CLOUD COMPUTING TECHNOLOGY PRACTICUM REPORT

AUTHENTICATION

CLOUD COMPUTING TECHNOLOGY PRACTICUM

PLUG - H

By:

Nama : Sabila Khairina Saktiwati

NIM : 123220085

INFORMATICS STUDY PROGRAM

DEPARTMENT OF INFORMATICS

FACULTY OF INDUSTRIAL ENGINEERING

UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN"

YOGYAKARTA

2025

APPROVAL PAGE
HALAMAN PERSETUJUAN
PRACTICUM REPORT

CLOUD SQL AND CLOUD STORAGE
CLOUD COMPUTING TECHNOLOGY PRACTICUM
PLUG - H

By:

Sabila Khairina Saktiwati 123220085

Checked and approved by the Practicum Assistant of Cloud Computing Technology.
At the date of: ……

.

Approved by.

Practicum Assistant	**Practicum Assistant**
Muhammad Rafli, S.Kom.	**Sayang Sani**
NIM 123210078	**NIM 123210044**

PREFACE

Praise and gratitude are expressed to the presence of God Almighty for His blessings and grace, which have enabled the completion of this practicum report. This report is prepared as part of fulfilling academic requirements and as a form of accountability for the practicum activities that have been carried out.

The author would like to express sincere appreciation to Muhammad Rafli S.Kom, and Sayang Sani for their guidance and assistance throughout the practicum.

The author is fully aware that this report is still far from perfect. Therefore, constructive criticism and suggestions are highly expected for future improvements. Hopefully, this report can provide benefits to all relevant parties.

Yogyakarta, 24 May 2025

Author

TABLE OF CONTENTS

LIST OF FIGURE

CHAPTER I
INTRODUCTION

1.1 Background

In the era of modern web applications, authentication is a critical component in ensuring data security, user privacy, and access control. JSON Web Token (JWT) has emerged as a robust standard for stateless and scalable authentication mechanisms, particularly in RESTful architectures. JWT enables secure transmission of claims between parties, allowing user identity verification without maintaining session state on the server (Chaturvedi, 2022). It is widely adopted due to its lightweight structure, cross-platform compatibility, and ability to integrate with various backend technologies.

In developing cloud-based systems, such as a notes management application, integrating JWT authentication becomes increasingly essential. Ahmed and Mahmood (2019) proposed JWT-based schemes to ensure that applications maintain data confidentiality and integrity, especially when deployed across distributed systems. This becomes more relevant when using platforms like Google Cloud Platform (GCP), which supports scalable deployments, instance management, database hosting, and HTTP-triggered APIs—all aligning with the needs of modern, secure web applications.

The implementation of JWT on a cloud-deployed notes app involves integrating login, register, and logout features that interact with GCP-hosted resources such as Firestore and App Engine. This setup not only supports efficient session handling and user management but also ensures that unauthorized access to note data is strictly prohibited. Haekal (2016) and Adam et al. (2020) demonstrated the advantages of JWT authentication on RESTful services, proving its effectiveness in academic and institutional systems.

1.2 Problem Formulation

1. How can JSON Web Token (JWT) be effectively implemented to provide secure login, registration, and logout features in a notes-based web application?

2. How can GCP services (e.g., instances, database, triggers, and deployment tools) be utilized to support JWT-based authentication in a web environment?

1.3 Objectives

1. To implement JWT-based authentication (login, register, and logout) on an existing notes web application.
2. To integrate the authentication system with Google Cloud Platform, using services like Cloud Functions (triggers), and App Engine or Cloud Run for deployment.
3. To ensure secure token storage and proper user verification during each session.
4. To document the implementation process and outcomes in accordance with the academic report requirements.

1.4 Benefits

1. Users can securely log in and manage their personal notes without fear of unauthorized access.
2. Users can register new accounts and automatically receive secure tokens for future access.
3. Users can experience seamless session handling without the need for manual re-authentication.
4. Developers can deploy scalable, cloud-based services using GCP while maintaining a high level of security through JWT.

CHAPTER II

LITERATURE REVIEW

2.1 Google Cloud Instance

Google Cloud Instance refers to the virtual machines (VMs) hosted on Google Compute Engine (GCE) that provide scalable computing resources in the cloud. These instances can be configured with specific CPU, memory, and disk requirements to match the performance needs of different applications. In web application deployment, especially for backend services like authentication or data processing, instances provide a flexible environment where developers can install and run any necessary server-side software.

Instances play a crucial role in hosting web servers and backend APIs for cloud-native applications. For a notes application, they can be used to run authentication services, handle HTTP requests, and perform database operations. The advantage of using instances lies in their ability to scale up or down based on traffic, ensuring cost-effectiveness and availability.

Moreover, instances in GCP integrate well with other cloud services such as Cloud SQL, Cloud Storage, and Firestore, providing a seamless development ecosystem. Developers can use startup scripts, SSH access, and monitoring tools to automate tasks and manage the lifecycle of the instance. This makes them ideal for dynamic web applications where frequent updates and monitoring are essential.

In the context of JWT-based authentication, the backend service that generates and verifies tokens can be deployed on a Google Cloud Instance. This ensures that the service is always online, can handle multiple concurrent requests, and is integrated with the GCP environment for efficient logging, scaling, and monitoring.

2.2 Google Cloud Trigger

Google Cloud Triggers are event-driven mechanisms used in services like Cloud Functions or Cloud Run. A trigger executes a function or service when a specific event occurs—such as an HTTP request, a change in a Firestore document,

or a message arriving in a Pub/Sub topic. Triggers allow developers to build reactive, loosely coupled systems that are scalable and efficient.

For a notes application, triggers are particularly useful when implementing actions that respond to user inputs—such as sending a verification email after registration, logging login attempts, or cleaning up data after account deletion. These operations can be handled asynchronously without blocking the user experience, improving performance and reliability.

The integration of triggers with authentication systems provides additional security and automation. For instance, a trigger can validate JWTs on every request to a protected route or automatically log out users after a period of inactivity. This ensures that the application maintains a high level of security without requiring manual intervention.

Using GCP's serverless architecture, such as Cloud Functions with HTTP triggers, developers can also reduce the complexity of backend infrastructure. This allows focus on logic and functionality, while Google handles server provisioning, scaling, and maintenance.

2.3 Authentication

Authentication is the process of verifying a user's identity before granting access to a system or application. It is one of the core components of modern web applications, ensuring that only authorized users can access or modify protected resources. Traditional authentication approaches use session-based techniques, where a server stores the login state, often leading to scalability challenges.

