Item	Criteria	Marks allocated	Marks Awarded
1	Project Summary: Clear and concise summary of project scope, objectives, assumptions, project deliverables, schedule, and budget summary.	10	6
2	Clearly written project organization providing details of any external stakeholders, team member details, roles, and responsibilities, communication process and format	10	8
3	Managerial process plan: properly identified risks and risk mitigation strategies, Gantt chart with tasks, responsible person, milestones, and any task dependencies, meeting schedules, project control, and conflict resolution plan	20	16
4	Technical Process Plan: Clearly defined development methods, tools, and techniques including development methodologies, programming languages, design tools, version control, merging of components	20	20
5	Infrastructure Plan: Clearly written software and hardware requirements for development and using the software application	10	10
6	Supporting Process Plans: Clearly written configuration management plan, verification and validation plan, documentation plan listing, and describing the documents to be created.	10	10
7	Quality Assurance Plan: Appropriately defined quality assurance plan and problem resolution plan	10	10
8	Report Format and Structure: Grammatically correct, correctly structured sentences, document following good formatting, and correct report structure.	10	8
9	Subtotal	100	88
10	Convert to out of 15	15	13.2
11	Penalties (Late penalty, plagiarism, etc.,)		
12	**Final total**	15	13.2

2022

COIT13230

APPLICATION AND SOFTWARE DEVELOPMENT CAPSTONE PROJECT

ASSIGNMENT 1 – PROJECT PROPOSAL AND PLAN

DUE DATE: 29/07/2022

WORK INTEGRATED LEARNING MANAGEMENT APPLICATION (WILMA)

NICHOLAS MCGUFFIN - 12115435

NATHAN DOWNES - 12046570

RORY ALLEN - 12149026

NATHAN SNOW - 12060962

Contents

Tables

Figures

A. Project Summary√

The following summary briefly examines the aim, scope, objectives and constraints of a new and innovative Work Integrated Learning (WIL) application. Further, the expected deliverables, schedule, and a mock budget summary for the project are outlined for perusal by our Central Queensland University clients.

i. Aim

This project, aptly named the Work Integrated Learning Management Application (WILMA), aims to create an innovative web-based software application that acts as the central hub for placement providers, educators, and students to effectively communicate and collaborate on work placement projects.√

ii. Scope

The goals and requirements of the WILMA project are extensive. As such, the project scope can be broken down into multiple parts to better clarify the expectations of each stage of development. The stages are as follows:

1. Project Proposal and Project Plan

This section of the project aims to develop a proposal which identifies key management information and generates supporting documents that can be referred to throughout the entirety of the development process. Some of the processes it will examine include but are not limited to:

- Project overview
- Document planning and review processes
- Internal and external stakeholder roles and responsibilities
- Risk management, task completion, project control and conflict resolution procedures
- Technical process plan including tools, methods and techniques used to undertake scheduled project tasks.
- Infrastructure plan that details required hardware and software.
- Configuration and review management
- Document planning
- Quality assurance planning
- Resource scheduling
- Identifying and scoping project details

2. Project Requirements Specification and Design

The scope of this section focuses on analysing the system requirements of the project and using this information to design a solution that adheres to a professional standard. This will allow systematic progression throughout the entirety of the project as the system's goals are well understood, and the underlying system design is well documented. Some of the important scope features in this section of the project include:

- System requirement specification

    - Schedule and task allocation
    - User requirements
    - System architecture
    - Functional and non-functional system requirements
- Design model documentation
    - Chosen design pattern
    - Use case diagrams
    - User interface design
    - Sequence diagrams
    - UML class diagrams
    - ER diagrams
- Configuration management plan outlining updated development processes and plans

3. Final Project Delivery

The delivery of the final project includes both a complete and functioning application and its respective documentation. The enclosed information will explain the implementation, testing, and an in-depth user manual. The scope of this section is inclusive of the following:

- Implementation and testing documentation
    - User Interface implementation
    - Application Layer implementation
    - Data Access Layer implementation
    - Non-functional requirement implementation
    - Version Control procedures
    - User manual to explain installation and operational instructions
- Software development process review.
- Project Portfolio
    - Overview and business case descriptions
    - Project requirements explanation
    - Project design information
    - Test plan and supporting evidence
    - Peer review of team

4. Final Project Presentation

A brief presentation that is delivered to stakeholders upon completion of the project to give a brief overview of the application and its use cases, design components, implementation, and testing. A working example of the software will also be included to display the final iteration of the WILMA project and allow audience members to ask any questions regarding the software.

iii. Objectives

The objectives of the project define the major expected outcomes of the project. The fundamental goals that define the WILMA project include, but are not limited to:

- Understanding the requirements of a system that fully integrates the needs of Central Queensland University (CQU) lecturers, students, and industry partners in four disciplines - software development, network, business analysis and project management.
- Maintaining regular communication with stakeholders to adapt to project requirements that may potentially change throughout the length of the development process.
- Designing a web-based application that simplifies and streamlines the current WIL procedures.√
- Designing the platform to connect with an integrated server-side UI or a distributed external client-side UI (including mobile) via an OpenAPI compliant REST API.√
- Designing the application to be modular and scalable, allowing for simple additions to the product in future applications if required.√
- Ensuring the application is highly secure by implementing role-based authentication and authorisation applicable to each user.√
- Implementing an application that is beneficial to all stakeholders and successfully increases the efficiency of the WIL program. √
- Implementing features conducive to the above objective as follows:
    - Businesses can register their interest in work placement and add job or internship opportunities to a job 'board' for perusal by students and educators
    - Educators can moderate the job board and share vacancies with interested parties.
    - Students can view job vacancies and register their interests in certain disciplines.
    - Students can submit their resumes to the system, allowing educators and industry partners to view them and potentially recommend or offer them job opportunities.
    - The integration of a forum to give students an avenue to communicate professionally with potential employers and to post questions and answers.
- Testing the system and ensuring a fully functional application that meets user requirements is delivered at the end of the project.√
- Offering the client (CQU) a software maintenance plan for future modifications and system upkeep costs beyond the initial terms of the project.

iv. Assumptions and Constraints

The development of any large-scale project is met by constraints that can potentially limit the viability of certain operational choices. In the case of the WILMA project, budget, time, and resource constraints are apparent. As a group of students in our final year of study our budget is non-existent. This nullifies our capacity to hire more staff if required, purchase high quality software, hardware, or services, and forces the team to adhere to stringent time and resource constraints. Due to the nature of the project and its looming deadline, the team must adhere to strict time deadlines. As the WILMA project is underfunded and of a voluntary nature, team members have had to maintain other

work commitments - a factor which further increases the likelihood of time constraints on project deliverables. √

It has been assumed that the clients' resource requirements are compatible with a budget-free project. For example, the hardware, software, and resource needs of the customer can be guaranteed by their existing or newly purchased equipment. Furthermore, it is assumed that the time restraints imposed on us by the project deadline allow for the sufficient completion of project deliverables to a satisfactory level.√

v. Project Deliverables √

The client will periodically receive project deliverables throughout the duration of the WILMA project. The project schedule has been broken into a series of milestones that define the distinct aspects of the development process. The deliverables for this application are shown in Table 1.

