2022

Project Portfolio

COIT13230 - APPLICATION DEVELOPMENT PROJECT

RORY ALLEN

S# - 12149026 | DST

Contents

Project Summary

Project Overview

The product of this project is a software system which allows the management of information in a work integrated learning (WIL) program offered at Central Queensland University (CQU). The system will provide a web application that allows CQU to collaborate with industry partners to deliver work placement services for its stakeholders: CQU teaching staff, CQU students, and work placement providers within the industry. The system will facilitate CQU in offering its students work placements within industry partner's organisations, to assist them in entering the workforce upon graduation.

Business case

CQU's School of Engineering and Technology offers its students the opportunity to work alongside industry professionals through the WIL program, each year. Students undertaking such a placement benefit from the experience of working in a professional context and gain valuable knowledge to assist with their transition into the industry. As more opportunities arise, the need for an effective management system becomes apparent. Likewise, accessibility to a user-facing service that allows real-time collaboration between the various types of stakeholders would increase the program's effectiveness and reach.

The Work Integrated Learning Management Application (WILMA) is designed to be used by both the client (CQU) and any interested CQU students and industry partners. Educators at CQU can manage placements offered by industry partners and applications submitted by its students. Partners can register their availability in providing placements, submit position vacancies, and review potential candidates. Students can view the positions on offer and submit their applications for consideration.

The aim of the project is to provides a marketplace of industry positions, allowing partners and students to connect though a user-friendly interface. Management of placements should be a seamless experience, mediated by CQU educators that endeavour to provide their students with valuable industry exposure.

Validation of the product will be measured in terms of its usability and performance in providing this service for the client. A successful implementation will replace the existing WIL information workflows, and provide a hub for the WIL program, moving forward.

Scope of work

The WILMA system includes several components to facilitate management of the existing WIL program. It includes a robust registration subsystem, that allows each of the three stakeholder groups to register accounts and be assigned system usage roles. The account component allows users to manage their account in accordance with their role.

The position management component allows users to interact with jobs and placements appropriately for their role. Educators can create expressions of interest (EOI's), approve, and edit posted positions, and manage the status of these positions. Partners can create positions either from posted EOIs or *de novo*. Students can view and apply for approved positions.

The forum subsystem allows users to post and reply, questions and answers. This is open for all users to interact with each other and ask discuss work placement topics. Posts are categorised and tagged to allow intuitive sorting and searching for all users.

The notification component distributes updates on WILMA activities to users appropriately. For example, if an industry partner submits a placement opportunity which gets approved, they will be notified when students submit their applications for consideration. This subsystem batches all the

daily activities that pertain to the user and batches them in a single daily email, with links to the appropriate details.

Constraints

There were no major constraints imposed by the client besides the 12-week timeframe. The project was easily achievable in that time, but there were other constraints that were imposed purely due to the context. The team had no budget to invest in the project, so had to invest their time voluntarily for all phases of the project from planning to deployment. Their hardware and software resources were limited to what they already owned or could obtain for free. External time commitments also meant that management of their time and effort needed to be scheduled and strictly adhered to.

Stakeholders

The client, CQU, was the primary stakeholder. Educators from this group can use the WILMA with the most elevated role, allowing them to manage most aspects of the placement process. This group will be presented with formal documentation of the software detailing its operation and technical specifications.

The second type of stakeholder are CQU students seeking work placement. They will use the software to view and apply for WIL placements. These users are relatively technically proficient so will have no problem navigating the applications functionality.

The third group of stakeholders are the industry partners who will use the system to submit position opportunities, and review applications. The software is designed with usability in mind so even if they are not particularly technically adept, they should have no problem using the system.

Assumptions

It is assumed that, due to a working budget being unavailable, the client does not have extensive resources available to develop a high-end solution. A product of reasonable quality and performance would suit their requirements and should be achievable within the limited timeframe available. It can also be assumed that although the scope of the clients requirements are reasonable, user experience considerations may require that alternate workflows may provide a better solution which would, in turn, require additional investment of volunteer time to develop.

Project Requirements

User requirements

Updated user requirements are as follows:

- Facilitate placement of students in the WIL program, offered at CQU
- Manage information flow between three roles: Students, Educators, and Industry Partners.
- User roles are assigned based on selection of user type at registration.
- All users can manage their own account details, view listed positions, and post/reply in the forum.
- Educators can modify posted positions and forum content and can publish expressions of interest (EOI's) for placements that they would like to be provided by industry partners.
- Industry partners can post new position vacancies (pending approval), accept position applications, and create new placements (pending approval) from EOI's.
- Students can upload their resumes and apply for vacant positions.
- Once the term of a position has expired, partners can provide feedback on the student's performance.
- Designed to be reliable and performant (1000 concurrent users, database backed up daily, response time indistinguishable from local operations)
- Designed to be secure (users verified by email, access determined by role, SSL certificates, encrypted user data)
- Designed to be extensible (RESTful web service endpoints, template-based UI)
- Designed to be usability and maintained (Established UX heuristics, included operational support period, periodic service updates)

System Requirements

Functional Requirements

The current functional requirements of the WILMA system and their associated codes are listed below in **Table 1**:

Code	***Requirement***
FR-NM001	Users can create a new account
FR-NM002	Roles and authorities are assigned to a new user account based on their account type
FR-ND003	Users can log in/out
FR-RA004	Educators (and only educators) can create/update/delete placement expressions of interest and their status
FR-RA005	Educators can monitor/review/approve/deny/update placements (including their statuses)

FR-ND007	Users can view and update their profile information (including name, contact information, disciplines, and resumes)
FR-NS008	Industry partners can add a description about their business
FR-NS009	Jobs/placements can be filtered by categories and type (job or placement)
FR-NM010	Unregistered users can read about the system, contact system team/staff, or register for an account without needing to be signed in
FR-NS012	Users can view more details about a job or placement by clicking on it
FR-NS013	Students can submit applications for jobs/placements
FR-NS014	Partners get notified of new applications for their advertised job/placement
FR-NS015	Allow users to publicly ask and answer questions in a forum
FR-RA016	Partners can view EOIs and create a new placement from them
FR-NS017	Job/placement applications will include details the student's profile details
FR-NS018	Job/placement applications can include files (e.g., resume or cover letter)
FR-NS019	Applications will be emailed to industry partners at a set time each day via email (batch send)
FR-ND020	After having a successful applicant, jobs should be marked as filled
FR-ND021	Placements that are filled should be marked as occupied indicating they are not currently available
FR-ND022	Partners can leave feedback (submit a report) about the student after a placement finishes
FR-ND023	Educators can review student-placement feedback left by partners
FR-ND024	Placement can be marked as complete when the educator finishes reviewing the feedback report
FR-RA026	Created jobs and placement will be published directly to the marketplace
FR-RA027	Expressions of interest can be converted to a placement by interested partners
FR-NS033	Logged out users are returned to the public home page
FR-NS036	Q&A forum posts must be categorised
FR-NS037	Q&A forum posts can contain tags (predefined) to aid searching and filtering

Table 1: functional requirements of the WILMA system

Non-functional Requirements

The current non-functional requirements of the WILMA system and their associated codes are listed below in **Table 2**:

Code	***Requirement***
NR-NS041	User passwords only to be stored using a secure and trusted hash and salt algorithm implemented by BCryptPasswordEncoder
NR-NS042	Must only supply access to zones according to a user's roles & authorities
NR-NM043	Jobs/Placements will only be visible to registered users
NR-NM046	Email addresses must be verified before successfully registering

NR-NS047	System should be able to authenticate an API user via an API key
NR-NM048	All educators are given the ADMIN role by default
NR-NS049	Q&A forum content moderated only by users with the ADMI N role
NR-RA050	Expressions of interest only visible to educators and industry partners
NR-NS051	User login can be via (1) Username & Password or (2) Email & Password combinations
NR-NM053	User profile name and email fields are mandatory
NR-RA054	Jobs and placements must have a category
NR-UD055	Submitted applications will remain on a student's record, but get archived after 1 year
NR-UD057	System maintenance, updates, or upgrades should occur during quietest web traffic times (preferably Sunday) with user notice
NR-UD058	System must be operational and online during weekdays 4am – 12pm (expected peak usage)
NR-UD059	Cloud hosting will be used to minimize the risk of service downtime, with a minimum overall target of 98% uptime
NR-UD060	User experience should be maximised through use of a cached content delivery network (such as Cloudflare)
NR-UD061	Site MUST have a current SSL (Secure Sockets Layer) certificate
NR-UD062	Platform should be scaled initially to handle a load of 1000 concurrent users
NR-UD063	Persisted data will be stored in a cloud hosted MySQL relational database
NR-UD064	Database content must be backed up daily
NR-XX065	Codebase must be segregated using a monolithic modular design (Repository, Service, Configuration etc)
NR-XX066	Application must be able to deploy to a Linux or Windows server
NR-XX067	Application codebase must be able to run on Linux, Windows, and Mac OS
NR-NS068	Source code to include adequate Javadoc comments so that online documentation can be generated
NR-NS069	System must expose sufficient REST endpoints such that the system could implement an external/distributed frontend UI
NR-NS070	API endpoints should comply with the OpenAPI specification (swagger)

Table 2: non-functional requirements of the WILMA system

Design

Product Use cases

Use case diagrams have been updated to reflect the current workflow of the WILMA system. The registration use cases are described below in **Figure 1**. The position management use cases can be seen below in **Figure 2**. **Figure 3** below depicts the use cases pertaining to the Q&A forums. **Figure 4** below show the use cases for managing user profile details.

