ANDROID APPLICATION FOR USDA (U.S. DEPARTMENT OF AGRICULTURE) STRUCTURAL DESIGN SOFTWARE

by

NIKHITA ADDANKI

B.Tech, Gitam College of Engineering, 2010

A REPORT

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computing and Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2014

Approved by:

Major Professor
Dr. Mitchell L.Neilsen

Abstract

The computer industry has seen a growth in the development of mobile applications over the last few years. Tablet/Mobile applications are preferred over their desktop versions due to their increased accessibility and usability. Android is the most popular mobile OS in the world. It not only provides a world-class platform for creating several apps, but also consists of an open marketplace for distributing them to Android users everywhere. This openness has led to it being a favorite for consumers and developers alike, thereby leading to a strong growth in app consumption.

The main objective of the project is to design and develop an Android software application for USDA (U.S. Department of Agriculture) structural design that can be used on Android tablets. The different components of USDA that can be designed using this application are SingleCell, TwinCell, Cchan, Cbasin and Drpws3e.

The USDA (U.S. Department of Agriculture) structural design application was previously developed using FORTRAN. But FORTRAN is not supported by Android Tablets. So, F2J Translator software was used to convert the FORTRAN source files to java source files which are supported by Android. Also, many other formatters such as CommonIn, CommonOut, and SwapStreams were used to translate some common blocks of FORTRAN code that cannot be translated by F2J Translator.

The developed Android software allows users to access all different components of USDA structural design. Users can either directly enter the data in the forms provided or upload a file that already has data stored in it. When the application is run, the output can be accessed as a PDF file. Users can even send the output of a particular component to their personal email address. This output provided by the software application is helpful for design engineers to implement new structural designs.

Table of Contents

List of Figures

List of Tables

Acknowledgements

Firstly, I would like to thank my academic advisor, Dr. Mitchell L. Neilsen, for his excellent guidance and immense knowledge. I would also like to extend my thanks to Dr. Daniel Andersen and Dr. Torben Amtoft for their help and for serving on my committee.

I would like to acknowledge the academic support received from the staff and students of Kansas State University. Last, but not the least, I would like to thank my parents and friends who stood by me and encouraged me throughout my study period.

Chapter 1 - Introduction

1.1 Project Description

The basic idea of the project is to create an android application for Soil Conservation Service at the USDA (U.S. Department of Agriculture) (1) that annually designs a number of cast-in-place rectangular conduits for use in principal and emergency spillways passing through earth embankments. Thorough design of these rectangular conduit cross sections by manual methods is a time consuming process. So, initially, computer programs that execute the complete structural design of the rectangular conduit cross sections to design the task were written in FORTRAN for IBM 360 equipment. This project aims at developing an Android application (2) that can be used on tablet devices as usage of tablets has become preferable to that of desktop computers.

The project helps in developing different structural design components such as SINGLECELL (Single Cell Rectangular Conduits) (3) , TWINCELL (Twin Cell Rectangular Conduits) (4) , CCHAN (Reinforced Concrete Rectangular Channel) (5) , CBASIN (SAF Stilling Basin) (6) and DRPWS3E (Monolithic Straight Drop Spillways) (7). Since FORTRAN programs are not supported by Android devices, a FORTRAN to Java translator has been employed to adapt the programs for use on Android tablets. Once complete translation is finished, Graphical User interface is developed so that users can interact with the USDA android application.

1.2 Motivation

The programs written in FORTRAN can be used for Desktop Application. Now-a-days, use of tablets is very prevalent. Also, it provides increased usability, speed and accessibility. Android operating system is an open source and so it provides very good support to the developers thus enabling them to develop a variety of applications. As use of tablets is preferred to use of desktop computers, developing an Android application for USDA seemed more reasonable. To develop this application, the FORTRAN files (8) that were previously written should be converted to Java before it can be used on Android devices. It seemed more challenging and interesting as I can gain knowledge on basics of FORTRAN along with immense knowledge on Android application development.

Chapter 2 - Requirement Analysis

2.1 Requirement Gathering

Requirement gathering or elicitation plays a key role in developing a software product. The term elicitation means collecting requirements from customers and users. For USDA (U.S Department of Agriculture) Structural Design project, some of the requirements are collected from Dr. Mitchell L. Neilsen, Major Professor. Also, some documents related to Structural Design helped me in understanding the underlying design input and output requirements of the structural components: SingleCell, TwinCell, Cchan, Cbasin and Drpws3e.

The requirements for the project are:

- Graphical User Interface that supports Android devices should be developed in order to output structural design details for all the components.
- For all components, a provision should be made to either fill in the input data fields manually or upload a text file that auto fills the data fields of the input forms.
- The output should be displayed as a PDF file.
- The actual output should match the expected output (Expected output can be viewed in documents related to the components)
- The output PDF file could be mailed to the user.
- Project must run on Android tablets.

In order to implement these requirements, knowledge on Android application development is required. Also, as project involves FORTRAN translation, minimum knowledge of FORTRAN is beneficial.

2.2 Requirement Specification

2.2.1 Software Requirements

2.2.1.1 For F2J translation (FORTRAN to Java)

Operating System: Linux

Languages: FORTRAN, Java

Java Version: Java SE6

Tools: Eclipse Juno IDE, F2J Translator (FORTRAN to Java) (9)

Debugger: Eclipse Debugger

2.2.1.2 For Android application

Operating System: Windows 7

Languages: Java, XML, Android SDK

Tools: Eclipse Juno IDE

Technologies: Java, XML, Android

Debugger: Android Dalvik Debug Monitor Service, Android Samsung Tab 10.1

Framework: Android SDK version 3.0

2.2.2 Hardware Requirements

Processor: Pentium IV or higher

RAM: 512 MB

Disk Space: 250 MB or higher

Android Device: Any Android tablet with Operating system Honeycomb or higher

Chapter 3 - System Architecture and Design

3.1 System Architecture

Android is an operating system for devices such as smartphone and tablets. Android applications are developed in Java language with the help of Android SDK (Software Development Kit).

Figure 3.1 depicts the system architecture (10). It usually has a stack of layers and each layer has several components. Every layer in the stack provides services to the layer above it.

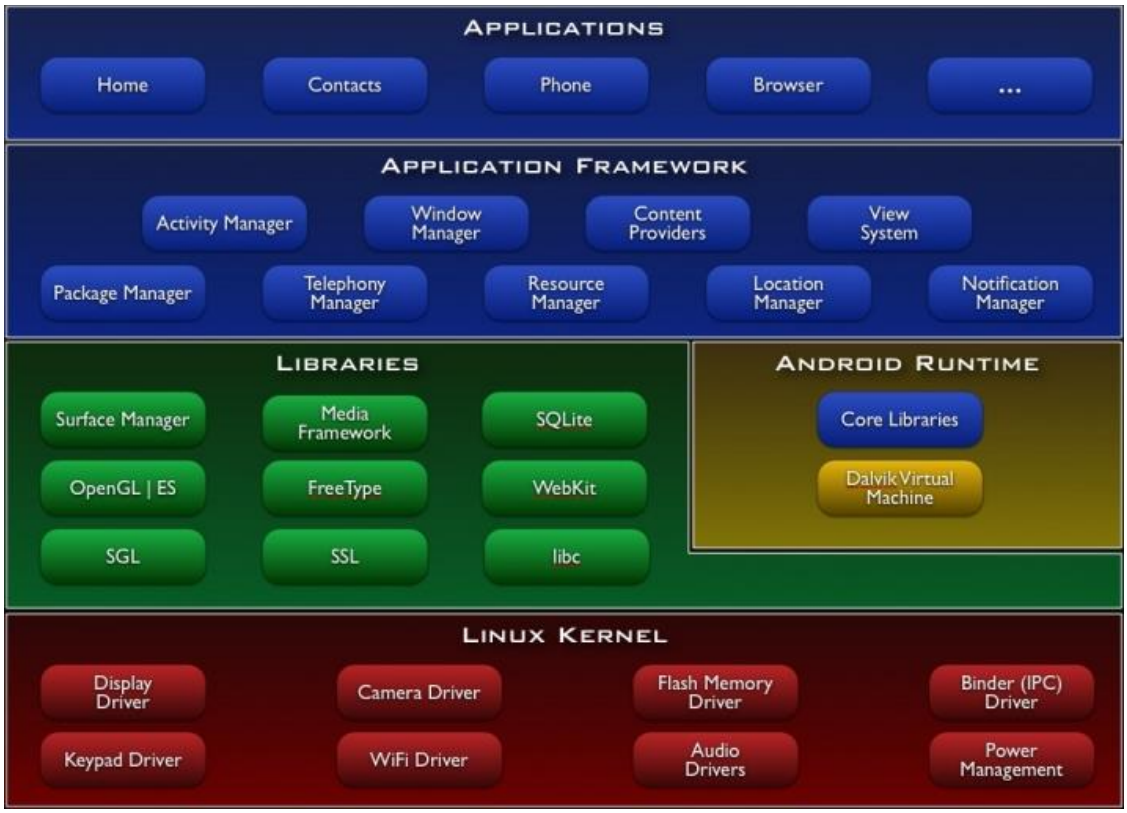

Figure 3.1 Android System architecture

The different layers are:

- Applications
- Application Framework
- Libraries
- Android Runtime
- Libraries
- Linux Kernel

3.1.1 Linux Kernel

The basic or lower most layer of the architecture is Linux Kernel. The whole Android operating system is built on this layer. It has many hardware drivers such as Display driver, Camera driver, Flash memory driver, Binder driver, Keypad driver, Wi-Fi driver, and Audio driver. These drivers help Linux interact with the hardware devices. This layer does not communicate with developers or end users. Also, this layer is very helpful for power management, memory management, security settings, networking and process management. As Android OS has many versions, Linux Kernel, on which Android runs, has evolved versions too. This Kernel is actually a part of Operating system that behaves as an abstraction layer between Android hardware and other software layers of the architecture stack.

3.1.2 Native Libraries

The layer above Linux Kernel is 'Libraries' (Figure 3.1). This layer helps the Android device handle many distinct types of data. There are different open source libraries written in C and C++ as follows:

- Surface Manager – compose windows on screen
- Media Framework – provides codecs to support recording, playback of audio and video
- SQLite – database engine to store data
- OpenGL|ES – 3D graphics library
- FreeType – render Fonts
- WebKit – web browser engine to display content of HTML
- SGL – render 2D graphics
- SSL – provide Internet security
- libc – System C libraries

3.1.3 Android Runtime

Android Runtime is on the same layer as Native Libraries, the layer above Linux Kernel.

- Core Libraries – core java libraries different from Java ME and SE libraries but include functionalities of Java SE libraries

- Dalvik Virtual Machine (DVM) – DVM is type of JVM machine used to run applications on Android devices as Android supports Java programming language. JVM runs .class files whereas DVM runs .dex files that are generated from .class files during compilation. Multiple instances of virtual machine can be created by DVM providing security, memory management, and isolation simultaneously. It takes the responsibility of running applications on Android devices.

3.1.4 Application Framework

Application Framework layer is above the Native libraries and Android Runtime layers. This is the layer which directly communicates with the applications developed by Android developers. It provides services to the applications in the form of .classes which can be used by developers. The different blocks of Application Framework are:

- Activity Manager – User interacts with the application by using the screen/interface that is provided by Activity Manager. There are different states in life cycle of Activity such as Start, Run, Pause, Stop, and Destroy. All these states are managed by Activity Manager.
- Window Manager – It not only organizes the processes and screens, but also defines how screens should be layered.
- Content providers – It facilitates exchange of data between applications.
- View system – Views in the windows are structured and this system is important for event handling.
- Package Manager – Manages installed packages and permissions related to the packages.
- Telephony Manager – Manage voice calls in case application needs to access voice calls.
- Resource Manager – Manages resources used in application
- Location Manager – Manages locations using Wi-Fi, GPS, cell tower
- Notification Manager – Manages all the notifications that show up if something unusual happens in the background.

3.1.5 Applications

The top layer of the stack of Android architecture is Applications. Android developers write applications in this layer only. All the end users of the application interact mainly with this layer of the architecture. There are some standard pre-installed applications such as Home, Contacts, Phone and Browser.

3.2 System Design

3.2.1 Use Case Diagram

The User acts as an actor. The different actions User can do are Run SingleCell, Run TwinCell, Run Cchan, Run Cbasin and Run Drpws3e. To run all these five components, User should first fill the input form which again requires user to select an input file. These two actions are considered <<include>>. User can fill the form and then save these inputs as a file which is indicated by 'Save input file'. So, this is indicated by <<extends>>. Once the components are run, User can view the output as PDF file and can also email the output as PDF. So, both these actions are considered <<extends>>.

