Creative Citizen Augmented Reality Application

On 11. November 2020   
Information Systems and New Media / Siegen University, Siegen

Author: Hassan Hussein

Hassan.hussein@student.uni-siegen.de

# Table of Contents

Introduction 3

System Requirments 4

Installation Guide 5

System Design 8

Application Desgin 10

Code Desgin 13

Database Desgin 15

Added Assets 18

Recommendations 19

# Introduction

The application is mainly built on an open source project called [Mapbox](https://docs.mapbox.com/unity/maps/examples/world-scale-manual-align-ar/). Mapbox itself is a ready made SDK for unity, that was built to help the developers to develop interactive augmented reality applications. The SDK helps the developers to create game objects via a C#-based API and graphical user interface. Mapbox supports both android and iOS builds, therefore it also supports Google AR core. This project was built also to support AR core.

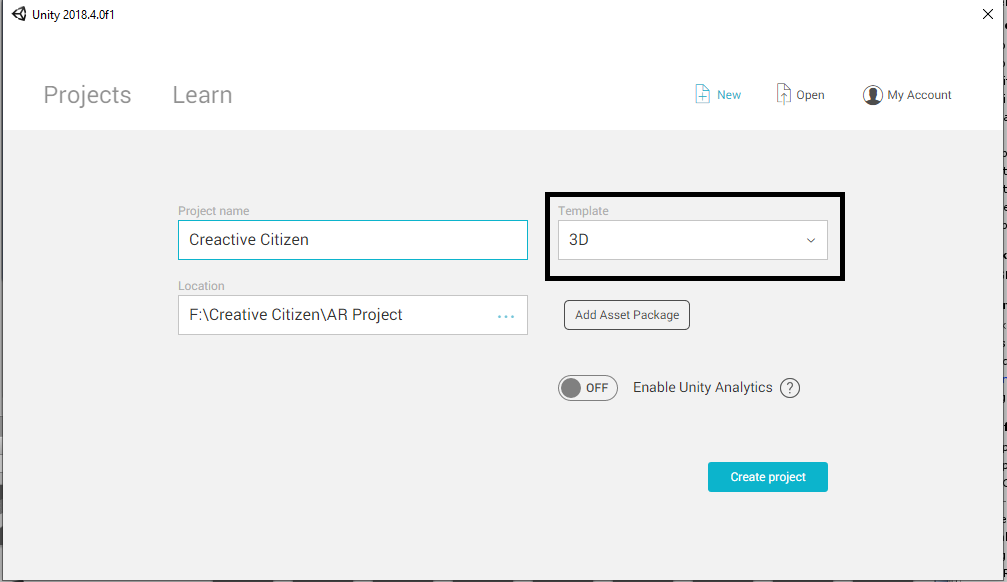
# System Requirements

|  |  |  |  |
| --- | --- | --- | --- |
|  | **System Name** | **Version** | **Type** |
| 1 | Unity | 2018.4.0f1 [here](https://unity3d.com/get-unity/download/archive) | Software |
| 2 | Android device supports AR Core | 7.0 or newer. Please check this [page](https://developers.google.com/ar/discover/supported-devices) | Hardware |
| 3 | USB Cable | Any | Hardware |

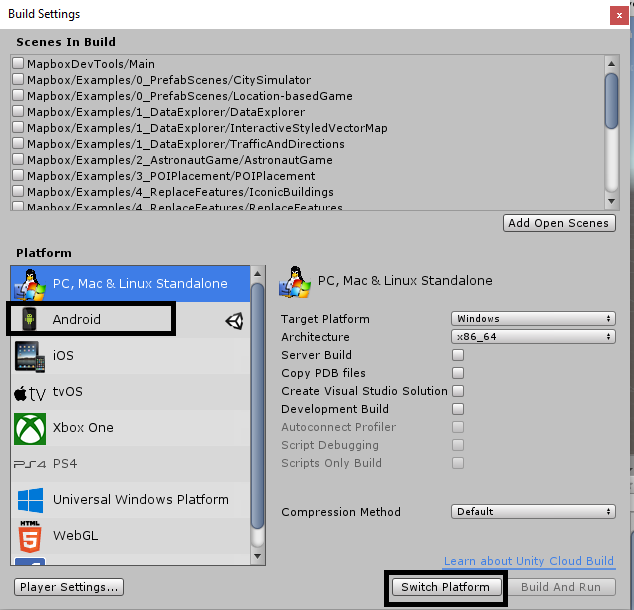
# Installation Guide

In this section we will learn how to install the augmented reality application on Android device. But before we do that, we should install the prober software, to make it work the best as expected. So it is very important at the first place to check the System Requirements section before you go ahead in this section.

