**EC327**

**Documentation for the Connect4 App**

**Team: Victorious Secret**

**Table of Contents**

Introduction……………………………………………………………………………………….3

Description of Work Progress………………………………………………………..……………4

Front End Documentation………………………………………………………………………5-6

Back End Documentation………………………………………………………………….……7-8

Marketing……………………………………………………………………………………….....9

Team Statement of Work……………………………………………………………………..….10

**Introduction**

This Android application is a game called Connect4. The game requires a two person input to play, with each player inserting markers into a six by seven slot board (this makes the board array consist of six rows and seven columns). Markers are placed into columns, and each successive marker addition into the column will stack on existing markers, up to seven times when the rows are all filled. The game has the simple objective of getting four markers lined up tangentially to win, which can be oriented with four markers in a row, four markers in a column, or four markers in a diagonal line. The first player to obtain a set of markers in one of these orientations is the winner.

**Note:** The game includes a hack: Any player can “control” the game since by hitting a full column that switches the current player’s piece to the opponent’s piece, so the game can be swayed in a person’s favor.

**Progress of Work**

Work was done one a regular basis to accommodate everyone’s schedules, usually during the weekends on Friday, Saturday, and Sunday. The first initial and informal meetings consisted on deciding what to do prior to the actual project development and splitting up roles among the group. Once the project was chosen upon and times were set, the progress of work went like so:

* Everyone with their assigned roles was to do research and figure out how to do their part of the code. This was for everyone to understand their role better and how it would relate to the overall project.
* The next step of the project involved figuring out how the Connect4 game would work and how it should be set up. The group decided upon making a Connect4 game with an introductory screen and then it would lead to the actual game board.
* The actual programming involved everyone coming together and piecing together everyone’s parts into a successful app. This is the most difficult part of the project as getting all the different pieces of code together often results in errors which need to be checked and removed to allow compilation and prevent runtime errors on the application.

**Front End Documentation**

**Note:** The front end was supposed to include a main that would send an array representing the current game board and check for winning conditions to the back end. Unfortunately run time errors occurred which are documented in the ‘Back End Documentation’ and are not included in this documentedversion of Connect4.

The front end of the project consists of four main files that creates the GUI:

* Connect4.java
* activity\_connect4.xml
* Board.java
* activity\_connect4\_board.xml

**Connect4.java**  
This java file creates an introductory screen with a simple start button in the middle of the screen taken from the **activity\_connect4.xml file**. The user hits the start button and this action leads to another page with the actual Connect4 game with a grid image in the background with a reset button at the bottom.  
**activity\_connect4.xml**  
This xml file displays the start button on the introductory display.  
**Board.java**  
This is the program where the actual Connect4 game is run from. Each column is a long transparent button, where a player clicks the column, he/she clicks the button and a marker falls into the column, and then the next player begins his turn. In each column, there is a count associated with the column and the button is no longer operable after six hits, which is the maximum number of times that column can be accessed in the Connect4 game as the column will be full of markers. When the button is clicked, the ‘public void onClick’ function operates by first figuring out the count number for the column, figuring out the precoded position for the marker, figuring out which player’s turn it is, and then placing the appropriate marker into a precoded position in the column. For each of the 42 slots, there is essentially a function that will determine the appropriate marker to place into the slot. At the bottom of the activity screen there is a reset button where anytime during gameplay, which removes all the markers from the board and resets the game.  
**activity\_connect4\_board.xml**  
This xml file makes the grid view for the Connect4 game, with 42 slots. The xml also makes the six column buttons transparent so the user will not see them when they are clicked. there is also the reset button that is present. There are 42 ImageView activities, each transparent and hiding a slot where a marker can be placed.

**Back End Documentation**

**Note:** The backend does not actually work on the actual application. It encounters several runtime errors after the introductory screen preventing anyone from access from the actual game. The original backend was supposed to work with a main in the frontend, but unfortunately the main does not work and is commented out in the front end.

