1 Documentation

This document will guide you through the installation of the project as well as describe important files.

1.1 Installation

Software used during development are

- Android Studio and a device running Android 4.1 (API level 14, Jelly Bean) or later
- Xcode 9.2 with Swift 3.0 or later

Before you start you have to download the CrowdS repository. If you can't access this page please contact Mihhail Matskin and ask to be granted permissions.

1.1.1 Google account

You will need a Google+ account to which permissions can be given to view/edit the Firebase project and Google Maps.

1.1.2 Android

- Download project from the Bitbucket repository
- Import project into Android Studio
- Connect project to Firebase
  - In Android Studio, click **Tools** and then **Firebase** and locate the Notifications section
  - Expand the Notifications section and click on **Receive notifications in your app**
  - Follow the instructions to connect the project with Firebase or Sync if it is already connected
- Build the project

Update URLs in the strings.xml file.

1.1.3 iOS

- Download project from the Bitbucket repository
- Import project into Xcode
- Build the project

To make Firebase Push Notifications work with the iOS version an active Apple Developer Subscription is required. This subscription will also be required to be able to publish the app on the Apple store or to test it on devices other than the one that is used for development.
When a subscription has been purchased, open the project in Xcode and click on **cs** on the left side bar, then make sure **General** is selected and locate the **Signing** section (see figure 1). Click on the drop down menu and press **Add account** (figure 2) and log in with the Apple ID with which you have the subscription on (figure 3). You might also need to go to the Firebase console to set up a new APNs authentication key, ask an admin of the project for permissions to visit this page and look at the Firebase documentation for guidance.

Figure 1:

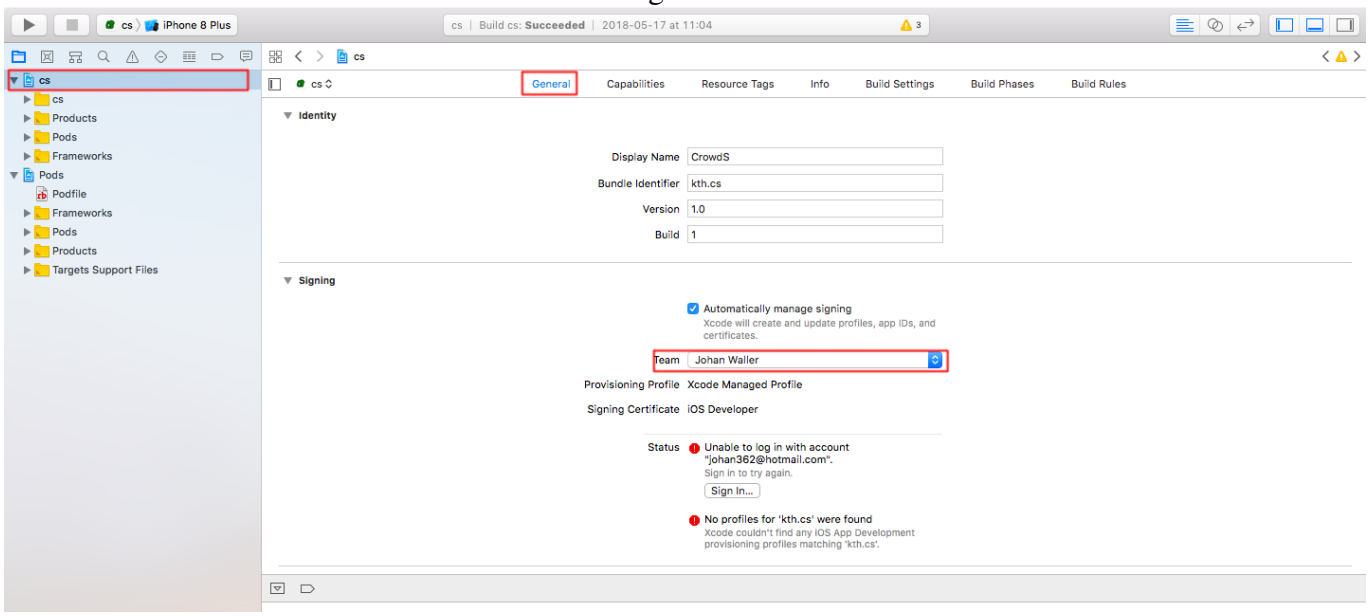


Figure 2:

