PROJECT PROPOSAL AND PLAN REPORT

Software Architecture Teaching: A Systematic Review

(Project ID: 2025-S1-38)

Project Sponsor
Dr Richa Awasthy

Project Mentor
Monika Gahalawat

Group ID: **2025-S1-38-11522TeamA**

Team Members

Sl. No.	Name	Student ID	Role	Contributions
1	Ugyen Lhatshok	u3265091	Leader	Section 1, 5 and 7
2	Pema Gyamtsho	u3267717	Member	Section 8 and 9 Formatting of the report
3	Thinley Dema	u3255665	Member	Section 1, 5 and 7
4	Thinley Rabgay	u3267425	Member	Section 3, 4 and 6 Cover Page Designing
5	Zhijia Ren	u3262965	Member	

Table of Contents

Project Title:

Software Architecture Teaching: A Systematic Review

1. Project Purpose

This research project aims to conduct a systematic review of software architecture teaching with a focus on aligning academic curricula with industry requirements. Leveraging Kitchenham's Principle and the SPIDER framework, the study will comprehensively analyse existing literature and gather evidence to involve educators, students, and industry professionals. The primary objective is to assess the effectiveness of various teaching methodologies in enhancing student academic performance, industry readiness, and overall instructional quality. Furthermore, the research will explore the key challenges faced by educators in implementing these strategies and provide actionable recommendations to bridge the gap between academic learning and industry needs.

2. Project Context/Background/Literature Review

The field of software architecture plays a pivotal role in software engineering, which focuses on system design principles, architectural patterns, and critical decision-making processes at the system level. Despite its significance, studies have indicated the gap between the competencies taught in academic programs and actual skill sets demanded by the software industry. In one of the studies, (Garousi et al. 2020) conducted a systematic literature review analysing 33 studies across 12 countries encompassing over 40000 data points and discovered misalignment between skills taught in software engineering education and those required in the industry. Likewise (Ellis and Garvey 2011), in their thesis, they identified that software engineering education often lags behind industry needs due to rapidly evolving technologies, which emphasises the gap that exists not only in technical skills but also in foundational knowledge areas crucial for industry success. This disparity impacts employability and raises concerns in terms of industry readiness.

Likewise, adopting adaptive teaching learning strategies that will align academic curricula with evolving industry requirements has also been proven to be essential. In a study conducted among software architects, where emphasis was placed on collaborative learning environments, it was revealed that such strategies not only improve academic performance but also reduce the likelihood of repeating common design mistakes (Weise, Brand & Hoorn, 2025). Similarly, (Bhargava et al. 2025) explored mastery learning strategies facilitated by microservice architectures, which support flexible academic policies and enhance student engagement and comprehension. (Rupakheti and Chenoweth 2015) in their experience report on teaching software architecture to undergraduate students, further emphasized the effectiveness of project-based learning in enhancing student engagement and providing practical experience crucial for understanding complex architectural concepts. They stressed the need for pedagogical strategies that adapt to the experience level of students, thereby

making abstract concepts more accessible and relatable. Such project-driven approaches complement collaborative and mastery learning strategies, as they collectively foster deeper understanding, reduce mistakes, and align learning outcomes more closely with industry expectations.

On the other hand, challenges in software architecture education often stem from fragmented research and the lack of consolidated strategies that educators can rely on. For instance, the studies (Souha et al. 2025) conducted on recommender systems and (Tan et al. 2025) on dynamic software-defined networks emphasise the potential use of recommender systems and dynamic network simulations to tailor educational content to individual student needs, thus addressing diverse learning paces and preferences. This lesson highlights the gap that exists in the teaching curriculum and also the key challenges educators face, like the lack of adaptable teaching strategies required to cater for individual learning differences. Without consolidated methods and tools for personalised instruction, it might be difficult to implement effective teaching strategies that ensure all students achieve the desired competencies in software architectural learning.

In conclusion, while there is a growing body of research focusing on diverse teaching strategies in software architecture education, there remains a need for comprehensive systematic reviews that synthesise these findings. Such reviews would aid in identifying best practices, addressing existing challenges, and formulating evidence-based recommendations that align academic programs more closely with industry expectations.

2.1. Problem Statement

Despite the critical role played by software architecture in software engineering, there is a significant gap between the competencies imparted in academic programs and the skill sets demanded by the software industry. Existing research emphasises the misalignments in both technical skills and foundational knowledge areas, which adversely impact graduates' employability and industry readiness. Although adaptive teaching and learning strategies have shown potential in bridging this gap, their integration into academic curricula remains fragmented. There is a lack of comprehensive systematic reviews that synthesise existing teaching strategies, making it challenging for educators to adopt evidence-based methods that ensure students achieve industry-relevant competencies. Addressing this problem requires a systematic exploration of existing literature to identify best practices, consolidate adaptable teaching strategies, and provide actionable recommendations that align software architecture education with evolving industry requirements.

2.2. Scope Description

The scope of this project is to develop a draft report, written in an academically appropriate style, which will summarise the methodology, findings, and key insights from the reviewed literature. In this research, we will be conducting a systematic review to evaluate and synthesise existing secondary data related to teaching strategies in software architecture education. The

emphasis will be on analysing peer-reviewed articles, conference papers, and systematic reviews to understand how various teaching methodologies impact academic performance, teaching effectiveness, and industry readiness. The primary goal of this study is to consolidate knowledge, identify best practices, highlight key challenges, and provide evidence-based recommendations for aligning academic programs with industry expectations. Additionally, the scope will include all the necessary scope items, supported by clearly articulated project objectives and desired outcomes. This structured approach will guide the review process, ensuring that the findings contribute valuable insights toward enhancing software architecture education and bridging the gap between academic instruction and industry needs.

2.2.1. In-Scope Items

2.2.1.1. Review and synthesise secondary data (peer-reviewed articles, conference papers, systematic reviews, etc) related to teaching strategies in software architecture education. Apply inclusion and exclusion criteria to ensure the relevance and quality of the selected studies.
2.2.1.2. Identify recurring teaching methodologies and synthesise findings on how these strategies influence academic performance, teaching effectiveness and industry readiness.
2.2.1.3. Identification of best practices and challenges reported in the literature that define the effective teaching of software architecture.
2.2.1.4. Gather evidence-based recommendations to enhance teaching effectiveness and align academic programs with industry expectations.

2.2.2. Out-of-Scope Items

2.2.2.1. The project will rely exclusively on secondary data, so primary data collection will not be conducted.
2.2.2.2. The study will not design or implement new teaching curricula nor cover hands-on technical training or development of new tools specific to software architecture frameworks.
2.2.2.3. The scope is limited to software architecture education and will not extend to broader software engineering topics unless they relate to architectural learning.
2.2.2.4. The project will not include longitudinal tracking of student performance post-graduation.