Project Deliverable Name	**Included Deliverables**	**Expected Delivery Date**	**Delivery Method**
Project Proposal and Project Plan	1 x Report Document	29/07/2022	Email and Online Submission. Hard copy on request.
Project Requirements Specification and Design	1 x Report Document	29/08/2022	Email and Online Submission. Hard copy on request.
Intermittent Project Progress Reports	1 x Progress Report 1 x Progress Report 1 x Progress Report 1 x Progress Report	12/08/2022 26/08/2022 16/09/2022 30/09/2022	Email and Online Submissions. Hard copies on request.
Final Project – Software & Design, Testing, User Manual and Project Portfolio	1 x Source Code Files 1 x Configuration and Data Files 1 x Design, Testing and User Manual	07/10/2022	Email and Online Submission. Hard copy on request.
Final Project - Presentation	1 x Power Point Presentation and relevant information.	Week of 10/10/2022	Face to face delivery

Table 1: Project Deliverables

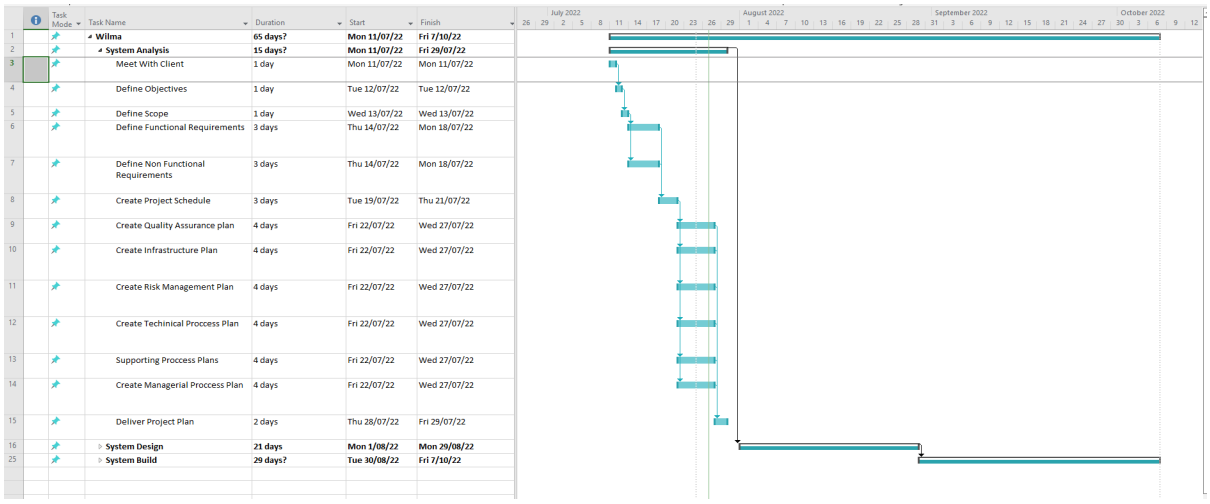
vi. Schedule and Budget Summary

The project schedule outlines the initial expected timeline of the development process and its milestones, including task interactions and dependencies. As the project is being developed using an Agile approach to project management and software development, it is possible for the schedule to experience slight deviations as system requirements shift and procedures are developed further. The following schedule depicts our initial schedule expectations:

Mode	Task Name	Duration	Start	Finish	Predecessors	Names
	Wilma	**65 days?**	**Mon 11/07/22**	**Fri 7/10/22**		
	System Analysis	**15 days?**	**Mon 11/07/22**	**Fri 29/07/22**		
	Meet With Client	1 day	Mon 11/07/22	Mon 11/07/22		
	Define Objectives	1 day	Tue 12/07/22	Tue 12/07/22	3	
	Define Scope	1 day	Wed 13/07/22	Wed 13/07/22	4	
	Define Functional Requirements	3 days	Thu 14/07/22	Mon 18/07/22	5	
	Define Non Functional Requirements	3 days	Thu 14/07/22	Mon 18/07/22	5	
	Create Project Schedule	3 days	Tue 19/07/22	Thu 21/07/22	6,7	
	Create Quality Assurance plan	4 days	Fri 22/07/22	Wed 27/07/22	8	
	Create Infrastructure Plan	4 days	Fri 22/07/22	Wed 27/07/22	8	
	Create Risk Management Plan	4 days	Fri 22/07/22	Wed 27/07/22	8	
	Create Techinical Proccess Plan	4 days	Fri 22/07/22	Wed 27/07/22	8	
	Supporting Proccess Plans	4 days	Fri 22/07/22	Wed 27/07/22	8	
	Create Managerial Proccess Plan	4 days	Fri 22/07/22	Wed 27/07/22	8	
	Deliver Project Plan	2 days	Thu 28/07/22	Fri 29/07/22	9,10,11,12,14,13	
	System Design	**21 days**	**Mon 1/08/22**	**Mon 29/08/22**	**2**	
	Class Design	7 days	Mon 1/08/22	Tue 9/08/22		
	Interface Design	5 days	Wed 10/08/22	Tue 16/08/22	17	
	Database Design	5 days	Wed 10/08/22	Tue 16/08/22	17	
	Build Test Plan	3 days	Wed 17/08/22	Fri 19/08/22	17,18,19	
	Review Design	2 days	Mon 22/08/22	Tue 23/08/22	17,18,19,20	
	Submit Progress Report 2A	3 days	Tue 9/08/22	Thu 11/08/22		
	Submit Progress Report 2B	3 days	Fri 12/08/22	Tue 16/08/22	22	
	Submit Project Requirements Specification and Design	4 days	Wed 24/08/22	Mon 29/08/22	21,23	
	System Build	**29 days?**	**Tue 30/08/22**	**Fri 7/10/22**	**16**	
	Build Database	6 days	Tue 30/08/22	Tue 6/09/22		
	Create Project Code	17 days	Wed 7/09/22	Thu 29/09/22	26	
	Build Testing	2 days	Fri 30/09/22	Mon 3/10/22	27	
	Submit Project Report 2C	3 days	Wed 14/09/22	Fri 16/09/22		
	Submit Project Report 2D	3 days	Wed 28/09/22	Fri 30/09/22		
	Release	1 day?	Tue 4/10/22	Tue 4/10/22	28	
	Final Report	4 days	Tue 4/10/22	Fri 7/10/22	28	

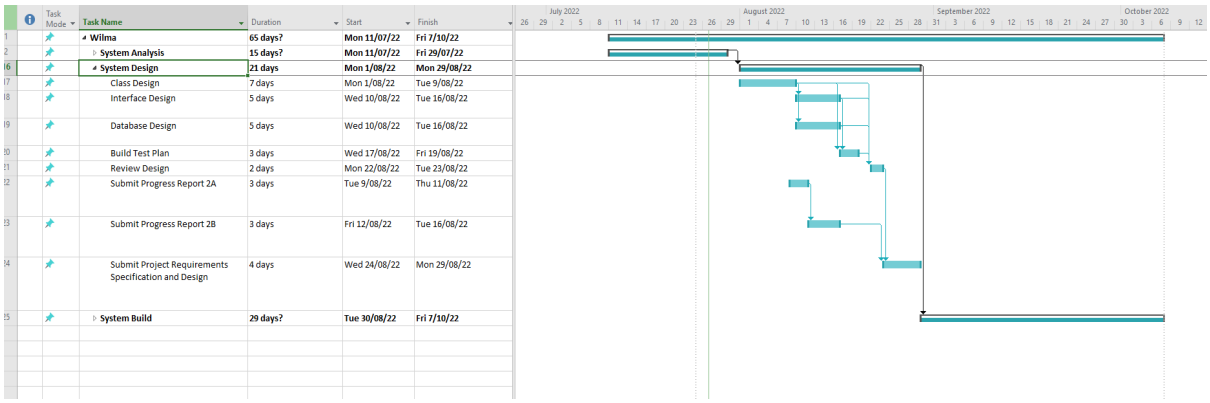
Figure 1: Gantt Task List

The task list shown in Figure 1 explains the tasks and allocated time in which the project will last. It delves deeper into the larger milestones of the project and displays the approximate dates that each task will be started and completed. The tasks aren't assigned to a specific person due to the agile nature of this project and the assumption that changes will be made frequently to the scope of the project. All of our team members are acting in the capacity of full-stack software engineers and can work on every aspect of the development process.√



