MCAST

ASSESSMENT AND INTERNAL VERIFICATION FRONT SHEET (Individual Criteria)

(Note : This version is to be used for an assignment brief issued to students via Classter)

Course Title	**B.Sc. (Hons.) in Software Development**		**Lecturer Name & Surname**	**Owen Sacco**	
Unit Number & Title	ITSFT-606-2101 - Distributed Programming				
Assignment Number, Title / Type	Assignment 2: Distributed Programming Home Assignment				
Date Set	19/04/2023	**Deadline Date**	21/05/2023		
Student Name		**ID Number**		**Class / Group**	

Assessment Criteria	Maximum Mark
KU2.1: Describe the context of a Microservices Architecture	5
SE2.3: Construct a solid and robust Microservices Architecture	10
KU3.1: Describe the context and core principles of an Event-driven Architecture	5
AA3.3: Use emerging libraries to consume Web Services and to persist data on the Client	7
SE3.4: Construct a solid and robust Event-driven Architecture	10
KU4.1: Describe the advantages and disadvantages of cloud computing	5
KU4.2: Describe the main services provided by cloud services	5
AA4.3: Use Cloud Web Services to create and deploy a Web Server	7
AA4.4: Use third party Web Service APIs to consume data	7
Total Mark	61

Notes to Students:

- This assignment brief has been approved and released by the Internal Verifier through Classter.
- Assessment marks and feedback by the lecturer will be available online via Classter (Http://mcast.classter.com) following release by the Internal Verifier
- Students submitting their assignment on Moodle/Unicheck will be requested to confirm online the following statements:

Student's declaration prior to handing-in of assignment

- I certify that the work submitted for this assignment is my own and that I have read and understood the respective Plagiarism Policy

Student's declaration on assessment special arrangements

- I certify that adequate support was given to me during the assignment through the Institute and/or the Inclusive Education Unit.
- I declare that I refused the special support offered by the Institute.

Assignment 2: Distributed Programming

Home Assignment

Assignment Guidelines

Read the following instructions carefully:

- The assignment coversheet should be the first sheet in your assignment. Moreover, the coversheet should be fully completed with all the necessary details.
- You are required to use .NET technologies for this assignment, and you may use any library or framework for developing the event-driven architecture and front-end Web application.
- All text/code *must be properly referenced*. In the absence of proper referencing, the assignment will be regarded as plagiarised.
- **Copying is strictly prohibited and will be penalized** in line with the College's disciplinary procedures.
- The deadline to submit all deliverables is **21/05/2023**
- You are required to submit all your assignment deliverables (as explained in the assignment brief) via the links on VLE/Moodle.
- You are required to commit all project files to a Git repository with frequent commits.
- You are required to host your microservices and Web application on free-hosting services and make sure that they are running and accessible.
- You are required to record a (video) demonstration of your project and ensure that you set the correct permissions by sharing it only with the lecturer.

eCommerce System

An eCommerce system handles customer orders, payments and shipping of ordered items to customers. In this assignment, you will be developing an eCommerce system for any product categories of your choice that provides the functionality as outlined in this brief. The eCommerce system should be designed using the Microservices architecture and the Event-driven architecture.

You are required to develop event-driven microservices and a Web application. When the user selects any of the features (outlined in this brief) from the Web application it will send the request to the microservices and the microservices will send back the response in JSON. The microservices should therefore have endpoints for each feature that will send back the response represented in JSON format. The Web application will parse the JSON formatted response and present the results accordingly.

The eCommerce system will consist of the following microservices:

- Customer microservice – this handles customer details including user account details for logging into the system and notifications about their orders.
- Products Catalogue microservice – this handles product catalogues by retrieving product details (including prices) from external third-party APIs such as eBay and/or Amazon APIs on Rapid API (i.e. no product details need to be stored in a database but only retrieved from the third party APIs).
- Order microservice – this handles customer orders and logs past customer orders.
- Payment microservice – this handles customer payments and audit trails customer transactions.
- Shipping microservice – this handles shipping of orders, tracks shipping and records past shipping events.

