THE UNIVERSITY OF SYDNEY

School of Information Technologies

COMP5138 Database Management Systems **Semester 2, 2014**

Assignment 3: Database Application Development

Group assignment (10%)

Introduction

The objectives of this assignment are to gain practical experience in interacting with a relational database using an Application Programming Interface (API) (JDBC). This assignment additionally provides an opportunity to use more advanced features of a database, such as stored procedures, functions, access control privileges, as well as the use of optimistic concurrency control techniques.

This is a group assignment for teams of about 3 members, and it is assumed that you will continue in your Assignment 2 group. You should inform your tutor as soon as possible if you wish to change groups.

Please also keep an eye on your email for any further announcements made on eLearning about this assignment.

Submission Details

The final submission of your application is due at 11:59pm on the 31st of October 2014. You should submit the *items for submission* (detailed below) in a single zip file via eLearning.

Items for submission

Please submit your solution in the 'Assignment' section of the unit's e-learning site by the deadline, including the following items in a single zip file:

- A single sql file containing the SQL statements necessary to generate the database schema and sample data. This should contain the original schema and insert SQL statements, and incorporate any changes or additions you may have made.
- An archive of your Eclipse Kepler project file, created as shown here: http://agile.csc.ncsu.edu/SEMaterials/tutorials/import_export/index.html#section2_0
- All team members must also confirm that none of their work is plagiarised by signing and submitting a cover sheet to their tutor for the final submission.

Marking

This assignment is worth 10% of your final grade for the unit of study. Your group's submission will be marked according to the attached rubric.

Group member participation

If members of your group do not contribute sufficiently you should alert your tutor as soon as possible. The tutor has the discretion to scale the group's mark for each member as follows:

Level of contribution	Proportion of final grade received
No participation.	0%
Full understanding of the submitted work.	50%
Minor contributor to the group's submission.	75%
Major contributor to the group's submission.	100%

IMPORTANT: Policy relating to Academic Dishonesty and Plagiarism.

All teams must declare that the work is original and not plagiarised from the work of others. In assessing a piece of submitted work, the School of IT may reproduce it entirely, may provide a copy to another member of faculty, and/or communicate a copy of this assignment to a plagiarism checking service or in-house computer program. A copy of the assignment may be maintained by the service or the School of IT for the purpose of future plagiarism checking.

Section 1: Introducing the Issue Tracking System

In this assignment you will be working with an Issue Tracking System under development known as the "Issue Tracker". The system still requires development in numerous areas including in the interaction with the database. Your main task is to handle requests for reads and writes to the database, which originate from the user interface (UI). We first describe the main features that the Issue Tracker should include from a UI perspective, and then indicate where the majority of database interaction code should be implemented.

Main Features

Logging On

The first screen a user is prompted with when starting the issue tracker is the screen show in Figure 1, where a user would enter in their user id. This feature is still under development and currently only requires that a user enters their id to view their associated ***issues***. Each issue has a creator, and can also have a resolver and/or verifier and/or a description. An issue is associated with a user if, for that issue, that user appears as the creator or resolver or verifier. Once logged in, a user will be able to see their associated issues. As a starting point, a user with id 1 is created in the assignment's attached sql file.

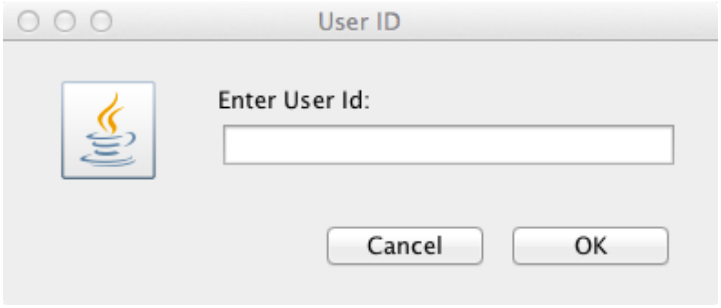

Figure 1 – Login Screen

Viewing Issue List

Once logged in, a user will be able to see the titles for all their associated issues in the issue list, as illustrated below in Figure 2.