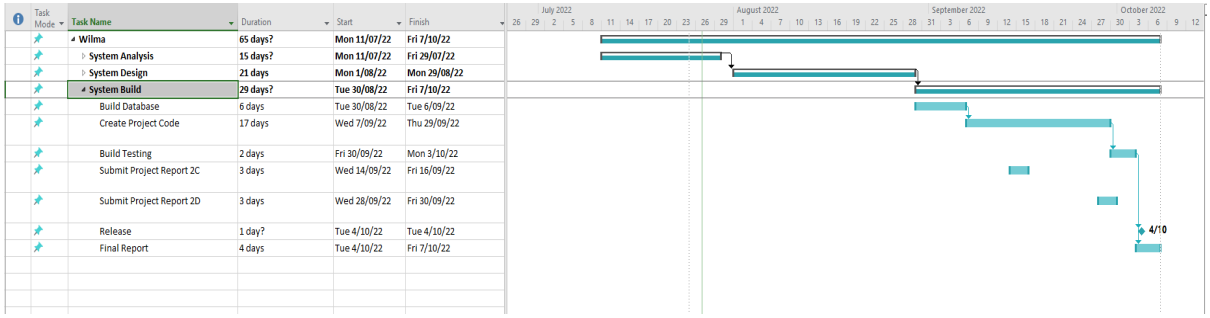
	Task Mode	Task Name	Duration	Start	Finish
1		**Wilma**	**65 days?**	**Mon 11/07/22**	**Fri 7/10/22**
2		**System Analysis**	**15 days?**	**Mon 11/07/22**	**Fri 29/07/22**
3		Meet With Client	1 day	Mon 11/07/22	Mon 11/07/22
4		Define Objectives	1 day	Tue 12/07/22	Tue 12/07/22
5		Define Scope	1 day	Wed 13/07/22	Wed 13/07/22
6		Define Functional Requirements	3 days	Thu 14/07/22	Mon 18/07/22
7		Define Non Functional Requirements	3 days	Thu 14/07/22	Mon 18/07/22
8		Create Project Schedule	3 days	Tue 19/07/22	Thu 21/07/22
9		Create Quality Assurance plan	4 days	Fri 22/07/22	Wed 27/07/22
10		Create Infrastructure Plan	4 days	Fri 22/07/22	Wed 27/07/22
11		Create Risk Management Plan	4 days	Fri 22/07/22	Wed 27/07/22
12		Create Techinical Proccess Plan	4 days	Fri 22/07/22	Wed 27/07/22
13		Supporting Proccess Plans	4 days	Fri 22/07/22	Wed 27/07/22
14		Create Managerial Proccess Plan	4 days	Fri 22/07/22	Wed 27/07/22
15		Deliver Project Plan	2 days	Thu 28/07/22	Fri 29/07/22
16		**System Design**	**21 days**	**Mon 1/08/22**	**Mon 29/08/22**
25		**System Build**	**29 days?**	**Tue 30/08/22**	**Fri 7/10/22**

Figure 2: Gantt Chart part 1



	Task Mode	Task Name	Duration	Start	Finish
1		**Wilma**	**65 days?**	**Mon 11/07/22**	**Fri 7/10/22**
2		**System Analysis**	**15 days?**	**Mon 11/07/22**	**Fri 29/07/22**
16		**System Design**	**21 days**	**Mon 1/08/22**	**Mon 29/08/22**
17		Class Design	7 days	Mon 1/08/22	Tue 9/08/22
18		Interface Design	5 days	Wed 10/08/22	Tue 16/08/22
19		Database Design	5 days	Wed 10/08/22	Tue 16/08/22
10		Build Test Plan	3 days	Wed 17/08/22	Fri 19/08/22
11		Review Design	2 days	Mon 22/08/22	Tue 23/08/22
12		Submit Progress Report 2A	3 days	Tue 9/08/22	Thu 11/08/22
13		Submit Progress Report 2B	3 days	Fri 12/08/22	Tue 16/08/22
14		Submit Project Requirements Specification and Design	4 days	Wed 24/08/22	Mon 29/08/22
25		**System Build**	**29 days?**	**Tue 30/08/22**	**Fri 7/10/22**

Figure 3: Gantt Chart part 2



	Task Mode	Task Name	Duration	Start	Finish
		Wilma	**65 days?**	**Mon 11/07/22**	**Fri 7/10/22**
		System Analysis	**15 days?**	**Mon 11/07/22**	**Fri 29/07/22**
		System Design	**21 days**	**Mon 1/08/22**	**Mon 29/08/22**
		System Build	**29 days?**	**Tue 30/08/22**	**Fri 7/10/22**
		Build Database	6 days	Tue 30/08/22	Tue 6/09/22
		Create Project Code	17 days	Wed 7/09/22	Thu 29/09/22
		Build Testing	2 days	Fri 30/09/22	Mon 3/10/22
		Submit Project Report 2C	3 days	Wed 14/09/22	Fri 16/09/22
		Submit Project Report 2D	3 days	Wed 28/09/22	Fri 30/09/22
		Release	1 day?	Tue 4/10/22	Tue 4/10/22
		Final Report	4 days	Tue 4/10/22	Fri 7/10/22

Figure 4: Gantt Chart part 3

The budget summary outlined in Table 2 is an important aspect of project documentation that allows for cost estimation, resource allocation, and adherence to budgeting constraints. As students, we are generous and work free of charge from the goodness of our hearts. To offer an example, the budget overview of the WILMA project is quite simple and would look like the table below if it were necessary:

Staff Salary Budget√

Staff ID	Staff Member Name	Hours Per Week	Length of Project (Weeks)	Hourly Cost ($)	Total Cost ($)
12115435	Nicholas M. Griffin	25	12	120	$36,000.00
12060962	Nathan Snow	25	12	120	$36,000.00
12149026	Rory Allen	25	12	120	$36,000.00
12046570	Nathan Downes	25	12	120	$36,000.00
TOTAL		**100**	**48**	**$480.00**	**$144,000.00**

Table 2: Example of budget overview

Required Resources Budget

The WILMA project primarily uses technology that is open source. Further, other resources used that would require payment - such as Microsoft Visio and Project - are also provided for free to students, which removes any additional budget requirements from the project. With that in mind, the final budget estimation is approximately **$144,000.00**.√

B. Evolution of the Plan

Throughout the development of the project, the client will be given deliverables at certain milestones. These milestones have been outlined previously in Table 1. At each of the milestones, the current iteration of the project plan will be reviewed and updated where required. To maintain a well-structured and achievable plan it is important to consider some of the following scenarios which could be indicative of planning complications:

- Crucial tasks remaining uncompleted at the end of Sprints
- Large task backlog that never gets smaller
- Required weekly time commitment of team members increases beyond expectations
- Scope creep
- Client dissatisfaction
- Project deliverables not being completed to a professional standard

At each milestone, the review process will be conducted, and any relevant stakeholders will be informed immediately of any changes that are made to the project plan.√

C. Project Organisation√

The organisational structure of the project is inclusive of both external and internal stakeholders and aims to explain the roles, responsibilities, and communication methods with the distinct stakeholders of the WILMA project.

i. External Stakeholders

The first external stakeholder of the WILMA project is our client, Central Queensland University (CQU). CQU has declared Lily Li as their representative, and she can be contacted 7 days a week via email. Regular meetings will be held with Lily on an ongoing, weekly basis in a casual setting to

ensure frequent communication about project progress and to update our client on any significant changes that may occur. Formal documentation will also be given to the client at certain milestones throughout the project to ensure they are immersed in the planning, design and development process.

Another group of project stakeholders is the students that will use the product beyond its delivery. Communication with students will be limited throughout the duration of the project, but formal user manual documentation will be provided upon the release of WILMA.

Industry business partners are an important group of stakeholders that also stand to benefit from the product beyond its delivery. Communication with this group of stakeholders will be sporadic. Some of the current users of CQU's Work Integrated Learning program will be asked if they could complete a semi-formal questionnaire to evaluate their requirements of the system. At the completion of the project, formal user manual documentation will also be provided to the industry business partners.

ii. Internal Stakeholders

The internal stakeholders of the WILMA project include our team consisting of Nicholas McGuffin, Rory Allen, Nathan Downes and Nathan Snow. Each team member will be given the opportunity to manage the project for an approximate three-week period, making us equally responsible for its success. Our communication is conducted in a variety of ways:

- **Formal:**
    - Microsoft Teams – document sharing, formal communication
    - Jira – Task allocation and management
    - GitHub – Version control, WILMA documentation, product review and analysis
    - E-mail – Internal formal communication and document sharing (where required).
    - Discord - Weekly formal team meetings. Meeting minutes are taken.
- **Semi-Formal:**
    - Discord – Less important questions, conveying task completion and review requirements to other team members, basic team management.

The responsibilities for the WILMA development team are outlined in Table 3 below.