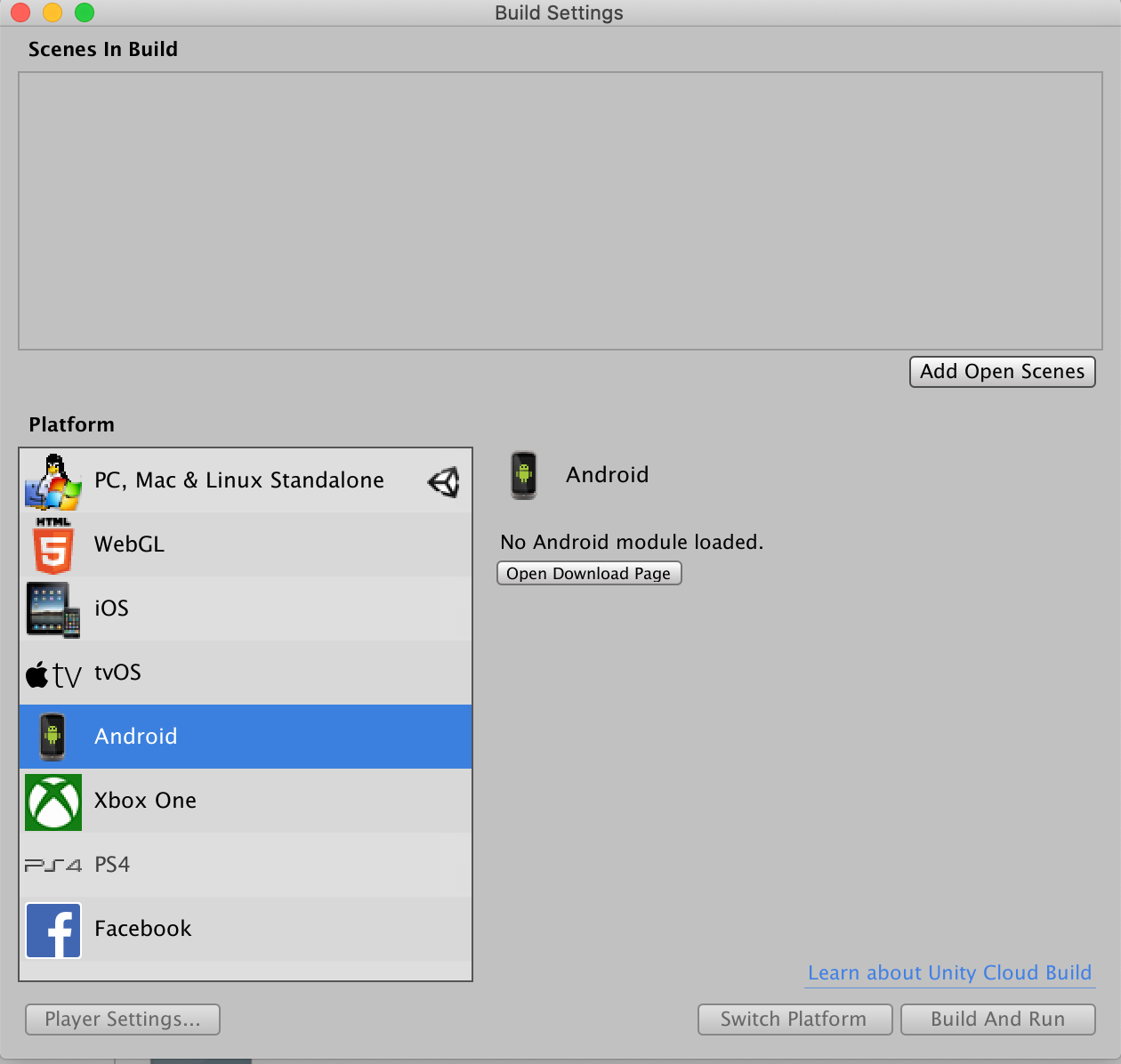
1. Connect your android device to your computer using the USB cable.
2. In addition, you should prepare your android device for the build by enabling developer options and USB debugging by following these [steps](https://developer.android.com/studio/debug/dev-options)
3. Download the project from our GitHub repository [here](https://github.com/Webo1980/Creactive-Citizen/tree/master) , extract it, and find the folder sdkproject
4. Open unity and select create a new 3D project, give your project a name, select the folder sdkproject.