Users must first register for a new account within the system by providing generic details such as first name, surname, email (which will be used as a username too) and password. Once a user logs in, the user will be able to view a catalogue of product categories and once the user selects a product category, the user will then be able to see a list of products (related to the selected product category) extracted from external third-party APIs such as eBay and/or Amazon APIs on Rapid API via the product catalogue microservice. The system should provide users to order, pay and ship products through an event-driven architecture. When a user confirms an order by confirming a payment, this triggers an event that will call the respective microservices to create the order, log the payment, record the shipping details and notify the customer that the order has been confirmed and created. The system's base currency will be in Euro however the system should be able to support other currencies when customers pay for their orders by retrieving exchange rates obtained from external APIs such as Exchange Rates API or Currency Scoop.

Users should also have a feature where the user can view current orders (not yet received) and past orders. When selecting any order, the user should be able to view order details such as the product(s) ordered with their individual price, the total ordered cost (including shipping cost), and also shipping details. The system should also provide administrators to log in and change the shipping status of the orders. The system should provide the following shipping statuses: 1) "ordered received not yet dispatched", 2) "order dispatched" and 3) "order delivered and received". When an order is complete, its status is automatically set to the first status (i.e. 1) "ordered received not yet dispatched"). Administrators should be able to change the shipping status through an event-driven architecture that will trigger an event and call the respective microservices to change the order status and notify the customer about the change in their order status.

Moreover, the system should provide an inbox feature where users can view any notifications received by the system. Users should receive notifications when an order is complete (i.e. once a user confirms the payment of an order) and when the order's shipping status is changed.

All data within the system should be logged and stored in a cloud-based database such as Google Cloud SQL, Cloud Firestore, MongoDB or as preferred.

You should therefore implement the following features and requirements (tasks):

- **[Task 1]** A Customer microservice that allows users to manage their account. This microservice should:
    - Allow a user to register for a new user account with the eCommerce system (through the microservice) - details should be stored in a cloud-based database.
    - Allows a user to login into their user account (through the microservice).
    - Provide user account details when requested from the Web application - i.e. the Web application should have a user account details page containing user account details and preferences.
    - Allow users to receive and view system notifications (i.e. inbox).

[3 marks]

- **[Task 2]** A Product Catalogue microservice that retrieve products and product details from external third-party APIs. This microservice should:
    - Retrieve a list of products for a specific product category (through external third-party APIs).
    - Retrieve product details for a specific product (through external third-party APIs).

[3.5 marks]

- **[Task 3]** An Order microservice that allows users to manage orders. This microservice should:
    - Allow users to create orders – orders are created through an event (see task 6) that is triggered once the user confirms the order payment from the Web application.
    - Allow users to view a list of orders.
    - Allow users to retrieve order details for a specific order.

[3.5 marks]

- **[Task 4]** A Payment microservice that handles payments and audit trails payments. This microservice should:
    - Allows users to pay for orders – payments are logged through an event (see task 6) that is triggered once the user confirms the order payment from the Web application.
    - Retrieve order payment details.

[3.5 marks]

- **[Task 5]** A Shipping microservice that handles shipping of orders. This microservice should:
    - Allows orders to be shipped – shipment details are logged through an event (see task 6) that is triggered once the user confirms the order payment from the Web application.
    - Allow order shipping status to be updated by administrators – order shipping status is changed through an event (see task 7) that is triggered once the administrator changes the status of an order through the Web application.
    - Retrieve order shipping details.

[3.5 marks]

- **[Task 6]** An event-driven function that will create an order once the user confirms the payment of the order from the Web application. This event should create the order, log the payment, record the shipping details and notify the customer that the order has been confirmed and created.

[5 marks]

- **[Task 7]** An event-driven function that will allow administrators to change the shipping status of an order through the Web application. This event should update the order shipping status and notify the customer about the change. The system should provide the following shipping statuses: 1) “ordered received not yet dispatched”, 2) “order dispatched” and 3) “order delivered and received”. The first status (i.e. 1) “ordered received not yet dispatched”) is set by default once an order has been created.

