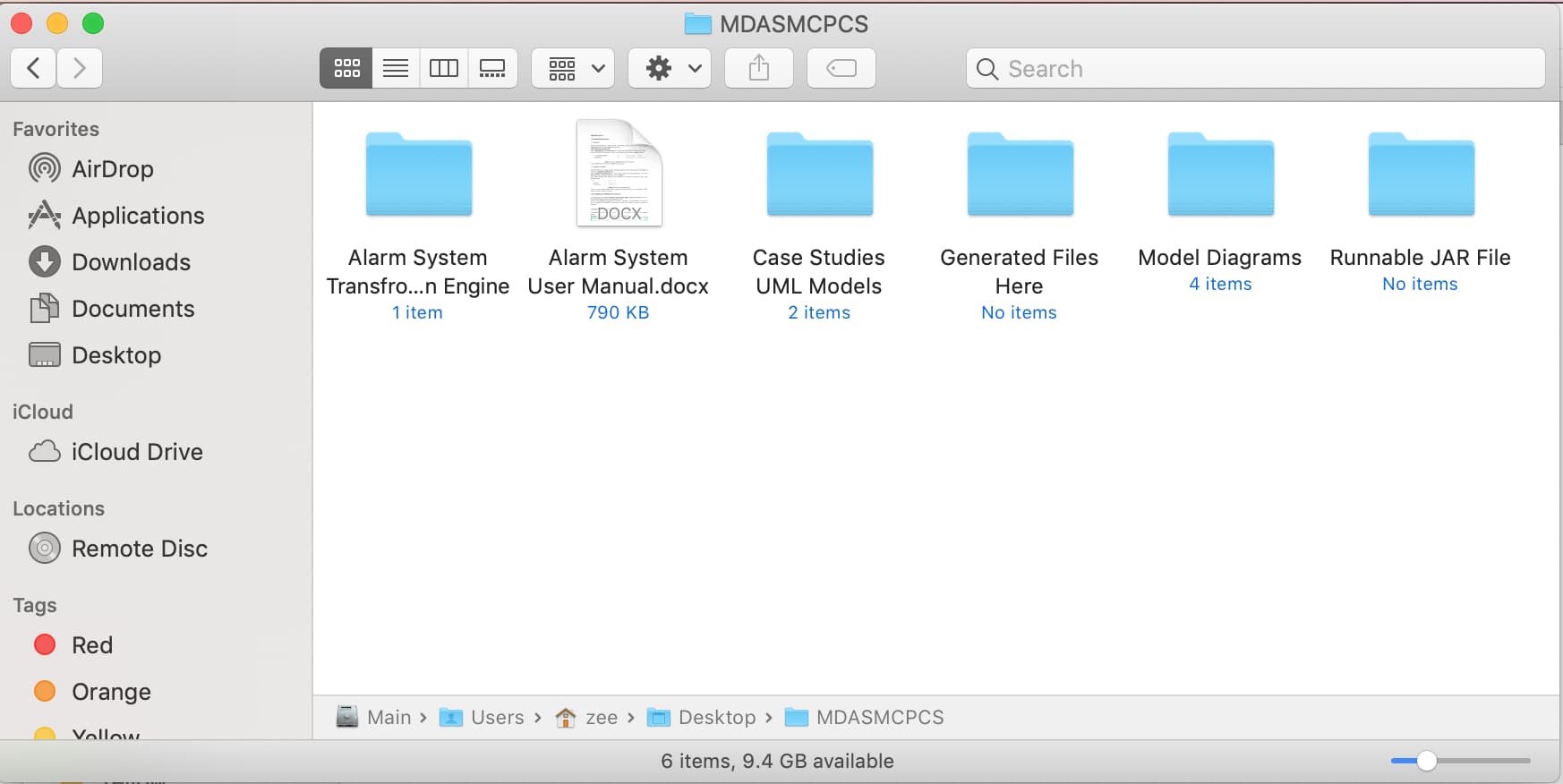
**USER MANUAL**

1. **Download Instructions**
   1. **MDASMCPCS**

Download Model-Driven Alarms Server and Mobile Clients for Process Control System (MDASMCPCS) from MDASMCPCS website as: “**MDASMCPCS.zip**”

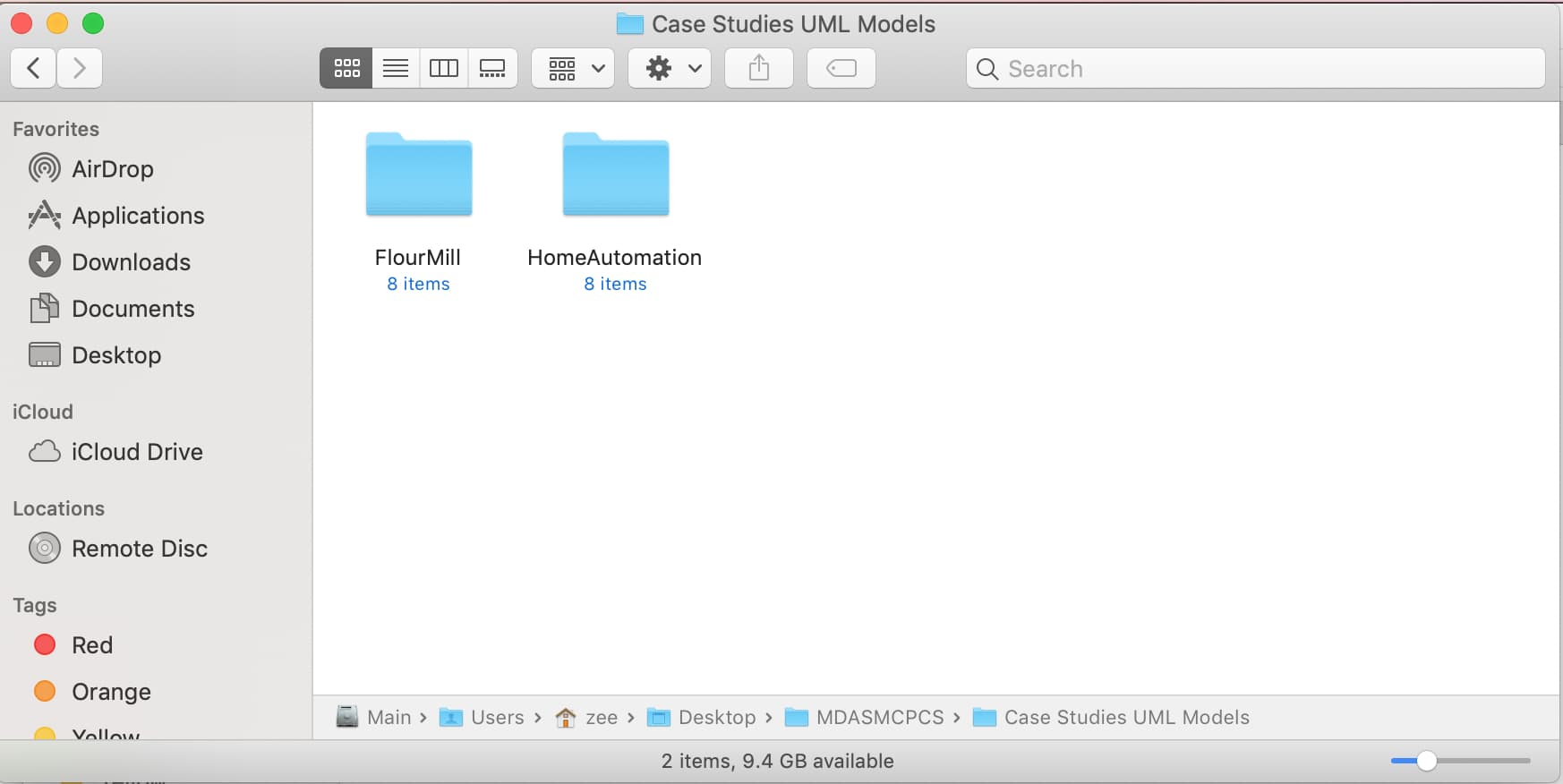
Extract **MDASMCPCS.zip** file. You will find “MDASMCPCS” Folder. In the “MDASMCPCS” folder, you will find the following files shown in **Figure 1**below.



