| | | | |
|---|---|---|---|
| **COMP1610 (2020/21)** | **Programming Enterprise Components** | **Header ID** | **Contribution 100% of module** |
| **Module Leader Dr Mahtab Hossain** | | **25703** | **Deadline Date Friday 16/07/2021** |

This coursework should take an average student who is up-to-date with tutorial work approximately 50 hours

Feedback and grades are normally made available within 15 working days of the coursework deadline

**Learning Outcomes:**

On completing this module successfully, you will be able to:
1. Design, develop and deploy reliable and secure enterprise applications using a variety of Jakarta EE technologies.
2. Critically evaluate and compare enterprise features in Jakarta EE and determine their applicability in the creation of enterprise applications.
3. Demonstrate in-depth knowledge and understanding of techniques and technologies for the development of distributed systems, including the use of service-oriented architectures.

**Plagiarism is presenting somebody else's work as your own. It includes: copying information directly from the Web or books without referencing the material; submitting joint coursework as an individual effort; copying another student's coursework; stealing coursework from another student and submitting it as your own work. Suspected plagiarism will be investigated and if found to have occurred will be dealt with according to the procedures set down by the University. Please see your student handbook for further details of what is / isn't plagiarism.**

All material copied or amended from any source (e.g. internet, books) must be referenced correctly according to the reference style you are using.

Your work will be submitted for electronic plagiarism checking. Any attempt to bypass our plagiarism detection systems will be treated as a severe Assessment Offence.

**Coursework Submission Requirements**

- An electronic copy of your work for this coursework must be fully uploaded by **23:30 pm** on the Deadline Date of Friday **16/07/2021** using the link on the coursework Moodle page for COMP1610.
- For this coursework you must submit a single Acrobat PDF document.
  In general, any text in the document must not be an image (i.e., must not be scanned) and would normally be generated from other documents (e.g., MS Office using "Save As .. PDF").
- For this coursework you must also upload a single ZIP file containing supporting evidence (e.g., code). The database with data should be exported as a single SQL file, and submitted with the coursework as well.
- There are limits on the file size (current values are on Moodle).
- Make sure that any files you upload are virus-free and not protected by a password or corrupted otherwise they will be treated as null submissions.
- Your work will be marked online and comments on your work and a provisional grade will be available from the Coursework page on Moodle. A news item will be posted when the comments are available, and also when the grade is available in BannerWeb.
- You must NOT submit a paper copy of this coursework, or include the Banner header sheet.
- All coursework must be submitted as above. Under no circumstances can they be accepted by academic staff

The University website has details of the current Coursework Regulations, including details of penalties for late submission, procedures for Extenuating Circumstances, and penalties for Assessment Offences. See http://www2.gre.ac.uk/current-students/regs

# A Web-based Wallet system using Jakarta EE technologies.

This is an individual piece of coursework worth of 100%.

## Coursework Specification

An international money transfer company has approached you to build an application for them to manage Web-based wallet accounts for their customers. They emphasised on the reliability, scalability and easy extensibility aspects (if new features need to be added or existing ones might be modified) of the developed application.

The functional requirements of the system are outlined below in terms of operations [activities that the customer can perform on their accounts], and views [visualisation aspects]:

i) A customer can open account in three different currencies, e.g., GBP, USD and EUR. However, the developed application should keep provisions to add accounts for other currencies [e.g., INR, NGN, etc.] in future as well to be relevant in the face of expanding their business in other continents. In other words, the developed code should be easily extensible if needed to incorporate more account types.

Keep the attributes of the currency accounts as compact as possible. For example, an account identifier, account name, balance may be necessary attributes for each account. Please add additional attributes only if they are required for addressing the functionalities which are mentioned below. Follow similar principle for the customer attributes too (name, passport/id, address, phone number, email, and his/her login credentials might be necessary).

ii) A customer should be able to transfer or move his/her money from one currency account to another via the application besides the usual deposit, withdraw and view balance operations. You should also keep provisions to pay from a customer's account to an external account [payee 'wallet' account].

iii) A few views are primarily required: 1) a summarised view of number of accounts held by a single customer and the balance of each account in its own currency, and 2) A detailed view of his/her particular account with transactions for the current month,

iv) A user interface (UI) addressing all the above functionalities. It should be a Web application using Jakarta EE, which offers 1) customer log-in and 2) the execution of the above functionalities that are mentioned in i), ii) and iii) through the UI. The implementation should use entity classes and java beans for back-end functionality and JSP/JSF for the front-end (including HTML/CSS incorporation if it is required).

