- **User Manual**
- *Run Application and Test*

Under root directory (IOT-System-JAVAFX-based), use “gradle run” to run application, and use “gradle test” to run tests. Note that supported resolution for test is 1280 x 800.

For thermostat temperature, the textfield is a bit awkward, but inputting a new temperature in textfield will update temperature in model. To test “camera is full”, simply click “Record” button several times and you will see “Camera is full” message showing up. For some acceptance tests that are not easy to test manually, they are already included in unit tests. Please see camera test case 6 in CameraTest.java, thermostat test cases 6 in ThermostatTest.java, Lightbulb test case 3 & 4 in LightbulbTest.java.

- *Log in*

There are two users available for initial log in.

User Type	**Username**	**Password**
AdminUser	10086	123456
BasicUser	10000	654321

- *Usability Guide*
  - When logged in as an admin user, two windows are overlapped, drag main interface window a bit to see admin interface window.
  - To safely shutdown the system, click “Shut Down” button on the top of admin interface. Note that windows are not closed intentionally so that you can verify device states are changed and logs generated. Also note that though thermostats are turned off, temperatures are recorded, so the next time you start the system, original temperatures will be shown.
  - For video in camera, clicking “Record” plays video, clicking “Stop Recording” pauses video, clicking “Turn Off” stops playing video.
  - To assign a device to a user, go to device list in admin interface and enter username beside one device, click “Assign” button to assign that device to user.
  - To check status of devices, click “Check Status” button on the top of admin interface and system will start checking status every 1 minute. The thread will not terminate until “Shut Down” button is clicked. Collected data is shown in activity logs section on admin interface.
  - For a table column that is too narrow to show all the content, please drag header area to adjust column width, especially for device uuid.
  - If you find a window without scrollbars and the content is not completely shown, try to resize window and the scrollbars will show up.

- **Proposed solution**
- *Assumptions*
  - There is only one admin user in system. There are two pre-added users (one admin and one basic) for testing purpose, and these two users are not removable. Only admin user can shut down the system.
  - Status is a dummy feature, please see buttons, labels (and alert list) to determine device state. This design choice is made especially because states for camera are complicated (off, on and not recording, on and recording) and it is hard to integrate those states within Status enum. In current design, boolean isOn is used to indicate if a device is turned on.
  - A basic user cannot see any devices until some devices are assigned to him.
  - Both admin user and basic user should manipulate device status on main interface.

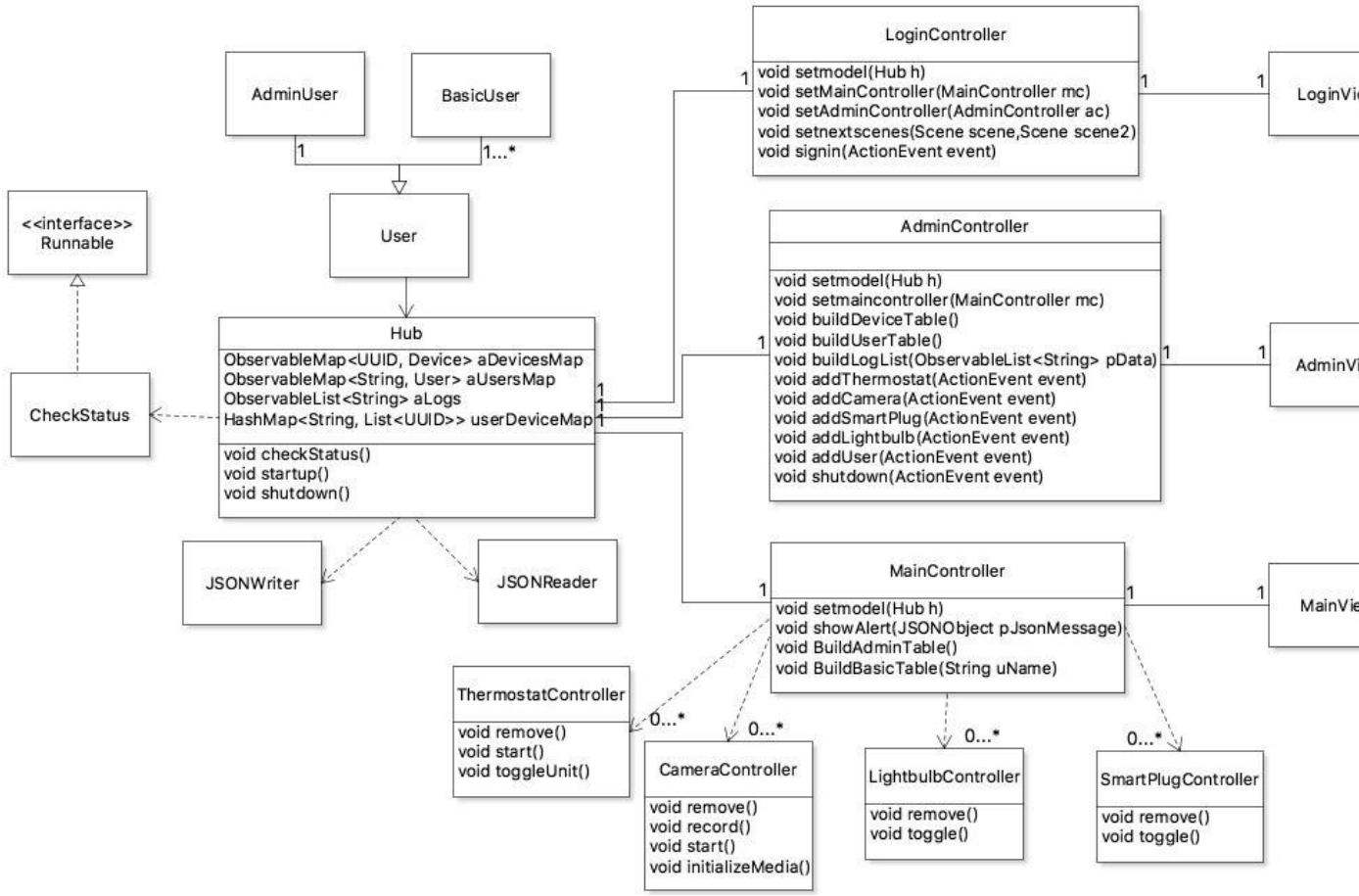
- *MVC*

Model:
Model includes devices, users and hub (mostly everything built in assignment 2), where hub (the mediator) is considered as an "interface" of model, and provides methods that controllers need. Helper classes like JSONReader and JSONWriter are added to assist startup() and shutdown() in hub. After user clicks "Shut Down" on admin interface, core data (device map, user map, log list, user device map) will be written into data.json, and those data will be read back into hub once the system starts up. Minimal changes are made to models (compared to assignment 2) and most business logic resides in controllers.

View:
Most views are built in FXML while devices/users/logs added at runtime are constructed in Java. There are three FXML files: AdminView.fxml, MainView.fxml and LoginView.fxml, representing admin interface, main interface and login interface, respectively. Note that main interface is the user interface that a basic user can see.

Controller:
There are three major controllers, one for each view: AdminController, MainController and LoginController, and they are used to get data from model and populate data on the view. Besides, to lessen the burden of MainController, we implement separate device controllers, namely CameraController, ThermostatController, LightbulbController and SmartPlugController. Each device has its own controller, i.e. when a new camera is created, a corresponding CameraController is created for it.

- *UML - Class Diagram*



Note that App class is omitted from class diagram for cleanliness. App class (entry point of

application) depends on Hub (model), all the views, and all the controllers. Components built in assignment 2 (e.g. devices) are omitted.