2.3. Project Boundaries

2.3.1. Project Objectives

2.3.1.1. Conduct a Rigorous Systematic Literature Review.
2.3.1.2. Assess reported outcomes of different teaching strategies on academic performance and industry readiness.

2.3.1.3. Explore the key challenges highlighted in the literature that hinder effective software architecture education.
2.3.1.4. Formulate Evidence-Based Recommendations for educators to enhance software architecture education.

2.3.2. Project Deliverables

2.3.2.1. A comprehensive report summarising the methodology, findings, and key insights from the reviewed literature.
2.3.2.2. An analysis of effective teaching methodologies for software architecture education.
2.3.2.3. Documentation of the key challenges faced by educators and proposed solutions derived from the literature.
2.3.2.4. A set of evidence-based recommendations aligning software architecture education to industry needs.

2.3.3. Assumptions

2.3.3.1. The project assumes unrestricted access to academic databases such as IEEE Xplore, ACM Digital Library, Scopus, and ScienceDirect.
2.3.3.2. We assume that a sufficient number of peer-reviewed studies are available to support our searches and substantiate our findings.
2.3.3.3. The literature reviewed will not exhibit systematic bias (that could limit the generalizability of findings).

2.3.4. Constraints

2.3.4.1. Time Limitations: The project is time-bound, which may limit the breadth of the literature considered.
2.3.4.2. Exclusion of Non-English Studies: Only English-language publications will be considered, potentially excluding relevant studies published in other languages.
2.3.4.3. Variation in Educational Contexts: Differences in educational systems, geographical regions, and institutional contexts may affect the generalizability of the findings.

2.3.5. Inclusion and Exclusion Criteria

Inclusion Criteria *(what studies to include)*	**Exclusion Criteria** *(What studies to exclude)*
Studies focusing on teaching strategies in software architecture education.	Studies addressing general software engineering topics without specific relevance to software architecture.
Peer-reviewed articles, conference papers, and systematic reviews published within the from 2010-2024.	Articles published before the 2010.
Studies discussing outcomes such as academic performance, industry readiness, and teaching effectiveness.	Studies that do not report relevant outcomes related to educational effectiveness.
English-language publications to ensure accessibility and consistency.	Studies published in languages other than English.
Research involving higher education (undergraduate and postgraduate programs).	Studies focusing on high school education, corporate training, or non-academic settings.

Table 2.1: A table showing inclusion and exclusion criteria.

2.3.6. Project Outcome

2.3.6.1. Comprehensive Synthesis: A consolidated understanding of existing teaching strategies in software architecture education.
2.3.6.2. Best Practices: Identification of the most effective teaching methodologies that align with industry expectations.
2.3.6.3. Challenges and Solutions: Insights into the common challenges educators face and literature-backed solutions.
2.3.6.4. Curriculum Recommendations: Evidence-based recommendations for aligning software architecture curricula with industry-relevant skills.

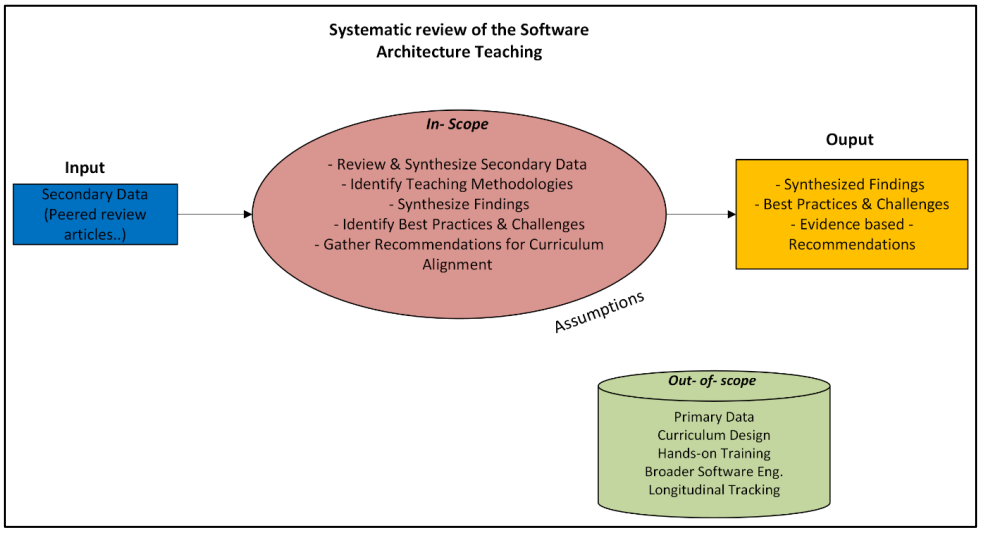


Figure 2.1: System diagram of the project boundaries

3. Requirements Analysis

3.1. Technical Requirements

3.1.1. Academic Databases

To identify and gather relevant studies, we need access to credible journal articles published in renowned academic journals. We can get access to reliable databases such as platform such as IEEE Xplore, Scopus, Google Scholar, and Web of Science by authenticating with our student ID through access control of the University's portal.

3.1.2. Documentation Tools

The team will be using the GitHub Repository as a documentation tool for future purposes. Any work, including the minutes of meetings and draft articles, all related to this project will be secured and uploaded to the GitHub Repository. It will create easily accessible articles for the team that are referred to for the study purpose. We will also use MS Project Professional to generate a Gantt Chart for planning, task management, timeline, milestone and progress tracking of the project. It will act as a crucial tool for the organisation and growth (Ramachandran and Karthick 2019) of the project. Moreover, MS Word will be used for writing the report, whereas EndNote will be used for the correct referencing.

3.1.3. Collaboration Tools

For seamless communication and collaboration, WhatsApp is used for easy communication, and SharePoint, Microsoft Teams and Outlook are used for team meetings and document-sharing purposes. The team members are expected to use daily check-in communication channels for the smooth progress of the project.

3.2. Functional Requirements

3.2.1. Meeting and Reporting

The weekly progress meetings will be held with the Sponsor and Mentor and all the team members are expected to attend. The team will present the progress report and minutes of weekly meetings to the sponsor and mentor. The frequency of the meetings in a week may vary depending on the urgency and requirements. The team members are expected to share the challenges faced, which will be resolved in the team, otherwise with support from the sponsor and mentor if it is unresolved.

3.2.2. Writing and Reviewing

The systematic review paper must follow academic writing standards including formatting and style. It should contain all components of systematic review writing. The team members will contribute to the writing of the draft or final paper with a vigorous review of what we contributed and other's work as well. The paper must also include the feedback and suggestions given by the sponsor and mentor if any.

