**Mobile Application Development**

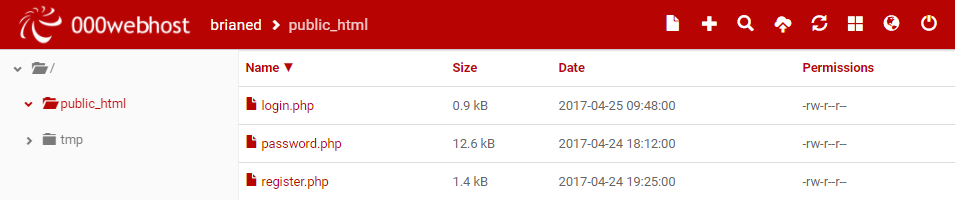
**Asteroids!**

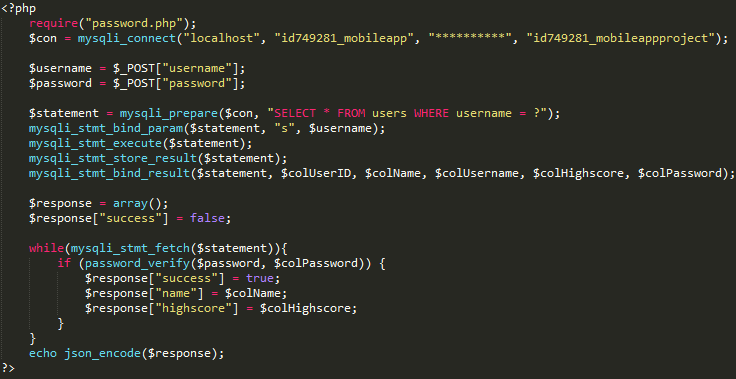
**Brian Dooley (15123529) – Focused on UI elements, Login/Registration System, MediaPlayer, Daily notifications, YouTube API, Credits page also included MiniGame for registered users.**

**Artem Semenov (15164748) – Focused on the Asteroids game engine, user joystick controls, movement of ship, firing of weapon, asteroid spawn and collisions.**

**LoginActivity**

This is the launcher activity. In this activity there are two fields located at the top of the screen. Users, who have previously registered, can enter their username and password and click the Login button to be taken to their personal user area. This login feature is handled using the Volley library which allows the transmitting of network data. Three PHP files stored on my freely hosted site (<https://www.000webhost.com/>) are used to handle the login, registration and password protection processes. These can be seen in the file management portion of the hosting site below. The password.php file can be found [here](https://github.com/ircmaxell/password_compat/blob/master/lib/password.php).



In the login.php screenshot below a connection to the Database is established. The username and password are collected and checked against the database. If correct, a JSON response will be sent back with a success message. If the details are incorrect or the details are not in the database, the response will be false and the user will not be logged in. The AlertDialog class will display a Login failed message if the JSON response failed and the user can try again.

If the username and password details are entered and the user clicks the login button, they will be logged in or an error message displayed. Below the login button is a Play as guest button. This button will allow a user to go directly to the Asteroids game without the need for logging in or registering.

Clicking on login will log the user into their user area once the details are correct. The user area is covered in the UserAreaActivity heading.

Clicking the History button will take you to the HistoryOfAsteroids activity.

Clicking on the Credits button will take you to the ProjectCredits activity.

Clicking on the Sound On/Off button will stop the music file currently being played by the MediaPlayer class. I have implemented several options to stop the sound across the app. To play the history of asteroids video the sound must be muted. There is also an option located in the user area to allow the user to stop playing the music.

The last option on the launcher screen is the Register Here touchable textview. This will bring the user to the RegisterActivity activity screen. This is described in more detail under the RegisterActivity heading.

When the back button is pressed a Toast message will notify the use to hit the back button a second time to exit the app.

**HistoryOfAsteroids**

In this activity I have integrated the YouTube API into the application. Setting up a new Project [here](https://console.developers.google.com/apis/dashboard?project=asteroids-165718&duration=PT1H) was essential to getting the YouTube video to play in the application. I set up this project using the YouTube Data API and used the generated API key to attach it to the application. For security google requires the use of the SHA-1 certificate fingerprint of the app and this was collected through Android Studio and used to create the Android API key. The YouTube player API was downloaded from [here](https://developers.google.com/youtube/android/player/downloads/) and the jar file was placed in the libs folder where it could be included as a dependency (The dependencies for this app can be found in the build.gradle scripts file). When the user clicks on the button to watch the video and Toast message will pop up if the MediaPlayer music file is playing. You will be unable to watch the video until you have clicked the Mute button located below the watch button. In Android Studio, there can sometimes be rendering problems when viewing the activity\_history\_of\_asteroids.xml file in the Design view but these rendering problems are not an issue when the app is run.

