KUALA LUMPUR UNIVERSITY OF SCIENCE AND TECHNOLOGY

(formerly known as Infrastructure University Kuala Lumpur, IUKL)

CSC1015

ETHICS AND PROFESSIONAL CONDUCTS

COURSEWORK 1 (CW1)

Career Research and Personal Value System

Submitted by:

ISMAIL ABDIRISAQ ABDIQANI

Bachelor of Software Engineering (Honours)

Submitted to:

Ts. Hasnani Hassan

Date of Submission: 20 May 2026

ACKNOWLEDGEMENT

Firstly, I am extremely thankful to Ts. Hasnani Hassan for developing this coursework that has provided me a meaningful way to begin my early professional development as an undergraduate student in Software Engineering. This structured research process has not only increased my knowledge of the career area I selected (Part A) but it also allowed me to reflect on my own values, strengths and weaknesses.

In addition, I wish to thank all the authors referenced in the literature and scholarly references that have supported each section of this report. The references were appropriately acknowledged by citation according to APA format.

Lastly, I want to say thank you to my family who encouraged me along the way and sacrificed their time so that I may achieve this educational goal.

TABLE OF CONTENTS

Acknowledgement **2**

Table of Contents **3**

PART A: Information Technology Career Research

1.0 Introduction 4

2.0 Historical Overview of Software Engineering 4

3.0 Education and Certification Preparation 6

4.0 Entry-Level Employment Patterns, Responsibilities and Career Pivots. 8

5.0 Career Advancement, Continuous Learning and Washout Risks ... 10

6.0 Job Market: Companies, Industries and Opportunities 13

7.0 Conclusion 15

8.0 Lesson Learned and IEEE Professional Codes of Conduct 16

9.0 References 18

PART B: SWOT Analysis and Personal Development Plan

10.0 Reflective Statement 19

11.0 SWOT Analysis Summary 20

12.0 Challenges to Overcome and Talents to Capitalize..... 21

13.0 Personal Development Plan (PDP) 22

PART C: Resume and Cover letter

14.0 Resume.....28

15.0 Cover Letter.....29

PART A: INFORMATION TECHNOLOGY CAREER RESEARCH PROJECT

Career: Software Engineer

1.0 Introduction

This paper presents a thorough investigation of career opportunities in the field of Software Engineering. The Software Engineering Profession emerged as an academic discipline limited in scope; it is today recognized as a vital industry that supports many aspects of our lives. I have chosen to focus my investigation on the profession because I believe that Software Engineering is at the foundational level of how society functions today – whether through Financial Systems, Telecommunications, Healthcare or Government. Software Engineers are employed in nearly every area of business and government. The next part of this paper explores each of the above areas along with other important dimensions of the career opportunity. These include:

- An overview of the history of the software engineering profession.
- A review of the education and certifications available to students interested in pursuing a career as a software engineer.
- A description of the various paths open to new graduates entering the software engineering workforce.
- A discussion of how software engineers can advance in their careers and the potential for career stagnation.
- A review of current trends within the employment environment of Malaysia specifically related to technology.

Lastly, there will be a final chapter discussing what was learned during this project and those experiences' connection to IEEE's professional codes and conduct.

2.0 History of Software Engineering

2.1 Background: The Software Crisis (1950s - 1960s)

In the beginning, programming computers was done by a handful of highly skilled people who saw writing code as a form of art; they did not see themselves bound by rules or standards. When companies commissioned large-scale software projects, they quickly realized that relying upon unstructured and informal methods for creating these programs would be

insufficient. The result was that the vast majority of projects ran over budget, were late and produced defective products, which is why this period in time is known as the "software crisis" (Naur & Randell, 1969).

A turning point occurred in 1968, when NATO held its first international conference on software problems in Garmisch, Germany. At this meeting, computer scientists agreed to formally define the problem, and to develop formal solutions. In doing so, they established a new name for this type of work: "Software engineering." They used this name to emphasize that developing software requires the same rigor, discipline and scientific methodology as traditional engineering disciplines (Naur & Randell, 1969).

2.2 Formalization and Methodology (1970s–1980s)

The 1970's are where the first formalized approaches to software development began to take shape. In 1970, Winston Royce created a methodology called the Waterfall model. This model was a sequential process divided into phases. It provided a basis for organizing how we develop software. Also, during the late 1960's and early 1970's were Edsger Dijkstra's push for structured programming and Niklaus Wirth's work creating the programming language PASCAL which advocated using coding techniques that produce readable, maintainable code versus just writing ad-hoc scripts (Pressman & Maxim, 2020).