3.3. Operational or Non-functional

3.3.1. Timeliness

All the tasks will be bound by the deadlines, and team members are expected to complete the task on time. All the team members are expected to make timely contribution for the timely submission of the project.

3.3.2. Team Commitment

All the team members are expected to dedicate their time to completing their individual tasks before 48 hours of submission so that they will have enough time to review the feedback and comments from other team members.

3.3.3. Effective Communication

The team members will use only the agreed channels for communication purposes, and any issues or concerns to be discussed. The team members are expected to respond to any issues or give feedback on the other team's work within 24 hours so that it will maintain the team's progress on track. Moreover, team members are expected to respect their views and accept diversity.

4. Chosen Option/Method and Deliveries

4.1. Project Option

This project aims to do a systematic review of existing literature on software architecture teaching. This will involve identifying and analysing the papers or studies to provide a comprehensive overview of the current state of teaching strategies and approaches, the kind of assessment tools, and the learning objectives and challenges in this field. By doing this, it will ensure that only relevant studies are considered since it is rigorous and methodological. It will provide a comprehensive understanding (Aromataris and Pearson 2014) of the software architecture widely covered under different themes. It will also ensure that the findings are reproducible.

For a successful project, we are going to use Kitchenham's principles, which is a well-recognised and proven framework (Kitchenham and Brereton 2013) for systematic review that has three main phases: planning, analysis, and reporting of the findings. Kitchenham's guideline is mostly used by medical researchers and can be adapted in software engineering research (Keele 2007; Kitchenham 2004). Kichenham's method focuses on a well-defined protocol to guide the systematic review process minimising the biases and increasing the reliability and reproducibility. Moreover, there are many stages in each phase depending on the scope and nature of the systematic review of the study.

The figure below displays the implementation of Kitchenham's method modified in the context of our project, which was adapted from (Prabowo et al. 2021).

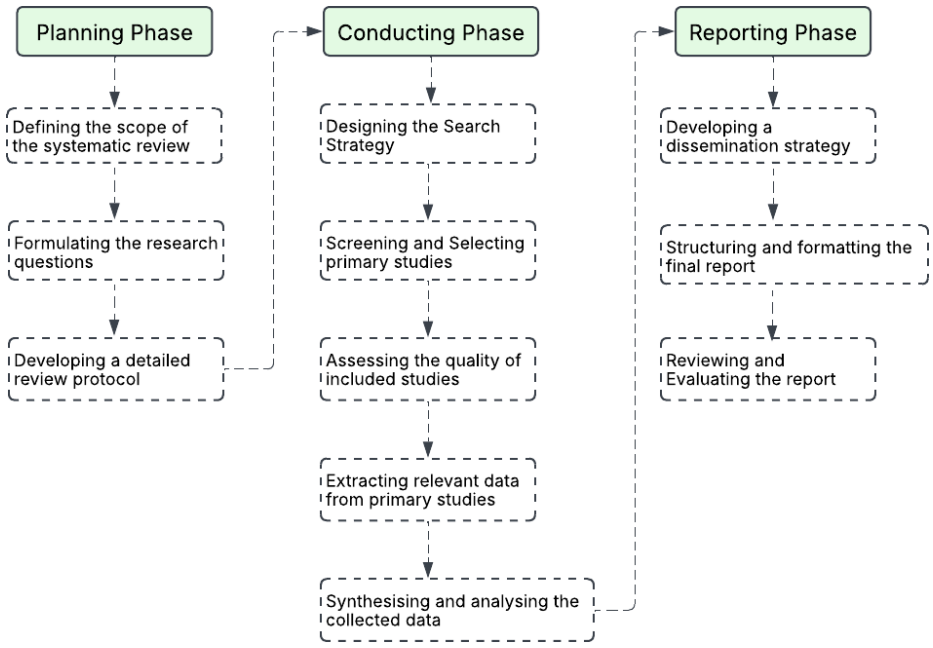


Figure 4.1: Kitchenham's Model with stages of each phase

To supplement, we will be using SPIDER (Cooke et al. 2012) tool to develop the research questions, which will guide the entire research process. Likewise, PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analysis) (Theile and Beall 2024) will be used for the reporting of the findings of the project. It is a well-established framework, providing clear guidelines for each stage of the process, enhancing transparency and reproducibility.

The alternative approach that can be adopted is mixed-method research. This approach will bring a deeper understanding of the study by relating the statistical view supported by research-based reasons. The insight of the statistical view will be gained through the conduct of the survey among the universities (lecturers, tutors and students). However, this approach would not be possible for this project since it will take time to seek ethical issues, and the project must be completed within a short period.

Therefore, we have decided to work with Kitchenham's principles along with PRISMA and SPIDER for this project to align with our goals of the study.

4.2. Project Deliveries

4.2.1. Project Proposal and Plan

A detailed plan outlining the project scopes and boundaries, approach, requirement and quality assurance analysis, schedules, and risks analysis and mitigation strategies with other requirements.

4.2.2. Systematic Review

At least 50 papers from 2010 to 2024 will be reviewed under different themes, forming a systematic review document that will be submitted to the sponsor for guidance and feedback.

4.2.3. Draft Report

The draft report of the project covering the detailed plans as included in the proposal will be submitted for final feedback.

4.2.4. Poster

The poster will be designed to highlight the significant findings with project scopes and background. Moreover, it will also include other important aspects of the project which will convey a message to the audience.

