Technical Specification for Scripter

Table of Contents

1. Introduction

1.1 Overview

Scripter is a task automation application for Android devices. It allows users to write tasks using Javascript, and these tasks are automatically performed on the user's device when specific facts are met. Scripter's purpose is to simplify everyday actions a user would normally perform manually on their device, i.e turning their phone to silent mode when they get to work. What makes this application different than other similar applications is that it allows the user to write their own task scripts from scratch and download them to their device, rather than being limited to default tasks

already created for them. We believe this to be attractive to those who already have coding knowledge such as software developers. Scripter is universally available on all of the user's Android devices. The user will be able to download their scripts to any of their devices, and then enable which tasks operate on each device using an on/off toggle. The user can upload their scripts to their own webspace, and then enter the URL into Scripter on their device to download the task.

1.2 Glossary

Android - Mobile operating system developed by Google.

Android SDK - Android Software Development Kit, developer tool used to create applications for Android devices.

Javascript - A high-level, dynamic programming language of Web content production.

URL - Uniform Resource Locator, a reference to a web resource that specifies its location on a computer network and a mechanism for retrieving it.

JavaScript Interpreter - Executes lines of JavaScript code in a way for it to be used with Java and Android SDK.

Web Server - A computer system that processes requests via the basic network protocols to distribute information on the internet.

Rhino - A JavaScript engine written fully in Java, used for interpreting between JavaScript and Java

UI - User Interface is the visual part of the application which the user interacts with.

XML - Extensible Markup Language is a language that is used for encoding and formatting documents that is readable by humans and machines.

Method - In computer programming, a method is a segment of code with a name which can be re-used. Methods can also take values as parameters which they can use during their execution.

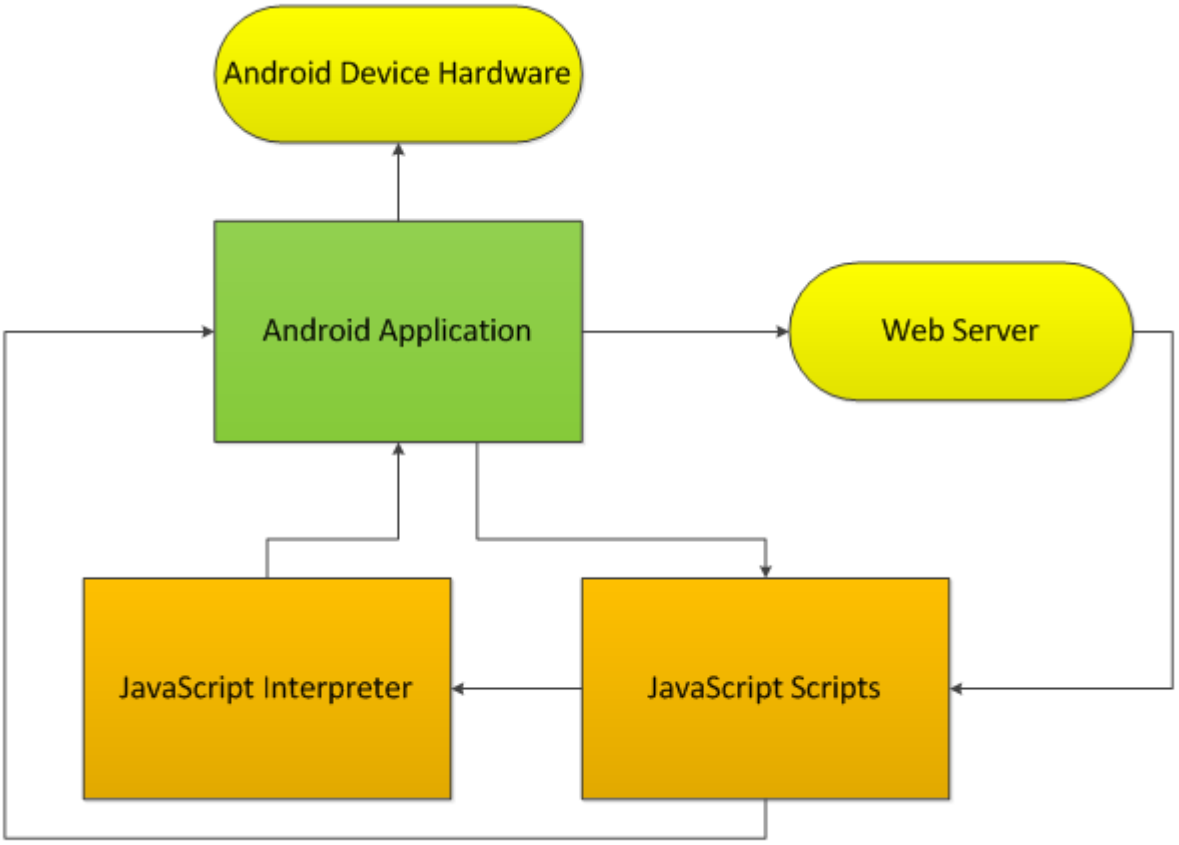
Class - In computer programming, a class is a data structure containing data, constructors and methods.

