COLLEGE OF ARTS, TECHNOLOGY AND ENVIRONMENT
ACADEMIC YEAR 2023/24

Assessment Brief

Submission and feedback dates

Submission deadline: Part I: Before 14:00 on 13 November 2023

Part II: Before 14:00 on 8 January 2024

is not eligible for 48 hour late submission window

Marks and Feedback due on: Within 20 working days from the submission deadline

N.B. all times are 24-hour clock, current local time (at time of submission) in the UK

Submission details

Module title and code:	Advanced Software Development
Assessment type:	Portfolio
Assessment title:	Portfolio of two parts (part I: analysis and design; part II: Implementation, testing and agile approach)
Assessment weighting:	90% of total module mark

Size or length of assessment: Please see deliverables for specific details

TABLE OF CONTENTS

Module learning outcomes assessed by this task

1. *Analyse problems in order to identify software-solution approaches and requirements for computer-based software-intensive systems.*

2. *Compare and contrast software development methodologies and choose one suitable for a given application.*

3. *Design, implement, test and manage reasonably sized software system considering database and GUI components.*

4. *Develop the necessary transferable skills – e.g., communication, delegation, openness, decision making, flexibility and tolerance.*

5. *Discuss the need for security in the context of system development.*

Completing your assessment

1. What am I required to do on this assessment?

This is a **GROUP assignment** and to be completed in groups of **up to five students**. As a group you have been asked to analyse the following case study, perform requirement analysis, design, implement and test the above system. Please note responsibilities for each group member should be clearly defined as marks will be adjusted based on your individual as well as group contributions. Therefore, you must maintain evidence logs of your contributions to the project e.g., through regular Git commits.

More specifically:

- The assessment requires you to analyse, design and implement an object-oriented system based on the following case study.
- You will analyse requirements and produce detailed object models and designs from system requirements.
- Use the modelling concepts provided by Unified Modelling Language (UML).
- You'll have a question and answers session with your tutor(s) to explain your analysis and design as a group.
- Using your design, you will implement the software and perform software testing by applying a combination of manual and automated testing tools.
- You will write a group report on the software methodology applied on the project.
- Finally, you will demonstrate your system and answer questions about your design, software implementation and the use of your software development approach as a group.

Case Study: Horizon Restaurant Management System (HRMS)

The Horizon Restaurants (HR) is a successful chain of restaurants in Birmingham, Bristol, Cardiff, Glasgow, Manchester, Nottingham and London. Each city has at least two restaurants at different locations. HR is looking for an IT solution to manage their chain of restaurants and several internal processes. They expect to grow their business in future and open new branches in other cities. They want their IT solution to be flexible but integrated so that the HR directors should be able to get a holistic view of their restaurants and staff performances for future investments. Their expectation is that each restaurant branch has its local instance of HRMS that handles all transactions and data locally but the backend of the system is connected to all other restaurants in the HR chain and provide dedicated and secure Application Programming Interface (APIs) to extract data for analysis and reporting purposes.

HR has identified the following main components (see Figure 1**), but they would like the software development team (i.e., your group) to go through a rigorous requirements development process to ensure all requirements are properly handled at design and**

implementation stages of the software development lifecycle. These initial components are:

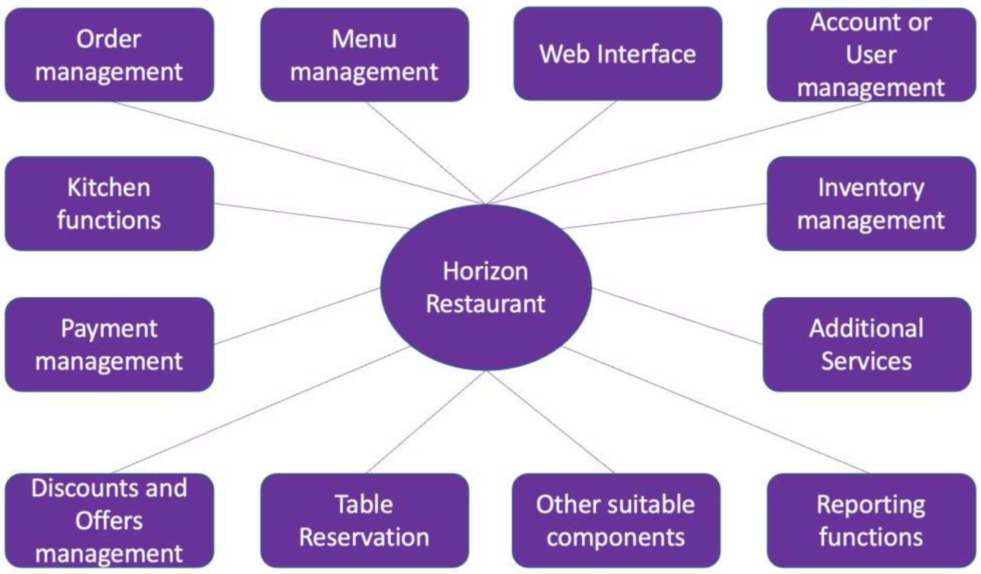

Figure 1: Horizon Restaurant Components

Component details are covered in Appendix – A.

More specifically there are following elements:

You are asked to produce the following deliverables for the case study given above.

<u>Part-I: Analysis and Design</u>

Element 1. As a group specify/validate requirements and produce a use case diagram to capture the functionality for the system to be built. Your use case diagram should be self-explanatory and you are not required to provide use case descriptions.

Element 2. As a group produce a class diagram to meet all the requirements captured in the use case diagram. You may add a brief explanation for your design decisions.

Element 3. As a group produce sequence diagrams (one from each group member) for the selected use cases. The sequence diagrams should provide meaningful behavioural information about the system.

<u>Part-II: Implementation and Testing</u>

Element 4. Write a joint group report (up to 700 words) outlining the steps that you followed to develop the system in relation to an Agile development episode and justify your choices. You should consider benefits and challenges faced and may consider the following steps:

- **Strategy planning:** This is the initial phase for our system development process where we as developers need to identify system users to avoid building wrong solution and identify role and responsibilities of each team member
- **Continuous team iterations:** The development phases of our project are more flexible as we continuously iterate between planning and implementation compare to conventional development methodology which is too rigid and strict
- **Team coordination and communication:** The essence of our project is effective coordination and communication among members either through face-to-face or online conversation
- **Simplicity:** Our main objective is to practice simplicity which is important for the system to avoid the structure being too complicated
- **Work Plan (Not mandatory but if they can produce would be good):** Project Backlog, SPRINT Cycle

Element 5. Develop the system using a suitable **Object Oriented Programming** language (e.g., Python as we'll use Python programming language for this module) and Database(s) considering all the functionalities described above.

- Implement the components you are responsible for and integrate with the components implemented by your group members.
- You should design and implement suitable database(s) and fill it with suitable mock data for testing and demonstration.
- You should be creative and come up with your own User Interface Design for different abovementioned GUI needs.
- You should also implement non-functional requirements (e.g., security measures) to a good standard.
- You should be able to demonstrate the functionality of the system during the group demo and question / answer sessions.

Element 6: Testing is typically a part of the program development. You should use a test strategy based on both manual and automated testing approaches to test your system thoroughly. You should identify all the suitable test cases for all the classes implemented. When you test your code, you should make sure that your program does not allow bad data to be stored into your objects. Deliberately feed in your program out of range or wrong data and try to make it fail, see if you can swipe bad values into the member variables. One example test case is shown below.

Test Case #	Description	Test dataset/Input	Expected output	Actual output
1	Booking staff is able to get price and book five tickets for a selected listing and date	Selecting film from listing on a specific future date and time, ticket type, and quantity	Shows seats availability and total price	As expected [Pass/Fail]

Submission

<u>You must use the Blackboard electronic submission system to submit your work.</u> Each student will have to upload complete package individually. Electronically submitted deliverables include:

For Part I (Analysis and Design): Each student needs to upload the following deliverable.