**Figure 1.** Files in “MDASMCPCS” folder

* 1. **Sample Case Studies**

There are two case studies available that we deployed to validate MDASMCPCS. You can find these case studies in “Case Studies UML Models” folder as shown in **Figure 2**. They both contains UML models for the respective case study developed in Eclipse using Papyrus plugin.



**Figure 2.** Sample Case Study folder

You can use the existing UML models to generate complete Code or you can update the UML model to include modeling of System.

1. **Prerequisites for MDASMCPCS**

It is mandatory to have following software installed in your machine to use MDASMCPCS:

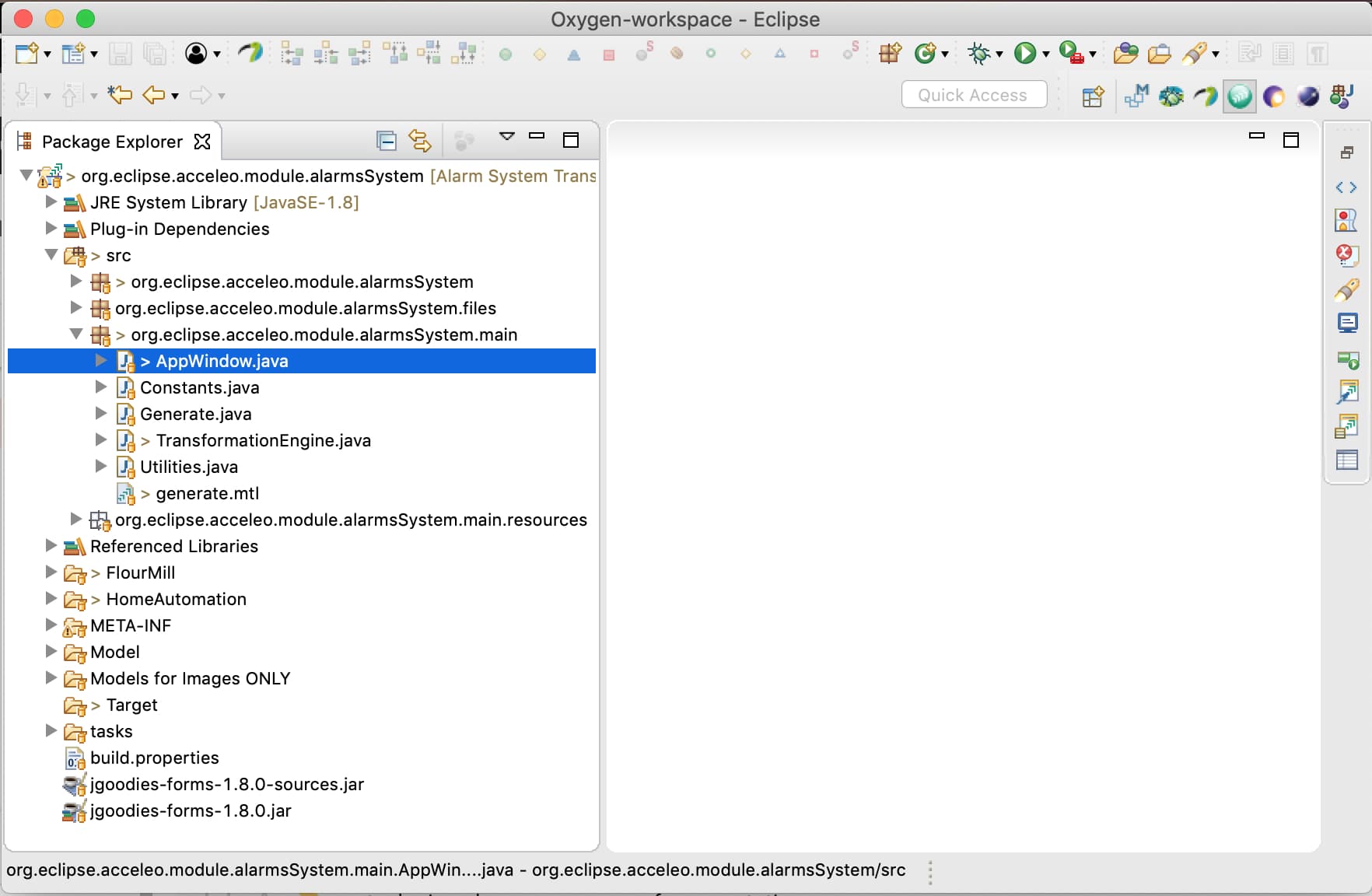
* Java Runtime Environment (JRE) version 8 or above
* Eclipse Oxygen Version 4.7.3 or above
* Papyrus - Eclipse Plugin
* Xcode version 10 or above
* Android Studio 3.4 or above
* IntelliJ IDEA 2018.2 (Community Edition)

We have tested MDASMCPCS on MacBook Pro macOS Mojave Version 10.14.3. However, we are confident that MDASMCPCS can also be executed on previous versions of macOS.