Margaret Hamilton is one of the many famous people from this period. Hamilton headed the group to develop the onboard flight computer system for NASA's Apollo missions. Not only did she deliver software capable of working properly under conditions critical to life, but she was also one of the first developers to recognize software development should be viewed as an engineering discipline (Sommerville, 2016). Her efforts have earned her much credit for bringing legitimacy to the idea of "software engineering" (in practice).

2.3 Object-Orientation and the Web Era (1985–2000)

The OOP paradigm emerged in the early 80's as a widely adopted

methodology. The emergence of C++ from Bjarne Stroustrup in 1983 and the introduction of Java in 1995 changed the way people think about developing software by promoting modular design, reuse and abstraction. In addition, the development of the World Wide Web by Tim Berners-Lee in 1991 created a brand-new area of opportunity for the software engineer and allowed for the creation of the web development community which greatly increased the scope of what it means to be a software engineer.

2.4 Agile Revolution and Cloud Computing (2000–2015)

The year 2000 was a time for an historic culture shift in software engineering. That is when the Agile Manifesto of 2001 came into being. It was developed by Kent Beck along with sixteen other developers. They sought to abandon the strict nature of Waterfall models of software engineering in favor of the flexible iterative model, collaborating with customers, and adapting to changes in requirements (Pressman & Maxim, 2020). Agile models like Scrum and Kanban are now widespread across business. As well as this shift in software development methodology there were also fundamental shifts in how we deliver and architect software. This was due to the advent of Amazon Web Services (AWS) in 2006 followed by the launch of Google Cloud Platform and Microsoft Azure. These three companies launched the cloud-computing era.

It should be noted that Linus Torvalds, who created the Linux Kernel in 1991 and developed Git in 2005, have had the greatest influence on how software is delivered. Virtually all professional software development occurs using Git. Additionally, nearly every server, cloud service and embedded device uses a version of Linux.

2.5 The Present: AI-Augmented Engineering (2016–Present)

The new generation is one where the convergence of Software Engineering and Artificial Intelligence will shape the way we work. Development tools, like GitHub Copilot (which utilizes Large Language Models) are already automating many aspects of routine code generation; therefore, engineers are transitioning towards using their time for tasks such as system design, architecturally

thinking about how components interact with each other and decomposing problems into manageable pieces (Stack Overflow, 2023).

3.0 Education and Certification Preparation

3.1 Academic Pathways

The bachelor's degree that is most considered acceptable for employment as a software engineer is either an undergraduate degree in Information Technology, Software Engineering, or Computer Science. These three degrees have the greatest alignment with professional practice since both provide students with the foundational knowledge and skills necessary to perform as well in the workplace. In addition, the Software Engineering degree has a higher emphasis on specific topics of relevance to professional practice than do other degrees in computing, such as software architectures; project planning, development, and execution; quality control; and professional ethics (IEEE Computer Society, 2014).

Additionally, post-graduate education can also be beneficial when pursuing a career path towards senior technical leadership or as an engineering manager. Typically, this type of advanced education is completed by individuals who already have extensive work experience in the field of software engineering.

3.2 Critical Elective Courses

To be as prepared for a career as possible, students can expand on their general studies by taking courses that include:

- Data Structures and Algorithms — This will give them the tools to do well at tech interviews, and to create the most effective systems.
- Database Management Systems — The ability to use both relational databases (SQL) and non-relational databases (NoSQL) are requirements in almost every engineering position.
- Computer Networking and Security — These two concepts are important for learning how to build distributed systems and designing systems securely.
- Cloud Computing and Infrastructure — As cloud computing continues to grow and become the standard in enterprise environments, experience

with Amazon Web Services (AWS) or Microsoft's Azure is becoming an expectation.

3.3 Professional Certifications

Industry certifications can be shown as a credible source of evidence that is recognized within the industry and which validates your knowledge and skills in relation to specific areas of technical competence. Therefore they will add strength to the candidates overall profile especially when entering or transitioning into an Entry to mid level role (MDEC, 2022). The high value certifications are listed below:

- Aws certified developer – Associate: this is relevant to the aws based cloud application development area.
- Google professional cloud developer: this certification is recognized for those with expertise in developing systems using google cloud platform (GCP).
- Microsoft certified: azure developer Associate: this is generally applicable for enterprise environments where microsoft technologies are used.
- Oracle certified professional java developer: this is a strong indication of your ability to perform in backend java engineering roles.
- Linux Foundation Certified Engineer (LFCE): this demonstrates your advanced abilities in relation to linux systems.
- CompTIA security+: this is a general Entry-level cybersecurity certification; it is therefore applicable to all engineering specializations.
- Certified Kubernetes Administrator (CKA): this is highly regarded in DevOps and platform engineering.