4.2.5. Final Report

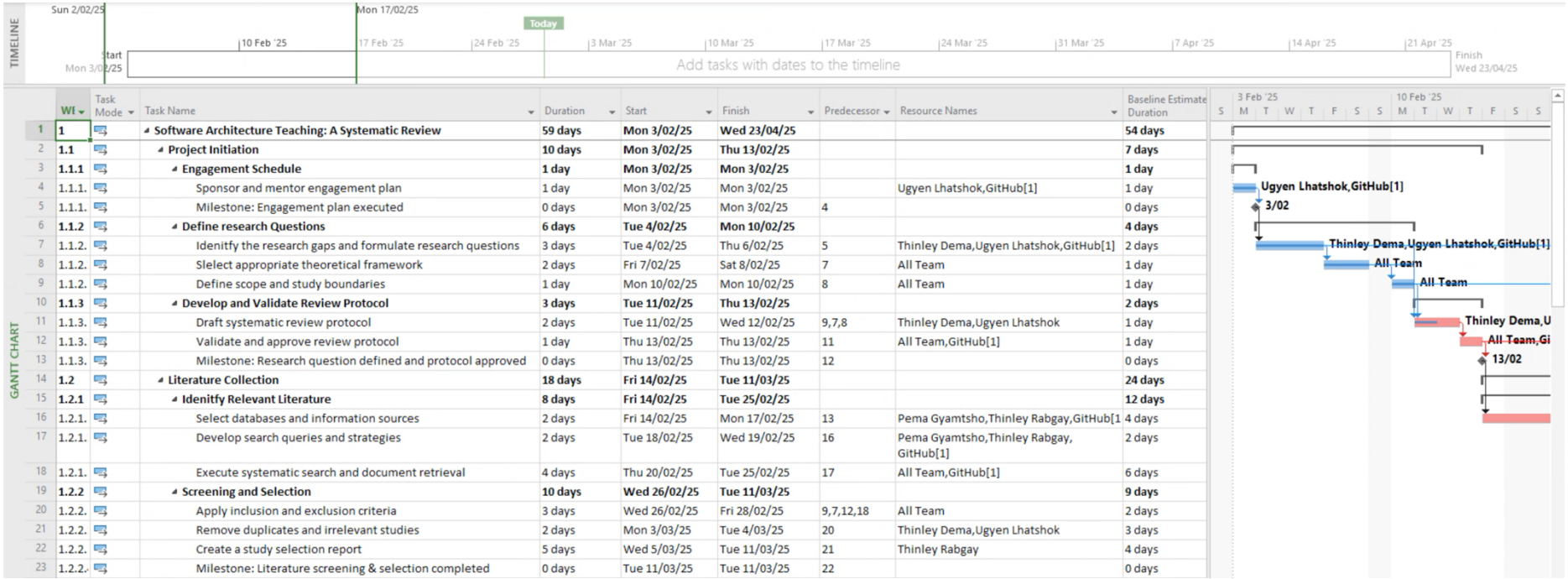
A comprehensive document including all the elements of the study, refined by the feedback from the sponsor, will be submitted, which can serve as the final document for publication or presentation at the conference.

4.2.6. Presentation

The structured presentation of the project will be ready to convey the findings to the sponsor, audience and scholars who are very interested in related fields of study.

5. Detailed Project Plan

A Gantt chart is a powerful visual tool used in project planning and management to illustrate a project's schedule. Developed by Henry Gantt in the early 20th century, it provides a clear timeline of tasks, their durations, and dependencies, making it an indispensable resource for project managers. In project planning, Gantt charts play a critical role in ensuring that projects stay on track and within scope. They enable our project team to break down complex projects into manageable tasks, assign responsibilities, and set realistic deadlines. This clarity helps prevent delays, avoid resource conflicts, and maintain alignment among our team members. Additionally, Gantt charts allow for real-time updates, enabling our team members to adapt to changes and mitigate risks promptly. Gantt charts provide a structured approach to planning, execution, and monitoring, ultimately contributing to the successful delivery of projects on time and within budget.



	WI	Task Mode	Task Name	Duration	Start	Finish	Predecessor	Resource Names	Baseline Estimate Duration
1	1		**Software Architecture Teaching: A Systematic Review**	**59 days**	**Mon 3/02/25**	**Wed 23/04/25**			**54 days**
2	1.1		**Project Initiation**	**10 days**	**Mon 3/02/25**	**Thu 13/02/25**			**7 days**
3	1.1.1		**Engagement Schedule**	**1 day**	**Mon 3/02/25**	**Mon 3/02/25**			**1 day**
4	1.1.1.		Sponsor and mentor engagement plan	1 day	Mon 3/02/25	Mon 3/02/25		Ugyen Lhatshok,GitHub[1]	1 day
5	1.1.1.		Milestone: Engagement plan executed	0 days	Mon 3/02/25	Mon 3/02/25	4		0 days
6	1.1.2		**Define research Questions**	**6 days**	**Tue 4/02/25**	**Mon 10/02/25**			**4 days**
7	1.1.2.		Idenitfy the research gaps and formulate research questions	3 days	Tue 4/02/25	Thu 6/02/25	5	Thinley Dema,Ugyen Lhatshok,GitHub[1]	2 days
8	1.1.2.		Slelect appropriate theoretical framework	2 days	Fri 7/02/25	Sat 8/02/25	7	All Team	1 day
9	1.1.2.		Define scope and study boundaries	1 day	Mon 10/02/25	Mon 10/02/25	8	All Team	1 day
10	1.1.3		**Develop and Validate Review Protocol**	**3 days**	**Tue 11/02/25**	**Thu 13/02/25**			**2 days**
11	1.1.3.		Draft systematic review protocol	2 days	Tue 11/02/25	Wed 12/02/25	9,7,8	Thinley Dema,Ugyen Lhatshok	1 day
12	1.1.3.		Validate and approve review protocol	1 day	Thu 13/02/25	Thu 13/02/25	11	All Team,GitHub[1]	1 day
13	1.1.3.		Milestone: Research question defined and protocol approved	0 days	Thu 13/02/25	Thu 13/02/25	12		0 days
14	1.2		**Literature Collection**	**18 days**	**Fri 14/02/25**	**Tue 11/03/25**			**24 days**
15	1.2.1		**Idenitfy Relevant Literature**	**8 days**	**Fri 14/02/25**	**Tue 25/02/25**			**12 days**
16	1.2.1.		Select databases and information sources	2 days	Fri 14/02/25	Mon 17/02/25	13	Pema Gyamtsho,Thinley Rabgay,GitHub[1	4 days
17	1.2.1.		Develop search queries and strategies	2 days	Tue 18/02/25	Wed 19/02/25	16	Pema Gyamtsho,Thinley Rabgay, GitHub[1]	2 days
18	1.2.1.		Execute systematic search and document retrieval	4 days	Thu 20/02/25	Tue 25/02/25	17	All Team,GitHub[1]	6 days
19	1.2.2		**Screening and Selection**	**10 days**	**Wed 26/02/25**	**Tue 11/03/25**			**9 days**
20	1.2.2.		Apply inclusion and exclusion criteria	3 days	Wed 26/02/25	Fri 28/02/25	9,7,12,18	All Team	2 days
21	1.2.2.		Remove duplicates and irrelevant studies	2 days	Mon 3/03/25	Tue 4/03/25	20	Thinley Dema,Ugyen Lhatshok	3 days
22	1.2.2.		Create a study selection report	5 days	Wed 5/03/25	Tue 11/03/25	21	Thinley Rabgay	4 days
23	1.2.2.		Milestone: Literature screening & selection completed	0 days	Tue 11/03/25	Tue 11/03/25	22		0 days

