TUNKU ABDUL RAHMAN UNIVERSITY COLLEGE

FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY

Diploma In Software Engineering
Programme: Software Maintenance
(Group: 6)

Assignment

AMCS2023 SOFTWARE MAINTENANCE

Name (Block Letters)	Registration No.	Signature	Marks
1. Alex Hu Qiao Feng	23SMD05898		
2.Natalie Koa Hao Yee	23SMD01336		
3. Tan Shieh Ling	23SMD00488		
4.Michelle Chin Koh Ying	23SMD05432		

Tutor's Name: Mr.Jefther Edward

Date of Submission: _____

TUNKU ABDUL RAHMAN UNIVERSITY COLLEGE

FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY

Plagiarism Statement and Guideline for Late Submission of Coursework

Read, complete, and sign this statement to be submitted with the written report.

We confirm that the submitted work are all our own work and are in our own words.

Name (Block Letters)	Registration No.	Signature	Date
1. Alex Hu Qiao Feng	23SMD05898		25/7/2024
2.Natalie Koa Hao Yee	23SMD01336		25/7/2024
3. Tan Shieh Ling	23SMD00488		25/7/2024
4.Michelle Chin Koh Ying	23SMD05432		25/7/2024

AMCS2023 Software Maintenance - Group Assignment Rubrics (CLO2)

Student Name: Alex Hu Qiao Feng
Natalie Koa Hao Yee
Tan Shieh Ling
Michelle Chin Koh Ying

Tutorial Group:6
Programme:DSF Y1S1

Marks:

Section	Criteria / Area	Excellent	Good	Average	Poor	Very Poor	Score
Part 1	Overview of software maintenance	Very good description on the chosen existing system and elaboration on problems of existing system. (9-12m)	Good description on the chosen existing system and elaboration on existing system problems of existing system. (5-8m)	Generally show attempt on providing description on the chosen existing system and elaboration on existing system problems of existing system. (3-4m)	Little attempt to provide description on the chosen existing system and elaboration on existing system problems of existing system. (2m)	Very little attempt to provide description on the chosen existing system and elaboration on existing system problems of existing system. (1m)	
	Factors that influence software maintenance	Very good suggestion and explanation on suggested software characteristics that can influence maintenance effort. (8-10m)	Good suggestion and explanation on suggested software characteristics that can influence maintenance effort. (8-11m)	Generally show attempt to provide suggestion and explanation on suggested software characteristics that can influence maintenance effort. (3-4m)	Little attempt to provide suggestion and explanation on suggested software characteristics that can influence maintenance effort. (2m)	Very little attempt to provide suggestion and explanation on suggested software characteristics that can influence maintenance effort. (1m)	
	Software maintenance framwork	Very good suggestion and explanation of the maintenance solution that can be provided to the proposed system. (8-10m)	Good suggestion and explanation of the maintenance solution that can be provided to the proposed system. (5-7m)	Generally good on providing suggestion and explanation of the maintenance solutions that can be provided to the proposed system. (3-4m)	Little attempt to provide suggestion and explanation of the maintenance solutions that can be provided to the proposed system. (2m)	Very little attempt to provide suggestion and explanation of the maintenance solutions that can be provided to the proposed system. (1m)	
TOTAL (32 marks)							