Finally, there are also programs available from the Malaysia Digital Economy Corporation (MDEC) in addition to HRDF claimable courses related to technology, these can also be seen as supplements to formal education, and are recognized locally by employers.

3.4 Experience-Based Preparation

A student's academic record alone is not enough. In many cases, employers will place a higher emphasis on those with demonstrated hands-on experience

through their involvement in internships; open source contribution; and/or personal projects that can be found on GitHub or Hackathons. A demonstration of working software is very often given greater weight during an employer's first review process than a candidate's GPA (Stack Overflow, 2023).

4.0 Entry-Level Employment Patterns, Responsibilities and Career Pivots

4.1 Entry-Level Job Titles and Requirements

The most common entry level job titles in the field of software engineering include:

- Software Engineer I
- Associate Software Engineer
- Graduate Software Developer
- Junior Software Engineer

These positions require the following qualifications:

Typically, a bachelor's degree in some area of computer science, and a portfolio of your own projects or work you have done as part of school assignments. Internships help too but are becoming less important due to experience. A portfolio of personal or academic projects and an internship will be given preference over education alone.

Many employers also want to see that you can "do"; rather than just talk about what you did. If you can create something and show it off on GitHub, then you may get hired before someone who has a higher gpa.

4.2 Job Responsibilities at Entry Level

Junior software developers are generally expected to:

- Implement as per their assigned feature tasks from a technical perspective (with direction from Senior Developers).
- Develop unit and integration testing to verify that the code they have written is correct and will perform well.
- Determine, identify and solve software bugs or performance problems.
- Be an active participant in code review activities; submit your own code for review and participate in reviewing other's code.
- Assist with writing technical documentation related to the project. This can include but is not limited to: comment on code, document the

application programming interface (API) of the program, create diagrams of how the systems work together.

- Actively attend all Agile ceremonies (daily stand-ups, sprint planning, retrospectives and reviews) in which development team members are required to attend.
- Work closely with Product Managers, Designers and Quality Assurance Engineers within cross-functional teams.

4.3 Knowledge, Skills and Attitude Required

The following competency profile defines a hirable entry-level candidate:

Knowledge	Skills	Attitude
1 OOP programming language experience (C# , Python, Java or JavaScript)	Logic of problem-solving through breaking down problems into smaller parts	Curiosity about the subject matter and a desire to continue learning
Use of Version Control through GIT	Troubleshooting & identifying source issues (debugging)	Openness to new ideas and willingness to listen to others in order to grow as an individual (humility)
Understanding of fundamental Data Structures and Algorithms	Ability to clearly communicate thoughts and opinions both verbally and in writing	Accountability for Professionalism and Reliability
Familiarity with SDLC and Agile/Scrum methodology	Experience working on a team collaboratively	Ability to remain calm and work well under pressure
Understanding of Relational Databases and Basic SQL	Ability to manage time appropriately when there are deadlines	Commitment to producing quality code that is ethical.

Understanding of REST API concepts	Ability to quickly learn without direct supervision.	Supportive Collaborative Attitude

4.4 Career Pivot Options

Software Engineering is a solid platform from which one can make successful laterals. Regardless of whether there is a downturn in employment due to technological changes, the individual's evolving interests, or changes in the industry, many individuals have successfully made these types of moves and been able to do so because they have had prior engineering experience.

1. Product Manager: Individuals that understand how products work at a very detailed level and have developed good communication skills will be in great position to move to a Product Manager role. They will be able to bridge both the technical and business aspects of their company.

2. Data Scientist / Machine Learning Engineer: Upskilled in Python, Statistics and Machine Learning frameworks (TensorFlow, PyTorch) will give individuals an easy transition as they are currently using large data sets.

3. DevOps / SRE: If an engineer has a passion for infrastructure, CI/CD pipelines, Cloud Operations then they could consider transitioning into this area. This area is growing rapidly and has high demand.

4. Cyber Security Engineer: There is a movement towards specializing in Cyber Security Engineering and there are certifications available through CompTIA (Security+) and EC-Council (CEH). These certifications are very relevant to companies involved in Banking, Government, Telecommunications, etc.

5. Technical Consultant / Solutions Architect: More senior engineers use their broad base of knowledge to help guide other companies on strategic decisions related to technology. Many companies like

Accenture, Deloitte and IBM pay well for services provided by Senior Engineers in this capacity.