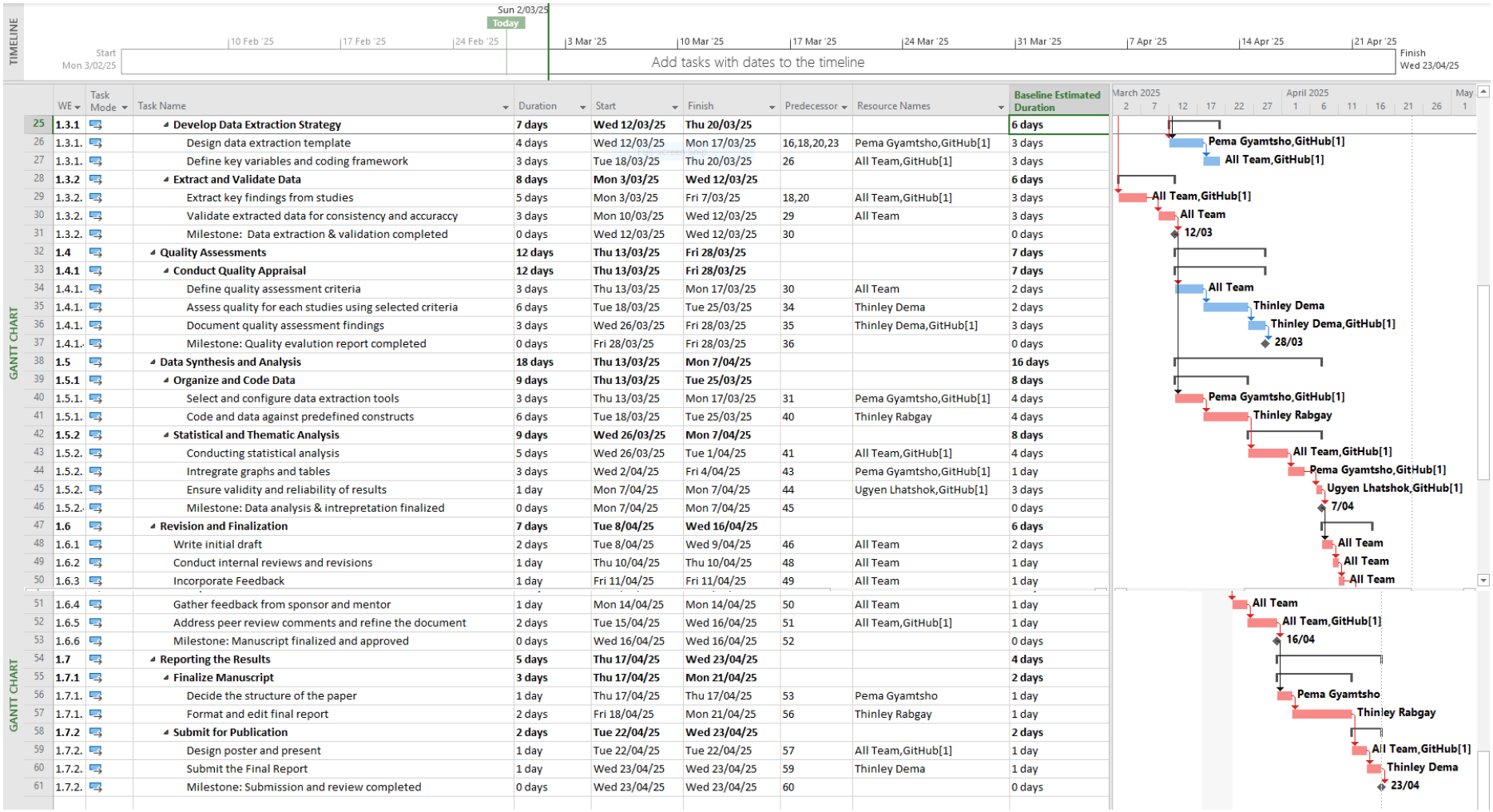


Figure 5.1: A Gantt Chart featuring required components of critical project planning.

5.1. Critical Path Analysis

Activity	Activity Description	Predecessor	Duration
A	Project Initiation	Na	10
B	Literature Collection	A	18
C	Data Extraction	B	14
D	Quality Assessment	C	12
E	Data Synthesis and Analysis	C & D	18
F	Revision and Finalization	E	7
G	Reporting the Results	E & F	5

Table 5.1: Critical path analysis

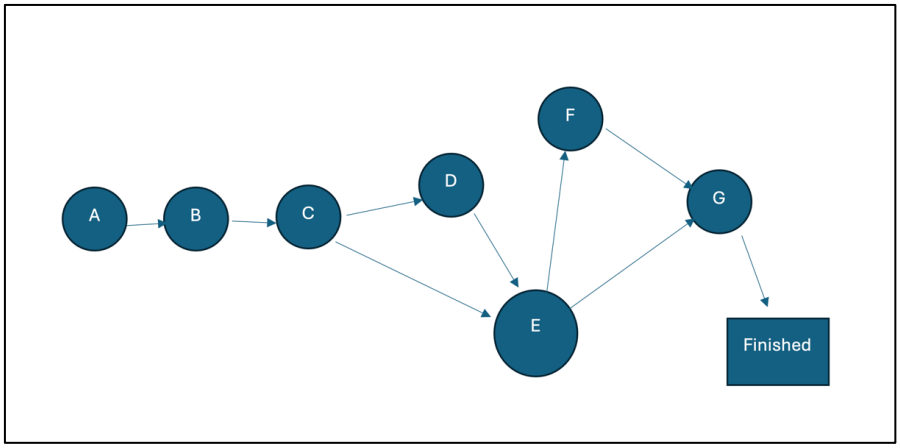


Figure 5.1: Activity on the Node (AON) Network Diagram

Path no	Path Sequence	Path Calculation	Total Days
Path 1	A+B+C+D+E+F+G	10+18+14+12+18+7+5	84
Path 2	A+B+C+D+E+G	10+18+14+12+18+5	77
Path 3	A+B+C+E+F+G	10+18+14+18+7+5	72
Path 4	A+B+C+E+G	10+18+14+18+5	65

Table 5.2: Critical path calculation

After a thorough analysis of all potential project pathways, it has become evident that path 1 is the most critical route for our project's success. This path distinguishes itself as the longest and most complex, involving multiple key milestones and tasks essential to the overall timeline. By prioritising and focusing on this critical pathway, we can ensure that our project progresses efficiently and effectively, minimising potential delays in achieving our objectives.