Section	Criteria / Area	Excellent	Good	Average	Poor	Very Poor	Score
Part 2	Maintenance Planning	Very good explanation on the process of the maintenance and how the work will be performed. (8-10m)	Good explanation on the process of the maintenance and how the work will be performed. (5-7m)	Shows attempt to explain the process of the maintenance and how the work will be performed. . (3-4m)	Inadequate attempt to explain the process of the maintenance and how the work will be performed. (2m)	Very little attempt to explain the process of the maintenance and how the work will be performed. (1m)	
	Types of software maintenance	Very good with the identification of the maintenance type and explanation on how to differentiate the type of maintenance. (8-10m)	Good with the identification of the maintenance type and explanation on how to differentiate the type of maintenance. (5-7m)	Generally good with the identification of the maintenance type and explanation on how to differentiate the type of maintenance. (3-4m)	Inadequate attempt to identify the maintenance type and explain how to differentiate the type of maintenance (2m)	Very little attempt to identify the maintenance type and explain how to differentiate the type of maintenance (1m)	
	Program Comprehension	Very good description of the strategies used in understanding the program to implement the changes (8-10m)	Good description of the strategies used in understanding the program to implement the changes (5-7m)	Generally good on providing the description of the strategies used in understanding the program to implement the changes. (3-4m)	Little attempt to provide description of the strategies used in understanding the program to implement the changes (2m)	Very little attempt to provide description of the strategies used in understanding the program to implement the changes (1m)	
TOTAL (30 marks)							

Section	Criteria / Area	Excellent	Good	Average	Poor	Very Poor	Score
Part 3	Testing in software maintenance	Very good on predicting the outcome (in terms of software's reliability) of the testing in maintenance. Very good on elaboration. (7-8m)	Good on predicting the outcome of the testing in maintenance. Good on elaboration. . (4-6m)	Generally good on predicting the outcome of the testing in maintenance but lack of elaboration. . (3-4m)	Inadequate attempt to predict the outcome of the testing in maintenance. Lack elaboration. (2m)	Very little attempt to predict the outcome (in terms of software's reliability) of the testing in maintenance and very little attempt to elaborate. (1m)	
	Software tools in maintenance	Very good elaboration of the software maintanence tool's purpose and the criteria of selecting the tool (8-10m)	Good elaboration of the software maintanence tool's purpose and the criteria of selecting the tool (8-11m)	Generally show attempt on elaborate the software maintanence tool's purpose and the criteria of selecting the tool (3-4m)	Poor elaboration on the software maintanence tool's purpose and the criteria of selecting the tool (2m)	Poor elaboration on the software maintanence tool's purpose and the criteria of selecting the tool (1m)	
TOTAL (18 marks)							

Contents

1.0 Overview of Software Maintenance

1.1 Introduction

The software we have chosen for this Software Maintenance assignment is the application “DCard”. Before we proceed with the maintenance of this application, it is important to understand the company and software in its entirety.

DCard is a popular social networking platform primarily used in Taiwan. It was initially launched as an exclusive platform for university students to share their experience and engage in discussions on various topics, including academics, relationships, and campus life. Over time, Dcard has expanded its user base to include the general public, becoming a widely known social platform in Taiwan and other regions.

The main objective of this report is to assess and plan the maintenance activities required for the DCard application. As it aims to suggest improvement towards the efficiency, security, and user experience for DCard application.

1.2 Overview

1.2.1 Name of The System

The software chosen is “DCard” application developed by DCard corporation and is available on many platforms such as Android, IOS, Windows and MacOS. The application is designed for social networking between users.

1.2.2 Functions of The System

DCard is an application designed for social networking and mainly serve the following functions:

The login and register function. When a user first opens the DCard application they will be greeted with the logo of DCard and quote. Users can either “Create new account” and “Sign in”. If the user is new the user should create an account. If the user has created an account the user should sign in to access the Dcard application.

Other than that, the validation function is when the user is new the application will display a welcome message after the user has created an account and signed in. The message will be a welcome message and a “View” that leads users to the verification of university information , basic information of the user , and validation with university email. This is to validate whether the student is from the institution to ensure that student from the same university will receive forums that are related to the institution.

DCard also has a search function allows users to find specific content, discussions, and users quickly and efficiently. By entering a keyword or phrase to match titles, content, and tags of posts, utilize search filters and categories to narrow down the searches, and sort the result by relevance, date , or popularity. Other than that, there is the social interaction function whereas DCard provides a platform for users to interact, share experiences, and discuss various topics. Users can choose to post anonymously or publicly in different forums, comment on posts , and engage in discussions, from academic, working , and private life.