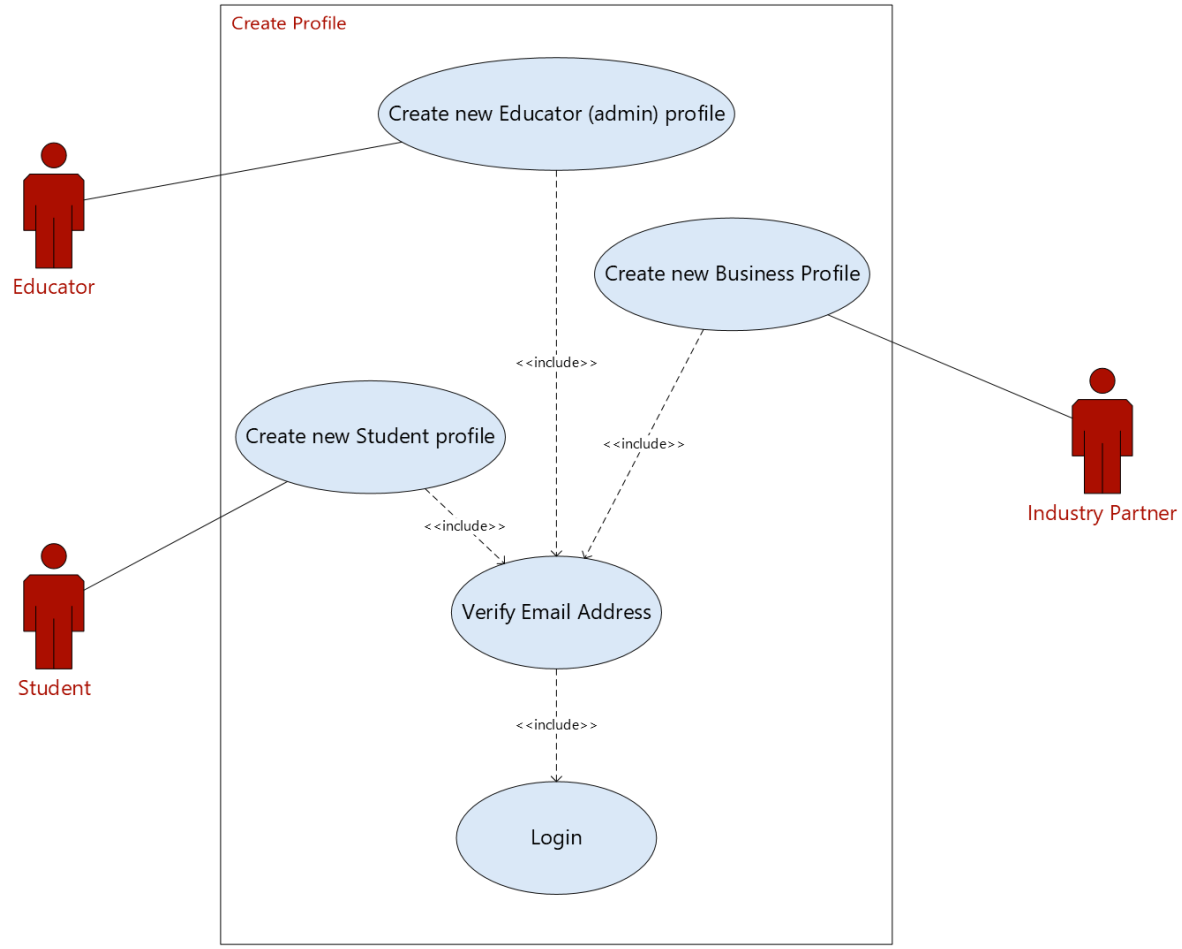

Figure 1: Registration process for the WILMA system

Figure 2: Position management use-cases for the WILMA system

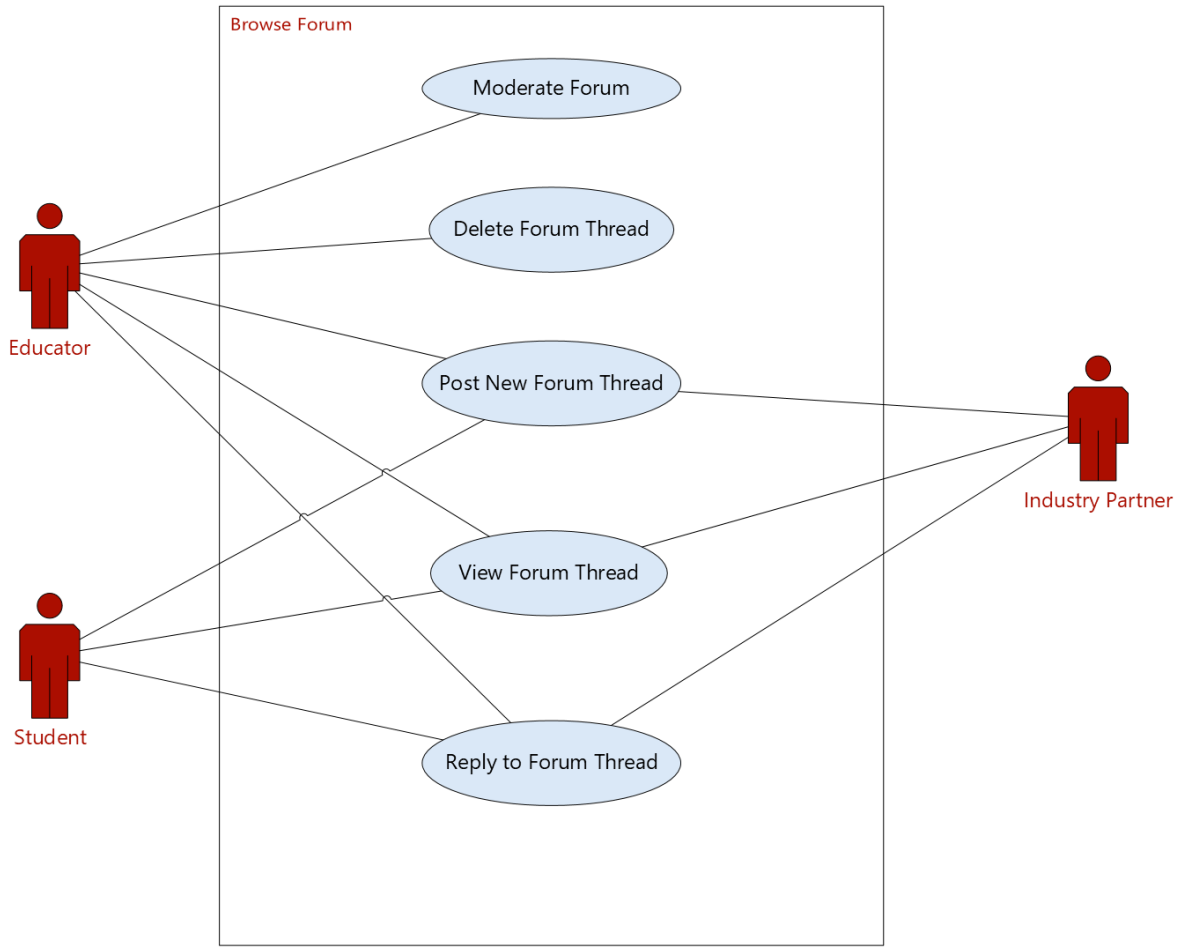

Figure 3: Forum use-cases for the WILMA system

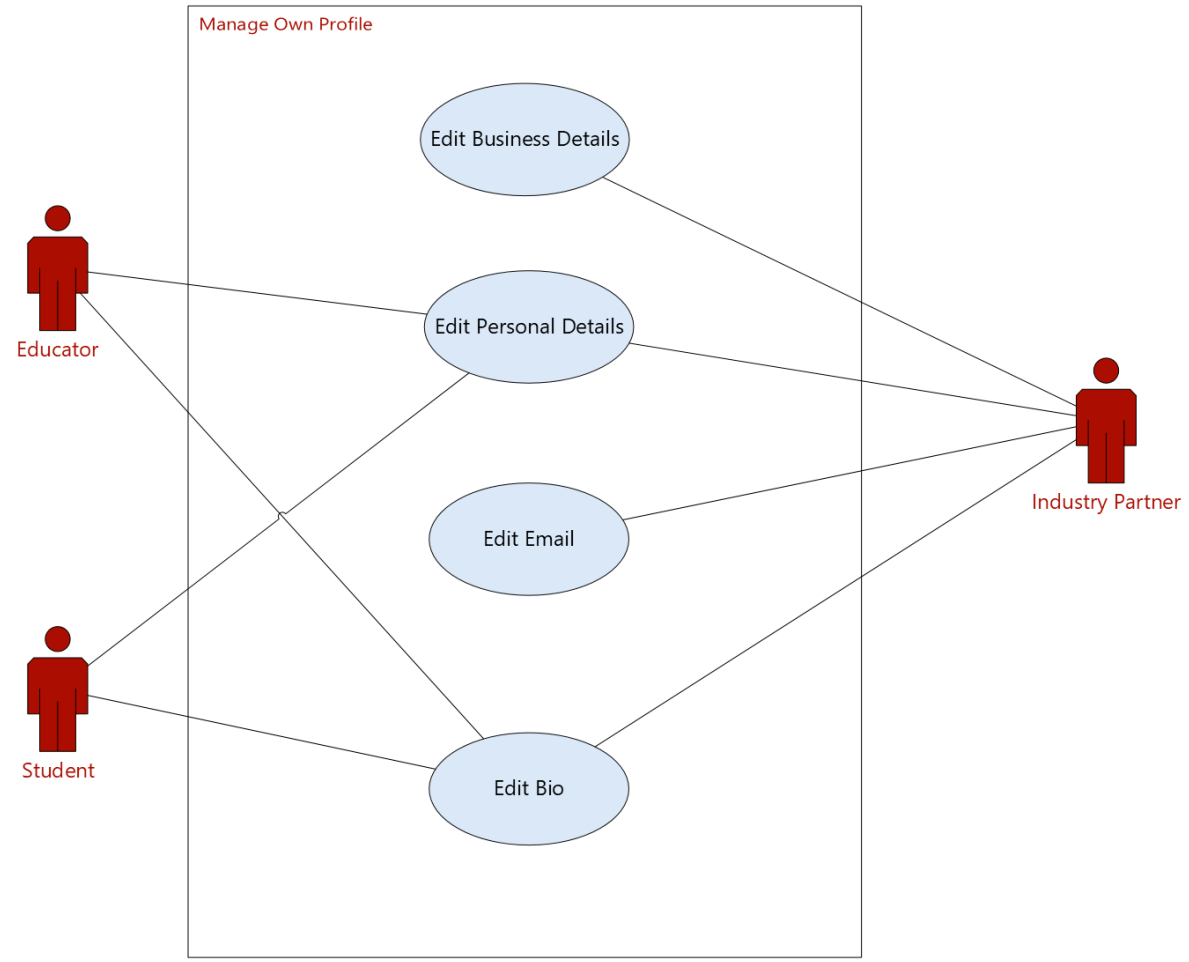

Figure 4: Profile management use-cases for the WILMA system

UML class diagrams

The UML class diagrams remain unchanged from the original design documentation.