1. One PDF document that covers Element 1 to Element 3.

Do not forget to attend questions and answer session as a group.

For Part II (implementation and testing): Each student needs to upload the following deliverables in one .zip file. The naming standard of the zip file is WP1234567.zip where 1234567 is the student Id.

1. For Element 4, a reflective group report in PDF format.
2. For Element 5, the following deliverables should be compressed in a .zip file.
    a. A software system based on the problem specifications, with source code in ZIP file (using e.g., 7Zip). This should contain all the files and folders for the full working system. All program files must have student ID and student name who has written the code. You should also include list of additional Python packages (e.g., using pip list command) needed to run your software. <u>Please do not include python distribution folder in your ZIP file.</u>
    b. Provide a text file with instructions on how to start and use your software system e.g., any passwords etc.
    c. Provide relevant Database(s) (e.g., MySQL or MongoDB) dump.
3. For Element 6, evidence of testing e.g., test cases, unit tests and outputs. These can be included in a separate PDF document.

Do not forget to attend the question / answer and demo session as a group.

2. Where should I start?

- Group formation and protocols
    - You can start by talking to your class fellows to form a group. Then you can define roles and responsibilities for each student in the group.
    - It will help you to form group with students who are timetabled in the same practical session but this is not a requirement.
    - Agree a code of conduct for your group activities (see lecture 1 slides)
    - Plan your project and agree tasks, timeline for deliverables, communication protocol, meetings schedule, code repository, progress review

- Problem statement
    - Read the assessment task carefully, discuss with your group and if there is any ambiguity then clarify it by talking to your lab tutors. Your lab tutors will play role of representative of HR. If you have questions about this assignment or if you would like to clarify requirements, then please discuss with your lab tutors during your practical lab sessions or post on Discussion Board on Blackboard.
    - As there is a lot of potential for creativity, learning and applying new skills, <u>you must get your requirements analysis and specification validated by your lab tutors to determine the scope of the project for you and your group. This will also be your opportunity to discuss priority of the components if there are less than 5 students in a group.</u>
    - In general, keep it simple. Have a bigger picture in mind but complete one feature at a time.

3. What do I need to do to pass?

You would need to gain at least 40% marks to pass this assessment. Please refer to the marking criteria to see details of marks for both parts of this assessment.

4. How do I achieve high marks in this assessment?

To gain higher marks you would need to make sure you attempt all aspect of part I and part II. Further, the quality and the correct level of details covered in the completed work will result in higher marks. This will be possible by engaging in the module and group project activities. Do not forget to attend question and answer and demo sessions for both part 1 and part 2.

5. How does the learning and teaching relate to the assessment?

- Part I of the assessment is directly related to teaching sessions in week 2 (use case diagrams), week 3 (class diagrams), week 4 and week 5 (SOLID principles and design patterns).
- Building on the knowledge in first five weeks, the Part II of the assessment is further related to week 6 (Agile software development) to 12 for software implementation (graphical user interface, coding, database connectivity and testing).
- This indicates that engaging in the module will help you to learn and apply software development concepts on your group project.

6. What additional resources may help me complete this assessment?

- Your main source will be learning material on the Blackboard. You will be referred to topic specific external resources in lecture slides.
- In addition, UWE library study skills pages can be useful i.e., https://www.uwe.ac.uk/study/study-support/study-skills
- Your practical sessions will have dedicated slot to discuss your project progress and gain formative feedback
- Blackboard discussion board on module page can be used to post queries for your lab tutors or discuss specific topics with other students

7. What do I do if I am concerned about completing this assessment?

UWE Bristol offer a range of Assessment Support Options that you can explore through this link, and both Academic Support and Wellbeing Support are available.

For further information, please see the Academic Survival Guide.

8. How do I avoid an Assessment Offence on this module? [2]

Use the support above if you feel unable to submit your own work for this module.