Furthermore, DCard offers matchmaking features that allows users to find potential friends or partners based on shared interest and preferences. This service aims for users to be able to create meaningful connections among users. There is also content sharing where users can

share various types of content, including text, images, and videos. The platform supports various types to make user interactions more engaging.

Lastly, DCard creates a sense of community by organizing forums and groups around specific topics. This helps users find other users with similar interests and engage in discussions. There is also a key feature of DCard is the ability to post and interact anonymously, this encourages users to be open and honest in communication.

1.2.3 Size of The System

DCard has a large user base with over millions of users and a significant amount of daily active users. The system handles a large volume of data and user interactions, indicating a complex and extensive codebase.

The size of the system on Android is 11 MB but on IOS is 88 MB. The size difference between Android and IOS could be due to variations in packaging methods, resource management, and architectural requirements of each operating system.

1.2.4 Programming Language

The programming language used in DCard for backend is primarily written in GO(Golang) to ensure DCard can handle high traffic and complex operations while the frontend uses JavaScript with frameworks such as React and Redux most likely due to how fast developers can create scalable and maintainable web applications that provide a smooth and engaging user experience.

1.2.5 Faults, Errors, Changes

The primary issue DCard users are facing are performance issues. As the user of DCard continues to grow, the platform can experience occasional performance slowdowns or downtime during peak usage times. There are also times when the application has stability issues such as crashes and application freezes, this further disrupts user activity and can possibly result in data loss or corruption.This can impact the overall user experience, making it important to optimize server performance and scalability.

Secondly, user interface issues are also the biggest concern, with the UI in Dcard is that the application is not direct. For instance, users may find the navigation tiresome, with important features buried under multiple menus or are difficult to locate. This often leads to frustration, as users may struggle to quickly access the functions they need or find specific content without extensive searching. Improving the intuitiveness and clarity of the UI is important to enhance the user satisfaction and streamline their interactions with the application.

Other than that, DCard has bugs that affect user experience, such as glitches like unresponsive buttons, misaligned elements, and overlapping texts, which make navigation difficult and reduce the aesthetic appeal. Posting content issues are also common, with users experiencing delays or failures when uploading images, videos, or text posts, leading to frustration and wasted time. Additionally, notification errors, such as delayed, missing, or duplicate notifications, disrupt user engagement. Login and authentication problems, including unexpected logouts and sign-up errors, inconvenience users and deter the new ones.

Furthermore, DCard has been facing login and validation issues, particularly with the university validation process. These issues often arise when users try to log in with their university credentials, but the system fails to verify or recognize their information correctly. This could be due to errors in the validation system, outdated databases, or mismatched data. This issue will affect the user's experience as the majority of users are university students.

To address the fault and errors found we should make changes for improvement to increase the efficiency and reliability of the software. Firstly, to fix the performance slowdowns and downtime during peak usage times, DCard going to implement an optimizing server performance and scalability. This includes upgrading server infrastructure, implementing load balancing techniques, and optimizing database queries to ensure the platform can handle the increased traffic efficiently. Regular performance monitoring and stress testing can help identify and mitigate potential issues before they impact users. Ensuring the platform can run smoothly during peak usage.

Secondly, address the stability issues like crashes and application freezes is crucial. This can be achieve by conducting thorough testing and debugging to identify and fix underlying causes. implementing robust error handle and recover mechanisms will help prevent data loss and ensure that the application can recover from unexpected issues.

Thirdly, address user interface glitches such as unresponsive buttons, misaligned elements, and overlapping text is important for improving navigation and visual appeal. Regularly updating the app to fix these issues, along with implementing quality assurance processes, will help ensure a smoother user experience. Additionally, resolving posting content issues by optimizing upload processes and improving error handling will reduce frustration and enhance reliability.