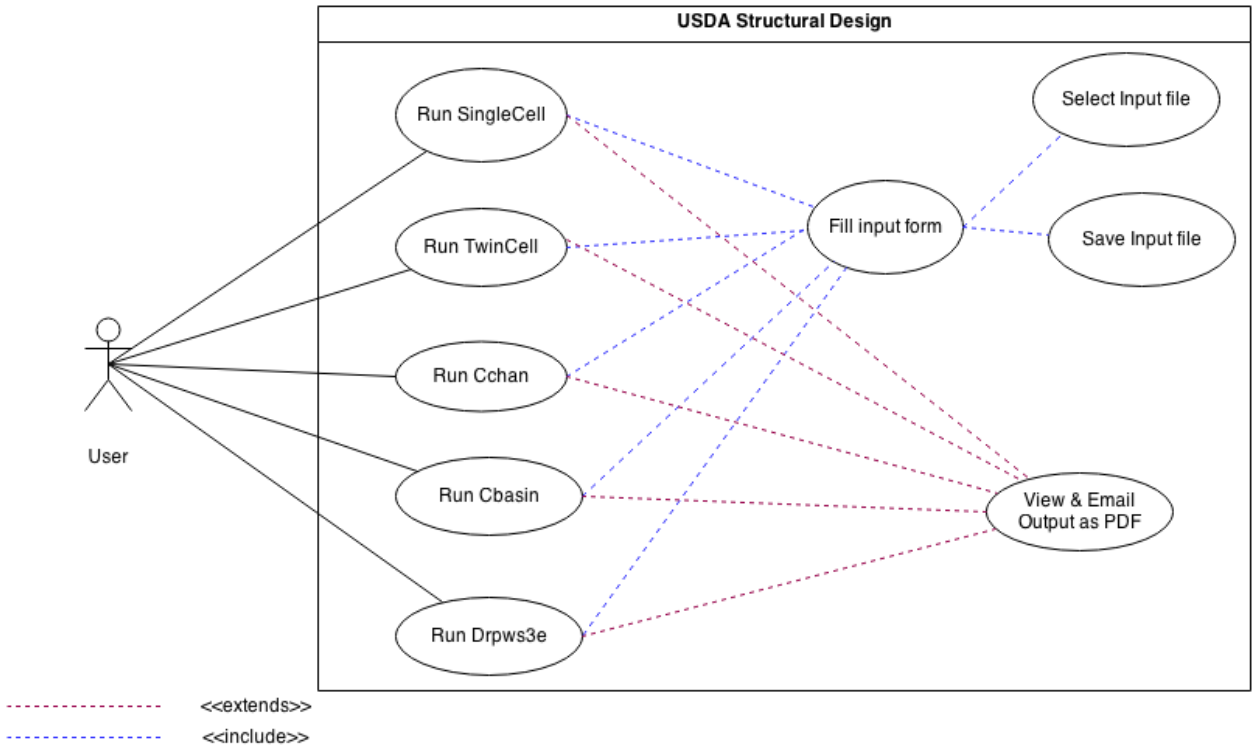

Figure 3.2 Use Case Diagram for USDA

Chapter 4 - Android Framework Components

4.1 AndroidManifest.xml File

AndroidManifest.xml is placed in the root directory of the project. This file provides information that the application should have before it can be run on the Android system. It declares the permissions that are required by the application to access any APIs (Application Programming Interface) or other applications. In USDA project, for every component designed, the input is a text file that can be selected from external memory. Also, the project needs internet in order to send an email to the user when requested. So, to access the external or internal storage and internet, permissions are declared with the <uses-permission> tag. It declares the 'minSdkVersion' and 'targetSdkVersion' required to run the application. It declares the activities that are required and linked to the application.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="ksu.USDADesignTools"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-permission android:name="android.permission.WRITE_INTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.READ_INTERNAL_STORAGE" />

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="18" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launch_usda"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="Main"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
```

Figure 4.1 Android Manifest XML (part 1)

```
        <activity
            android:name="ksu.USDADesignTools.MainFragActivity"</activity>
        <activity
            android:name="Singlecell"
            android:label="@string/singcell" ></activity>
        <activity
            android:name="Twincell"
            android:label="@string/twincell" ></activity>
        <activity
            android:name="Cchan"
            android:label="@string/cchan" ></activity>
        <activity
            android:name="Cbasin"
            android:label="@string/cbasin" ></activity>
        <activity
            android:name="Drpws3e"
            android:label="@string/drpws3e" ></activity>
        <activity
            android:name="OpenFile"
            android:label="@string/open" ></activity>
        <activity
            android:name="SaveFile"
            android:label="@string/save" ></activity>
    </application>
</manifest>
```

Figure 4.2 Android Manifest XML (part 2)

4.2 Activities

Activities (<activity>…</activity>) are embedded between the <application> tags. Only activities represented using <activity> tags can be run by the application and visually seen by the system. The Activity has attributes such as Name, label, icon etc.

Attributes used in Manifest.xml for USDA project are:

`android:name`

It determines the name of the class with which the activity is implemented. As there is no default for this attribute, the name should be specified.

`android:label`

It determines the text that is displayed on the screen. So, it helps the user know his location in the project. Label can directly be set to a raw string or can be accessed using @string/somename where somename is placed in strings.xml referring some string value. The USDA application developed has activities such as Singlecell, Twincell, Cchan, Cbasin, Drpws3e, OpenFile and SaveFile.

4.3 Intent

Intent allows users to request an action to be performed. It allows communication between components of the application. The components of the application can be activities, content providers, services etc. To launch or start an activity, Intent should be used with 'startActivity'. Intent has information such as action and data. The action is a string that determines the action to be performed, for example, ACTION_MAIN, ACTION_VIEW, ACTION_SEND, ACTION_EDIT etc. The data indicates the data on which the action is to be performed, for example, Data in Uri. While creating the intent, the type of data should also be defined.

USDA project uses Intents to view the Output generated by the structural components and email the same output as pdf.

```
File output = getFileStreamPath("Output.pdf");
Uri filepath = Uri.fromFile(output);
Intent outputIntent = new Intent(Intent.ACTION_VIEW);
outputIntent.setDataAndType(filepath, "application/pdf");
outputIntent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
try{
    startActivity(outputIntent);
}
catch(ActivityNotFoundException e2){
    Toast.makeText(Singlecell.this, "Please install PDF Viewer",
            Toast.LENGTH_LONG).show();
}
```

Figure 4.3 Intent Code to View Output as PDF

```
if(!singemail.getText().toString().trim().equals(""))
{
EditText emailet = (EditText) findViewById(R.id.singemail);
Intent emailIntent = new Intent(Intent.ACTION_SEND);
emailIntent.setType("text/plain");
String emails = emailet.getText().toString();
emailIntent.putExtra(Intent.EXTRA_EMAIL,new String[]{emails});
emailIntent.putExtra(Intent.EXTRA_SUBJECT, "SINGLECELL OUTPUT");
emailIntent.putExtra(Intent.EXTRA_TEXT, "SINGLECELL");

