**App Manual for ‘Cambridge LED Lab’**

**Contents:**

Things Left to Do ------------------------------------------------------------------------ 1

Adding a New Topic ------------------------------------------------------------------------ 2

Adding a New Experiment --------------------------------------------------------------- 4

Required Details when Uploading the App --------------------------------------------- 6

**Things to Do:**

* Add more classes
* See “Adding a New Topic”
* Add more experiments
* See “Adding a New Experiment”
* Add games
* Replace the button which goes directly to the YouTube video with a button that takes you to a page similar to that of the “Laboratory Page”, which contains games. This has already been created (though not complete), with the java file “GamesPage.java” and the layout file “res/layout/activity\_page\_games.xml”.
* The icon for the “Games” button has already been made and is in the drawable folder, so references to “res/drawable/ ic\_button\_video.png” should be replaced with “res/drawable/ic\_button\_games.png”, this should only be found in “java/<package path>/HomePage.java”
* A layout file called “res/layout/list\_item\_search\_games.xml” should be created for search results that are games and this should be inflated and adapted in “java/<package path>/SearchResultsAdapter.java”, when the SearchElement has type “GAME”. This will probably be similar to the “res/layout/list\_item\_search\_experiment.xml”, which is the layout for search results that are experiments.
* Change live image view on home page
* The live image view should display images that are related to specific classes (when enough are created) and should open the related class when clicked.
* Code related to this is currently in “java/<package name>/HomePage.java”

**Adding a New Topic:**

* Create the text that will accompany each page in each class, and add to app as string in “res/values/strings.xml”
* The text on each page should be no longer than about 400 characters long.
* Each string should be named sensibly to keep order when there are a larger number, such as “ClassText\_<topic name with no spaces><level>\_Page<page number>”, where the level is 1, 2 or 3 and the page number increments from 1.
* Create the image or animation for each page, and save it to drawable folder.
* The resolution should be good enough for tablets as well as phones (e.g. 1080 x 1080 pixels)
* Each one should be a .gif file whether its animated or not.
* Most of the backgrounds should be transparent to keep a uniform style across the classes.
* Each file should be named sensibly to keep order when there are a larger number, such as “animation\_< topic name with no spaces >\_<level>\_ <page number>”, where the level is 1, 2 or 3 and the page number increments from 1.
* The files should be saved to the folder: “res/drawable” (note: the drawable folder does not allow you to name files with capitals or spaces in)
* Create an icon for the topic and save it to the drawable folder
* These should be round in shape and most should keep to the grey-blue colour scheme of previous icons
* It’s probably best to save it as 24-bit .png file with transparency.
* The resolution should be good enough for all possible devices (e.g. 720x720 pixels)
* The icons should be named sensibly (e.g. “ic\_class\_<topic name>”, ic is frequently used for icons), and saved in the “res/drawable” folder.
* Ensure this icon looks good on the list of classes and the search results list
* Add the topic to the list of topics
* Open the TopicInformationInitialser.java file
* Create an array of strings that are the names of topics that are expected to be known before starting this new topic.
* Create an array of 3 integers which are the number of pages in the first, second and third class respectively.
* For each of the three classes, create an array of references to the animations/images in the drawable folder. These must be in page order, and the references have the form “R.drawable.<file name excluding .gif>”. The size of each array must match the corresponding page count.
* For each of the three classes, create an array of strings which are the texts that accompany each page. The strings are retrieved from the references in “res/values/strings.xml” using: “<context>.getResources().getString(R.string.<string name>”.
* The size of each array must match the corresponding page count.
* Create a new TopicInformation instance using “new TopicInformation(<topic name>, <reference to icon>, <expectations>, <page counts>, <animations for 1st class>, <animations for 2nd class>, <animations for 3rd class>, <strings for 1st class>,

<strings for 2st class>, <strings for 3st class>)”

* Add the new TopicInformation instance to the list of topics using “.add(<topic>)”
* Change the position of the ‘currently unavailable’ mark by changing the integer in the if statement in “ClassListAdapter.java” under the label “Add ‘currently unavailable’ foreground to unused topics”
* or delete the whole section to remove it completely

**Notes:**

* Unlike when adding new experiments, the classes are automatically added to the search lists when added to the TopicInformationInitialiser.java file.