Name	Role	Responsibilities
Nicholas McGuffin	Project Manager (11/07/22 - 29/07/22) Full-Stack Software Engineer	Project Management Project Design Project Implementation Software Development Product Review Product Validation Quality Control
Rory Allen	Project Manager (30/07/22 - 28/08/22) Full-Stack Software Engineer	Project Management Project Design Project Implementation Software Development Product Review Product Validation Quality Control
Nathan Downes	Project Manager (29/08/22 - 18/09/22) Full-Stack Software Engineer	Project Management Project Design Project Implementation Software Development Product Review Product Validation Quality Control
Nathan Snow	Project Manager (19/09/22 - 14/10/22) Full-Stack Software Engineer	Project Management Project Design Project Implementation Software Development Product Review Product Validation Quality Control

Table 3: Roles and Responsibilities of the Team

D. Managerial Process Plan

i. Risk Management Plan

The software project's schedule and its product's quality are threatened by a variety of potential risks over the course of the software development life cycle. Sommerville (2016) describes the importance of identifying risks, and of developing strategies to avoid, mitigate, and/or implement contingencies to the effects of those risks identified. The threat that a risk poses can be measured by its probability (i.e., Low, Medium, or High likelihood), and the severity of its effects (i.e., Catastrophic, Serious, Tolerable, or Insignificant). Table 1 below lists potential risks to the success of the project (in descending probability), and proposed avoidance/mitigation/contingency strategies for each.

e.g.

Risk 1: A team member is sick

Mitigation 1: Keep all work accessible for all group members. Another team member can carry on the task…

Risk 2: The source code is lost

Mitigation 2: Back up the work onto OneDrive. …..

ii. Meeting Schedules√

In line with the agile methodology practices and to ensure maximum communication between stakeholders, a series of meetings have been scheduled on a weekly basis. The first of these meetings is scheduled for Thursdays and involves all stakeholders including but not limited to:

- The client/s
- The project manager/scrum master
- The whole development team

Such meetings can be seen in the following table.

Date	Purpose	Attendees
14-Jul-22	Project commencement	All stakeholder internal and external
21-Jul-22	Weekly stakeholder meeting	All stakeholder internal and external
28-Jul-22	Weekly stakeholder meeting	All stakeholder internal and external
04-Aug-22	Weekly stakeholder meeting	All stakeholder internal and external
11-Aug-22	Weekly stakeholder meeting	All stakeholder internal and external
18-Aug-22	Weekly stakeholder meeting	All stakeholder internal and external
25-Aug-22	Weekly stakeholder meeting	All stakeholder internal and external
01-Sep-22	Weekly stakeholder meeting	All stakeholder internal and external
08-Sep-22	Weekly stakeholder meeting	All stakeholder internal and external
15-Sep-22	Weekly stakeholder meeting	All stakeholder internal and external
22-Sep-22	Weekly stakeholder meeting	All stakeholder internal and external
29-Sep-22	Weekly stakeholder meeting	All stakeholder internal and external
06-Oct-22	Project end	All stakeholder internal and external

Table 4: Internal and External Meeting Schedule

The second weekly meeting series has been scheduled as per the table below. These meetings are for the development team to review and discuss aspects of the current or next sprint including tasks and responsibilities. These meetings are currently scheduled to take place on Saturdays due to external commitments of the development team.

Date	Purpose	Atendees
16-Jul-22	Sprint 1 planning and project start	Development team
23-Jul-22	Sprint 1 progress review	Development team
30-Jul-22	Sprint 1 review, sprint 2 start, and rotate project manager	Development team
06-Aug-22	Sprint 2 progress review	Development team
13-Aug-22	Sprint 2 progress review	Development team
20-Aug-22	Sprint 2 review, sprint 3 start, and rotate project manager	Development team
27-Aug-22	Sprint 3 progress review	Development team
03-Sep-22	Sprint 3 progress review	Development team
10-Sep-22	Sprint 3 review, sprint 4 start, and rotate project manager	Development team
17-Sep-22	Sprint 4 progress review	Development team
24-Sep-22	Sprint 4 progress review	Development team
01-Oct-22	Review and handover project	Development team

Table 5: Internal Meeting Schedule

iii. Control Plan√

The development team takes responsibility for the project deliverables and management of the project schedule. Quality assurance will be managed entirely internally. Code review will be organised by individual developers prior to submission. The internal (development team peer) reviewer will perform testing as per the defined test plan procedures.

Weekly development meetings are scheduled with all members (as described above in Meeting Schedules). Current progress and issues will be presented. The team will assess the project progress in relation to schedule milestones. Meeting minutes are distributed shortly after meeting completion.

The distributed meeting minutes detail expectations for the following week in terms of which tasks need to be completed by which team members. Progress is relayed between team members in real time through communication in discord chat and task management in Jira.

Weekly stakeholder meetings are scheduled between at least one development team representative and the client (as described above in Meeting Schedules). Current progress is demonstrated to the client. Any changes to requirements presented by the client are communicated back to the rest of the team for consideration, and a special interim development meeting can be scheduled to discuss scope changes, as necessary (as described in Project Control and Conflict Resolution Management below).

iv. Project Control and Conflict Resolution Management

Maintaining control of the project in the event of major disruptions will require careful coordination of the team's already stretched resources. Major changes that have the potential to affect the project's milestones need to be discussed and agreed upon by the entire development team through a special interim meeting held as soon as possible, or the next scheduled weekly meeting. √

Likewise, internal disputes will require a special interim meeting to be held at the team's earliest convenience. Conflict will be discussed with the entire team present, and if the issue cannot be resolved diplomatically, the project manager will determine the outcome.√

Problems that are identified with existing code are to be documented immediately and resolutions given the highest priority in the backlog. This forms part of the team's quality assurance strategy (See Supporting Process Plans below) and allows the team to keep on top of system requirements while maintaining the project schedule.√

Lastly, if project milestones are behind schedule, the team will decide upon altering the project's scope to maintain the existing project completion date. Alternatively, the team may decide to request an extension to the project completion date, with the client's approval.√

E. Technical Process Plan

The Work Integrated Learning Management Application (WILMA) will use a layered architectural design as depicted in the diagram below.

A typical workflow involves HTTP requests being sent from the client to the controller, which will first be intercepted by a security framework to authenticate and authorize client users. After security measures have finished, the security provider forwards the HTTP request on to the controller which will then access the appropriate service to handle its task. The service will then request data from the repository layer (which handles all the database interaction) and performs the appropriate logic before returning the result to the controller.
The controller then, based on the client request, will either return HTML content which is rendered dynamically by the Thymeleaf template engine into static HTML, or return a JSON response containing the requested data.
A security √

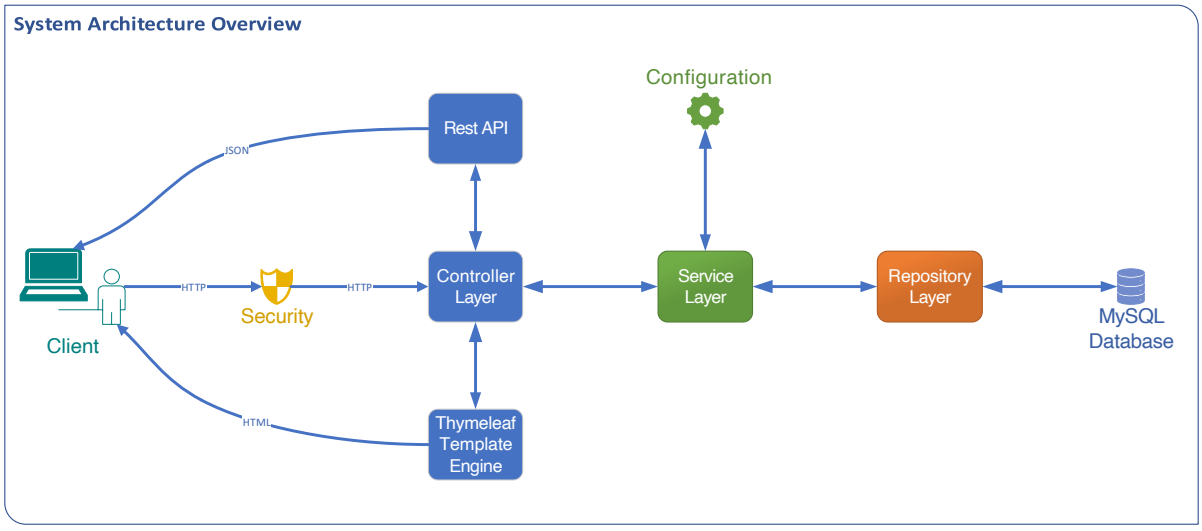


Figure 5: System Architecture Overview

Initial high level use cases identified for the WILMA system are as shown in the use case diagrams below. We can see from both the level 0 and level 1 diagrams that entities for Educator, Partner (employer), and Student are obvious, and although different they share many functional similarities. Basic requirements for a student are to

