**School of Computing's MSc project ideas database system**

ENTWA COURSEWORK

UP710297

BSc. (Hons) Software Engineering

**CONTENTS**

1.INTRODUCTION

2. DESIGN

3. IMPLEMENTATION AND TESTING

4. SUMMARY

5. REFERENCES

**1. INTRODUCTION**

The new system aims to provide more options to search project ideas and organisations and to ensure that only owners and designated administrators can edit and withdraw or archive projects ideas and organisations. An owner is one who creates the project idea or organisation.

The objectives of the coursework are to:

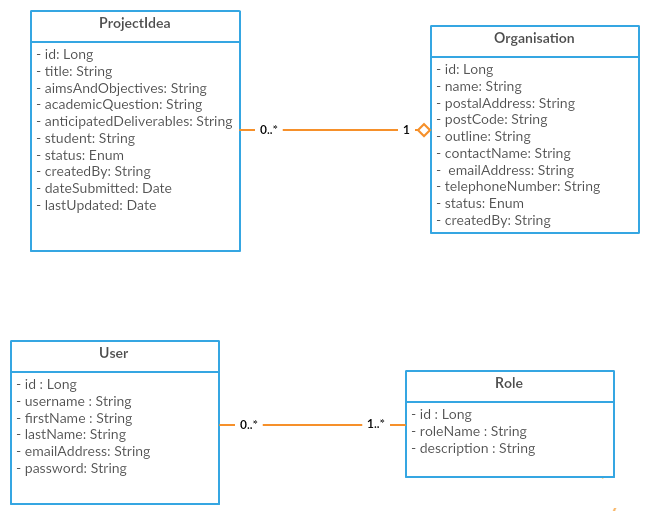
* implement the search functionality using Ajax
* implement concepts and skills learnt during lectures and tutorials
* develop a secure Java EE application using JSF and a security realm
* develop a robust application by separating concerns into different layers e.g. business logic layer, data access layer, persistence layer

**2. DESIGN**

This section describes the relationship between classes and highlights the user interface design.

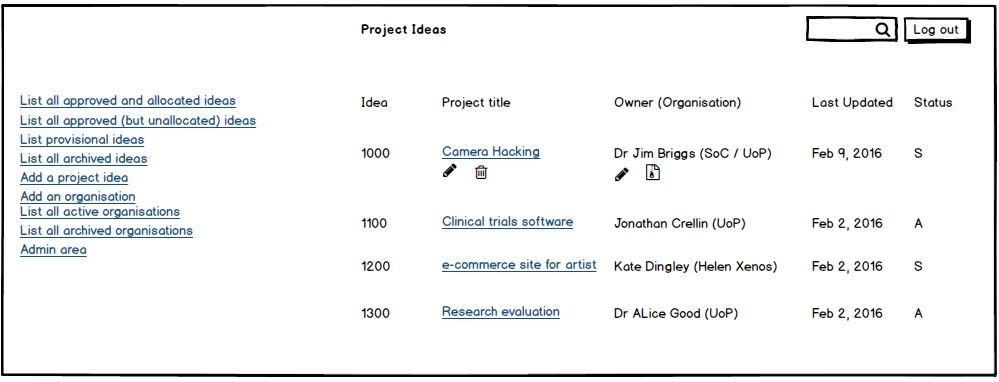
The roles available to a user are super user, admin, user (regular user) and anonymous. A super user can create and manage users; and manage all project ideas and organisations. An admin can manage all project ideas and organisations. A user (regular user) can create project ideas and organisations; and can manage only project ideas and organisations the user created. An anonymous user has only read privileges therefore can’t create and manage project ideas and organisations.

**2.1 CLASS DIAGRAMS**

****

An Organisation has a one-many relationship with ProjectIdea: a project idea belongs to an organisation and an organisation has many project ideas. Whilst the User class has a many-many relationship with the Role class: a user can have one to many roles and a role belongs to zero to many users.

**2.2 USER INTERFACE DESIGN**



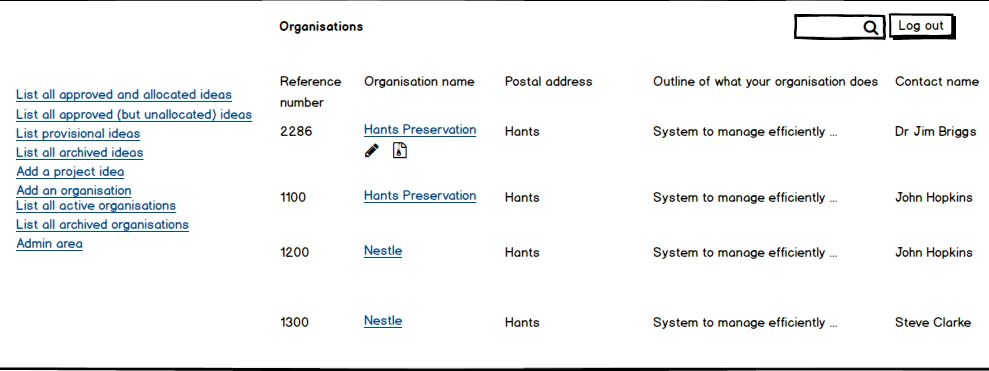
For starters, the location of the navigation bar has been moved from the bottom to the left. This makes it easier to locate. Secondly, the first four links: list all approved and allocated ideas; list all approved (but unallocated) ideas; list provisional ideas; and list all archived ideas enable users to list project ideas based on their status – provisional, approved or allocated. This provides more options to the user to locate a project idea.