The whole coursework is divided into six tasks which you are required to carry out. The tasks are described in detail in the following. Please read and follow the instructions carefully. Task 2 should be attempted only after completing Task 1.

**Task 1: 35 marks**
A complete working system addressing the functionalities required thus far. Remember, there are many provisions to utilise object-oriented (OO) design and programming for this case study. Make sure you outline them in your report – see Section 3 of Task 6 for more information.

You are free to draw any valid assumption that are not in conflict with the system's functional requirements. For example, the customers may already be loaded inside the system/database "offline". There is no need for a registration system for customers in this coursework. You may initialize the data in your database using an SQL script (or equivalent) so that it has some initial data in order to fully demonstrate the functionality that you have implemented. You can use any relational database management system that you prefer (e.g.,

JavaDB, MySQL, etc.). Make sure the database is normalised with primary/foreign key constraints.

You can assume a customer only need a login ID and password to access the system that you have implemented. In other words, your application's first page or window should be a login screen requiring only the login ID and password. You SHOULD have a default pre-existing account created **[log ID: ha07, password: ha07]** which will be used by your tutor for login while marking.

**Task 2: 10 marks**
Suppose a new type of payment method (e.g., recurring monthly payment on a specific date) needs to be added on top of the existing ones (withdraw/deposit/transfer) as discussed in functionality (ii) [Page 3]. Also, achieve the task of adding one more currency account (INR) for customers.

Remember these tasks will only be attempted after you finish Task 1 – not at the same time. As a result, you will be able to identify and discuss the pros and cons of integrating this new type of payment method (and account type) to the already existing system which we are looking for here.

In your documentation (Section 2 of Task 6), please address the following under this task:
1) A list of files that you needed to change to incorporate
2) Screenshot of the code segments which were added or modified to achieve this.
3) A discussion of pros and cons of your followed approach. Reflect if there could have been any better approach. If yes, how?

**Task 3: 10 marks**
Incorporate ORM Frameworks using a Java Persistence API (e.g., Hibernate) inside your application. You can adopt this from the beginning in Task 1 (might be easier) or can adopt it even after completing the above tasks.

**Task 4: 20 marks**
Create a Web Service (SOAP or REST) which exposes this application's functionality to be consumed by other applications. The Web service will accept a date from a customer and his currency account detail and will return that day's number of transactions (+list of transactions) for that account.

Subsequently, create a small Java application [e.g., desktop client] that will retrieve the transactions of a given day from the server. For example, if a customer wants to retrieve the transactions of 1st January, 2021 of his/her particular currency account, the date (parameter) will be included inside the request together with the currency account details from this client. The reply from the server will consist of the number of transactions + the list of transactions on that day for that account. Keep your client application as simple as possible [any type of simple UI is accepted for this application – Graphical UI is not required but you are free to build one if you prefer]. Note that, you may also need to authorise this application with the server [customer's login credentials] to retrieve the transaction of his/her account on that day.

**Task 5: 10 marks**
The above functionalities are the core requirements of the application. Additionally, there is an opportunity if you want to demonstrate additional skillset. You can integrate any relevant additional features (i.e., functionalities) making use of Jakarta EE technologies for this task. Furthermore, if you can containerise the whole application using Jakarta EE, GitHub, Docker etc. (or even hosted from Cloud) to make it platform independent, that can enhance your development skills overall. This is just only an indication of the additional feature [not mandatory] – you are of course free to choose any feature here as long as they are Jakarta EE development related tools/techniques.