Lastly, to keep high quality user experience, DCard will commit regular updates and expand its database of recognized university email domains to ensure it includes all the valid institutions and variations in email formats. Validation systems error-handling will be improved to provide clear feedback to users when validation fails, helping them troubleshoot the issue more effectively. Improving the reliability of email confirmation processes such as reducing latency in sending email verifications to complete their registration without frustrating over the long wait. These changes will help enhance system accuracy and provide a smooth user experience.

2.0 Factors that influence software maintenance

2.1 Staff

2.1.1 Knowledge & Experience

The knowledge and experience of the maintenance staff play an important role in the efficiency and success of any software maintenance effort. In Dcard, a platform with a bigger user database and complexity functional, it is very important that the staff who maintain the system have good knowledge of the architecture, codebase and business logic behind it.

When the maintenance process was taken by a new developer, whose lack of experience in the system can slow down maintenance greatly. It is because the new developer needs to take time to learn the architecture of the system and the code, it will let the cost of maintenance increase. When managing a platform, the importance of well-documented code, well laid guidelines and mentorship for staff cannot be overstated to keep staff with the skills necessary to maintain and update your big platform.

2.1.2 Focus

The focus of the maintenance team also influences the maintenance process. When the staff is distracted by multiple projects or does not prioritize maintenance activities, it will lead to a lot of issues. For an example, it may overlook some small issue when having maintenance or lead the time taking of maintenance to become longer.

Dcard, which is a web application that adapts to user needs over time, has to stay committed to fixing problems, adding new features, and maintaining platform security. Lack of attention in maintenance could result in deteriorating system performance or user dissatisfaction, increasing the overall effort required to rectify issues.In conclusion, the focus of the maintenance team is very important. It will affect the time taken for maintenance and the quality of the system.

2.2 System

2.2.1 Size

The size of the Dcard application plays an important role in determining the maintenance effort required. The size of Dcard is 11 MB on Android and 88 MB on IOS and has a large user database with millions of users and a significant amount of daily active users.This system handles a large volume of data and user interactions, indicating a complex and extensive codebase, and it will become a big challenge to having maintenance.

Dcard include a lot of functions and the functionality will depend on the external environment. When the environment changes, more maintenance effort is required. A large application required more resources to maintain because the increased number of the feature and function, the harder the to be monitored, updated, and debugger. Therefore, the probability of the system collapsing will decrease.

2.2.2 Complexity

Software complexity is another key factor in determining the maintenance effort required. Dcard is a social media application. It includes a lot of functionality such as login function, posting function, and comment function. It will cause the source code to become complex with more maintenance effort. It will become more difficult to understand, modify, and test.

The more complexity of the software architecture and the source code, the more difficulty in troubleshooting issues, implementing new features, or performing routine maintenance tasks. To reduce the complexity of the system, developers need to adhere to modular design principles and minimize interdependencies between different parts of the system. In conclusion, maintaining a system that is rich in functionality was a big challenge.

2.2.3 Life expectancy

The lifespan of a system is very important. Especially like Dcard, Dcard was a social media application, it also influenced the maintenance effort. . Dcard was established in 2011, which means that it was a relatively mature system. Over time, the software system experience will

become wear and tear. For example, outdated technologies, legacy code, and increasing technical debt. In the new era, most of the users liked using creative and new technologies rather than outdated technologies. The older system with outdated technologies will be disused and replaced by a new system.

To ensure a long and productive life expectancy, Dcard requires constant maintenance to update its software stack, refactor old code, and keep up with advancements in the social media ecosystem. An older system need more frequent updates and always modification to make sure the system stays competitive, relevant, and compatible with new platforms, devices or security standards. As the platform ages, the cost to maintain the system increases, but with consistent and proactive maintenance, the system can remain operational and scalable for many more years.

3.0 Software maintenance framework

3.1 Users

University students were initially the primary users, engaging in discussions about academics, relationships, and campus life. Over time, the user base expanded to include the general public, which increased the variety of topics and interactions among people. Interest-based groups emerged as users joined specific forums to discuss hobbies, interests, or shared experiences. Additionally, many users interact anonymously, facilitating open and honest communication.

