DS – SE3020

Assignment 2 – REST API

Note: This assignment carries 20 marks.

This is a group project of max. 2 members. You may stick to the same group of the assignment 1 or form a different group. If you want, you can do it individually. Irrespective of whether the submission is individual or group, the marking standard will be the same.

Assume that you have been asked to develop a fire alarm monitoring system. Following are the requirements given by the client and/or the Business Analyst.

- The system should have a web client application where users can view the status of all fire alarm sensors. For each sensor, the web application should display whether the fire alarm sensor is active, the location (floor no, room no), and the smoke level (1-10) and the CO2 level (1-10 scale). If the smoke level or CO2 level is above 5, then they should be marked in red. You can decide on the user interface as you wish. The sensor details should be updated every 40 seconds.
- A desktop client application where users can view the same information from a desktop client. The information is refreshed every 30 seconds.
- The desktop client application should have an administrator login, where an administrator can add/register new fire alarm sensors. To register, the floor number and the room no should be given. The administrator can edit sensor details as well.
- An alert can be displayed on the desktop client when the CO2 level or Smoke level is moves to a value greater than 5, of any sensor. An email and an SMS can be sent in such an occasion.

1. Based on the above information, come up with a set of RESTful web services API to implement the system (you may use any technology to implement the services). You may use any web server to deploy the services (e.g. Apache Tomcat if you're using Java).

2. Develop an Asynchronous web client, using which the users may access the system. You may use any Javascript framework that supports asynchronous programming (Angular, React, etc.) to do this. Alternatively, you can also use regular JQuery + AJAX to develop the client. The client may connect to the REST API.

3. Develop an RMI Server and an RMI desktop client as the desktop client application. The RMI Server may connect to the REST API. The RMI server may send the Email/SMS updates when the relevant event happens (when the CO2 or smoke level goes up). For sending Emails/SMS messages, you need not use actual services and you can just use dummy services. If you can find an actual service to do those, you may use it. RMI Server may check the sensor status every 5 seconds to get the upto-date readings.

4. You may use any database as your datastore to store sensor information and user information. Depending on the requirement, a standard relational database would be suitable as the data is mostly structured.

5. You may implement a simple client application to simulate the behavior of a Fire sensor. These applications should send the fire alarm status to the REST API every 30 seconds. You can run multiple applications of this kind to emulate multiple sensors.

Deliverables

A readme.txt file with the following details.

- *T*he student registration nos, names, contact nos, contact emails.
- Create a demo video of the working project (you may use a screen capturing tool like OBSStudio to create it https://obsproject.com/). Upload the video to youtube. Add the link to the youtube video to the readme.txt. Video length may be 5-10 minutes. You may explain the features of the system and also the technical details as you wish.
- Create a github project. Copy the link to the github project with the project deliverables. Github project should contain the following.

1. Source code and binaries of the RESTful Web Services.
2. Source code and binaries of the web client.
3. Source code and binaries of the RMI Client/Server.
4. A *install.txt* document, listing down the steps to deploy the above deliverables.
5. Any database scripts or any other data-store documents (xml documents, flat files, etc.) that you may have used to store the sample data.
6. An 8-10 page report. The report should include a high level architectural diagram showing the services and their interconnectivity. Also, it should list out the service interfaces exposed by each service and should briefly explain each of the workflows used in the system (you may use sequence diagrams to do this). You may add additional design diagrams as you wish. You can also include the details about the authentication/security mechanisms adopted (it is not mandatory to implement security related features).

   You may use code snippets in the report to explain the above.

   The report **MUST** have an appendix with all the code that you have written (excluding the auto-generated code). **Do not paste screenshots of the code in the appendix and copy the code as text.**