1. **Execution of MMDACG Generator**

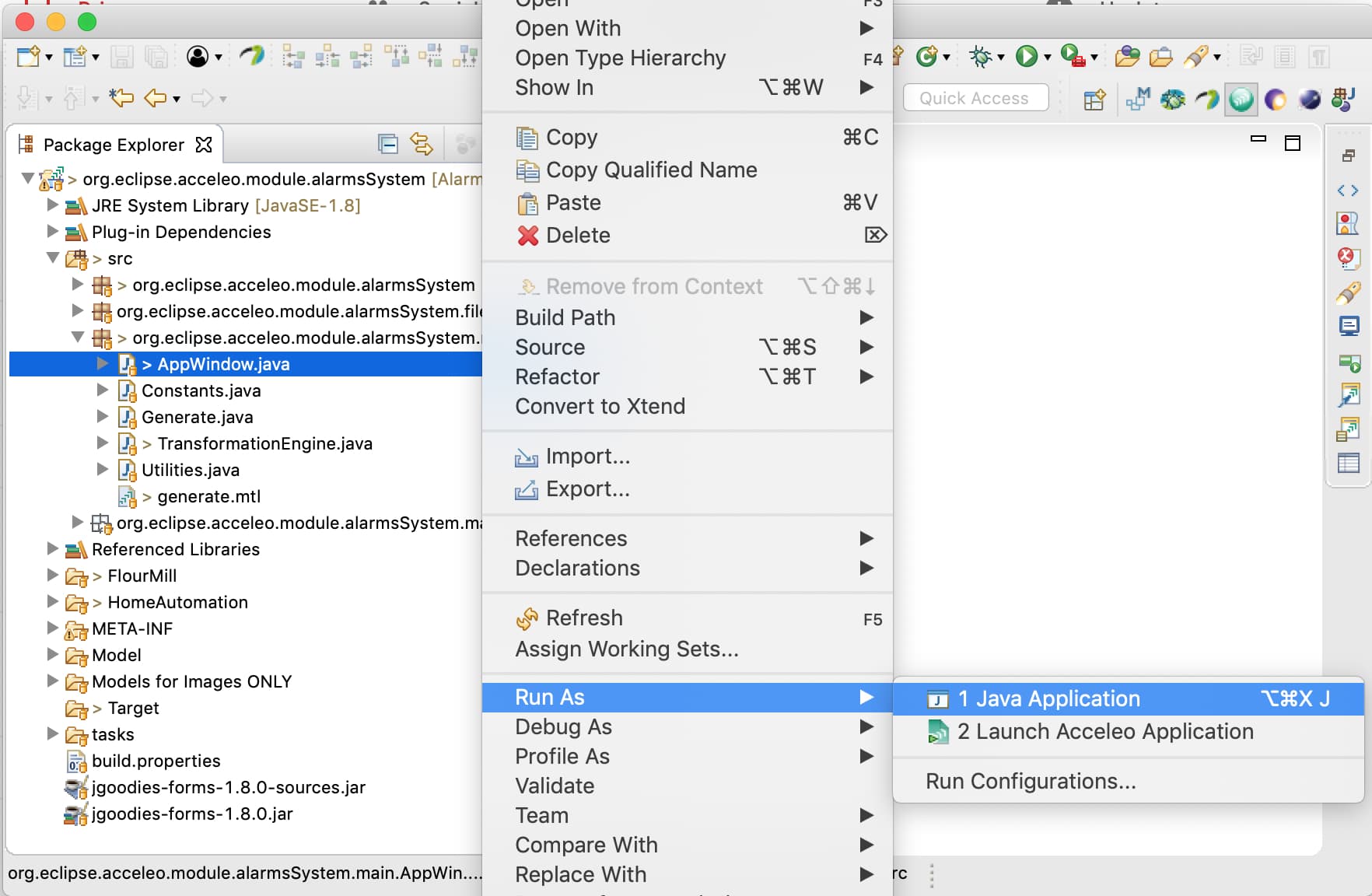
To execute theMDASMCPCS, follow the steps:

1. Open the “Alarm System Transformation Engine” in Eclipse and it shall be like shown in **Figure 3**.

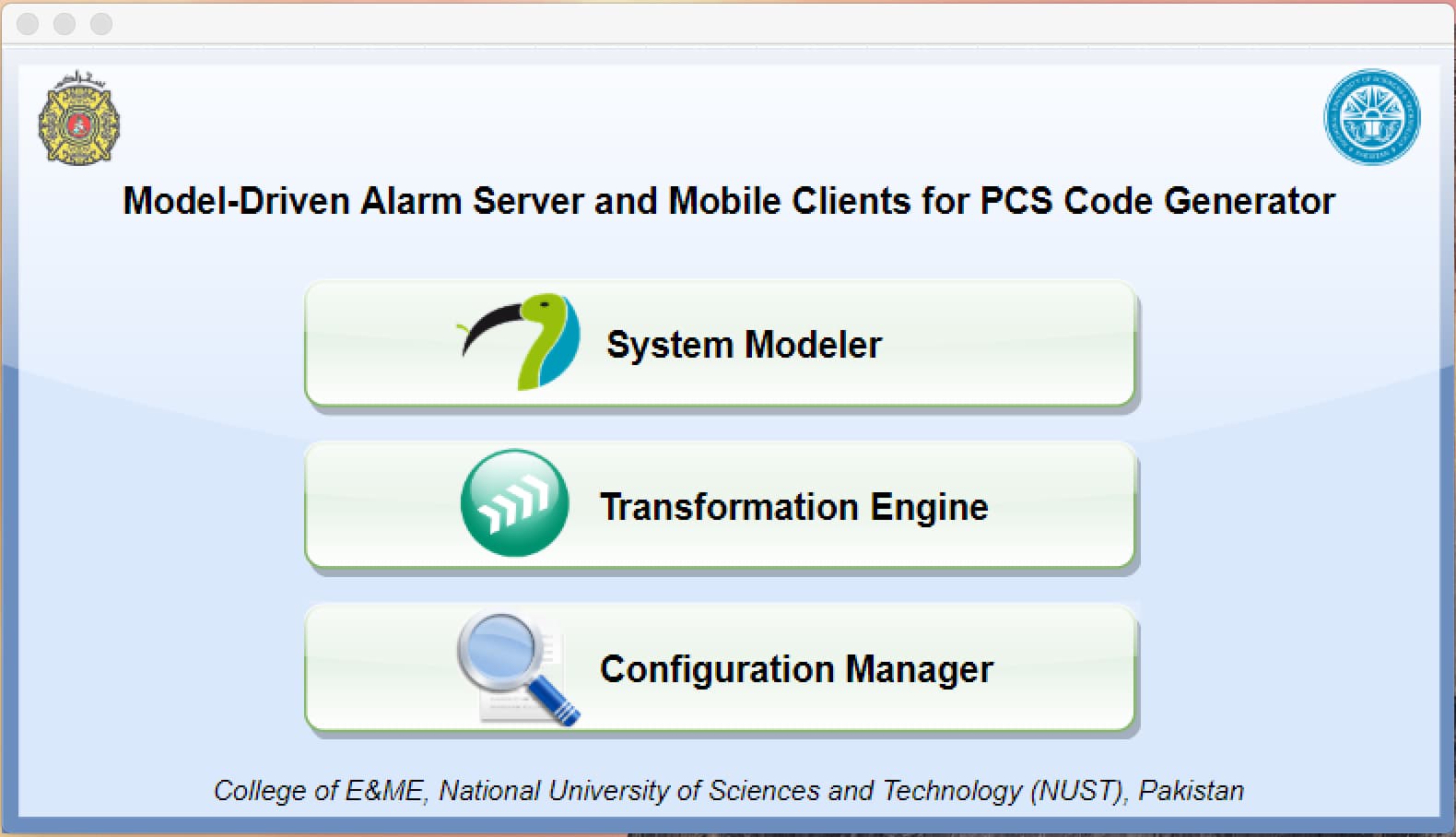
****

**Figure 3.**Opening Transformation Engine in Eclipse

1. Right click on “AppWindow.java” file and Run it as “Java Application” as shown in **Figure 4**.

****

**Figure 4.** Run as Java Application

It shall compile the project and show the Main interface of the MDASMCPCS as shown in **Figure 5**.

**Figure 5.** Main interface of MDASMCPCS

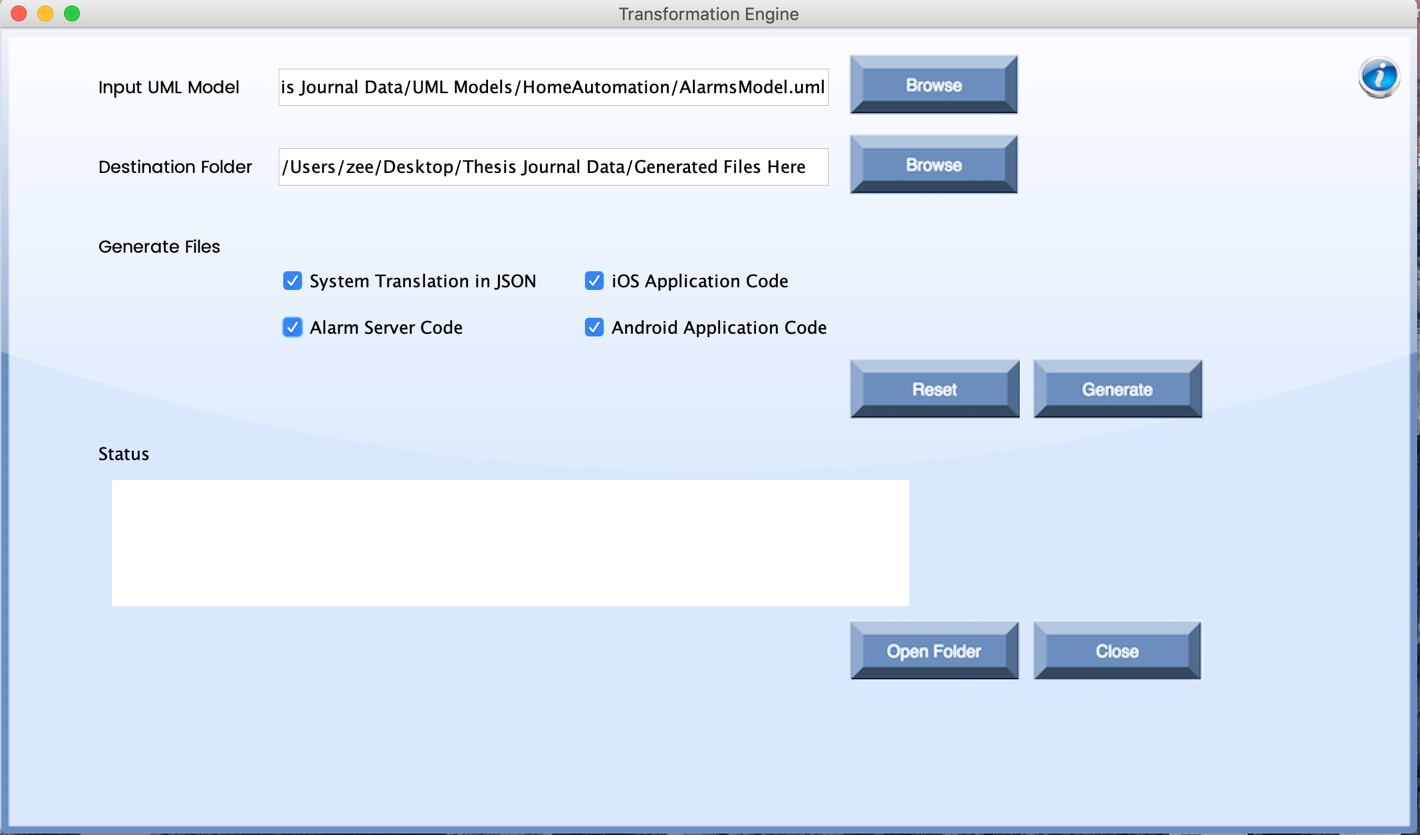
The MDASMCPCS contains three main functionalities.

* **System Modeler:** It allows you to model your own system or modify existing system using Papyrus in Unified Modeling Language (UML).
* **Transformation Engine:** It enables the user to select the UML mode of a system to generate its source code as per transformation rules deployed.
* **Configuration Manager:** It redirects the user to an UI editor to modify the generated document containing tags alarmed in JSON format. User can modify the associated information about alarms and their tags (i.e. associated PLC devices).
  1. **Papyrus**

By clicking on “**System Modeler**” Eclipse environment is opened and in order to allow IFML modeling “Eclipse IFML plugin” must be installed.