6. IT Entrepreneur: The analytical thinking/problem solving/skills acquired while learning Systems Design/Engineering enable individuals to create technology based businesses.

7. Academic Researcher/Lecturer: Those engineers that have strong theoretical bases can further their education through Post Graduate studies and enter the Academic arena.

5.0 Career Advancement, Continuous Learning and Washout Risks

5.1 The Software Engineering Career Ladder

The software engineering career ladder is one of the most well-defined in the technology industry, offering clear progression milestones across both individual contributor (IC) and management tracks:

Level	Typical Experience	Key Characteristics
Junior Software Engineer	0–2 years	Implements features under supervision; actively learning.
Mid-Level Software Engineer	2–5 years	Works independently; mentors juniors; owns components.
Senior Software Engineer	5–8 years	Leads system design decisions set technical standards.
Lead / Staff Engineer	8–12 years	Cross-team technical influence defines architecture.

Principal / Distinguished Engineer	12+ years	Company-wide technical strategy; deeply specialized.
Engineering Manager / Director	Varies	Management track; people's leadership and delivery ownership.
VP Engineering / CTO	Senior leadership	Organisational technical vision and strategic direction.

5.2 Continuous Education Requirements

Software engineering is one of the few professions where the pace of evolution of key tools and methodologies is as fast. As such continuous learning isn't just an option – it's a requirement (IEEE, 2020) of remaining competitive. As such successful engineers will have invested in a variety of ways of acquiring additional knowledge:

- Certifications: The acquisition of industry certification through programs such as AWS, Microsoft Azure, Google Cloud Platform, or Kubernetes certifications demonstrates your familiarity with modern commercial systems.
- Attending Conferences: Events like AWS re:Invent, Google I/O, PyCon, and local developer events (e.g. DevFest KL), are great places to see what's happening now in terms of emerging trends.
- Acquiring Education Through Online Platforms: Resources like Coursera, Pluralsight, O'Reilly Learning, and Udemy allow you to structure how and when you learn specific topics.
- Contributing To Open Source Projects On GitHub: By contributing to public repositories on GitHub you'll develop a reputation for being able to work collaboratively with others, you'll also be able to keep yourself aware of emerging standards by working directly within the source code.
- Reading About Technology: Accessing articles published in resources like the ACM Digital Library, IEEE Xplore, and reading about emerging

topics on Hacker News will help you continue to grow technically while staying informed about technological advancements.

- Mentorship And Feedback: Working under the guidance of someone else who has more experience than you will accelerate your technical development. Providing mentorship to someone less experienced than you will also help them to develop faster and create a mutually beneficial relationship.

5.3 Risks of Being "Washed Out"

Although software engineering provides many benefits, there are still serious risks associated with being washed out of this field. These risks come from various sources, but they can be categorized into five distinct areas of concern:

- Technology Obsolescence: If you limit yourself to developing applications using a single set of technologies and refuse to update those skills based upon changes in the industry then you may become irrelevant. For example, cloud native development has changed the landscape for companies building distributed applications on premise. Anyone who has developed their skill sets solely around building applications that run exclusively on premise is likely going to find themselves with reduced job prospects.
- Lack Of Skill Development In Areas Outside Of Coding: While technical ability is important, simply having that ability is unlikely enough to get you promoted past a mid-level position. If you're unable to explain complicated concepts to non-developers; if you can't take constructive criticism from developers; or if you can't build relationships outside of your immediate team then your chances of getting promoted significantly reduce.
- Building A Poor Reputation Due To Your Code: Developing a pattern of producing untested code; writing poor documentation; or creating code that no one wants to maintain will damage your professional reputation. This makes it difficult to transition internally.
- Psychological Burnout: Many companies especially start-ups create an environment where employees feel pressured to constantly produce results. Employees who don't establish boundaries and consistently push

themselves too hard will eventually suffer from mental health issues or physical exhaustion that could potentially shorten their career.

- Not Building Relationships With Other Professionals Within The Industry: More often than ever engineering teams are made up of people who are skilled in different areas. Engineers who isolate themselves from other professionals will miss out on opportunities for growth, for sponsorship for new role opportunities.
- Being An Expert At Something That Has Become Obsolete: Having deep knowledge of something that has been replaced or diminished due to market forces (for example: certain legacy enterprise frameworks) creates barriers to finding better employment.