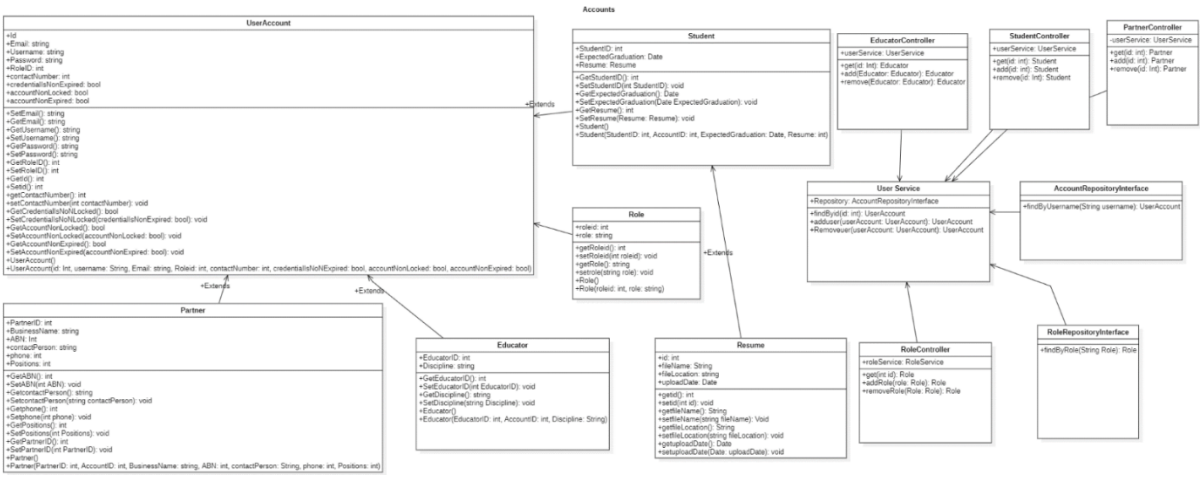

Figure 5:Class Diagram #1 for WILMA System

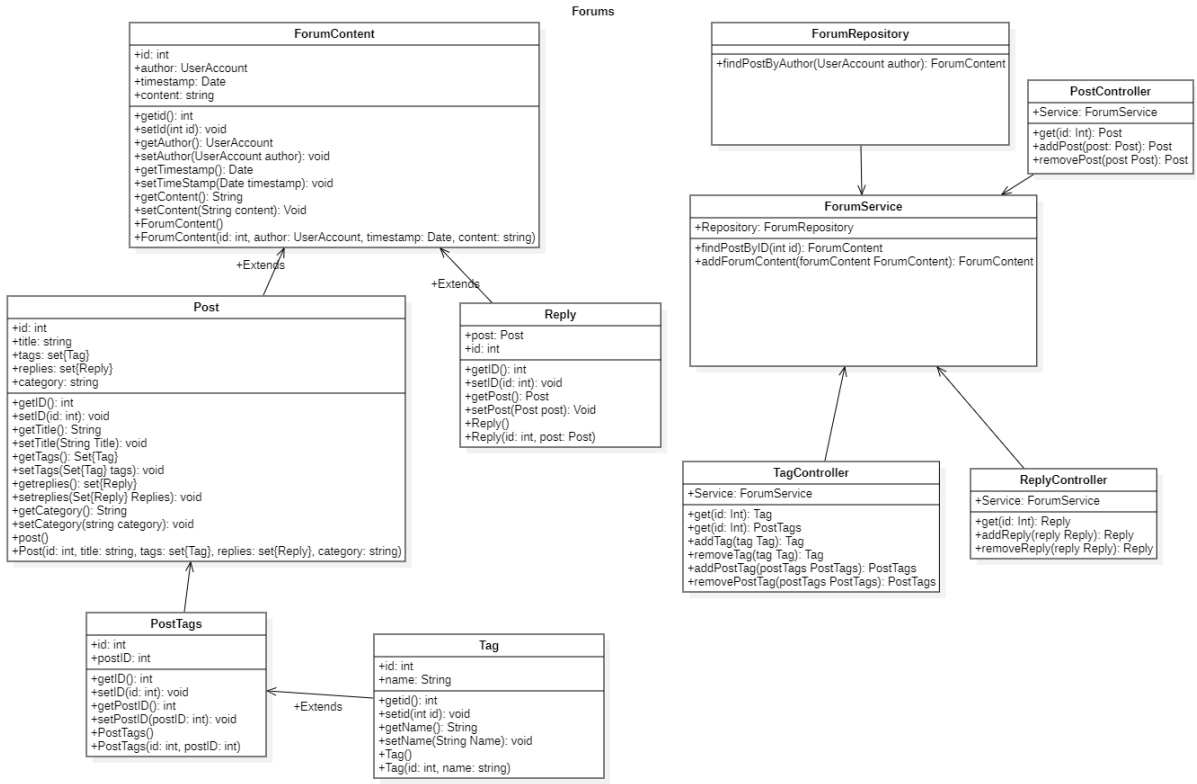

Figure 6: Class Diagram #2 for WILMA System

System Architecture

WILMA is divided into several modules including:

- **Config**: Configuration files responsible for error handling, security, authentication, and mail handling.
- **Entity**: Includes model classes and data transfer object
- **Service**: Application layer business logic service classes
- **Repository**: Repository interfaces extend JPA repository interface and provide methods related to persistence. Only services interact with repositories
- **Web**: Web controllers, RESTful API controllers, connection settings

See the architectural diagram in **Figure 7** below.

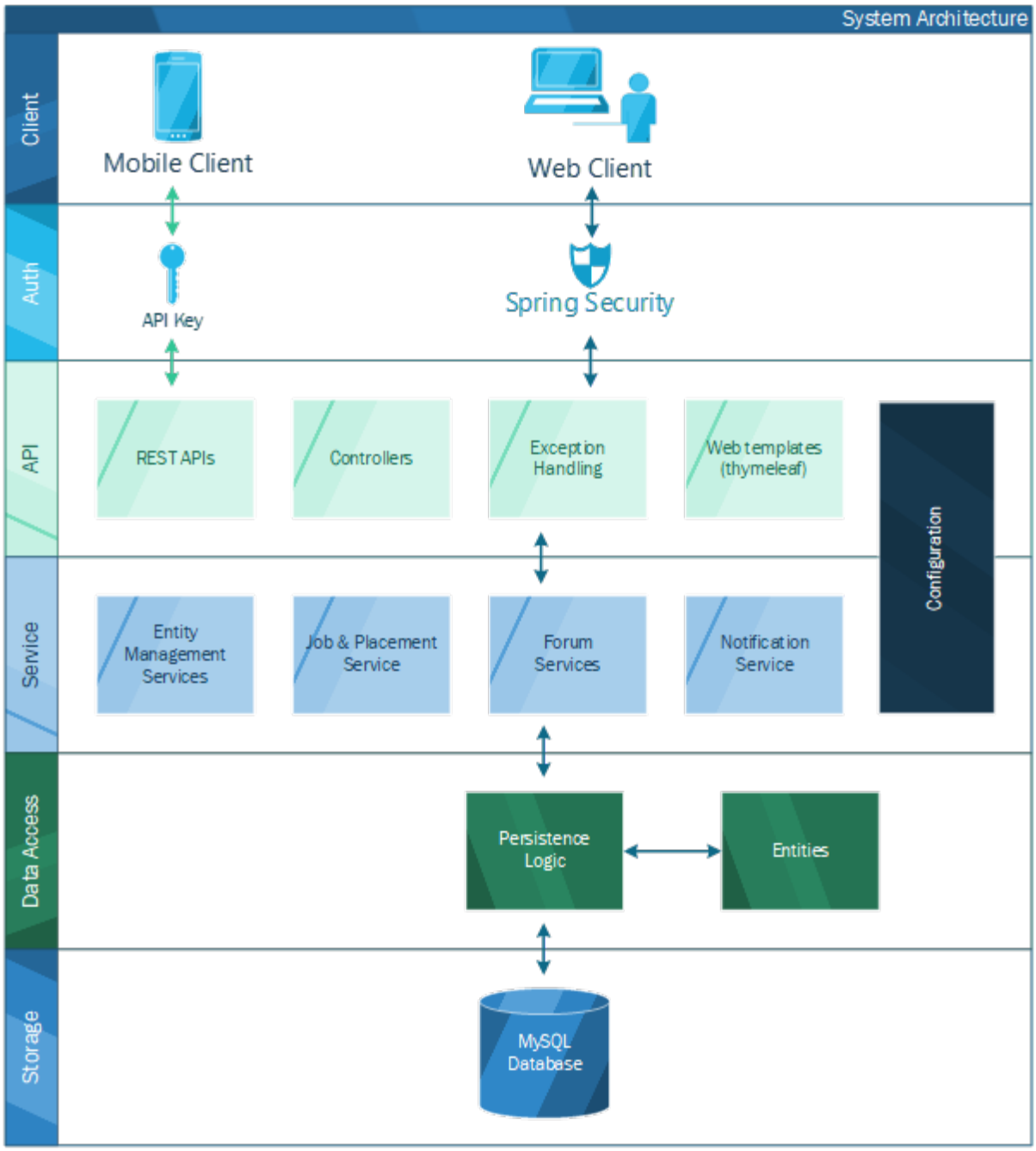

Figure 7: System Architecture for the WILMA system

Design Pattern

WILMA follows the Model-View-Controller (MVC) design pattern. **Figure 8** below depicts how the components of the system interact.

The View consists of user facing components such as Thymeleaf Templates and external UI components (e.g., optional native mobile user interfaces). View components have no reference to the Model components; interaction is passed between the View components and their relevant Controller components.

Controller components mediate interaction between the UI and the data model. These components include Web Controllers for handling web pages, redirects, etc., and REST Controllers for returning

JSON responses. Controllers interact with the Model through injected references to Services in the Model.
The Model consists of Service, Entity, and Repository components. Entities are the data structures of the Model. Controllers interact with these Entities through Service components within the Model. Repositories interact directly with the database, and Services mediate interaction between the Controller components and the Repositories.

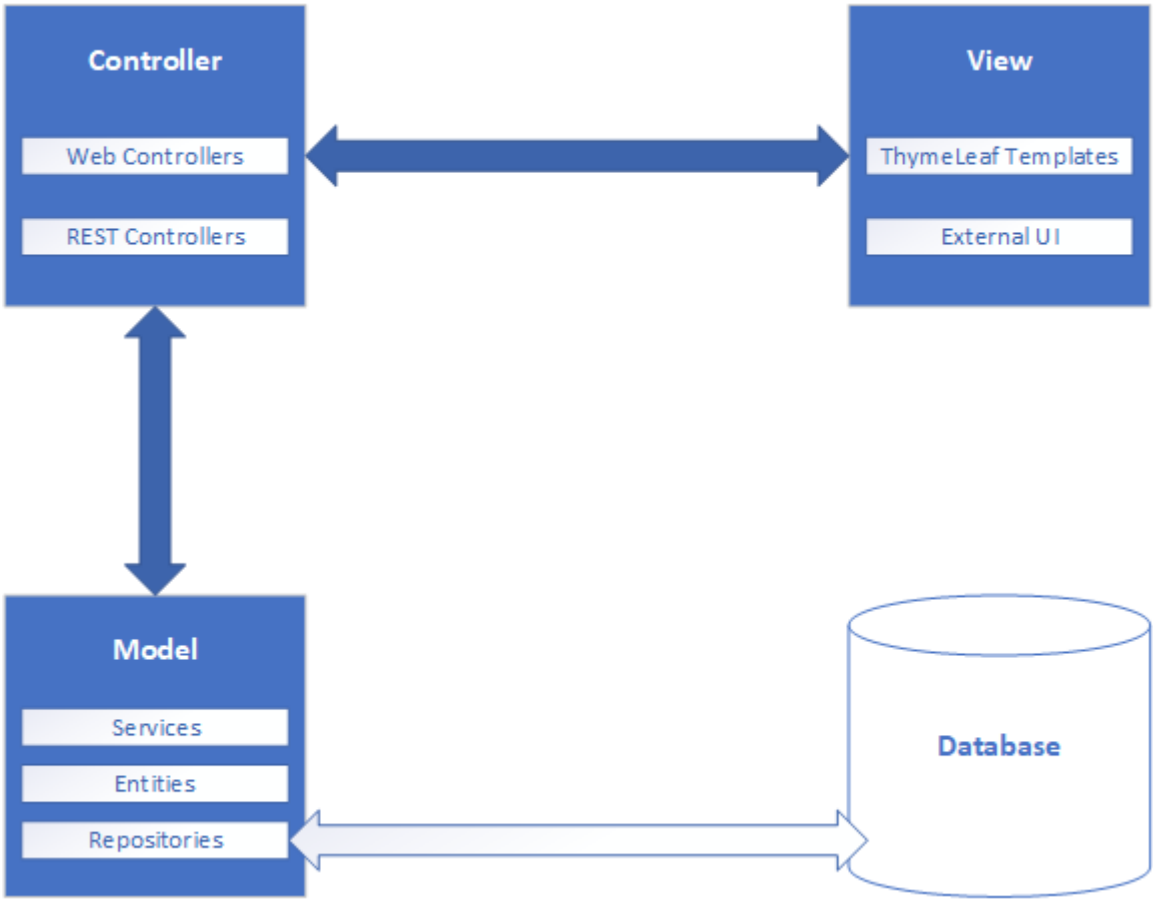

Figure 8: MVC design pattern implemented in the WILMA project

Entity Relationship Diagrams

Below (**Figure 9**) is an updated E-R Diagram depicting relationships between the various entities:

Figure 9: ER Diagram of WILMA system

User Interface Design

Below (**Figures 10-18**) are the updated UI captures of a variety of pages in the WILMA system.