The figure above represents the general view of each of the four links. The logged-in user, Dr. Jim Briggs, is the owner of “Camera Hacking” project idea and “SoC/UoP” organisation and he can consequently only edit, withdraw or delete them. As shown above, the system doesn’t provide the option to edit, withdraw or archive a project idea or organisation that doesn’t belong to Dr Jim Briggs. This ensures a good user experience.

The search box at the top is implemented using Ajax and displays, depending on the view, approved and allocated ideas, approved (but unallocated) ideas, provisional ideas or archived ideas – an archived project idea belongs to an archived organisation. This also provides a good user experience.

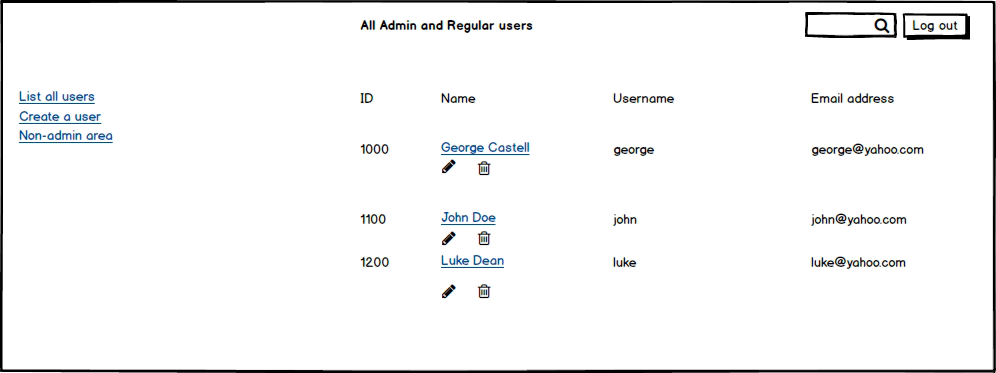
The “Add a project idea” and “Add an organisation” links reuse the “Submit a project idea” view in the current system -- editing an organisation and a project idea also makes use of this view. The only change made is to the drop down menu that shows a list of organisations: one can search the list of organisations using Ajax. This makes it easier to select an organisation. I didn’t see the need to majorly alter this view because the aims of the client are to improve the searching functionality of the system and authorise each user; there was no complaint regarding the project idea or organisation creation.

Viewing a project idea is another unchanged view and viewing an organisation is a subset of the view of a project idea.

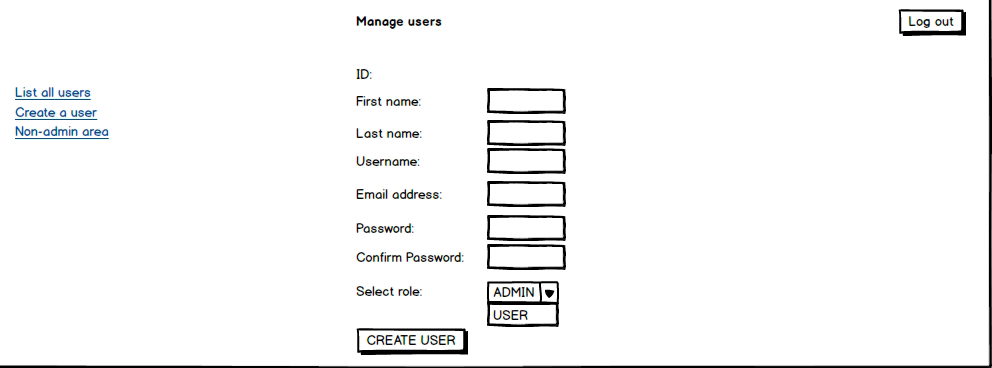


An organisation can be archived by changing its status from “Active” to “Archived”. This helps to prune down the list of organisations by displaying only active organisations and also maintain a record by not deleting an organisation.

The “List all active organisations” and “List all archived organisations” use the view shown above. As shown Dr. Jim Briggs owns Hants Preservation with a reference number of 2286 but not the other three organisations. The search box is implemented using Ajax.



Only super users can visit the admin area. The admin area lists all users and with the search box, a super user can search the list of users by Ajax. Ajax provides a good user experience and the view above makes it easy to manage users.



The figure above is used to create and manage users. A super user can assign admin privileges to a designated user with the form above.

I’m convinced that the design of the user interfaces will improve the user experience of the user, enable users to search better for a project idea and organisation and is user friendly.

**3. IMPLEMENTATION AND TESTING**

This section describes the approach taken to develop the application, experiences acquired, choice of development tools and how components were tested. It also highlights areas of development that were particularly tricky, and explains how I overcame them.