Modern applications, especially those built on RESTful APIs and microservices, prefer token-based authentication methods such as OAuth 2.0 and JWT. These approaches are stateless, meaning the server does not store session information, allowing better scalability and simpler architecture.

In the context of your notes application, authentication enables users to log in securely and access only their data. This is typically implemented by requiring a username and password, which are validated on the server. If valid, the server issues a token (JWT), which the client uses to authenticate future requests. This method

simplifies user session management and allows APIs to remain stateless and scalable.

Good authentication systems also incorporate security best practices such as rate limiting, password hashing (e.g., using bcrypt), multi-factor authentication (MFA), and HTTPS-only communication. These measures protect against common vulnerabilities like brute force attacks, credential stuffing, and token theft.

2.4 JSON Web Token (JWT)

JSON Web Token (JWT) is a compact, URL-safe means of representing claims between two parties. It is widely used in web authentication systems to securely transmit user identity and privileges. A typical JWT consists of three parts: a header (which defines the algorithm used), a payload (which contains the claims), and a signature (used to verify the token's authenticity).

JWT supports stateless authentication by embedding all necessary user information inside the token, which is signed using a secret or public/private key pair. Once a user logs in, the server issues a JWT and the client stores it—usually in localStorage or sessionStorage. For every subsequent request, the client includes the token in the header, and the server verifies it without needing to store session data.

In a Google Cloud-based notes app, JWTs are ideal for protecting routes that involve user data—such as creating, editing, or deleting notes. Tokens can be verified by middleware before any sensitive operation is allowed. The self-contained nature of JWTs means that services like Google Cloud Functions or App Engine can validate user identity without querying a session store.

Security with JWTs depends heavily on how tokens are handled. It is essential to set expiration times, rotate secrets, use HTTPS to prevent token leakage, and validate the token structure and signature. Implementing these correctly, as shown by Haekal (2016) and Ahmed & Mahmood (2019), results in secure and scalable web applications.

CHAPTER III
METHODOLOGY

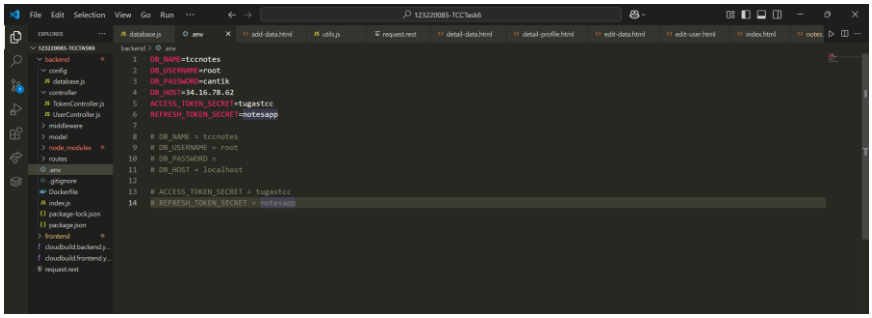
3.1 Problem Analysis

Deploying a notes application with JWT-based authentication presents several technical challenges that require careful analysis and planning. The system must support secure user login and registration, as well as allow authenticated users to create, edit, and delete personal notes. A major concern lies in ensuring that only authorized users can access or modify their own data, which demands reliable token generation, validation, and role enforcement. Additionally, deploying the app on a cloud platform such as Google Cloud requires proper configuration of services like instances, triggers, and HTTP endpoints to ensure scalability, security, and availability. Mismanagement of token storage, inadequate route protection, or insecure data access rules can result in vulnerabilities such as unauthorized access or data leakage.

3.2 Solution Design

3.2.1 Create File in Backend Folder for Authentication

1. Config database.js with .env



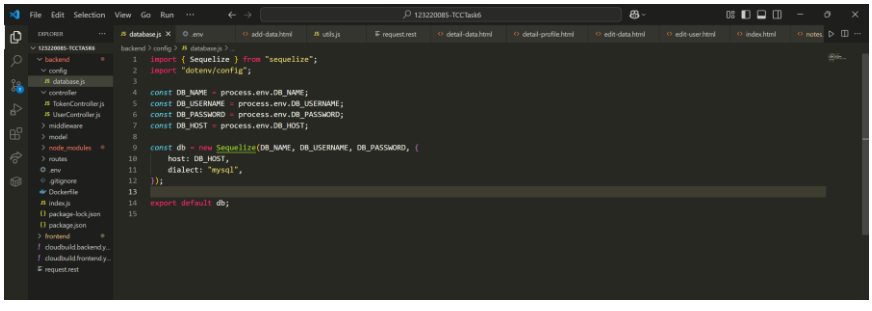
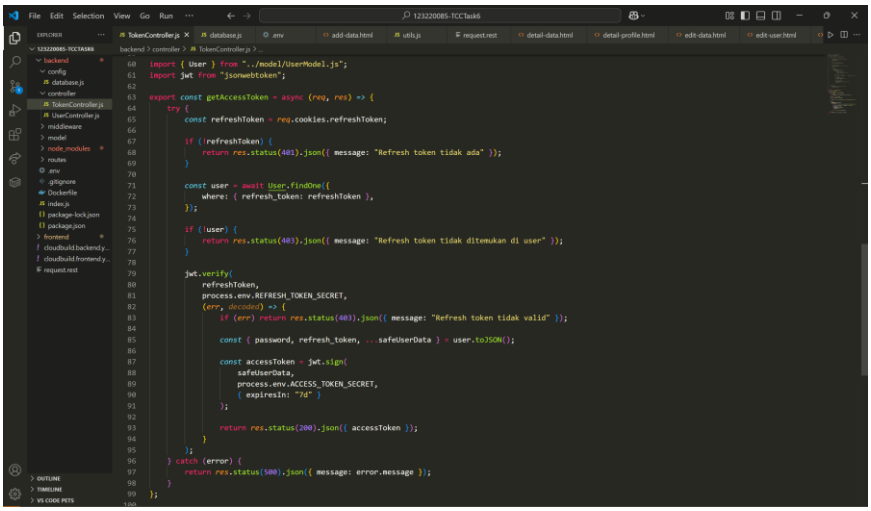


