Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 8, Number 2, April 2024
DOI : https://doi.org/10.33395/v8i2.13386
e-ISSN : 2541-2019
p-ISSN : 2541-044X

Implementation of App Engine and Cloud Storage as REST API on Smart Farm Application

Khoirul Azkiya[1)*], Muhamad Irsan[2)], Muhammad Faris Fathoni[3)]
[1)2)3)]Telkom University, Indonesia
[1)]khoirul.azkya@gmail.com, [2)]irsanfaiz@telkomuniversity.ac.id, [3)]mfarisfwork@telkomuniversity.ac.id

Submitted :Jan 11, 2024 | **Accepted** : Jan 14, 2024 | **Published** : Apr 1, 2024

Abstract: Smart Farm is an agricultural application that uses machine learning and cloud computing technology to improve efficiency in the farming process. Technological advancement and sustainable agriculture are two essential aspects of supporting global food security. This research investigates the implementation of App Engine and Cloud Storage in developing REST API in Smart Farm applications. By utilizing cloud computing technology, such as App Engine, and cloud storage, such as Cloud Storage, we can create efficient solutions to monitor and manage agriculture better. This research implements an App Engine and Cloud Storage to develop a REST API that allows Smart Farm application users to access data and control farming devices efficiently. The authors designed, developed, and tested this system to ensure optimal performance and reliability in agricultural data collection and distribution. This method has several significant advantages. First, App Engine allows for easy scalability, ensuring the system can handle increased data demand without disruption. Secondly, Cloud Storage provides secure and scalable storage for agricultural data, which can be accessed from anywhere. This provides easy and quick access to critical data for farmers. Moreover, the use of cloud technology also reduces infrastructure and maintenance costs. The developed system integrates the App Engine and Cloud Storage with the Smart Farm application. The App Engine is a processing engine that receives user requests via the REST API, processes the required data, and provides appropriate responses. Like image data, farm data is stored and managed on Cloud Storage. Users can access this data through the Smart Farm app or other devices, enabling better farming monitoring and decision-making.

Keywords: REST API, App Engine, Cloud Storage, Smart Farm, black box.

INTRODUCTION

Agriculture has become the main focus of efforts to maintain food security in a globalized era that demands sustainability (Harison et al., 2017). In Indonesia, agriculture is a source of livelihood and a key factor in meeting people's needs (Harison et al., 2017). Unfortunately, the efficiency of farmers' work is often less than optimal. The decisions made affect agricultural yields and increase the time spent (Achyar et al., 2020). Although some have adequate knowledge of the agricultural sector, many farmers in Indonesia continue to rely on personal experience and instinct in making decisions on the farm (Achyar et al., 2020).

The rapid development of technology has facilitated the development of web services and web-based applications as a solution in supporting various human activities. Many web applications today are inseparable from two main components: front end and back end. The front end is the part of the web application that interacts directly with the user, while the back end includes aspects of the server, application, and database (Guntara & Azkarin, 2023). Back-end development focuses on the server-side aspects. The program code of the back-end application allows interaction between the browser and retrieval of data from the database (Guntara & Azkarin, 2023).

So, the Smart Farm application was developed using Application Programming Interface (API) technology. API is a software interface that contains a collection of instructions or functions, allowing communication between various software applications (Ariantara et al., 2020). Representational State Transfer (REST) is an API architecture that facilitates the transfer and request of data over the HTTP protocol, known for its ease in the mobile cloud paradigm (Suzanti et al., 2020). The REST API here will integrate the application with the server and database. This application is planned to support farmers by providing quick access through the application to

* Corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 8, Number 2, April 2024
DOI : https://doi.org/10.33395/v8i2.13386
e-ISSN : 2541-2019
p-ISSN : 2541-044X

optimize the farmer's work process.

This application development incorporates cloud computing technologies, such as App Engine for building web applications and Cloud Storage for data storage. App Engine provides efficient infrastructure, while Cloud Storage provides flexibility in storing the application's data needs (Cloud, 2023). They are an essential part of the cloud technology infrastructure that supports multiplatform application development and management. Through web service architecture, applications can be integrated with various platforms through the same protocol, enabling more efficient communication between programs (Choirudin & Adil, 2019).