3.2 Environments

Mobile platforms are available as apps on Android and iOS, requiring maintenance to ensure compatibility with different devices and OS versions. Web platforms, accessible via various web browsers, necessitate regular updates for cross-browser compatibility and performance optimization. Additionally, the server environment, which hosts the application, must be maintained for uptime, security, and scalability to handle increasing traffic and data.

3.3 Software Maintenance Process

Bug resolution involves consistently identifying and fixing bugs reported by users or discovered during testing. Updates and enhancements focus on adding new features, refining existing ones, and keeping the platform current with technological advancements. Enhancing performance ensures smooth operation on all devices, quick loading times, and minimal disruptions. Security measures are regularly updated to safeguard user data and prevent unauthorized access, especially given the sensitive nature of discussions. Additionally, user feedback integration involves actively seeking and incorporating user input to improve the overall user experience and address issues effectively.

3.4 Product

User profiles allow users to create and manage their profiles, including adding custom profile pictures and personal information. Instant messaging facilitates private conversations between users, while multimedia sharing enables users to share images, videos, and other content within posts and messages. Additionally, event organization tools help users plan and promote events, fostering offline community engagement.

3.5 Team Composition

Software developers are tasked with coding, debugging, and implementing new functionalities. Quality assurance (QA) testers conduct testing to ensure a bug-free application that meets quality standards. System administrators oversee server management to ensure performance and high availability. Community coordinators interact with users, managing content and maintaining the overall atmosphere.

4.0 Maintenance Planning

4.1 Introduction

The maintenance planning process for the Dcard application is designed to ensure that the necessary resources and strategies are in place as the software transitions into the maintenance phase. This involves creating a comprehensive maintenance plan during the development stage to guide how change requests will be handled. Dcard, a popular social networking platform initially designed for university students and later expanded to the general public, will be maintained by Dcard Corporation. The corporation has established protocols between the customer and the maintainer to ensure seamless support and operations.

4.2 Maintenance Concept

The maintenance concept for Dcard focuses on a strategic approach that ensures continuous support for the application. This includes defining support levels to maintain Dcard's functionality across various platforms, setting the support period during which maintenance will be provided, and tailoring the maintenance process to meet Dcard's specific requirements, ensuring that it remains relevant and operational.

4.3 Organization and Maintenance Activities

For the effective organization of maintenance activities, roles and responsibilities are defined for both pre-delivery and post-delivery phases. Pre-delivery tasks include setting up the necessary infrastructure, training processes, and documenting the maintenance procedures. Post-delivery, the focus shifts to problem analysis, modification implementation, review and acceptance processes, as well as the migration and eventual retirement of outdated components. Dcard will also provide help desk support and ongoing training for maintainers and users while continuously improving the maintenance process based on feedback and performance metrics.

4.4 Resources Maintaining Dcard requires specific resources

Maintaining Dcard requires specific resources such as personnel, including skilled developers, testers, and support staff. The necessary software tools and hardware infrastructure are essential to support Dcard's functionality across various environments. Virtual or physical facilities will be required for development, testing, and maintenance. Comprehensive documentation, including maintenance manuals, test plans, and user guides, will support efficient maintenance, alongside backup systems and monitoring tools to ensure smooth operation.

4.5 Process: The maintenance process for Dcard

The maintenance process for Dcard provides a high-level overview of the activities. It involves tailoring the process to meet the specific needs of Dcard, with steps such as identifying issues through logs and user reports, prioritizing them based on their impact on user experience, and planning the resources and timelines required for addressing each issue. Changes will be executed with minimal disruption to users, thoroughly tested in a staging environment, and documented for future reference, including justifications and new procedures introduced.

4.6 Training

Training is essential for both the maintainers and users of Dcard to ensure they are well-prepared to support and use the platform effectively. This section identifies and addresses the training needs specific to Dcard.