6.0 Job Market: Companies, Industries and Opportunities

6.1 Global Employment Landscape

The global demand for software developers has grown at an extraordinary pace, both in terms of the number of positions available and the long-term trends. According to projections by the U.S. Bureau of Labor Statistics, employment opportunities for software developers will grow approximately 25 percent through 2032 significantly faster than the national average for all other jobs (U.S. Bureau of Labor Statistics, 2024) making this one of the fastest growing job categories. A survey conducted by Stack Overflow, which includes responses from over 90,000 participants in 185 different countries, further supports the fact that software engineers continue to be the best paid and most sought-after professionals in every industry (Stack Overflow, 2023).
Technology-based companies are among the major international employers. Examples of some of these companies include:
-Alphabet (Google), which employs over 140,000 people and has software development as its primary function.
- Microsoft employs over 200,000 people globally and is one of the top employers for engineers working on cloud computing systems, enterprise applications, and artificial intelligence platforms.
-Amazon (AWS and Amazon.com) is currently the largest employer of cloud computing engineers.

Each company also has multiple locations throughout the globe where they hire engineers to work within their distributed global networks.

6.2 Malaysian Job Market

Malaysia represents an increasingly attractive and expanding marketplace for software developers due to the Malaysian government's commitment outlined within its Malaysia Digital Economy Blueprint (My DIGITAL) that sets a goal of creating 500,000 digital technology employees by 2025 (MDEC, 2022).
The large technology talent gap in Malaysia will continue to be a driving force behind continued high demand for qualified professionals in the following fields:

6.2.1 Telecommunications

Major Telco's in Malaysia have made considerable investments in their digital transformations and software based Network Infrastructure. Examples include:

- Maxis Telecommunications Sdn Bhd: Developing In-House Engineering Capabilities to support Network Automation, Development of Digital Products and Cloud Native Applications.
- Telekom Malaysia Berhad (TM): National Telco Provider, Currently Hiring in various positions for digital infrastructure and cloud-based developments.
- Axiata Group: Parent Company of Celcom; has created numerous engineering center locations throughout Asia that are responsible for supporting digital services.
- Digi Telecommunications (Now merged with Celcom under name "Celcom Digi"): Major Recruiter for network software & digital platform engineers.

6.2.2 Banking, Financial Services and Fintech

- Maybank & CIMB: Both have large in-house technology departments that employ engineering teams to develop their own digital banking platforms.
- PayNet (Payments Network Malaysia) : National payment clearinghouse , also major employer for back-end and security engineers.
- Grab Financial Group & BigPay : Regional fintechs with engineering centers in KL.

- RHB Bank, AmBank & Hong Leong Bank: Each are currently implementing various forms of digital transformation programs as they expand their engineering talent pools.

6.2.3 Technology and E-commerce

- Shopee / Sea Group: Has an engineering center in KL, one of the largest tech employers in SEA.
- Carsome: First Malaysian unicorn; Significant employer of product and platform engineers.
- Aerodyne Group: Large drone technology firm creating opportunities for engineers working on embedded systems and real time systems.
- IBM Malaysia, Dell Technologies, HP & Accenture: Major global tech companies operating in Malaysia.

6.2.4 Government and quasi-government institutions

- Malaysia Digital Economy Corporation (MDEC): Develops digital policies and is home to many tech strategy and engineering jobs.
- Petronas Digital: Petronas' digital and data unit; A major employer of engineering talent.
- MAMPU & PADU: Agencies driving government digital transformations.
- LHDN Digital (Inland Revenue Board): Actively modernising its technology infrastructure.

6.3 Justification Broad and Growing Opportunities

There is substantial evidence supporting the conclusion that there are numerous, expanding opportunities for employment in software engineering both globally and domestically. Additionally, because of the current boom in the regional fintech industry, coupled with the ongoing digitization of virtually every other industry worldwide, this is likely to continue to be an expanding opportunity. As one example, Malaysia has become a hub for technology companies in Southeast Asia through the investment and operational presence of many large multinational corporations; at the same time, the number of local "unicorns" such as Carsome continues to grow. This means that job opportunities will likely be available at all levels of firm size, in various sectors, and in all areas of specialization in software engineering (MDEC, 2022).

Finally, the development of remote and hybrid working arrangements which have become common practice in most countries since 2020 has increased the potential for software engineers from Malaysia who wish to contribute to global software engineering teams from their home country, thus greatly increasing the actual or effective job market available to new entrants into the profession.