By implementing App Engine and Cloud Storage as REST API on the Smart Farm application, it is expected that farmers can access the application and upload images of their plants more efficiently. App Engine and Cloud Storage as REST API on Smart Farm application will increase farmers' productivity, reduce the risk of crop failure due to plant diseases, and strengthen the country's food security. Using technology, Indonesia is expected to become 'Lumbung Padi Asia' capable of producing high-quality rice and maintaining stable and quality food availability.

LITERATURE REVIEW

Related Research

Research related and relevant to the background of the Smart Farm application problem. Table 1 is a summary of previous research related to this research, as shown in Table 1.

Table 1. Related Research

Author / Year	*Titles*	*Problems*	*Solutions*	*Results*
Wiji Sulistiani, Wiwin Sulistyo / 2020	Implementasi Web Service dengan Metode REST Berbasis Golang pada Layanan Google Cloud Platform di PT Sumber Alfaria Trijaya, Tbk	PT Sumber Alfaria Trijaya, Tbk, faced problems synchronizing master data between on-premises servers in branch offices and cloud storage on the Google Cloud Platform. This problem impacts the Master Data Management (MDM) strategy.	Implemented Golang-based Web Service application with REST method and connected on-premises application with cloud storage. Allows real-time access to synchronized master data, optimally used for various business needs in multiple applications and divisions.	The results showed that the implementation successfully synchronized the Master Data thoroughly. The system running on Google Cloud Platform services and accessed through the MasterStore application runs smoothly, facilitating fast data exchange with smaller data sizes in JSON format. This gives management better capabilities in making business decisions.
Adam Firdaus, Dean Apriana Ramadhan / 2021	Pengembangan Back End Berbasis REST API pada Sistem E-Partisipasi dan E-Inisiatif Patriot Pangan	The Food Insecurity Agency has a system that needs to be more responsive to detect food insecurity in the community. The limitation of public participation in the Patriot Pangan application is limited to data reporting only, so it is necessary to develop features in the application that aim to attract public involvement.	This research involves the development of a REST API-based backend using Node.JS with Express.Js framework and MongoDB non-relational database. This research also includes the implementation of access tokens in the form of JSON Web Tokens (JWT) to improve API access security.	The development of the REST API-based back end of the E-Participation and E-Initiative Food Patriot application uses Node.JS with the Express.Js framework and the MongoDB non-relational database. This development was carried out using the scrum method for five sprints and resulted in a total of 23 APIs that have been successfully tested. This system is expected to help detect food insecurity cases early and help the community achieve crowdfunding to overcome food insecurity cases quickly.

* Corresponding author

Rangga Gelar Guntara, Varinia Azkarin / 2023	Implementasi dan Pengujian REST API Sistem Reservasi Ruang Rapat dengan Metode Black Box Testing (Studi Kasus: PT Lizzie Parra Kreasi)	Highlighting the shortcomings in the backend application and database management system that still needs to be available to facilitate the management of meeting room reservation data at PT Lizzie Parra Kreasi.	REST API development is needed as an intermediary between the front end and the server to facilitate data exchange. The author designed and implemented a REST API using the Node.js platform to improve the efficiency of the reservation system at PT Lizzie Parra Kreasi.	Implementing the REST API using Node.js provides superior performance compared to implementing the REST API using the PHP programming language. Testing with the black box testing method also shows that each function on each API endpoint runs according to the expected output, both through the HTTP GET, POST, PUT, and DELETE methods. These findings confirm that the REST API for the meeting room reservation system at PT Lizzie Parra Kreasi can be designed and implemented effectively, allowing the client application to exchange data with the database. Thus, the reservation history can be stored properly.

Representational State Transfer

REST (Representational State Transfer) is a standard protocol for web communication using Hypertext Transfer Protocol (HTTP) links (Susanti & Mailoa, 2020). It enables clients to send requests via HTTP methods, and servers respond with REST Responses. REST messages consist of headers and bodies. The HTTP header records each transaction on HTTP, while the body contains the data to be sent (Susanti & Mailoa, 2020). The primary REST methods include GET, POST, PUT, and DELETE (Perkasa & Setiawan, 2018).