try
{
    File output_e = getFileStreamPath("Output.pdf");
    Uri output_epath = Uri.fromFile(output_e);
    emailIntent.putExtra(Intent.EXTRA_STREAM, output_epath);
    startActivity(Intent.createChooser(emailIntent, "Choose a client"));
}
catch (Exception e){
    Toast.makeText(Singlecell.this, "Sorry! Attachment Failed",
            Toast.LENGTH_SHORT).show();
}
}
```

Figure 4.4 Intent Code to Email Output as PDF

Chapter 5 - Implementation

5.1 FORTRAN to Java Translation

As discussed previously, Android operating system does not support FORTRAN language but it greatly supports Java programming language. As the basic idea of project is to create an Android application, the FORTRAN files that provide the structural design should be translated to Java source files to produce class files that can be used with JVM. For this purpose, F2J translator is used.

5.1.1 F2J Translator (FORTRAN to Java)

The primary motivation behind developing F2J (FORTRAN to Java) translator (9) was to convert numerical algebra software written in FORTRAN to Java class files. The formal compiler, F2J translator, translates programs that are written using subset of FORTRAN77 into programs that can be executed on JVM (Java Virtual Machines). Hence, the reference source code in the class files is translated from FORTRAN to Java. The F2J was developed not to translate the complete program, but to translate the FORTRAN linear algebra libraries. Some COMMON blocks, IO operations and string operations of the FORTRAN programs cannot be translated into Java. So, these components are removed from the files before translation. After the FORTRAN files (with selective blocks and operations removed) are translated successfully, the removed components are added back to the program.

5.1.2 Other Formatters

For complete translation of the FORTRAN programs, other formatters such as CommonOut, CommonIn, SwapStreams, CommentRemove must be used. A brief description of each of these formatters is provided below.

5.1.2.1 CommonOut

FORTRAN source files may contain many named and unnamed COMMON blocks. As the F2J translator is not designed to translate unnamed common blocks, it may not translate the program correctly. The CommonOut formatter helps in removing these blocks from the FORTRAN files. It takes the FORTRAN source file as input, removes the blocks from it and gives another FORTRAN file as output. The removed blocks are placed in a text file and the file

is given as another output. The text file also contains the name of the functions to which the removed common blocks belong.

The CommonOut formatter can be used as follows:

'Java CommonOut COA.FOR COB.FOR commonout.txt'

Here COA.FOR stands for the input FORTRAN source file, COB.FOR stands for the output FORTRAN file (after removing COMMON blocks), commonout.txt stands for the output text file with the removed COMMON blocks and the functions to which they belong. CommonOut is the java program that acts as formatter.

5.1.2.2 CommentRemove

The CommentRemove formatter removes all the comments from the FORTRAN source files. It takes in FORTRAN file as input and outputs another FORTRAN file. The input FORTRAN file should be the output FORTRAN file of CommonOut formatter.

The CommentRemove formatter can be used as follows:

'Java CommentRemove COB.FOR COC.FOR'

Here COB.FOR is the input of CommentRemove formatter and output of CommonOut formatter. COC.FOR is output of CommentRemove formatter. COC.FOR is used as an input for F2J translator. This FORTRAN file is then converted into JAVA files.

5.1.2.3 CommonIn

This CommonIn formatter replaces common global variables within Common.java and commonout.txt.

The CommonIn formatter can be used as follows:

'Java CommonIn COI1.java Common.java commonout.txt COI2.java'

Here CommonIn is the name of the formatter, COI1.java is the input java file (output file created from the F2J translator), Common.java is predefined java file which contains all global variables, commonout.txt is output text file from CommonOut formatter, and COI2.java is the output java file of CommonIn formatter.

5.1.2.4 SwapStreams

The SwapStreams formatter takes a java file as input and delivers another java file as output. The input java file is the output of CommonIn formatter. This formatter replaces all

“Util.f77write(” occurrences in the java file with “UtilStream.f77write(out,”. ‘UtilStream.f77write(out‘ implies the method ‘f77write’ of UtilStream class writes the output to the ‘out’, object of PrintStream. Usage of PrintStream helps in writing the output to any file. This formatter is used so that the translated output file from SwapStreams can use a method ‘f77write’ of UtilStream class.

This SwapStreams formatter is used as follows:

‘Java SwapStreams SSI.java SSO.java’

SSI.java is the input file to SwapStreams (translated java file and output of CommonIn formatter). SSO.java is the output file with translated ‘UtilStream.f77write(out‘.

5.2 F2J (FORTRAN to Java) Translation Process

The FORTRAN to Java translation is done using F2J (FORTRAN to Java) translator and the other formatters (CommonOut, CommentRemove, CommonIn and SwapStreams). The process of translation is same for all the projects: SINGLECELL, TWINCELL, CCHAN, CBASIN and DRPWS3E. The stepwise procedure is described below:

1. **CommonOut** formatter is used on FORTRAN files of all the projects. The FORTRAN file is given as input to produce another FORTRAN file and a text file as output. The output FORTRAN file does not contain any common blocks as they will be removed. The output text file contains all the removed common blocks.

   Usage: **Java CommonOut FILEI.FOR FILECO.FOR commonout.txt**

   FILEI.FOR – The input FORTRAN files can be SINGCELL.FOR, TWINCELL.FOR, CCHAN.FOR, CBASIN.FOR, or DRPWS3E.FOR.

   FILECO.FOR – The output FORTRAN files can be SINGCELLCO.FOR, TWINCELLCO.FOR, CCHANCO.FOR, CBASINCO.FOR, or DRPWS3ECO.FOR.

   commonout.txt – Individual commonout.txt file for every project.