Constructor - In computer programming, a constructor is a named segment of code used to create an object instance of a class.

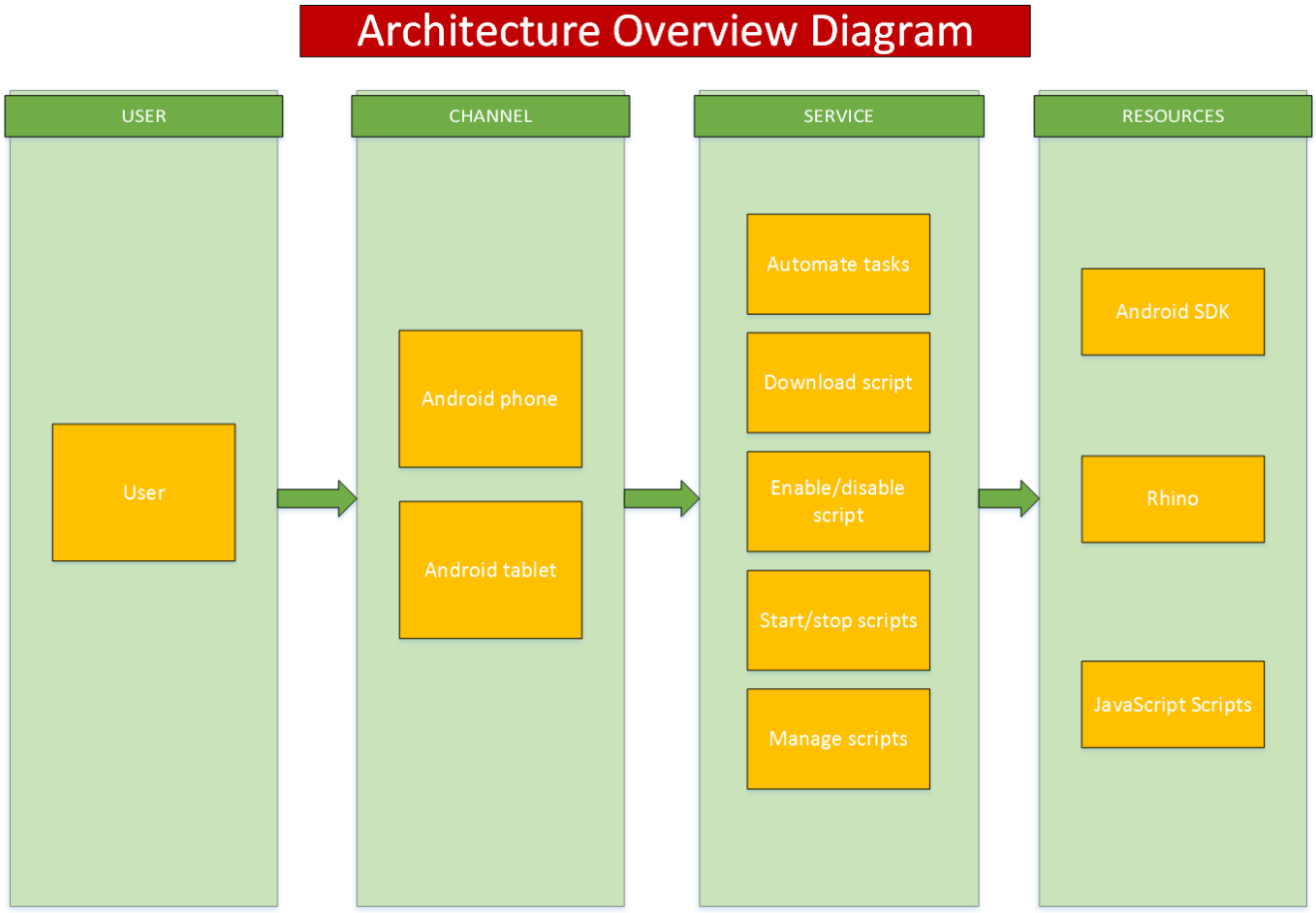
Override - A term used in computer programming which refers to re-writing a pre-existing method with new meaning.