6. Project Governance

6.1. Organisational Chart

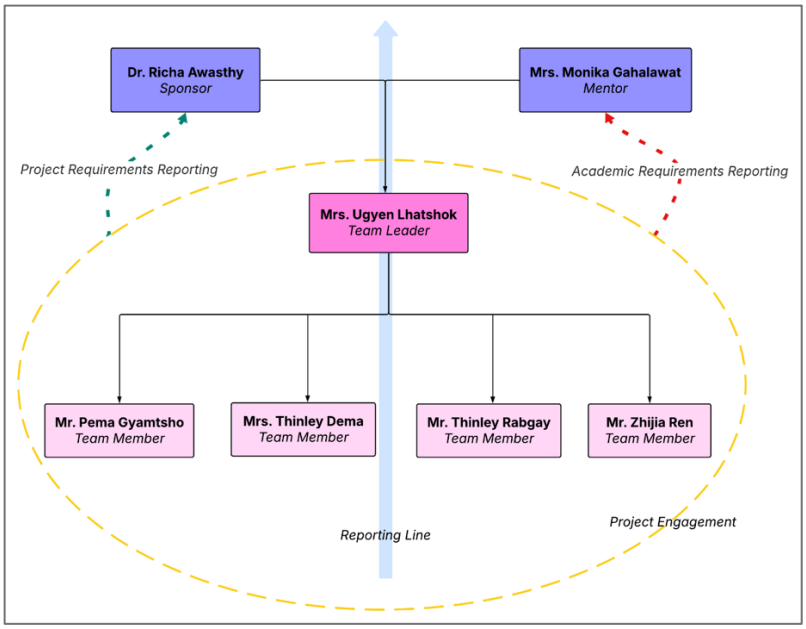


Figure 6.1: Organisational with upward reporting line

6.2. Roles and Responsibilities

Sl	Name	Role	Responsibilities
1	Ugyen Lhatshok	Team Leader	• Serve as a contact person with sponsor and mentor • Manage the project plan • Distribute the tasks among the members • track the whole project and individual progress
2	Pema Gyamtsho	Team Member	• Draft and review the project • Contribute to literature reviewing • Contribute and complete the project • Keep a systematic record of papers reviewed • Design and format the paper in academic style • Take active participation in the project discussion • Systematic record of the project process (GitHub)
3	Thinley Dema		
4	Thinley Rabgay		
5	Zhijia Ren		

Figure 6.1: Team roles and responsibilities

Note: Depending on the progress of the project, the responsibilities of each team member may change and will not be limited to the above-mentioned tasks.

7. Risk Analysis and Management Plan

a. Probability Rating	
Very low-10% estimated probability	1
Low- 30% estimated probability	2
Medium-60% estimated probability	3
High- 80% estimated probabilty	4
Very High- 95% estimated probability	5

Table 7.1: Probability rating class

b. Impact Rating				
Very low 1	Low 2	Medium 3	High 4	Very High 5

Table 7.2: Impact rating class.

Risk Prioritisation Matrix					
Probability	Impact				
	Very Low	Low	Medium	High	Very High
Rare	1	2	3	4	5
Possible	2	4	6	8	10
Likely	3	6	9	12	15
Often	4	8	12	16	20
Frequent/Almost Certain	5	10	15	20	25

15-25	Extreme Risk
7-12	High Risk
4-6	Moderate Risk
1-3	Low Risk

Table 7.3: Risk Prioritisation Matrix

7.1. Risk Register

Sl. No.	Risk Category	Risk Description	Risk Matrix			Owner	Mitigation Plan	Contingency plan
			Proba-bility	Impact	Severity			
1	Limited access to key databases	The project assumes unrestricted access to databases such as IEEE Xplore, ACM Digital Library, Scopus, and ScienceDirect. However, institutional restrictions or subscription limitations may impact comprehensiveness of the study.	3	4	7	Pema Gyamtsho	Secure access through institutional affiliations (University of Canberra)	Adjust search strategies to focus on accessible databases
2	Insufficient quantity of relevant studies	If only limited number of peer-reviewed studies are available, it may hinder the ability to draw robust conclusions, affecting the generalizability and relevance of the research findings.	3	2	5	Pema Gyamtsho	Expand search terms and use broader keywords while ensuring relevance. Include grey literature such as conference proceedings, theses, and technical reports.	If data remain limited, consider including related studies with explicit justification for their inclusion. Conduct a narrative review if a full systematic review proves unfeasible.

							Apply snowballing techniques by reviewing the reference lists of key studies.	
3	Challenges in synthesizing diverse teaching methodologies	The wide variety of teaching strategies, study designs, and outcome measures across the literature may result in difficulties in synthesizing findings coherently, potentially compromising the clarity and consistency of the conclusions drawn.	3	3	6	Thinley Dema	Use thematic analysis or to group similar methodologies. Establish clear criteria for categorizing and comparing different strategies. Engage sponsor for validation of synthesis approaches.	Adjust the synthesis method (e.g., switch from quantitative meta-analysis to qualitative synthesis) if heterogeneity is too high. Present findings as separate thematic clusters rather than forcing integration.
4	Data Management and Organization Challenges	Managing large volumes of literature, tracking data extraction, and ensuring proper organization could pose challenges, increasing the risk of data loss, duplication, or inconsistencies, ultimately affecting the quality and efficiency of the review process.	3	3	6	Ugyen Lhatshok	Utilize systematic review management software (if available and if needed) for efficient data handling. Implement standardized data extraction forms and protocols.	Regularly back up data in secure cloud storage or institutional repositories.

							Maintain proper version control and file organization systems.	
5	Scope creep	The vast scope of the study may lead to the inclusion of broader software engineering topics or unrelated teaching strategies, potentially diluting the study's focus, causing deviations from the primary research objectives, and increasing the overall workload.	3	4	7	Ugyen Lhatshok & Thinley Dema	Clearly define scope boundaries and objectives at the project's outset. Use a scope management plan with formal change control processes for evaluating scope adjustments. Conduct regular scope reviews to ensure alignment with research objectives.	If scope expansion is necessary, reassess the project timeline and resource requirements. Defer lower-priority tasks or topics to future research phases.
6	Time constraints	Limited time for conducting a comprehensive systematic review, including searching, screening, analysing, and synthesizing literature, may result in rushed processes, overlooked studies, or incomplete analysis, impacting the overall	4	5	9	Thinley Rabgay	Develop a detailed project schedule with realistic timelines and key milestones. Prioritize tasks using frameworks like MoSCoW (Must-have,	Reallocate tasks among team members if bottlenecks occur. Focus on delivering core outcomes if time becomes critically constrained.

		quality and credibility of the project.					Should-have, Could-have, Won't-have). Allocate buffer time for critical tasks to accommodate unexpected delays.	
7	Communication Breakdown	Inadequate communication among team members and stakeholders can put significant risk to effective communication leading to misunderstandings, delays, and conflicts, that will negatively impact the project's success.	2	4	6	Thinley Rabgay	Establish clear communication channels (e.g., Slack, Microsoft Teams) and protocols. Schedule regular meetings for progress updates and issue resolution. Clearly define roles, responsibilities, and reporting structures.	Implement escalation procedures for unresolved communication issues. Designate a communication coordinator to streamline information flow among team members.

