Assignment 1

Introduction

Working on your own, design and develop a web application. Using JavaServer Faces and JDBC, develop a website with a range of functionality that includes secure login and the ability to manage data. Demonstrate your completed application in a lab session.

Final deadline: Your lab session on Monday 12 September 2015 (Week 7). This occurs during StuVac due to a public holiday later in the semester.

Weight: 40% of your final grade

Assessment Type: Individual project

Learning Objectives: This activity assesses objectives 1, 2, 3, 6 and 7 from the subject outline.

Background

Some of the most successful websites on the internet are simple ideas that have been well executed. At the core of these websites is the ability to create, read, update and delete (CRUD) data.

In this assignment, you will create your own web application that uses Java EE technologies to implement CRUD functionality.

Task

For this assignment, you will create a simple web application that incorporates the following minimum functionality:

- Login
- Create a record
- View or search for records
- Edit a record
- Delete a record
- Logout

The records can be anything you wish. You will need to decide for yourself what purpose your website will serve:

- You might consider building an application to help a loved family member (*e.g.*, a Wedding Invitation Tracker).
- You might consider building an application for your work (*e.g.*, a Safety Incident Register).
- You could build an extremely low-fidelity imitation of a popular website (*e.g.*, create your own Tinder or Reddit).

You will not be assessed on the quality of your idea. The idea does not have to be 'good' so long as you implement it well (*i.e.*, using patterns and principles covered in this subject).

Please note:

- You must **NOT** create an application that is sexist, pornographic, vulgar, hateful or otherwise inappropriate in a professional setting.
- You must **NOT** create an 'Address Book' or 'To do list' application.

Functional Requirements

Core Functionality

Your web application will need to have login functionality as well as the ability to create, read, update and delete ("CRUD") records in a simple database schema.

In other words, your web application must have the following core functionality:

1. Log in (for the core functionality, you can have built-in user accounts)
2. Create a record (e.g., `INSERT INTO ...`)

3. List records (e.g., `SELECT ... WHERE ...`)
4. View a record (e.g., `SELECT ... WHERE id = ?`)
5. Update a record (e.g., `UPDATE...`)
6. Delete (or hide) a record (e.g., `DELETE ...` or `UPDATE ... SET visible = 'N' WHERE id = ?`)
7. Log out
8. Your application **must** provide useful error messages if the user enters invalid input

Innovation Functionality

In addition to the core functionality, 10 marks are devoted to innovative use of Java EE technologies. To get 10 marks, you would be expected to choose to implement three of the following features.

1. Ability to create new accounts
2. Upload and viewing of photos or videos
3. Ability for users to export their data in XML or CSV format
4. Polished, beautiful user interface
5. Use of a multiple-step 'wizard'-style user interface (*e.g.*, a multi-step account creation or a multi-step credit-card data entry)
6. The ability to log in and create, read, update and delete **more than** one type of record
7. Responsive functionality using AJAX or Websockets

You are not limited to this list: please feel free to negotiate alternative functionality of comparable difficulty with your tutor.

Functionality Proposals

You must propose your application to your tutor during the first two weeks of semester.

Your tutor will provide the following template for you to complete:

> ***(Application Name)***:
> My application will help *(user)* to *(need)* by keeping track of *(record)*.
> My application will allow users to login and then create, read, update and delete *(records)*.
> In addition, my application will *(provide innovation functionality)*.

Consider the following three examples of how to use the above template:

1.

> ***UTS Accident Register:***

My application will help *UTS staff* to *improve safety on campus* by keeping track of *accidents that have occurred*.
My application will allow users to login and then create, read, update and delete *accident reports*.
In addition, my application will *offer a step-by-step accident report creation wizard (5), allow users to upload photos (2) and have a beautiful user interface (4)*.

2. ***My Tinder:***
My application will help *single people* to *find love* by keeping track of *matches*.
My application will allow users login and then create, read, update and delete *profile matches*.
In addition, my application will *allow users to create new accounts (1), post messages to each other (6) and have responsive live notifications of matches (7)*.

3. ***Ben's Cookbook:***
My application will help *Ben* to *cook* by keeping track of *recipes*.
My application will allow users login and then create, read, update and delete *recipes*.
In addition, my application will *allow Ben to upload photos (1), have friends comment on recipes (6) and allow export into Excel using CSV (3)*.

Non-Functional Requirements

1. You must use JavaServer Faces for user interface development. You are permitted to use other technologies such as Servlets and JSP as appropriate. However, the core functional requirements must be implemented using JavaServer Faces.
2. You must store data in a JavaDB database and use JDBC to access the database.
3. You must use the Java EE authentication facilities built into Glassfish or an established security framework (do not write your own security code).
4. You must use git to track your changes and you must regularly push your commits to a private repository shared with your tutor on BitBucket.
5. Your application must run on GlassFish 4.1.1.

