A Multi-Factor Authentication System

Jayasudha Kalamegam, Kritthika Shanmugam, SathiyaShivani Sathish Kumar
Team Number: 3, CS504-Software Engineering, School of Technology and Computing, City University of Seattle
shanmugamkritthika@cityuniversity.edu
jayasudhakalamegam@cityuniversity.edu
sathishkumarsathiya@cityuniversity.edu

Abstract

The report presents four popular MFA systems that help enhance security in our digital world. Indeed, the real world now needs MFA to add an additional layer of security by making users prove their identity in more ways than one. In other words, even when somebody manages to steal our password, that person still cannot enter our account without being able to complete additional steps. Among the several MFA systems, we have reviewed are Google Authenticator, Microsoft's Azure Active Director Multi-Factor Authentication, Duo Security, and Yubikey. We select them based on usability and kinds of operations and security features. Here, all these selected MFA systems are described briefly with simple examples for better illustration of how they work along with screenshots. In the comparison, we reviewed how different systems performed against user experience, compatibility with other tools, and how complex they are to implement. Google Authenticator provides a user-friendly application that generates login codes. Microsoft's Azure MFA secures accounts using multiple identity verification steps. Duo Security is great for companies because of its smart authentication capabilities. Yubikey uses a physical device for security reasons; thus, it makes it difficult for hackers to access any account. We focused on generating QR codes for one-time passwords and we thought that we can chose Google Authenticator as the best one because of its simplicity and broad acceptance in securing the application. Its user-friendly interface makes setup and TOTP generation straightforward, while its compatibility with multiple platforms ensures smooth integration with our application.

Keywords: Multi-Factor Authentication (MFA), Identity Theft, Security, Google Authenticator, Duo Security, Yubikey, User Experience.

1. INTRODUCTION

In today's digital landscape, securing online applications and sensitive data is critical. Traditional authentication methods, such as username and password, are vulnerable to breaches due to weak password habits and the rise of cyber-attacks. Multi-factor authentication (MFA) has become an essential security mechanism to mitigate these risks. This project aims to design and implement a multi-factor authentication system that enhances login security. The system requires users to authenticate themselves using three factors: a username, password, and a one-time PIN generated by Google Authenticator. By integrating MFA, we ensure that even if a user's password is compromised, unauthorized access to their account remains prevented. This approach is aligned with the industry's best practices, offering a layered defense against credential theft and impersonation. The project will demonstrate how a combination of user credentials and dynamic PIN generation can significantly strengthen user account security.

2. LITERATURE REVIEW

Loubser (2021) highlights the foundational principles of software engineering for beginners,

emphasizing secure software development practices. The book underscores the importance of integrating multi-factor authentication (MFA) systems into applications to enhance security by adding additional layers of user verification.

Paptan et al. (2024) provide a detailed implementation framework for MFA systems, showcasing a practical approach to building secure authentication processes. Their study illustrates how MFA systems, including username-password and time-based one-time PIN (TOTP) mechanisms, significantly mitigate security threats like phishing and brute-force attacks.

Reynolds et al. (2018) investigate the usability challenges and benefits of hardware-based authentication methods, such as YubiKey. While the focus is on usability, the study draws parallels with software-based MFA solutions, emphasizing the balance between security and user convenience.

Nash et al. (2024) conducted a security analysis of widely used authenticator apps like Google Authenticator and Microsoft Authenticator. They highlight vulnerabilities, such as weak QR code implementations and the risks of synchronization, while emphasizing these apps' effectiveness in generating secure TOTP codes. This research underlines the importance of securing MFA implementation to prevent exploitation.

The paper by Qureshi and Kale (2024) discusses the design and implementation of a Multi-Factor Authentication (MFA) system, focusing on enhancing security in digital platforms. It emphasizes the integration of various authentication methods, including username, password, and PIN, aiming to provide a robust and secure user verification process in the context of Industry 4.0 advancements.

Together, these studies emphasize the necessity of MFA for robust cybersecurity, balancing usability, and highlighting best practices for secure development and deployment.

3. AN OVERVIEW AND MECHANISM OF MULTI-FACTOR AUTHENTICATION

Multi-Factor Authentication, commonly referred to as MFA, is a system security enhancement methodology that involves proving multiple factors for any user authentication. This is a layered approach based on the combination of something that the user knows, such as a password; something that the user has, like a one-time password or hardware key; and something that the user is, such as biometrics including fingerprint and facial recognition. The main purpose of MFA is to ensure that, when one level of authentication is compromised, access cannot occur because additional and independent layers of verification are needed.