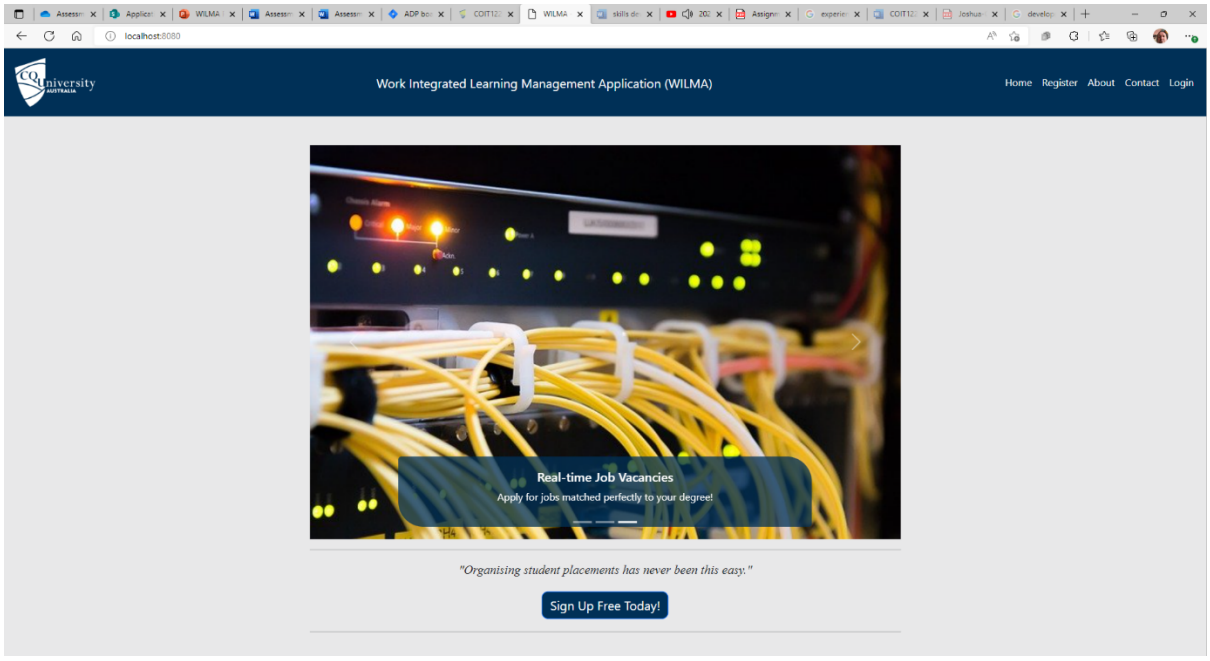

Figure 10: Home page

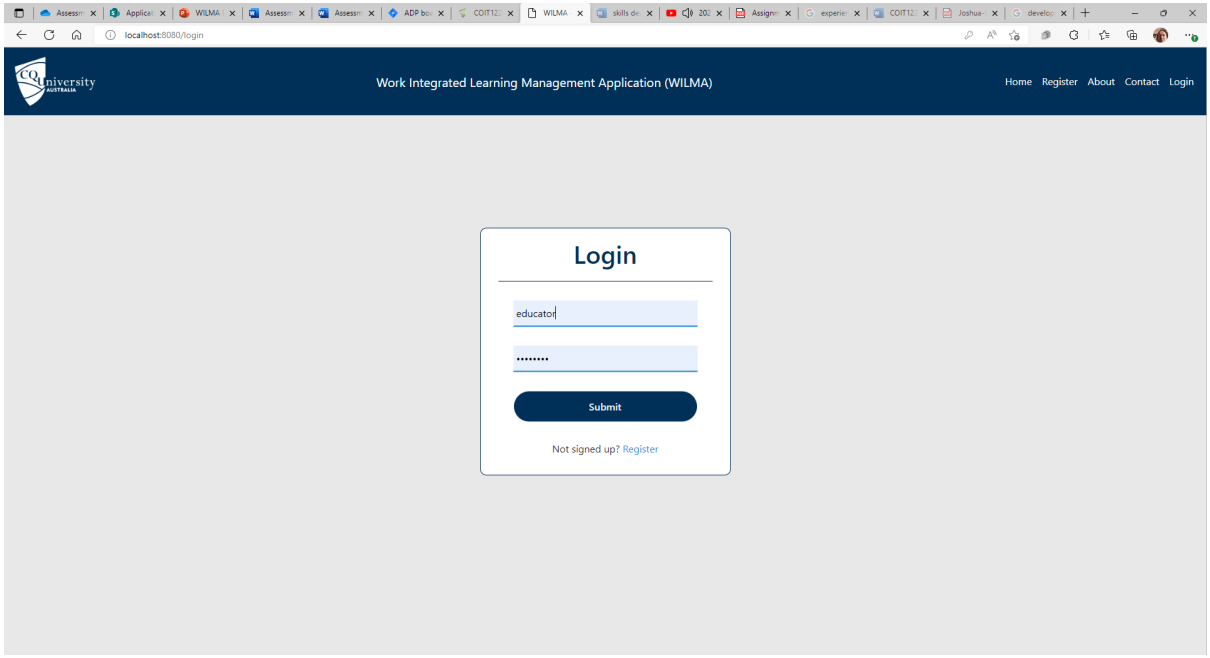

Figure 11: Login page

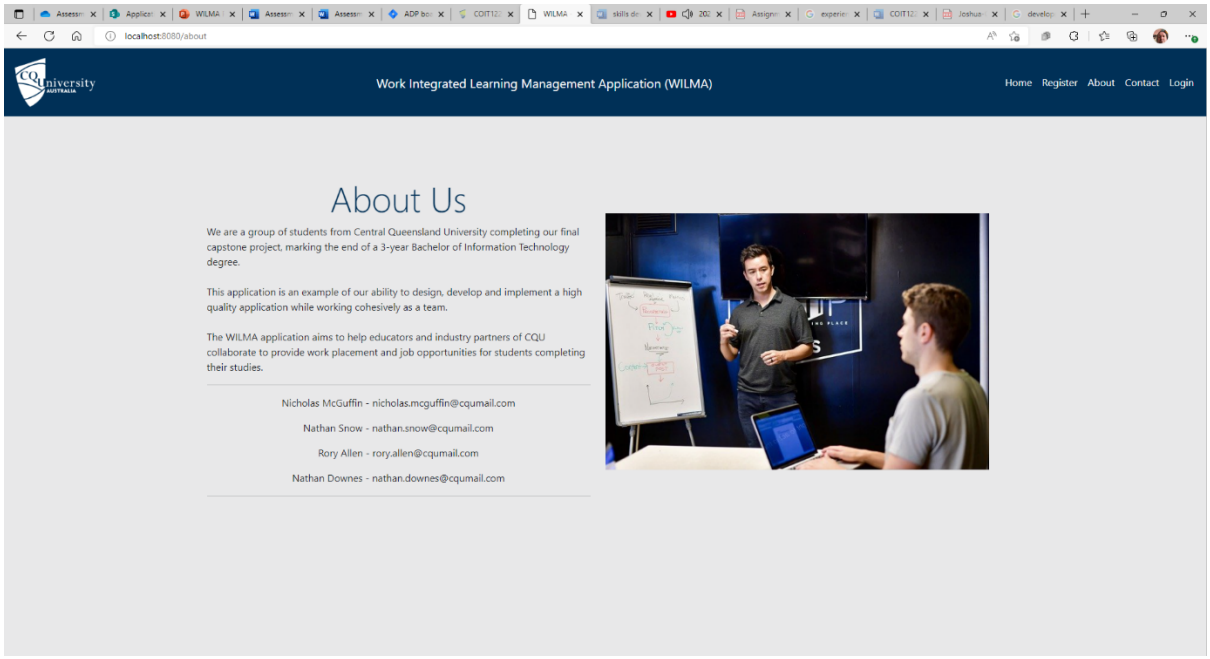

Figure 12: About page

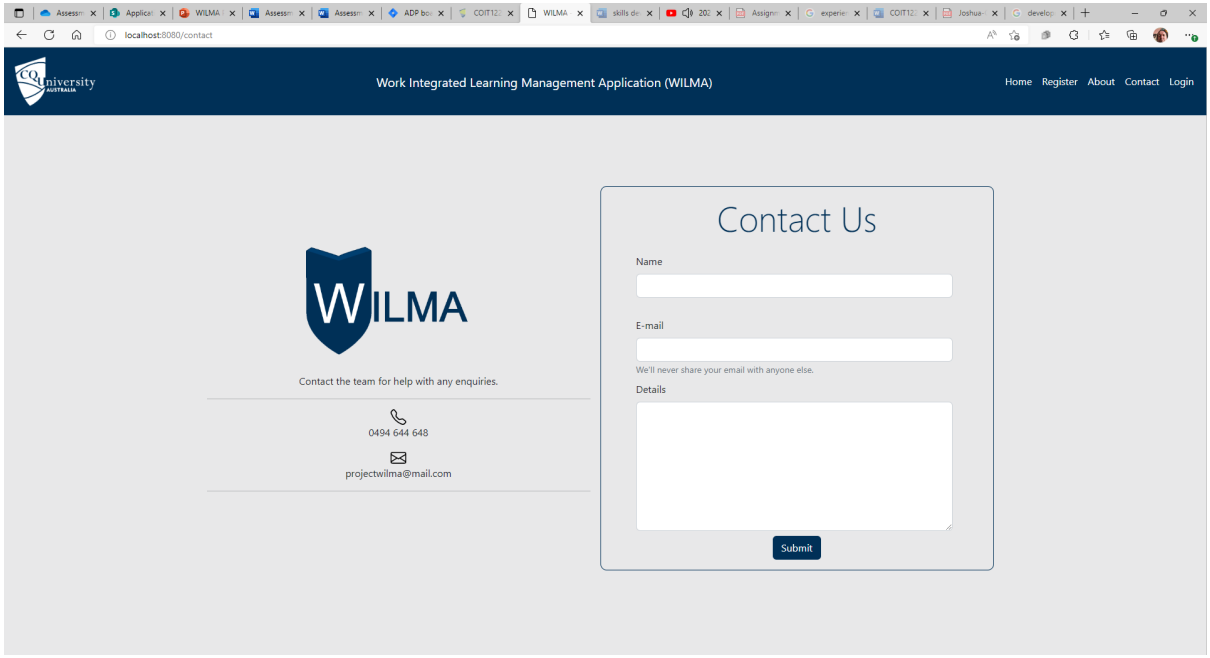

Figure 13: Contact page

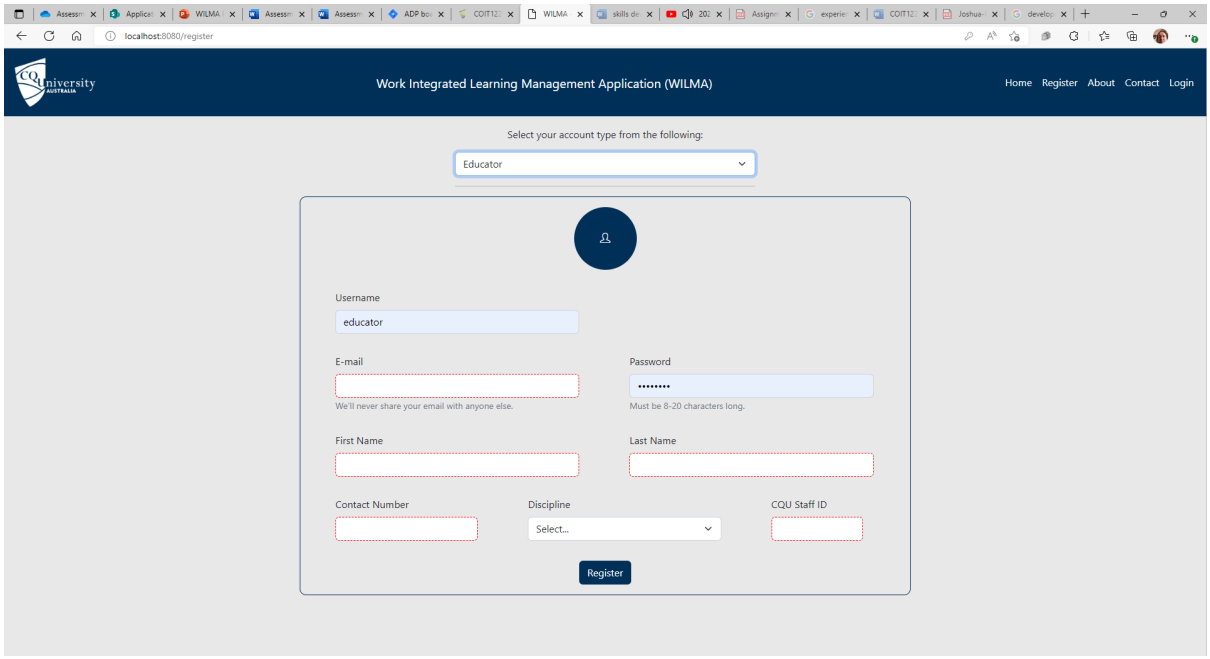

Figure 14: Registration page

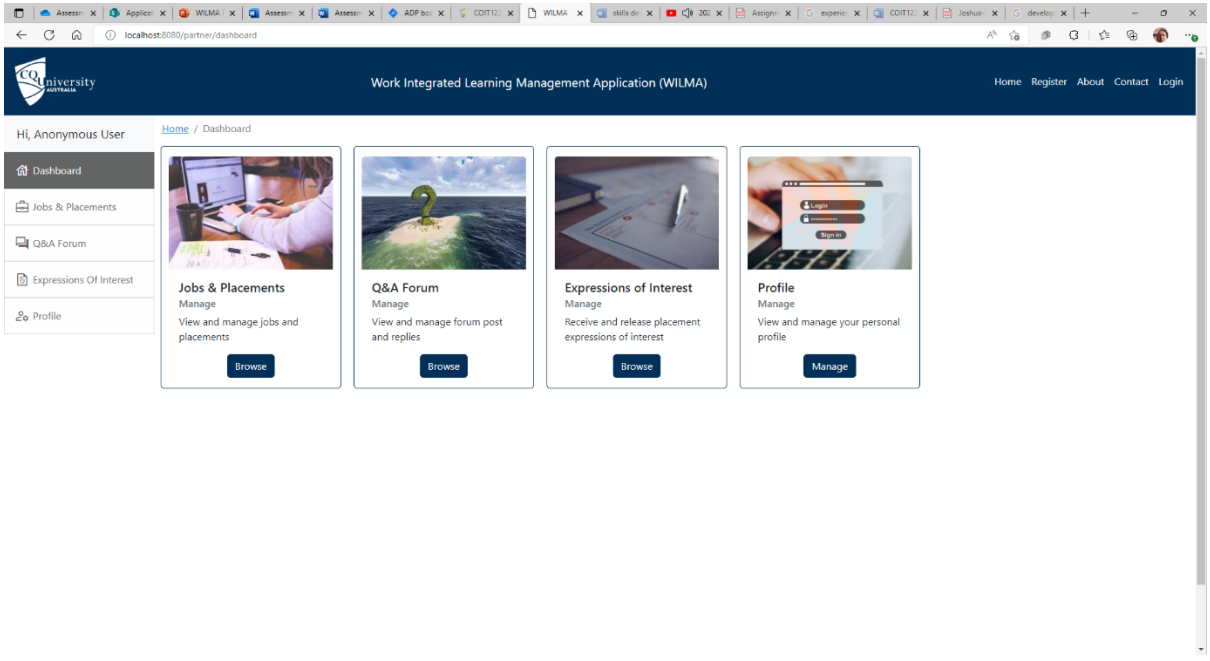

Figure 15: Partner dashboard

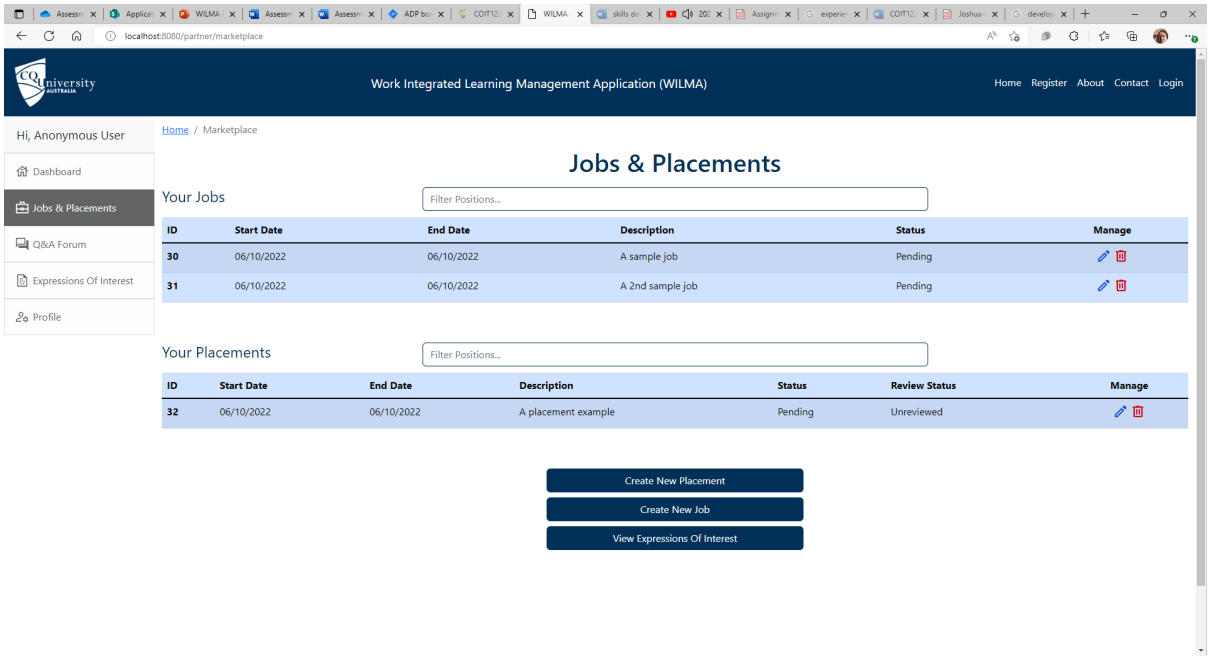