- View jobs and placements
- View and post questions and answers in a forum√
- Apply for a job or placement

- Send their CV

An employer (or placement provider) needs to

- Offer a job and/or placement
- View and post questions and answers in a forum
- View jobs and placements by both them and other employers
- Send a feedback report to the system (for a placement undertaken)

An educator represents anyone on behalf of the university, will need to

- View and post questions and answers in a forum
- View jobs and placements
- Approve (moderate) placements and jobs before they are advertised
- Update the placements (change, delete)
- View feedback report
- Change placement status (active, in process or complete)

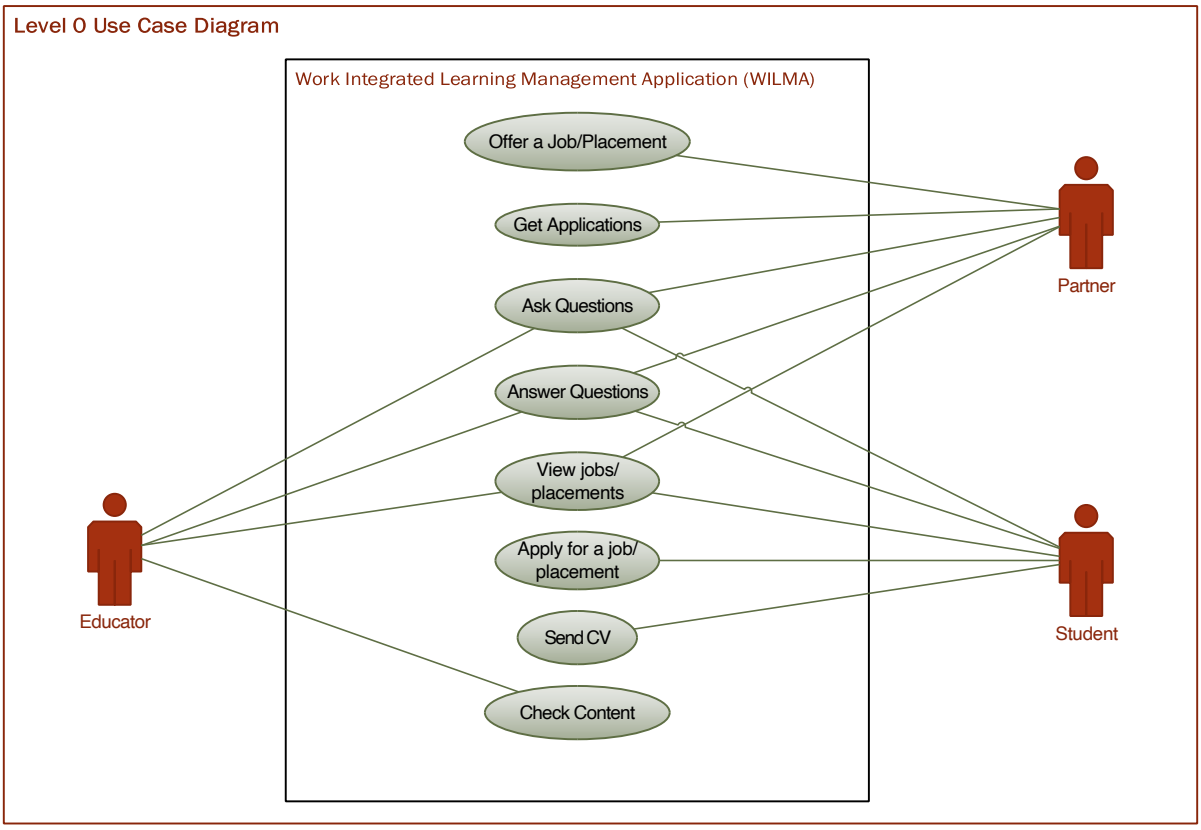


Figure 6: Level 0 Use Case Diagram

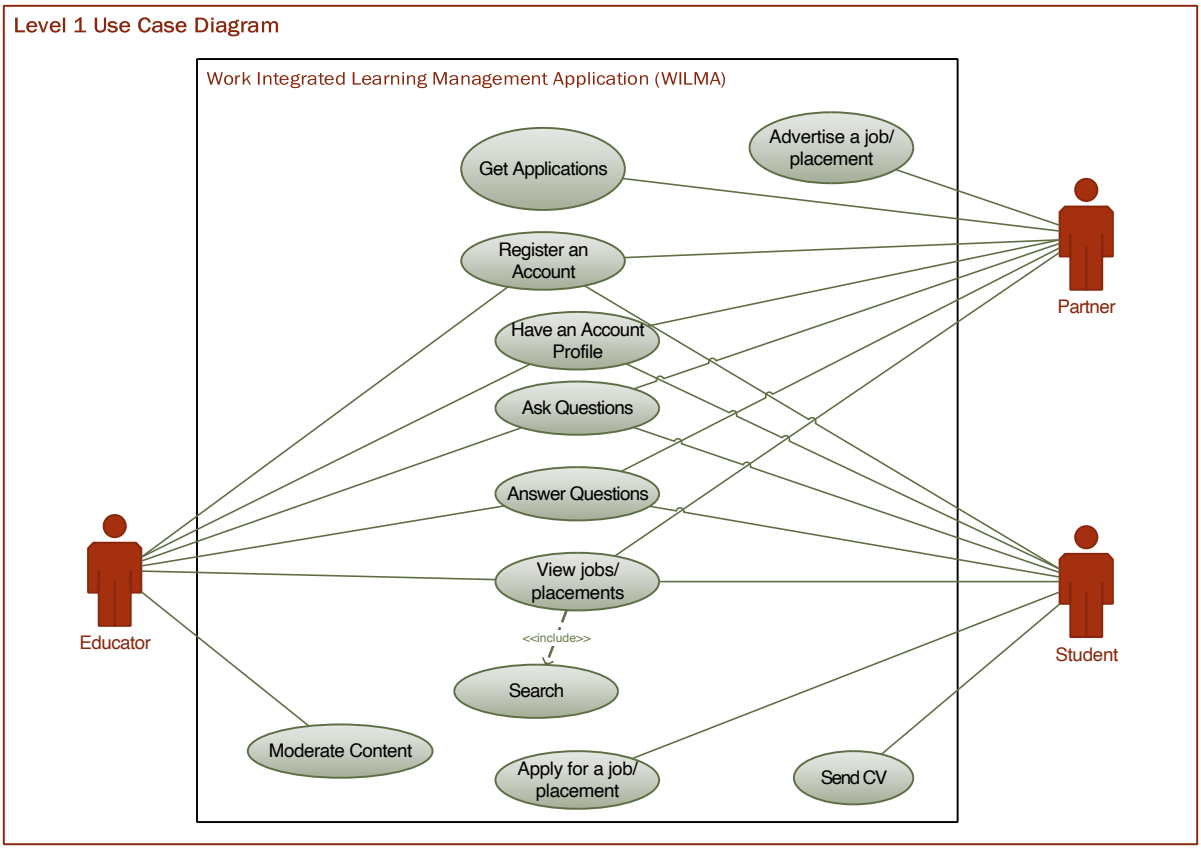


Figure 7: Level 1 Use Case Diagram

i. Methods, tools, and techniques

a) Development Methodologies

The WILMA development team has chosen to proceed with an "agile scrum" methodology for a couple of main reasons.√

Firstly, the nature of the agile scrum methodology will help minimize the risk of bugs due to smaller and more frequent system releases, and the requirements involved in making changes will be inherently smaller. The development team has also identified a few uncertainties about the project direction in general, and so the agile approach leaves us open to tweaking such project directives.

The development team has agreed that each sprint shall be scheduled for a length of 2 weeks, and as shown in the image below each iteration will consist of:

- Planning the sprint, including assigning user stories and identifying feature dependencies.
- Designing the code, tests, and GUI views.
- Actual application/component development.
- Unit testing.
- Deploying the code – This could be locally, to the *development* branch (which will again run tests), or to a live production environment.
- Review the sprint including deployed code and any issues encountered along the way.