**Adding a New Experiment:**

* Create a new activity
* In ‘Android Studio’, go to “file/New/Activity/Empty Activity”
* In “Activity Name”, use a sensible name for the new experiment, such as “Experiment<experiment name>”
* Ensure that the layout file will also be named something sensible (e.g. “activity\_experiment\_<experiment name>”), although the autogenerated name is usually quite good.
* Ensure “Launcher Activity” is unticked and “Backwards Compatibility” is ticked
* Ensure the package name matches the rest of the app (this should almost never be changed)
* Click “Finish”
* Create layout for new experiment
* Find the layout file for the new experiment in “res/layout” and open it up
* Now design the layout of the experiment. (it might be easiest to begin by using the “Design” view and then fine tune the layout with the “Text” view
* To keep consistency across all experiments, use the colour “palePink” in “res/values/colours.xml” as the background and copy across the top of other experiments which all contain a title banner and header with the name of the experiment.
* You may have to adapt the layout file for different screen sizes
* Add functionality to the experiment by adding to the corresponding .java file
* Find the java file for the new experiment in “java/<package path>” and open it up.
* Use the code from other experiments for the title banner and experiment header (to set fonts and click listeners). You will need to change this slightly by replacing instances of the other experiment with instances of the new one.
* Add unique code to the file to give the experiment some unique interactivity.
* Create an icon for the experiment button and save it to the drawable folder.
* Use the previous designs as a starting point to ensure that the same grey background is used
* The resolution should be good enough for all screen sizes (e.g. 720 x 720 pixels)
* It’s probably best to save it as 24-bit .png file with transparency.
* It should be named sensibly (e.g. “ic\_experiment\_<experiment name>”) in the “res/drawable” folder
* Add new experiment to “Laboratory Page”
* In “res/layout/activity\_page\_laboratory.xml”, find the first unused button (or add a new row if none are unused) and set the background of the button to the new design using: ‘android: background = “@drawable/<icon name>”’
* Give this new button an appropriate id value, such as “@+id/<experiment name>ExperimentButton”
* In “java/<package path>/LaboratoryPage.java”, adapt the code used for other experiments to add a click listener to the new button that opens the new experiment activity
* Create an icon for the experiment in the search results and save it to the drawable folder.
* Use the design for the button and adapt it slightly (not something completely different)
* This usually just means removing the background and increasing the size of what’s remaining
* The resolution should be good enough for all screen sizes (e.g. 720 x 720 pixels)
* It’s probably best to save it as 24-bit .png file with transparency.
* It should be named sensibly (e.g. “ic\_search\_<experiment name>”) in the “res/drawable” folder
* Add new experiment to the list of possible searches
* In “java/<package path>/PossibleSearchesInitialiser.java”, create a new array of strings, which are the tags used to find the experiment. Whilst capitals are accounted for (so use lowercase only), spaces are not and the can only be the result of a search if the text that is searched is contained within any of the tags.
* Create a new SearchElement instance using: “new SearchElement(“EXPERIMENT”, <tags>, <experiment name>, <reference to search icon>, <instance of experiment class>)”
* Add this to the list of possible searches using: “.add(<search element>), and this should be added in order of how it will appear in the search results list.

**Notes:**

* Each experiment is unique and so it is impossible to describe how to make each one, unlike classes where the all have a similar structure.
* You may want an experiment which contains more than one activity and/or fragments. In this case, make sure the experiment button in the “Laboratory Page” opens the desired activity when it is pressed and then open other activities from there when you want to. The design of the app such that in the title banner, there is always a button to go to the home page and a button to go to the previous page. This should be maintained when multiple activities are used for an experiment, but consider whether the back button should go to the “Laboratory Page” or the previous activity of the experiment (it will depend on whether you want the user to consider multiple activities in an experiment as 1 page or many). Multiple activities should be a last resort, but it is often unavoidable.

**Required Details when Uploading the App:**

* To access the Google Play Console, use the social media google account.
* Username: ‘cam.gan.group’
* Password: ‘socialmedia-camgan’
* Use the following details when building a signed apk for ‘LED Lab’:
* Keystore: ‘ledlab.jks’, in ‘Keystores’ folder
* Keystore password: ‘ledlabkeystorepassword2017’
* Key: ‘ledlabkey2017’
* Key password: ‘ledlabkeypassword2017’
* Use the following details when building a signed apk for ‘LED Switch’:
* Keystore: ‘ledswitch.jks’, in ‘Keystores’ folder
* Keystore password: ‘ledswitchkeystorepassword2017’
* Key: ‘ledswitchkey2017’
* Key password: ‘ledswitchkeypassword2017’