Figure 1. Step 3.2.1.1

2. Create Files in Controller Folder (`getAccessToken, login, logout, register`) and User, Notes CRUD



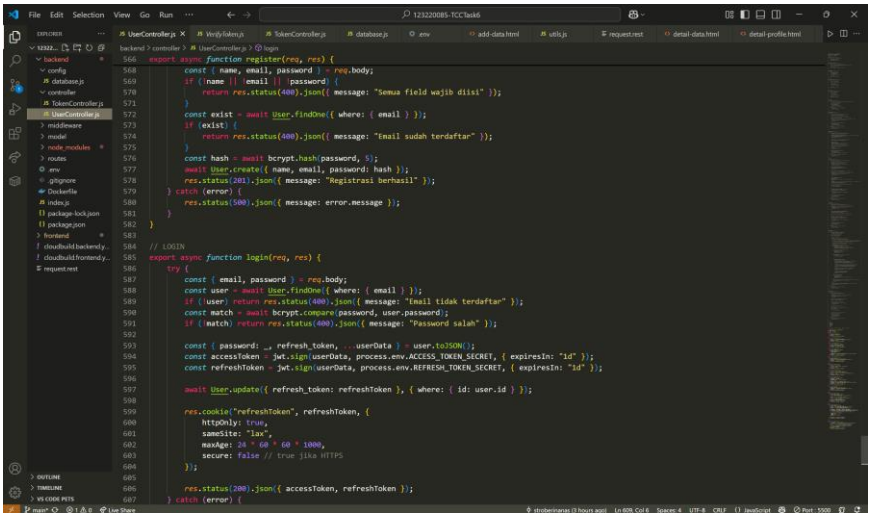


Figure 2. Step 3.2.1.2

3. Create verifyToken.js in Middleware folder

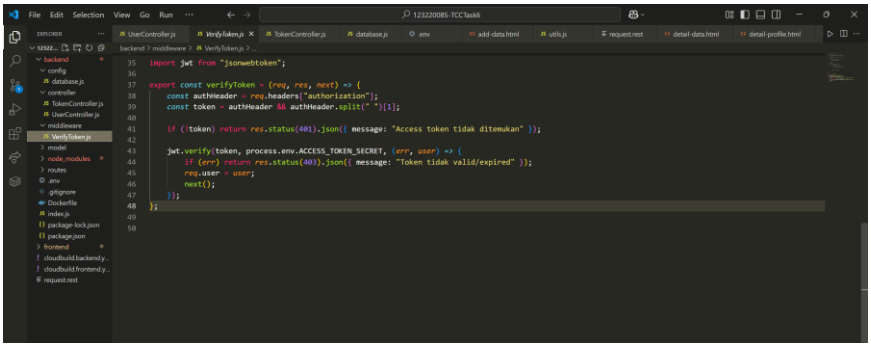


Figure 3. Step 3.2.1.3

4. Create userModel.js in Model folder

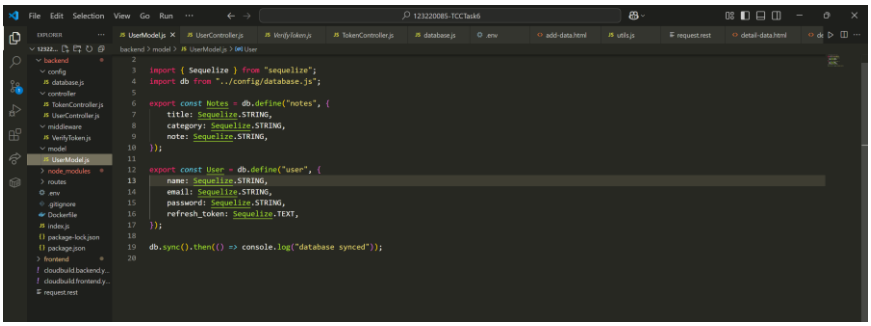


Figure 4. Step 3.2.1.4

5. Create index.js

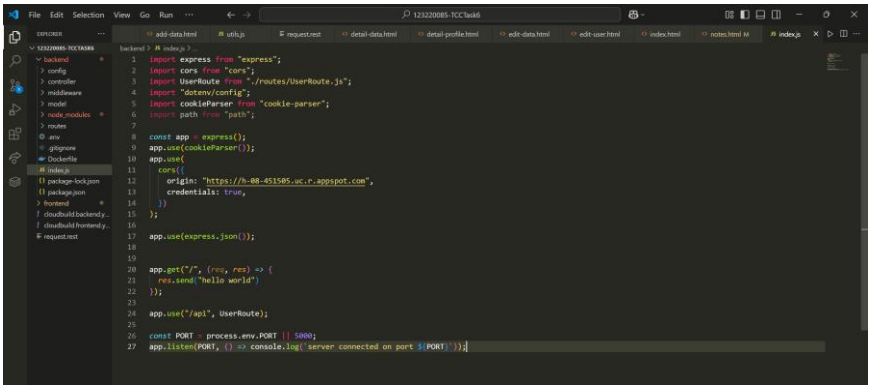


Figure 5. Step 3.2.1.5

3.2.2 Create File in Frontend Folder for Login and Register Page

1. Create index.html for user and register page

