CIRCUITRY TROUBLESHOOTING
MOBILE APPLICATION

MAINTENANCE MANUAL

Prepared by:
Kyle Baxter *(Project Manager)*
Beverly Abadines *(Assistant Project Manager)*

Advisor:
Dr Arturo l. Concepcion

Section 1: File Structures

Here are the following folders and files within necessary for the functionality of the application:

[FOLDER] app/res/layout

Activity_assembly1.xml

- The creation of a diagram that based on the information that has been given from the data_input activity. It will show the proper diagram (in the future) that will show the corresponding diagram with the information that was correctly implemented. For the time being, it will show a test that the array has been passed through and by showing that a face is there proves that it exists.

Activity_componenslist.xml

- The input screen showing the correct components that the user will choose. By choosing the part, it would go over to the data input activity.

Activity_data_input.xml

- Enables the user to input the first position of the component as well as the second position of the component. The third input will show if the correct input that the user previously chose was correct.

Activity_grid.xml

- Displays the Elvis II board that is what the user is hoping to have to understand the rest of the application.

Activity_guide.xml

- A welcoming screen that will help the user understand the process of the application as well as give a general idea of what functions this application has.

Activity_main.xml

- The main screen of the application. Transfers to another activity through a button.

Activity_optionsboard.xml

- Gives organization throughout the Elvis II board by choosing the sections.

Content_main.xml

- The main menu screen that will either tell the user to go into the rest of the application or go to the user guide.

Splashh.xml

- Welcoming splash screen that shows the name of the application and then jumps for the activity main.

[FOLDER] app/java/com.example.micaflor.cs2

(Contains all java class files)

Assembly1.java

- Carries over the array that holds all of the string values to show and be manipulated on. It will be parsed for a string alphabet value as well as a integer value and will start to evaluate the connections based on the values.

Capacitor.java

- Specific class appointed to the capacitor that checks if its has been implemented in the right direction by the user. It will check the polarity as well as capacitance and then returns the actual capacitance as well as the polarity.

Componentslist.java

- Series of onclick listeners that will each will have intents connection to the data input class.

DataInput.java

- Enables the input from the edittext inputs to be added into an array and then goes through a series of self checks that will flag the user if certain errors are established.

Endpoints.java

- Strict-like class that will take the first end of the input and the second end of the other input and then group them together.

Grid.java

- Established the image to be zoomable (in the future) along with another button to send to the next activity.

Guide.java

- Shows the layout which has the text to guide the user.

LED.java

- Specific class appointed to the LED that checks if the polarity as well as the position is correct. After that, creates an array that has the destinations as well as the flags within.

Logic.java

- This class will take the inputs made from the data input class and will parse through for the possible values that will be used to connect to each other and send over to the assembly class.

Mainactivity.java

- Main menu class that establishes two button that either takes you to the user guide class or the grid class.

[FOLDER] apps/srs/main/res/values

- Contains all of the strings.xml, dimes.xml, and styles.xml files

Section 2: Instructions

Here is how to acquire,build, and deploy the application:

1) Acquire the zip files that hold the source files, either given from the project manager or the repository.
2) Run Android Studio (compiler version 1.5) and do the following:
    a) From menu, select import existing project
    b) Let the gradle finish building and it should show the main activity rendering
3) To compile and build, do the following:
    a) Press the Run button in the toolbar above
    b) In addition, Android Studio will ask you to choose to deploy the application to an emulator on your computer or to a connected device. (NOTE: Due to the emulator being slow during the loading process, the team advises using a connected device for testing and/or using)
4) To deploy the application, make sure that the connected device has its developer options set as well as debug mode from those options. The emulator doesn't have to go through this process but by running the emulator, one has to wait for the application to load.

Section 3: Good Implementations

The following are some of the positive and novel implementations of the application:

- The application succeeds in proper design representation of the kind of circuit boards that the user will be using when need help through connections of a circuit board.
- Array manipulation through the data input as well as converting the contents to the array into a listview that the user can observe and see.
- The application is simple to use through a beginning user guide as well as guided buttons provided by the UI team and our artist, Hyundeuk (Richard) Cho.

Section 4: Needs Improvements

- There are still some checks that are needed to go through when it comes to the input that the user puts in.
- If the user accidentally presses the add to list button without providing the input, the program crashes.
- If the user inputs the positions without a space that's in between, the program crashes.
- If the assemble button is pressed, there is no progress towards providing the input to diagrams step, but only providing information that the array is read clearly.

Current progress is being made to fix these bugs before the initial presentation stage but they are still knowledgeable.

Section 5: Overall Recommendations

Regarding future points needed to be considered in order for future teams to continue development without any waste of time or misunderstanding of information.

- Input the correct diagrams read from the input of the user to consider proper debugging of their own circuit.
- Consider the user input to be more graphical instead of manual input.
- Immediately research tutorials as well as informational guides on android studio development as well as java class programming.