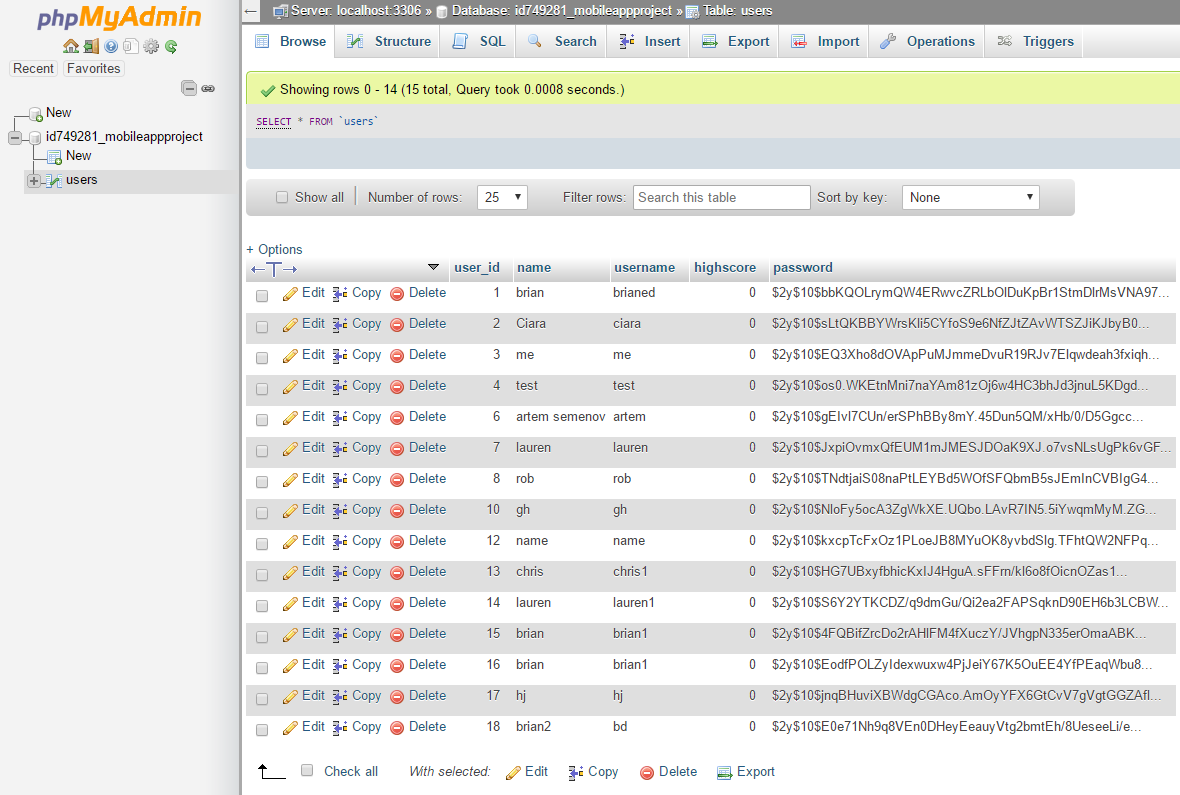
**ProjectCredits**

In this activity there is a button located at the top of the screen and a RelativeLayout located in the rest of the activity to hold the fragment. When the user clicks the Project Members button the first fragment will be displayed showing the first member of the project. The buttons option will then change to display the second member and a second fragment will be displayed showing the second member of the project. Clicking on the back button will bring the user back to the LoginActivity activity. The FragmentManager class is used to interact with the fragment objects in this activity and the FragmentTransaction class is used to add and commit the fragments. The two fragment classes used are CreditsFragment and CreditsFragmentTwo. These two fragment classes are added to the fragment\_container layout in the activity\_project\_credits.xml file using the FragmentTransaction class.