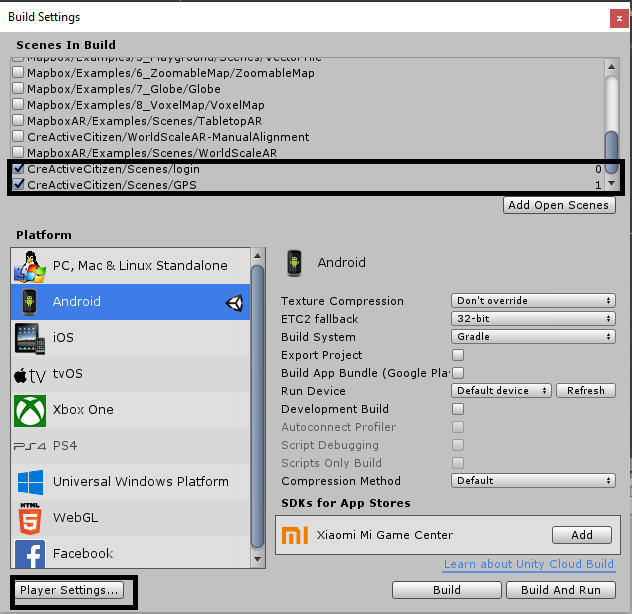
1. In unity select File -> Build Settings and then select Android, and then switch platform.



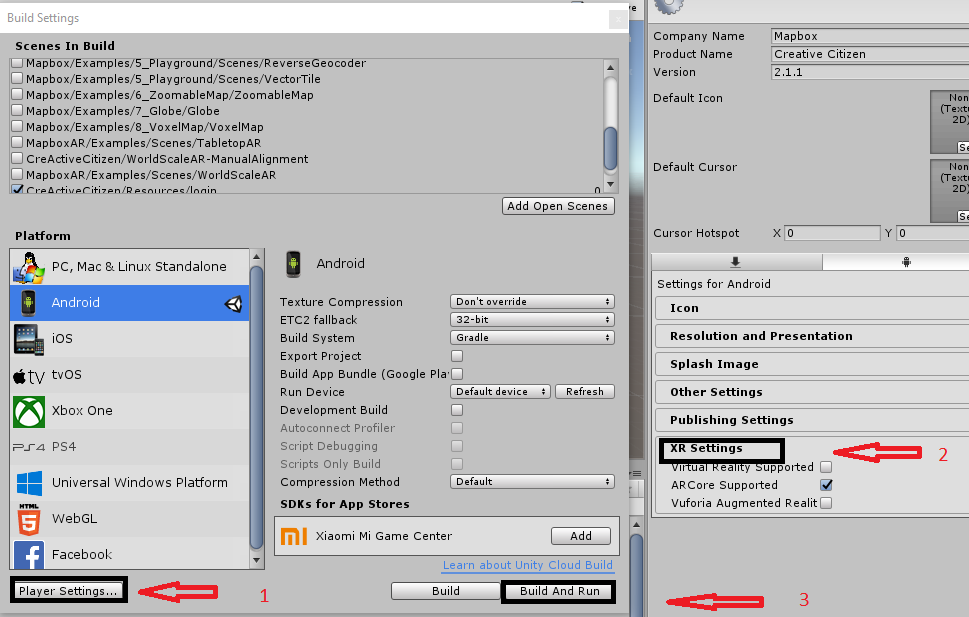
1. In case you see the below screen, you will need to download Android SDK from [here](https://developer.android.com/studio) and install it on your computer (laptop/PC).



1. Make sure that at least GPS, and Login scenes are checked and next to login is 0, while next to GPS is 1 in the below list, if not click on the checkbox next to login first and then the checkbox next to GPS.



1. Click on player settings, and then other settings. Make sure that the minimum API level is Android 7 or higher (it depends on the Android version on your device)
2. Scroll down a little bit and click XR settings and then click on ARCore Supported.
3. Leave every thins the same in the player settings. Now, your build is ready, and you can click on build and run