2. System Architecture

2.1 Initial Architectural Design



2.2 Current Architectural Design

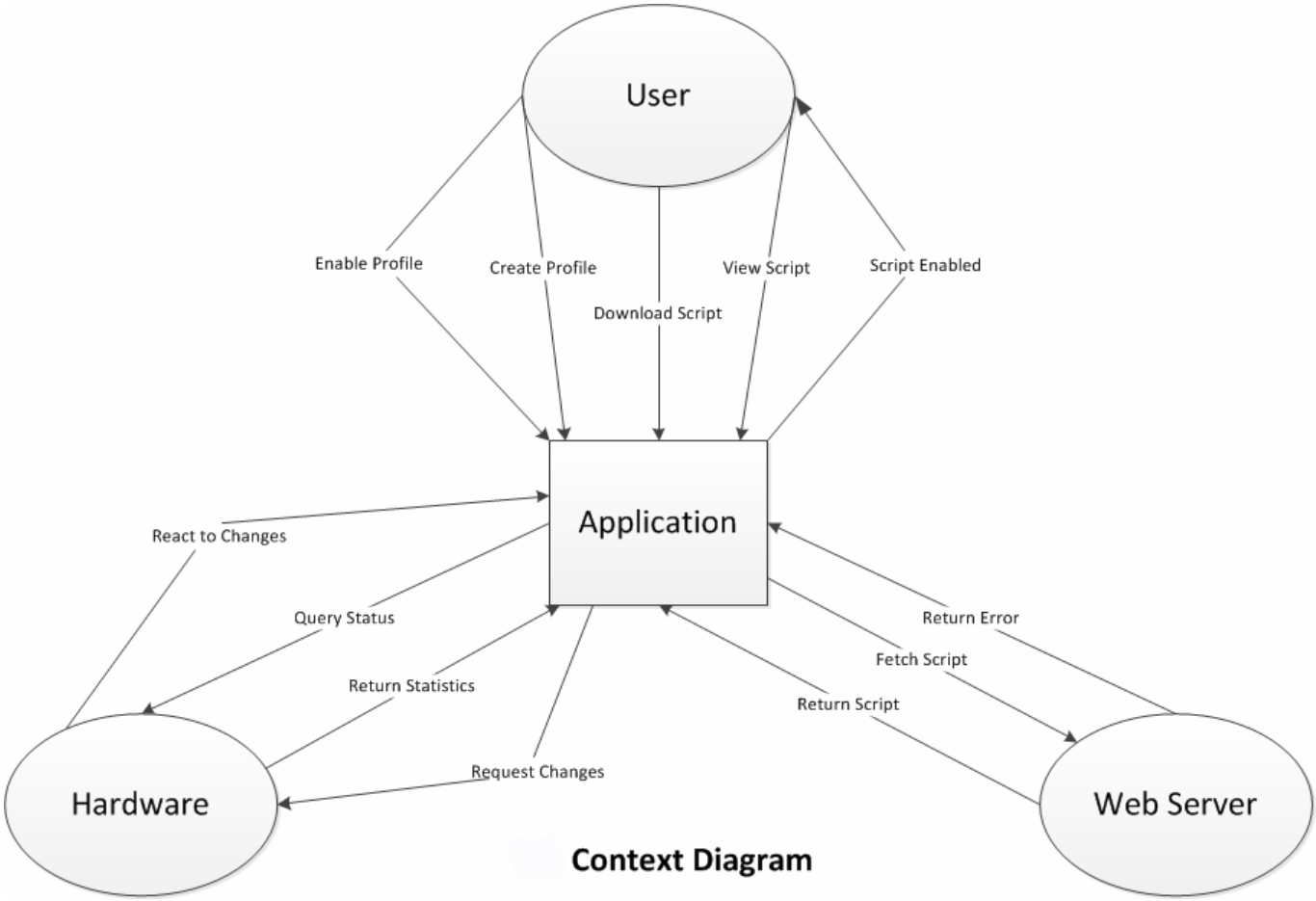


The User interacts with our application through Android devices. These include mobile phones or tablets running and Android operating system. The diagram above illustrates the services on offer to the user through these channels, and the resources used to provide them.

