FIT5042 Enterprise Application Development for the Web
Semester 2, 2017
C/D/HD Code Specification

A Banking Enterprise Application

- This is an individual assignment. Discussion among students is encouraged but working in groups is NOT acceptable.
- Credit Feedback Deadline: Monday week 7
- Distinction Feedback Deadline: Monday week 9
- High Distinction Feedback Deadline: Monday week 11
- Grading Due date: Week 12 Friday at midnight.
- Late penalty (for grading deadline): 20% of possible marks per day, excluding weekend and public holiday.

1. Overview

You are to build a web-based Java prototype of a Banking Enterprise Application. You are required to produce a system with Java enterprise technologies, such as JSF, RESTful Web services, EJB, Persistence API, application client and etc.

This document is intended to be open-ended and specifies only the minimal requirements for the Tasks. If you want to implement other business rules or features that are not specified here, you may do so, as long as you document your assumptions and ensure that the assumptions made are reasonable. Consult your tutor or lecturer if you are not certain whether the assumptions you made are reasonable or not.

Note that the entire document describes the requirements of the complete system. You do NOT have to meet all the requirements to pass the unit. Please refer to section 2 for the different requirements to reach the different assessment levels.

2. Assessment Guide

Appropriate and sufficient data validations and corresponding error messages are to be implemented. The system should also ensure that the user is not allowed to violate the integrity of the existing data in the system.

For every submission, all source code must conform to the Java coding standard and must be well commented.

You must complete all the features and technical requirements on one level before moving up to the next. No marks will be awarded for the higher-grade level's functionality if you have not completed ALL of the functionalities and technical requirements in the successive lower-grade levels.

** The design of your implementation must use the Object-Oriented programming paradigm and must be flexible and easy to maintain.

Credit Level

Features

- Two or more tables (entities) appropriately linked (with foreign keys)
- A member of Public can search for a Banking transaction by its Type, Banking transaction Name, Banking transaction No.
- Search results are displayed in tabular format with heading. Only Banking transaction Name and Banking transaction No will be shown.
- Has the option to view the full details of a Banking transaction from the search result.

Technical Requirements

- The system must provide a web client (An application client is only required for HD level).
- Web clients must be implemented in JSF.
- Some of the user input fields should have validation.
- The persistence of the data is managed via Persistence API.
- Make use of either application managed entity manager or container managed entity manager.
- Retrieval of data must be done using JPQL.

Distinction Level

Features

- All features specified in the Credit Level.
- Bank Workers can View/Add/Update/Delete Users (either public or banking workers).
- Members of Public can use Banking transactions using the system, e.g transferring funds between accounts (When finished using the Banking transaction this information must be stored).

Technical Requirements

- Web client must be implemented in JSF. (An application client is only required for HD level)
- All user input must have some form of input validation (a variety of different validations should be used)
- The persistence of the data is managed via Persistence API.
- Make use of either application managed entity manager or container managed entity manager.
- The interaction between clients and database must be handled by EJBs.
- Make sure BOTH Criteria API and JPQL are used for data retrieval (in different examples, no need to use both in the same example).

High Distinction Level

Features

- All features specified in the Distinction Level.
- Bank Workers and members of Public are required to login using a username and password to access the system.
- Bank Workers can search members of Public by a *combination* of ID, last name, first name, type and email.
- When adding a Banking transaction type to the system, the information of the Banking transaction can be obtained via a web service instead of being entered manually. [You can create a simple web service to do this]. (Or implement your own web service design and consume it)
- Make use of some features from javaScript frameworks/ ajax etc
- An application client of some of the application features must be implemented

Technical Requirements

- Web client must be implemented in JSF.
- The persistence of the data is managed via Persistence API.
- The interaction between clients and database must be handled by EJBs.
- Make sure both Criteria API and JPQL are used for data retrieval.

- Ability of mapping inheritance to database must be demonstrated.
- In addition to other forms of validation Bean validations must be used to validate data.
- Consumption of web service must be conducted in EJBs (in addition to any other methods used).
- Application should be secured using JAAS API.
- Application clients must be implemented using GUI with swing library. The GUI can be manually created by coding, but can also be auto-generated by IDE tools.

3. Business Entities

This section outlines some entities that are required by the system. Depending on your design, you might, and likely will, need more business entities to facilitate the operations of your solution.

3.1. Banking System User

There are two types of Banking System users: Bank Workers and Public. Each user has the following information stored in the database:

- ID number
- Last name
- First name
- Email
- Password
- Type (Bank Worker, Public)
- Address
- Phone number

On top of the information above, a member of Public also has the information about their Banking accounts/transactions stored in the database. Similarly each Bank Worker has the information about the activity they have made stored in the database. Refers to section 3.2 for details regarding the interaction.