Adoption of MFA has become requisite in today's digital landscape in order to respond to increasing cyber threats such as phishing and credential theft. The extra layer of security makes it difficult for hackers to hack into any account or system, thereby giving greater protection to sensitive data. We looked into different MFA mechanisms in the study, including one-time passwords or OTPs, hardware keys, biometrics, and push notifications. The critical focus is on the feasibility of systems that are widely used, easy to use, and can provide robust protection against attacks for a wide range of users and organizations.

4. SELECTION, COMPARISON, AND ITS ANALYSIS

For the evaluation of MFA systems, we analyzed four solutions: Google Authenticator, Microsoft Azure Active Directory Multi-Factor Authentication, Duo Security, and YubiKey. Those were selected on the basis of their popularity, ease of use, and capability to offer strong security features.

Google Authenticator

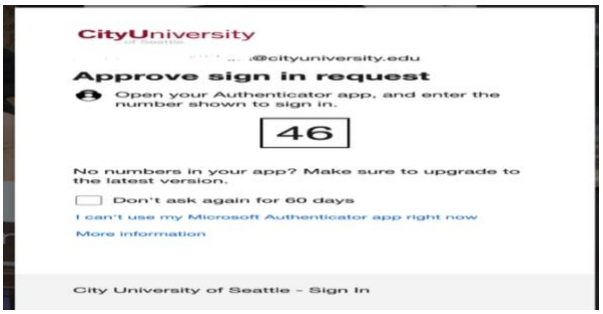
Google Authenticator offers time-based one-time passwords, thereby providing a simple and highly secure way for users to authenticate.

Besides, it is cross-operating system, lightweight, and very easy to be integrated into various systems. Users can generate an authentication code right on their smartphones, which makes it perfect for either small- or medium-scale applications or personal use. Its simplicity and reliability have made it one of the most popular MFA tools globally.

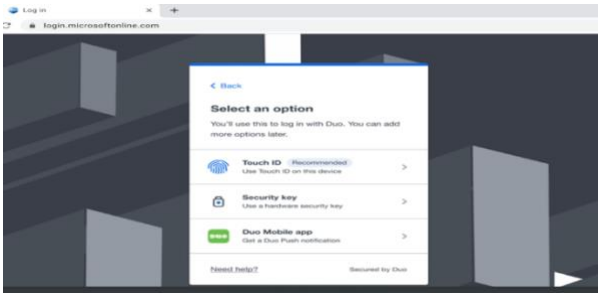
Microsoft Azure Active Directory MFA

Azure MFA has been fully designed for enterprise use cases. With Azure MFA, users get options like phone calls, SMS, and mobile app notifications to authenticate. Enterprise-oriented features include scalability, compatibility with Microsoft's

ecosystem, and support for advanced identity management.



Duo Security
Duo Security provides smart authentication, which includes various features like risk-based prompts and device recognition. Duo balances flexibility with robust security to provide seamless authentication for end-users.



YubiKey
YubiKey is outstanding among them, it connects to physical devices and provides great two-factor authentication, which requires the user to press during login to drastically reduce unauthorized access.

Comparison and Selection
By comparing these systems, we found that Google Authenticator is the best system that would fit our implementation. Due to its simplicity and ease of use, it became compatible for our case study. While Azure MFA remains an excellent solution for enterprises, and YubiKey is outstanding in security, the operational complexity and physical requirements of these make them less practical for the current needs. While Duo Security is highly adaptable, it is a bit beyond what would be required by our project. For the balance of usability and security, Google Authenticator hits the sweet spot for both individual users and smaller organizations.

In this project, we decided to implement Google Authenticator, which is a form of MFA. Our focus will be on implementing one-time password generation through TOTP and its integration into an authentication workflow using Flask and Docker.