One of REST advantages is its utilization of commonly used Internet communication protocols like HTTP. It is lightweight and widely adopted in cloud-based API development by companies such as Amazon, Microsoft, and Google (Nurfauzia, 2021). Due to this, REST is easily implementable across various platforms. Services on the web that adhere to REST principles are often referred to as RESTful API.

App Engine on Google Cloud Platform

Google's App Engine serves as a hosting service designed to assist developers in running web applications without the need to delve into intricate infrastructure aspects (Yusrizal et al., 2017).

Given its utilization of a non-relational database system, App Engine proves suitable for managing and storing unstructured data (Nugroho & Mustofa, n.d.). Within the Smart Farm context, the App Engine functions as a pivotal platform for hosting cloud-based applications. Its roles include backend management, provision of REST API, seamless integration with Cloud Storage, and fortification of application security measures.

Furthermore, App Engine offers robust security and authentication functionalities, enhancing the protection of sensitive data within Smart Farm applications.

Cloud Storage on Google Cloud Platform

Cloud Storage is commonly characterized as a digital data storage service accessible over the Internet. Its utilization presents several advantages, including heightened data security, adaptable data accessibility, and a reduced risk of data loss or corruption (Farizy & Eriana, 2011).

Within the realm of Smart Farming, Cloud Storage serves as a pivotal solution for storing various data types, including images and configuration files. This strategic usage enables Smart Farms to optimize their data management and analysis processes, fostering effective and efficient operations.

Node.js

Node.js stands as a JavaScript framework utilized in web development, catering to both client and server-side

* Corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 8, Number 2, April 2024
DOI : https://doi.org/10.33395/v8i2.13386
e-ISSN : 2541-2019
p-ISSN : 2541-044X

scripting (Kurniawan et al., 2020). In the context of Smart Farm applications, Node.js serves as the backend technology for implementing REST API.

Its primary strengths lie in delivering exceptional responsiveness and performance, facilitating seamless integration with various devices, and optimizing applications to enhance efficiency and productivity within farming operations.

MongoDB

MongoDB stands out as a NoSQL database designed for application development. Unlike traditional SQL databases, MongoDB stores documents in BSON (Binary JSON) format, enhancing readability (Bhaswara et al., 2017). Its architecture offers flexibility, enabling seamless database scaling and integration with a diverse array of programming languages.

In the context of Smart Farm applications, MongoDB proves invaluable as a data storage solution, particularly for managing access information of farmers who have registered accounts within the Smart Farm system.

JSON

JSON stands for JavaScript Object Notation, a data exchange format invented by Douglas Crockford in 2006 (Warsito et al., 2017). Designed to be understood by both computers and humans (Triawan & Prasetiyo, 2019). JSON has a lighter data size than XML, so the data exchange process becomes faster (Warsito et al., 2017). The use of JSON format is necessary in designing applications that use RESTful API and Web Service methods, as it allows efficient storage and data exchange between various application components (Warsito et al., 2017).

Black Box Testing

Black Box testing focuses on the functionality of the software without paying attention to details (Wijaya & Astuti, 2021). This method involves observing the execution results through test data, thus emphasizing the system's functionality (Hanifah et al., 2012). A series of tests on inputs and functions are carried out to find errors or errors so that the system can be improved and run as expected (Wijaya & Astuti, 2021; Wiradiputra et al., 2021). In the context of testing in the Smart Farm application, Black Box Testing with the Equivalence Partitions technique will be used. This technique helps prepare test cases and test functionality and detect input errors on the test page (Wijaya & Astuti, 2021). Testing is done by entering data that does not match the data type or using random data (Hanifah et al., 2012).