The most common assessment offence in this assessment can be:

- Avoid copying or using code from other sources without acknowledging or complying to usage conditions
- Avoid using others work and claiming it as your own

Marks and Feedback

Your assessment will be marked according to the following marking criteria.

Part I submission will be followed by Question Answers during your timetabled practical session in the same week. You'll get formative verbal feedback and a mark for your Part I. This should help you to plan part II. You will get summative written feedback for Part II.

You can use these to evaluate your own work before you submit.

Part-I (45 marks)

	0.0-2.9	3.0-3.9	4.0-4.9	5.0-5.9	6.0-6.9	7.0-8.4	8.5-10.0	Mark & Advice for Improvement
Requirements and Use case diagram (10)	- Little or no requirement specified - Little or no diagram provided or diagram is mostly wrong (overall less than 30% complete/correct).	- Partial requirements but not validated - Partial diagram provided but actors and/or use-cases/relationships are only partially correct (overall less than 40% complete/correct).	- Partial requirements specified and validated - Partial diagram provided, actors and/or use-cases/relationships are correct (overall less than 50% complete/correct).	- Good number of requirements specified but partially validated -Almost complete diagram provided, actors and/or use-cases/relationships are only partially correct (overall less than 60% complete/correct).	- Good number of requirements specified and mostly validated - Almost complete diagram provided, actors and/or use-cases/relationships are correct. (overall, less than 70% complete/correct)	- Requirements specified and cover full scope of the system; Not all the requirements are validated. -Complete diagram provided, actors and/or use-cases/relationships are partially correct (overall less than 85% complete/correct)	- Requirements specified and cover full scope of the system; All requirements are validated. - Complete diagram provided, actors and/or use-cases/relationships are correct (100% complete/correct)	
Class diagram* (10)	Little or no diagram provided or diagram is mostly wrong (overall less than 30%	Partial diagram provided but Class naming convention, Class relationships, Attributes/methods of the classes	Partial diagram provided but Class naming convention, Class relationships, Attributes/methods of the classes	Almost complete diagram provided, Class naming convention, Class relationships, Attributes/methods of the classes	Almost complete diagram provided, Class naming convention, Class relationships, Attributes/methods of the classes	Complete diagram provided, Class naming convention, Class relationships, Attributes/method	Complete diagram provided, Class naming convention, Class relationships, Attributes/method	

	complete/correct).	and their visibility and type are only partially correct (overall, less than 40% complete/correct).	and their visibility and type are correct (overall, less than 50% complete/correct).	and their visibility and type are only partially correct (overall less than 60% complete/correct).	and their visibility and type are correct (overall less than 70% complete/correct).	s of the classes and their visibility and type are partially correct (overall less than 85% complete/correct)	s of the classes and their visibility and type are correct (100% complete/correct)	
Sequence diagrams (10)	Little or no diagram provided or diagrams are mostly wrong (overall less than 30% complete/correct).	Partial diagram(s) provided but Lifeline Notation, Activation Bars, and Message Arrows are only partially correct (overall less than 40% complete/correct).	Partial diagram(s) provided, Lifeline Notation, Activation Bars, and Message Arrows are correct (overall less than 50% complete/correct).	Almost complete diagram(s) provided but Lifeline Notation, Activation Bars, and Message Arrows are only partially correct (overall less than 60% complete/correct).	Almost complete diagram(s) provided, Lifeline Notation, Activation Bars, and Message Arrows are correct (overall less than 70% complete/correct).	Complete diagram(s) provided but Lifeline Notation, Activation Bars, and Message Arrows are only partially correct (overall less than 85% complete/correct).	Complete diagram(s) provided, Lifeline Notation, Activation Bars, and Message Arrows are correct (100% complete/correct).	
Individual Q&A (15)	Absent 0.0 Inadequate (barely able to explain the analysis/design and/or work done) 0.0-4.0 Good (good explanation of analysis/design and/or work done) 4.1-8.0 Very Good (very good explanation of analysis/design and/or work done) 8.1-12.0 Excellent (excellent explanation of analysis/design and/or work done) 12.1-15.							