**The Error Log is as followed:**

12-10 20:31:24.989: E/AndroidRuntime(2483): FATAL EXCEPTION: main  
12-10 20:31:24.989: E/AndroidRuntime(2483): java.lang.RuntimeException: Unable to start activity ComponentInfo{com.example.connect4/com.example.connect4.Board}: java.lang.NullPointerException  
12-10 20:31:24.989: E/AndroidRuntime(2483): at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:1955)  
12-10 20:31:24.989: E/AndroidRuntime(2483): at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:1980)  
12-10 20:31:24.989: E/AndroidRuntime(2483): at android.app.ActivityThread.access$600(ActivityThread.java:122)  
12-10 20:31:24.989: E/AndroidRuntime(2483): at android.app.ActivityThread$H.handleMessage(ActivityThread.java:1146)  
12-10 20:31:24.989: E/AndroidRuntime(2483): at android.os.Handler.dispatchMessage(Handler.java:99)  
12-10 20:31:24.989: E/AndroidRuntime(2483): at android.os.Looper.loop(Looper.java:137)  
12-10 20:31:24.989: E/AndroidRuntime(2483): at android.app.ActivityThread.main(ActivityThread.java:4340)  
12-10 20:31:24.989: E/AndroidRuntime(2483): at java.lang.reflect.Method.invokeNative(Native Method)  
12-10 20:31:24.989: E/AndroidRuntime(2483): at java.lang.reflect.Method.invoke(Method.java:511)  
12-10 20:31:24.989: E/AndroidRuntime(2483): at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:784)  
12-10 20:31:24.989: E/AndroidRuntime(2483): at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:551)  
12-10 20:31:24.989: E/AndroidRuntime(2483): at dalvik.system.NativeStart.main(Native Method)  
12-10 20:31:24.989: E/AndroidRuntime(2483): Caused by: java.lang.NullPointerException  
12-10 20:31:24.989: E/AndroidRuntime(2483): at java.lang.AbstractStringBuilder.replace0(AbstractStringBuilder.java:440)  
12-10 20:31:24.989: E/AndroidRuntime(2483): at java.lang.StringBuilder.replace(StringBuilder.java:637)  
12-10 20:31:24.989: E/AndroidRuntime(2483): at com.example.connect4.Board.onCreate(Board.java:134)  
12-10 20:31:24.989: E/AndroidRuntime(2483): at android.app.Activity.performCreate(Activity.java:4465)  
12-10 20:31:24.989: E/AndroidRuntime(2483): at android.app.Instrumentation.callActivityOnCreate(Instrumentation.java:1049)  
12-10 20:31:24.989: E/AndroidRuntime(2483): at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:1919)  
12-10 20:31:24.989: E/AndroidRuntime(2483): ... 11 more

The back end of the software uses C/C++ mainly to compare values to check winning/end-game conditions. The main Java file passes an array to the C source code so the back end can understand what the state of the game is like without having to handle persistence of variables that may not exist after an individual function call. The C/C++ code simply uses the array to compare to possible end-game conditions every time the function gets called. The way this differs from the C/C++ code we’re used to is that there is no Main running throughout the lifetime of the program. The back end just communicates to the Java file, telling it that an end-game condition has been met or not met. The Java end handles everything else until it is told not to.

**Marketing Documentation**

The target market for this game is for people who want to carry a portable version of Connect4 to have it available to play anytime and anywhere with a friend locally. Since everything today is shifting towards software, converting classical games like chess, checkers, and tic-tac-toe into software allows these popular classics to be a part of modern times, where people need not carry a board and pieces for play when it is all simply on a phone. People can simply go one with their day carry their favorite classical games and at a moment’s notice can play with a friend on their Android device.

While many apps are designed for people to play alone for their personal entertainment, the Connect4 game is designed for people to play with others and promotes a blend of a real social interface and software. Compared to playing with an AI, playing with another person feels much more enjoyable than playing with an AI that will play on calculated moves and is programmed to win. Unlike other apps which require the use of the internet for players to access the multiplayer function, this app will always allow two people to play without it, which is helpful in areas where no internet is available.

For the types of phone this app is available on is that it is for Android devices that have the API of 14, which while it is not the most updated version of Android available, it is one of the more advanced ones from the early days of the Android phones. The reasoning behind this choice is that most people have Android devices that are not from the early days of Android which technology has come away from and most people have stopped using, but most do not have the most advanced API either because those are included in the newest phones and people do not always buy the newest technology when it comes out because it is costly to upgrade.

**Team Statement of Work**

This statement of work declares that each member completed their assigned roles, assisted the group when necessary, and helped contribute to the semi-completion of the Android application.  
  
Vinny DeGenova-Java Styling and Backend help  
  
Collins Neill-Deconstruction and interfacing, Coding for Frontend and Backend  
  
Danny Taylor-Backend interface, C coding for Backend  
  
Raymond Li-Team leader and Documentation