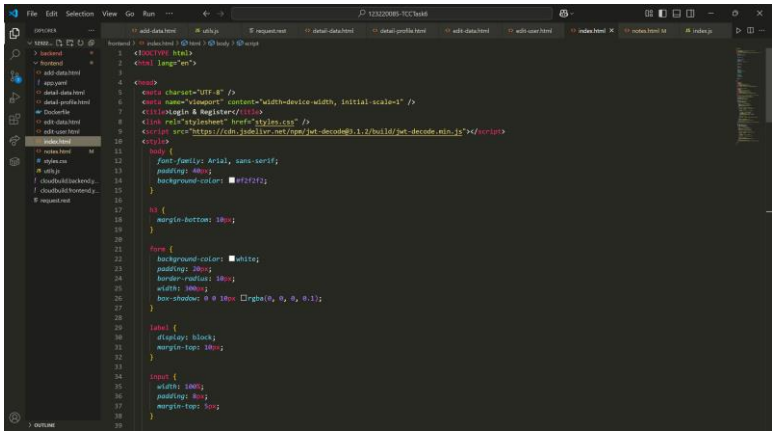


Figure 6. Step 3.2.2.1

2. Create detail-profile.html for the user information page

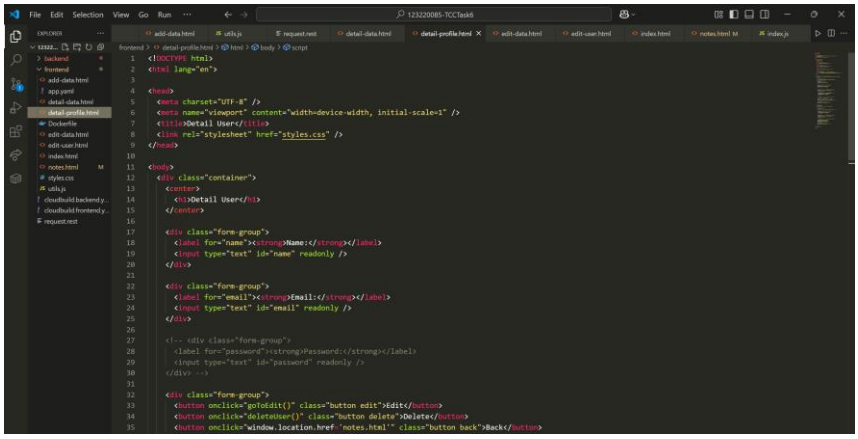


Figure 7. Step 3.2.2.2

3.2.3 Config and Deploy to Google Cloud Platform

1. Push File to Github https://github.com/stroberinanas/notesapp.git
2. Enable Instance

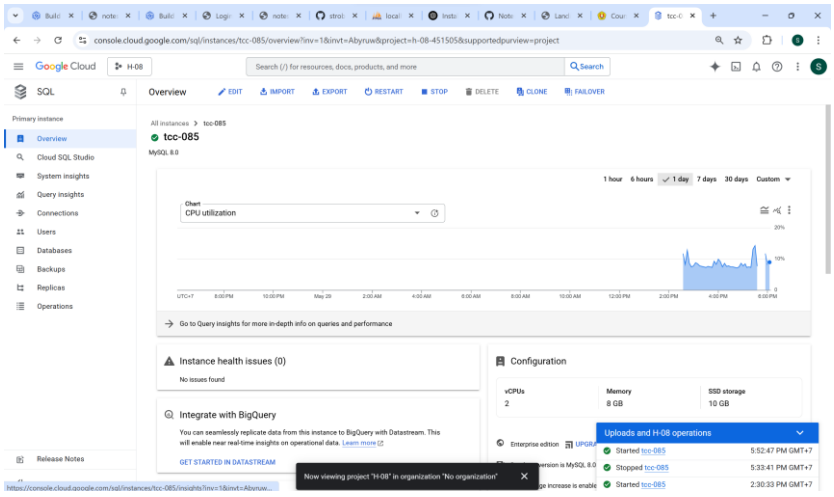


Figure 8. Step 3.2.3.2

3. Config Trigger suitable with .env file

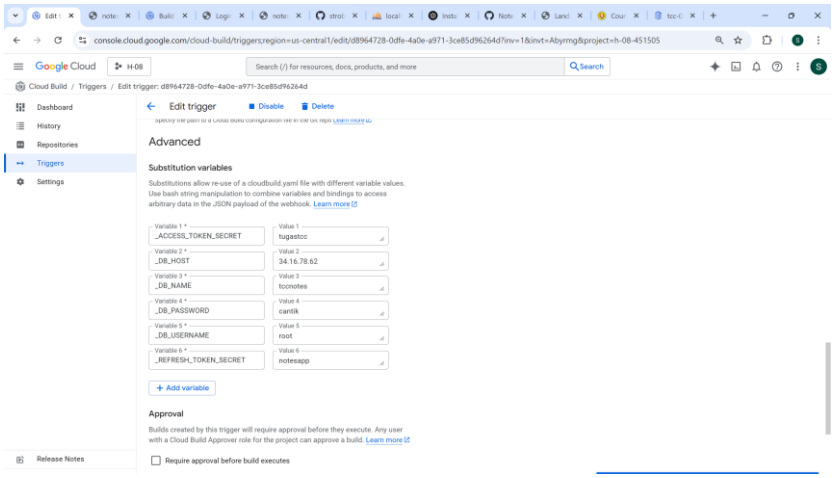


Figure 9. Step 3.2.3.3

4. Start trigger for be and open the url

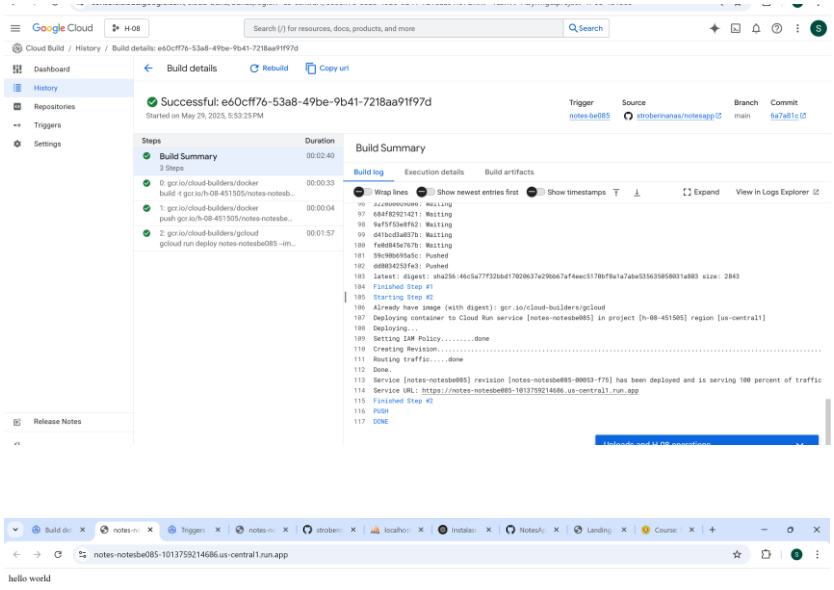
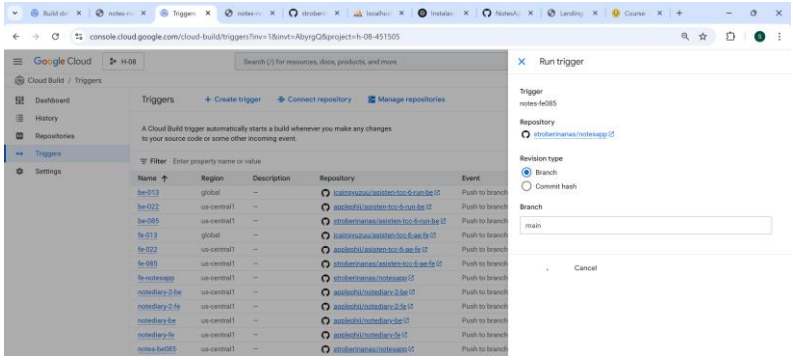