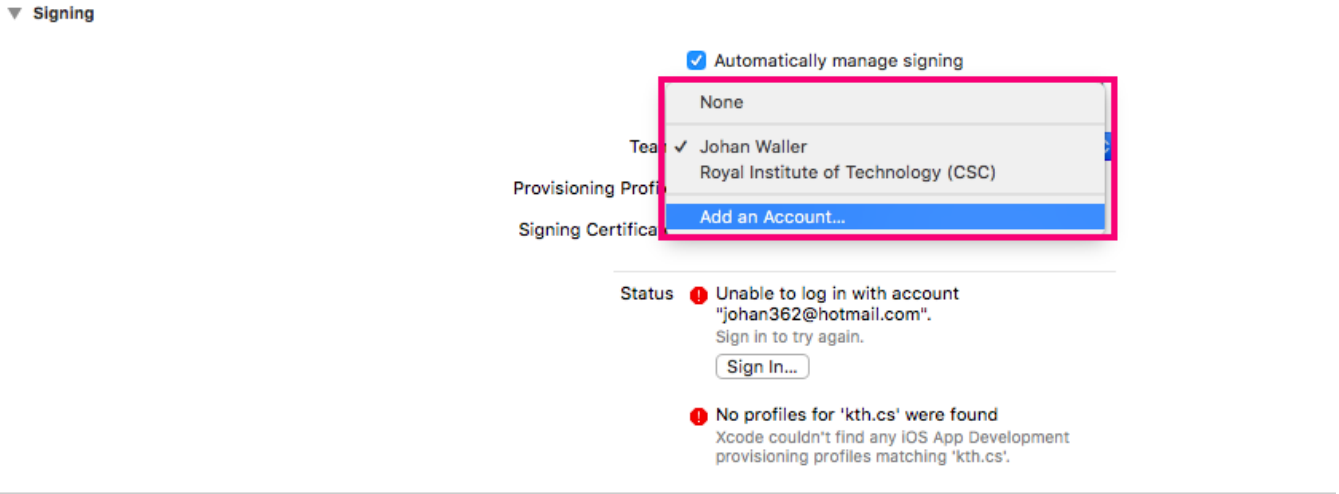
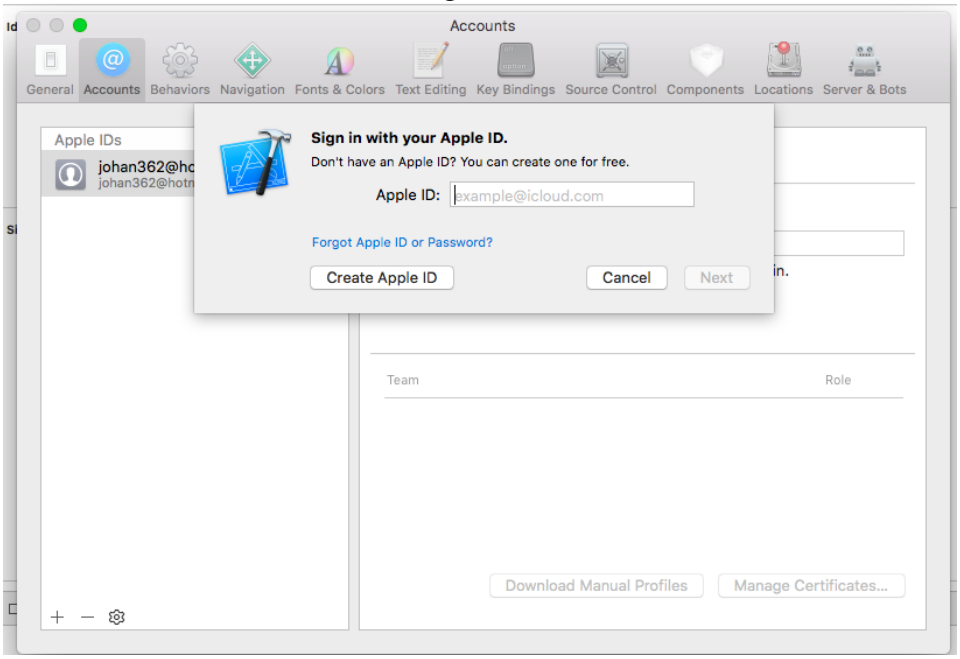