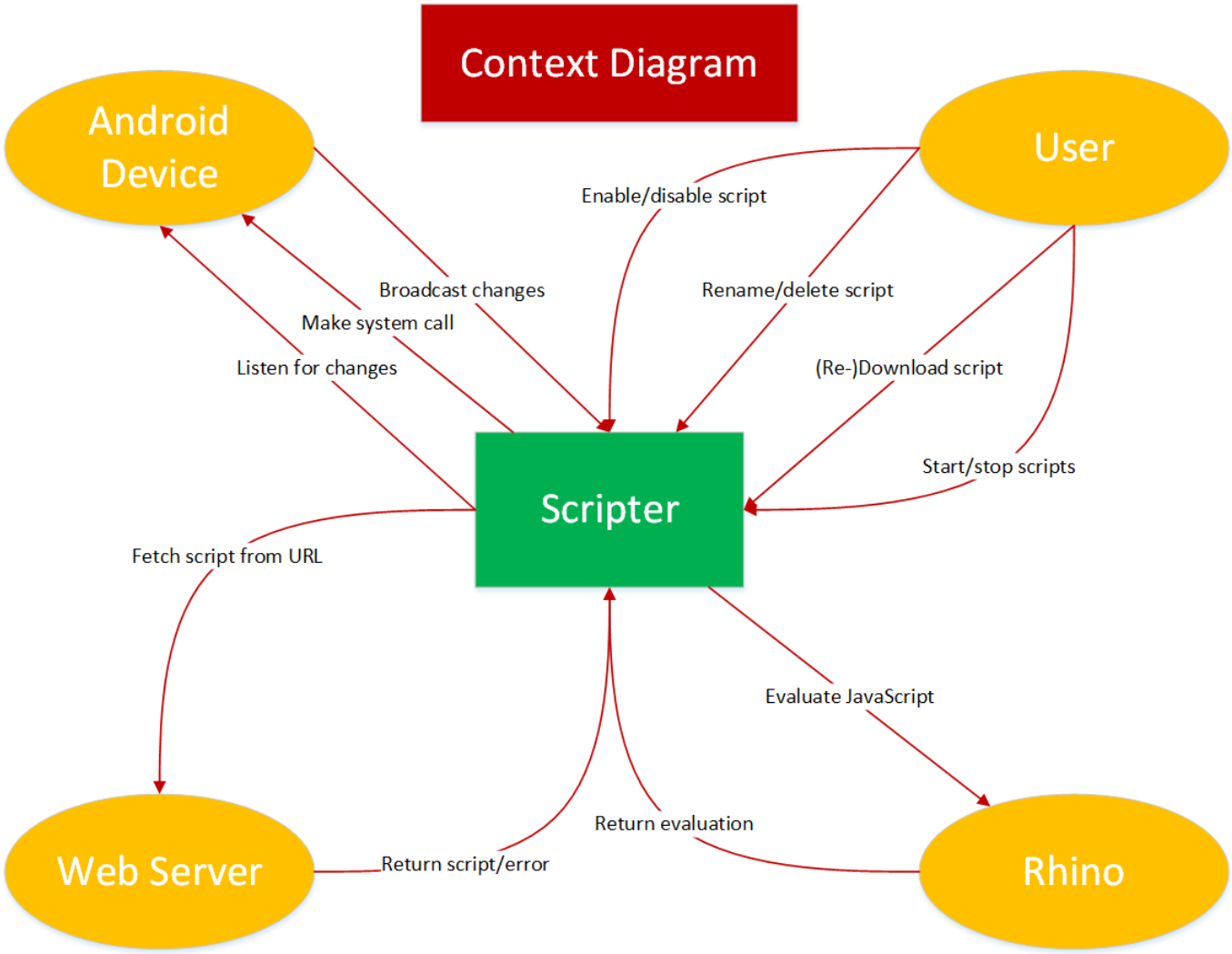
For example, the application makes calls to functions in the Android SDK in order to interact with the users device. The Javascript scripts are a resource as they are created by the user themselves, and the application makes use of them using Rhino as an interpreter.