Bitbucket provides free private Git hosting. Please sign up to bitbucket with your student email address. This will give you a free academic account. Ensure that any repositories that you create are private and invite user `benatuts` and user `ryanuts` to your repository with write access (i.e., the tutors). http://www.bitbucket.org/

Deliverables

You may demonstrate your assignment in any lab session between weeks 1 and 7 (inclusive).

You will receive a one mark bonus for submitting in Week 6. You will receive a two mark bonus for submitting on or before Week 5.

You can demonstrate this assignment on lab computers or on your own laptop. You may develop using any operating system (including Windows or Mac OS) and any Java EE application server that is compatible with Glassfish 4.1.1.

During the demonstration, you will need to submit your project files as a single ZIP to UTS Online. Your submission will be subject to checks for plagiarism and may also be reviewed by other students in the subject as part of a weekly laboratory exercise.

The ZIP file should contain:

1. Your complete source code (e.g., your NetBeans project folder)
2. SQL statements to create the application's database
3. Any other files required to run your application

Suggested Timeline

- Week 1: Choose an idea and propose to your tutor (3 hours)
- Week 2: Sketch draft screen designs and code in HTML (3 hours)
- Week 3: Create a project and set up a source code repository (3 hours)
- Week 4: Implement user interface logic using JavaServer Faces (6 hours)
- Week 5: Create database schema and data access objects (6 hours)
- Week 6: Implement innovation functionality (9 hours)
- Week 7: Present assignment

You should present your screen and database designs during (or prior) to Week 4 to receive feedback on your progress.

Assuming that you have completed the relevant tutorial exercises, the full functionality of this assignment is expected to take approximately 30 hours to complete.

Feedback

Your assignment will be marked during the lab session. You will run the project for your tutor and he will test your system and examine the source code. You will have an opportunity to discuss your assignment and mark with your tutor.

Groups for Assignment 2 will be allocated based on the order that students complete Assignment 1. The intent of this strategy is to ensure that you are put in a group with similarly motivated students. However, your tutor may use discretion to ensure that this results in an equitable outcome.

Group Work and Misconduct

This is an individual assignment. You are encouraged to discuss your idea with your peers. However, all other work that you submit as part of this assignment must be entirely your own and any assistance properly identified and acknowledged. You are permitted to make use of libraries and other resources found online but you must *clearly* identify the source.

If you are in doubt, please ask your tutor. Please refer to the faculty's handbook for more information about Student Misconduct: https://my.feit.uts.edu.au/modules/myfeit/downloads/StudentGuide_Online.pdf

Grading

Please consider the marking criteria carefully and allocate your effort accordingly.

This assignment is marked out of 40 and counts 40% towards your final grade. You will be marked against the criteria listed below. Your mark is calculated by adding together the grade for each criterion.

Your mark may be varied by penalties and bonus marks as detailed in the criteria. The lowest possible grade is zero. The maximum possible grade is 40.

Before you demonstrate your system, you should thoroughly test your system:

- Attempt to log in with an incorrect password
- Enter invalid data and check that validation works on all fields

- Check that you can log out
- Attempt to access secure pages without logging in

Marking Criteria

Core Functionality

- 0: The system is unusable or regularly crashes.
- 5: The system does not implement all of the core functionality or validation does not work.
- 10: The system implements all of the core features. User friendly error messages are displayed when the user enters invalid input.

Innovation Functionality

- 0: The innovation functionality is not implemented.
- 5: Half the proposed innovation functionality is well implemented or all the innovation functionality is implemented but poorly designed.
- 10: The system implements all of the proposed innovation functionality and the functionality is appropriately implemented and well designed.

Non-functional requirements

- 0: The system does not make use of JDBC, security or JavaServer Faces
- 2: The system is using Java EE technologies but incompletely or inappropriately.
- 5: The system uses Java EE technologies elegantly and appropriately, including connection pooling and parameterized queries. Connections are closed.

Design and architecture

- 0: The system lacks structure or coherent design.
- 1: The system does not separate code into layers. Data access code, domain logic and presentation logic are integrated. Presentation logic is mixed with HTML.
- 2: The system has been partially separated into layers but there is some mixing of presentation, domain and data access logic.
- 5: The system is separated into clear layers with separate responsibilities.

Documentation and coding style

- 0: There is little to no documentation. Code is messy. There are large amounts of unused or commented-out code.
- 2: Some code has comments. Classes, methods and variables have clear names.
- 5: **All** non-trivial methods and classes have comments. All code is consistently clear and well presented. Technologies are used appropriately and idiomatically (i.e., using standard coding conventions for the technology). There is no unnecessary duplication of code.

Revision control

- 0: Revision control has not been used.
- 2: Revision control has been used.
- 5: There are at least 8 meaningful commits and every commit has a clearly written commit message explaining the changes made.

Bonus and Penalties

- -5: The system crashes during demonstration
- -5: No proposal given to your tutor during the first two weeks of semester
- -5: Each day late (five mark penalty per day late)
- +1: The system is demonstrated in Week 6
- +2: The system is demonstrated in or before Week 5