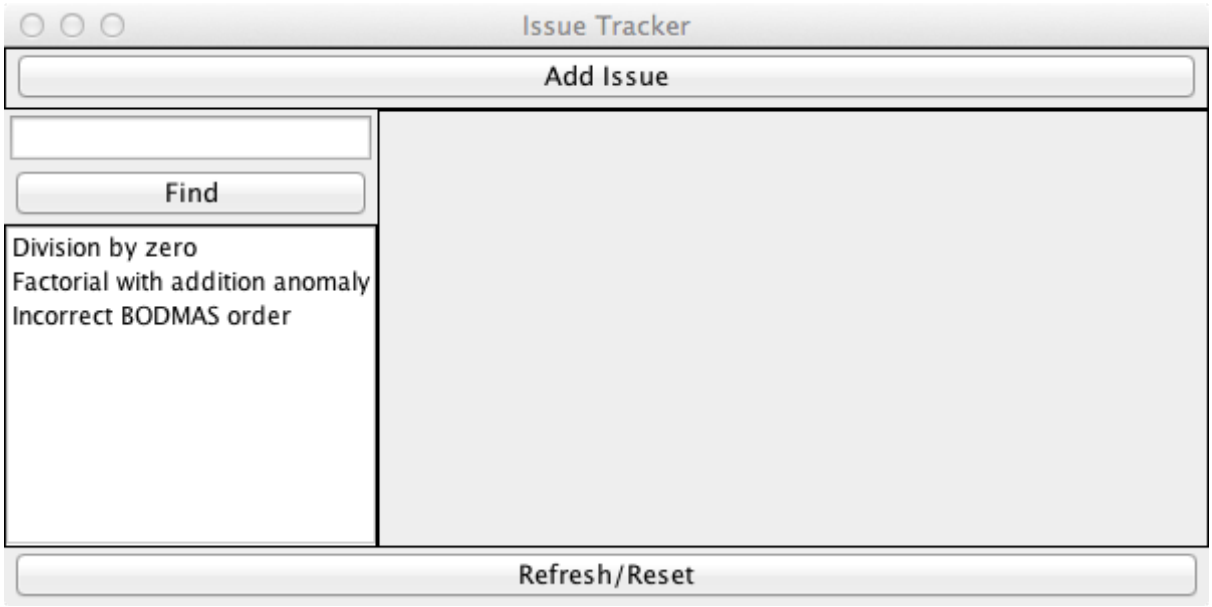

Figure 2 – Viewing Issue List

Finding Issues

A user can also search *through their associated issues* by giving the issue tracker a search expression as shown in Figure 3, and clicking on Find. This search expression is composed of an optional *name phrase*, followed by an optional sequence of *title/description phrases*. When a *name phrase* is specified (delimited by a starting and ending '@' character), then only issues that have a user with a first or last name that includes the *name phrase* will be shown. When a *title/description* phrase is used, then only issues including the *title/description* phrase in the title or description will be shown. Multiple title/description phrases can be entered by separating these phrases with the "|" character. When a sequence of title/description phrases are entered like this, then issues containing any of the keywords in the sequence in either the title or description field are shown. When both *name* and *title/description* phrases are specified, then both name AND title/description conditions must hold for an issue in the user's associated issues in order for it to be shown. A number of example search expressions are shown below, along with their meaning:

@Dean@zero This will return (all of the logged in user's associated issues that have a creator or resolver or verifier with a first or last name that includes the phrase 'Dean') and (includes the phrase 'zero' in the title or description.)

@Dean@zero|BODMAS This will return (all of the logged in user's associated issues that have a creator or resolver or verifier with a first or last name that includes the phrase 'Dean') and (includes the phrase 'zero' or 'BODMAS' in the title or description.)

@Dean@ This will return all of the logged in user's associated issues that have a creator or resolver or verifier with a first or last name that includes the phrase 'Dean'.

zero This will return all of the logged in user's associated issues that include the phrase 'zero' in the title or description

zero|BODMAS This will return all of the logged in user's associated issues that include the phrase 'zero' or 'BODMAS' in the title or description. An example of this search expression is shown in Figure 3.

A blank search expression that is searched should show all of the logged in user's associated issues.