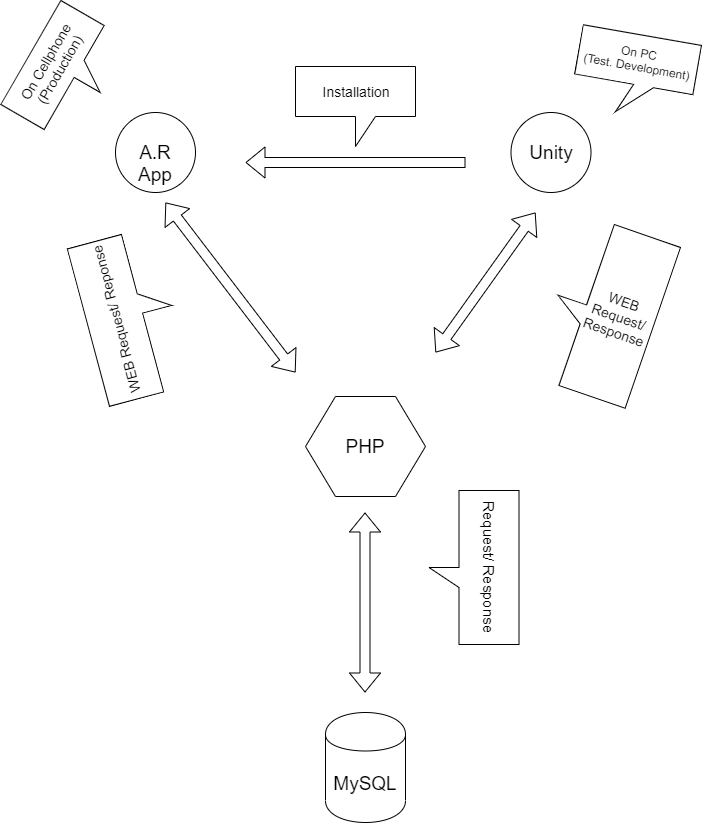
1. After the application is installed on your device, open the application and then the application will ask you for some permissions please accept it.
2. Log in using your username and password, if you see a black screen you might need to give some additional permissions to the app. From your android device go to settings->apps and then click on your app name (if you did not change it before the build in the player settings section the name should be “creative citizen”) and then -> authorization and then give permissions to the camera.
3. Close the application and make sure to remove it from your task manager and then reopen it again. Make sure that you have internet connection either using a Wi-Fi or using your mobile data. In addition, you will need to enable the GPS in your android device.

# System Design

The application is designed as shown in the below diagram. The system is developed mainly in 3 layers as following:

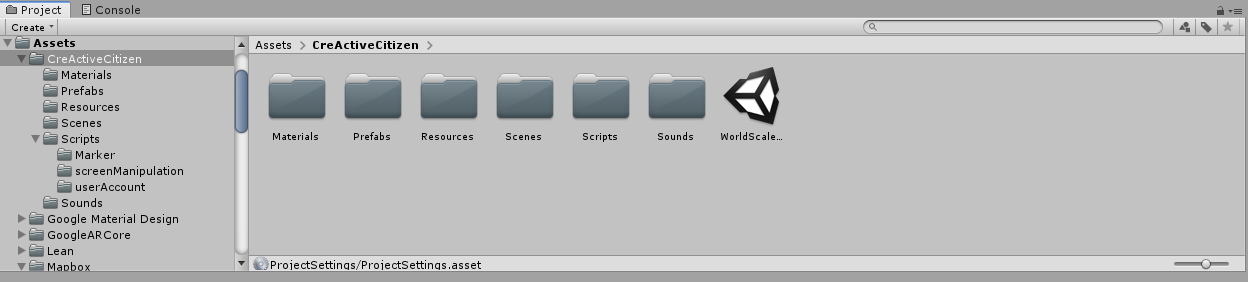
1. **Unity**: is used as the main design interface for the application. It is mainly used to design and integrate the C# scripts with the game objects. It is also used as a testing tool to test the application before deploying it to the android device.
2. **PHP& MySQL:** Both are built on a web server to save the user information and markers’ location (longitude& latitude). MySQL was selected because it is an open source database, and it has a high integrate with PHP. Furthermore, PHP is a scalable scripting language that is widely used in the web, and it has a good compatibility to integrate with the C# scripts. Both PHP, and MySQL are acting as the application backend where it send/ request the user data between the android device and the end user.
3. **Android Device:** The device serves as the application frontend, where the user can interact with application and set his/ her marker and design the scene on his/ her own.

The crucial question here is why saving the data on the database and not on the user device? The reason behind that, that the philosophy behind the application itself requires that the user should be able to see the other users’ markers. In that case it will be not helpful to save the users markers location on their own devices because in that case it will be impossible to show their markers to the other users.