Figure 16: Marketplace

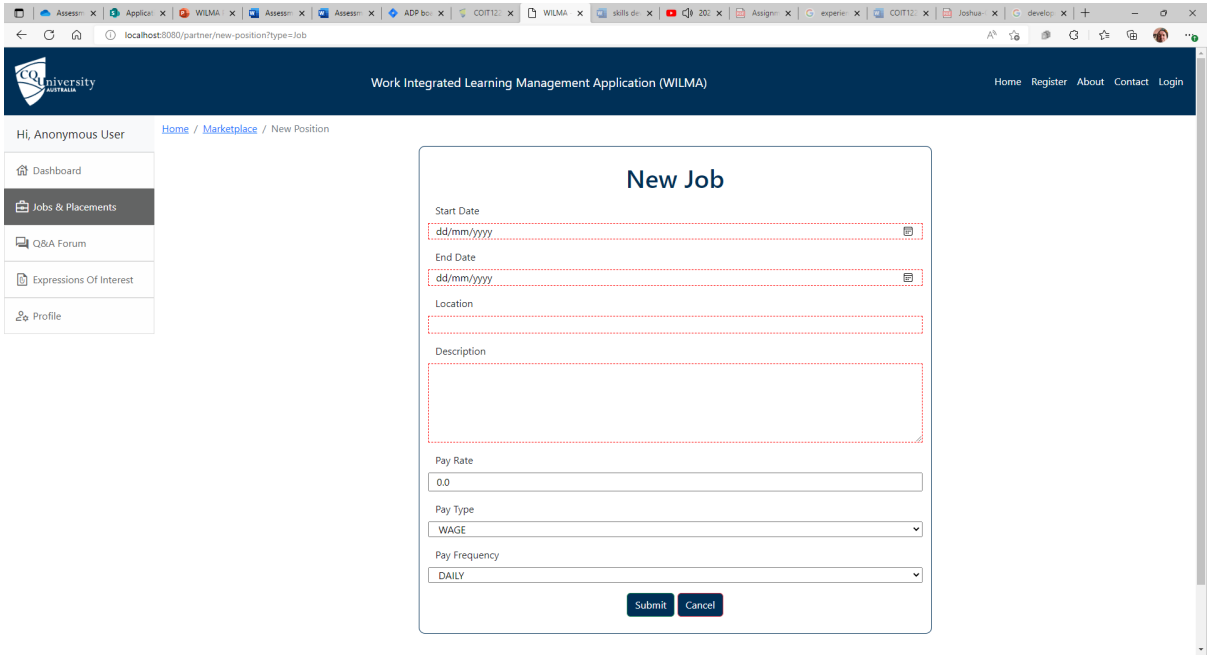

Figure 17: Create Job page

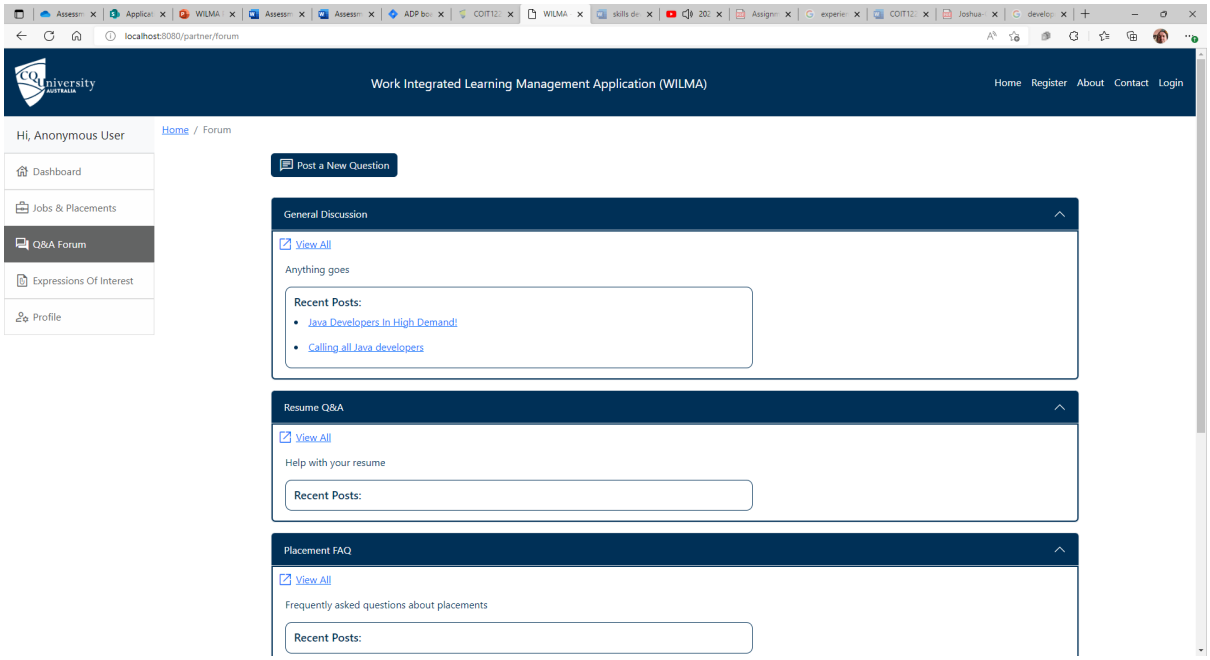

Figure 18: Forum page

Behavioural Models

Figures 19 and **20** below depict sequence diagrams for create and read processes. These sequences remain unchanged since the design document specifications and are reflective of all of the CRUD operations used across the WILMA system.