3. High-Level Design

3.1 Initial Context Diagram

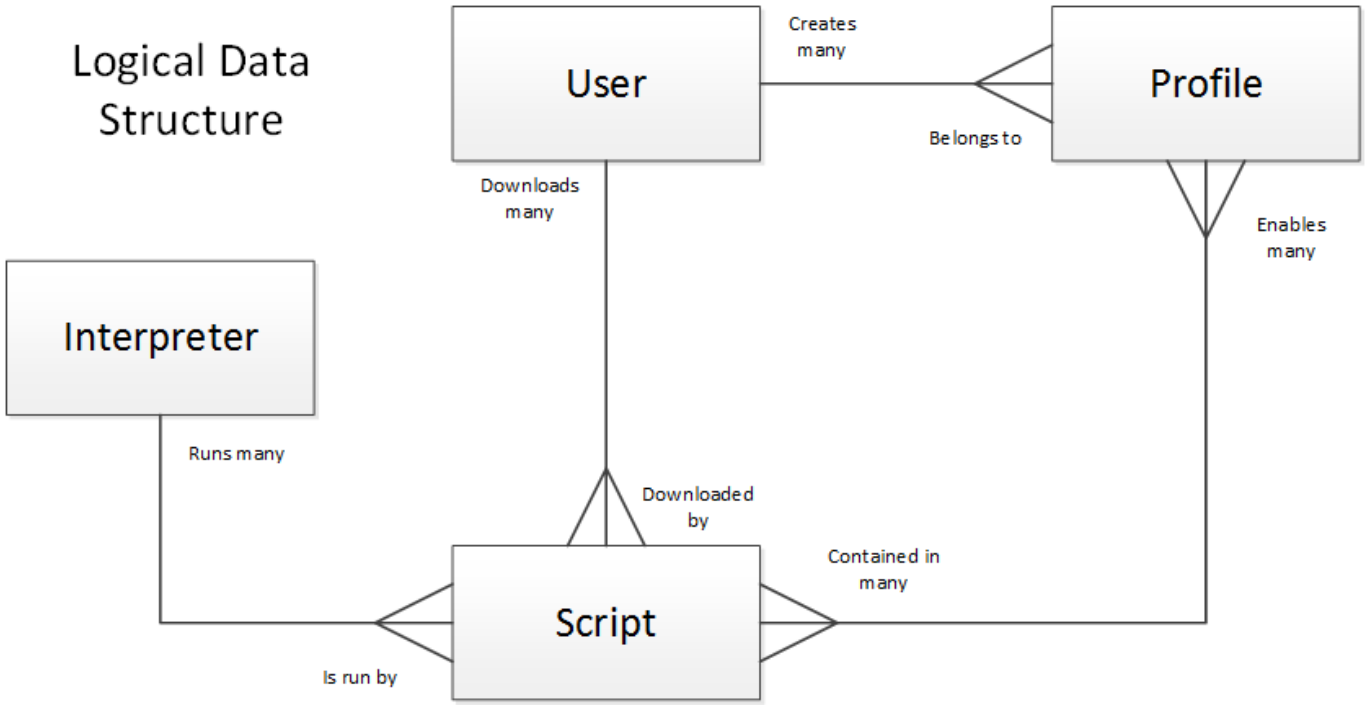


3.2 Current Context Diagram



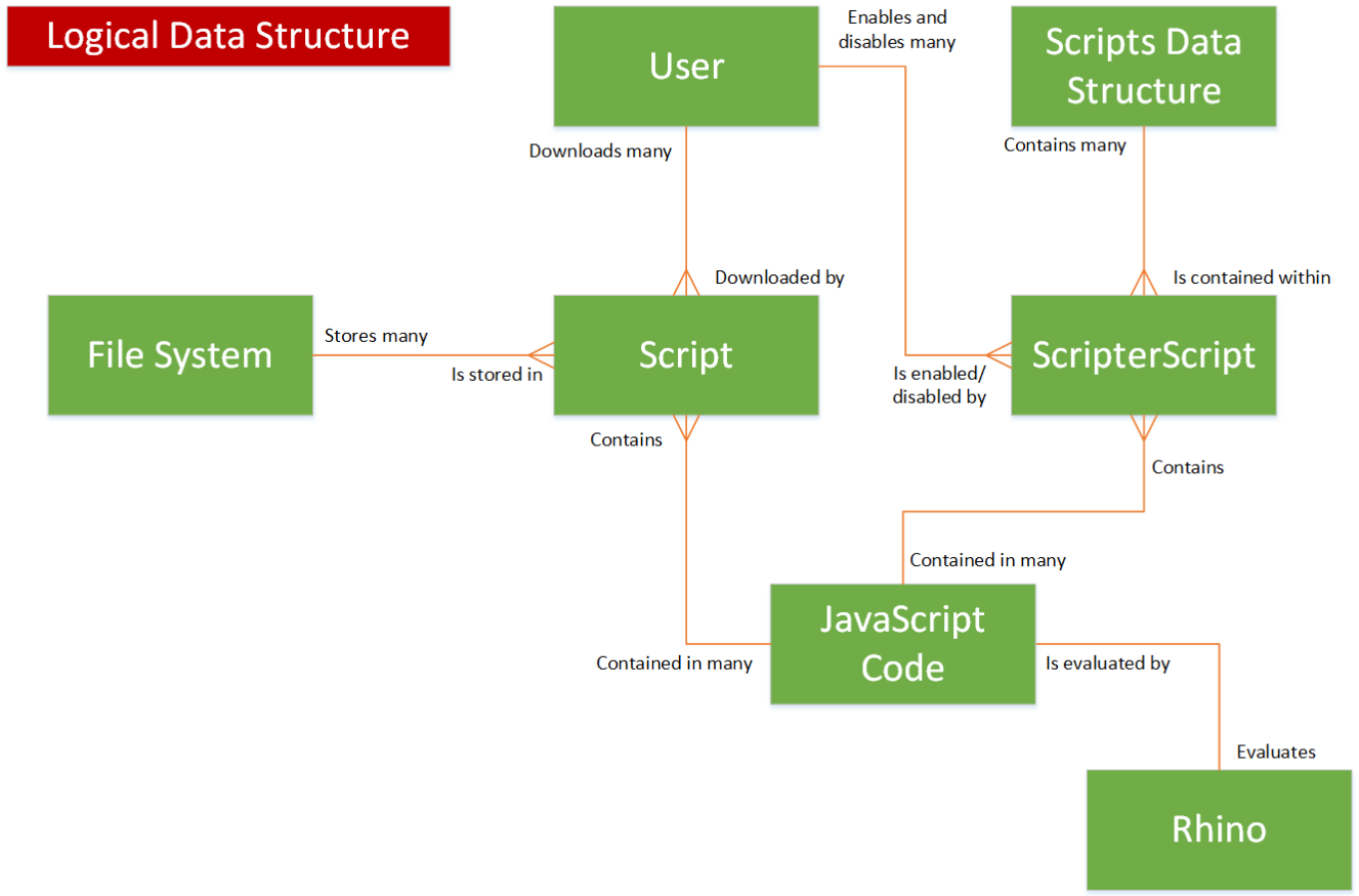
The Context Diagram shows the interactions between the external entities and our application. We removed profiles from the initial design and added the use of the Rhino interpreter. The application was also given the name Scripter. Gathering system statistics was removed as it was not an essential feature and the use of our time was prioritised on other, more critical features.