2. **CommentRemove** formatter is used to remove any comments that are in the FORTRAN files. It takes FORTRAN file as input and produces a FORTRAN file (with comments removed) as output.

   Usage: **Java CommentRemove FILECO.FOR FILEO.FOR**

FILECO.FOR – The input FORTRAN files can be SINGCELLCO.FOR, TWINCELLCO.FOR, CCHANCO.FOR, CBASINCO.FOR, or DRPWS3ECO.FOR. (Output files of CommonOut formatter)

FILEO.FOR – The output file of CommentRemove formatter (SINGCELLO.FOR, TWINCELLO.FOR, CCHANO.FOR, CBASINO.FOR, or DRPWS3EO.FOR)

3. **F2J translator** is used to translate the FORTRAN files (Output of CommentRemove formatter) to java files.

   Usage: **f2java FILEO.FOR**

   FILEO.FOR - SINGCELLO.FOR, TWINCELLO.FOR, CCHANO.FOR, CBASINO.FOR, or DRPWS3EO.FOR

   After running the above command, it shows some warnings of parse errors. Parse errors can be due to Comments, Strings and DO loops. Once these errors are rectified, run the above step again to obtain an output.

4. **CommonIn** formatter should be used in order to reintroduce the commons that were placed in Common.java and commonout.txt back to java files.

   Usage: **Java CommonIn CII.java Common.java commonout.txt CIO.java**

   CII.java – input java file (w/o commons)

   Commonout.txt – output text file from CommonOut formatter

   CIO.java – output java file (with commons reintroduced)

5. **SwapStreams** formatter is used to replace f77Write statements to allow the output to be written to file instead of writing it to stdout.

   Usage: Java SwapStreams CIO.java CIS.java

   CIO.java – The input file to SwapStreams formatter (output java files of CommonIn formatter)

6. The f77 read statements should be replaced with proper java statements and so, that file is considered as input.
7. The project must be recompiled to create .class files
8. Run **javab *.class** on all the class files created in Step 7.
9. Complete translation of FORTRAN to Java is done.

5.3 Graphical User Interface

Graphical User Interface (GUI) is an interface that is important in software engineering. It helps users communicate with the devices easily. GUI also helps in enhancing the efficiency and usability of the developed project.

5.3.1 Main Page

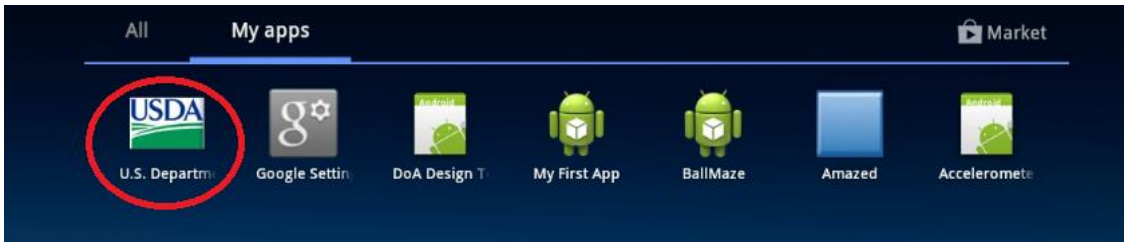

Figure 5.1 Launch icon for USDA project

The launch icon has the logo of USDA and the name of the project 'U.S. Department of Agriculture (USDA) Structural Design'. Main page is the first page that is displayed when the project is launched. The title bar of the main page also displays logo of USDA and the name of the project 'U.S. Department of Agriculture (USDA) Structural Design'. The page displays all the Structural Design Components developed in the project along with the logo of USDA. All circular buttons on the page relate to the components (SingleCell, TwinCell, Cchan, Cbasin and Drpws3e) developed. The button SINGLECELL, TWINCELL, CCHAN, CBASIN, and DRPWS3E navigates to the input forms of 'SINGLE CELL RECTANGULAR CONDUITS' (SINGLECELL), 'TWIN CELL RECTANGULAR CONDUITS' (TWINCELL), 'REINFORCED CONCRETE RECTANGULAR CHANNEL (CCHAN)', 'SAF STILLING BASIN (CBASIN)', and 'MONOLITHIC STRAIGHT DROP SPILLWAYS (DRPWS3E)' respectively. This page is designed both in landscape and portrait orientations. The layout is designed and saved as main.xml in 'Layout' folder of the project. Android considers same layout as default in both the orientations and picks the xml from 'Layout' folder. For the layout to be designed differently for landscape, xml is saved with the same name i.e. main.xml but in folder named 'Layout-land'. So, for landscape mode, main.xml is picked from 'Layout-land'.

Figure 5.2 Landscape Orientation of Main page

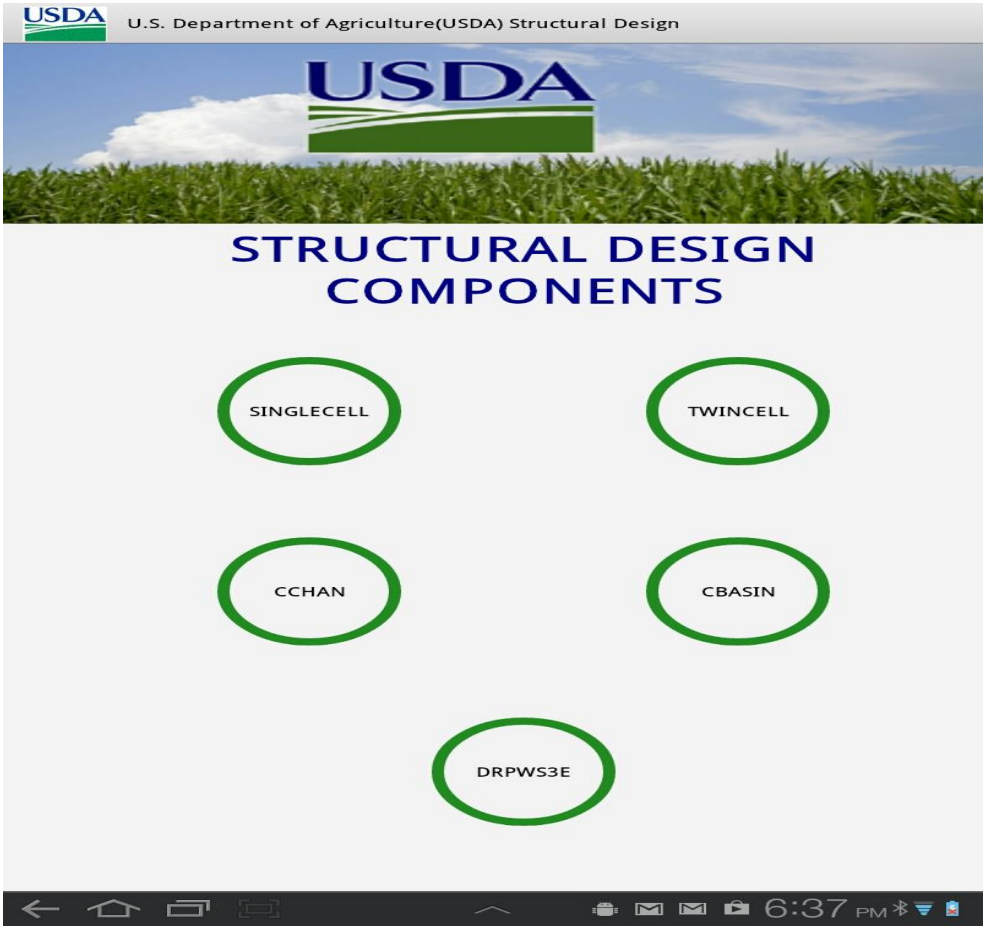

Figure 5.3 Portrait Orientation of Main page

5.3.2 Action Bar

The Action bar, one of the important design elements, provides an identity for the app built. It helps us know user's location in USDA application and in navigating to other sections of the project. The Action bar can have Menu and Submenu items. For every component developed (SINGLECELL, TWINCELL, CCHAN, CBASIN and DRPWS3E), the Action bar remains the same.

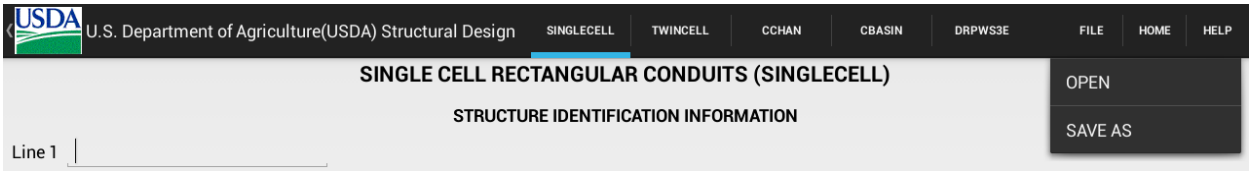

Figure 5.4 Action bar of USDA

For this project, the Menu and Submenu items are as follows:

- FILE – The user need not enter the values of all the fields manually. Instead, the user can select an input file of text format. The format of the input file can be seen in the HELP section.
    - OPEN – Once the user clicks on 'OPEN' submenu item, it navigates to the screen where the user is provided an option to select a file to be uploaded. Once the file is selected, the application fills all the fields with the data in the file appropriately. If the format of inputs in the file does not match the fields, the application shows an error and stops. The layout of the OPEN screen is written in open_file.xml and is placed in layout folder. The activity that is triggered on clicking OPEN is OpenFile.java.
    - SAVE AS – You can manually type in and save an input text file or you can fill in the fields of the form and then click on 'save as' to save the data in a text format. Once clicked on 'Save As', it navigates to a screen where user can save the file with any name. This file can be used again when the user clicks on 'OPEN' to fill in the fields with data. The layout of the SAVE AS screen is written in save_file.xml and is placed in layout folder. The activity that is triggered on clicking SAVE AS is SaveFile.java.
- HOME – This menu item helps the user to navigate to the main page of the USDA project. From this page, the user can again navigate to any section of the project.

- HELP – This menu item helps user get some information about the component the user currently is in. For instance, the user is in the component 'SINGLE CELL RECTANGULAR CONDUITS (SINGLECELL) as in the Figure 5.4, HELP navigates to the document related to SINGLECELL. If the user is in the component 'MONOLITHIC STRAIGHT DROP SPILLWAYS (DRPWS3E)', HELP navigates to the document related to DRPWS3E and so on.

5.3.3 SINGLECELL (Single Cell Rectangular Conduits)