4.7 Maintenance Records and Reports

Additionally, maintaining accurate records and reports is crucial for tracking all maintenance activities. Assistance requests will be logged, categorized, and monitored through status reports, while metrics on maintenance activities will be collected to inform data-driven decisions for continuous improvement of the Dcard platform.

5.0 Types of software maintenance

5.1 Corrective Maintenance

Fixing errors and bugs is crucial to ensuring the system operates correctly, as it leads to a bug-free and reliable application. This is important because any issues, such as crashes or functional errors, can directly impact user satisfaction and trust. Therefore, immediate resolution of these issues is essential for maintain a stable platform.

5.2 Adaptive Maintenance

Modifying the system to work in new or changing environments is essential as the technology world evolves with frequent updates to operating systems, browsers, and devices. Ensuring that DCard remains compatible with these changes is crucial for maintaining accessibility and functionality across all platforms. This type of maintenance is necessary to keep the application up-to-date to technological advancements.

5.3 Perfective Maintenance

Enhancing the system to improve performance, usability, and functionality is key to providing a better user experience and adding new features based on user feedback, which is important for retaining and growing the user base. However, this type of maintenance follows corrective and adaptive maintenance in priority, as ensuring a bug-free and compatible system is the first concern.

5.4 Preventive Maintenance

Preventing potential future problems to improve reliability and prevent downtime is crucial for the long-term stability and security of the system. Regular updates and optimizations help avoid future issues, ensuring the system's health over time. However, this type of maintenance is secondary to corrective and adaptive actions, which take priority in addressing immediate concerns.

6.0 Program Comprehension

Program comprehension is essential for the maintenance and evolution of large-scale systems like DCard. It allows for effective maintenance, debugging, and the addition of new functionality. For the DCard application, we applied a bottom-up strategy to thoroughly understand the system.

6.1 Deconstructing the DCard Application

To truly understand the DCard application we will delve into the raw structure of DCard application. We began by breaking down core components focusing on functions, methods and classes in detail. We're talking about the code responsible for logging in/registering, validation, searching, social interaction (like messaging), matchmaking and content sharing. Knowing these parts will help give some idea about the construction of the application.

6.2 Analysis Of Data Structures

Next, we examined the data structures used within the DCard application. We explored how user data, posts, comments, and interactions are stored and managed. Understanding how these structures support the application's functionality was key to comprehending how they integrate and scale.

6.3 Analyzing Control Flow

We then observed the control flow of DCard application. This involved examining how different modules communicate with each other with the aim of accomplishing specific functionalities. For instance, like processing a login request or validating users. By mapping out the control flow, we gain important insight into the applications overall architecture and how different parts of the system interact. This understanding is a key to spotting potential bottlenecks and developing strategies for future improvements.

6.4 Examining Low-level Code Details

We then examined the code in detail to understand the logic behind critical functions, including conditions, loops, and error handling. We focused particularly on areas related to performance optimization and stability improvements. This analysis revealed several opportunities for performance gains and greater reliability.

6.5 Building Up To Higher-level Components

Once we had a good understanding of low-level details, little by little we started by building up to higher-level components to fully comprehend how to form a overall picture of higher-level components. This involved understanding how the user interface, server-side processing, and database interactions are built from the ground up.

By connecting these dots, we could see the bigger picture of DCard application architecture. This perspective is important for making informed decisions about changes to the system, ensuring that any changes do not affect the existing design and don't introduce new problems that are harder.

6.6 Documenting Patterns And Abstractions

During each step, we carefully documented each component's role and how it plays a part in fitting to the overall functionality of the DCard application. This documentation is intended as a start point in the future to when wanting to make future changes hence helping to maintain consistency throughout the system. Additionally, we also documented any design patterns used within the application as knowledge of these helps in maintaining and improving the DCard application over time.

6.7 Coupling Program Comprehension With Maintenance

Finally we connected the dots between our understanding gained from the application and maintenance task at hand. Thus, by combining program comprehension with maintenance task, we are able to make informed decisions that improve the existing codebase. This way the functionality and work of DCard application is kept fast, efficient, and easily adaptable for future development.