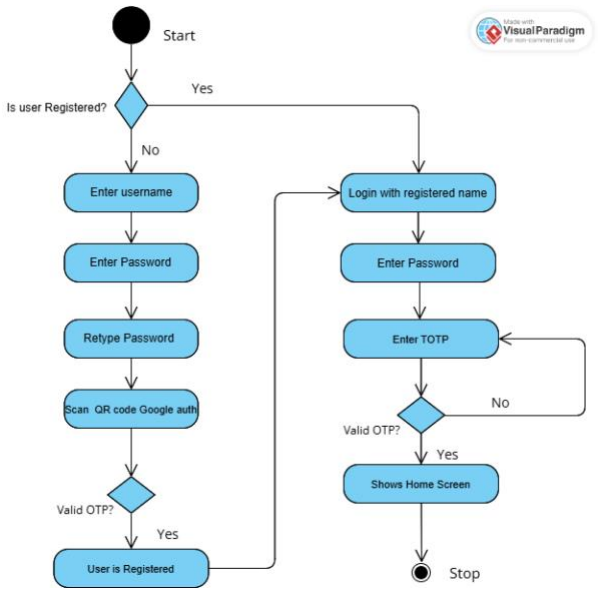
Flask Framework
Flask, a lightweight and versatile web framework in Python, will be used to build the authentication system. Its simplicity and extensive library support make it a suitable choice for developing secure and scalable applications.

Docker for Containerization
Docker will be utilized to containerize the application, ensuring consistent deployment across different environments. This approach will help streamline the development, testing, and deployment processes while maintaining portability and security.

We will be demonstrating an integration between Flask and Docker to perform the whole process in an MFA application. Using Google Authenticator will reduce friction while offering protection through multi-factor authentication. This decision also emphasizes the balance sought between security and accessibility of our solution.

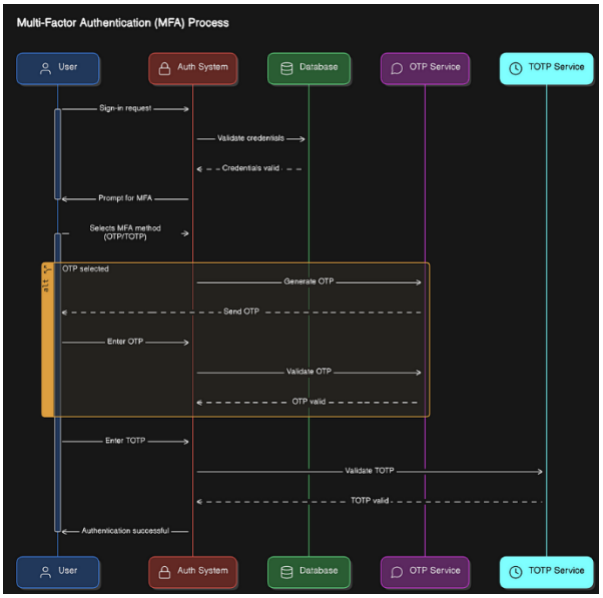
5. ACTIVITY DIAGRAM FOR MFA

This activity diagram illustrates the process flow for a multi-factor authentication (MFA) system involving user registration and login using Google Authenticator for added security. The process begins by checking if the user is registered. If the user is not registered, they must create an account by entering a username, password, and retyping the password to confirm. Once these steps are completed successfully, the user is registered. For already registered users, the login process begins by entering their registered username and password. The system then prompts them to scan a QR code using the Google Authenticator app, linking their account with the app for time-based one-time password (TOTP) generation. The user retrieves the OTP from the app and enters it into the system. The system then verifies the OTP's validity. If the OTP is valid, the user is granted access and redirected to the home screen. If the OTP is invalid, the user is asked to re-enter a valid OTP, ensuring secure access to the application. This diagram clearly explains the two main paths (registration and login), highlighting how MFA is seamlessly integrated to enhance account security through TOTP verification. The design ensures that both new and returning users follow a secure authentication process.



6. MFA ARCHITECTURE

The Multi-Factor Authentication (MFA) process depicted in the architecture diagram referred from medium article demonstrates a secure and layered user authentication workflow. The system consists of five main components: the User, Auth System, Database, OTP Service, and TOTP Service.



The process begins when a user initiates a sign-in request by providing their credentials (username and password). The authentication system validates these credentials against stored data in the database. Upon successful validation, the system prompts the user to select an MFA method—either One-Time Password (OTP) or Time-Based One-Time Password (TOTP).