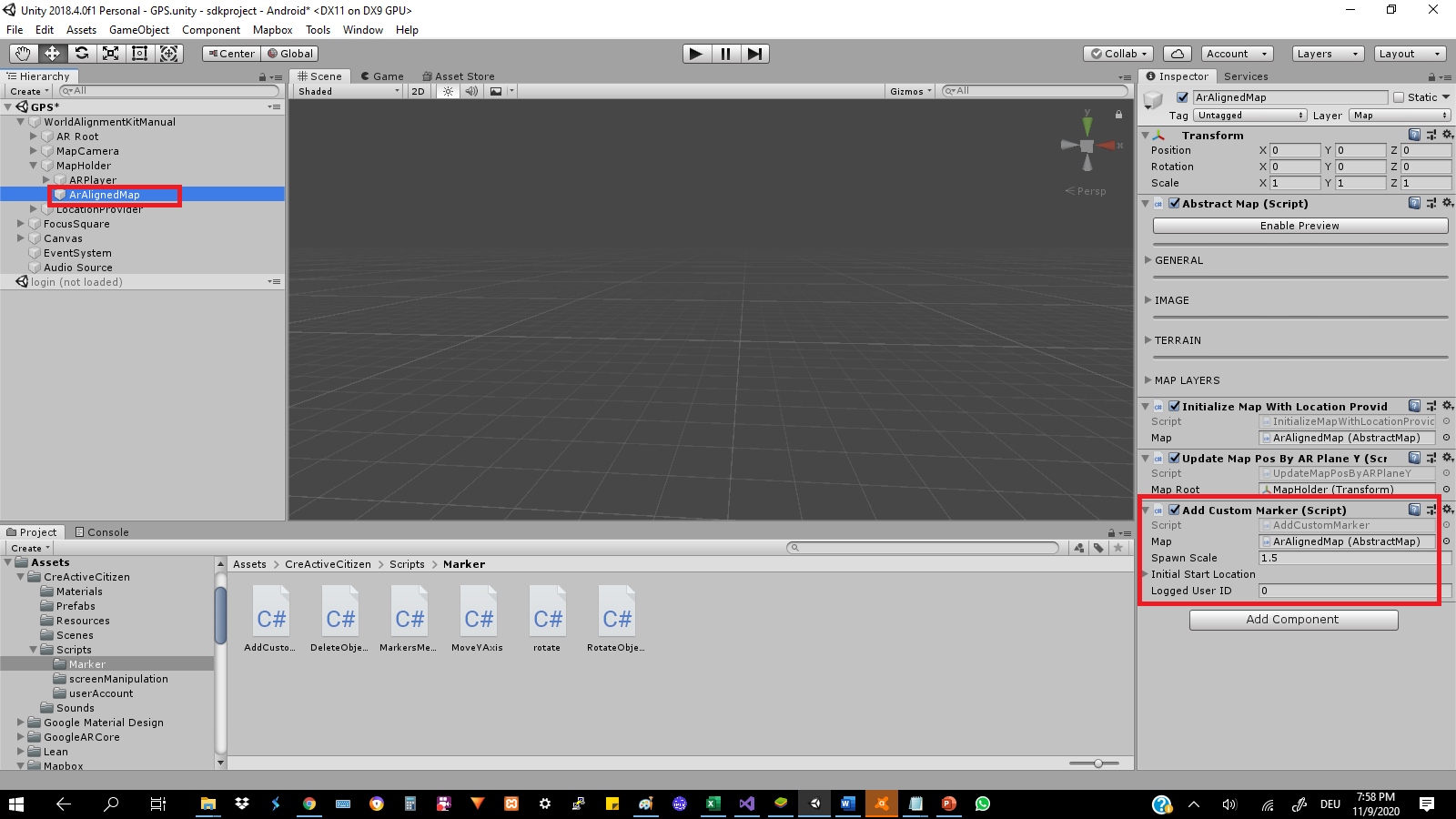
# Application Design

After importing the application to unity you can find it on the path Assets/ CreActiveCitizen.



As it is shown in the previous photo, there are mainly the following folders:

1. **Materials**: In this folder are mainly all icons, that are used in the application. Also, it should contain all the materials, that will be used in the future (no materials were developed for this project so far).
2. **Prefabs**: In this folder there is all the prefabs that are used in the project. It was mainly used to back up the designs. Just in case I needed to revert some changes. The prefab of any design could be easily created be dragging it from the hierarchy and drop it to the prefab folder.
3. **Resources:** It is very important to state at the beginning, that this folder name should not be changed because the markers menu is searching for the markers objects in this folder. In this folder all the 3D models of the markers are located.
   1. **Sounds:** According to the android guideline, it is highly recommended to play sound along side with the user actions to give him/her the feeling that something is happening in the system. This functionality is newly implemented to the application. The folder is subfolder of Resources for code wise reasons. It is highly recommended to save all the future sounds in this folder to make the system more organized.
4. **Scenes:** In this folder all the scenes are loaded, where I redirect the user from a scene to another depending on their actions. For example, when the application is loaded, at the beginning I load the login scene, and when the user can login successfully the application redirect him/her automatically to the GPS scene, where the user can experience the augmented reality.
5. **Scripts:** In this folder all the C# scripts, that are related to the application are located. In this folder is mainly the DB Manager script located. This script is responsible to handle all the database requests between the front and back end. The folder also has some subfolders as following:
   1. **Marker:** In this folder, you will find all the scripts that deals with the object. Below is the list of these scripts:
      1. **Add Custom Marker:** In this script all the magic happens, as here the user can add a marker of his/her choice. Also, this script is responsible for displaying all the markers that are in a circle of 100 meters based on the user current location(the list of markers is populated by a php script and passed as a JSON format to the script). Furthermore, all the functionalities that are related to the markers itself are written in this script. This is script is attached to the map, to see that go to scenes folder and find GPS scene double click it, and then at the hierarchy expand it. You will find the script under GPS-> WorldAlignmentKitManual-> MapHolder-> ArAlignedMap



