

router.post('/setLessonsCompleted', (req, res) => {

    const userId = req.body.userId
    const lessonId = req.body.lessonId

    const queryString = "INSERT INTO lessons (studentId, lessonId) VALUES(?, ?) ON DUPLICATE KEY UPDATE lessonId=?;"
    server.connection().query(queryString, [userId, lessonId, lessonId], (err, results, fields) => {
      if (err) {
        console.log(err)
        res.send({
            "code":500,
            "success":"Setting lessons failed"
            });
      } else {
        res.send({
            "code":200,
            "success":"Adding lessons sucessfull",
          });
      }
    });
});

router.get('/getLessonsCompleted/:id', (req, res) => {
    console.log("Fetching user with id: " + req.params.id)

    const studentId = req.params.id
    const queryString = "SELECT * FROM lessons WHERE studentId = ?"
    server.connection().query(queryString, [studentId], (err, results, fields) => {
      if (err) {
        console.log("Failed to get lessons: " + err)
        res.sendStatus(500)
        return
      }
      res.send({
        "code":200,
        "success":"lessons retrieved successfully",
        "completedLessons":results
        });
        console.log(results[0])
    })
})

module.exports = router
```

Here is a sample file, lessons.js, which is used by app.js. A router is defined in the file which has two routes, one GET route and one POST route. Finally, at the end the file the router is exported so that it can be used by app.js.

In the POST route ('/setLessonsCompleted') two fields userId and lessonId are expected in the POST body. The query string is defined and then the database is queried with the defined query. The response is then returned.

In the GET route ('/getLessonsCompleted/:id') the parameter id is used in the query string which is then passed to the database. The response is then returned.

SENDING NOTIFICATIONS

Notifications have been implemented in the app to sends email when certain events happen. When a new assignment is uploaded in a class or a student uploads an assignment submission or a teacher

grades an assignment submission, an email is sent to the appropriate recipient to inform them of it. e.g. email sent to student to notify them of new assignment in the classroom.

Below is the code used to set up the notification server:

```
var nodemailer = require('nodemailer');

var transporter = nodemailer.createTransport({
  service: 'gmail',
  auth: {
    user: process.env.EMAIL,
    pass: process.env.PASSWORD
  }
});

var mailOptions = {
  from: process.env.EMAIL,
};

function sendMail(subject, text, mailList) {

    mailOptions.to = mailList
    mailOptions.subject = subject
    mailOptions.text = text

    transporter.sendMail(mailOptions, function(error, info){
        if (error) {
          console.log(error);
        } else {
          console.log('Email sent: ' + info.response);
        }
      });
}

module.exports = {
    sendMail: sendMail
}
```

The function sendMail is exported so it can be used anywhere within the app to send an email to a list of emails with a subject and body. The email account and password used to set up the email service are stored as environment variables for security reasons.

WEB APP

The following snippet shows some of the routes of the web app which are defined in one of the router files in the Node.js REST Api. The first route ('/login) renders the login page. The second route ('/dashboard') only renders the dashboard if the user is already signed in. This stops unauthorized access in the app. This is checked in the function checkSignIn which checks if there is a current session

user, if there is not then an error is returned, and the user is redirected to the login page. Similarly clicking the logout button will direct the user to the logout route which will kill the user session and redirect to the login.

```
const express = require('express')
const passwordHash = require('password-hash');
const server = require('./server')
const router = express.Router()

router.get('/login', (req, res) => {
    console.log(req.session.user);
    res.render('login');
})

router.get('/dashboard', checkSignIn, function(req, res){
    console.log(req.session.user);
    res.render('dashboard', {email: req.session.user.email})
});

router.use('/dashboard', function(err, req, res, next){
    console.log(err);
    //User should be authenticated! Redirect him to log in.
    res.redirect('/login');
});

router.get('/logout', function(req, res){
    req.session.destroy(function(){
        console.log("user logged out.")
    });
    res.redirect('/login');
 });
```