7.0 Conclusion

This research on careers has also proven that the role of software engineer is not just a solid and profitable career opportunity but one of the most impactful and changing in today's world. Software engineering began as a defined discipline at the 1968 NATO Conference, then expanded into what we see today, as a disciplined approach using AI to augment every area of software development including cloud computing, mobile apps, and embedded systems. Sustained effort is required to be successful in software engineering. This includes continued investment in both technical knowledge and understanding of how each piece of technology fits together (systems thinking), ethics in developing solutions, being able to communicate effectively with clients, users, managers etc., and a willingness to continue to learn and grow throughout their career. The job market in Malaysia is creating many new opportunities due to national digital transformation programs and a rapidly growing number of local and international tech companies that are looking to hire professionals in the software engineering field.

8.0 Lesson Learned and IEEE Professional Codes of Conduct

8.1 Lessons Learned from This Research

As an outcome of this career research I am truly grateful for the opportunity. Before this assignment, my view of software engineering was generally technology based and related primarily to programming languages, algorithms and system design. My previous views were greatly expanded from this research in numerous ways.

Historically, the major problems experienced within the industry (i.e., software) have rarely been technology based. The software crisis of the 1960s was really a crisis of professionalism, communication, and process. As such, the lesson learned then is still applicable today. Engineers who do not adhere to standards of professional conduct, document their work appropriately or collaborate

effectively with others introduce similar failures into their work that existed fifty years ago.

Employment research further emphasized that in order to sustain a career one must proactively invest time and resources in ongoing education and training. Technology changes much faster than most educational curriculums can adequately address; engineers who solely rely upon knowledge they gained during their educational tenure will ultimately fall behind other engineers that adopt a "continuing education" approach to a life-long career.

Lastly, examining career transitions and risk of being washed out provided insight into the fact that soft skills, communications, emotional intelligence and ethical decision making, while supplemental to technical expertise at all levels of seniority (beyond junior), represent equally important components of an engineer's capabilities regardless of seniority.

8.2 Alignment with IEEE Professional Codes of Conduct

IEEE's Code of Ethics states 8 core values of ethics for professionals (IEEE, 2020) and this study demonstrated the specific ways those values apply to my developing professional practice:

1. Public Safety & Welfare

Increasingly, software engineers are creating critical infrastructures (financial systems, medical devices etc.) on which people's lives and livelihoods depend. Therefore, this study strengthened my commitment to producing correct, secure and thoroughly tested code as an act of public duty.

2. Avoiding Conflicts of Interest

Fintech and Telecommunications have been shown to be areas of engineering work that can create pressures on engineers to sacrifice the safety of customers' data and/or compromise customer data privacy for profit. The IEEE principle against conflict of interest will provide me with a formal basis to determine whether I should accept certain commercial opportunities.

3. Honesty and Realism

All Engineers have a duty to describe what they can do realistically; including describing the limits of their software. Misrepresenting the reliability of your system is an example of violating this principle.

4. Refusal to Accept Bribes & Corruption

In addition to being a violation of professional ethics, it is also illegal for engineers who become involved in purchasing decisions or access to sensitive systems to violate these standards.

5. To Improve Understanding of Technology

I share this value through my PDP by making commitments to ongoing learning, sharing knowledge and contributing to the advancement of our profession as a collective responsibility.

6. Maintain Your Professional Standards

The washout risk assessment outlined in section 5.3 is a compelling demonstration of the risks associated with engineers becoming out of touch technically. According to IEEE, this is now formally required of all engineers.

7. Seek/ Offer Constructive Criticism

This principle is embodied in Code Review practices used widely in software development. It is something I will be working on integrating into my shared collaboration habits.

8. Treat All Colleagues Equitably

With many software developers coming from different countries and backgrounds, there is a strong need for inclusionary and fair treatment of all colleagues regardless of their cultural origins.

9.0 References

Bureau of Labor Statistics. (2024). Software developers, quality assurance analysts, and testers. U.S. Department of Labor, Occupational Outlook Handbook. https://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm

IEEE. (2020). IEEE code of ethics. Institute of Electrical and Electronics Engineers. https://www.ieee.org/about/corporate/governance/p7-8.html

IEEE Computer Society. (2014). Guide to the software engineering body of knowledge (SWEBOK v3.0). IEEE Computer Society Press. https://www.computer.org/education/bodies-of-knowledge/software-engineering

Malaysia Digital Economy Corporation. (2022). Malaysia digital economy blueprint (MyDIGITAL). MDEC. https://www.mdec.my/digital-economy/mydigital

Naur, P., & Randell, B. (Eds.). (1969). Software engineering: Report of a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7-11 October 1968. NATO Scientific Affairs Division.