* You may use other UML diagrams such as component or/and composite structure diagrams to complement and show the structure of system clearly.

Part-II (55 marks)								
	0.0-2.9	3.0-3.9	4.0-4.9	5.0-5.9	6.0-6.9	7.0-8.4	8.5-10.0	Mark & Advice for Improvement
Agile development report (10)	Little or no description provided (overall less than 30% complete/correct).	Partial description provided but texts/justification are not written reasonably accurately (overall less than 40% complete/correct).	Partial description provided, texts/justification are written reasonably accurately (overall less than 50% complete/correct).	Almost complete description provided but texts/justification are not written reasonably accurately (overall less than 60% complete/correct).	Almost complete description provided, texts/justification are written roughly accurately (overall less than 70% complete/correct).	Complete description provided, texts/justification are written reasonably accurately (overall less than 85% complete/correct).	Complete description provided, texts/justification are written very well and accurately (100% complete/correct).	
	0.0-6.0	6.1-9.0	9.1-12.0	12.1-15.0	15.1-18.0	18.1-20.0	21.1-25.0	Mark & Advice for Improvement
Implementation (25)	Little or no development and mapping to the class diagram; no persistent storage (overall less than 30% complete/correct).	Partial development and mapping to the class diagram, compiles and runs, but GUI displays input and output messages only partially correctly; persistent storage but with	Partial development, compiles and runs, GUI displays input and output messages only partially correctly; persistent storage but with update anomalies	Partial development and mapping to the class diagram, compiles and runs, but GUI displays input and output messages only partially correctly; persistent storage but with	Almost complete development and good mapping to the class diagram, compiles and runs, GUI displays input and output messages only partially correctly; persistent storage without update anomalies (overall less than 70% complete/correct).	Complete development including security measures and good mapping to the class diagram, compiles and runs, but GUI displays input and output messages only	Complete development including security measures, compiles and runs, GUI displays input and output messages correctly. One or more Databases are used for	

		update anomalies (overall less than 40% complete/correct).	(overall less than 50% complete/correct).	update anomalies (overall less than 60% complete/correct).		partially correctly; persistent storage without update anomalies (overall less than 85% complete/correct).	persistent storage without update anomalies. (100% complete/correct).	
	0.0-2.9	3.0-3.9	4.0-4.9	5.0-5.9	6.0-6.9	7.0-8.4	8.5-10.0	Mark & Advice for Improvement
Testing (10)	Little or no test cases/testing; no testing strategy (overall less than 30% tested).	Partially tested, no testing strategy (overall less than 40% tested).	Partially tested, no testing strategy (overall less than 50% tested).	Partially tested, very limited or basic testing strategy (overall less than 60% tested).	Almost tested, reasonable testing strategy (overall less than 70% tested).	Almost tested, good testing strategy (overall less than 85% tested).	Completely tested, sound testing strategy (overall less than 100% tested).	
Demo & Individual Q&A (10)	Absent 0.0 Inadequate (barely able to explain the analysis/design and/or work done) 0.0- 3.0 Good (good explanation of coding/testing and/or work done) 3.1- 6.0 Very Good (very good explanation of coding/testing and/or work done) 6.1- 8.0 Excellent (excellent explanation of coding/testing and/or work done) 8.1- 10.0							

For element 4 (agile development report) and in line with UWE Bristol's Assessment Content Limit Policy (formerly the Word Count Policy), word count includes all text, including (but not limited to): the main body of text (including headings), all citations (both in and out of brackets), text boxes, tables and graphs, figures and diagrams, quotes, lists.