* + 1. **Delete Object:** This script is attached to prefab closeMarker under folder Assets/ CreActiveCitizen/Prefabs. This prefab is attached to each marker object manually under Assets/ CreActiveCitizen/Resources and it is shown and hidden using C# script according to the marker case for example it is shown on the top of marker, if it was set by the logged in user. (the close object interaction code e.g., show, hide is written in script add custom marker)
    2. **Marker Menu:** This script is responsible for handling all actions related to the menu. The script is attached to object GPS ->Canvas -> Objects List
  1. **Screen Manipulation:** There is only two scripts here, one is responsible for sharing the screen and one is responsible for taking a screenshot.
  2. **User Account:** In this folder, there is only one script. This script is very important and responsible formanaging all functionalities related to the user account, such as login, register, forget password. All the functionalities are existed but not working fully, as there is a problem when moving between the scenes, the user information is lost. But the user can login normally and after login his/her user ID is registered to the system memory as an identification ID, that is used all time to interact with the database.

# Code Design

The code was designed in a way to make the system very generic and give the ability for more scalability in the future. There are some functionalities were added recently in a short time. So, it is highly recommended to refactor the code as soon as possible. So that, in the future changes, would it be easier for tracking and changing. There are almost comments in every line inside the folder Assets/ CreActiveCitizen/Scripts, and any other C# script in the project is a part of the Mapbox project, and it is highly recommended to refer back to the project official documents if needed.

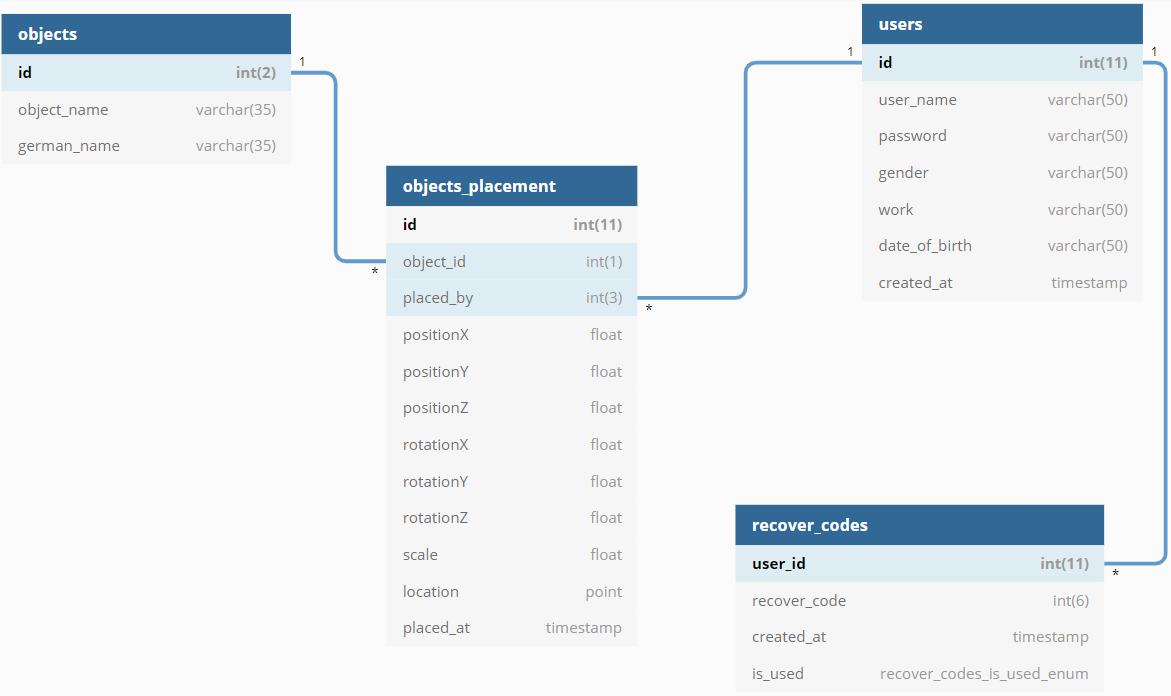
In this section we will learn more about how the relation between the application parts (C#,PHP, MySQL) are linked and how the data flow works in the application. We can see the application from the code level as three tires as following:

The script DBManager acts as the bridge between the C# scripts and the PHP scripts. The data is passed between the C# and the PHP is in JSON format, while the data that is passed between the PHP and MySQL, is in SQL query format. The below table shows which functions are called in each script.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C# Script | C# Method | DB Management Script (D.B.M) | D.B.M Method | PHP Script | Table |
| Add Custom Marker .cs | Save Marker Info () | Post Coordinates () | Add Coordinates .php | Objects \_ placement |
| Spawn Markers on Start () | Get Coordinates () | Get Coordinates .php |
| Markers Menu .cs | Start () | Get Objects Lists () | Get Objects List .php | objects |
| Login Manager .cs | Check Login Info () | Check Login Data () | Check User. php | Users |
| Register () | Register () | Register .php |
| Forget Password () | Recover Password () | Forget Password .php |
| Change Password () | Change Password () | Change Password .php |
| Verify E Mail () | After Registration () | Verify Account .php | Recover \_codes |

# Database Design

The below database diagram shows the relations between the different tables:



The below table explains each table, and data type and which column is needed for:

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Column Name | Datatype | Notes |
| Objects | ID | Integer | Primary key |
| Object\_ name | Varchar | Should be the same name as the object name in folder /Assets/Cre Active Citizen/Resources |
| German\_ name | Varchar | This is what is showed in the app menu |
|  | | | |
| Objects Placement | Object \_id | Integer | Foreign Key of table objects |
| Placed \_by | Integer | Foreign Key of table users |
| Position X | Float | Object position in A.R environment on the axes x |
| Position Y | Float | Object position in A.R environment on the axes y |
| Position Z | Float | Object position in A.R environment on the axes z |
| Rotation X | Float | The object rotation in the AR environment using [Euler Angles](https://docs.unity3d.com/ScriptReference/Transform-eulerAngles.html) calculation on the axes x |
| Rotation Y | Float | The object rotation in the AR environment using [Euler Angles](https://docs.unity3d.com/ScriptReference/Transform-eulerAngles.html) calculation on the axes y |
| Rotation Z | Float | The object rotation in the AR environment using [Euler Angles](https://docs.unity3d.com/ScriptReference/Transform-eulerAngles.html) calculation on the axes z |
| Scale | Float | The object scale according to unity measures |
| Location | Point | The object location based on the GPS system in two coordinates (longitude, latitude) |
| Placed \_at | Timestamp | The time when the object was set in the AR environment |
|  | | | |
| Users | Id | Integer | Primary Key |
| User \_name | Varchar | Usually it is email address |
| Password | Varchar | One-way encrypted password using method MD5 |
| Gender | Varchar | It is better to be changed later to Enum type, to be standard |
| Work | Varchar |  |
| Date \_of \_birth | Varchar |  |
| Created \_at | Timestamp | When was the account created for the first time |
|  | | | |
| Recover \_ codes | User \_id | Integer | Primary key |
| Recover \_code | Integer | Randomly generated code to allow the user to recover the account |
| Created \_at | Timestamp | When was the code was created (it is recommended for security reasons to make the code expire after some time e.g. 24 hours) |
| Is \_used | Enum | A flag to identify whether the code was activated or not |

# Added Assets

There are some assets that were imported from unity assets to enrich the experience. The assets are listed below:

1. **Lean**: This asset is used to give the object the elasticity and scalability. So, the user can move the marker in all directions and can also resize it to the desired size. This asset is imported from this [link](https://assetstore.unity.com/packages/tools/input-management/lean-touch-30111). The asset components could be added as a component manually to any object but in our case it is added using scripts, as the markers are added dynamically and components also should be added like this. Below is an example form script add custom marker:

obj.AddComponent<LeanTouch>();

var Scale = obj.AddComponent<LeanPinchScale>();

Scale.Sensitivity = 1f;

Scale.Dampening = -1;