Table 7.1: Risk register

8. Stakeholder Analysis and Communications Plan

Stakeholder analysis identifies the key individuals, groups, or organisations affected by or influencing the project. This method involves systematic gathering and analysis of quantitative and qualitative information about stakeholders to determine different interests that should be considered for the project (Project Management 2017). It helps prioritise engagement efforts and tailor communication strategies to meet stakeholders' needs.

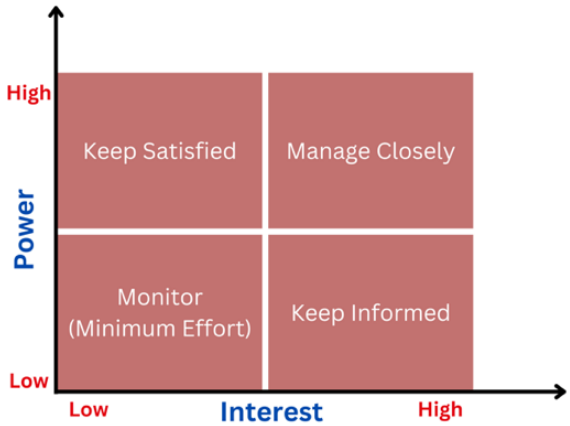
8.1. Stakeholder identification

The following stakeholders have been identified for this project:

Stakeholder Group	Role/Interest	Influence Level	Impact Level
Project Team	Researchers conducting the systematic review.	High	High
Sponsor	Overall sponsoring of the project, defining scope and outcome of the project and monitoring progress and result of the project.	High	High
Mentor	Mentoring researchers, reviewing progress of academic aspect of the project and overall guidance and support.	Low	High
Tutor/Lecturer	Course content and overall progress of the unit	Low	Low
University of Canberra	University offering resources (library, online portals, repository, subscription to databases, software licenses, meeting venues, academic accreditation)	Low	High

Table 8.1: Identification of stakeholders with their roles or interest, influence level and impact level

8.2. Stakeholder prioritisation



Using a Power-Interest Grid, stakeholders are prioritised based on their level of influence and interest in the project:

Figure 8.1: Power-Interest Grid (Management.com 2003)

High Power, High Interest *(Manage Closely)*	High Power, Low Interest *(Keep Satisfied)*	Low Power, High Interest *(Keep Informed)*	Low Power, Low Interest *(Monitor)*
Project Team Sponsor	University of Canberra	Mentor	Tutor/Lecturer

Table 8.2: Power-Interest Grid

8.3. Communication Objectives

One of the successes of the project is directly dependent on the efficiency and effectiveness of communication (Project Management 2017). It is crucial to strategically plan richness in communication that can convey a variety of information easier and faster (Project Management 2017).

The objectives of communication for this project are to:

- Keep stakeholders informed about project progress and findings.
- Gather feedback to ensure the review addresses relevant issues.
- Disseminate results effectively to maximise impact.

8.3.1. Communication Methods and Frequency

Stakeholder Group	Communication Method	Frequency	Responsible Party
Project Team	Weekly team meetings, shared project documents.	Weekly	Project Lead
Sponsor	Weekly meetings to update progress and seek guidance	Weekly	Project Lead
Mentor	Need-based meetings for support and directions	Need-based	Project Lead

Table 8.3: Communication methods and frequency

8.3.2. Communications Channels

The following communication channels will be used to share any updates related to the research project:

8.3.2.1. **Email:** For formal updates and sharing of reports.
8.3.2.2. **Meetings:** For collaborative discussions, guidance and feedback using tools such as Microsoft Teams, WhatsApp instant messenger, Calendar app, and online documentation apps.
8.3.2.3. **Websites:** To disseminate findings from the research report to a wider audience.

8.3.2.4. **Presentations and webinars**: To disseminate findings from the research report to a wider audience.

8.3.2.5. **E-newsletters:** To disseminate findings from the research report to a wider audience.

9. Quality Assurance and Measures of Success

Quality assurance (QA) is of utmost importance to any project to ensure project deliverables meet the required accuracy standards, reliability, and relevance for the project with the predefined research framework and strategies. Without a proper quality control tool and quality assurance system in place, it has been found that many academic findings are not reproducible (Camargo and Olivares 2016). Therefore, the process of assuring quality assurance for the project will focus on succinct criteria and specific measures to assess and evaluate project outcomes and deliverables. Ultimately, these measures are expected to ensure that research on systematic review is conducted rigorously, adhering to Kitchenham's guidelines and SPIDER framework that will produce reliable, meaningful and impactful actions and insights for the learning communities of software architecture.

9.1. Quality Goals and Objectives

By adhering to these quality goals and objectives, the project will produce valuable insights and actionable recommendations for educators and researchers in the field.

9.1.1. To ensure proven standards and methodologies to conduct systematic reviews such as Kitchenham's guidelines and SPIDER framework to ensure transparency, reproducibility, and credibility.

9.1.2. To produce high-quality outputs and deliver comprehensive insights by identifying and synthesising key themes, challenges, and best practices in software architecture teaching.

9.1.3. To produce a well-structured, clear, and actionable final report that meets academic and professional standards.

9.1.4. To ensure the findings are relevant and useful for educators, researchers, and practitioners in software architecture.

9.1.5. To complete the project on time, within scope, and in compliance with ethical research practices.

9.2. Quality Assurance criteria

The quality assurance criteria for this project are based on established standards for systematic reviews and academic research which are critical to guide the interpretation of findings and recommendations for further research (Kitchenham and Charters 2007). These criteria will also guide the development, execution, and evaluation of the project deliverables. The key QA criteria include:

9.2.1. Consistency and Methodological Reliability: The systematic review will follow a well-defined and transparent methodology, adhering to guidelines such as Kitchenham's principle and SPIDER framework. The research questions, inclusion/exclusion criteria, and search strategy will be clearly defined and justified. Finally, the data extraction and synthesis processes will be systematic, reproducible, and unbiased.

9.2.2. Relevance and Impact: The systematic review must address the goals and objectives of the project as defined by the research questions. The findings will provide actionable insights for educators, researchers, and practitioners in the field. The deliverables would contribute to the advancement of software architecture education and inform future research directions.