The Singlecell.java program that is translated from SINGLECELL FORTRAN program using F2J translator and other formatters is used to execute the structural design of SINGLECELL. This program takes input clear width and height of conduit, load combinations and design mode. The program then determines thickness of slabs and sidewalls. The fields in the SINGLECELL input form are:

- Structure Identification Information
  - Line 1 – Optional Comment Line 1. Input type is Text.
  - Line 2 – Optional Comment Line 2. Input type is Text.
- Mode
  - Foundation Type – Input type is Radio button. Default value is 'Rock Foundation'.
    - Rock Foundation
    - Earth Foundation
  - Internal Water Load – Input type is Radio button. Default value is 'Yes'.
    - Yes
    - No

Table 5.1 Design Modes in Single cell

Design Mode	Foundation Type	Internal Water Load
00	Earth Foundation	No
01	Earth Foundation	Yes
10	Rock Foundation	No
11	Rock Foundation	Yes

- Dimensions – Input type is 'Numerical Decimals' and units is in ft. (feet).Both the fields are required.

    - Clear Height – height of the rectangular conduit.
    - Clear Width – width of the rectangular conduit.
- Load Combinations – Input type is 'Numerical Decimals'. All are required fields. Units in psf. (Pound per square foot). There are two combinations:
    - Combination 1
        - Max Vertical Unit Load (PV1)
        - Min Horizontal Unit Load (PH1)
    - Combination 2
        - Min Vertical Unit Load (PV2)
        - Max Horizontal Unit Load (PH1)
- Email (Copy of Output to be sent) – Optional field to enter an email id. The output on running the SINGLECELL is generated as a PDF and is sent to the email id provided.
- RESET – Input type is button. As the name says, when RESET is clicked, all the fields will be cleared and a new form is displayed.
- RUN SINGLECELL – Input type is button. Once the button is clicked, it validates the form if all the required fields are filled. If any field is left empty, it pops up an error dialog box asking to fill up all the fields in order to run the project. Once the form is validated, the output is generated as a PDF file.

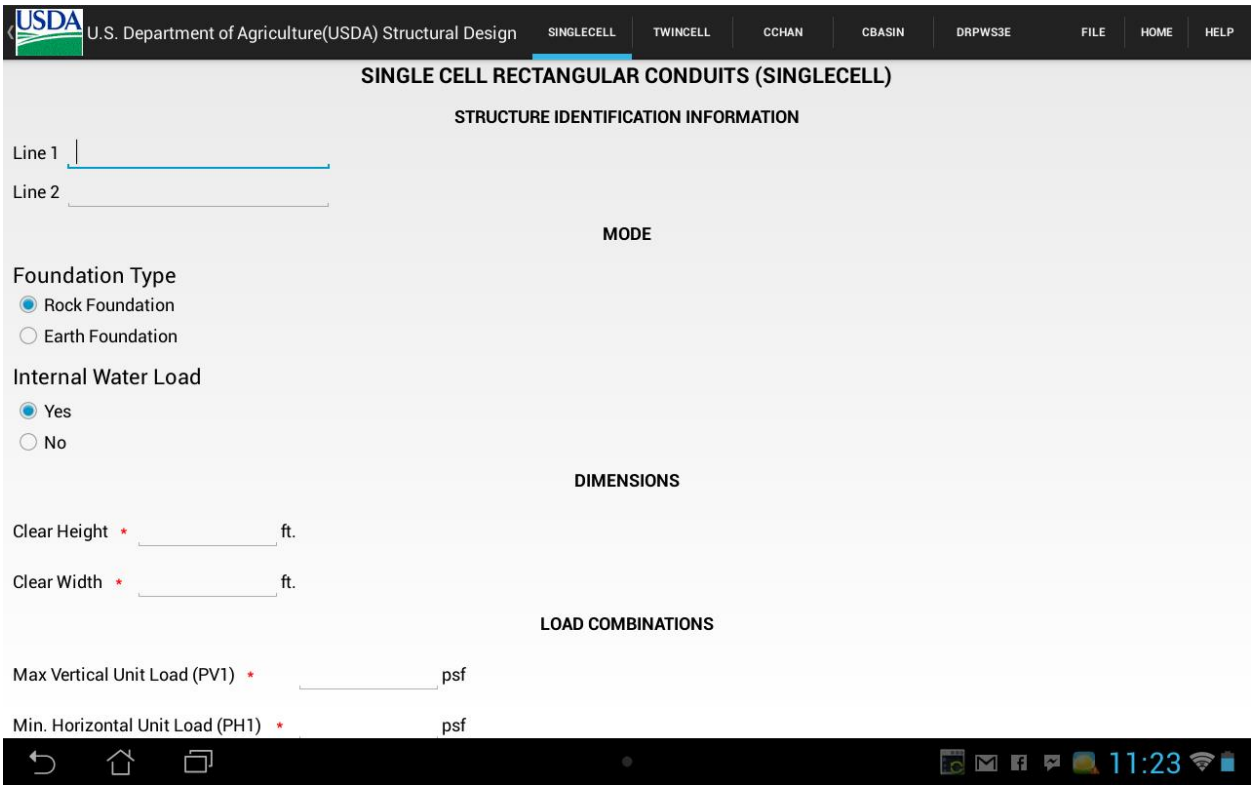

Figure 5.5 Single cell input form (Part 1)

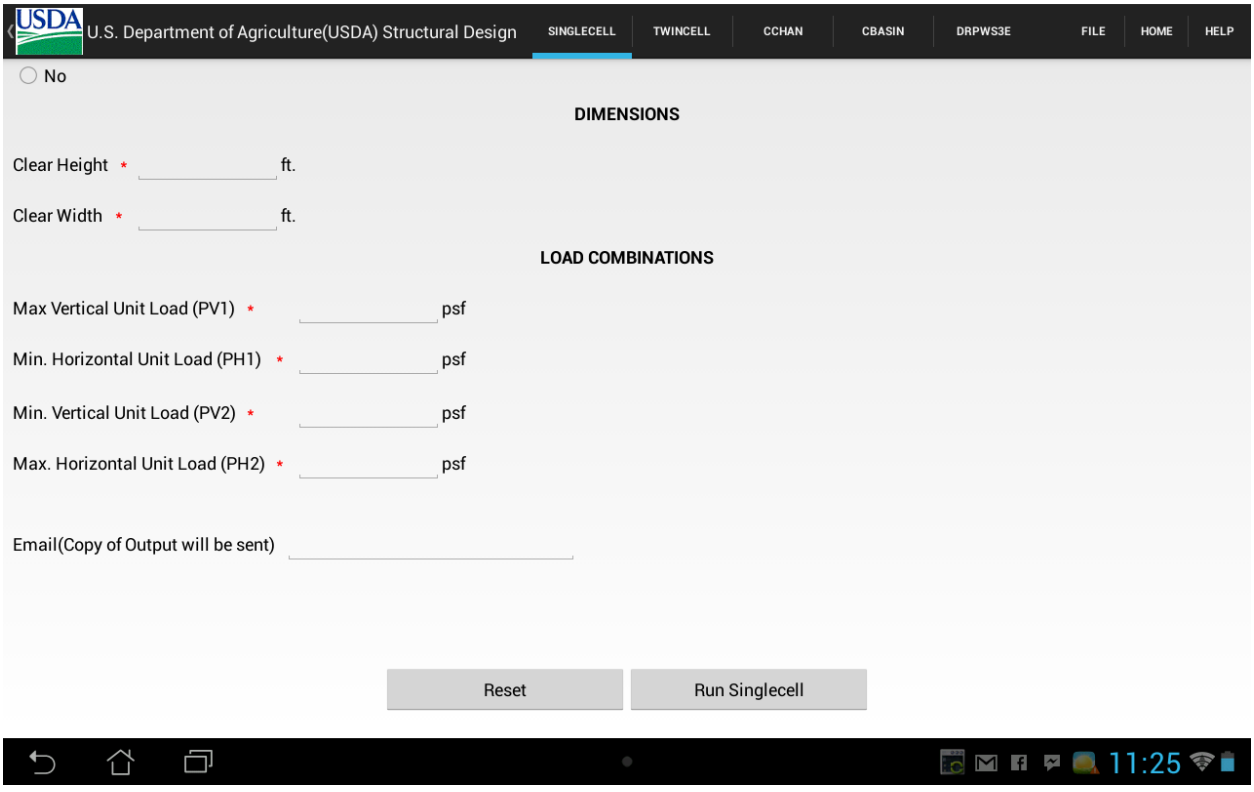

Figure 5.6 Single Cell Input Form (Part 2)

5.3.4 TWIN CELL (Twin Cell Rectangular Conduits)

The Twincell.java program that is translated from TWINCELL FORTRAN program using F2J translator and other formatters is used to execute the structural design of TWINCELL. This program takes input clear width and height of conduit, load combinations and design mode. The program then determines thickness of slabs and sidewalls and center wall. The fields in the TWINCELL input form are:

- Structure Identification Information
  - Line 1 – Optional Comment Line 1. Input type is Text.
  - Line 2 – Optional Comment Line 2. Input type is Text.
- Mode
  - Foundation Type – Input type is Radio button. Default value is 'Rock Foundation'.
    - Rock Foundation
    - Earth Foundation
  - Internal Water Load – Input type is Radio button. Default value is 'Yes'.
    - Yes
    - No

Table 5.2 Design Modes in Twin cell

Design Mode	Foundation Type	Internal Water Load
00	Earth Foundation	No
01	Earth Foundation	Yes
10	Rock Foundation	No
11	Rock Foundation	Yes

- Dimensions – Input type is 'Numerical Decimals'. Both the fields are required.
  - Clear Height – height of the rectangular conduit
  - Clear Width – width of the rectangular conduit
- Load Combinations – Input type is 'Numerical Decimals'. All are required fields. Units in psf. (Pound per square foot). There are two combinations
  - Combination 1
    - Max Vertical Unit Load (PV1)
    - Min Horizontal Unit Load (PH1)

- Combination 2
    - Min Vertical Unit Load (PV2)
    - Max Horizontal Unit Load (PH1)

- Email (Copy of Output to be sent) – Optional field to enter an email id. The output on running the TWINCELL is generated as a PDF and the same is sent to the email id provided.
- RESET – Input type is button. As the name says, when RESET is clicked, all the fields will be cleared and a new form is displayed.
- RUN TWINCELL – Input type is button. Once the button is clicked, it validates the form if all the required fields are filled. If any of the field is empty, it pops up an error message asking to fill all the fields to run the project. Once the form is validated, the output is generated as a PDF.

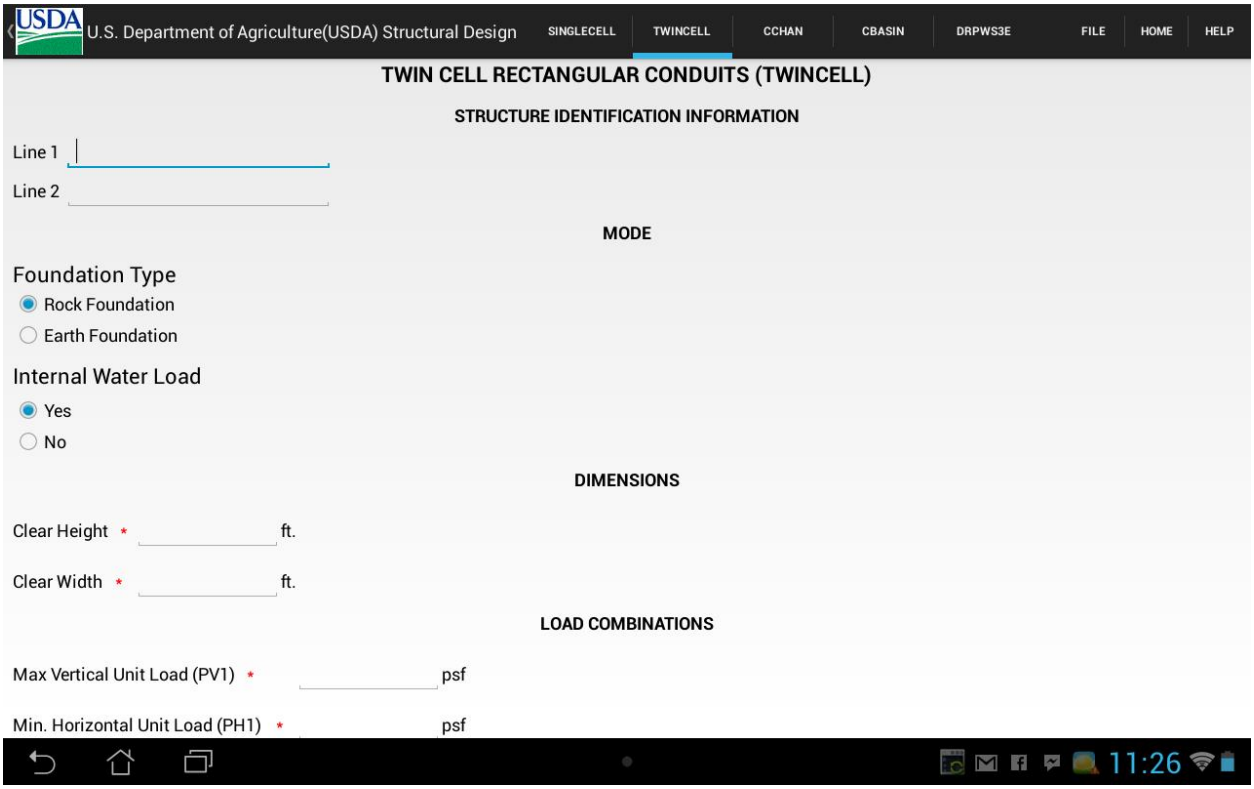

Figure 5.7 Twin Cell input form (Part 1)

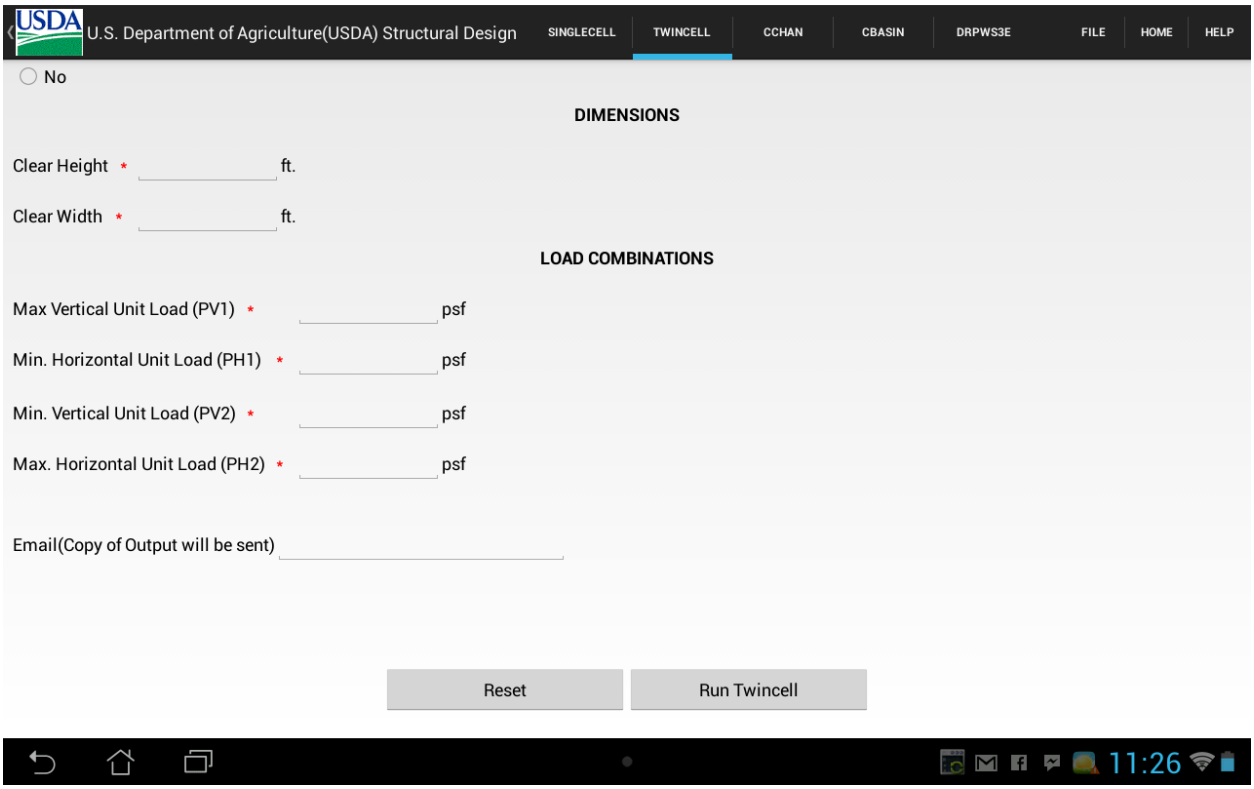

Figure 5.8 Twin cell input form (Part 2)

5.3.5 CCHAN (Reinforced Concrete Rectangular Channel)

The Cchan.java program that is translated from CCHAN FORTRAN program using F2J translator and other formatters is used to execute the structural design of CCHAN. There are four types of structural channels: T1F, T3F, T3FV and T1S. This program can be executed in two modes such as preliminary design in selecting type of structural channel and detail design of specified channel. This program takes input Primary data, secondary data and then determines concrete thickness and distances, evaluates required steel areas and spacing. The fields of CCHAN input form are:

- Structured Identification Information
    - Line 1 – Optional Comment Line 1. Input type is text.
    - Line 2 – Optional Comment Line 2. Input type is text.
- Primary Data
    - B (ft.), HT (ft.), HB (ft.) – Input type is ‘Numerical Decimals’. All fields are required and have units in feet.

- Design – Input type is ‘Radio button’. Only one design mode can be selected at once. There are five design modes such as 0, 1, 2, 3, and 4. If ‘Design’ = ‘0’, then four preliminary designs (T1F, T3F, T3FV and T1S) are performed. If Design is 1, T1F is performed. If Design is 2, T3F is performed. If Design is 3, T3FV is performed. If Design is 4, T1S is performed.
- Default 1, Default 2, Default 3 – Input type is ‘checkbox’. The user can select one or more Default types at once.

- Secondary Data