Figure 10. Step 3.2.3.4

5. Start trigger for fe and open the url



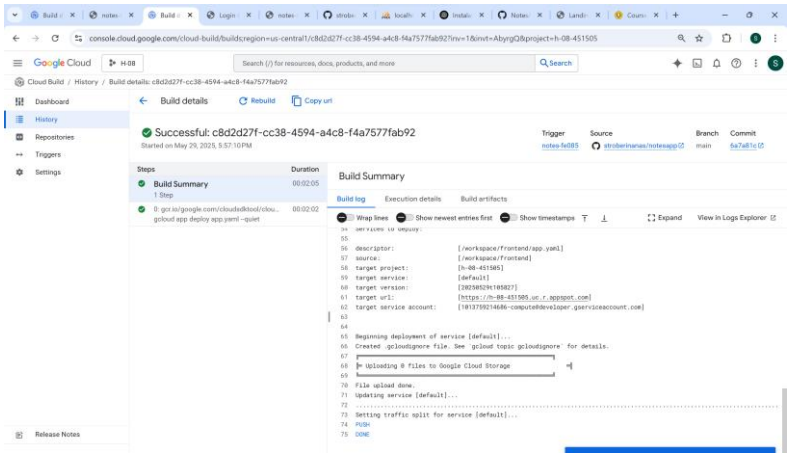
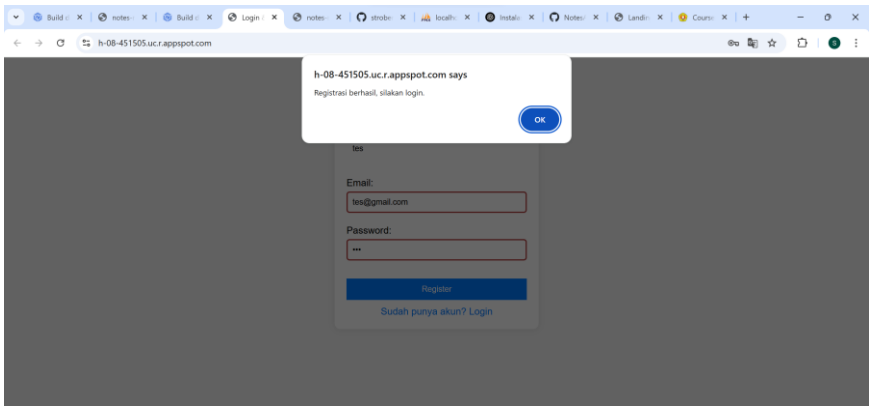
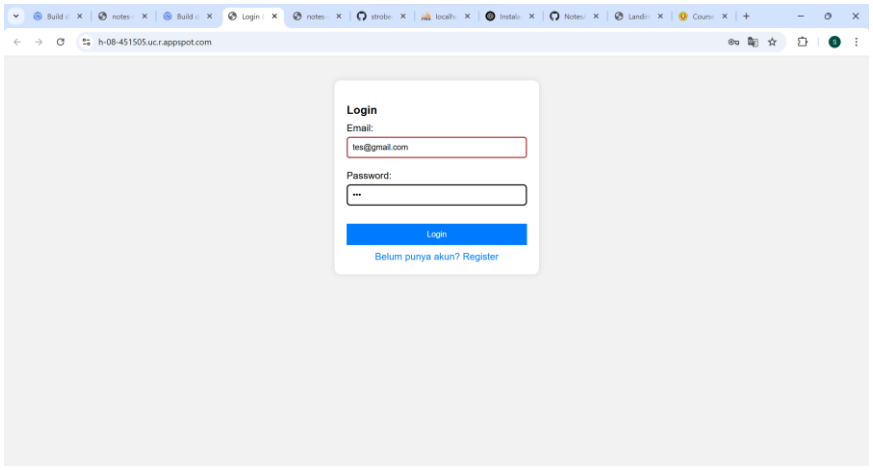