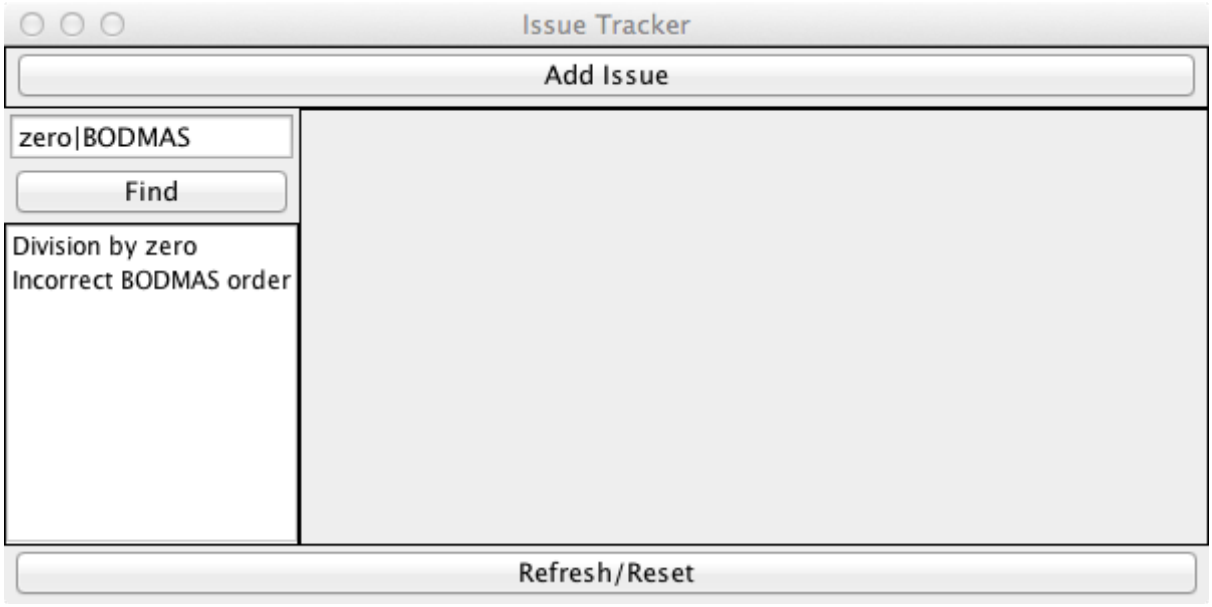

Figure 3 – Finding Issues

Viewing an Issue's details

When the title of an issue is selected on the issue list, then the details for that issue should be shown, as illustrated in Figure 4. The description for the issue is shown in the large text area under the Verifier field.

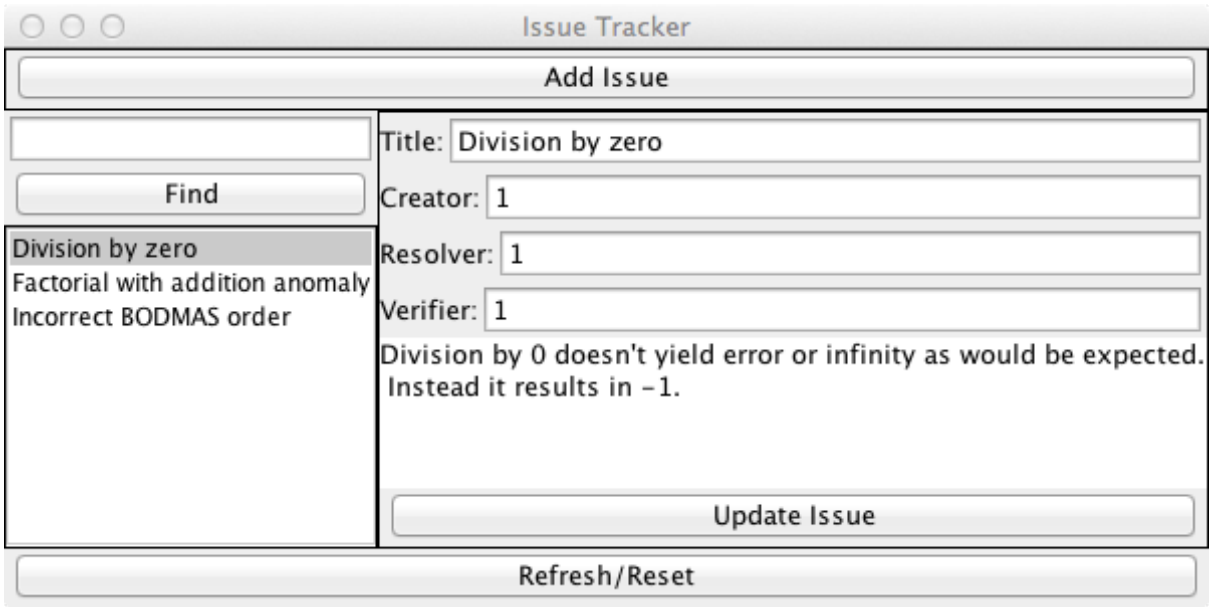

Figure 4 – View an Issue's Details

Adding an Issue

Users may also add a new issue by clicking on the Add Issue button, entering issue details in the popup dialog that appears, and then clicking on Add Issue.