  - Default 1 – Input type is ‘Numerical Decimals’. All are required fields. These fields are displayed when Default 1 of primary data is checked.
    - HW1 (ft.), HW2(ft.) , HWP, GMOIST (pcf.), GSAT (pcf.), KO1, KO2, FLOATR
  - Default 2 – Input type is ‘Numerical Decimals’. All are required fields. These fields are displayed when Default 2 of primary data is checked.
    - MAXFTG (ft.), JOINTS, MFOUND (pcf.)
  - Default 3 – Input type is ‘Numerical Decimals’. All are required fields. These fields are displayed when Default 3 of primary data is checked.
    - CFSC, CFSS, KPASS
- Email (Copy of Output to be sent) – Optional field to enter an email id. The output on running the CCHAN is generated as a PDF and the same is sent to the email id provided.
- RESET – Input type is button. As the name says, when RESET is clicked, all the fields will be replaced by blanks and a new form is displayed.
- RUN CCHAN – Input type is button. Once the button is clicked, it validates the form if all the required fields are filled. If any of the fields is empty, it pops up an error message asking to fill all the fields to run the project. Once the form is validated, the output is generated as a PDF.

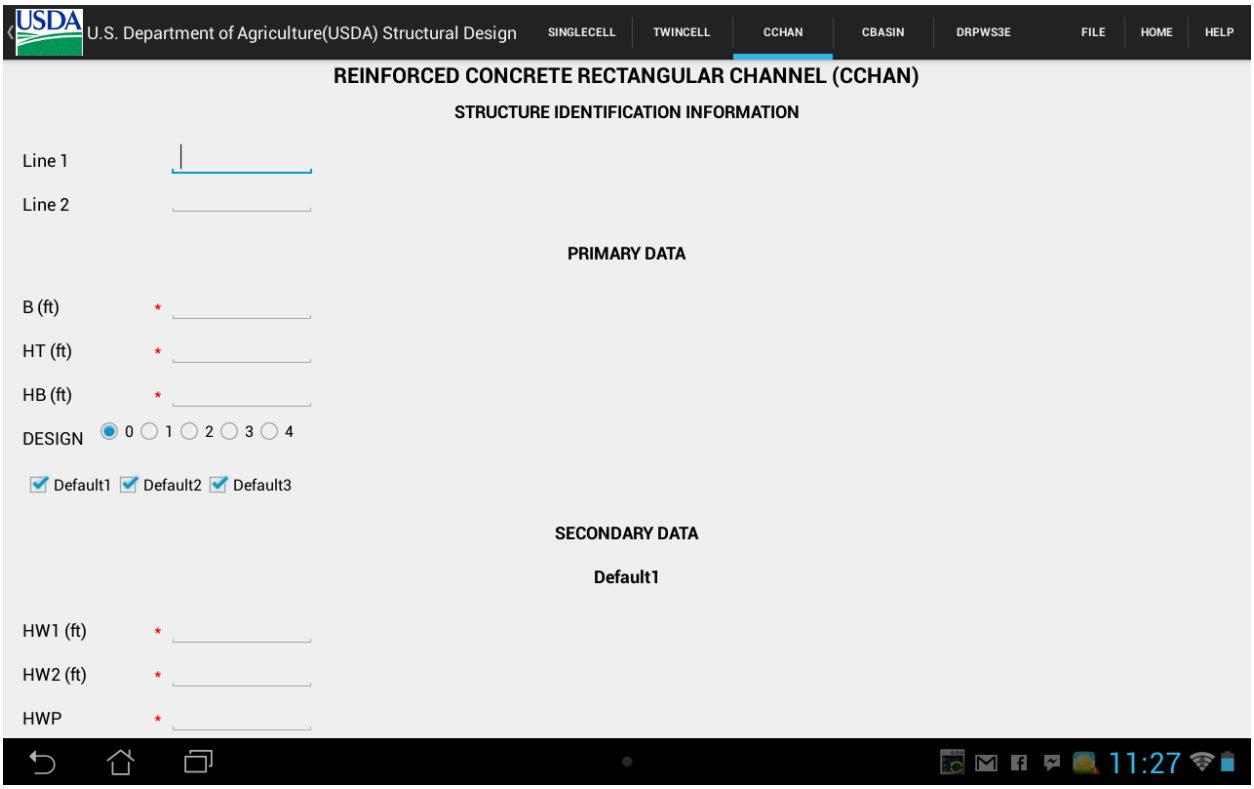

Figure 5.9 CCHAN input form (Part 1)

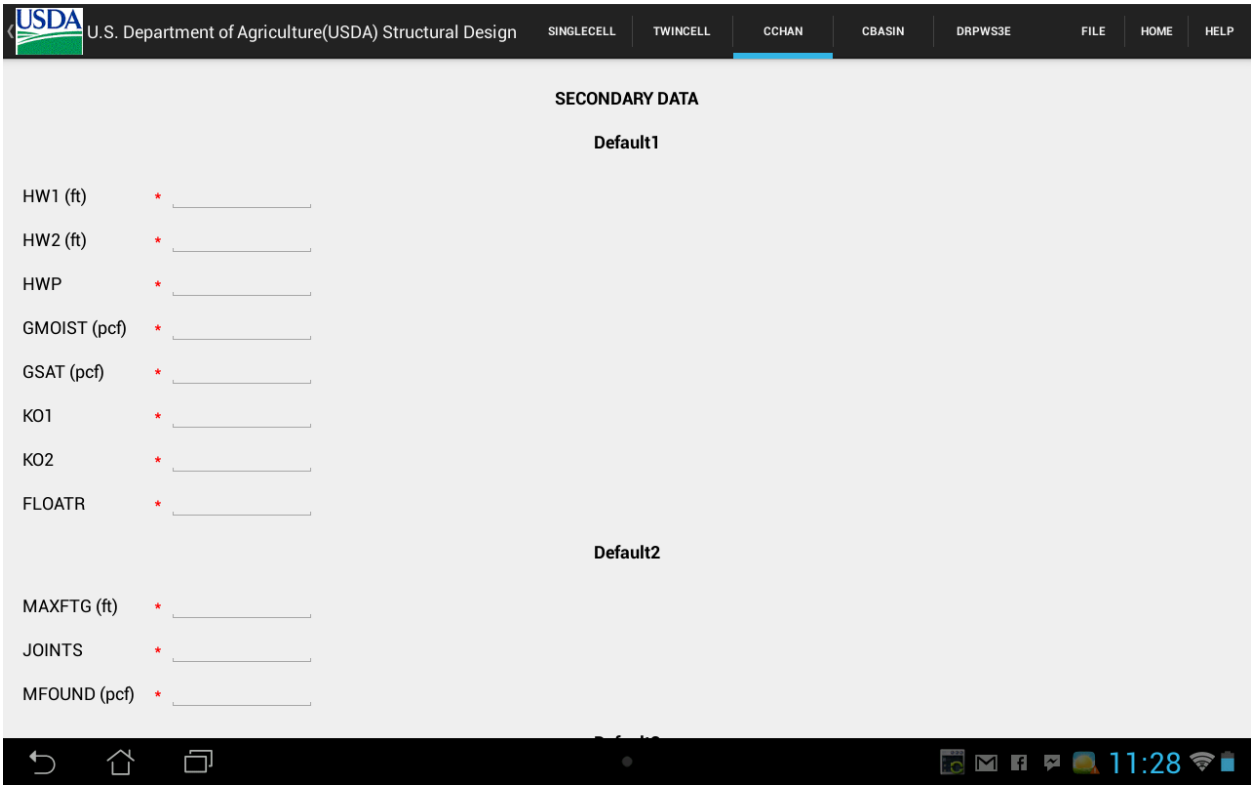

Figure 5.10 CCHAN input form (Part 2)

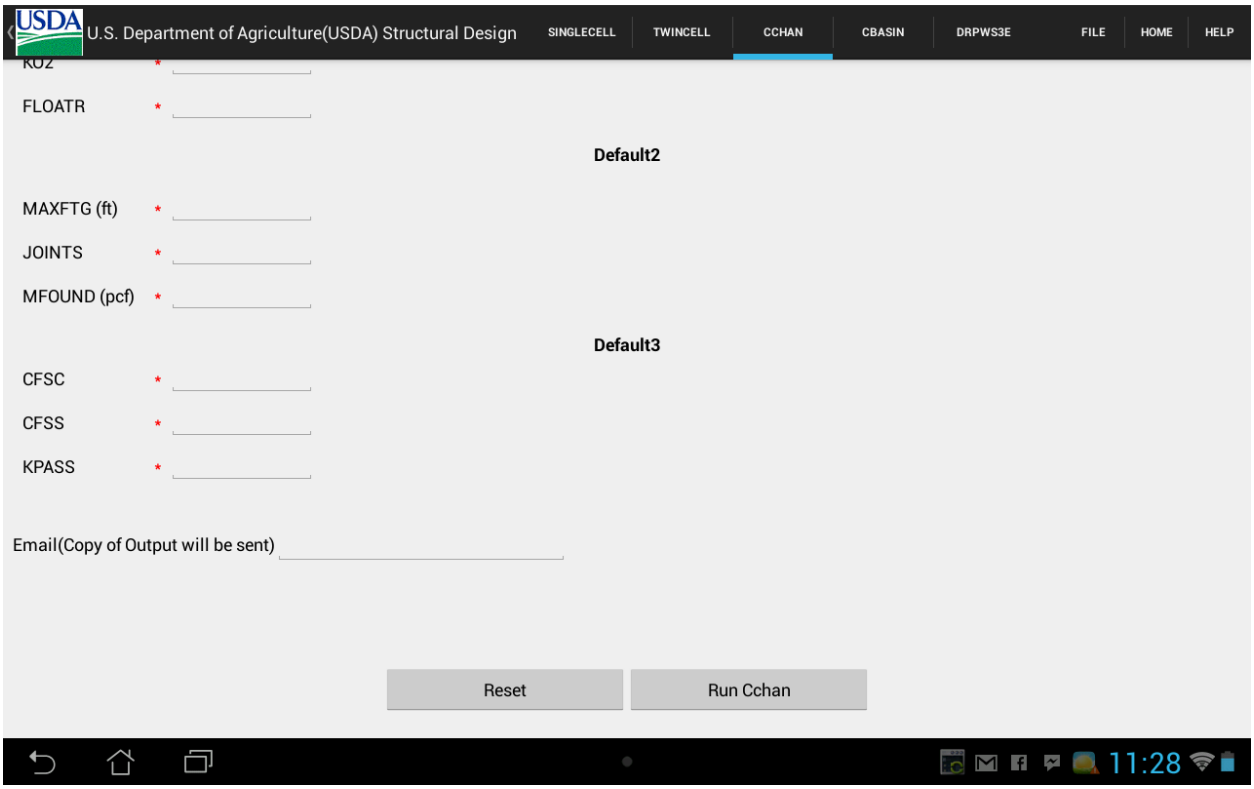

Figure 5.11 CCHAN input form (Part 3)

5.3.6 CBASIN (SAF Stilling Basin)

The Cbasin.java program that is translated from CBASIN FORTRAN program using F2J translator and other formatters is used to execute the structural design of CBASIN. There are four types of SAF stilling basins such as Type (A), Type (B), Type (C), Wingwalls. The program determines detail design of any stilling basin selected. The fields of CBASIN input form are:

- Structured Identification Information
    - Line 1 – Optional Comment Line 1. Input type is text.
    - Line 2 – Optional Comment Line 2. Input type is text.
- Primary Data
    - W (ft.), J (ft.), LB (ft.), N (ft.), D1 (ft.), V1 (fps) – Input type is 'Numerical Decimals'. All fields are required. (ft. : feet, fps : foot-pound-second)
    - Design – Input type is 'Radio button'. Only one design mode can be selected at once. There are four design modes such as 0, 1, 2, and 3. If 'Design' = '0', then three preliminary designs (Type (A), Type (B), Type (C)) are performed.

If Design is 1, Type (A) is performed. If Design is 2, Type (B) is performed. If Design is 3, Type (C) is performed.

  - Default 1, Default 2, Default 3 – Input type is ‘checkbox’. The user can select one or more default types at once.

- Secondary Data
  - Default 1 – Input type is ‘Numerical Decimals’. All are required fields. These fields are displayed when Default 1 of primary data is checked.
    - HB (ft.), HTW2 (ft.) , HUP2 (ft.), HTW1 (ft.), HUP1 (ft.)
  - Default 2 – Input type is ‘Numerical Decimals’. All are required fields. These fields are displayed when Default 2 of primary data is checked.
    - MAXFTG (ft.), FLOATR, SLIDER, ZS, BAT (ft.)
  - Default 3 – Input type is ‘Numerical Decimals’. All are required fields. These fields are displayed when Default 3 of primary data is checked.
    - GM (pcf), GS (pcf), KO, CFSC, HTW (ft.), TTW (in.)
- Email (Copy of Output to be sent) – Optional field to enter an email id. The output on running the CBASIN is generated as a PDF and the same is sent to the email id provided.
- RESET – Input type is button. As the name says, when RESET is clicked, all the fields will be replaced by blanks and a new form is displayed.
- RUN CBASIN – Input type is button. Once the button is clicked, it validates the form if all the required fields are filled. If any of the fields is empty, it pops up an error message asking to fill all the fields to run the project. Once the form is validated, the output is generated as a PDF.

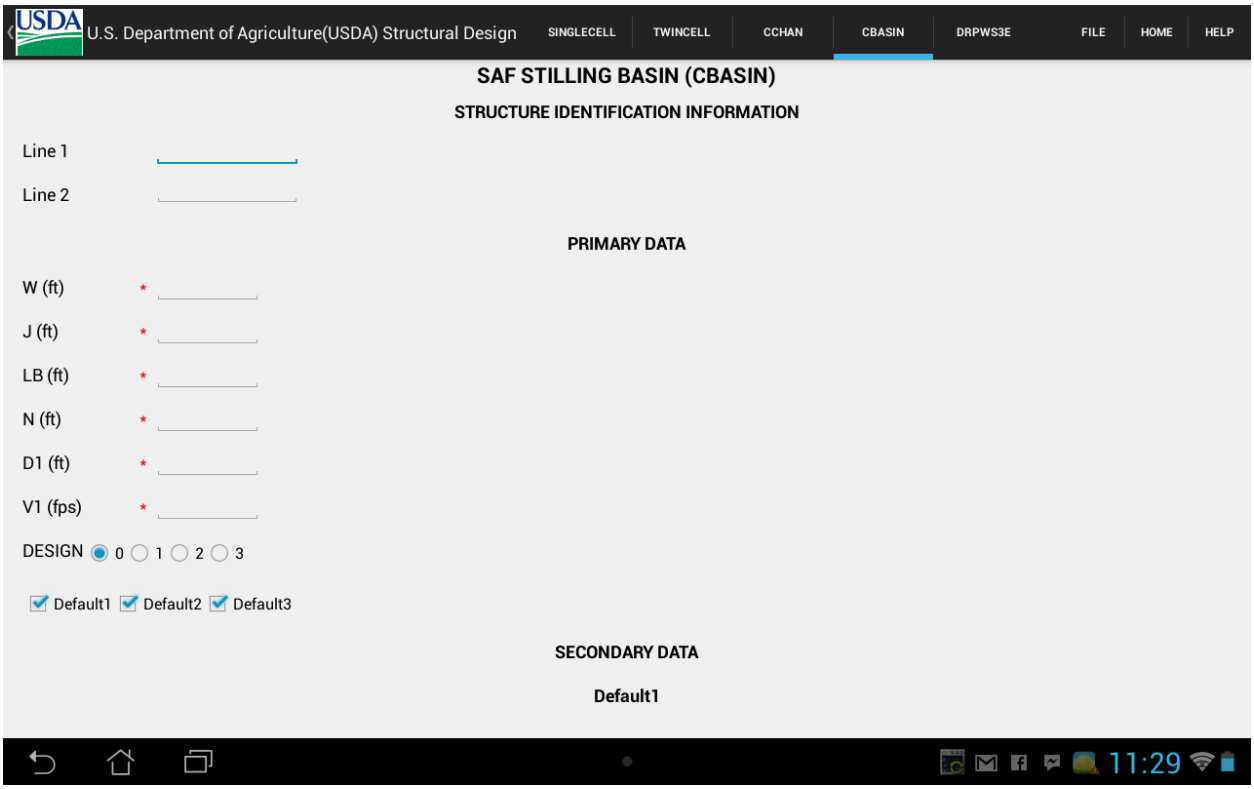

Figure 5.12 CBASIN input form (Part 1)

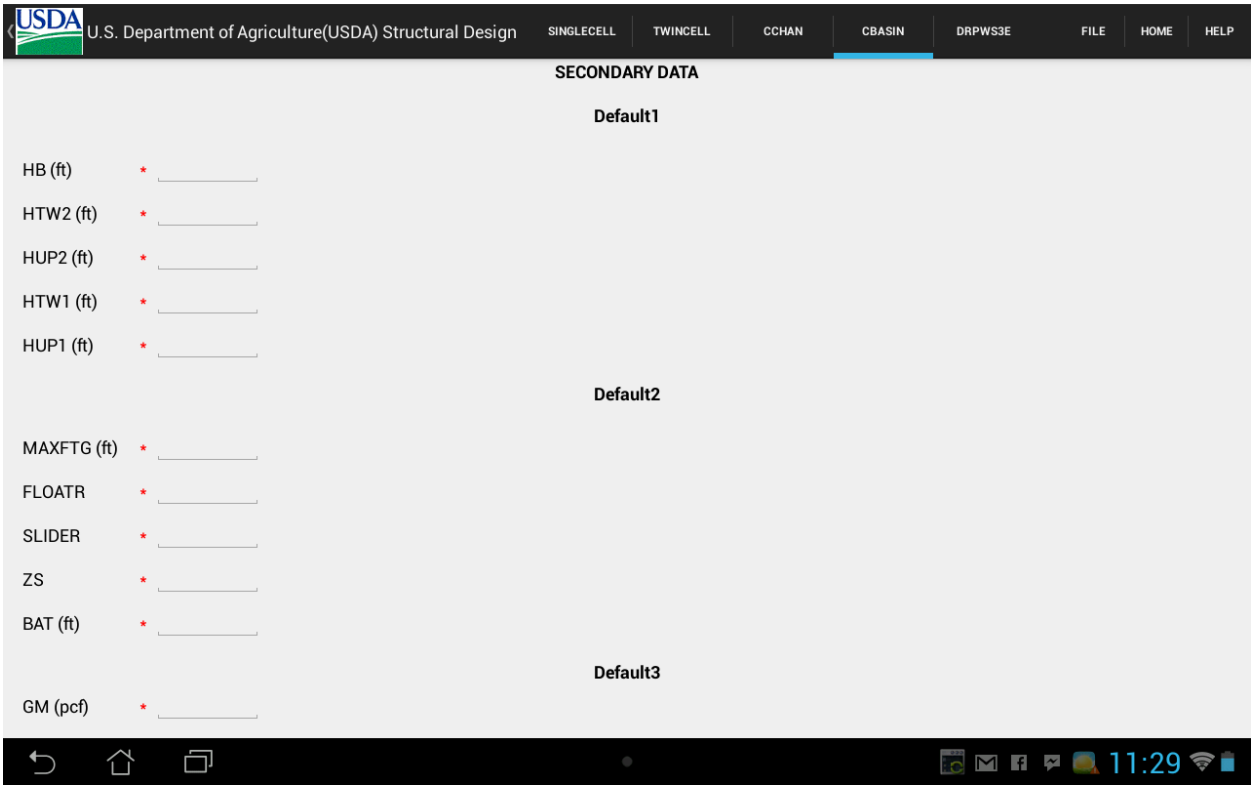

Figure 5.13 CBASIN input form (Part 2)

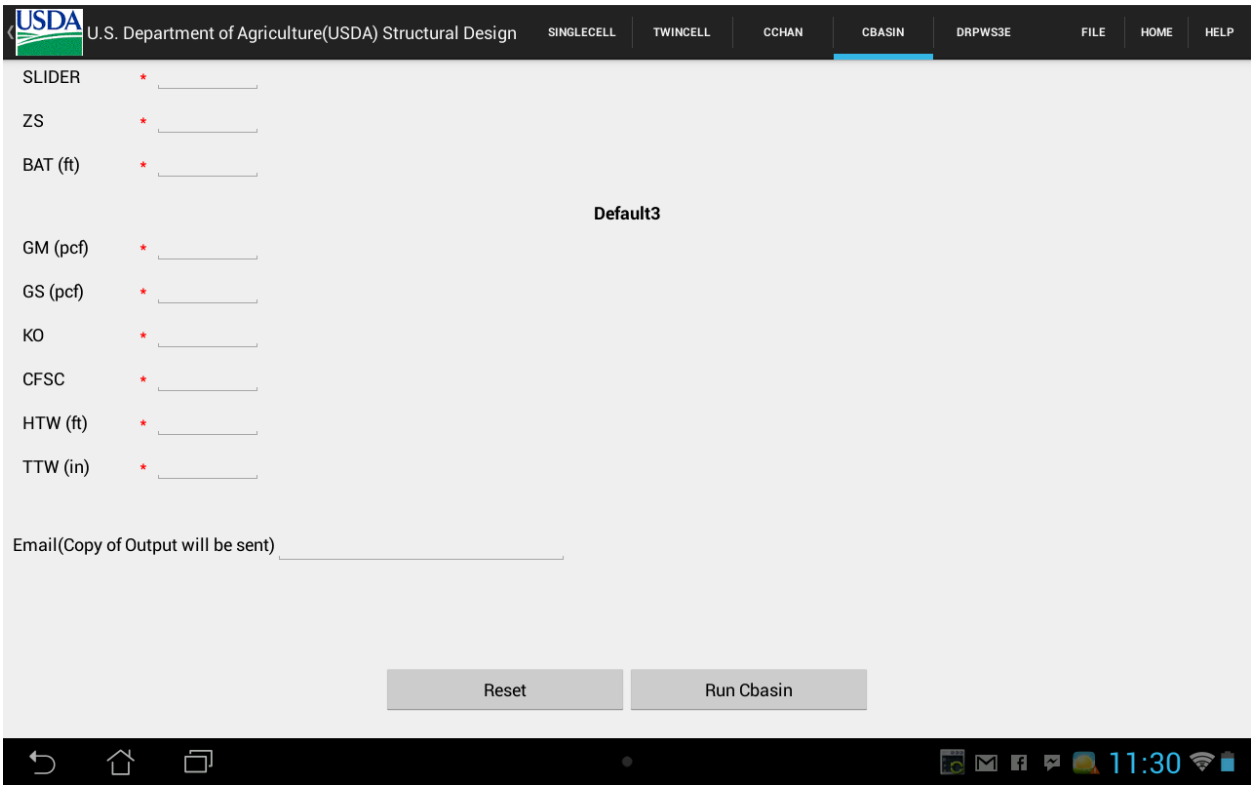

Figure 5.14 CBASIN input form (Part 3)

5.3.7 DRPWS3E (Monolithic Straight Drop Spillways)

The Drpws3e.java program that is translated from DRPWS3E FORTRAN program using F2J translator and other formatters is used to execute the structural design of DRPWS3E. The input fields of DRPWS3E are:

- Structure Identification Information
  - Line 1 – Optional Comment Line 1. Input type is text.
  - Line 2 – Optional Comment Line 2. Input type is text.
- Primary Data
  - H, F, S, J, L, LB – Input type is ‘Numerical Decimals’. All are required fields.
  - Design – Input type is ‘Radio button’. Only one can be selected at once. There are 8 design modes in Drpws3e such as 0, 1, 2, 3, 100, 101, 102, and 103. Preliminary design modes – 0, 1, 2 and 3. Detail design modes – 100, 101, 102, and 103.

- Defaults – Input type is checkbox. Checkbox values - Default 1, Default 2, Default 3, Default 4, Default 5, Default 6, and Default 7. One or more options can be checked.

- Secondary Data - Input type is ‘Numerical Decimals’. All are required fields.
  - Default 1 – These fields are displayed when Default 1 of primary data is checked.
    - CREEPR, FLOATR, SLIDER, BAT, SWLDRN.
  - Default 2 – These fields are displayed when Default 2 of primary data is checked.
    - HB, ZPS, HTOE, TTOE, CFSS, CFSC
  - Default 3 – These fields are displayed when Default 3 of primary data is checked.
    - KOH, GMH, GSH
  - Default 4 – These fields are displayed when Default 4 of primary data is checked.
    - KOF, GMF, GSF, KPF
  - Default 5 – These fields are displayed when Default 5 of primary data is checked.
    - KOW, GMW, GSW, KPW
  - Default 6 – These fields are displayed when Default 6 of primary data is checked.
    - DW2, HEAD2, TAIL2, HEAD1
  - Default 7 – These fields are displayed when Default 7 of primary data is checked.
    - DWM2, DWM3, DWM4, DWM5, HEADM2, HEADM3, HEADM4, HEADM5, TAILM2, TAILM3, TAILM4, TAILM5
- User Interactive Switches
  - R-C Design Method
    - Working Stress Design
    - Strength Design