* 1. **Transformation**

By clicking on “**Transformation Engine**” in the main interface, interface for “**Acceleo** **Code Transformation Engine**” is opened as shown in **Figure 6**



**Figure 6.** Interface for Model to Code Transformation Engine

**Input UML Model:** Browse button is used to select the UML model for the case study.

**Destination Folder:** Browse button is used to specify the destination folder for the generated files.

**Generate Files:** User can select the required files from the given four options by checking the checkbox.

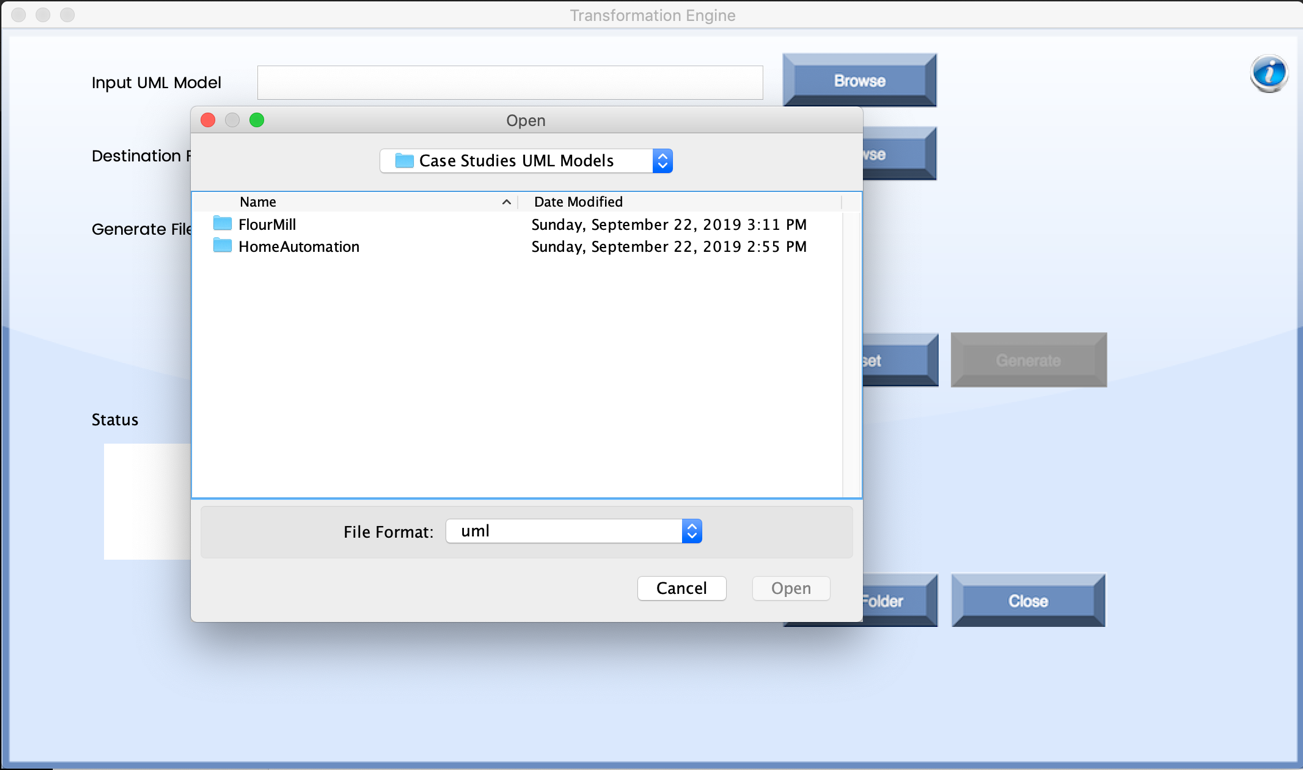
**Reset:** This button clears all the current selections to defile new configurations.

**Generate:** This button transforms the selected UML models into the required testing artifacts. It is mandatory to fill all the above field in order to click generate button.

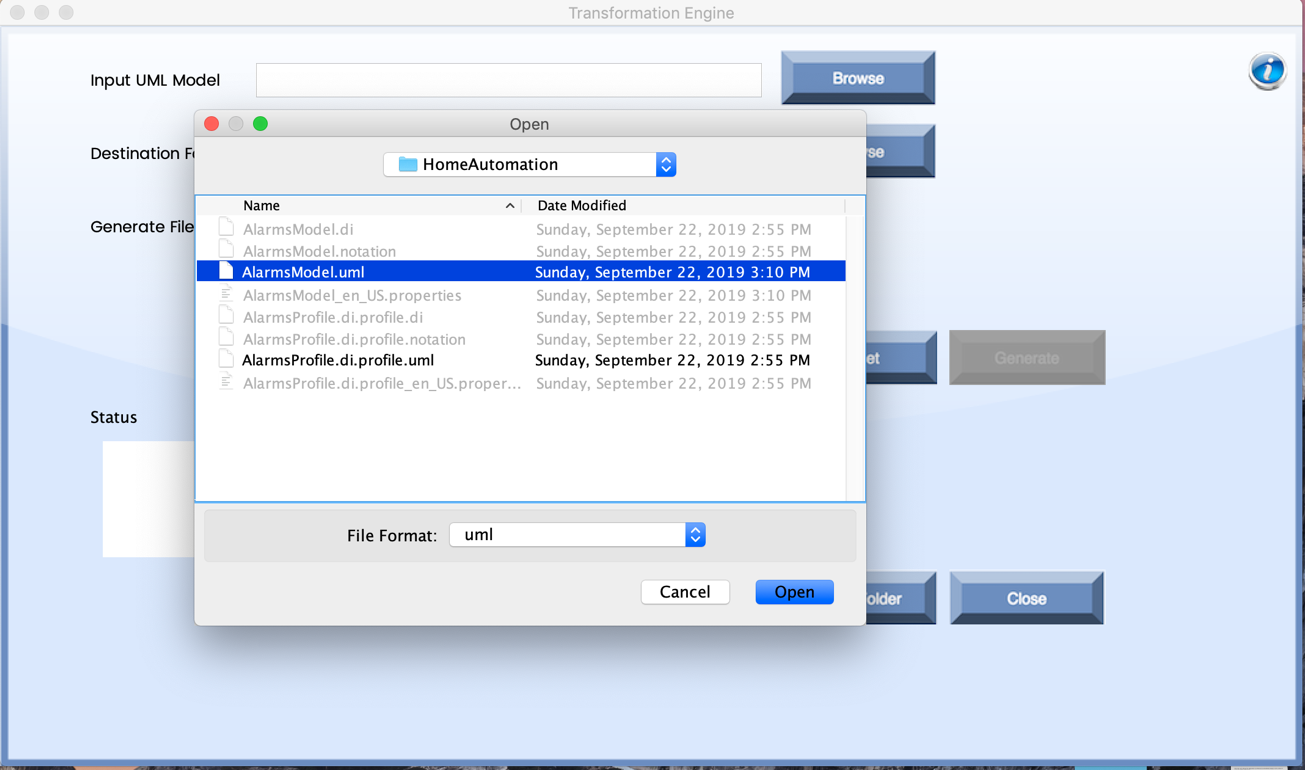
**Status:** This displays the status of current transformations i.e. List of generated files or Files Generated with Errors (in case of any problem in transformation).