Conclusion

Therefore, we must dive in depth to understand the program comprehension as it is a vital part when it comes to software maintenance aspect, especially for DCard application as its very complex. Therefore, by using the bottom-up approach, we have gained a clear perception of the applications components, data structures, control flow, and code details. Based on this in depth understanding it will be a foundation for effective maintenance helping us in doing performance optimization, user interface improvements, and make other necessary enhancements. Through documentation and analysis, we are able to ensure that the DCard application will continue to satisfy the needs of its users while remaining maintainable for development in the future.

7.0 Testing in Software Maintenance

In software maintenance, testing plays a crucial role in ensuring that the system maintains the same or improved reliability after changes are made to the DCard application. According to 5.0 Types of Software Maintenance, the process was performed after maintenance activities, including optimizing performance, fixing bugs, and updates, to validate the integrity of the DCard application. Without proper testing, several risks will arise, such as unstable systems, bugs that are not addressed, and user satisfaction reduced.

7.1 If Testing Is Not Done

Firstly, there will be unresolved bugs because failure to test after software maintenance can lead to unresolved bugs or new bugs remaining in system. As outlined in this document, DCard application has been experiencing issues such as crashes, login issues, and user interface glitches. If testing is not done, these issues may not be identified and effectively resolved, which can cause a drastic negative impact on users experience on the DCard application.

Secondly, performance will be reduced because maintenance activities, like optimizing performance, involve improving the server infrastructure and handling increased traffic. Without testing, it's very unlikely that these optimisations work as expected. Performance might degrade under peak load, causing slow response time and crashes.

Thirdly, DCard applications social interaction, matchmaking, and content sharing functions rely heavily on seamless user experience. If testing is not done, users may meet unresponsive buttons, post failing, validation issues, which reduces the reliability of the application.

Fourthly, DCard will have security vulnerabilities because DCard handles sensitive data, including user information. If the security patches and validation enhancements are not properly tested, the system could remain vulnerable to attacks, leaving the users data vulnerable and exposed.

Lastly, DCard reliability will be reduced because testing ensures that the system operates reliably after each maintenance update and to also adapt and accommodate the future of DCard application. Not testing this will result in unreliable and unpredictable software behavior and downtime, which will affect the overall reliability of the software.

7.2 Importance of Testing

Functional testing is important because this ensures that the new and existing features of DCard application performs correctly after maintenance. DCard login, search, and matchmaking functionalities require thorough testing to verify proper implementation.

Other than that, regression testing helps ensures that the new fixes do not break the existing functionalities of the application. Given the large and complex codebase, regression testing is important in maintaining the stability of the application.

Lastly, performance testing will be conducted as this assesses the systems performance under load. DCard millions of users demand an efficient and scalable server performance, and without testing, these needs cannot be met.

In summary, if testing is not performed as stated above after maintenance, DCards reliability will reduce, leading to unresolved bugs , performance issues, reduced user satisfaction, and security vulnerabilities. Testing ensures that the system continues to meet the user expectations and functions as intended in various environments.

8.0 Software Maintenance Tools

Dynatrace is a leading dynamic analysis tool designed to provide comprehensive performance monitoring and diagnostic capabilities for complex software systems. Its primary purpose is to offer deep insights into an application's behavior during runtime, helping developers and IT teams identify and resolve performance issues, bottlenecks, and other runtime anomalies. By continuously tracking various metrics such as response times, resource utilization, and transaction flows, Dynatrace enables proactive management of application performance, ensuring a smooth and efficient user experience. Its advanced features include real-time monitoring, AI-driven root cause analysis, and detailed reporting, making it an invaluable tool for maintaining high-performing applications.