  - Material Properties

        - Soil Conservation Default Values
        - Corps of Engineers Default Values
        - User Supplied Values
    - Other Options
        - Flotation Criteria
            - Soil Conservation Service
            - Corps of Engineers
        - Moment/Thrust/Shear Report
            - Yes
            - No
        - Full or Summary Report
            - Full
            - Summary

- Email (Copy of Output to be sent) – Optional field to enter an email id. The output on running the DRPWS3E is generated as a PDF and the same is sent to the email id provided.
- RESET – Input type is button. As the name says, when RESET is clicked, all the fields will be cleared and a new form is displayed.
- RUN DRPWS3E – Input type is button. Once the button is clicked, it validates the form if all the required fields are filled. If any of the fields is empty, it pops up an error message asking to fill all the fields to run the project. Once the form is validated, the output is generated as a PDF.

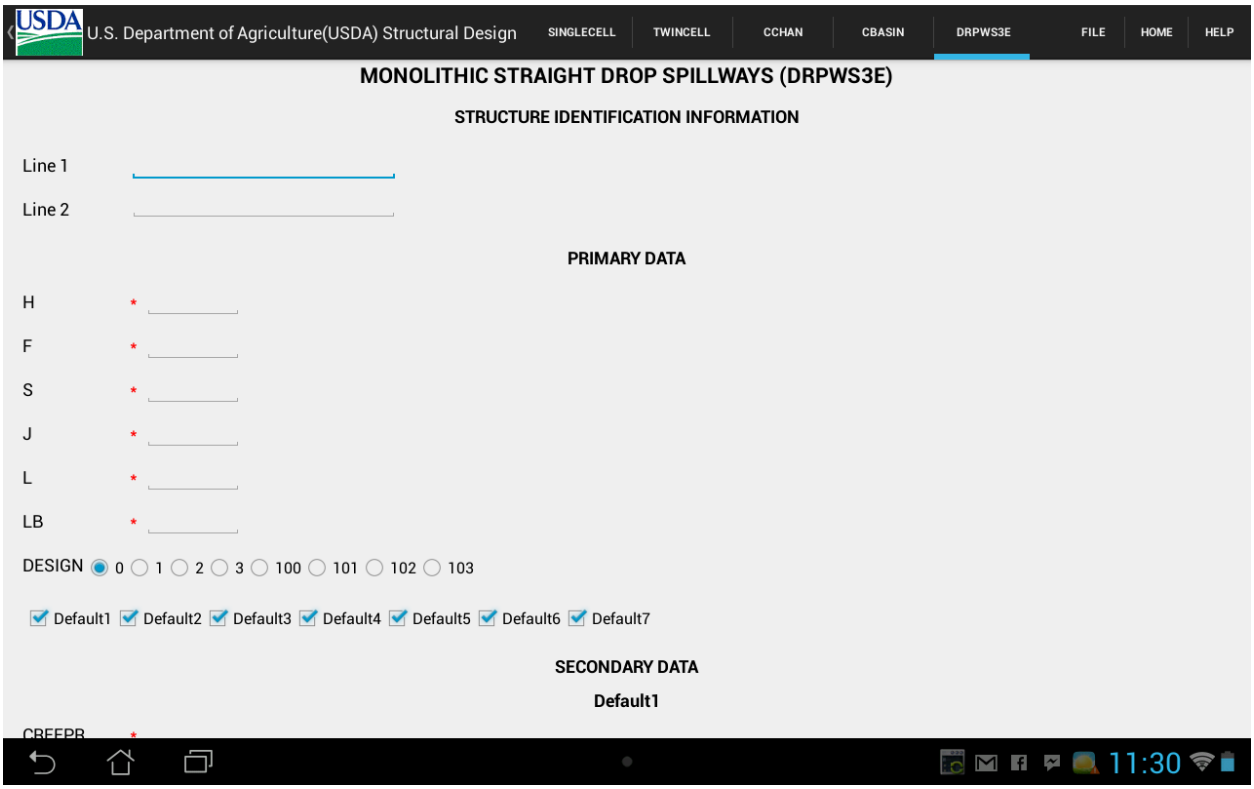

Figure 5.15 DRPWS3E input form (Part 1)

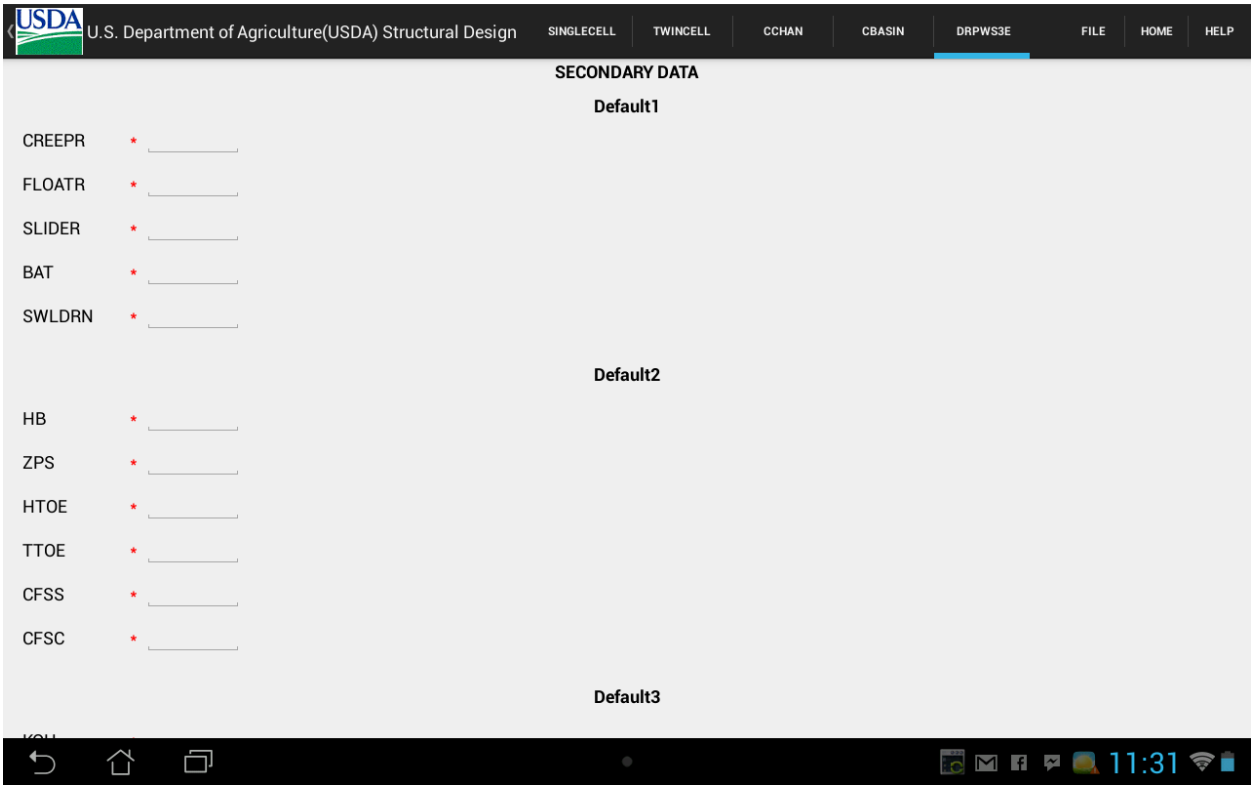

Figure 5.16 DRPWS3E input form (Part 2)

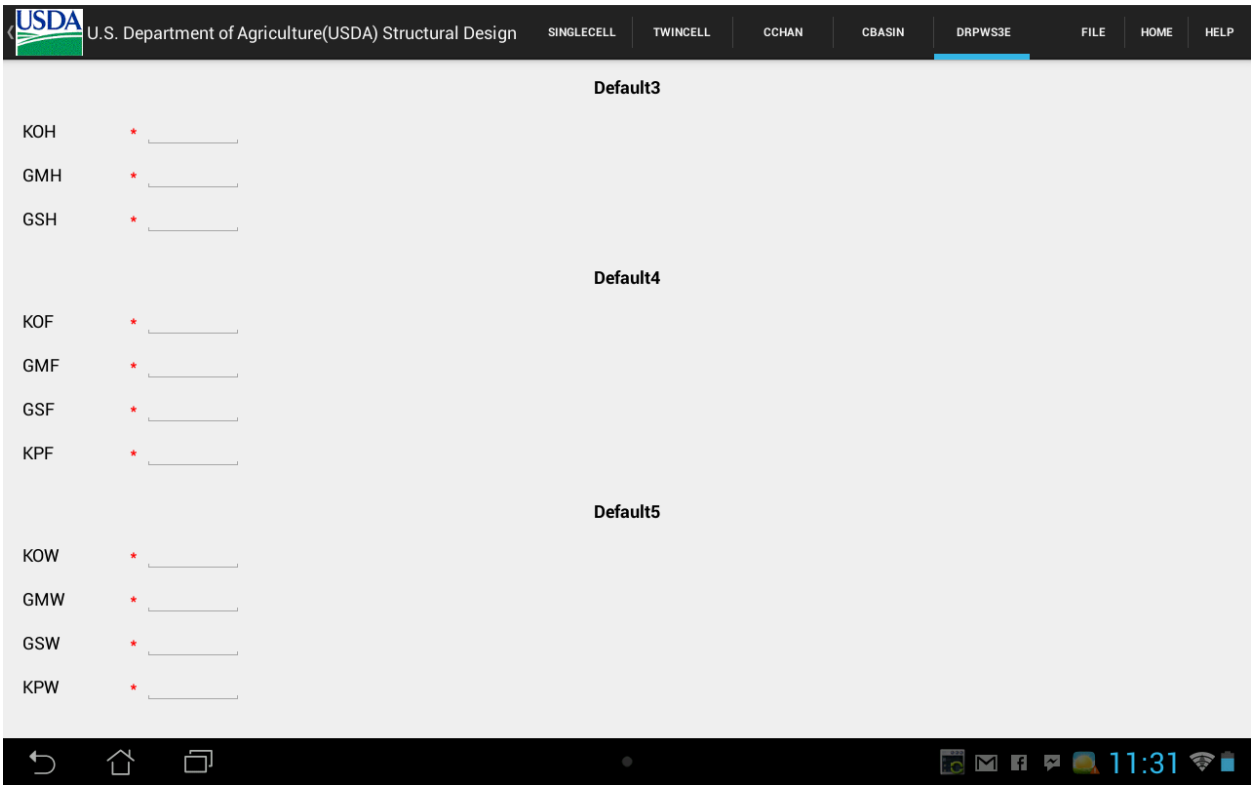

Figure 5.17 DRPWS3E input form (Part 3)

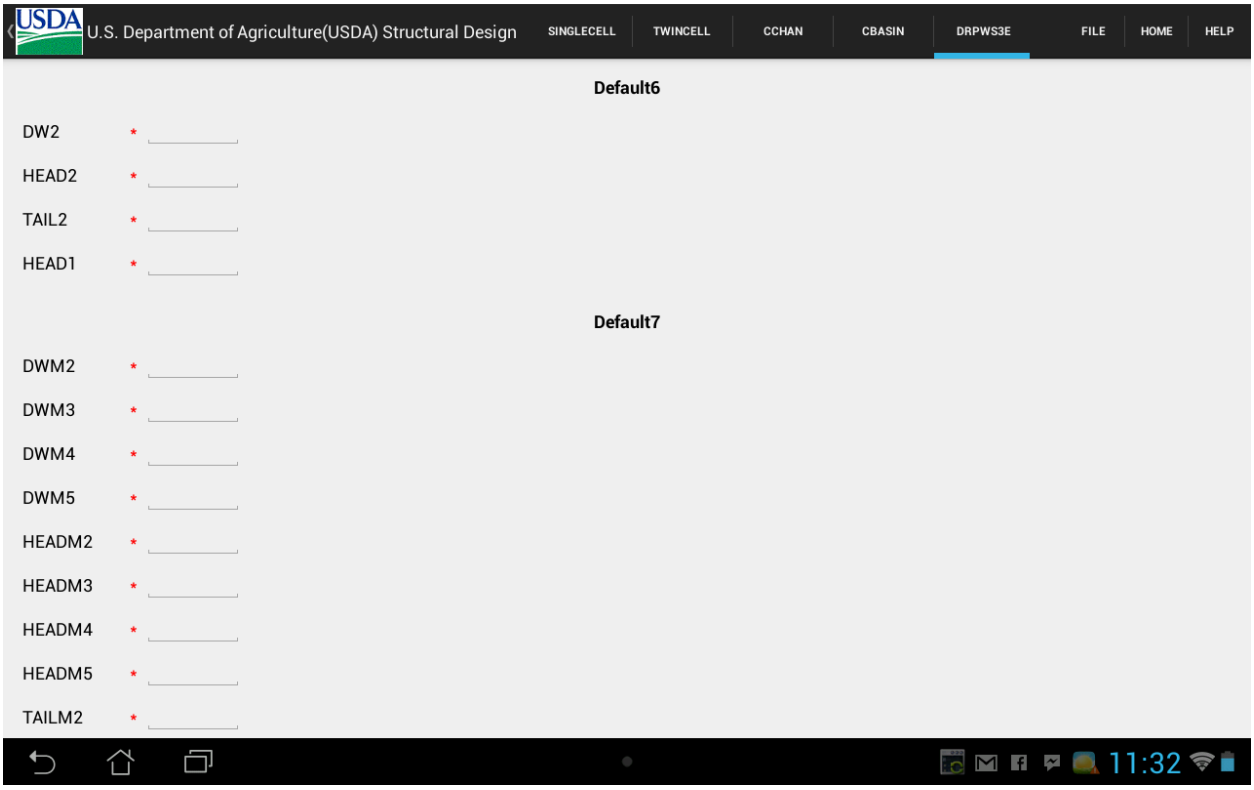

Figure 5.18 DRPWS3E input form (Part 4)

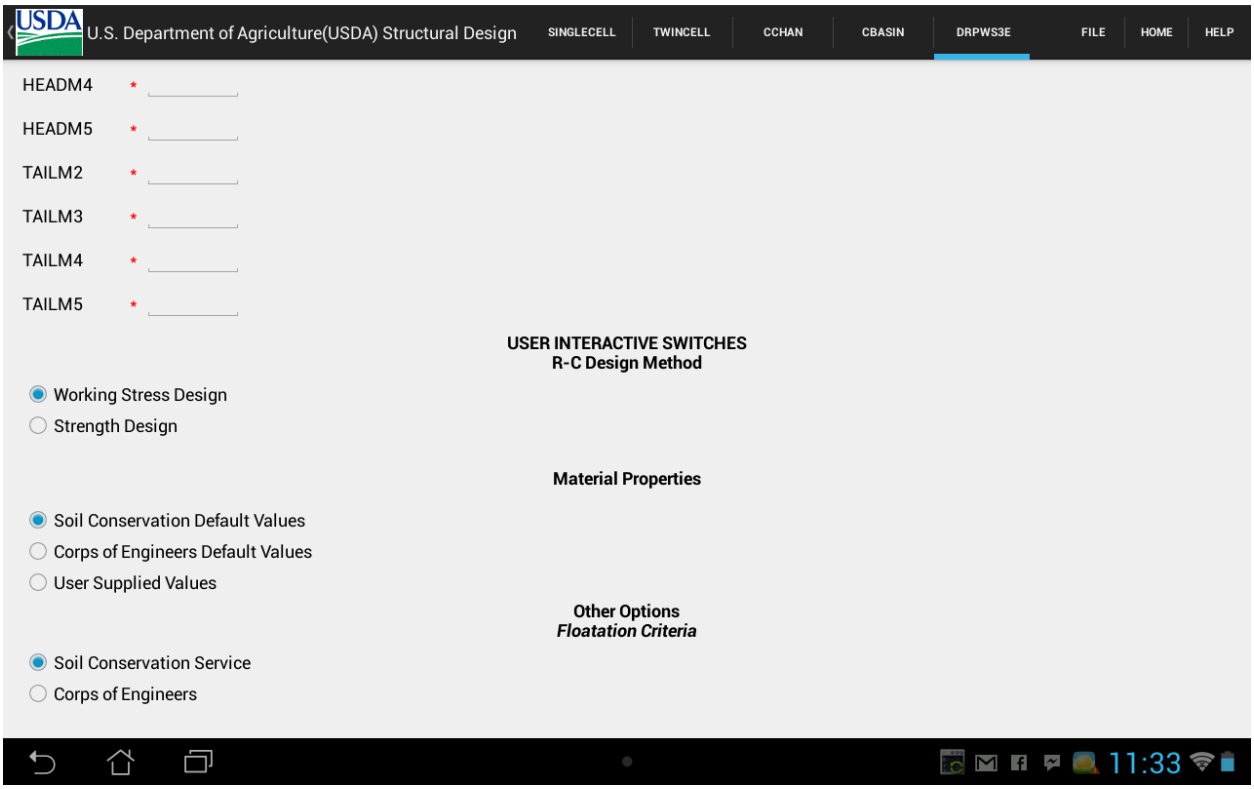

Figure 5.19 DRPWS3E input form (Part 5)

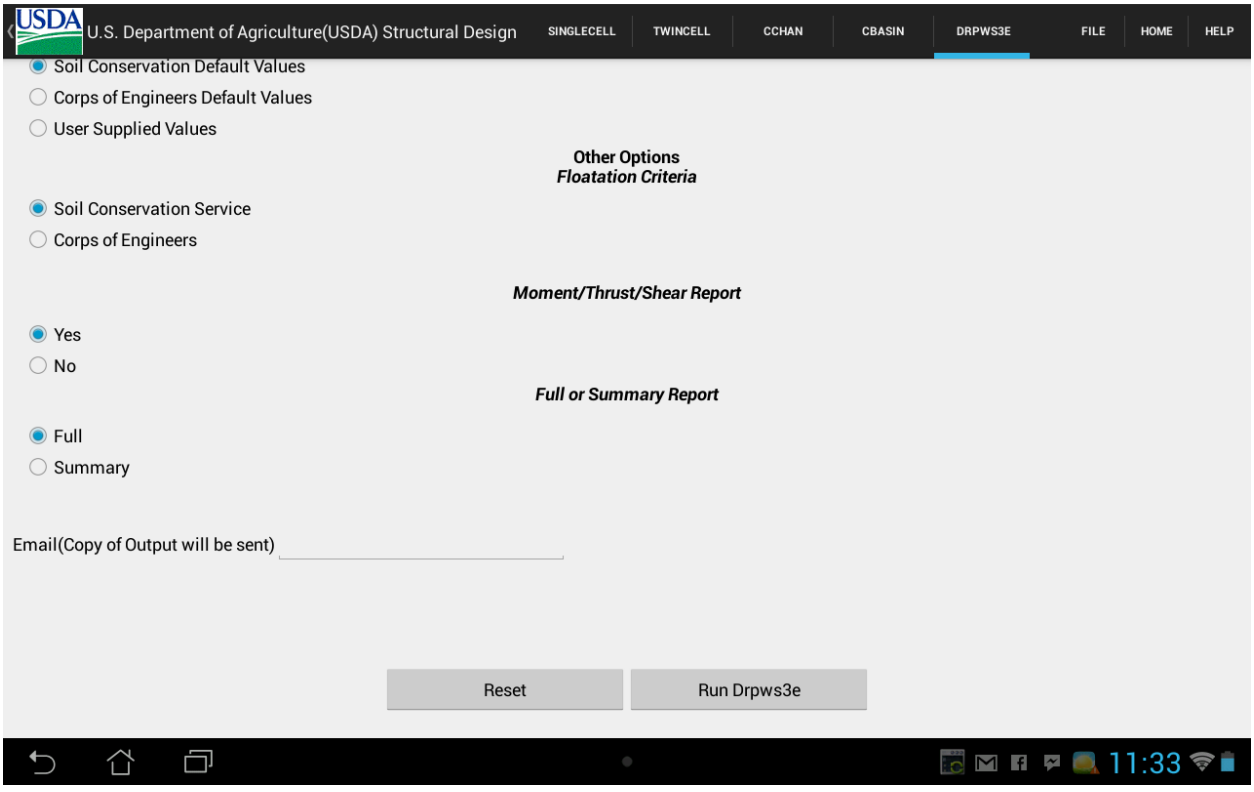

Figure 5.20 DRPWS3E input form (Part 6)

Chapter 6 - Testing and Logging

6.1 Logger and Debugger

Eclipse IDE already has a default JDWP-complaint debugger that helps in viewing the values of variables, run the code through every step and pause application execution at any step. As USDA project is developed in Eclipse IDE, JDWP-complaint debugger is used to debug the code whenever required.

The debugging environment of Android has various components:

- Dalvik Debug Monitor Server (DDMS) – It is a debugging tool that provides logcat, thread and heap information on device. This is integrated in Eclipse on installing Android SDK.
- Android Device – Application should be run on the device for it to be debugged.

6.2 Unit Testing

Unit testing or Component testing is one of the types of Software testing that helps in verifying specific block of code in the program.

Table 6.1 Unit Test Cases for USDA Structural Design

S. No	Test Case Description	Expected Result	Actual Result
1	On click of launch icon	Navigate to the Main page of USDA application	Pass
2	On launch of Main page	Display USDA logo and buttons of structural components SINGLECELL, TWINCELL, CCHAN, CBASIN and DRPWS3E	Pass
3	On Click of ‘Singlecell’ Button	Navigate to the input form of SINGLCELL. Display Action bar named ‘Single Cell Rectangular Conduits’ and fields of ‘Singlecell’ page.	Pass
4	On Click of ‘Twincell’ Button	Navigate to the input form of TWINCELL. Display Action bar named ‘Twin Cell Rectangular Conduits’ and	Pass

		fields of ‘Twincell’ page.	
5	On Click of ‘Cchan’ Button	Navigate to the input form of CCHAN. Display Action bar named ‘Reinforced Concrete Rectangular Channel (CCHAN)’ and fields of ‘Cchan’ page.	Pass
6	On Click of ‘Cbasin’ Button	Navigate to the input form of CBASIN. Display Action bar named ‘SAF stilling Basin (CBASIN)’ and fields of ‘Cbasin’ page.	Pass
7	On Click of ‘Drpws3e’ Button	Navigate to the input form of DRPWS3E. Display Action bar named ‘Monolithic Straight Drop Spillways (DRPWS3E)’ and fields of ‘Drpws3e’ page.	Pass
8	On View of Action bar in every screen	Display Menu items File, Home and Help; submenu items Open, Save As for menu item File.	Pass
9	On click of HOME on Action Bar	Navigate to Main page of USDA project.	Pass
10	On Click of RESET button on any form	Replaces all the fields with blanks and display a new form for Singlecell, Twincell, Cchan, Cbasin and Drpws3e.	Pass
11	On click of Run Singlecell button	If all fields are not filled, throw an error asking user to fill the fields. If all fields are filled, the output is displayed as pdf.	Pass
12	On click of Run Twincell button	If all fields are not filled, throw an error asking user to fill the fields. If all fields are filled, the output is displayed as pdf.	Pass
13	On click of Run Cchan button	If all fields are not filled, throw an error asking user to fill the fields. If all fields are filled, the output is displayed as pdf.	Pass
14	On click of Run Cbasin button	If all fields are not filled, throw an error asking user to fill the fields. If all fields	Pass

		are filled, the output is displayed as pdf.	
15	On click of Run Drpws3e button	If all fields are not filled, throw an error asking user to fill the fields. If all fields are filled, the output is displayed as pdf.	Pass
16	On checking checkboxes Default1, Default2, Default 3 in CCHAN	Displays fields of Default1, Default2, Default3	Pass
17	Unchecked checkboxes Default1, Default2, Default 3 in CCHAN	Do not display fields of Default1, Default2, Default3	Pass
18	On checking checkboxes Default1, Default2, Default 3 in CBASIN	Displays fields of Default1, Default2, Default3	Pass
19	Unchecked checkboxes Default1, Default2, Default 3 in CBASIN	Do not display fields of Default1, Default2, Default3	Pass
20	On checking checkboxes Default1, Default2, Default 3, Default4, Default5, Default6, Default7 in DRPWS3E	Displays fields of Default1, Default2, Default3, Default4, Default5, Default6, Default7	Pass
21	Unchecked checkboxes Default1, Default2, Default 3, Default4, Default5, Default6, Default7 in DRPWS3E	Do not display fields of Default1, Default2, Default3, Default4, Default5, Default6, Default7	Pass

6.3 Integration Testing

Integration testing is type of software testing that verifies if the interaction between components on integration works successfully. In USDA project, the activities on the Action bar are integrated for all components and so testing the same proves Integration Testing is successful.

Table 6.2 Integration Test Cases for USDA Structural Design

S. No	Test Case Description	Expected Result	Actual Result