**Open Folder:** This button is used to open the folder where output folder containing the generated files is placed.

**Close:** This button closes the interface.

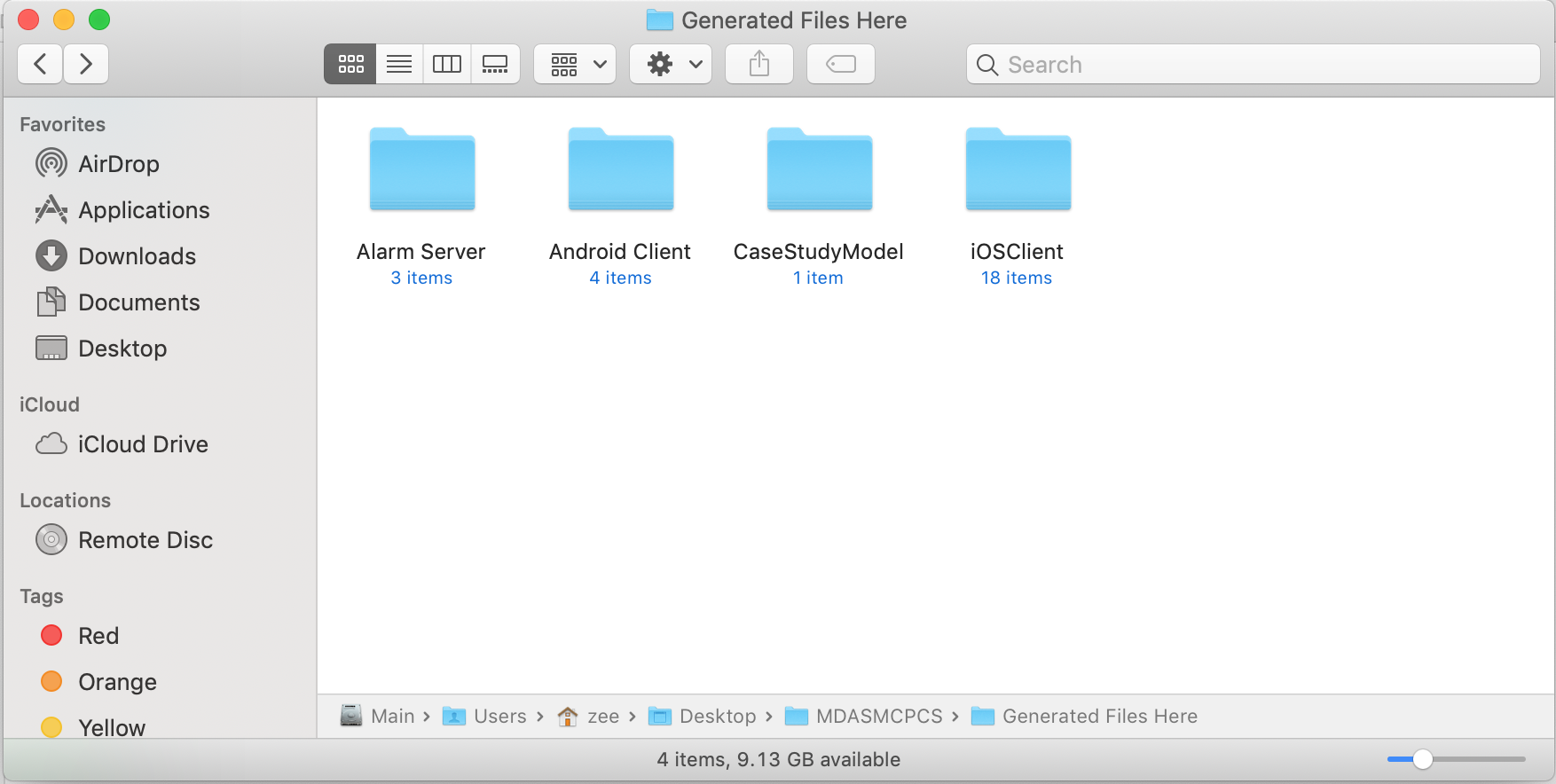


**Figure 7.** Selection of Case Study using browse button



**Figure 8.** Selection of UML model of Home Automation Case study

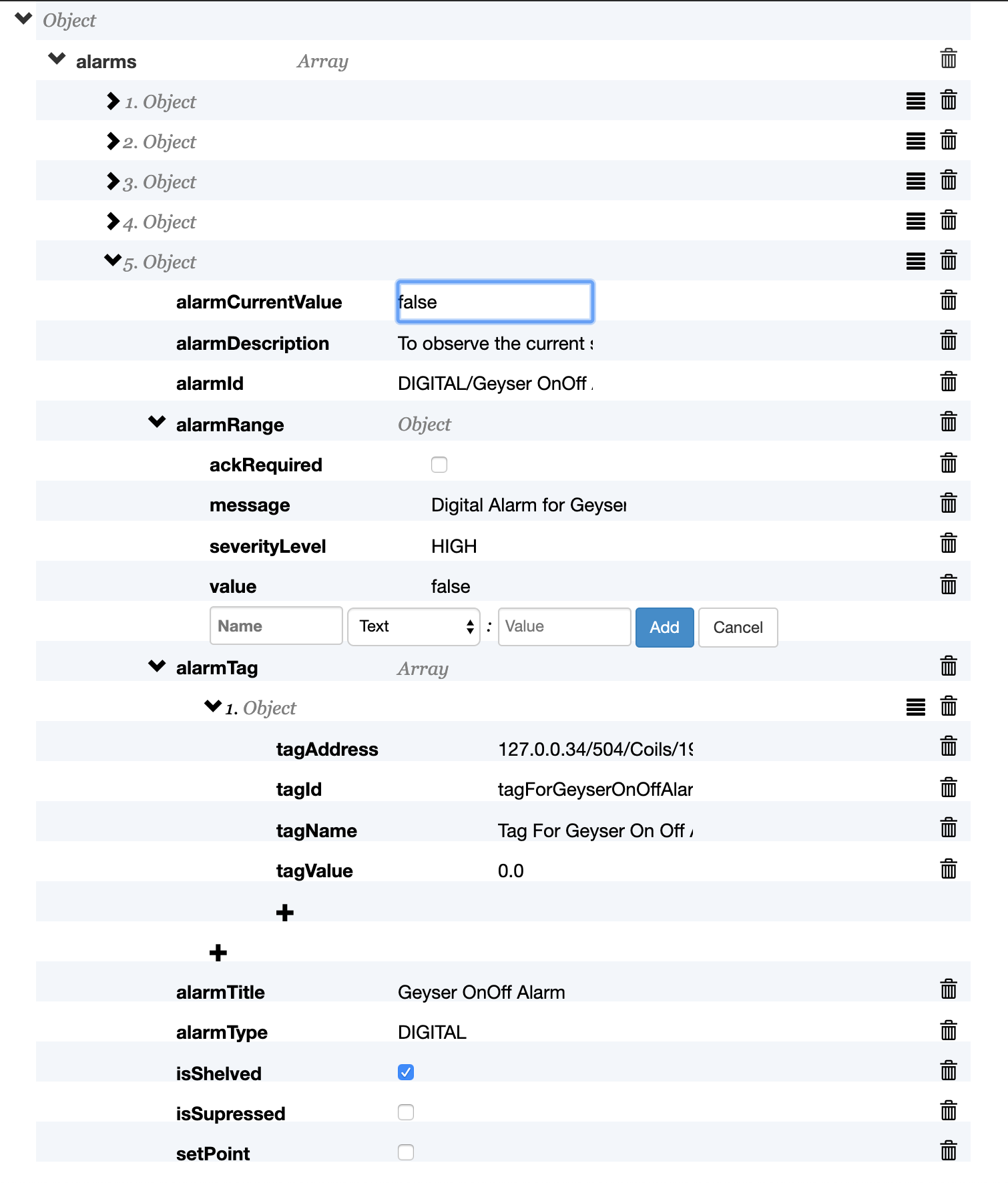
The UML models shall be transformed into desired textual Artifacts i.e. Alarm Server, iOS Client, Android Client, and JSON translation of the modeled Alarm system in “Generated Files Here” folder as shown in **Figure 9**