Figure 11. Step 3.2.3.5

6. Check all the User's Features





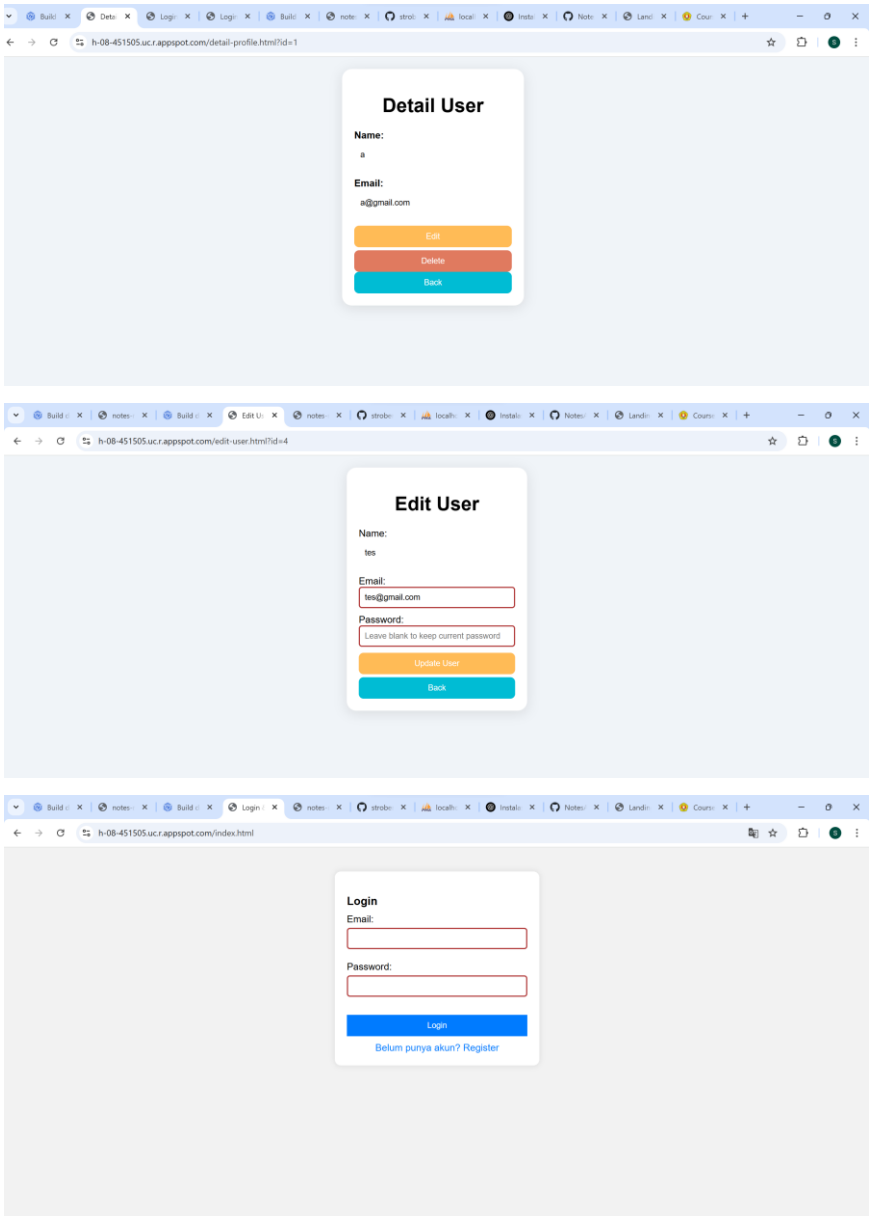
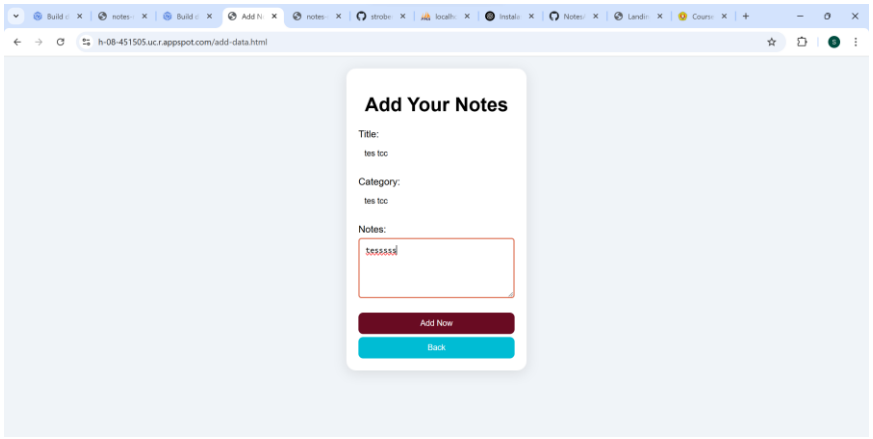
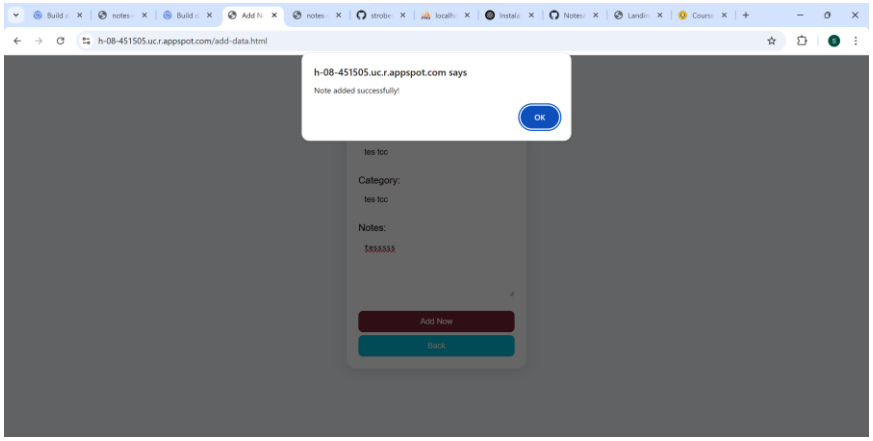
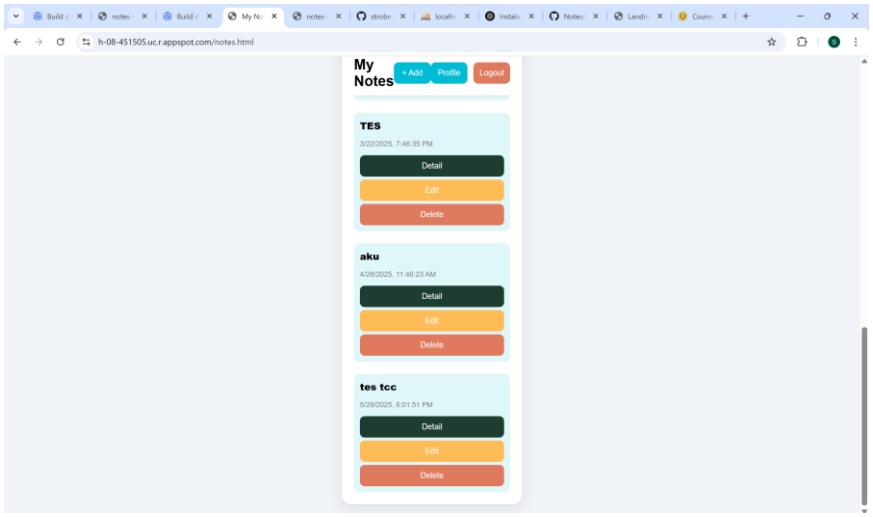