METHOD

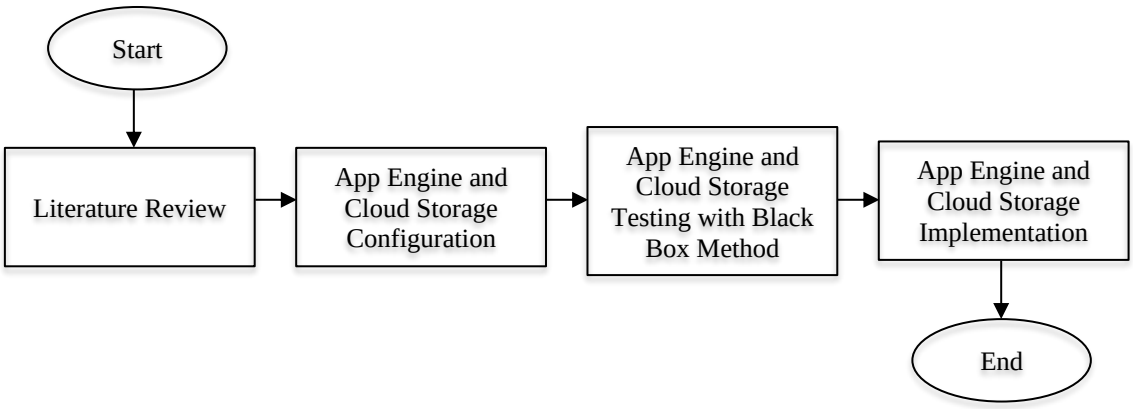

Figure 1. Research Method

The core of this research is the implementation of App Engine and Cloud Storage as REST API in the Smart Farm application. It involves concept understanding, requirement identification, designing, and practical implementation. Concept understanding is necessary to configure App Engine and Cloud Storage.
After understanding what is needed, the author will design a system so that the App Engine and Cloud Storage can be integrated into a REST API, making it easier to access data and services. This involves configuring the App Engine and Cloud Storage. Next is the technical implementation to develop the REST API using NodeJS, which will bridge the Smart Farm application and the cloud, as shown in Figure 1.

* Corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 8, Number 2, April 2024
DOI : https://doi.org/10.33395/v8i2.13386
e-ISSN : 2541-2019
p-ISSN : 2541-044X

This step results in a working URL or endpoint to access data and services in Cloud Storage via the App Engine. Through this testing, there is great potential to improve the performance of the Smart Farm application and enable more efficient use of cloud technology.

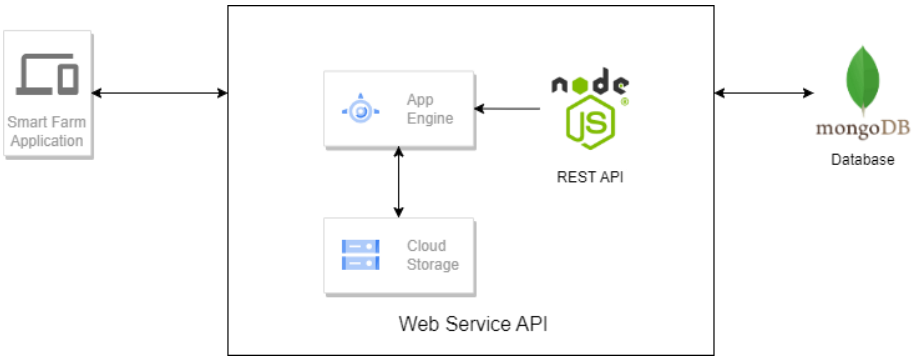

Figure 2. System Architecture

The system architecture in this research helps organize how the required components, such as App Engine, Cloud Storage, REST API as a Web Service API, and MongoDB as a Database, will interact, as shown in Figure 2. This system architecture will be helpful in planning functionality or structure to achieve the desired goals. App Engine provides computing and storage services for web and mobile applications and runs applications created with various programming languages and frameworks. Cloud Storage provides data storage services in the cloud and supports various data types, including images. NodeJS, as a REST API, will provide an interface to access data and functionality from cloud services. The interaction on the App Engine will access data from Cloud Storage using NodeJS as a REST API; the App Engine will process the data and generate a response, and then the response will be sent to the user via the REST API.

The method that will be used in this research is the Black Box Testing method. Black Box Testing will focus on the functionality of the Endpoint generated by the App Engine without paying attention to details. The technique that will be used in this Black Box Testing is Equivalence Partitions. This technique will compile test cases and functionality and detect input errors on the test page (Wijaya & Astuti, 2021). Testing is done by entering data that does not match the data type or using random data (Hanifah et al., 2012).