Pressman, R. S., & Maxim, B. R. (2020). Software engineering: A practitioner's approach (9th ed.). McGraw-Hill Education.

Sommerville, I. (2016). Software engineering (10th ed.). Pearson Education.

Stack Overflow. (2023). Stack Overflow developer survey 2023. https://survey.stackoverflow.co/2023/

PART B: SWOT ANALYSIS AND PERSONAL DEVELOPMENT PLAN

10.0 Reflective Statement

As part of undertaking this SWOT analysis, I have found it to be the most worthwhile task throughout my academic journey so far. Coming into a high-pressure and constantly changing sector as a semester two student, by being able to honestly evaluate my attitudes, abilities, and knowledge using a structured approach, I feel I now understand where I need to make different choices and how these will guide me going forward.

I firmly believe that my greatest strength has been developing a solid base of systems-based computing. In comparison to several of my peer students who have better programming skills but are less familiar with how things work at lower levels of abstraction, I have always placed emphasis on studying operating systems, memory management, and how hardware integrates with software. This isn't just a topic I'm interested in academically; it is the technical identity I am making conscious efforts towards developing a career as a backend infrastructure and fintech systems engineer. Notably, demonstrating my leadership and initiative particularly in respect to creating and managing a student savings organization demonstrates to me that I can apply theoretical knowledge of systems to practical, accountable systems and that is something I want to continue to bring to future professional engineering teams. In addition to having demonstrated both analytical problem-solving ability and the ability to stay composed under pressure, which were developed from various past positions, I believe these qualities will increase in value as I progress in my career.

However, honest self-evaluation requires acknowledging my weaknesses with similar rigor. Because I am currently in semester two, I am still accumulating much of the specialized knowledge that employers place the highest value on. While advanced cryptography, compiler design, and large-scale distributed systems are all topics that are still in front of me in the curriculum, the lack of actual experience working in the commercial technology industry is a legitimate deficit. Specifically, I have never actually worked in a real engineering team, reviewed production code, or operated within the constraints of deploying code in a "real world" environment. I am also cognizant of an individual's tendency

towards over-optimizing: taking too long to refine low-level aspects of a project while the actual professional requirements call for speed of iterative process. While there are numerous opportunities available to me moving forward, I find the Malaysian fintech/digital infrastructure boom coupled with the growing need for fast, systems-aware engineers to be truly exciting. Due to the open-source nature of the ecosystem, I can develop a reputable public resume without commercial employment. Professional social networks and local hackathons provide immediate ways for me to bridge the gap in terms of visibility/experience that exists within my profile. Obtaining industry certification in areas like Linux/AWS will enhance my application prospects for internships that I plan on pursuing during my remaining time as an academic.

The threats that exist for me are structural and common among others in this space therefore they are no less dangerous. The job market for entry level computer science graduates continues to become increasingly saturated. With the integration of AI into software development occurring at an increasing rate, the bar for what defines valuable contributions of humans is rising. A developer who simply writes syntax will be replaced; however, an engineer who designs systems, thinks critically regarding system architecture, and solves new problems will excel. Additionally, macroeconomic trends impacting technology hiring create additional uncertainty for those looking to enter this profession. By defining differentiation through deep specialization versus broad generalization, I am more resolved than ever before.

11.0 SWOT Analysis Summary

The SWOT infographic (Figure 1) is presented in Appendix A. The following table summarizes the key points across all four quadrants:

STRENGTHS	WEAKNESSES
• Foundational system knowledge (OS, memory, hardware-software integration)	• Early academic stage ; lacking advanced specialization modules

• Strong soft skills and emotional intelligence • Demonstrated leadership and initiative (student savings circle) • Analytical problem-solving via real-world technical field studies • Collaborative and adaptive mindset; proven project and budget management	• No formal corporate tech industry or internship experience yet • No exposure to large-scale distributed production codebases • Unfamiliarity with fintech compliance (PDPA, payment protocols) • Tendency toward over-optimization, slowing rapid prototyping
OPPORTUNITIES	**THREATS**
• Booming fintech and infrastructure market in Malaysia and ASEAN • Robust open-source ecosystem for portfolio building • Professional networking via LinkedIn and developer communities • Industry-standard certifications (Linux Foundation, AWS) • Hackathons and tech competitions for real-world skill validation	• Highly saturated entry-level market for CS graduates • Rapidly shifting tech landscape requires constant self-learning • Rising interview standards (DSA, system design) for early-stage students • AI integration automating basic coding tasks, raising the value bar • Macroeconomic fluctuations affecting tech hiring cycles

12.0 Challenges to Overcome and Talents to Capitalize

12.1 TwoKey Challenges to Overcome

Beyond the weaknesses in the swot analysis, two strategic challenges require attention over the next three years.