Business rules for the member of Public user:

- Each user is associated with a unique ID number in the system
- The last name and first name must not contain numeric value
- An email address must be in a valid email format (e.g. christopher.messom@monash.edu).
- A user must be in one of the two types: Bank Worker or Public
- A phone number must be valid (i.e. if the phone number is a landline number, the first digit of the phone number must be 9 and the number must be 8 digits long. On the other hand, if the number is a mobile number, the first digit of the phone number must be 0 and the number must be 10 digits long)

3.2. Banking transaction

The Banking System maintains a list of Banking transactions types, which can be used by the Public and viewed/ edited by the Bank Worker. Each Banking transaction has the following information stored in the database:

- Banking transaction No(A unique identifier of the Banking transaction)
- Banking transaction name
- Type (When there are different types of Banking transaction available)
- Description

Business rules for the Banking transaction:

- A Banking transaction must be in one of the Banking transaction types (can't be null).
- Each Banking transaction is associated with a Banking transaction No
- A member of Public can have unlimited Bank transactions, and a Banking transaction ID number must be associated to one and only one member of Public.

4. System Functionalities

4.1. Member of Public

The system permits a member of Public to:

- Search for a Banking transaction by the following combination of criteria:
  - Banking transaction No
  - Banking transaction name
  - Type
  - Description
- View the details of a Banking transaction item selected;
- View his/her own personal details.
- Search results are first displayed in tabular format with a heading. Only Banking transaction No, Banking transaction name will be shown.
- The system should have the option to view the full details of the Banking transaction (including all details) from the search result.

4.2. Bank Worker

The system permits to do everything a member of Public can plus:

- Search for a member of Public in the system by the following combination of criteria:
  - ID Number
  - Last Name
  - First Name
  - Email

- Search results are first displayed in tabular format with a heading. Only the ID number, last name, first name and email will be shown.
- View the personal details (including the information about any previous and existing used Banking transactions(s)) of any particular member of Public selected.
- Add an item to the Banking transactions type, delete an item from the Banking transaction type and update the details of an item in the Banking transaction type;
- Add a user to the system, delete a user from the system and update a user's details

5. Design of the Website

The overall web site should have a consistent look and feel. Within each page of the web site there must be a navigation bar that links to other pages.

6. Technology

The prototype described above must be implemented using Java EE technologies (e.g. EJB, JSF, RESTful Web Banking transactions and Persistence API). A fully functional Java EE system must be developed for the assignment. You must follow the details below to set up the development environment for your solution.

Development Environment:

- JDK 1.8
- Java EE 7
- Netbeans 8.2 or above
- GlassFish 4.0+
- JavaDB that comes with the Java EE Development Kit

7. Assignment Deliverables

As well as the Doubtfire feedback uploads, you must submit a zip file consisting of all the source code for your work including:

- Source code
- Script for generating tables / test data in database
- Instructions for running the application (including the name, username and password of the database used)
- A completed individual assignment cover sheet

All the files above should be uploaded to Moodle as a zip file and use the following naming convention: FIT5042-StudentID.zip. For instance, if your student id number is 12345678, then the file you submit should be named **FIT5042-12345678.zip**.

You must submit your work by the submission deadline on the due date (a late penalty of 20% per day, not including weekends and public holidays, of the possible marks will apply - up to a maximum of 100%).

For Distinction and High Distinction Grades you will be interviewed during week 12 or 13. You can also expect to be asked to explain your design, discuss design decisions and alternatives, and modify your code as required.

It is your responsibility to make an appointment with your tutor for the interview. The interview appointments will be arranged in the tutorial classes in Week 12. You must arrange your appointment with your tutor in person if you are an on-campus student or via skype (or similar) if you are an off-campus student. No appointment will be arranged via other channels.

8. Plagiarism

Before submitting your assignment, please make sure that you have not breached the University's plagiarism and cheating policy. It is the student's responsibility to familiarize themselves with the contents of these documents.

Plagiarism and cheating are regarded as very serious offences. In cases where cheating has been confirmed, students would be severely penalized, from losing all marks for an assignment, to facing disciplinary action at the Faculty level. While we would wish that all our students adhere to sound ethical conduct and honesty, I will ask you to acquaint yourself with the following policy from the Plagiarism Procedures of Monash, available at

http://www.policy.monash.edu/policy-bank/academic/education/conduct/student-academic-integrity-managing-plagiarism-collusion-procedures.html