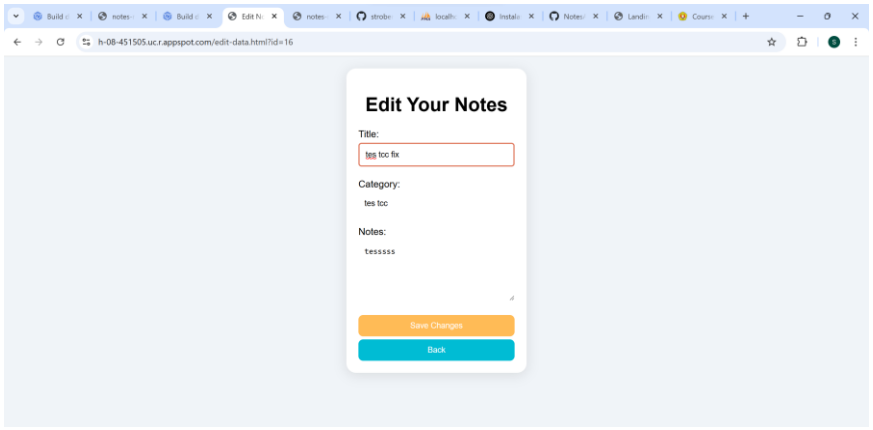
Figure 12. Step 3.2.3.6

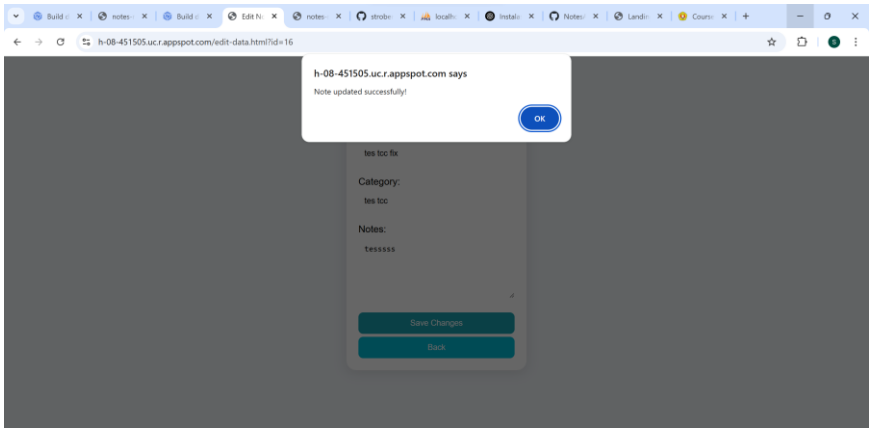
7. Check all the Notes Features

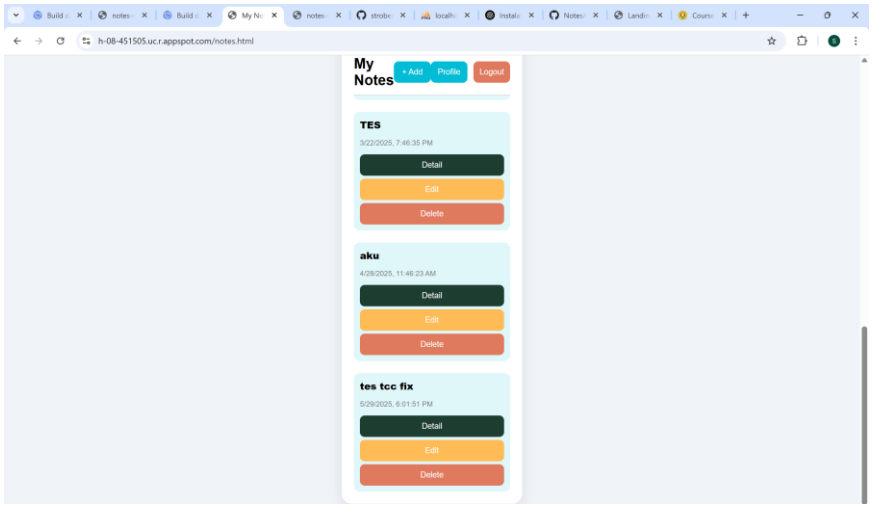

h-08-451505.uc.r.appspot.com/add-data.html
Add Your Notes
Title:
tes tcc
Category:
tes tcc
Notes:
tesssss
Add Now
Back


h-08-451505.uc.r.appspot.com/add-data.html
h-08-451505.uc.r.appspot.com says
Note added successfully!
OK
tes tcc
Category:
tes tcc
Notes:
tesssss
Add Now
Back


h-08-451505.uc.r.appspot.com/notes.html
My Notes
+ Add
Profile
Logout
TES
3/22/2025, 7:46:35 PM
Detail
Edit
Delete
aku
4/28/2025, 11:46:23 AM
Detail
Edit
Delete
tes tcc
5/29/2025, 6:01:51 PM
Detail
Edit
Delete


h-08-451505.uc.r.appspot.com/edit-data.html?id=16
Edit Your Notes
Title:
tes tcc fix
Category:
tes tcc
Notes:
tesssss
Save Changes
Back


h-08-451505.uc.r.appspot.com/edit-data.html?id=16
h-08-451505.uc.r.appspot.com says
Note updated successfully!
OK
tes tcc fix
Category:
tes tcc
Notes:
tesssss
Save Changes
Back


h-08-451505.uc.r.appspot.com/notes.html
My Notes
+ Add
Profile
Logout
TES
3/22/2025, 7:46:35 PM
Detail
Edit
Delete
aku
4/28/2025, 11:46:23 AM
Detail
Edit
Delete
tes tcc fix
5/29/2025, 6:01:51 PM
Detail
Edit
Delete