Challenge one – bridging academic to commercial engineering Experience gap: i will be challenged by transitioning from an environment where effort and learning are encouraged to one that is based on delivering value to a team and product as measure of contribution. This requires intentional construction of a practical portfolio with real projects deployed to production; open-source contributions reviewed by peers; and ideally an internship placement before graduation.

Challenge two - maintaining Depth while managing academic breadth: University curricula are designed to provide students with a broad education but the job market rewards specialization. Addressing the tension between fulfillment of academic requirements and specialization in systems and fintech engineering will require deliberate time management and the willingness to go well above and beyond the syllabus in my top priorities when competing for time with other academic responsibilities.

12.2 Three Talents to Capitalize

Skill 1 — systems level analytical thinking: reasoning about how hardware interacts with software, memory hierarchy and operating system behavior is something which is virtually unheard of among students at Semester 2 of a degree program. My base of understanding in these areas provides a foundation upon which i can build unique project portfolios (a custom memory allocator, a multi-threaded web server) that set me apart from peers whose portfolios consist solely of application-level projects.

Skill 2 — Cross domain professional Experience: Prior to attending University, i have held positions in financial services, including supervising staff, and overseeing daily financial operations. This has provided me with business process knowledge and operational accountability that most engineering students lack. In addition to providing me with a significant competitive advantage in the area of fintech engineering, it also provides me with a broader range of career options than would otherwise be available.

Skill 3 — emotional intelligence and composure under Pressure: the ability to maintain composure in difficult working conditions (as demonstrated through three demanding professional roles by the age of 19), while maintaining control of complex interpersonal/operational issues is a rare quality in highly technical fields. As such, i intend to Capitalize on this skill by seeking out opportunities for leadership within student organizations/hackathons/group projects etc., thereby establishing a reputation as a leader capable of facilitating collaboration and decision making within high-Pressure, fast-paced environments.

13.0 Personal Development Plan (PDP)

The following table presents five goals with corresponding action plans, timelines, success indicators, and current remarks on progress:

Focus Area	SMART Goal & Specific Action Steps	Timeline	Success Indicator (KPI)
Technical Systems Depth	Objective: Develop an experiential, hands-on appreciation of how to design scalable, low latency architectures. • Create a stand-alone multi-threaded server or develop your own custom memory allocator in C/C++. • Explore complex distributed systems architecture beyond typical class content.	Next 6 Months	The systems level project is complete, working properly and is documented/uploaded onto GitHub.
Fintech Industry Awareness	Objective: Obtain deeper knowledge of specific engineering architecture designs, and their needs in the Fintech space. • Allocate 30 minutes per week to reading about the engineering blog posts of top tier companies (Stripe/PayPal/CloudFlare) . • Read three detailed case studies about designing low-latency trading platforms.	Continuously (every six weeks, review previous year's documentation)	A document summarizing my major takeaways from 20+ cases documenting how to architect low latency solutions.
Professional Networking & Visibility	Objective: Establish myself professionally through networking and establish relationships with peers and leaders that can provide guidance and opportunities. • Update LinkedIn profile to illustrate my interest in engineering systems and fintech.	Semester 3	. An active LinkedIn network with 50+ contacts in related fields and proof of attending at least one event.

	• Make contact with five people who are engineers/architects in the Fintech industry. • Attend at least one local tech meet-up/hackathon/seminar.		
Advanced Environment Mastery	Objective: Transition away from normal student environment configurations toward industry standards for developing environments, and learning common CLI workflows. • Set up a UNIX/Linux-based OS workflow that will be used for all my personal development. • Learn how to script using shells to automate repetitive development tasks.	3 Months	Elimination of dependence upon GUI tools for every routine task, development of three custom automation scripts.
Secure & Ethical Software Engineering	Objective: Get started with some of the fundamental principles of ethical software development practices and data protection regulations. • Complete an online certification/micro-credential program focused on secure coding (OWASP Top Ten), etc. • Research the local laws governing the use of financial information in Fintech (PDPA, etc.) that govern the application of your software.	4 Months	Obtain certification in secure programming and create a compliance checklist so you can check against it during each subsequent application.

Note: The SWOT infographic diagram (Figure 1) submitted with this assignment is attached as Appendix A, per the original infographic prepared during the research phase.