**RegisterActivity**

In this activity the user will be able to register a new account using a name, username and password. All fields must have values entered in order to register or an error message will popuo using the AlertBuilder class. All fields must also be at least 2 characters long. The registration process is also handled using the Volley library and the file used to check users for registration is the register.php file. This is shown in the screenshot below. The user’s data is collected from the edittext fields and checked against the database to ensure username is unique. The password entered is hashed in the database for added security encase of a breach in the database and the user’s details can be accessed. If registration is successful a JSON response is sent back with a success value. If registration fails the user can try again. If the registration is successful the user will be brought back to the login activity where they can then login using these details. The register.php file and the associated database screenshots are located below. Unfortunately highscores are not being implemented as of now and is an option for future development by either using SharedPreferences to store the details on the phone or the initial expectation of storing the retrieving these details from the database.





**UserAreaActivity**

At the top of the activity there is a personalized textview that displays the name of the user that has logged in. In the centre of the activity there are three options. The first option is to play the Asteroids game which when touched will bring the user to the game. The second option is to play a mini game which will start the mini game and an option to turn the sound on or off.

Below these buttons there is two button for setting up the daily notification. This daily notification is currently set to activate at 2pm every day when it is set. By default it is set to off. When the user touches the on button the notification will be active and will display at 2pm (may not appear for a few seconds into the minute). A Toast message will appear when the user hits the on button letting them know the notification is active. If the user presses the off button the notification will no longer occur and a Toast message will be displayed indicating this. The user will not have the option to manually set a time for this but the code implementing the time for this feature is located in the UserAreaActivity class on line 47. The AlarmManager class is used to access the system alarm services. Its purpose here is to allow a notification to be produced once every day using setRepeating(). The notification\_receiver class extends BroadcastReceiver so that broadcast intents can be used. When this class receives the context and the intent a notification can be set. When this notification is displayed on the user’s device they can click it and it will direct them back to the LoginActivity activity.

**MiniGame**

The mini game is made up of several files. These files are appended with ‘MiniGame’ to avoid clashing with the Asteroids main game class files. The java files included are Animation, Background, BottomBorder, GameObject, GamePanel, MainThread, MiniGame, Missile, Player, Smoke and TopBorder. The game is played by touching once on the screen. The player will begin to drop lower on the map. Touching and holding the screen will move the player up. Missiles spawn from the right of the screen and make their way to the left of the screen. If these rockets hit the player the game will stop and reset. The longer the player survives the faster the rockets will move. There are also a top and bottom borders, which if the player flies into will kill the player and restart the game.

* MiniGame – The main activity of the game, this is where the full screen is set and the game panel is created
* GamePanelMiniGame – This class extends SurfaceView for drawing the surface on the device. This class has most of the functionality of the mini game and is always updating the different objects of the game. The collisions are calculated in this class. When the player dies the new game is cleared so the player starts from the start again.
* MainThreadMiniGame – In this class the frames are capped to 30 and override the run method from the Thread class. The game will be updated and everything drawn on the screen 30 times a second.
* BackgroundMiniGame – This class handles the background images that scrolls from right to left
* GameObjectMiniGame – This is an abstract class that will not be instantiated. It holds the values for all the objects used throughout the game. Objects in the game will extend this class.
* PlayerMiniGame – This class is for creating the player image and positioning of the player on the screen
* AnimationMiniGame – This class is for going through the images of the player and missile to give them the appearance of animatation.
* SmokeMiniGame – This class extends the GameObject class and prints a smoke object that moves off the screen to the left of the players image
* MissileMiniGame – This class extends the GameObject class and prints a Missile object that spawns on the right of the screen and moves to the left. The speed of the missile increases the longer the player survives
* BottomBorderMiniGame – This class extends the GameObject class and print the image used for the bottom border
* TopBorderMiniGame - This class extends the GameObject class and print the image used for the top border

**Main Game**

The main game consists of a recreation of the classic game “Asteroids”, which involves a small spaceship flying around and shooting at the rocks flying across the screen. The game screen presented to the user contains a digital analogue joystick, which is used to control the rotation of the ship and two buttons, for firing and boosting. There is a small delay between the amount of bullets a user is allowed to fire at a time. The boost button starts to move the ship forward. Pressing the boost button again will stop the ship from moving.

The calculations for the game are mostly written in standard Java, apart from a few calls to the Android API such as the Matrix class.

* Engine – This is the activity in which the game is contained
* GamePanel – This is a SurfaceView used to draw what happens in the game
* DisplayThread – This is the thread which calls the update() and draw() methods for the game, allowing for all calculations to happen
* Joystick – This is a View which is used to take user input and use it to rotate the ship.

Unfortunately, due to difficulty with exiting the SurfaceView class, there was not enough time to let the user view their score in the ScoresScreen view, despite it being recorded.