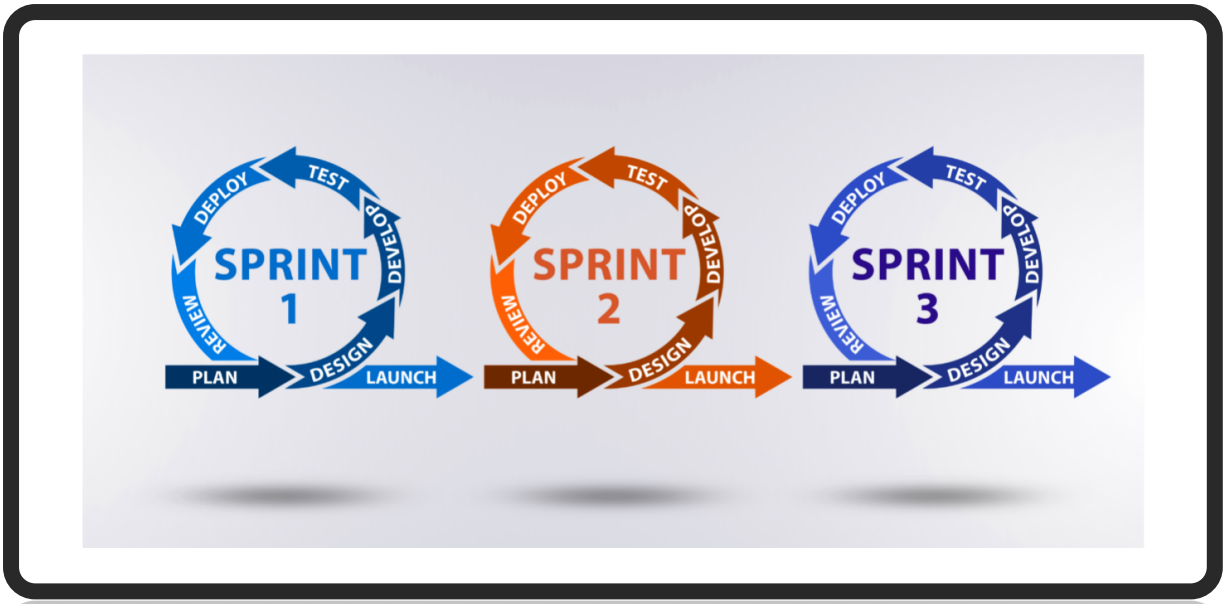


Figure 8: Agile Scrum Methodology (source https://kruschecompany.com/agile-software-development/)

b) Programming Languages

The WILMA server-side application is primarily built using Java version 11 due to its long term support (LTS) and widespread adoption and compatibility. Initially the development team was hopeful on working with Java development kit (JDK) 17, but investigation revealed that not all major cloud providers have full support for JDK17 at this stage, which does not appear to be the case for JDK11. With that in mind the decision was made to use JDK11 to reduce the risk of compatibility issues.√

Additionally, some of the application components also contain markdown (md), yml (YAML), HTML, CSS, and JavaScript files for static web content and configuration settings.√

c) Design tools used

Microsoft Visio is the chosen architectural design tool for the WILMA system due to both its availability amongst the development team and standardized UML components and relational mapping that will minimize the risk of diagram confusion.

According to one of Figma's forums, while a Figma account is required to collaborate on designs, an account is not required in order to view draft design files with the ".fig" file extension. This is an important aspect as not all stakeholders will want or need to collaborate on the UI designs, but certainly need to view and interact with them. Additionally, Figma supports all the required export file formats including PNG, JPG, SVG and PDF.√

d) Procedures to be followed for design, build, and test

Design Procedure√

As this system makes use of a modular architecture and package names make many tasks obvious, it is important to recognise a few rules to ensure code design and housekeeping expectations are kept in check.

1. Read and understand the documentation, code, and hierarchy to ensure your solution aligns with the current structure of the system.
2. The entry point for this application is the **web** module which contains web and REST controllers to suit their individual tasks. To maintain abstraction, security, and overall good

practice is it important that no logic be placed in these controllers; instead, the expected approach would be to create an appropriately named “service” in the **service** module that contains the logic and calls upon your repositories. Then using dependency injection, inject your service into your controller so that the result from your service may be passed back to the client.

3. By design, this application is built using the Spring Boot framework which comes with a lot of functionality baked in. To save unnecessary clutter and reinventing of the wheel, ensure that your code first makes use of functionality already available before writing custom code.
4. Testing and the way we build our test cases is also a part of WILMA’s design. Rather than build test cases ‘method-by-method’, it is expected that tests be meaningful and demonstrate the code solving a purpose. For example you might have a test method named “addQuestionToForum()” which solves a more wholesome purpose than a simple test case like “setQuestionId()”.
5. Use abstraction wherever possible to hide method implementation.
6. Write and structure your code anticipating change. All code is going to be updates as technology demands require and a sign of great work is making our code extensible.
7. Make sure to include documentation including diagrams, sketches, code comments (Javadoc comments are a must), or any other prototypes.

Build procedure√

Exact build procedures will vary slightly depending on the IDE being used, but generally should be much the same as we are using the maven build tool.

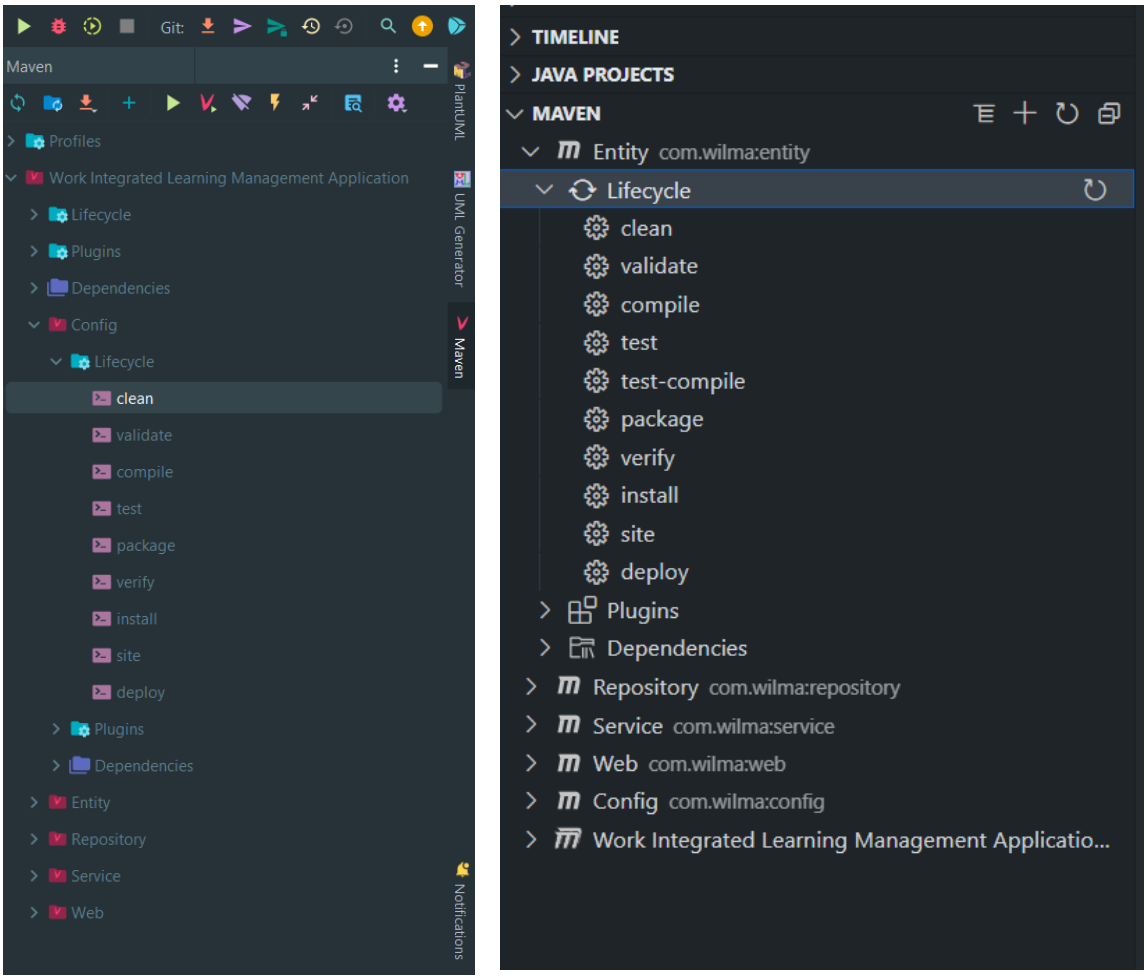


Figure 9: IntelliJ and VSCode Maven Tool Window

NetBeans behaves a little differently to IntelliJ and VSCode, as once you open the project you need to open each individual module (below left) then right click and select “Clean and Build” as shown below on the right. See here for further information on Maven goals including “clean”.