If the OTP method is chosen, the system interacts with the OTP Service to generate a one-time code and sends it to the user via a predefined communication channel (e.g., email or SMS). The user then inputs the OTP, which the system validates to grant access.

Alternatively, if the TOTP method is selected, the system relies on the TOTP Service for dynamic code generation, which is synchronized with a shared secret key linked to the user's account. The user retrieves the TOTP from an authenticator app, such as Google Authenticator, and enters it into the system. After validating the TOTP, the system completes the authentication process, granting access to the user.

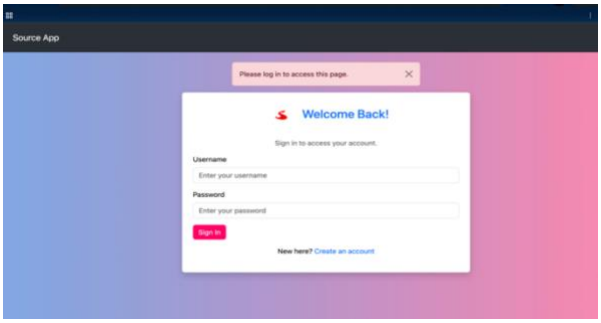
The diagram highlights the integration of services, ensuring a robust authentication mechanism that combines ease of use and high security. This architecture exemplifies how MFA can prevent unauthorized access even if a user's password is compromised.

7. METHODOLOGY

This project starts with the implementation of a simple login system named source app where users can sign up and log in with their username and password.

It is developed based on Flask, and for form handling, it relies on Flask-WTF; for secure password hashing, it uses Flask-Bcrypt; and for database management, it uses Flask-SQLAlchemy. The registration page ensures that usernames are unique and passwords meet minimum security standards. The login interface is user-friendly and combines HTML and CSS with Flask templates for an interface that is both clean and intuitive.

The future work would be devoted to security enhancements through the integration of Google Authenticator, adding 2FA. In particular, the planned future work concerns the implementation of TOTP with PyOTP. The user, upon registration, will attach the Google Authenticator account by scanning the generated QR code from the system, storing the secret key securely in the database. Logging in will involve entering a username and password, then a one-time PIN generated from Google Authenticator. This step greatly improves account security and brings the system up to date with modern authentication standards.



Besides, the containerization with Docker is foreseen in this project. Docker would be beneficial for simplifying deployments by packaging an application and its operational dependencies into lightweight containers that make it portable and scalable. In such a way, the system could easily be deployed in different environments without configuration problems; thus, it is reliable and flexible.

The staged approach to learning from basic authentication to progressively advanced technologies like 2FA, QR code generation, and Docker reflects a journey of growth and balance in simplicity and scalability. This equips the team with hands-on experience in secure application development and modern deployment practices, setting the stage for the development of a robust authentication system with ease of use.

8. PROJECT EXPLANATION

This project focuses on enhancing authentication security by implementing a Multi-Factor Authentication (MFA) system using Flask and Google Authenticator. The backend, developed with Flask, manages user registration, login, and MFA verification routes. Passwords are securely hashed using Flask-Bcrypt, while user credentials and TOTP secrets are stored in an encrypted SQLAlchemy database. PyOTP is used to generate Time-Based One-Time Passwords (TOTP), which are linked to user accounts during registration. A QR code is generated using the qrcode library, allowing users to set up Google Authenticator for TOTP generation.

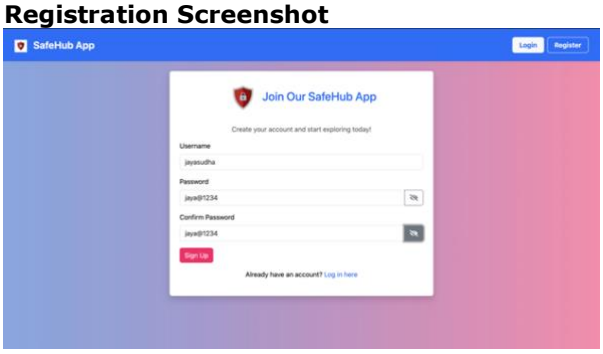
The frontend, designed with HTML, CSS, and Bootstrap, offers a responsive and user-friendly interface for registration, login, and OTP validation. The application is containerized using Docker for scalable and consistent deployment across environments. Comprehensive testing ensures smooth workflows for registration, login, and OTP validation. This project provides a secure, user-friendly MFA solution to address modern authentication challenges.