Figure 3:



1.2 Important files

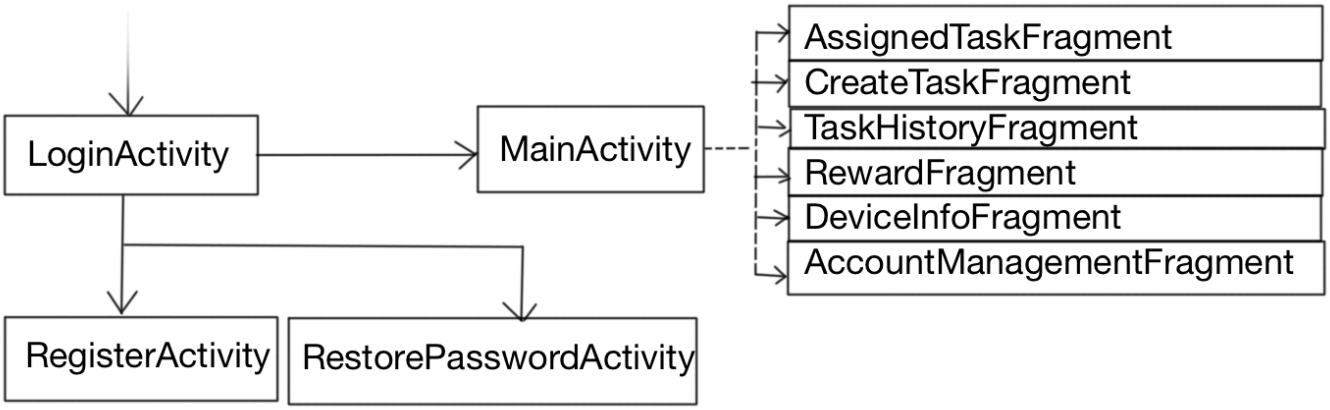
I will list the most important files and give a short explanation of what they do and what their purpose is.

1.2.1 Android

The transition flow is illustrated in figure 4.

Class	Description
AndroidManifest.xml	In this file you specify which permissions the app needs as well as the components the app consist of (Activities, Services and Receivers).
res/values/strings.xml	Contains all strings in the application. It is good practice to have Strings in this file and reference to them in the code.
app/Config.java	This file has definitions of app wide constant variables such as app version etc.
activity/LoginActivity.java	The first activity that is shown when starting the app. Handles the log in functionality.
activity/MainActivity.java	As the name suggests, this activity is the main activity. It is the container for the fragments, which is swapped in depending on what is pressed on the slide bar. The default fragment that is shown is the AssignedTaskFragment.
assets/assigned_hit.txt assets/create_hit.txt	These files contains the class names and package of existing HIT types. These files are used when creating the the view for choosing HIT type as well as the assigned HIT view. When adding new HIT types, define the user interfaces and include their names and package in these files.
assets/sensorTypes.txt	This text file contains all supported senors. To add a new sensor to support, add its name and number. The number can usually be found on the Android documentation.
fragment/CreateHitFragment.java fragment/CreateSensingFragment.java	These files uses the information stored in **assigned_hit** and the **create_hit** file to dynamically populate a list.
fragment/CreateHitTestFragment.java fragment/AssignedTestFragment.java	These classes are placeholders. They load the class that is specified in the bundle that is passed during the fragment swap. This is why the class name and the package is needed to be known.
HITs/assigned/* HITs/create/*	Contains helper functions Contains helper functions
service/MyFirebaseMessagingService.java	This class is a Service running in the background that is handling all incoming communications from the server. Broadcasts are used to communicate between activities.
service/SensorService.java	This class is a Service running in the background that is launched when a new sensor task has been received. Data is read from the specified sensor and are sent back to the server.
utils/Item.java	Used for populating lists
utils/SystemUtils.java	Contains helper functions

Figure 4:


LoginActivity
MainActivity
AssignedTaskFragment
CreateTaskFragment
TaskHistoryFragment
RewardFragment
DeviceInfoFragment
AccountManagementFragment
RegisterActivity
RestorePasswordActivity