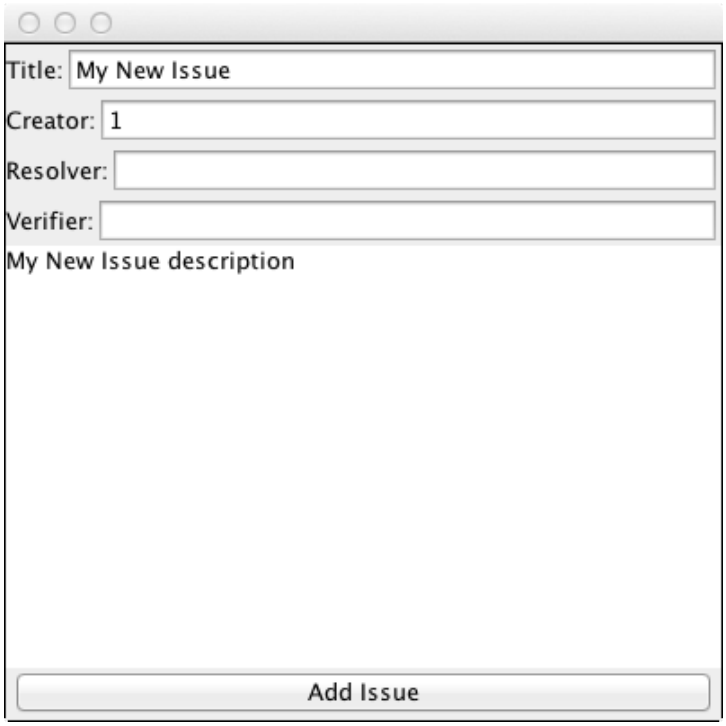

Figure 5 – Adding a new Issue

Updating Issues

Users can also update an issue by modifying data in the issue details screen as shown in Figure 6, and clicking on the Update Issue Button.

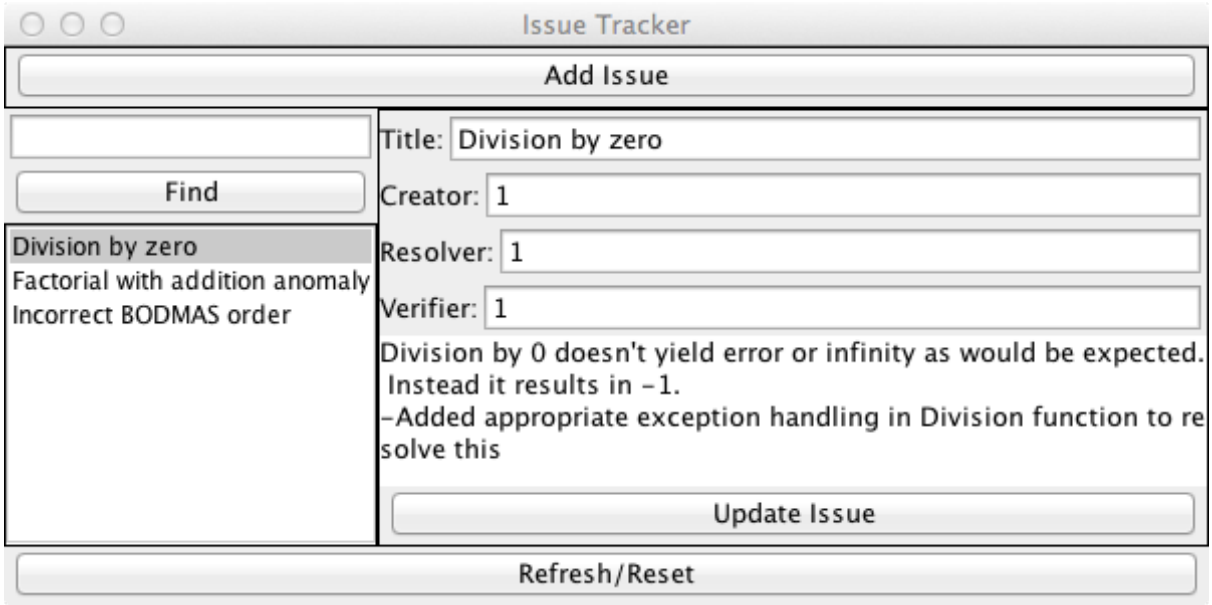

Figure 6 – Updating an Issue

To update all data on the UI to reflect database contents you can press Refresh/Reset.