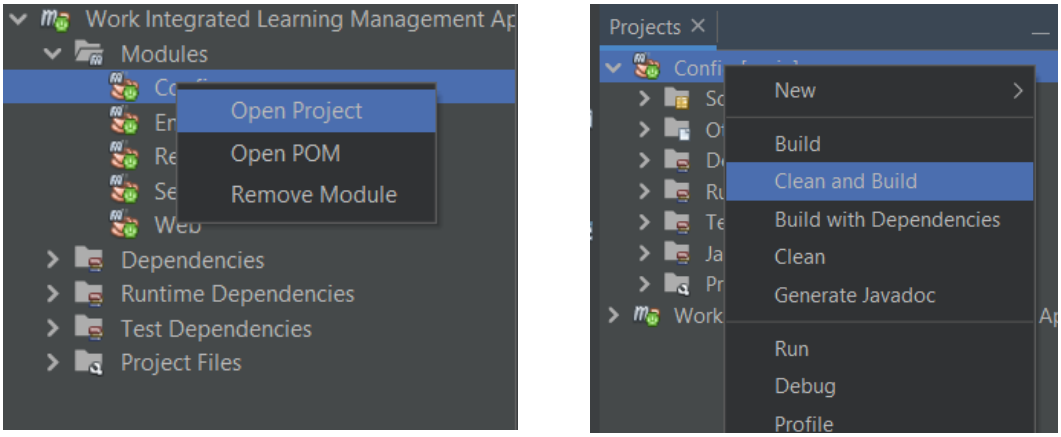


Figure 10: NetBeans Build Process

Testing Procedure √

All code being merged into the WILMA system will be subject to testing and should therefore be accompanied by specific relevant unit tests (the current testing framework in use is Junit5), however it is not a requirement that there be tests each and every method (particularly accessors and mutators) as shown below.

```
//This is not expected
@Test
Void setStudent(){
    //Test logic
}
@Test
Void getStudent(){
    //Test logic
}
```

```
//This is expected
@Test
Void enrolStudentIntoClasses{
    //Test logic
}
```

Figure 11: Unit Test Demonstration

However, unit tests are not to perform transactions on any live databases as this can have serious unwanted side effects that are difficult to trace. Instead it is required that transactional unit tests incorporate the Mockito framework to 'mock' classes or interfaces that interact with any real data, an example can be found here. Mockito and Junit5 are included in the WILMA application via the spring-boot-starter-test dependency, so no additional dependencies or libraries are required.

e) Version control √

The WILMA application source code makes use of the git versioning system and is hosted online in a GitHub repository due to its widespread adoption, community support, and abundance of reference and instructional material.

f) Merging of components √

The GitHub repository implements a **developer** branch which is used as a "staging" branch prior to merging changes into the master branch, and this is for two major reasons.

Firstly, it adds a platform for contributor pull requests to be merged into which can also then run automated testing via GitHub actions. Many feature pull requests can be merged into the development branch and at the end of a development iteration/sprint (ensuring all of the development branches' automated tests pass) a pull request can be created to merge the contents of the development branch in with the master branch which will then become a new release.

Secondly, future intentions for this application include setting up a continuous delivery and deployment (CICD) pipeline that will merge and deploy the application to a live cloud server. These deployments will be sourced from the master branch and are triggered by branch events such as merging pull requests and creating an interruption to the service of the live application during this process. Hence, the implementation of the development branch allows developers to merge their work and tests to be run *before* making a merge to the master branch and causing any disruptions.

h) Maintaining documents updated√

Project documentation will be accessible to collaborators via the repository wiki and can also be cloned to your local machine using https://github.com/nick-mcguffin/Application-Development-Project.wiki.git.

Using the repository wiki enables us to keep the project documentation together with the source code yet still separated as far as versioning is concerned.

Although GitHub does not currently support branches for wiki versioning, changes will be committed and pushed to the master branch (totally unrelated to the source code master branch) and will still be versioned with the full commit histories.

F. Infrastructure Plan

i. Developer Tools√

a) Operating System

This application is built using Java (JDK11) and as a result it is cross-platform compatible and can therefore be developed on Windows, Linux, or MacOS.

b) IDEs

There is a large choice of IDEs/editors that can be used to develop components of the WILMA system, particularly those that have Maven and Tomcat server support, some of which include:

1. IntelliJ IDEA (Community, Education, or Ultimate editions)
2. Visual Studio Code (VSCode) – There are lots of extensions available, but none required
3. Spring Tools Suite – A spring flavour of the Eclipse IDE redesigned by the Spring team
4. Eclipse IDE – Open source IDE
5. Apache NetBeans – Recommend NB Spring Boot Plugin

Text editors such as Notepad++, Sublime Text, and Atom can also be used for lightweight editing but may have issues performing maven goals or launching the embedded tomcat server, however many of these commands can be performed via a command line terminal.

c) Unit Testing Tools

The WILMA system incorporates the Junit test framework, in particular Junit5, which is widely adopted and respected around the world.

To mitigate the risk of alteration or damage to live data, the Mockito framework will be used to mock transactional operations. This way transactional operations can still be tested in an effective manner while being safely isolated from any production databases.

d) Servers

As a Spring Boot application, WILMA comes packaged with an embedded and preconfigured Tomcat web server so there is no immediate configuration required, however server configuration can be set manually in the project “application.yml” files.

For example, the default server port for spring boot applications is 8080 but Amazon Web Services (AWS) Elastic Beanstalk (ELB) deployment guide states that AWS requires the application to run on port 5000 and the easiest way to comply is to change the application server port to 5000 and not the ELB settings to port 8080 (Villa 2022). This can easily be done in the application.yml file by making the following change:

```
server:
  port: 5000 #8080
```

Figure 12: Changing the server port

ii) Hardware required for Development and Client Side√

The following outlines the minimum hardware required to develop this system (Oracle 2022):

- A desktop, laptop, or virtual machine running Windows, Linux or MacOS
- At minimum a Pentium 2 266 MHz processor
- 181 MB disc space for development tools
- 128 MB RAM

iii) Client-Side Software Requirements√

In the event of a live deployment, the WILMA system will be available to browse and interact with via a simple web browser on a PC or smart device with no further configuration.

If the WILMA system is to be deployed and viewed locally, once running the client will be available for viewing in any web browser at http://localhost:8080/

G. Supporting Process Plans √

i. Configuration Management Plan

System source code will be versioned using the Git versioning system to trace and monitor changes, with the repository being housed on GitHub due to its widely accepted usage, integration, and community support & documentation.

All collaborative changes made to the source code must be requested via GitHub pull requests, which will be reviewed and considered by the team. A **Pull Request Template** will be provided to ensure sufficient information is provided to the reviewer.

Similarly, issues can be created to bring a bug or requested feature to the team's awareness. Templates will also be provided for both **Bug Reports** and **Feature Requests** to help ensure adequate information is provided.

To help maintain the high standard of code and documentation, the repository will also feature a set of **Contributor Guidelines** which will outline expectations and answers to commonly/frequently asked questions. One such guideline will be to use the "feature branch" approach as shown below in figure 1, specifying that each proposed feature, for example *"user authentication"*, would be in a branch of its own and have pull request/s relating to just that feature.

In addition to the feature branch approach shown in figure 1, this project includes an additional failsafe in the form of a **development branch** (or *staging* branch) which can be visualised in figure 2. All pull requests and/or issues will be merged with the development branch which is also equipped with automated testing via GitHub Actions. Once tests pass on the development branch and the team is satisfied with the changes made, the development branch can be merged into the master branch and subsequently become a "release". The master branch will also run automated build tests, both as a sanity check and to feed a status badge in the master branch readme file, as this branch [master] is the source that will, in future releases, feed a continuous integration & delivery pipeline.