```
function checkSignIn(req, res, next){
    if(req.session.user){
        next();     //If session exists, proceed to page
    } else {
        var err = new Error("Not logged in!");
        console.log(req.session.user);
        next(err);  //Error, trying to access unauthorized page!
    }
}

module.exports = router
```

PROBLEMS SOLVED

LEARNING NEW LANGUAGES

Before this year, I had never coded in JavaScript or used Node.js. As well as this, I had never created a REST Api to deal with calls to the database.

To overcome this, I followed npm documentation and YouTube tutorials on creating Node.js REST Api's and linking Node.js with MySQL databases. Overall it was a very worthwhile experience and I learned a lot by doing it.

IMPLEMENTING AUGMENTED REALITY

Figuring out how to implement augmented reality in the project was a difficult process. I had no previous experience using or developing augmented reality software. By researching online, I realized there were several possible platforms for development. The decision was made to use ARCore. ARCore is a software development kit developed by Google that allows for augmented reality applications to be built.

Because ARCore is only out of Beta one year, resources available on the software were not plentiful. A lot of time was spent reading through the developer's docs, as there was a lack of articles and sample projects available. As well as this, because the development kit is based on OpenGL, some knowledge of OpenGL was needed to be able to use the software. This was done through online resources and videos, however the CA4007 module on Computer Graphics was also useful in helping to grasp the concepts needed.

CLOUD ANCHORS PROBLEM

For several days, there was a problem with hosting 'cloud anchors' which are used in sharing augmented reality models. This feature had a bug in the implementation meant when an anchor was hosted, the result was never being returned from the server.

This problem persisted for a considerable length of time however the problem was stemming from not properly listening to the response from the server.

There was a lot of debugging involved in solving the issue. The android app was verified to ensure there were no nulls on the android side when trying to host (there was not), then checks were made to ensure the requests were being received on the server side (they were) and finally if the result was being caught on the android side (it was not). The problem was pinpointed to the android side on the server response. Following much debugging on this side, the listener was found to have been set up incorrectly to listen for the result. Fixing the problem took a lot longer than had been initially expected.

GITLAB CI

For several weeks the CI builds of the project were failing due to problems on the CI side when running gradle. This was a major problem as it meant individual commits and merge requests couldn't be distinguished as failing due to changes which had been made by the developer or if it was due to the CI.

There was a problem running some scripts in the test suite and following some debugging it was discovered that the root cause was that multiple Docker containers were trying to share one Gradle cache on the host machine (which is not possible). This was causing the build to fail. To fix this, separate caches were set up for each stage of the CI. The job stage and job name were added into the path to the cache. By doing this it sets up a separate cache for each job in each stage and in turn means there is no sharing of caches between jobs.

It was a small fix but finding the source of the problem in the first place was difficult and time consuming. However, it was very important to get the CI back working so that the builds could be verified.

TESTING REST API

Finding a suitable way to sufficiently test the REST Api was a problem which was faced during the development. The main problem was that nearly all the REST Api routes were calls to the database and contained no actual business logic. This meant that if the database was mocked or stubbed then testing these routes would be pointless. It was also not an option to create tests which would involve calls to the real database as this is bad practice. The reason for this is that it would contaminate the database with test data and possible delete real user data.

The solution to this problem was to export the current schema from the real database and use it to create a test database. By doing this, the test database was an exact copy of the real database so testing this would be the same as testing the real database. In the test file, a test environment was created which pointed instead to the test database. This meant that every time the tests were ran, they ran across the test database, leaving the real database untouched and uncontaminated.

HOW TO INCLUDE THE PEDAGOGY ELEMENT

When creating the lessons in the app there were several factors which were difficult. First, creating lessons which incorporated augmented reality was quite a challenge. The main problem was incorporating an adequate pedagogy element in the lessons, as this is an area which developers do not generally have any experience in.

To solve this problem, various TG teachers were approached to get their input and I went to St. Pats to meet with Joe Travers, head of School of Inclusive and Special Education. Joe talked about 3 main concepts: Linguistics, Concept & Procedure in introducing the pedagogy element. He noted that the 'concept' element had been hit by using AR, but it was the linguistics and the procedure of portraying these concepts that needed work. He gave great advice as to how to introduce these into the lessons and this was of great benefit.