9. RESULTS

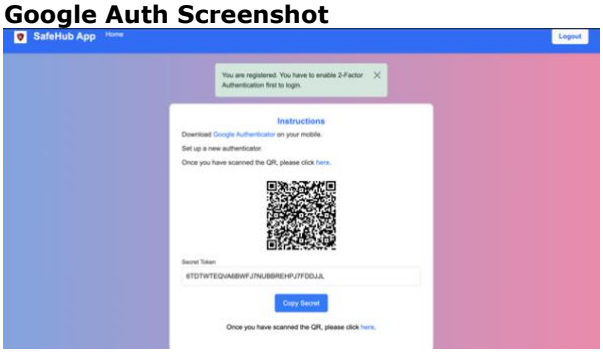
The development of this project demonstrated the successful implementation of a robust Flask-based web application integrating user authentication, database management, and a blueprint structure. Key results include the creation of modular components for user registration, login, and logout, leveraging Flask-WTF for secure and validated forms. The use of Flask-Bcrypt enabled password hashing, ensuring user data security.

The application effectively handled user authentication and authorization, utilizing Flask-Login for session management. This allowed authenticated users to access protected routes while restricting unauthorized access. A SQLite database served as the backend, managed seamlessly with SQLAlchemy ORM and Flask-Migrate for database migrations. These tools simplified handling user data and adding features without complex configurations.
Additionally, the project demonstrated the use of environment variables for configuration, ensuring a secure and flexible deployment setup. The application maintained clean separation between concerns through the use of blueprints, aiding in scalability and maintainability.