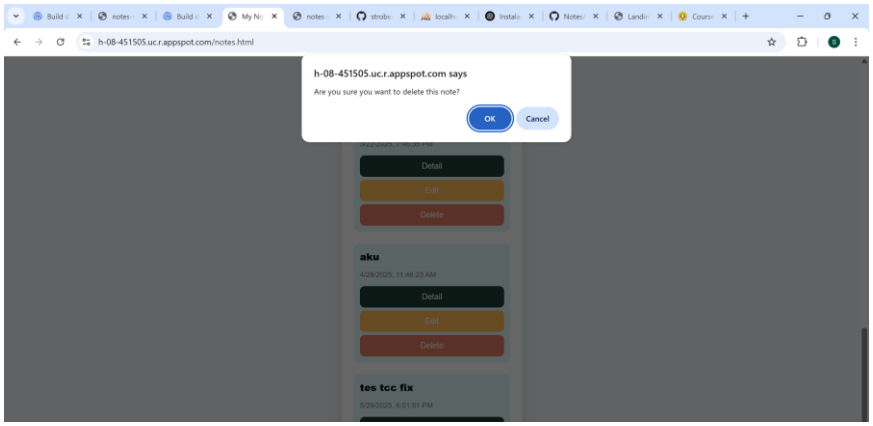


Figure 13. Step 3.2.3.7

CHAPTER IV
RESULTS AND DISCUSSION

4.1 Results

4.1.1 Deploying Back-End and Front-End

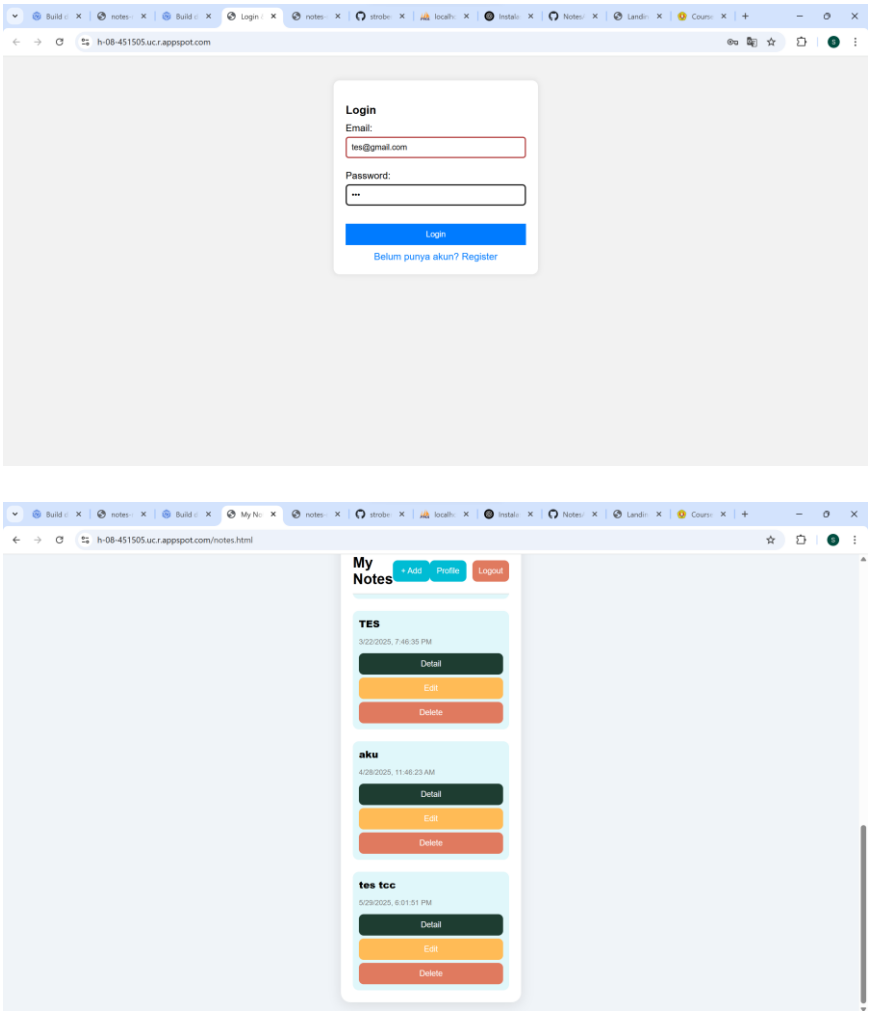


Figure 14. Deploying BackEnd and FrontEnd Succesfully

This figure illustrates the successful implementation and deployment of JWT-based authentication in the notes application as part of Task 7. The top screenshot shows the login page where users are required to enter their email and password. Upon successful authentication, a JSON Web Token (JWT) is issued by the back-end server and stored securely on the client side (in this case, localStorage). This token is then included in the Authorization header for all protected API requests. The bottom screenshot displays the notes dashboard that becomes accessible only after the user is authenticated. The dashboard shows user-

specific notes fetched using the JWT and provides options to view details, edit content, or delete notes. The system also supports session persistence and secure logout functionality. All authentication and authorization processes are handled using JWT, ensuring secure and stateless communication between front-end and back-end services. This implementation demonstrates the ability to protect routes, manage session tokens, and enforce user-based access control in a full-stack web application.

4.2 Discussion

The implementation of JSON Web Token (JWT) provides a robust foundation for managing user sessions without the overhead of maintaining server-side state. Upon successful login or registration, users are issued a signed token that is then used to authenticate further actions, such as creating or modifying notes. This stateless mechanism simplifies the backend structure and improves scalability, especially in a cloud environment like GCP. By integrating services such as Google Compute Engine (for the API server), Firestore (for real-time data storage), and HTTP triggers (for cloud function execution), the application gains elasticity and responsiveness. However, token management must follow best practices—such as setting expiration times, using HTTPS, and verifying token signatures—to prevent misuse. Furthermore, GCP deployment introduces its own complexities, including configuring service permissions, managing environment variables securely, and ensuring proper firewall and CORS policies are enforced.

CHAPTER V
CLOSING

5.1 Conclusion

The deployment of a cloud-based notes application using JWT authentication effectively demonstrates the synergy between modern security standards and cloud-native infrastructure. JWT enables secure, stateless user sessions while allowing the application to maintain high performance and modularity. With the help of GCP services like Compute Engine, the backend becomes scalable and manageable, supporting dynamic user interactions with high availability. Overall, the implementation meets the goals of security, reliability, and usability, though it requires attention to cloud-specific configurations and best security practices.

5.2 Suggestions

Future improvements should focus on enhancing security and user experience. Implementing refresh tokens and multi-factor authentication (MFA) can further secure the login process. Adding real-time feedback and UI enhancements can help users manage notes more efficiently. On the deployment side, integrating CI/CD pipelines would streamline updates, while monitoring tools like Google Cloud Monitoring can help identify performance bottlenecks or security threats. Lastly, to support scalability for larger user bases, migrating to containerized environments (e.g., using Cloud Run or GKE) could be considered for better resource management.

BIBLIOGRAPHY

[1] Chaturvedi, A. (2022, June). Comparison of different authentication techniques and steps to implement robust JWT authentication. In *2022 7th International Conference on Communication and Electronics Systems (ICCES)* (pp. 772-779). IEEE.

[2] Ahmed, S., & Mahmood, Q. (2019, November). An authentication based scheme for applications using JSON web token. In *2019 22nd international multitopic conference (INMIC)* (pp. 1-6). IEEE.

[3] Haekal, M. (2016, October). Token-based authentication using JSON web token on SIKASIR RESTful web service. In *2016 International Conference on Informatics and Computing (ICIC)* (pp. 175-179). IEEE.

[4] Adam, S. I., Moedjahedy, J. H., & Maramis, J. (2020, October). RESTful web service implementation on Unklab information system using JSON web token (JWT). In *2020 2nd International Conference on Cybernetics and Intelligent System (ICORIS)* (pp. 1-6). IEEE.

[5] Hunter, T., & Porter, S. (2018). *Google Cloud Platform for developers: build highly scalable cloud solutions with the power of Google Cloud Platform*. Packt Publishing Ltd.

APPENDICES

[1] https://github.com/stroberinanas/123220085-TCCPract.git

[2] https://github.com/stroberinanas/notesapp.git