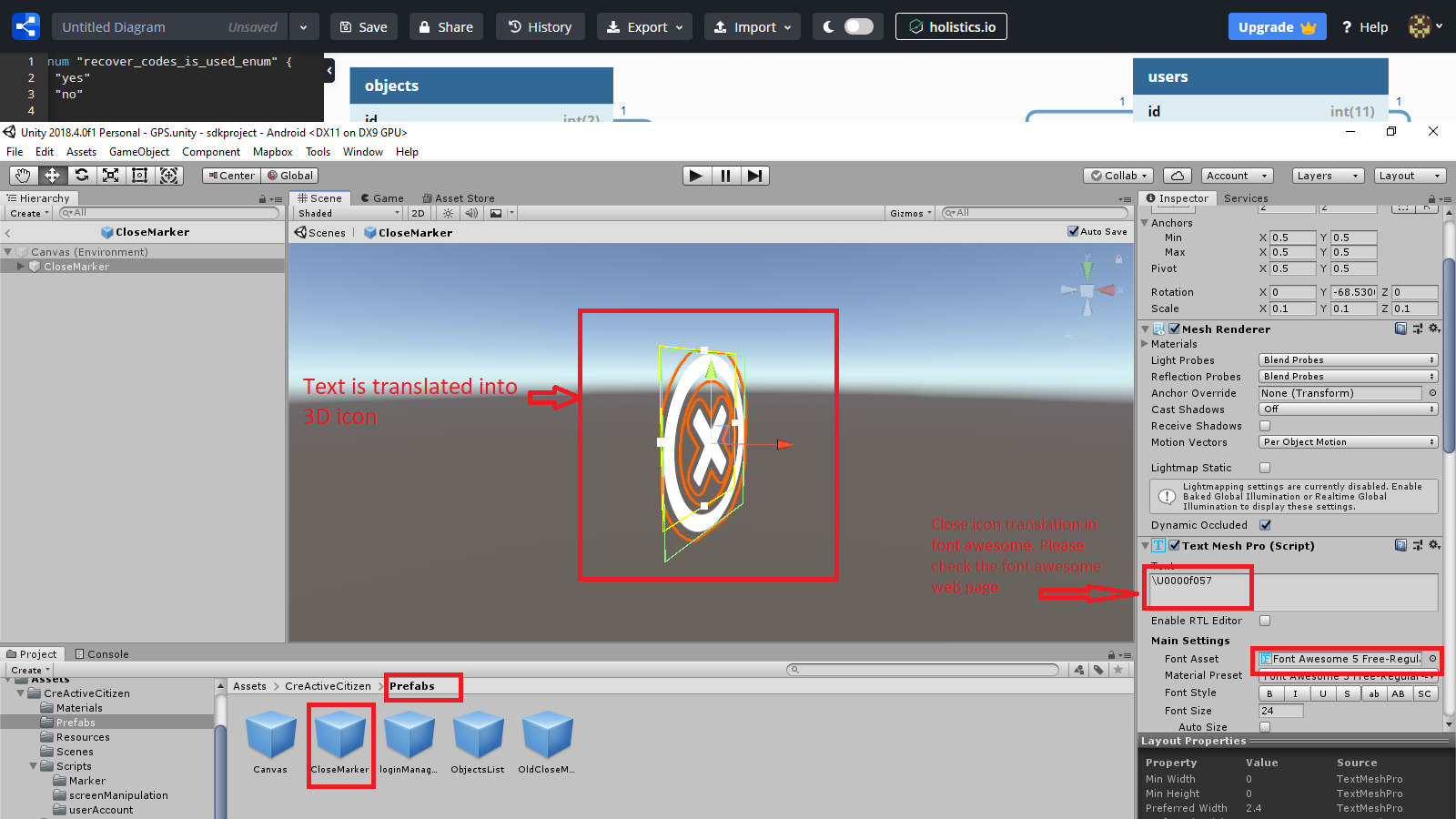
var Twist = obj.AddComponent<LeanTwistRotateAxis>();

Space World = default;

Twist.Space = World;

var Move = obj.AddComponent<LeanDragTranslate>();

1. **Text Mesh Pro**: This asset is used to add mainly used to add some icons using the font awesome. It is mainly imported to create the close icon, as the photo icons in the AR environment looks like a 2D image, while the Text Mesh Pro looks like a 3D object. Please look at the below screen.



# Recommendations

1. Testing AR apps on unity is a nightmare, as some functionalities cannot be tested from unity like touching the screen for example. So, with every little change you must deploy the app on your android device and then test it. The deployment time usually takes around 7-10 minutes depending on your hardware efficiency. But this is not now a problem since unity has offered a great tool called Unity Remote. This tool must be installed to your android device, and when you connect your device with the USB cable to your computer, so you can test the running application on unity from your device. More information about the tool and how you can use it from [here](https://docs.unity3d.com/2019.3/Documentation/Manual/UnityRemote5.html)
2. I highly recommend using Google Material Design tool from [here](https://assetstore.unity.com/packages/tools/particles-effects/google-material-design-47141)
3. It is also recommended as a best practice, that there should be 3 different isolated servers one for development, the second for testing, and the third for development. So, we make sure that the developer can build the new functionalities or modify the already exists one on a server. Once the developer has finished a build, it should be deployed on the test server and could be tested there. In case, the test is passed successfully, the same code should be deployed on the production. If not, the developer should rework to the fix the errors, and then provide an new version.