3.3 Initial Logical Data Structure

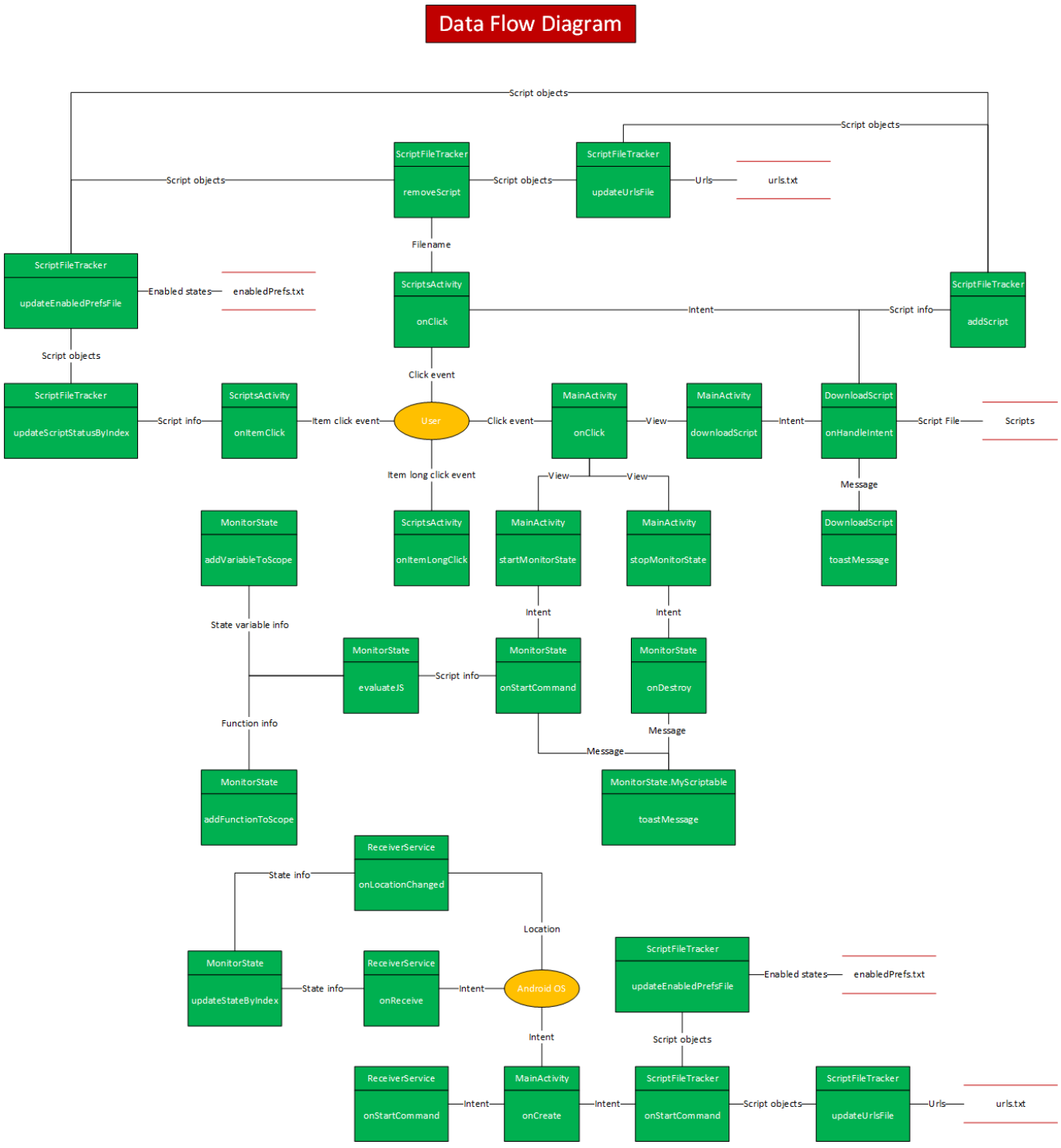


In our initial design, we had planned to have user profiles (as seen in above diagram) containing multiple scripts representing the desired tasks, which the user could enable and disable. As work progressed, we realised our script structure was essentially a profile in itself. Scripts can contain multiple tasks. Therefore enabling and disabling script files covered the profile functionality originally planned. This led to a cleaner and simpler system containing only individual scripts.

