Assignment 2 – REST API

Version 03

Note: This assignment carries 20 marks. This is a group project.

Assume that you have been asked to develop a collaborative Agri product online purchasing platform. Following are the requirements given by the client and/or the Business Analyst.

- The system should have a web interface where buyers can shop for items uploaded by farmers.
- A service should be there where farmers can add/update/delete items.
- A service should be there where buyers can search/buy items.
- A buyer may buy multiple items.
- Once an item or a collection of items are added to cart, the buyer may select the delivery option, where a request may be sent to a delivery service (so there should be a third-party delivery service).
- The payment for the items bought can be made using credit cards or using the mobile phone number (which would be added to the user's mobile bill).
- The system can connect to a payment gateway for credit card transactions. The information that should be submitted includes the credit card number, amount, CVC number (3 digit no. at the back of the credit card) and card holder's name.
- The users can choose to make the payment using a mobile company's service to credit the mobile bill. The information that should be transferred includes the mobile phone number, a six-digit pin number and the amount to be charged.
- Once the payment is made the user should be given a confirmation of the appointment via SMS and email.

Implementation

1. Based on the above information, come up with a set of RESTful web services to implement the system (you may use any technology to implement the services).

2. Use the WSO2 EI (Enterprise Integration – ESB) to integrate services at the backend and expose a common web API.

   For example, you can do some transformation at the EI to route the payment to either the banking payment gateway or the mobile operator, based on some parameter of the payment request message.

   Hint: Refer the following documentation on ESB service integration for a guide to do this.

   https://docs.wso2.com/display/EI660/Routing+Requests+Based+on+Message+Content

   You can expose the rest of the services also through the WSO2 EI to the client. The advantage of this would be that the client(s) will see the same web API and do not have to access different services at the back-end. The WSO2 EI will route each request to the relevant service at the back-end.

3. 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. You can also use regular JQuery + AJAX to develop the client.

   Since there's a REST api in the backend, other types of clients (e.g.mobile clients) can reuse the backend business logic easily in the future. However, for the scope of the assignment, implementing just an asynchronous web client is sufficient.

4. Use appropriate security/authentication mechanisms to uniquely identify each user and to authenticate each user. There should be two roles, buyer, and farmers.

Deliverables

1. Source code of the RESTful Web Services.
2. Source code of the web client.
3. WSO2 Enterprise Integration project (developed using Eclipse Developer Studio).
4. A readme.txt document, listing down the steps to deploy the above deliverables.

5. Members.txt file, containing the names, registration numbers and the IDs of the group members.
6. Any database scripts or any other data-store documents (xml documents, flat files, etc.) that you may have used to store the sample data (e.g. shopping item details).

7. An 10-12 page report in pdf format. The report should include a high level architectural diagram showing the services and their interconnectivity. Also, it should list out the interfaces (NOT the user interfaces, but the service interfaces) exposed by each service and should briefly explain each of the workflows used in the system (you may use diagrams to do this). You can also include the details about the authentication/security mechanisms adopted.

Important

- 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. If screenshots are added in the appendix, marks will be reduced.**
- The report should be no less than **10 pages (excluding the appendix)** of length. **The report is the main component that should be marked. However, the code should be there to validate the implementation.**
- **Note:** You may **implement dummy services** to simulate the payment gateway, mobile payment gateway and the delivery service. For email and SMS notifications, you may try to use an available service on the Internet. If you cannot find one, you can use dummy services to implement those as well. **For instance, the dummy payment gateway service can accept the relevant set of input parameters and just return a message saying the "payment successful", rather than doing an actual payment. You can implement similar dummy services for the sms, email, mobile payment and delivery services.**
- **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 your submission size is larger than 10 MB, you may upload the submission to Dropbox (**only use Dropbox**) and share the link. If you're sharing the link, include the dropbox link in the readme.txt file. Make sure that it is properly shared and accessible. www.dropbox.com

- **Submission:** All files should be uploaded in a single zip archive. The zip file name should be your SLIIT registration number of the member who is uploading the submission. Only one member needs to upload the submission. All members will get the same mark.

Important

Use the following directory structure the upload the answer. You may zip the entire folder and name it using your registration number.

<<Reg No>>

- Services (contains the REST services)
- Client (Web client source files)
- Readme.txt (instructions on how to deploy the deliverables, if you're sharing a dropbox link, copy the link here)
- Members.txt
- WSO2 Enterprise Integration Eclipse project (developed using developer studio)
- Report (including the appendix) in pdf format.

Note: If you find it too difficult to do the service integration using WSO2 EI, you may skip it and directly integrate the services at the client. You will only lose the marks for the service integration (5th point of the marking rubric) and you may get the marks for the rest.

Marking Rubric

Group Submission (report, implementation, and coding)

Criteria	Good (10-8)	Average (4-7)	Poor (0-3)	Comment	Marks
Application of SOA principles in the architecture and the design	Identified the suitable architecture and SOA principles are exceptionally applied	Average architecture and not properly designed structure	Not properly planed before the development and very poor design		
Having clearly defined interfaces, that facilitate reusability User authentication and security mechanism	Identity and design perfect user-friendly client-side UI's	Identified all the UI's but not reusable	Poor UI, less than 3 interfaces		
Quality and the readability of the code, with meaningful and detailed comments.	Good coding practice with proper coding standards	Average level of coding with less quality yet working code.	Very basic level of coding with errors		
Integration of services using the Enterprise Service Bus (ESB)	All the services are properly integrated	(1-2) services are missing /not working	Poor service integration more than 2 services are not working		
Comprehensiveness and the quality of the report	Service interfaces and architecture diagram /system diagram are perfectly included. workflow explained and code snippets included.	Some components are missing but average report with essential components	Irrelevant component in the report as user UI and code screenshots. Poor report		
				Total/50	

Individual VIVA (In this section marks will vary based on the individual performance)

Criteria	Very Good	Average	Poor	Comment	Marks
Demonstration of the part requested and have an overall idea about the entire project. implemented project (level of implementation, follows coding standards and best practices)	Very good understanding on entire system and demonstrates strong development skills. (15-20)	substantial contribution and implemented the project to an average level. (9-14)	Some very basic level of implementation and no proper understanding on the entire system (1-8)		
Understanding of the web services	Able to explain the given service (7-10)	Average understanding on webservices (5-7)	Poor understanding on rest webservices (1-4)		
WSO2 EI integration /mobile client/web client	Able to explain any given part and good contribution (7-10)	Sustainable contribution to average level (5-7)	Poor engagement (1-4)		
Presentation and Communication Skills	Very Good (7-10)	Average (5-7)	Poor (1-4)		
				Total/50	