|  |  |  |
| --- | --- | --- |
| **Software Implementation Checklist** | **Faculty of Computing, Engineering and the Built Environment** | New Logo Tiny |

|  |  |
| --- | --- |
| Please fill in your name and student ID in the table below. | |
| **Student Name** |  |
| **Student Number** |  |
| **Course and Year** | 2017/2018 |
| **Module Code** | CMP4266 |
| **Module Title** | Computer Programming |
| **Module Leader** | Dr. Shadi Basurra |
| **Assessment items:** | 1. Software implementation checklist (Compulsory). 2. Design documentation: 500 words paper inclusive of diagrams submitted online 3. All of your source code and associated project files. 4. 1000-word testing and evaluation documentation. |

|  |  |
| --- | --- |
| Students are required to complete this software implementation checklist for their Bank System developed in Python. You need to select only the features that were implemented in your code. You can select all/some features from any marking range as long as they have been implemented in the code submitted for this assessment. For example, you can select all/some features from the marking range 40%-49% and all/some features under the range 70%-79% etc. This implementation checklist should be submitted alongside the design documentation, implementation source code of the system and the testing and evaluation documentation.  **Important notice**: This checklist will assist the lecturer when marking your code, hence, you should only select the feature requirements that have been implemented in your code. Even if some features are not working correctly, you can still select them as long as there is evidence in your code showing the implementation attempt. However, it is not acceptable for a student to claim the implementation of features that were not attempted/implemented in the game. False claims are clear indications that the student does not understand the submitted code, hence, the submission will be investigated further for plagiarism, and the lecturer marking the assessment may invite the student to explain all/parts of the submitted code. | |
| **GUI program development and user manual** | |
| **Achieving a mark of 40% to maximum of 50%** | |
| The game **must** implement **all** the following: | |
| **Create the necessary classes and functions which allow Customers to perform the following tasks:** | |
| * Customer Login |  |
| * Deposit money |  |
| * Withdraw money. |  |
| * Check bank balance. |  |
| * View customer details e.g. name and address. |  |
| * Update customer information e.g. name and address. |  |
| **Create the necessary classes and functions which allow admins to perform the following tasks:** | |
| * Admin login |  |
| * Search for a particular customer to perform various banking operations on his/her account i.e. check balance, deposit/withdraw money etc. |  |
| * Close account i.e. remove customer from the system |  |
| * Update customer information i.e. name and address |  |
| * Update admin own information i.e. name and address |  |
| * Print a customer details |  |
| * Print all customers details |  |
| **Achieving a mark of 51% to maximum of 60%** | |
| The game **must** implement **all** the above and the following: | |
| * Development of a suitable GUI to perform all the above banking functions |  |
| **Achieving a mark of 61% to maximum of 70%** | |
| The game **must** implement **all** the above and the following: | |
| * Customers can have different types of bank accounts. Accounts will differ in their name, interest rate and overdraft limit etc. |  |
| * The bank system should be able to store and load customer and admin data from and into a file. |  |
| * Save the status of the system to the backend storage (e.g. file storage) at any time. Also when the system is closed it should update its backend storage, such that the system can load the up to date status at the next start. |  |
| * Customers should be able to request a loan, if the loan is within a the bank loan limit e.g. £10,000, then the system will grant the loan with 30% probability. If the bank refuses to grant the load, a message should be given to the customer to notify them about the decision. However, when the loan is accepted, the requested amount will be deposited to the customer’s account. Also, a loan payment due data and time should be generated e.g. 21 days from now. Customers should be able to clear their loan at any time, however if the repayment due date has passed, a daily fee should be added to the account. Hint, Random and time functions can be used to implement this feature. |  |
| * Development of a suitable GUI to perform all the above banking functions. |  |
| **Achieving a mark of 71% and over** | |
| The game **must** implement **all** the above and the following: | |
| * Admin should be able to view all customers’ loan report. A loan report should include a list with all customer names, loan amount and loan due data. Also, Admins should be able to view the report based various orders e.g. loan amount and repayment due date and time. |  |
| * Transferring funds/money from one account to another. For example: one customer wants to send money to another customer. |  |
| * 30% of the loan requests may require further approval by admins. Hence, 60% from all loan request refusals should be further processed by admins for approval. Admins at this stage have all the right to accept / reject a loan request. |  |
| * Development of a suitable GUI to perform all the above banking functions. |  |