RESULT

Endpoint Design

The following Endpoint design includes URI (Uniform Resource Identifier). This character identifies from a web, or it can also be called the location of a web, which functions to process input from the user. Method or HTTP Method is the method of HTTP Request that will be done on the URI, and Process is what will happen if the user accesses the URI, as shown in Table 2.

Table 2. Endpoint Design

Method	*URI*	*Process*
GET	/	Main page
POST	/api/v1/users/signin	User logging in
POST	/api/v1/users/signup	Registering user into the database
POST	/api/v1/uploads	User uploads an image

Cloud Computing Implementation

The following Endpoint design includes URI (Uniform Resource Identifier). This character identifies from a web, or it can also be called the location of a web, which functions to process input from the user. Method or HTTP Method is the method of HTTP Request that will be done on the URI, and Process is what will happen if the user accesses the URI.

The results obtained from the App Engine Implementation are that the NodeJS script created and deployed on the App Engine will display the main page of the REST API, which uses the HTTP Method GET as an HTTP Request, as shown in Figure 3.

* Corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 8, Number 2, April 2024
DOI : https://doi.org/10.33395/v8i2.13386
e-ISSN : 2541-2019
p-ISSN : 2541-044X

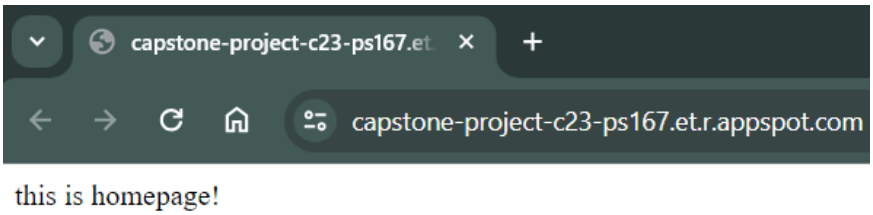

Figure 3. App Engine Implementation Result

The results obtained from the Cloud Storage Implementation are buckets that store images from user upload input in the Smart Farm application, as shown in Figure 4.

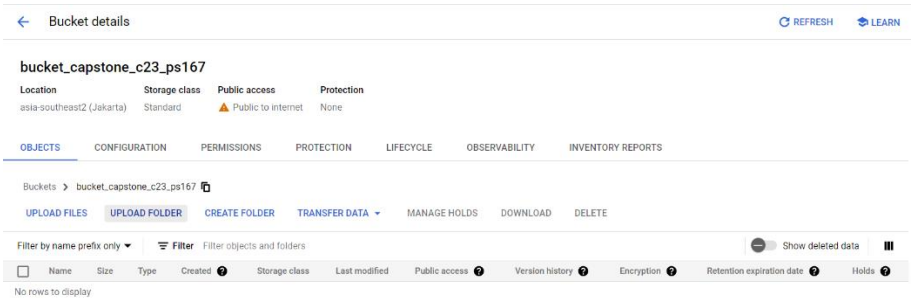

Figure 4. Cloud Storage Implementation Result

Endpoint Implementation

After designing, the author implements each endpoint, and the author implements the App Engine with the database when the user logs in or registers in the Smart Farm application. The author tests the endpoint using the Postman application, which functions to manage HTTP Request and HTTP Method on a REST API that has been created.

The payload generates JSON as a message that the image
has been successfully uploaded to Cloud Storage with data containing a URL from the image that can be accessed, as shown in Figure 5.

