| | | | |
|---|---|---|---|
| **COMP1605 (2018/19)** | **Enterprise Patterns and Frameworks** | **Header ID** | **Contribution 100% of course** |
| **Course Leader Dr Mahtab Hossain** | **Release Date Monday 08/10/2018** | **300501** | **Deadline Date Monday 17/12/2018** |
| This coursework should take an average student who is up-to-date with tutorial work approximately 50 hours<br><br>Feedback and grades are normally made available within 15 working days of the coursework deadline | | | |
| **Learning Outcomes:**<br><br>On completing this course successfully you will be able to:<br>A. Design and implement an enterprise application using appropriate technologies, frameworks and design patterns.<br>B. Critically evaluate technologies, frameworks and design patterns in the design and implementation of enterprise applications.<br>C. Demonstrate an in-depth understanding of issues affecting the success of enterprise applications. | | | |

**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 **11:55 pm** on the Deadline Date of Monday **17/12/2018** using the link on the coursework Moodle page for COMP1605.
- 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 bank account management system using Object Oriented design and principle, and adopting design patterns and frameworks.

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

## Coursework Specification

A financial institution (bank) has approached you to build an application for them to manage bank 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 three different types of accounts, e.g., savings, current/checking, and ISA (individual savings account). However, the developed application should keep provisions to add other types of accounts [e.g., investment, child, etc.] in future as well to be relevant in the face of changing banking demography. In other words, the developed code should be easily extensible if needed to incorporate more account types.

Keep the attributes of the bank accounts as compact as possible. For example, bank sort code, branch name, IBAN, and account number are necessary attributes for each account. Please add additional attributes only if they are required for addressing the functionalities which are mentioned here. 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 account to another via the application besides the usual deposit, withdraw and view balance operations.

iii) You need to keep provisions for a customer to pay via multiple ways from his/her account. For example, direct transfer to another customer's account, direct debit, credit card, or even external [PayPal] payment method should be enabled from his/her checking/current accounts. Similar to (i), the developed code should be easily extensible if newer payment methods need to be added in future.

iv) A number of views are primarily required: 1) a summarised view of number of accounts for a particular customer and the balance of each account, and 2) A detailed view of a particular account with transactions for the current month,

v) A user interface (UI) addressing all the above functionalities. There is no restriction in terms of the environment – it can be a mobile application, a Web application or even a desktop application. You are free to choose any that you feel comfortable with. However, the UI needs to be simple facilitating easy-navigation in order to execute all the functionalities listed above.

The whole coursework is divided into five tasks which you are required to carry out. The tasks are described in detail in the following. Please read and follow the instructions carefully. All the tasks need to be addressed in chronological order, i.e., attempt Task 2 only after completing Task 1, etc.

**Task 1: 40 marks**
A complete working system addressing the functionalities required thus far. Remember, there are many provisions to utilise object oriented design and programming for this case study. Make sure you outline them in your deliverables – see "Deliverables" section 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 also no need for registering them or a registration system for this coursework. You may initialize the data in your database using an SQL script (or equivalent) so it has some initial data with a sufficient variety of information in order to fully demonstrate the functionality that you have implemented. You can use the Derby database or any other database that you see fit for your purpose. For any decision taken please justify your choice.

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 account created **[log ID: ha07, password: ha07]** which will be used by your tutor for login.

**Task 2: 10 marks**
Suppose a new type of payment method (e.g., standing order) needs to be added on top of the existing ones as discussed in (iii) [Page 3]. Remember this task 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 to the already existing system which we are looking for here.

In your report, you need to 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 manipulated 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**
Suppose a new type of operation (e.g., top-up mobile phone) needs to be added on top of the existing ones for the account as discussed in (ii) [Page 3]. Remember this task will only be attempted after you finish Task 2.

In your report, you need to 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 manipulated 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 4: 15 marks**
Suppose the bank came back to you again to add the provisions for joint accounts [another type of account] for customers. Also, they required a another view which should list all associated customers for a particular account which a customer can view from his/her application (e.g., in case it is a joint account, should list name and address of the other customers besides his/her own). Remember this task will only be attempted after you finish Task 3.

In your report, you need to 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 manipulated 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 5: 25 marks**
Critically reflect on your experience of using the various design and coding techniques that you may have adopted in the context of building this system, especially:
i) If you have utilised any of the four pillars of OOP. Identify where and why [i.e., benefits]?
ii) If you have utilised any of the design patterns [can be both from our classroom lectures or others]. Identify where and why [i.e., benefits].
iii) If you have utilised any of the frameworks [can be both from our classroom lectures or others]. Identify where and why [i.e., benefits]. Testing frameworks or continuous development tools can be mentioned here as well.

Remember, we are not looking for how many design patterns or frameworks that you can possibly adapt here. It is more important to understand the context, and their relevance of usage/adoption with respect to the problem domain, and your own reflection.

## Final Deliverables

You should upload your report as a PDF file. In addition, you must upload your code as a zip file (TWO files for your submission – ONE PDF report and ONE ZIP file for code). Details of both 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 of your source code.

A) A cover page.
B) A statement of functionality that outlines which parts of the tasks 1-5 are complete and which are not.
C) A description of any bugs in your program (all software has bugs!). Bugs declared in here will lose fewer marks than ones that you don't declare!
D) Class Diagram of Task 1. An ERD of database design.
E) Five different sections under the headings Task 1, Task 2.....Task 5 respectively, and address the requirements that have been listed under each task's description above.
F) Final Class Diagram [this one should address Task 2, 3, and 4 together, and is expected be a bit different from Class Diagram of (D)].
G) A reference list.

ii) A ZIP file submitted by the due date containing the following structure:

- Four folders named Task1, Task2, Task3 and Task4 executables with code, respectively. In each folder, there should be a README file to explain how they should be run (clear instructions!).
- An already created account as [**login ID: ha07, password: ha07**] for easy login to the developed system.
- The database script i.e. the SQL commands that you have used to create the tables in your database and initialise the data. This should be provided as a SINGLE SQL script file. Please ensure the following details for your database:
  - **Database name: ha07, Username: ha07, Password: ha07**

An acceptance testing of the prototype will be scheduled during the 13th week of the term to provide formative feedback. A demonstration may be scheduled after your final upload if the tutor has difficulty running the uploaded prototype.

## 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 – 5) of the coursework. Failing to do so may result in ***possible failure*** on the whole coursework.

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

- Task 1:
  - System Requirements: A working system using OO design principle, and programming. Addressed all the functional requirements as specified inside the description of Task 1 above. Quality of the code.
  - Design Documentation: ERD, Class Diagram.
- Task 2:
  - System Requirements: Successful integration of new type of payment method (standing order).
  - The fulfillment of three different criteria discussed inside Task 2.
- Task 3:
  - System Requirements: Successful integration of new type of operations [mobile phone top-up] involving the accounts.
  - The fulfillment of three different criteria discussed inside Task 3.
- Task 4:
  - System Requirements: Successful integration of joint account creation provision, and also a new 'view' adoption,
  - The fulfillment of three different criteria discussed inside Task 4.
- Task 5:
  - The overall system's class diagram after incorporating Task 2, 3 and 4.
  - Justification of the design pattern and frameworks used [if any].
  - 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.