I used a combination of the bottom-up approach and top-down approach by first building the entities and façade, followed by building the user interfaces. Primefaces 5.0 was used to build the interfaces.

Building the user interfaces was quite challenging in the beginning especially populating a drop-down menu with a list of objects and using a converter to convert a string representation of an object back to an object. I was able to overcome these challenges by using Stackoverflow and Google.

The separation of concerns design has been added to my skill set. Each layer e.g. persistence, façade, service etc. has a specific purpose and is well separated. This is a very clean way to develop software applications and I intend to implement this design in future projects. I found the sample applications given during the lectures and tutorials particularly helpful in adhering to this design.

I was also fascinated with the built-in security especially the JDBC security realm -- setting up the JDBC realm wasn’t easy but Google came to my aid; easier implementation of Ajax using Primefaces than with JQuery or JavaScript; minimal CSS needed to layout the application; implementation of validations in JSF pages using “required” and “requiredmessage” attributes; glue/binding between the JSF pages and the backend through JavaBeans; and dependency injections using tags like @Inject and @EJB.

The Netbeans IDE (Netbeans 8.1) is a refreshing change from IDEs for dynamic programming languages. The syntax completion, compiler and coding hinting to mention a few, were very useful. The Glassfish server is a bit quirky: one minute the application deploys without errors then next minute it refuses to deploy because of “errors”. Stopping the server and running “Clean and Build”, “Deploy” and “Run” usually fixed these issues. I have mixed feelings about consulting the server log when there’s an error/bug. Sometimes the file and line causing the error is explicitly stated and other times it’s not. When not explicitly stated, the Netbeans debug feature was extremely helpful. The ability to watch variables in the debug console helped in locating the source of bugs.

Unfortunately I didn’t write any test scripts to test the components. Testing was performed manually by ensuring the requirements stated in the requirements specification had been met.

Concluding this section without mentioning how instrumental Git was in developing this application will be an injustice. Creating git branches to fix bugs and committing a working state helped immensely. Dropbox wasn’t used as much as github though.

**4. SUMMARY**

This section summarises what I’ve undertaken in the coursework; good and design choices; and what I would have done better or alternatively with the benefit of hindsight.

The application does the following:

* authenticates and authorises each user thereby improving security of the application
* ensures a user with an anonymous privilege has only READ rights
* ensures a regular user only manages (edits and withdraws) project ideas the user owns i.e. created
* ensures a regular user only manages (edits and archives) organisations the user owns
* ensures a designated user can manage any project idea or organisation
* ensures a super-user can create a new user with admin or regular user privileges
* ensures a super-user can designate a user as an admin
* provides options to search for an organisation based on its status
* provides more options to search for a project idea based on its status
* doesn’t provide the option to manage a project idea or organisation if the user isn’t the owner or an admin or a super-user.

The choice of PrimeFaces was good as it provided me with access to some nice UI components like a drop-down menu with an Ajax search field, vertical navigation bar and horizontal menu bar. Another good design choice was to design the structure of the application using layouts to give each page a similar structure. Letting users know which fields are compulsory at a glance was also a good design choice. The database design was also spot-on in my opinion.

A better design should show the project ideas belonging to an organisation by toggling the organisation. Also giving users the option to change their details is needed. Once a user has been assigned a password, the system doesn’t provide an option to change the password; a better design should incorporate this.

The aesthetic quality of the application is quite poor so more styling is needed to beautify the application. I’m also a bit uncomfortable with the overlapping property of the content section and the position of the footer. Ideally the position of the footer should be at the end of the content section. Mobile friendliness wasn’t considered when building the application but it should!

I should have also written unit tests using JUnit and integration tests for the application. With tests in place, the confidence level when making changes will be high due to regression testing.

Further tests are needed with large data to ascertain the actual performance level of the application. The test data used was small and to gauge the performance against this small test data could lead to inaccurate results.

Finally, the application should handle exceptions or bugs better. Codes known to throw exceptions or errors need to be wrapped in a try-catch block and users should be informed accordingly. This provides a better user experience.

**5. REFERENCES**

* Darcin, Y. (2010). *Powerful Ajax with Update and Process attributes in PrimeFaces.* Retrieved March, 14, 2016, from <http://yigitdarcin.com/2010/01/27/powerful-ajax-with-update-and-process-attributes-in-primefaces/>
* Goncalves, A. (2013). Beginning Java EE 7. Great Britain: Amazon
* Heffelfinger, R. (2014). *Java EE 7 with GlassFish 4 Application Server* (3rd ed.). Birmingham: Packt.
* *How to generate MD5 Hash in Java - String Byte Array digest Example.* Retrieved March, 14, 2016, from <http://javarevisited.blogspot.co.uk/2013/03/generate-md5-hash-in-java-string-byte-array-example-tutorial.html>
* Jugo, G. (2011). *JDBC security realm with glassfish and jsf.* Retrieved March, 14, 2016, from <http://jugojava.blogspot.co.uk/2011/02/jdbc-security-realm-with-glassfish-and.html>