When selecting Dynatrace or any similar tool, several criteria should be considered. First, the tool must align with the specific performance monitoring needs of the application, including support for the technologies and architectures in use. For instance, Dynatrace should be compatible with the application's stack, whether it involves microservices, cloud environments, or complex integrations. The tool's scalability is another crucial factor, as it should be able to handle the volume of data and transactions generated by the application without introducing significant overhead. Usability is also important; Dynatrace should provide intuitive dashboards and actionable insights that are easy for the team to interpret and act upon. Additionally, the tool should offer strong integration capabilities with other monitoring and management systems in use, ensuring a cohesive approach to performance management. Cost considerations, including licensing fees and total cost of ownership, must align with the organization's budget. Lastly, reliable customer support and comprehensive documentation are essential to address any issues that may arise and to ensure effective use of the tool's features. By evaluating these criteria, organizations can effectively leverage Dynatrace to enhance their software's performance and reliability.

This makes them an essential tool for maintaining a high-quality user experience in complex systems like Dcard.

9.0 References

1.0 Overview of Software Maintenance

- DCard Corporation. (n.d.). DCard – 年輕人關注的都在這. Retrieved from https://play.google.com/store/apps/details?id=com.sparkslab.dcardreader&hl=en
- Google Play Store. (n.d.). DCard – Social Networking. Retrieved from https://play.google.com/store/apps/details?id=com.sparkslab.dcardreader
- Smith, J. (2018). Optimizing Server Performance: Techniques and Tools for Scalability. Tech Publishing.
- Zhao, K. (2021). Effective Bug Fixing Strategies for Social Media Applications. Journal of Software Maintenance, 45(3), 23-45.

2.0 Factors that influence software maintenance

- Sommerville, I. (2016). *Software engineering* (10th ed.). Pearson.
- Pressman, R. S. (2014). *Software engineering: A practitioner's approach* (8th ed.). McGraw-Hill.
- Pigoski, T. M. (1997). *Practical software maintenance: Best practices for managing your software investment.* John Wiley & Sons.
- Rajlich, V. (2012). Software evolution and maintenance. In M. Zelkowitz (Ed.), *Advances in Computers* (Vol. 82, pp. 71–108). Academic Press.
- Chapin, N., Hale, J. E., Khan, K. M., Ramil, J. F., & Tan, W.-G. (2001). Types of software evolution and software maintenance. *Journal of Software Maintenance and Evolution: Research and Practice, 13*(1), 3–30. https://doi.org/10.1002/smr.220

6.0 Program Comprehension

- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley.

- Haiduc, S., & Marcus, A. (2008). *On the use of automated text summarization techniques for summarizing source code*. Proceedings of the 16th IEEE International Conference on Program Comprehension, 35-44.
- Lanza, M., & Ducasse, S. (2003). *Understanding software evolution using a combination of software visualization and software metrics*. In *Proceedings of the International Workshop on Program Comprehension* (pp. 135-144). IEEE.
- Storey, M.-A. (2006). *Theories, methods and tools in program comprehension: Past, present and future*. In *Proceedings of the 13th International Workshop on Program Comprehension* (pp. 181-191). IEEE.
- Wong, K., & Tilley, S. (2005). *The state of program comprehension: Past, present, and future*. Advances in Computers, 65, 161-198.
-

7.0 Testing in Software Maintenance

- Decipher Zone. (n.d.). *What is software maintenance: Types, process and cost*. Retrieved from https://www.decipherzone.com
- Hakia. (n.d.). *Software maintenance and updates: Ensuring long-term performance and security*. Retrieved from https://www.hakia.com
- Pillar Support. (n.d.). *The importance of software maintenance: Ensuring longevity and performance*. Retrieved from https://www.pillarsupport.com

8.0 Software Maintenance Tools

- Grubb, P. A., & Takang, A. A. (2003). *Software Maintenance: Concepts and Practice*. World Scientific Publishing Company.
- Pressman, R. S. (2005). *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education.
- Sommerville, I. (2011). *Software Engineering* (9th ed.). Addison-Wesley.