[5 marks]

- **[Task 8]** A Web application that provides a user-friendly interface to allow the user to interact with all the features provided by the microservices.

[4.5 marks]

- **[Task 9]** The microservices endpoints should provide the data formatted in JSON. The Web application should be able to parse the JSON data and display the results accordingly.

[2.5 marks]

- **[Task 10]** The microservices should interact with a cloud-based database that stores the user’s account details, user notifications, order details, payment details and shipping details.

[2.5 marks]

- **[Task 11]** Upload and deploy your microservices and Web application online to any free hosting of your choice such as Google App Engine, Back4App, Free ASP.net Hosting, MyASP.net or Firebase.

[7 marks]

- **[Task 12]** The Web application should be able to communicate with the microservices hosted online.

[2.5 marks]

Furthermore, you are required to record a video demonstration of not longer than 20 minutes. In the video recording you should demonstrate that all features are working through the Web application and also through testing tools such as Swagger. You are also requested to briefly explain your code demonstrating how you have implemented each feature. Therefore, it is important that in your video recording you:

- **[Task 13]** Explain how each microservice is implemented, deployed, and working through the Web application and testing tools such as Swagger.

[5 marks]

- **[Task 14]** Explain how the event-driven architecture is utilised in the eCommerce system by explaining how each event function is implemented, deployed, and working through the Web application and testing tools such as Swagger. Moreover, you are required to demonstrate using both customer accounts (for demonstrating orders) and administrator accounts (for demonstrating changing order shipping statuses)**.**

[5 marks]

- **[Task 15]** Explain how you have deployed the Web application, microservices and database to cloud-based hosting services. Moreover, demonstrate that Web application, microservices and database hosted online on cloud-based services are working and accessible. Furthermore, in your explanation, explain the advantages and disadvantages of using cloud services for hosting Web applications, microservices and databases.

[5 marks]

Deliverables:

1. Upload your code to a Git repository and submit the link to your repository on VLE/Moodle (make sure that your Git repository is accessible). It is important to commit your code in stages to show your progress whilst working on this assignment. Include clear comments and notes to explain key concepts in your code.
2. Host your Web Application, microservices and database to free hosting sites and provide the links on VLE/Moodle. When hosting online, make sure not to store any personal information but only test data.
3. Record a video demonstration of your assignment (not more than 20 minutes). In your demonstration, you should go through your application demonstrating each task and also provide an explanation of your code as explained above. Submit the link to your recording on VLE/Moodle. **Note**: Ensure that you set the correct permissions and share it only with your lecturer.

Marking Scheme

	Assessment Criteria	Marks Awarded	Task Marks
	Development		
SE2.3	Construct a solid and robust Microservices Architecture **[Task 1] – 3 marks** **[Task 3] – 3.5 marks** **[Task 5] – 3.5 marks**		**10**
AA3.3	Use emerging libraries to consume Web Services and to persist data on the Client. **[Task 8] – 4.5 marks** **[Task 9] – 2.5 marks**		**7**
SE3.4	Construct a solid and robust Event-driven Architecture. **[Task 6] – 5 marks** **[Task 7] – 5 marks**		**10**
KU4.2	Describe the main services provided by cloud services. **[Task 10] – 2.5 marks** **[Task 12] – 2.5 marks**		**5**
AA4.3	Use Cloud Web Services to create and deploy a Web Server. **[Task 11] – 7 marks**		**7**
AA4.4	Use third party Web Service APIs to consume data. **[Task 2] – 3.5 marks** **[Task 4] – 3.5 marks**		**7**

Demonstration			
KU2.1	Describe the context of a Microservices Architecture. **[Task 13] – 5 marks**		**5**
KU3.1	Describe the context and core principles of an Event-driven Architecture. **[Task 14] – 5 marks**		**5**
KU4.1	Describe the advantages and disadvantages of cloud computing. **[Task 15] – 5 marks**		**5**
Total			**61**

(End of Assignment Brief)