**Figure 9.** Generated Files of desired Textual Artifacts

* 1. **Configuration Manager**

It shall redirect the user to a web application for editing the alarms JSON generated in “CaseStudyModel” folder. User can copy the text and paste it there on web app in order to visualize validated JSON, add items, edit or even delete Alarm objects as show in **Figure 10**.



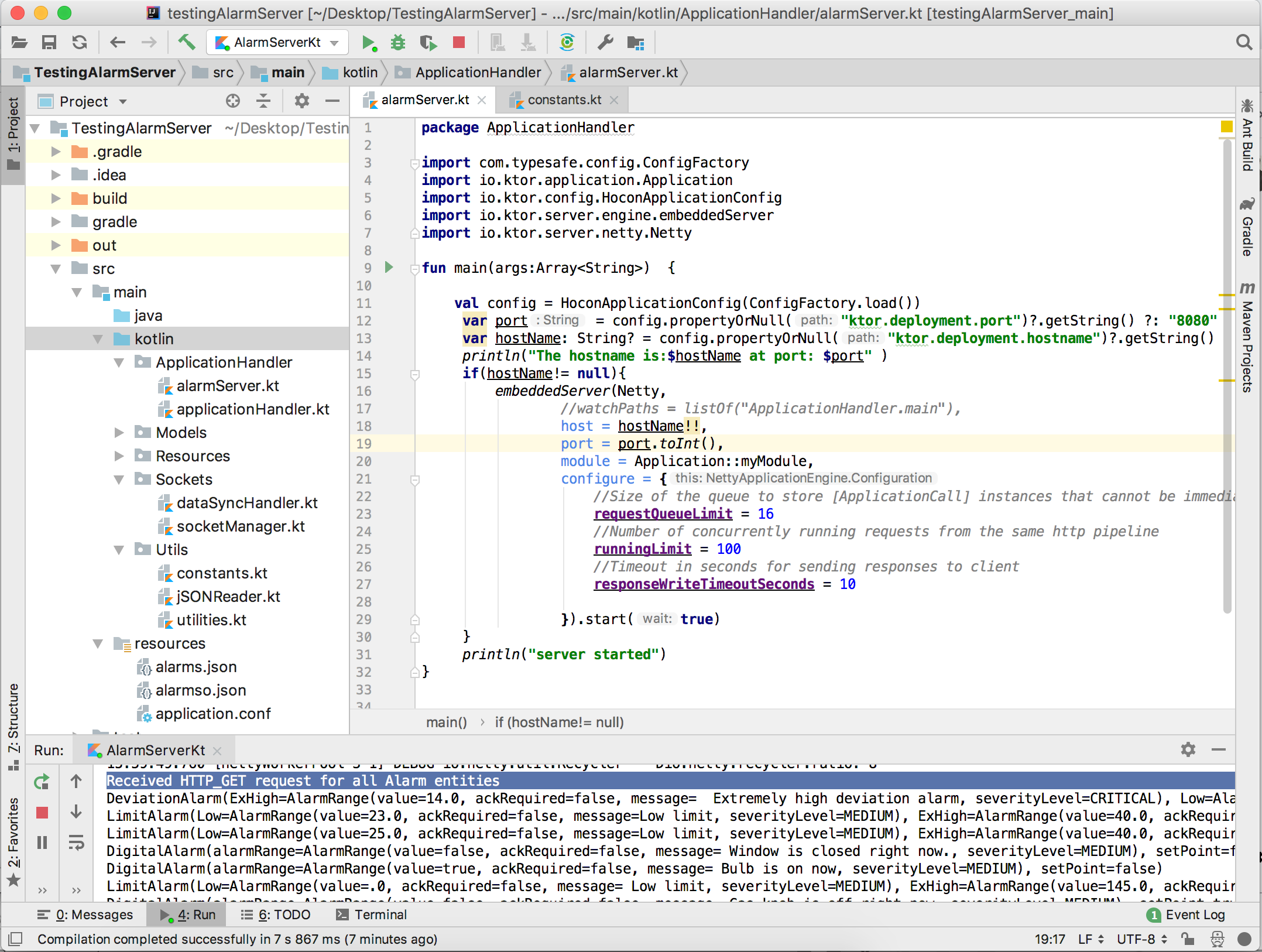
**Figure 10**. CRUD Operations on Alarms JSON to Configure Alarm Objects

