Distributed Systems (SE3020)

Programming project

Note: This project carries 30 marks. This is a group project. The group size is 3 to 4. It will be followed by a Viva.

Assume that you have been asked to develop a collaborative shopping platform for Ayurvedic/Herbal medicines and supplements (similar to https://www.iherb.com) . 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 sellers.
- A service should be there where sellers 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 is purchased, an administrator may manually verify the order and confirm the order.
- Once an item or a collection of items are bought, 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 such as DHL).
- From each purchase, a commission will be charged as the main revenue source of the platform. The commission should include the fees of the payment service.
- The payment for the items bought can be made using credit cards or paypal/payhere like payment integration services.
- 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.
- Once the payment is made the user should be given confirmation of the purchase via SMS and email.
- An interface should be there to track the status of an order (whether it's pending, confirmed, dispatched, or delivered).
- There should be a way for the users to review and rate suppliers as well as individual products.

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.

   Note. The WSO2 Enterprise Integrator 6.6 is no longer officially supported by WSO2. However, you may use the 6.6 installation updated to courseweb instead. Alternatively, you may use Enterprise integrator 7.1 available at the WSO2 web site. (https://wso2.com/micro-integrator/previous-releases/). If you're using any other ESB tool for integration, you may justify that in the report and viva.

   Alternatively,

   You may use the **Microservices architecture** to develop/integrate the API, in which case you need not use the WSO2 Enterprise Integrator. If you're using the Microservices architecture, make sure to use **Docker** and **Kubenetes**. (**optionally, you may use WSO2 Micro-integrator as well - https://wso2.com/integration/micro-integrator/). If you're using any other tool for Microservices orchestration/integration, you may justify that in the report and the viva.**

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 three roles, buyer, seller and administrator.

Deliverables

1. A text file called 'submission.txt', containing a Github repository link containing all the source code. The source code should contain the source of the back-end project containing the API, client, and any other relevant source/resource files (e.g. database scripts), arranged in a proper directory structure.
2. The 'submission.txt' should contain a youtube video link of a presentation/demo of the project. Each member may use maximum 3 mins to explain their contribution, so that the total video length should be no more than 12 mins.
3. A 'readme.txt' document, listing down the steps to deploy the above deliverables.
4. A 'members.txt' file, containing the names, registration numbers and the IDs of the group members.
5. A report in pdf format. The report should include a high level architectural diagram showing the services and their interconnectivity. Also, it should list 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 design diagrams of your choice to do this). You can also include the details about the authentication/security mechanisms adopted.

   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, only the minimum mark may be offered.**

Note: You may **implement dummy services** to simulate the external third-party services such as the payment and delivery services (if it's possible to use a real payment platform such as payhere sandbox environment, you may do so). 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 **20%, marks will be penalized**.

Submission: All files should be uploaded in a single zip archive. The zip file name should be 'DS-Assignment'. Only one member needs to upload the submission.

Submission Deadline: 11th week of the semester

Marking rubric is given below.

If using the WSO2 Enterprise Integrator

Criteria	Good (10-8)	Average (4-7)	Poor (0-3)
Application of SOA principles in the architecture and the design			
Having clearly defined interfaces, that facilitate reusability			
Quality and the readability of the code, with meaningful and detailed comments.			
Integration of services using the Enterprise Service Bus (ESB)			
Adoption of appropriate authentication/security mechanisms			
Comprehensiveness and the quality of the report			

If using the Microservice Architecture with Docker + Kubernetes

Criteria	Good (10-8)	Average (4-7)	Poor (0-3)
Application of Microservices principles in the architecture and the design			
Having clearly defined			

interfaces, that facilitate reusability			
Quality and the readability of the code, with meaningful and detailed comments.			
Integration/Orchestration of services using Docker + Kubernetes			
Adoption of appropriate authentication/security mechanisms			
Comprehensiveness and the quality of the report			