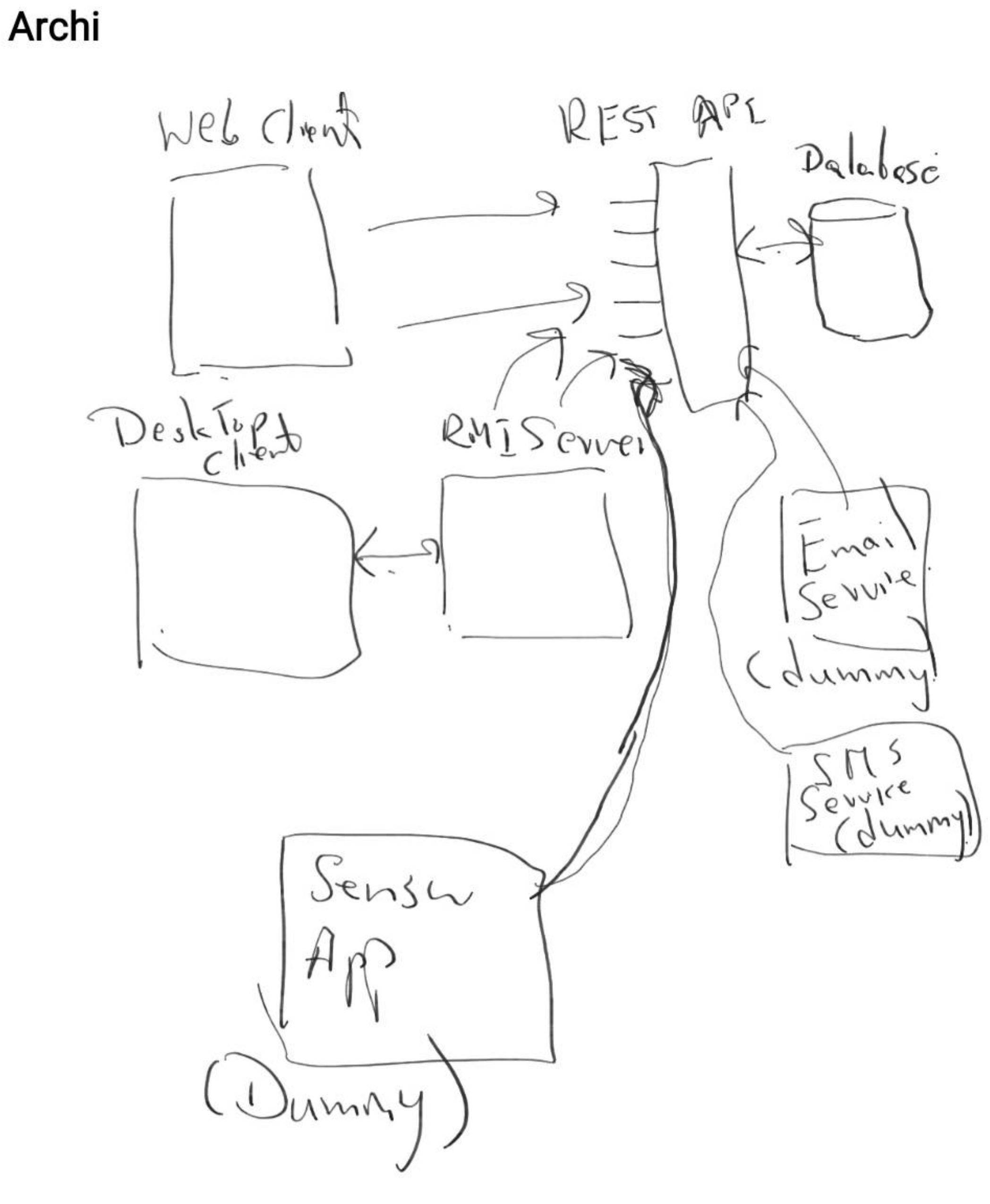
Note: All reports will be uploaded to Turnitin for plagiarism checking. If the turnitin similarity is above **30%**, **10%** of the marks will be reduced. For **50%** similarity, **50%** of the marks will be reduced. For reports with **80%** similarity, **no marks** will be given.

Note: If any of the above components are missing, the submission will be regarded as incomplete and no marks will be given.

Submission: The readme.txt file containing the github project link, student details and the youtube link.

Note:

Sample System Architecture Diagram (note that this is just one possible approach. You may come up with your own system architecture if you wish).



Marking Rubric

Following rubric will be used in evaluating the assignment.

Criteria	**Good (10-8)**	**Average (4-7)**	**Poor (0-3)**
Application of SOA principles in the architecture and the design			
Having clearly defined component interfaces, that facilitate reusability			
RMI Server/Client implementation			
Quality and the readability of the code, with meaningful and detailed comments.			
Comprehensiveness and the quality of the report			