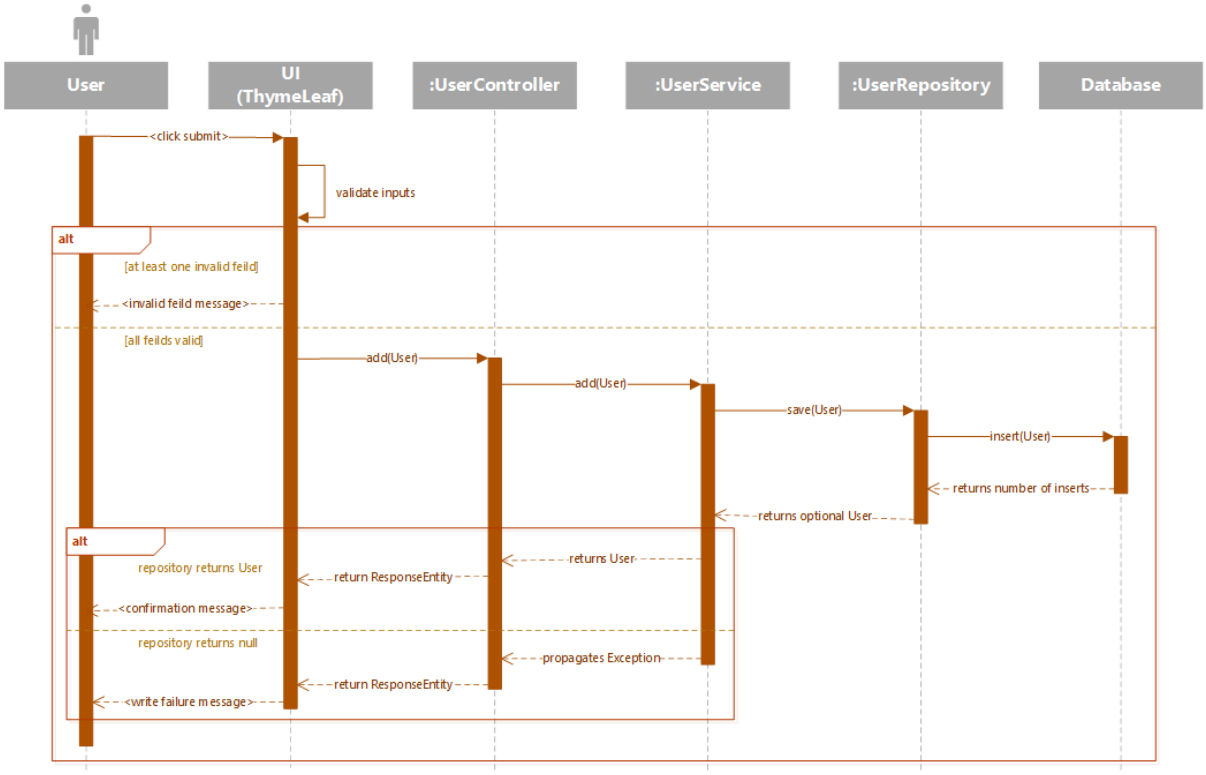

Figure 19: Sequence Diagram depicting new user creation process

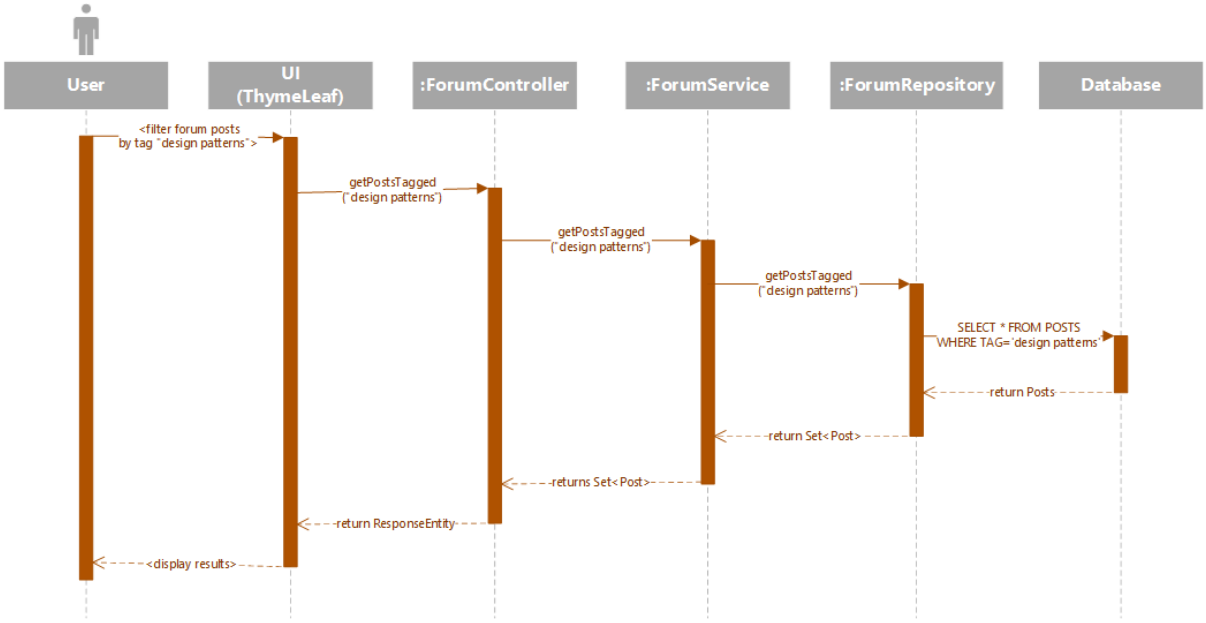

Figure 20: Sequence Diagram depicting filtering forum entries by tag process

Project Issues

A workflow issue that exists in the current iteration of the software is that expired positions are not automatically closed. Currently, placements that have been completed become available for partners to render their verdict of the performance of the placed student. Once they have submitted their feedback, educators can review the feedback, but need to manually close the position. This could be automated in a future update.

Another workflow issue is that of the creation of placements from EOI's. In its current state, partners that register their interest in an EOI open a new placement creation form. This form is prepopulated with data from the EOI, and the EOI's id number and category are prefixed to the placement's description. This is a 'stringly' solution, which works in practice in that it flags the pending placement as one created from an EOI to any educator that approves it. But this method is far from ideal, as it then requires the educator to go into the EOI listings and manually mark it as filled. It is proposed that in a future update to the system, an optional EOIid attribute be included in placement entities, so that if they are approved, the system can automatically mark its associated EOI as filled.

One known security issue is the fact that elevated roles are not validated. User roles are assigned at registration simply by selecting the user type from a dropdown. It is possible that someone could register as an Educator and be able to use the system with their elevated role, allowing them to edit and delete positions. This could be circumvented in a later release by validating their email address against known email domains of CQU staff.

User Manual

Please note

If you are fetching this project from the project GitHub repository, you will need to change the app password for the email configuration in two places, as the password currently listed was compromised and subsequently updated. You can find this application.yml file located in in the web/src/main/resources directory.

```
site-email: wilmaproject.dev@gmail.com
host: smtp.gmail.com
port: 587
username: wilmaproject.dev@gmail.com
password: cthzgagaedryycze # this password won't wonk-->   bditlkfcuotzojpu
properties:
  mail:
    smtp:
      connectiontimeout: 5000
      timeout: 5000
      writetimeout: 5000
```

This user manual will cover project setup and running the application in either an IDE or simply via the command line interface (CLI) in both Windows and Linux operating systems (this app has also been tested and developed on MacOS).
Basic use and creating a new user account will then be demonstrated, as well as providing login details for three basic demo user accounts that can be used to get familiar with the system and each of its user roles and authorizations.
Important credentials and configuration settings will then be listed and discussed, to help with future configuration changes and/or migrations.
The final WILMA project will come bundled with the following deliverable components:

- User manual (this document)
- Project report
- Project source code (including module .jar files)
- Source code documentation
- Access to the project GitHub repository

Installation Instructions

This application is not running on a live production server, as such it needs to be run locally via the following simple steps:

1. Download, save, and extract the contents of the zip file that either (1) came with this project, or (2) was downloaded from the project GitHub repository (please make sure to download the latest release).
   Please note that if you are using a Linux based machine, the tar.gz file is available via the release on the project GitHub repository.
2. Run the application locally on your machine via either one of the following methods:
   a. Run the program via your IDE
   Open the project in your favourite IDE and run the main class (web/src/main/java/com/wilma/web/App.java) to start the embedded Tomcat web server. We recommend using IntelliJ IDEA or VSCode, but most IDEs should work fine.
   b. Via the CLI
   In your terminal/console, navigate to the project root directory (you will know you are in the right place when you see "at least" all of the directories shown below.

```
config
entity
repository
service
sql
web
```

Once in the project root, run the following command in your terminal `java -jar web/target/web-0.0.1.jar` and press enter. You should now see the Spring Boot application starting in your terminal like you can see in the screenshot below. To kill the application process, simply press `Ctrl + C` in your terminal window.

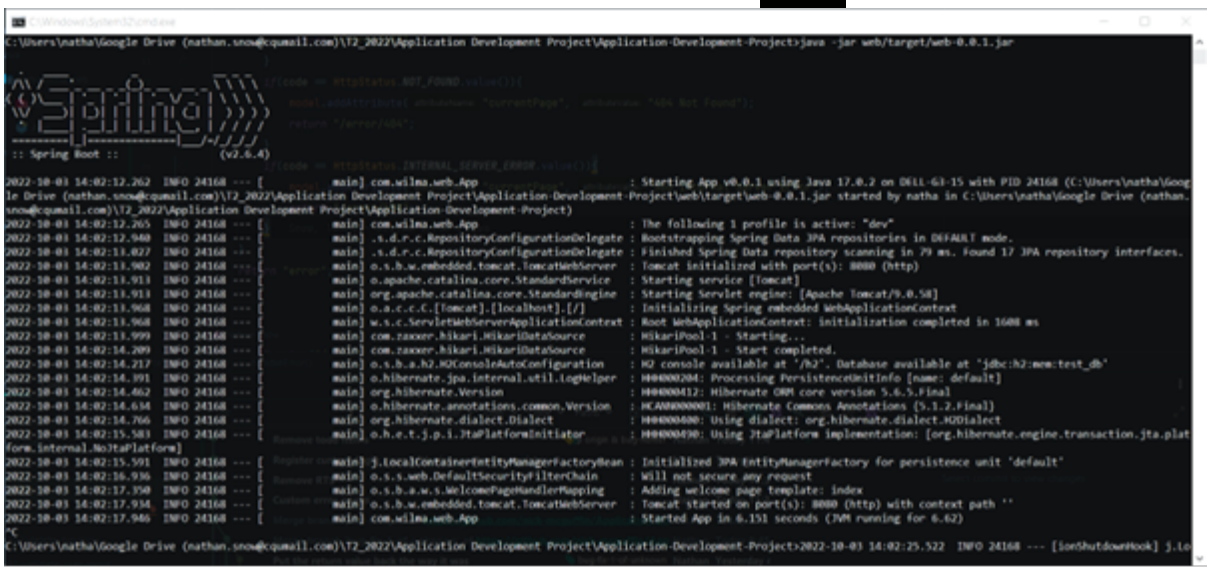

Figure 1: Running the application in the windows command prompt

Linux Terminal Only (No IDE or UI)

1. Save and extract the project tarball file into your chosen directory.

Note: You can run `curl -L https://github.com/nick-mcguffin/Application-Development-Project/tarball/development | tar -xz` directly in your terminal to download the tarball file, but you will need to compile the project first to build the required .jar files.