* 1. **Verification**

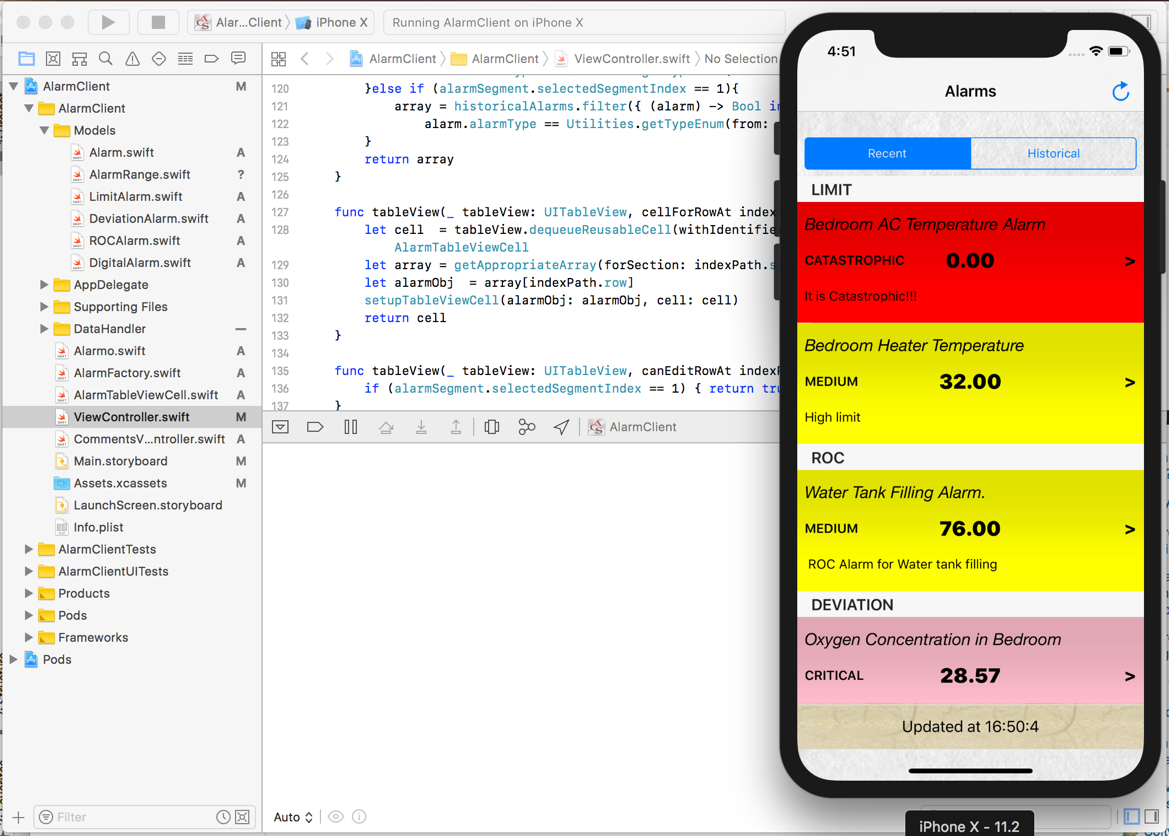
Since there are three types of desired artifacts i.e. iOS Client (in Swift Language), Android Client (in Kotlin Language) and Alarm Server (in Kotlin Language). Therefore, each application shall require a separate Compiler and Debugger to run the application i.e. Xcode, Android Studio and IntelliJ IDEA respectively.

User shall have to create a project in each IDE and the copy and paste the generated code files to run it. But for Alarm server code, user shall have to include “Alarm.json” file in the resources as well.

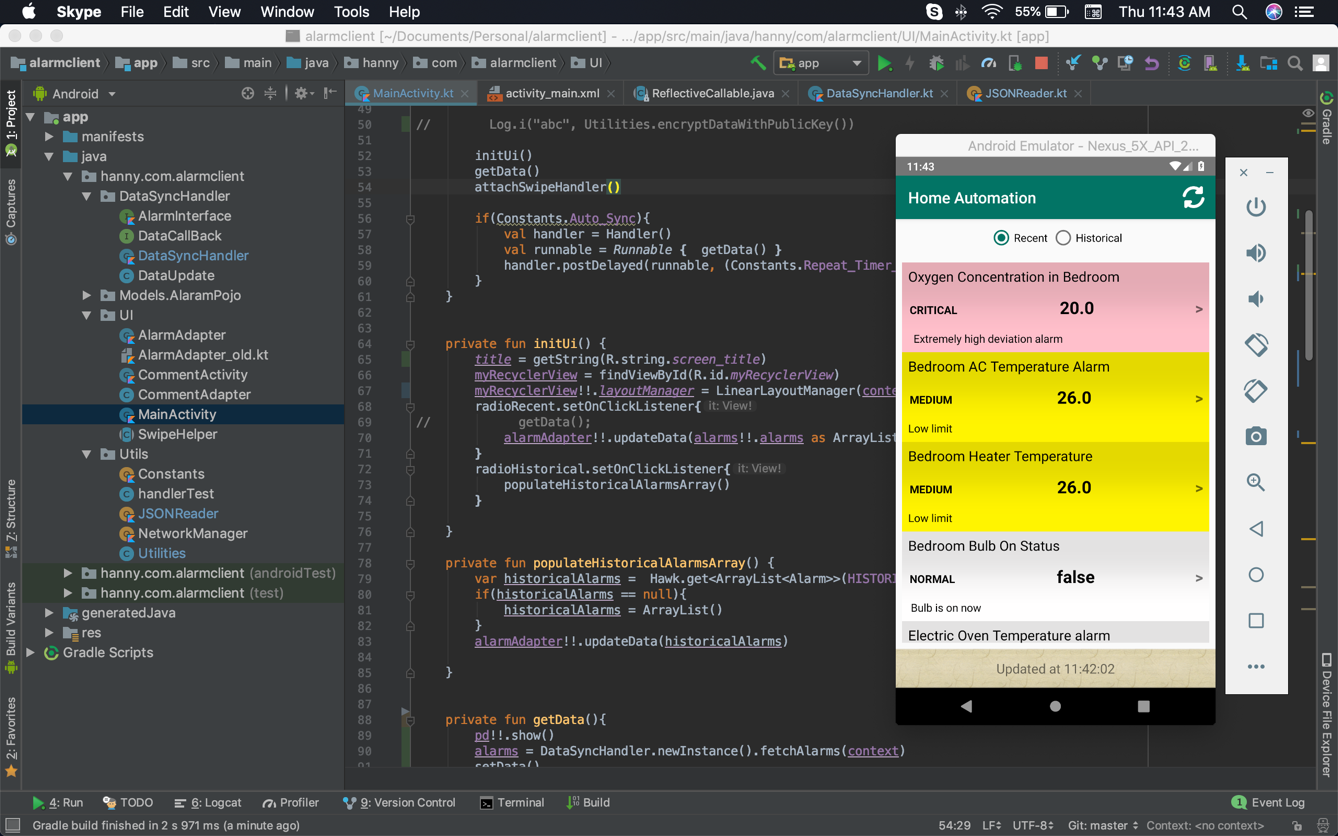
Following are the screenshots of deployed code for Alarm Server, iOS client and Android client in respective manner in **Figure 11**, **Figure 12**, **Figure 13**.



**Figure 11.** Deployed Alarm Server code in IntelliJ IDEA along with Alarms.JSON file



**Figure 12.** Deployed iOS Client code in Xcode for Home Automation Case study



**Figure 13.** Deployed Android Client code in Android Studio for Home Automation Case study