UWE Bristol's UWE's Assessment Offences Policy requires that you submit work that is entirely your own and reflects your own learning, so it is important to:

- Ensure you reference all sources used, using the UWE Harvard system and the guidance available on UWE's Study Skills referencing pages.
- Avoid copying and pasting any work into this assessment, including your own previous assessments, work from other students or internet sources
- Develop your own style, arguments and wording, so avoid copying sources and changing individual words but keeping, essentially, the same sentences and/or structures from other sources
- Never give your work to others who may copy it
- If an individual assessment, develop your own work and preparation, and do not allow anyone to make amends on your work (including proof-readers, who may highlight issues but not edit the work) and

When submitting your work, you will be required to confirm that the work is your own, and text-matching software and other methods are routinely used to check submissions against other submissions to the university and internet sources. Details of what constitutes plagiarism and how to avoid it can be found on UWE's Study Skills pages about avoiding plagiarism.

If you fail this assessment then you will need to take resit. The resit assessment is also a portfolio of part I and part II, but some requirements will be updated and the assessment criteria will be slightly adapted. You do not need to attend resit QA or Demo session, rather you'll be asked to prepare and submit a demo recording.

Appendix – A: Component Details

Account or user management: system account creation, account update, account deletion, login, logout. More specifically following roles are identified:

- Admin role can perform all functionalities including management of accounts.
- Manager role can perform Create Read Update and Delete (CRUD) operations for discounts, reservations, event management and various reports.
- Manager and Chef roles can perform CRUD operations on food menu

- Staff role can perform CRUD operations such as make reservation, take order, deliver services, inventory tasks, customer service etc.
- Please consider other necessary roles based on the requirements analysis and validation activity.

Order management: This component involves:

check menu, select food item(s), place order, review/update order, cancel order, pack order, deliver order, etc.

Table reservation: This component handles advance table reservation and includes check capacity and reserve table for selected date(s). Managers should be able to make reservation at another branch of HR.

Discounts and offers management: This component manages discounts and promotional offers on orders. Managers should also be able to offer on the spot discounts on bills. All HR staff are eligible for 20% discount on their orders by presenting their staff ID.

Reporting functions: This component generates various useful reports that can help managers to perform analysis and make decisions. Reports should be generated by gathering few parameters from the user e.g., duration, type of report, selected branch(es) etc. For managers, these reports may include monthly expenditure, sales, profits, inventory stock, staff performance, average serving time, etc. For HR directors reports across multiple restaurants should be generated e.g., underperforming branches. Reports should be viewable on screen as well as formatted for printouts or file (e.g., PDF where filename should be name of the report and date it is generated) download / email. Title of each report and the duration of the report data should be clear in the heading and sub-heading part of the report.

Kitchen functions: This component handles kitchen functions by kitchen staff and serving staff. More specifically:

- View the orders as bulks/grouping for easiness of preparation
- Mark orders as ready and system should calculate the time taken from the order placement to serving
- Mark some items as ‘Not available’ if not available and there should be option to replace or update the items
- View orders in sequence they come

Menu management: This component handles setting up food menus. More specifically:

- View/Add/update/delete food category to/from the menu
- View/Add/update/delete food item(s) to/from the menu
- Update price for a given food item
- Update additional information (description, photo, allergens, etc.) for a given food item
- Suggest food items based on specific dietary needs

Inventory management: This component maintains the inventory of stock levels including food items, drinks, cutlery etc. More specifically:

- CRUD operations on stock levels
- Inform when stocks are in re-order level

Invoicing and payment management: This component performs CRUD operations on order placement, discounts/offers, process payment (emulate only) and payment cancellation.

Additional Services: There can be other services offered by the HR such as catering and servicing for business events, parties, weddings etc. Think creatively.

Web interface: In addition to desktop software version, HR would be interested in web-based interface to provide selected features for customers and other selected stakeholders.

Other suitable components: HR would welcome suggestions for any other features which can add value to HR operations. Think creatively.

You should also consider other suitable **non-functional requirements** (e.g., security measures) necessary for the HR booking system.

For the above case study, the following stakeholders are identified:

Chef, back-end or kitchen staff, front-end staff, delivery staff, Manager, Admin user, Customer, HR directors