2. Change into the project root directory making sure that all the module directories are present.

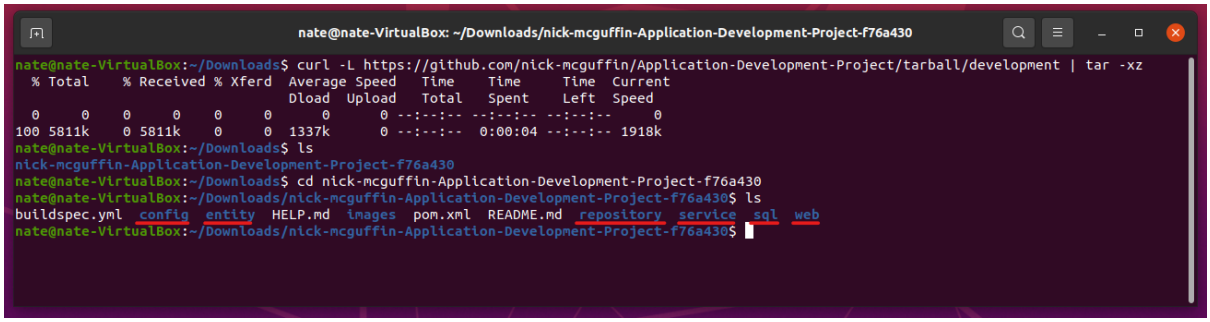

Figure 2: Opening the project directory in the Linux terminal

3. Run `java -jar web/target/web-0.0.1.jar`

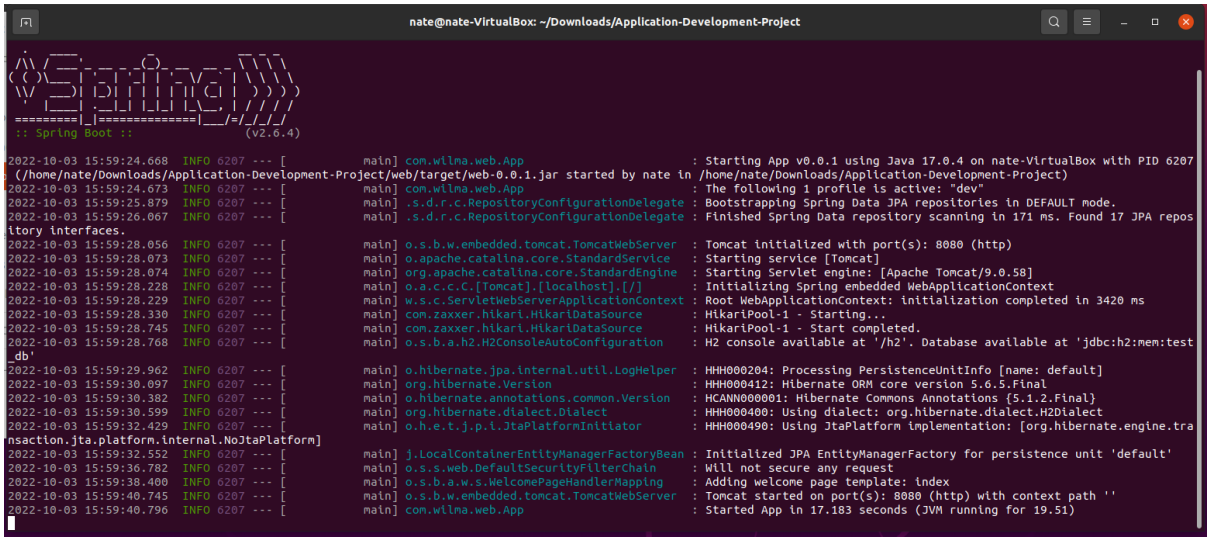

Figure 3: Running the app in the Linux terminal

You should now have the project running locally, and if you open your web browser and go to http://localhost:8080/ you should now see the WILMA home page.

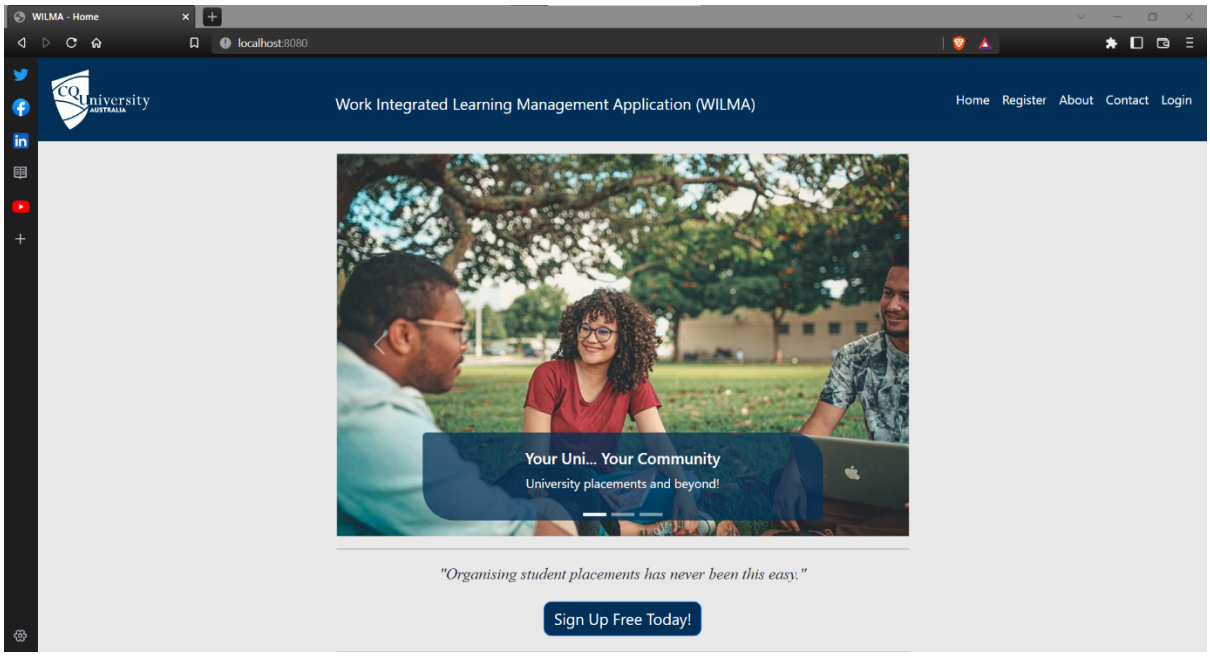

Figure 4: WILMA home page

Intro to the platform

You will notice that the web pages visible in the site menu (top right) are all openly accessible without the need for any authentication. This is intentional as it gives the public a way to visit the site and learn about its purpose or make any enquiries.

However, registered users (students, educators, and industry partners) can click the “login” link and enter their credentials to gain access to the members-only portal where they can access all appropriate areas including the WILMA marketplace (jobs and placements), the Q&A forum, manage their personal profile, and other features based on the user type and authorities.

Once a user has successfully logged in, they are directed to the dashboard for their role, for example students get routed to student/dashboard, partners go to partner/dashboard, and educators go to educator/dashboard. Users who attempt to access areas they are not authorised for, will be presented with a custom 403 – Forbidden page as shown below.

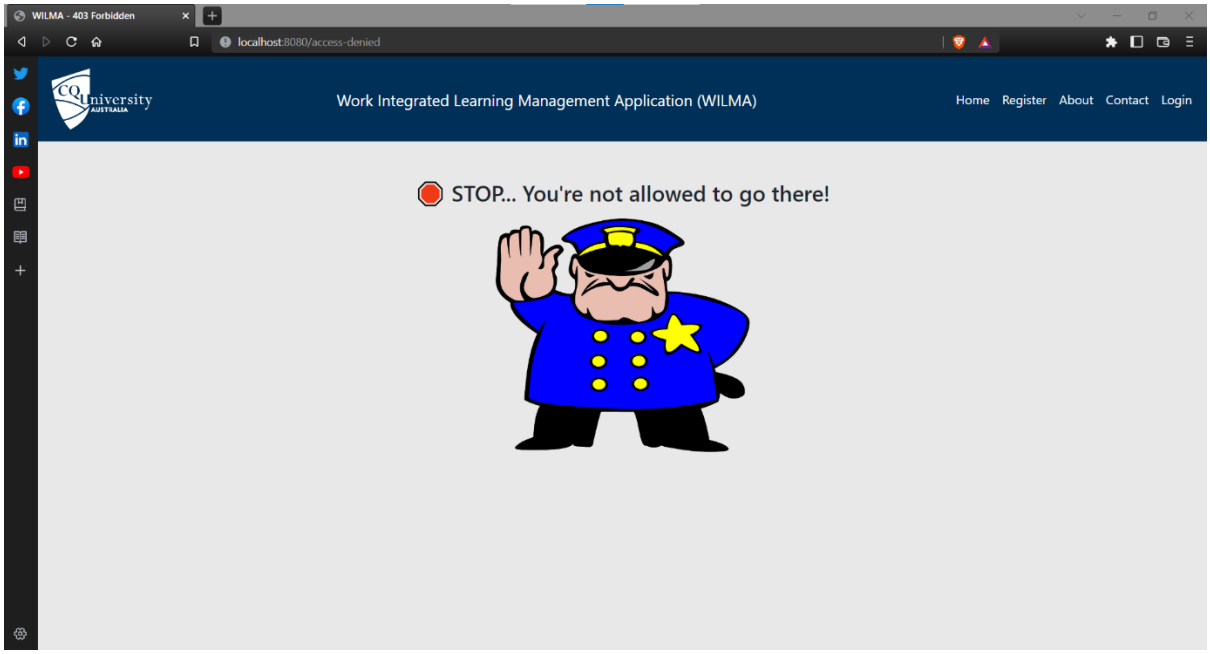

Figure 5: Custom access denied page (403 Forbidden)

In addition to blocking users from accessing areas they are not authorised for, the system will log a message indicating the user who tried to access the area, and what area it was.
The following is an example log message resulting from a student (with username "student") who has attempted to assess the endpoint http://localhost:8080/educator/marketplace at 4:36PM 03/10/2022.

Figure 6: Unauthorised access attempt log message

Setting up an account

The process of registering a new account is very straightforward.
We start by heading to the registration page via the "register" menu link, or we can also get there by clicking the "Register" link in the login form as shown below.

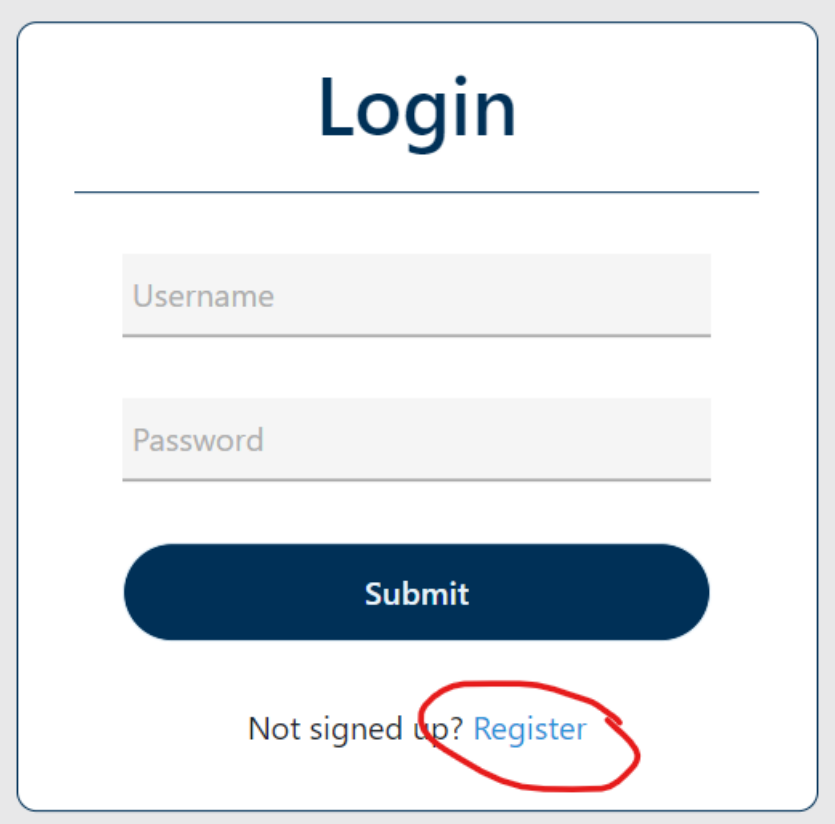

Figure 7: Users can access the registration page from the login form

Once at the registration page we select the type of user that applies to us, for our demo we'll choose "Student" and I'll use myself as the example.
After filling out the form details and clicking the "register" button, the system will verify that you in-fact have the email address given by sending an email with a verification link.