Database Interaction Code

The assignment 3 folder in the assignment section of eLearning will contain both a zip file encapsulating the Eclipse project for IssueTracker, as well as a sql file which contains sql code you need to run before starting the application to create the IssueTracker database.
To look through the IssueTracker code, you'll need to import the Eclipse project archive file into Eclipse. Please begin by trying to import the project into Eclipse Kepler using the steps below:

1) Right Click Assignment3Archive.zip > Extract All > Extract

2) Start Eclipse Kepler and choose a new location for your workspace in a new folder.

3) Click on File>Import>General>Existing Projects into Workspace>Next

4) Browse to the folder you extracted Assignment3Archive to and click OK.

5) Ensure the Assignment 3 Project is ticked in the Projects box, and click Finish.

If you can't import the zip file archive into Eclipse on your machine, you should also attempt to import the project using the instructions here:

http://agile.csc.ncsu.edu/SEMaterials/tutorials/import_export/index.html#section1_0

Once you import the project in Eclipse, you should notice that there are 3 main packages in the solution: Presentation, Business, and Data. The UI (in the Presentation package) is currently coded to invoke logic in the business layer (eg: see IssueProvider class in the Business package), which in turn delegates responsibility for interacting with the database to an appropriate repository in the data layer (eg: see OracleRepositoryProvider in the Data package). Notice that separating concerns in this way makes it easier to write a different RepositoryProvider in the data layer (eg: we could write a MySQLRepositoryProvider) if we wanted to change our database management system. In this assignment you will mainly be working on writing the appropriate queries and SQL commands to fulfil the IssueTracker functionality described in Section 1; where these SQL Commands and queries should be written in the OracleRepositoryProvider. The application's main method can be found in the IssueTrackerFrame class. You can run this main method to run the entire application by right clicking the IssueTrackerFrame file on the Project Explorer > Debug > Java Application.

Section 2: Your Task

Core Functionality

In this assignment you are provided with an Eclipse Project that must serve as the starting point for your assignment. Your task is to provide a complete implementation for the OracleRepositoryProvider, as well as make any modifications necessary to the database schema, the business and presentation packages that are necessary to provide the Issue Tracker features described in Section 1.

Extension

As an extension, you will also need to consider how to prevent the lost update anomaly without the use of long-running system transactions. Long running system transactions are often seen as undesirable since they tend to lock resources while they run, potentially making those resources unavailable for other transactions. Hence it is often recommended that transactions are short. However, outside of transactions, anomalies like lost updates, which we saw in lectures become possible. As an extension, you will need to extend the system to make use of ***optimistic offline locking*** to prevent lost updates resulting from multiple users updating the same issue at the same time. As a starting point in learning about optimistic offline locking, you might like to read through:

http://en.wikipedia.org/wiki/Optimistic_concurrency_control,
http://www.agiledata.org/essays/concurrencyControl.html#OptimisticLocking,
http://martinfowler.com/eaaCatalog/optimisticOfflineLock.html.

Your system should prevent an update to an Issue that would cause a lost update to occur.

You should not attempt the extension until you are entirely satisfied with all other components. Marks are not awarded for the extension unless full marks are achieved for all other sections.

Marking Rubric

Your submissions will be marked according to the following rubric, with a maximum possible score of 10 points.

	2 Points	4 Points
Core DB & Application Programming	Correct use of SQL in the data layer, sufficient for implementing the core functionality, including correct handling of NULL values.	Excellent use DB interaction code including: - excellent, clean encapsulation and implementation of SQL and java code in the appropriate data, business, or presentation package. - judicious use of SQL connections (eg: including correct use of close() depending on whether physical connections or a connection pool is used), - appropriate use of transactions - error conditions and exceptions handled appropriately

	1 Point	2 Points
Security	Database interactions demonstrate awareness of security issues, and avoid SQL injection attacks. This also includes considering what is displayed on both the console and application GUI when errors occur.	Demonstrated comprehensive understanding of security issues, with all appropriate steps taken to safeguard against SQL injection and limit database privileges of the client application.
Stored Procedures	At least one stored procedure or function implemented and called from the application.	At least one stored procedure or function that makes appropriate use of a cursor to return a result to the application.
Extension	Correct Implementation of Optimistic Offline Lock to prevent lost updates for the Issue update feature.	Application provides users with a notification when they cannot update an issue because the database has changed since they last read the issue. When the notification is received, users should be presented with the option, and be able to either re-load the issue or return to their edits for the issue.