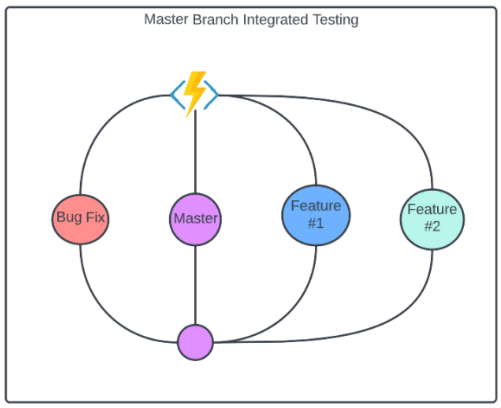


Figure 13: Default Feature Branching

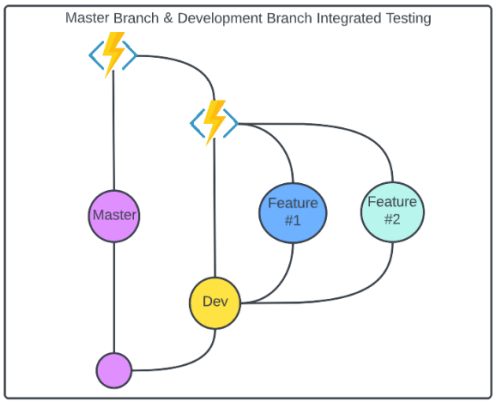


Figure 14: Using a Development (staging) Branch

ii. Verification and Validation Plan √

Verifying and validating project components before they are added to the system is paramount in ensuring the quality of the system in its entirety.

The following procedures are intended to help maintain project quality and professionalism of system components

a) Project Milestones

Project milestones, regardless of size, shall go through a peer review process where they are checked to ensure the highest quality outcome.

For the purposes of peer reviews, a “reviewer” must be someone familiar with the project, the project scope, and the project stakeholders.

Unless otherwise advised, a peer review process should include:

- Ensuring that milestones are broken up into manageable sprints.
- Ensure each person involved in the milestone development understands their tasks.
- Ensure persons involved have adequate training and skills for their allocated task/s.
- Be thoroughly checked for grammatical and spelling errors.

- Any documentation is to be clear, concise, and written in English.
- Ensuring there is sufficient documentation which has been added to:
    - Project management body of knowledge (PMBOK)
    - Operational manuals
    - Any other reference material
- Documents are saved in commonly found file formats for windows
- Not less than two (2) reviewers have critically reviewed the documentation

b) Source code and Documentation

All source code submitted to the project repository must be done so by way of a **Pull Request (PR)**. A specific PR template have been created to help contributors prepare for the review process.

Note: Don't commit your code directly to the master or development branches.

All source code and documentation will be checked for the following:

- Code has sufficient Javadoc comments (giving examples where needed).
- Code is clear and readable, being sure to use meaningful names and effective whitespace.
- Appropriate naming convention is used according to the language used.
- Code duplication is minimised through use of polymorphism and inheritance.
- Dependencies have been checked for vulnerabilities.
- There is sufficient tests for the code.
- There is no logic in views, use service/controller methods as helpers
- Use spaces between method parameters `[1, 2, 3]` *not* `[1,2,3]` and around operators `a + b` *not* `a+b`
- Documentation is sufficient for the code submitted
- Code compiles and runs error free
- Not less than two (2) reviewers have critically reviewed the code and documentation
- At least one (1) reviewer has compiled and ran the code

iii. Documentation Plan √

The definitions and schedules of project documents are outlined in the table below. Please note that deliverable documents are those that are produced for the client, while non-deliverable documents are for internal use only.

Document	Content	Deliverable	Audience	Completion	Template /Standard

User Manual	Exhaustive product usage instructions detailing every exposed user facing functionality	Yes	System Administrator	07/10/2022	NA
Quick Start Guide	Simple sequences of basic operations	Yes	Client	07/10/2022	NA
EULA	Detailed fair usage contract between developers and end users	Yes	All End Users	07/10/2022	NA
Design Specification	Includes requirements specification, class diagrams, ER diagrams, system sequence diagrams, use cases, etc.	No	Team	29/08/2022	IEEE 830-1998
Test Plan	Detailed procedure and schedule for unit, integration, system, and UA testing	No	Team	TBA	IEEE 829-1998
Meeting Minutes	Includes date, time, duration, and members present, plus an itemized list of discussion topics, and finalized decisions	No	Team	Weekly from 16/07/2022 plus additional interim meetings as required	Template defined in internal document

Figure 15: Document Schedule

iv. Quality Assurance Plan√

As is the case with most other aspects of project management in agile development environments, quality assurance is significantly less formal (Sommerville 2016). This will in no way affect the final quality of the product; however, as the review process is integrated into each sprint, it allows the team to address quality concerns throughout the entire development process rather than just the end. Our development team will embody a culture of good practice (as described below) which, in concert with a Quality Audit performed by the client, will allow for the deployment of a final software product of optimal quality.

As a small team with limited resources, we do not have the luxury of being able to implement formal review processes. Because of this, we each need to adhere to best practices to ensure that development is not only efficient but also maintains a level of quality from the very beginning. To achieve this, the team will adopt the following practices:

- Members will only submit changes reviewed and accepted by their peers.
- Members must not submit code that results in system failures. Integration testing is imperative.

- The system is owned by the team. Problems identified in a piece of code can be resolved by any member, not just its original author.

These practices constitute our development culture, but we extend these by including a review process at the end of each development sprint also (2-day review, per 2-week sprint). If, in these reviews, we identify that a system requirement does not meet its criterium, a fix will be added to the backlog of the next sprint with the highest priority.

In the final sprint, time is dedicated to ensuring that all non-functional requirement qualia meet their predefined criteria. These requirements are discussed with the client initially and reiterated at the beginning of each sprint. This ensures that when it comes to the client performing their final quality audit, the product performs as expected.

Agents	Role	Responsibility
Nathan, Nathan, Nick, Rory	Development Team	Unit Testing, Integration Testing, System Testing, Quality
Lily Li	Client	Quality Audit

Figure 16: Quality Assurance Roles

Requirement	Onus	Method	Criteria
Security	Development Team	Testing	Access/functionality limited by user role appropriately.
Maintainability	Development Team	Ongoing Review	Following handover, up to 15 hours of support per month (for 2 years) can be provided to the client. Additional support charged at $120 per hour. Periodic service updates provided gratis for 2 years. Additional service maintenance of $2400 per year required after 2 years.
Performance	Development Team Client	Testing, Ongoing Review	0.2s response time. Distributed systems should be indistinguishable from local operations. Should operate
Extensibility	Development Team	As needed	Exposed endpoints for future user modalities.
Availability	Development Team, Client	Testing, Ongoing Review	<2% downtime. All maintenance will be carried out outside of regular business hours to minimize interruptions to service.
Usability	Client	Testing, Ongoing Review	Adherence to accepted UX heuristics.

Figure 17: Acceptance Qualia

H. References

IEEE 1998, *Recommended Practice for Software Requirements Specifications,* in IEEE Std 830-1998*, pp. 1-40*, viewed 25 July 2022, https://ieeexplore.ieee.org/document/720574

IEEE 1998, *Standard for Software Test Documentation*, in IEEE Std 829-1998, pp. 1-64, viewed 25 July 2022, https://ieeexplore.ieee.org/document/741968

Oracle 2020 (1993), *14 Windows System Requirements for JDK and JRE*, viewed 25 July 2022, https://docs.oracle.com/javase/8/docs/technotes/guides/install/windows_system_requirements.html#A1097784

Sommerville, I 2016, *Software Engineering*, 10th edn, Pearson

Villa, J 2022 (2016), *Deploying a Spring Boot Application on AWS Using AWS Elastic Beanstalk*, November 9, viewed 25 July 2022, https://aws.amazon.com/blogs/devops/deploying-a-spring-boot-application-on-aws-using-aws-elastic-beanstalk/#:~:text=By%20default%2C%20Spring%20Boot%20applications%20will%20listen%20on%20port%208080.