RESULTS

FUTURE WORK

This project has a lot of scope for scaling in the future. There are several key areas which could be targeted to improve the application that would improve it. These are outlined below.

INCREASE NO. LESSONS

Currently, there are 10 lessons implemented in the app on various topics. These lessons were designed like the lessons in Googles own augmented reality teaching app – Expeditions and Geogebra.org, a math learning application online which can show similar concepts in 3D. The main information which was gained by observing these two applications was that the lessons were kept as short as possible with plenty of exploratory learning, but to increase the quantity of lessons. The aim of this app follows along the same lines, have many short lessons which hit small learning objectives, which overall meet the main learning objective. This will not be a quick process as designing and implementing augmented reality lessons is time consuming and difficult.

INCREASE NO. QUIZZES

Along with the lessons, the number of quizzes in the app will need to be increased and expanded. These can assess students on the lessons they have completed. They are very interactive and help in the learning process, so a greater quantity and variety would greatly improve the app.

MAKE THE APP CROSS PLATFORM

Making the app cross platform would be hugely beneficial. Schools use either Android or iPad Tablets for the most part so having the ability to use the app on both Android and iOS would open the app up to a larger user base.

Having decided to implement the augmented reality features through ARCore will allow for the iOS app to be developed using the same software.

EXPAND THE WEB APP

The web application also has a large scope for expansion. The web app was not initially in the scope of this project but due to an introduced requirement, it became necessary. Extra functionality for teachers and students in terms of the classroom and other resources can be added in the future. The web app will not include any augmented reality as it is only an interface for the users.

APP AS SUPPLEMENTARY RESOURCE

Due to the feature 'Augmented images', the app would serve perfectly as a supplementary resource to a school TG resource (Like a book). This feature would allow images in the book to come to life in augmented reality through the app.

FINAL REFLECTIONS & CONCLUSIONS

Overall, I am extremely satisfied with the finished product of my application. All the initial requirements have been satisfied and I also had the opportunity to add some adaptations and implement some new features. The goal was to deliver a product which could be effectively used by students and teachers in the learning of the subject TG using augmented reality. I feel I have done this and am happy with the product I now have.

From the research carried out on the topic and the user testing conducted I can conclude that augmented reality has proven to be a great way to convey the subject content. The research papers that were read through had already given great benefits on the use of augmented reality in the area and so it gave great confidence going forward with the project. User testing conducted in the secondary school also reaffirmed this as the students and teachers really enjoyed the app and found it useful.

I always had intentions to work on a project which involved augmented reality as it is an emerging area which had really interested me. This project gave great experience in the development of an augmented reality product and the special considerations which need to be taken when doing so. This is an area I would have a great interest in pursuing in the future.

APPENDICES

REFERENCES

1. Pedagogy Research - https://www.shmoop.com/teachers/curriculum/textbooks-resources/supplementary-resources.html
2. Pedagogy Research - https://www.teachingenglish.org.uk/blogs/davedodgson/bringing-outside-%E2%80%93-choosing-supplementary-resources-classroom
3. Diagrams – https://creately.com
4. ARCore - https://developers.google.com/ar/
5. Rest Api using retrofit - https://medium.com/@saquib3705/consuming-rest-api-using-retrofit-library-with-the-help-of-mvvm-dagger-livedata-and-rxjava2-in-67aebefe031d
6. Remote / Local Storage - https://medium.com/@eslam.hussein/dominate-remote-local-data-with-rx-retrofit-room-mvp-f2b13a0ac27b
7. Gitlab CI/CD - https://docs.gitlab.com/
8. Node.js and using its dependencies - https://www.npmjs.com/

APPENDIX 1 – RESEARCH PAPERS

[1] Figueiredo, Mauro, et al. "Learning technical drawing with augmented reality and holograms." Proceedings of EDU'14 (2014): 11-20.

[2] Tumkor, S., et al. "Integration of augmented reality into the CAD process." in Proceedings of the ASEE Annual Conference & Exposition, Atlanta, GA, USA. 2013.

[3] Murthy, Madhav, et al. "Augmented Reality as a tool for teaching a course on Elements of Engineering Drawing." Journal of Engineering Education Transformations (2015): 295-297.

[4] Horii, Hirosuke, and Yohei Miyajima. "Augmented reality-based support system for teaching hand-drawn mechanical drawing." Procedia-Social and Behavioral Sciences 103 (2013): 174-180.
[5] Álvarez, Ayala, et al. "From 2D to 3D: Teaching terrain representation in engineering studies through Augmented reality: Comparative versus 3D pdf." 2014 IEEE Frontiers in Education Conference (FIE) Proceedings. IEEE, 2014.