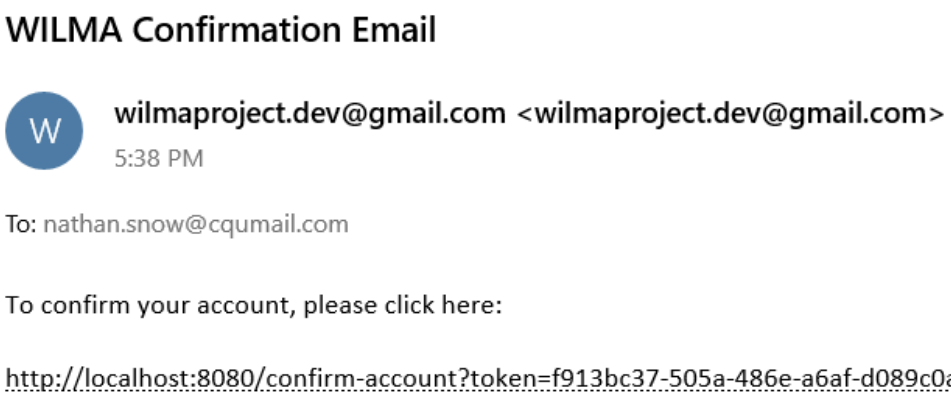

Figure 8: Verifying my email account to setup my WILMA account

Following the link will make a request to the system that verifies and activates your account and takes the user to a page informing them that their account is ready.

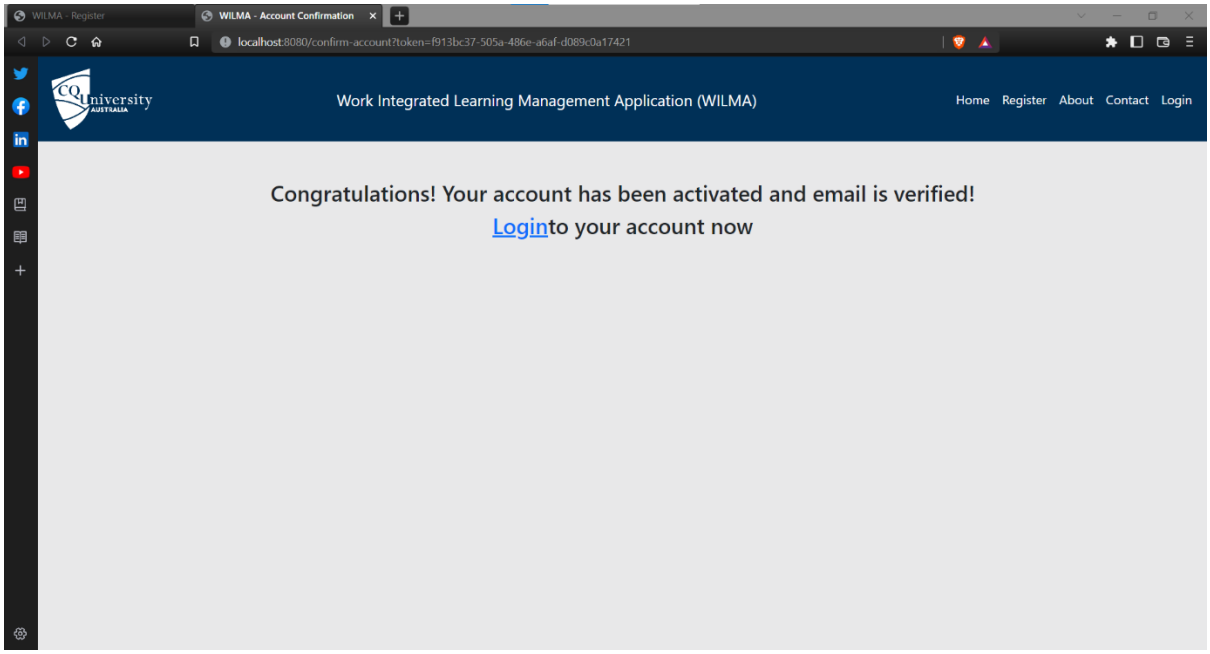

Figure 9: Account activation complete

The system makes a decision based on the user type and email domain given as to what type of user you are and therefore what default roles and access you should be granted.

Again, the system logs the login activity details so that it may be used to troubleshoot any resulting issues, and we can also see that the system determined my login target URL to be http://localhost:8080/student/dashboard based on my role of "student"

```
c.w.config.handlers.LoginSuccessHandler  : User "nathan22" (Id: 39) successfully authenticated
c.w.config.handlers.LoginSuccessHandler  : Target URL determined as /student/dashboard
```

Figure 10: Log showing successful login and role-based redirection

With my new student account, I can do things such as:

- View and apply for jobs and placements.
- Ask and reply to questions in the platforms integrated Q&A forum.
- Upload, download, and delete my personal files such as resumes and cover letters.
- Customise my profile details including my bio, profile picture, and expected graduation date.

Predefined Demo Users

To help you get started, we have predefined three simple demo users so you can login with their credentials and "test drive" the platform, but we encourage you to register your own account so that you may experience the complete registration experience for yourself.

USER TYPE	USERNAME	PASSWORD
STUDENT	student	student
EDUCATOR	educator	educator
INDUSTRY PARTNER	partner	partner

Figure 11: Predefined demo user accounts

Email account credentials

This application contains its own integrated email functionality, which is linked to the wilmaproject.dev@gmail.com Gmail account via Gmail's SMTP mail server enabling the application to send emails. The credentials and configuration for this functionality is as follows:

ITEM	VALUE	NOTES
EMAIL ADDRESS	wilmaproject.dev@gmail.com	Used to log in to the Google account
USERNAME	wilmaproject.dev@gmail.com	Used to log in to the Google account
PASSWORD	re9XBCi4b-tshUX	Used to log in to the Google account
APP PASSWORD	bditlkfcuotzojpu	Used to authenticate the app
SMTP HOST	smtp.gmail.com	SMTP mail server
SMTP PORT	587	Encrypted mail submission port
TLS	Yes	Requires a secure connection

Figure 12: Email account credentials and configuration

MySQL Database Credentials

The WILMA system is coupled to a live MySQL database hosted on AWS cloud infrastructure, the settings and credentials to access which are defined below.

ITEM	VALUE
DATABASE TYPE	MySQL
HOST	aussie-db.cjdfjxus4vpw.ap-southeast-2.rds.amazonaws.com
PORT	3306
SCHEMA NAME	wilma_system_db
USERNAME	wilma_db_user
PASSWORD	wilma_db_pa$$word

Figure 13: MySQL database credentials and configuration

It is worthwhile noting that this database is designed for development purposes, and while it's capacity is more than sufficient, the virtual machine hosting it is quite limited having just 1 vCPU and 1GB RAM.
This can, at times, cause a noticeable delay in the database transaction times, to the point where it can make the application appear briefly unresponsive.
In a live production scenario, this could be solved by (1) hosting the application on the same VM as the database (or at least in the same data centre), and (2) increasing the VM performance.

Accessing the REST API

The WILMA system is equipped with REST APIs that enable its functionalities to be remotely controlled. There are many endpoints for many purposes, a full list of which would be quite exhaustive.
The APIs comply with the OpenAPI V3.0 specification, all of which can be viewed and interacted with by visiting http://localhost:8080/swagger-ui/index.html, or you can view the JSON representation at http://localhost:8080/v3/api-docs .

Conclusion

The project to design and build the WILMA system was undertaken over 12 weeks. It began with a period of planning, in which the projects objectives and scope were determined, and generation of management, technical, and supporting process plans commenced. These were compiled into a project proposal and plan documentation, outlining the processes with which the project would be managed.

Following this the team discovered and defined the project's specifications, which would form the foundation for designing the product. During this phase, user requirements, system architecture, and functional and non-functional system requirements were developed. These along with system design documents, configuration management, and schedule and task allocation, were compiled into an extensive requirements specification and design document. This document would be closely followed during the development phase to follow.

Development of the product began with the team developing UI components to accommodate all use case workflows. This was followed by developing the data models and establishing the persistence layer. The application layer was then developed to connect the UI to the data access layer.

The final product meets all the clients needs as specified in the initial brief. It also extends several of the features outlined by the client to provide a more effective workflow.

Peer Review

Nathan Downes provided an adequate contribution to the group. They excelled in their willingness to aid other group members. They are not particularly proficient in their knowledge of the programming languages used in the project, but this (in part) could be attributed to the fact that the team were employing a framework that they had little to no experience using. Another weakness was their inability to align with the team's project management processes. On several occasions they were working on tasks that were assigned to other users, while not delivering on tasks that were assigned to them. Overall though they were very happy to help as best they could.

Nathan Snow provided an enormous amount of work to this project. They were responsible for setting up and configuring the initial code base and were instrumental in delivering the final product. They have very evident technical skill and exceptional debugging and testing skills. One weakness was their intolerance for some behaviours of other team members. It seamed as though, as they were the most technically proficient team member, other team members relied upon them to excess, and this caused them some frustration. In retrospect it may have been more productive to allow Nathan to solely aid the rest of the team and relieve them from development tasks entirely.

Nicholas McGuffin provided an exceptional contribution the project. Their efforts in developing the UI and styling it are what resulted in WILMA being such a beautiful and usable piece of software. Beyond this, they invested considerable time in managing the project over the course of development. Additionally, they were always in contact with all other team members keeping them focussed and making sure they were okay. I have worked with Nick on other projects, but in this one I cannot fault their performance.

Personal Experience

Personally, I thoroughly enjoyed collaborating on this project. This is the largest and most complicated development project I have worked on in my university career. It has been great to see how all the things I have learned over this course come together to produce a fantastic piece of software.

In terms of the SFIA skills matrix that we explored at the beginning of the project; I believe I have achieved all the goals that I set out to complete. Early in the planning phase, my contributions required an extensive amount of research (RSCH) into both planning methodologies and development standards. The requirements definition and management skills (REQM) were put into practice through both the planning and requirements gathering phases of the project. Software design (SWDN), database design (DBDS), and data modelling and design (DTAN) all formed the basis of the design phase and were essential in the development of the design documentation. User experience evaluation (USEV) was used both in the initial development of UI wireframes for the design specification, and in the iterative development of the UI and workflows.

The knowledge domains of the systems integration and build (SINT) and configuration management (CFMG) skills were both used in developing the technical process plan, infrastructure plan, and configurations management plans in the early stages of the project. They were utilised again during development in understanding the various configuration components used in Spring MVC (and other JPA-centric) projects.

Programming/software development (PROG) was the heart of the development phase. This allowed us to translate the specification and design documentation into organised, efficient code and ultimately a deployable product (PBMG). Testing (TEST) is the only skill that has any questions surrounding it. Although we did employ rigorous user testing, unit and integration testing did not

always utilise best practices such as automated testing procedures. That said, user testing was integrated into every sprint during the review process, so we were confident that the product would perform as expected. In future I would like to thoroughly explore automated testing frameworks.