1	On Click of OPEN submenu item of menu item FILE	Navigate to Open File screen with list of folders.	Pass
2	On selecting a file from Open File screen	Fill the form with values from the text file.	Pass
3	On Click of SAVE AS submenu item of menu item FILE	Navigate to Save As File screen and On Saving File, Display message 'Successfully saved a file'	Pass
4	On click of HOME on Action Bar	Navigate to Main page of USDA project.	Pass
5	On click of HELP on Action Bar in SingleCell screen	Open document related to ' Singlecell Rectangular Conduits'	Pass
6	On click of HELP on Action Bar in TwinCell screen	Open document related to 'Twincell Rectangular Conduits'	Pass
7	On click of HELP on Action Bar in Cchan screen	Open document related to 'Reinforced Concrete Rectangular Channel (CCHAN)'	Pass
8	On click of HELP on Action Bar in Cbasin screen	Open document related to 'SAF stilling Basin (CBASIN)'	Pass
9	On click of HELP on Action Bar in Drpws3e screen	Open document related to 'Monolithic Straight Drop Spillways (DRPWS3E)'	Pass

6.4 Performance Testing

Performance testing is a non-functional testing type in which developed software application is run to determine the time it takes from launch to end of the application under specific workload. TraceView (11) tool is used to test performance in Eclipse.

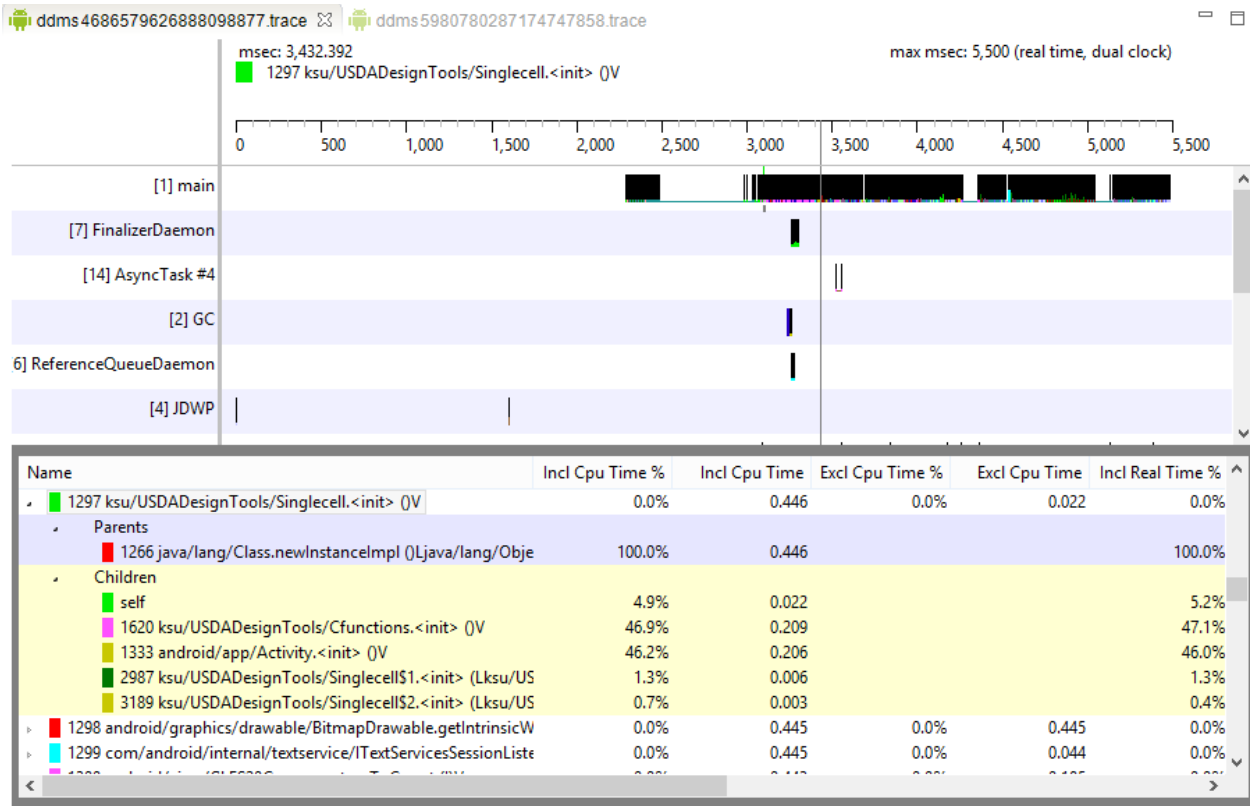

Figure 6.1 Sample TraceView Time and Profile panels for Single cell

The times taken for different components of USDA are:

Table 6.3 Performance testing of USDA

Component/Activity	**Inclusive CPU Time**	**Exclusive CPU Time**
SINGLECELL	0.446	0.022
TWIN CELL	0.446	0.022
CCHAN	0.532	0.030
CBASIN	0.538	0.033
DRPWS3E	0.547	0.045

The output of Traceview displays Time panel and Profile panel. The profile panel shows the time taken for an activity. The Figure 6.1 shows the time taken for Single Cell activity. It shows two times such as Inclusive CPU time and Exclusive CPU time.

6.5 Compatibility Testing

Compatibility testing is a non-functional testing type in which developed software application is run on different devices to ensure compatibility. The USDA application was run on different devices of API level more than 8. Also, USDA application is developed to support landscape and portrait mode. Different layouts were designed to support the orientations. The landscape orientation and portrait orientation are shown in Figure 5.2 and Figure 5.3 respectively. As application requires file upload and has forms with many inputs, it is best compatible with tablets when compared to mobile devices.

Chapter 7 - Conclusion and Future Work

There has been an increase in the usage of Android devices as they are more portable, accessible and usable in comparison to a desktop device. This has led to many desktop applications being developed for use on Android devices also. This report described the design of an Android application for the USDA (U.S. Department of Agriculture) structural design that can be used on Android tablets.

Since the programs are initially written in FORTRAN which can be applied on desktop computers, an F2J (FORTRAN to Java) translator and other formatters (CommonOut, CommentRemove, CommonIn, SwapStreams) were used to convert the programs to Java, the language that is supported by Android devices. Once the translation is completed, Graphical User interface is developed to display input forms for all the components (SINGCELL, TWINCELL, CCHAN, CBASIN and DRPWS3E) to support Android devices. The user can fill the input forms manually or can fill the form by selecting a file. Once the form is filled, the component can be run with a button click. Then, the form is validated and output is displayed as pdf. The user can also email the output to his/her personal email id by entering id in the input form before the application is run.

The application can further be developed to support other platforms like ios. Also, the application now takes input files from the storage of the device. The application can be further extended to select file inputs from server and upload the output files back to server. The application can also be extended to take the file inputs from the assets folder of the project and upload the output files back to the folder so that they can be accessed in any device.

Chapter 8 - References

1. USDA. *U.S. Department of Agriculture.* [Online] 04 06, 2014. [Cited: 04 07, 2014.] http://www.usda.gov/wps/portal/usda/usdahome.

2. Getting started|Android Developers. *Android.* [Online] [Cited: 02 03, 2014.] http://developer.android.com/training/index.html.

3. TR-210-42 - Single Cell Rectangular Conduits Criteria and Procedures for Structural Design. *NRCS eDirectives.* [Online] 12 1969. [Cited: 03 02, 2014.] http://directives.sc.egov.usda.gov/OpenNonWebContent.aspx?content=18621.wba.

4. TR-210-45 - Twin Cell Rectangular Conduits-Criteria and Procedures for Structural Design. *NCRS eDirectives.* [Online] 9 1970. [Cited: 02 10, 2014.] http://directives.sc.egov.usda.gov/OpenNonWebContent.aspx?content=18623.wba.

5. TR-210-54 - Structural Design of SAF Stilling Basins. *NCRS eDirectives.* [Online] 10 1974. [Cited: 03 10, 2014.] http://directives.sc.egov.usda.gov/OpenNonWebContent.aspx?content=18630.wba.

6. TR-210-54 - Structural Design of SAF Stilling Basins. *NCRS eDirectives.* [Online] 10 1970. [Cited: 03 13, 2014.] http://directives.sc.egov.usda.gov/OpenNonWebContent.aspx?content=18630.wba.

7. TR-210-63 - Structural Design of Monolithic Straight Drop Spillways. *NCRS eDirectives.* [Online] 2 1977. [Cited: 03 16, 2014.] http://directives.sc.egov.usda.gov/OpenNonWebContent.aspx?content=18638.wba.

8. TR-42, TR-45, TR-50, TR-54, TR-63. [Online] [Cited: 02 02, 2014.] http://sites.cis.ksu.edu/tr42/home.htm.

9. f2j. [Online] 04 07, 2014. [Cited: 02 02, 2014.] http://icl.cs.utk.edu/f2j/.

10. elinux.org-Android Architecture. *elinux.org.* [Online] 06 13, 2011. [Cited: 02 02, 2014.] http://elinux.org/Android_Architecture.

11. Android Developers. *Trace View.* [Online] [Cited: 04 02, 2014.] http://developer.android.com/tools/debugging/debugging-tracing.html.