3.4 Current Logical Data Structure



3.5 Current System Data Flow Diagram



The Data Flow Diagram shows the movement of data such as script files, objects and messages throughout our application. The external entities are shown in yellow. The methods and functions in the code are shown in green. Data stores such as text files and folders in the device's internal storage are shown in red. Above each method is the name of the class it belongs to.

4. Problems and Resolutions

Problem: Running code on the UI thread all the time.

Solution: Using Services and IntentServices to take unnecessary workload off of UI thread.

Problem: Learning to write UI/XML.

Solution: Trial and error, and lots of online tutorials.

Problem: Running application in background even after app has been closed.

Solution: Using startForeground() with a constant notification in notification bar.

Problem: Making states and functions available to our scripts.

Solution: Creating methods to add the states and functions to Rhino's scope (addFunctionToScope and addVariableToScope) and ensuring they were contained in a class that extended ScriptableObject (MyScriptable).

Problem: The functions that we added to the scope had to be non-static, but this caused issues due to a bug in Rhino.

Solution: Extended FunctionObject with MyFunctionObject in MonitorState and had to override its call() method to allow for instance methods.

Problems: Storing states of different types in a single data structure.

Solution: Used an array of Objects to store and update state variables.

Problem: Showing script names in a scrollable list that will update on it's own.

Solution: Used an ArrayAdapter representing an ArrayList of the script names with a ListView.

Problem: Running scripts only when their specific dependency states have changed.

Solution: The scripts define an array called dependencies which is fetched by Scripter and checked against the changed states before running a script.

Problem: Re-downloading a script from the same URL without re-entering it.

> **Solution:** Stored a script's URL in it's ScripterScript object. Similarly, these objects store a script's filename, code, dependencies and enabled state.

Problem: Busy waiting for state changes

> **Solution:** Using broadcast receivers to tell us when a state had changed rather than a constant loop of updating states unnecessarily.

Problem: Couldn't use a broadcast receiver for location updates.

> **Solution:** Used LocationManager.requestLocationUpdates() and had to override onLocationChanged().

Problem: Permissions issues with new Android versions

> **Solution:** Added permissions to MainActivity to prompt the user to permit access to the device.

Problem: Saving enabled/disabled status after application is closed.

> **Solution:** Saved the enabled/disabled status to enabledPrefs.txt.

Problem: Homepage cluttered by text entry fields.

> **Solution:** Used an AlertDialog to prompt for input

5. Installation Guide

How to install from GitLab

- Must have Android Studio installed; Open Android Studio.
- Create and start your emulator. This can be done in Android Studio by going to Tools > Android > AVD Manager > Create Emulator
- Follow the link to find APK file of project on GitLab:[*https://turing.computing.dcu.ie/bartleg3/2017-CA326-bartleg3-androidtaskautomationapp/blob/master/code/Scripter/app-debug.apk*]
- Download the APK file.

- Start your Windows CMD and go to android-sdk/platform-tools directory.
- Paste the APK file into the platorm-tools directory.
- Then type the following command into the CMD:adb install [.apk path]
- Example: adb install C:\Users\Name\MyProject\build\example.apk

How to install when available on Google Play Store

- Open the Google Play Store on your Android device
- Enter “Scripter” into the search bar at the top of the screen
- Scripter will appear in a list of results
- Tap the 3 dots and hit install, or tap on the app to open more information about it including screenshots, a description and user reviews.
- Tap install to start downloading the application. The user may be prompted to provide device permissions for the installation to continue.
- The on-going download and installation will show in the notification bar of the device.
- Once the install is complete, the user can tap the Open button in the Play Store listing, or tap the icon from the notification tray to open the application. Otherwise, Scripter will be accessible in the Apps menu on the device.

6. Resources

- Android tutorials and information: *https://developer.android.com/index.html*
- Rhino Interpreter: *https://developer.mozilla.org/en-US/docs/Mozilla/Projects/Rhino*
- Start/Stop/Download buttons include images from: *https://pixabay.com/*
- Docker image for testing: *https://hub.docker.com/r/kelvinlawson/android-studio/*
- Prism.js used for syntax highlighting on our website *http://prismjs.com/*