University of the West of England
BRISTOL

MODULAR PROGRAMME

COURSEWORK ASSESSMENT SPECIFICATION

Module Details

Module Code UFCF85-30-3	**Run** 18June/18	**Module Title** Enterprise Systems Development
Module Leader Dr Janith Chandrakantha	**Module Coordinator**	
Component and Element Number B:CW1		**Weighting:** (% of the Module's assessment) 40
Element Description GROUP-BASED DEMO (Group-based demonstration of software development)		**Total Assignment time** **4 Weeks**

Dates

Date Issued to Students: 17/06/2018	
Submission Place: Blackboard (as a complete zipped NetBeans project) Plus: In-class demonstration	**Submission Date** 17/07/2018
	Submission Time 2.00 pm

Deliverables

1. Zipped Netbeans project
2. In-class demonstration

Module Leader Signature

Dr Janith Chadnrakantha

Enterprise Software Development (UFCF85-30-3)

Group Coursework Assignment

Description

You (as a group) are asked to design and implement a software system using the features and functionality of the examples from the course book, the practical classes, had demonstrated in the lecture or been referred to in the scope of the course.

You should design and build your system using JSP/Servlets using MVC on a Java EE such as Glassfish server - with a local Java DB Relational database backend. You will be required to demonstrate and discuss your working system using Netbeans IDE.

You must use the best practice techniques as demonstrated or suggested in the lectures, practical/tutorial classes or given on Blackboard.

Your group will be treated as a unit and any individual member may be required to demonstrate complete knowledge of the system you are presenting.

Assessment System

You are required to submit your work in the Blackboard VLE as a zipped NetBeans project.

Demonstrations will take place in scheduled practical classes at the end of semester one.

- You must download (from Blackboard) and unzip your project.
- You must run the SQL scripts provided to create and populate the required tables.

Your NetBeans project must be runnable on the normal FET lab machines (or your own laptop which you must bring).

- It is your responsibility to attend scheduled classes – ***failure to demonstrate your system in class will be treated as a non-submission.***
- All group members will be awarded the same mark -
  - However, any group member **failing to take part** in the demonstration will be assessed **as a non-submission** and given zero marks.

The quality of your verbal expression in this demonstration is important – incoherent explanations will not achieve high marks. Please be advised that the demonstration lasts for a fixed duration of 20 minutes per group, so be prepared to concisely demonstrate and explain your system.

Specifications

XYZ Drivers Association decides to set up a solidarity fund to subsidise members for minor accidents so that the members can avoid making claims to Insurance companies. The association provides a certain amount of subsidy once a claim is made, and sums up all claims made to the end of each year. The members are expected to pay annual membership fee and the allocated portion of the total annual charge from lumpsum of the claims. The charges are allocated annually. Members will be able to make claims after 6 months of their membership, be allowed to make maximum 2 claims per year, and will not be supported if these circumstances are not met.

The Association aims to own a web application (**WebApp**) to streamline its business processes and help for a smooth administration. The **WebApp** will let a member log on (register first if required), pay fees and outstanding balances, and be able to make a claim. Registrations for membership require personal information including name, address, date of birth, and date of registration of the applicants. (A **web service** may be used for address lookup purposes). Once this information is submitted, the system should record it and confirm a provisional membership returning a user name and automatically generated password as well as charging him/her with annual membership fee. Once the membership fee is paid and been confirmed by the office (admin), the person is upgraded to full membership level. Members are suspended if they did not pay annual fee and charges, and will be allowed to resume their membership once they cleared outstanding balances.

An administrator is expected to process outstanding operations raised by the full members including confirmation of the payments, charge the members based on annual lumpsum and membership fee, and assess the eligibility of the members for the outstanding claims. They require having some facilities to browse through the list of members, search for members based on provided particular information.

The Web Application is expected to meet the following requirements:

1) At least the following pages are expected to be included:

    a) The main (home) page letting users select the type of user and action ahead

    b) Login/registration page for member users is required

    c) A Dashboard page for member users

    d) A management Dashboard page for admin users that lets to process the operations as required

2) A user should be able to navigate through the pages, smoothly, and especially be able to access to its own dashboard and the home page from any page.

3) A web service should be developed for a particular business process ***(e. g. the process for assessing if the members are eligible for support once they make claims)***, deployed and made up-running on a server (e.g. GlassFish) as a separate project, and then be invoked/integrated in the system as a service.

4) The groups are expected to use GitHub as the version control software to ease collaboration within the groups and to create evidence of their team work.

5) The whole system should be using

    (i) Java EE components following MVC patterns,
    (ii) interacting with database (Java DB / MYSQL), and
    (iii) deployed on a server, which has a container (GlassFish).

Collection of test cases

During the development you will be supplied with a sequence of test cases, which will be selected from the following functions:

1. **Login** as typical member user (e.g. user = “member1”, password = ”member1”) and create a session lasting for 20 mins. If not a user yet, **register** as an applicant for membership
2. See member’s **dashboard**, which consists of :-
    a. Check for **outstanding balance**
    b. Make a **payment**
    c. Submit a **claim**
    d. **List all** claims and payments to date
3. Navigate back-to main page to let **change user** (e.g. change to admin)
4. See **Admin’s** (management) **Dashboard** consisting of:-
    a. **List** all **members**
    b. List all outstanding **balances**
    c. List all **claims**
    d. List all **provisional member** applications
    e. **Process** individual **claims**
    f. Process membership applications and **upgrade** if payment is made
    g. **Suspend/Resume** membership
    h. Report **annual turnover** including total income and total pay-outs

Additionally - it can be useful for the system to use "**Filtering**" for authentication purposes, especially for **authenticating admin user** - a clear example of this would also be favourably considered. (GO through JAVA EE Security, Authentication)

You may assume the system will accept cookies. No threading/concurrency considerations are required. All data must be stored and retrieved from Java DB/MYSQL database.

You will be supplied with one SQL script to build tables containing some samples of members, claims and payments. (**XYZ_Assoc.sql**).

For the Demonstration

<u>Each group member should be prepared to perform any of these tasks or explanations.</u>

You will show that your database contains the users, members, claims and payments details, which will be provided to you well before the due date.

You will be asked to step through the test cases in the order provided.

You will discuss design and development considerations - for example by producing and explaining one or more "Interaction Diagrams" (e.g. Chapter 3, page 88 of course book).

You will be asked to show and explain your code - which must be readable and commented.

Demonstration and Marking

Part 1 The Test Cases.

- These must be demonstrated sequentially and as described.
    - Marks could be lost – e.g. for a lack of clarity or failure to demonstrate tests sequentially.
- Each successful test case is worth 5 marks (total 40).
    - Normally a test will pass or fail
    - Do not combine the Test Cases - each must be a distinct operation.

You must run the SQL script supplied

Ambiguity

If you think there may be some ambiguity in the specification or Test Cases you must check for clarification with the tutor in your normal practical class. False assumptions may cost you marks.

Part 2 Design Talk through.

You should briefly and concisely explain the design and implementation of the following features of your system (50 marks)

- The MVC pattern and project file structure (15 marks)
- The relational database (10 marks)
- Sessions (5 marks)
- The DD (web.xml) (5 marks)
- Web Services (develop and use) (10 marks)
- Using Filters (5 marks)

Part 3 Team work evidence.

Your team work will be assessed based on the evidence you provide via GitHub version control software. (10 marks)

- Evidence of each group member’s contribution (3 marks)
- Task management and allocation (3 marks)
- Clear history of collaboration (4 marks)