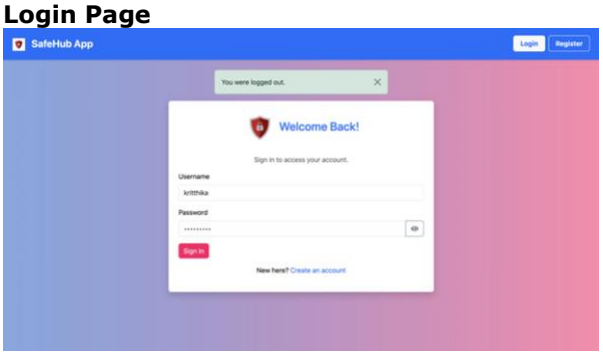
Registration Screenshot



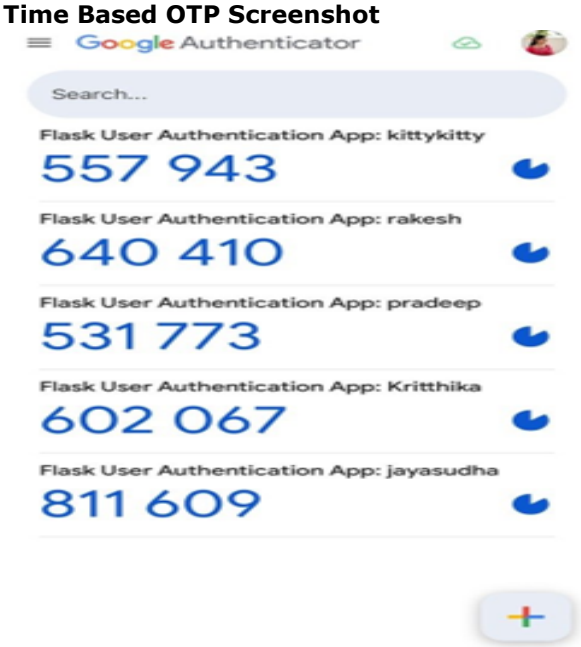
Google Auth Screenshot



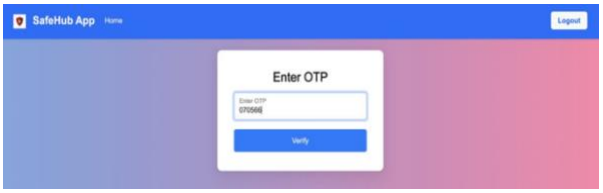
Login Page



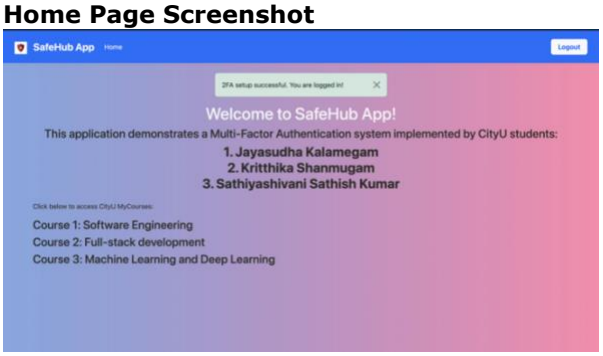
Time Based OTP Screenshot



Enter TOTP



Home Page Screenshot



The integration of Bootstrap 5 for UI components provided a responsive and visually appealing interface, enhancing user experience. Thorough testing of the application confirmed its stability and robustness, with seamless transitions between user actions like registration, login, and logout.

Overall, the project achieved its goal of delivering a secure, maintainable, and user-friendly web application.

10. CONCLUSION

This project highlighted the power and flexibility of Flask as a micro-framework for web development. By adopting best practices such as modular design, environment-based configuration, and secure user authentication, the application serves as a foundational example of a scalable web solution. The combination of Flask-Bcrypt, Flask-Login, and Flask-WTF provided robust security mechanisms, ensuring user data integrity and session safety.

The project highlighted the ease of integrating various Flask extensions to streamline common development tasks, such as database management and migration. Furthermore, using blueprints established a clear structure, making the application extendable for future features or modules. The use of Bootstrap for the front-end emphasized how design frameworks could significantly improve the visual and functional aspects of web applications.

In conclusion, this project not only met its functional requirements but also laid a foundation for scalable and secure application development.

11. FUTURE ENHANCEMENT

One of the key features that could significantly improve the user experience is the implementation of a "Forgot Password" functionality. This feature would allow users to reset their password securely in case they forget their login credentials. By integrating email-based verification or OTP systems, the password reset process can ensure both convenience and security, making the application more robust and user-friendly.

12. WORKLOAD ASSIGNMENT

Kritthika will focus on backend development and system integration. She will design and implement the authentication logic, including

username/password verification and integration with the OTP and TOTP services. Kritthika will also set up the database schema to securely store user credentials and secret keys, ensuring encryption and security best practices.

Jayasudha will take charge of MFA implementation and testing. She will integrate the OTP generation and delivery system and the TOTP functionality using Google Authenticator. Jayasudha will ensure these services work seamlessly with the backend proces. She will conduct unit and integration testing to validate the security and reliability of the MFA process.

SathiyaShivani will handle front-end development and user experience. She will design the user interface for login, MFA selection, and OTP/TOTP entry screens, ensuring they are user-friendly and responsive.

All members will collaborate on documenting and presentation content and finalize the content by splitting each topic and preparing accordingly.

13. REFERENCES

Loubser, N. (2021). Software engineering for absolute beginners : your guide to creating software products (1st ed. 2021.). APress.

Paptan, Honey & Garg, Deepanshu & Kumar, Harsh & Mehta, Komal. (2024). Multi Factor Authentication System.

Reynolds, J., Smith, T., Reese, K., Dickinson, L., Ruoti, S., & Seamons, K. (2018). A tale of two studies: The best and worst of YubiKey usability. In 2018 IEEE Symposium on Security and Privacy (SP) (pp. 872-888). IEEE. https://doi.org/10.1109/SP.2018.00067

Nash, A., Studiawan, H., Grispos, G., & Choo, K. K. R. (2024). Security analysis of Google Authenticator, Microsoft Authenticator, and Authy. In S. Goel & P. R. Nunes de Souza (Eds.), Digital forensics and cyber-crime - 14th EAI International Conference, ICDF2C 2023, Proceedings (pp. 197-206). Springer Science and Business Media. https://doi.org/10.1007/978-3-031-56583-0_13.

Rai, B. (2024, November 20). Step-by-Step guide to implementing MFA in your current system. *Medium*. https://medium.com/@biswas.rai101/step-by-step-guide-to-implementing-mfa-in-your-current-system-b1c19cf2b896

Qureshi, I., & Kale, V. K. (2024). Multi Factor Authentication System. *2024 OPJU International Technology Conference (OTCON) on Smart Computing for Innovation and Advancement in Industry 4.0*, *8*, 1–5.

What is Activity Diagram? (n.d.). https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/