**Task 6: 15 marks**
Prepare a final report which should contain the following sections:
*Section 1:* A concise table containing a checklist of the tasks that you have been able to implement. Please refer to the task list d discussed above. An example table may look like:

| Task | Implementation Summary |
|---|---|
| 1 | Implemented. Only one view (summary of accounts) is addressed though. |
| 2 | Implemented but the new currency account (INR) type could not be integrated with the existing system. |
| 3 | Fully implemented. |
| 4 | Implemented but only the server side (no client application) |
| 5 | No additional feature/functionality was implemented. |

*Section 2:* Screenshots demonstrating each of the tasks that you have implemented. Create separate sub-sections for each task and give captions and/or annotations for the screenshots to explain which functionalities of the task are being demonstrated. Do not forget to address the specific documentation requirements mentioned for Task 2 in this part.

*Section 3*: Critically reflect (maximum: 500 words) on your experience of using the various design and coding techniques that you may have adopted in the context of building this system, especially:
1) If you have utilised any of the four pillars of OOP. Identify where and why [i.e., benefits]?
2) The relevant Jakarta EE technologies adopted and their relevance (or benefits) with respect to the coursework.

## Final Deliverables

You should upload your report as a single PDF file under the specified submission system. In addition, you must upload your code as a zip file, and an SQL file with the existing database details. In general, three files for your submission – ONE PDF report and ONE ZIP file for code and ONE SQL file for database details. More details are given below.

i) A PDF document submitted by the due date containing the following sections **IN THE ORDER** given below. Do not include any other information. Do not include all your source code inside the report.

A) A cover page.
B) A demonstration video link (maximum duration: 10 minutes).
C) Section 1 of Task 6 (please read Task 6 description above).
D) Section 2 of Task 6 (please read Task 6 description above).
E) Section 3 of Task 6 (please read Task 6 description above).
F) A reference list (if applicable).
G) README section in the end explaining how your prototype should be run with clear instructions (you can just copy the README file's content here which is a requirement for the ZIP file below).

ii) A ZIP file submitted containing the code satisfying the following requirements:

- There should be a README file to explain how the prototype should be run.
- An already created account as [**login ID: ha07, password: ha07**] for easy login to the developed system.

iii) A DEMO [recording] file: MP4 file or any common video extension files

- You also need to upload a brief recording [maximum duration: 10 minutes] showing the achieved functionalities. Make sure you demonstrate all your achieved tasks' functionalities as mentioned above.

## Grading Criteria

**Distinction (>= 70%)**
Well-developed work that satisfies the coursework requirements in design and implementation to a high standard. Able to demonstrate detailed critical understanding of the relevant concepts. A well written report and reflections on the tasks completed, and an excellent prototype.

**Merit (60 – 69%)**
Work that addresses the requirements in design and implementation reasonably well. Shows a good understanding of the relevant concepts with a good report and prototype.

**Pass (50 – 59%)**
Barely meets the coursework requirements in design and implementation. Demonstrates a limited understanding of the relevant concepts with a report that fails to address many of the design choices' rationale, and a prototype that may be missing many of the functional requirements.

**Fail (<50%)**
Demonstrating little, incorrect or no understanding of the relevant concepts. A superficial report that fails to address most of the requirements both in design and implementation perspective.

## Assessment Criteria

You should engage with **all** the tasks (Task 1 – 6) of the coursework.

The report will be assessed for the following and the mark will be adjusted accordingly:

- Task 1:
  - A working system using OO design principle, and programming. Addressed all the functional requirements as specified inside the description of Task 1 above.
    Integration of the required Java EE technologies mentioned inside Task 1 description (e.g., entity classes, java beans, JSP, etc.).
    Quality of the code.
- Task 2:
  - Successful integration of new type of payment method (recurring) and new currency account creation (INR).
  - The documentation of three different criteria discussed inside Task 2 description.
- Task 3:
  - Successful integration of JPA. Demonstration of ORM understanding.
    Quality of code.
- Task 4:
  - Successful integration of Web service.
    Both server and client-side implementation details are addressed.
- Task 5:
  - Demonstration of Jakarta EE technologies understanding as part of the integrated additional feature/functionality.
- Task 6:
  - Documentation of the all required tasks.
  - Overall critical reflection. If good OO principles have been followed? Comparative discussion among various approaches that might be undertaken, and the justification of your own adopted approach. The precision, correctness and depth of the discussion will be evaluated.