9.2.3. Accuracy and Completeness: The literature search will be comprehensive, covering relevant databases, journals, and conference proceedings according to the predefined search criteria and protocols. All included studies will be accurately summarised and analysed for a complete and unbiased synthesis of the evidence in the final report.

9.2.4. Clarity and Presentation: The deliverables will be well-organised, clearly written, and accessible to the target audience. Various visual aids such as tables, charts, posters, and diagrams will be used to enhance understanding of the research findings. The report will have a discussion of limitations and implications for future research.

9.2.5. Timeline and Adherence to Schedule: The project will be completed within the agreed timeline, with milestones and deadlines met as planned. Any delays or deviations from the schedule will be documented and justified systematically.

9.2.6. Ethical Considerations: The review will adhere to ethical standards for academic research, including proper attribution of sources to avoid plagiarism. Any conflicts of interest, if any, will be disclosed.

9.3. Measure of Success

The success of the research project will be evaluated based on the following measures, which align with the QA criteria:

Criteria	Measure	Success Indicator
a. Consistency and Methodological Reliability	Adherence to Kitchenham's principle and SPIDER framework.	A detailed methodology section in the final report, a clear explanation of the search strategy, inclusion/exclusion criteria, and data extraction process.
b. Comprehensiveness of Literature Search	Number of databases searched, studies identified, and studies included in the review.	A comprehensive search strategy that covers major databases and results in a sufficient number of high-quality studies for analysis.
c. Depth of Analysis and Synthesis	Use of thematic analysis or meta-synthesis to identify key themes, patterns, and gaps in the literature.	A well-structured synthesis section in the final report that provides meaningful insights into software architecture teaching practices, challenges, and innovations.
d. Impact and Relevance	Number of actionable recommendations provided for educators, learners and researchers.	A discussion section that highlights the practical implications of the findings and suggests specific areas for future research.
e. Clarity and Presentation	Feedback from peer reviewers or stakeholders on the clarity and organization of the report.	Positive feedback indicating that the report is well-written, easy to understand, and effectively communicates the findings.
f. Timeliness	Adherence to the project deadlines and milestones.	Completion of all project deliverables within the agreed timeframe.
g. Ethical Compliance	Proper citation of all resources used for the research project.	A clean plagiarism report (e.g., using Turnitin or similar software) and a properly formatted reference list.

Table 9.1: Measure of success

9.3.1. Deliverable-Specific Quality Standards

The project deliverables will be evaluated against the following quality standards:

Project Deliverables	Quality Standards
Research Protocol	• Clearly defined research questions and objectives. • Detailed search strategy, including databases, keywords, and filters. • Inclusion/exclusion criteria and rationale. • Plan for data extraction and synthesis.
Literature Search Results	• Comprehensive list of studies identified through the search process. • Documentation of any deviations from the original search strategy.
Data Extraction and Quality Assessment	• Completed data extraction forms for all included studies. • Summary table of study characteristics (e.g., author, year, methodology, key findings). • Quality assessment scores for each study.
Synthesis and Analysis	• Thematic analysis or meta-synthesis of the findings. • Identification of key themes, patterns, and gaps in the literature. • Visual aids (e.g., tables, charts) to support the analysis.
Final Report	• Clear and concise executive summary. • Well-structured report with specific sections and sub-sections. • Discussion of limitations and implications for future research. • Well formatted references and appendices.

Table 9.2: Project deliverables and quality of standards

10. References

Aromataris E and Pearson A (2014) 'The systematic review: an overview', *AJN The American Journal of Nursing*, 114(3);53-8,

Bhargava M, Gandhi M, Qayumi E, Phelps V, Zhuang Z, Nathan N, Bernard C and Garcia D (2025) *GradeSync: A Tool for Automating Incomplete Processing to Support Mastery Learning*.

Camargo RS and Olivares IRB (2016) 'Quality management system for application of the analytical quality assurance cycle in a research project', *Journal of Physics: Conference Series*, 733(1);012046, doi:10.1088/1742-6596/733/1/012046

Cooke A, Smith D and Booth A (2012) 'Beyond PICO: the SPIDER tool for qualitative evidence synthesis', *Qualitative health research*, 22(10);1435-43,

Ellis J and Garvey EG (2011) 'Introduction: Teaching Digital Media', *Transformations*, 22(1);11-2,4,

Garousi V, Giray G, Tuzun E, Catal C and Felderer M (2020) 'Closing the Gap Between Software Engineering Education and Industrial Needs', *IEEE Software*, 37(2);68-77, doi:10.1109/MS.2018.2880823

Keele S (2007) *Guidelines for performing systematic literature reviews in software engineering*, Technical report, ver. 2.3 ebse technical report. ebse

Kitchenham B (2004) 'Procedures for performing systematic reviews', *Keele, UK, Keele University*, 33(2004);1-26,

Kitchenham B and Brereton P (2013) 'A systematic review of systematic review process research in software engineering', *Information and software technology*, 55(12);2049-75,

Kitchenham B and Charters S (2007) 'Guidelines for performing Systematic Literature Reviews in Software Engineering', 2,

Management.com P (2003) *Stakeholder Analysis using the Power Interest Grid*, Project Management Institute, accessed February 26.

Prabowo YD, Kristijantoro AI, Warnars H and Budiharto W (2021) 'Systematic literature review on abstractive text summarization using kitchenham method', *ICIC Express Letters, Part B: Applications*, 12(11);1075-80,

Project Management I (2017) *A guide to the project management body of knowledge (PMBOK guide)*, 6th edition. edn, Project Management Institute, Newtown Square, Pennsylvania.

Ramachandran K and Karthick K (2019) 'Gantt chart: An important tool of management', *Int. J. Innov. Technol. Explor. Eng*, 8(7);140-2,

Rupakheti CR and Chenoweth SV (2015) *Teaching Software Architecture to Undergraduate Students: An Experience Report*.

Souha A, Ouaddi C, Benaddi L, Naimi L, Bouziane EM and Jakimi A (2025) 'MDE-Based Approach for Accelerating the Development of Recommender Systems in Smart Tourism', *IEEE Access*, 13;31615-29, doi:10.1109/ACCESS.2025.3543058

Tan KY, Tan SC and Chuah TC (2025) 'A Multi-Phase DRL-Driven SDN Migration Framework Addressing Budget, Legacy Service Compatibility, and Dynamic Traffic', *IEEE Access*, 13;33202-19, doi:10.1109/ACCESS.2025.3543236

Theile CM and Beall AL (2024) 'Conducting a Systematic Review of the Literature', *Journal of Dental Hygiene*, 98(2),