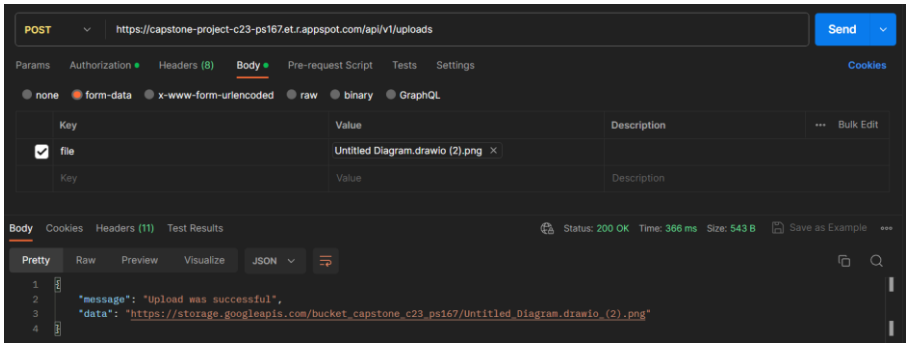

Figure 5. Result of API User Uploads an Image

The payload will produce JSON in the form of a success status, which means that the request requested by the user when inputting data into the Smart Farm application has been successfully received by the server, as shown in Figure 6. The payload token is JSON Web Token (JWT), which provides an identifier to the user that the user has logged in. Payload data contains details of user input from the Smart Farm application during registration; the password will be encrypted with the crypt method. The data has been added to MongoDB as collections, as shown in Figure 7.

* Corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 8, Number 2, April 2024
DOI : https://doi.org/10.33395/v8i2.13386
e-ISSN : 2541-2019
p-ISSN : 2541-044X

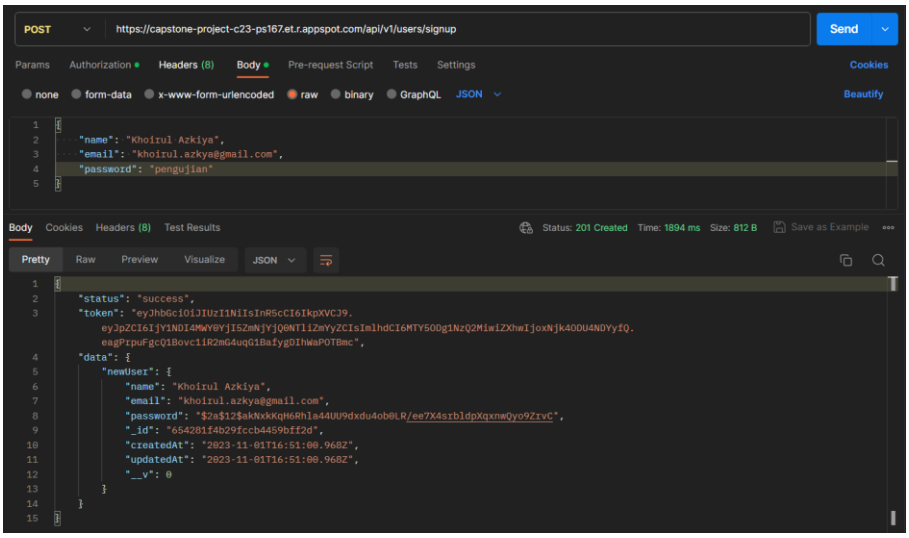

Figure 6. Result of API User Account Registration

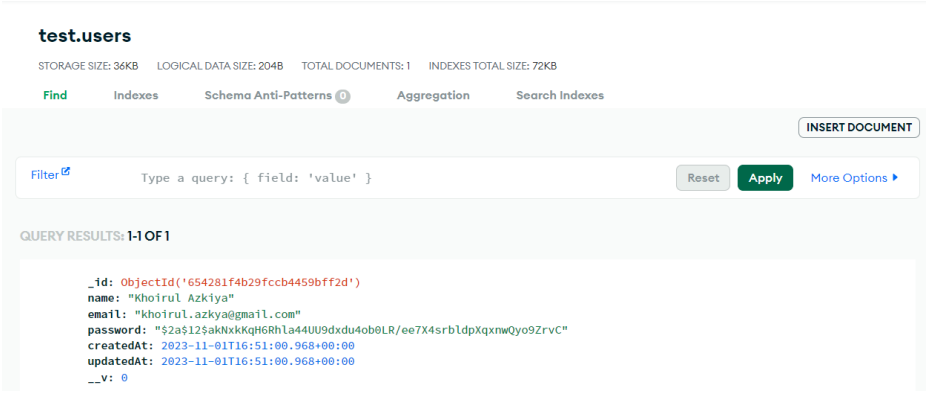

Figure 7. Results of Account Data in MongoDB

Next, the author logs in to the Smart Farm application. The resulting payload is a message that has successfully logged into the Smart Farm application with the identifier given to the token in JSON Web Token, as shown in Figure 8.

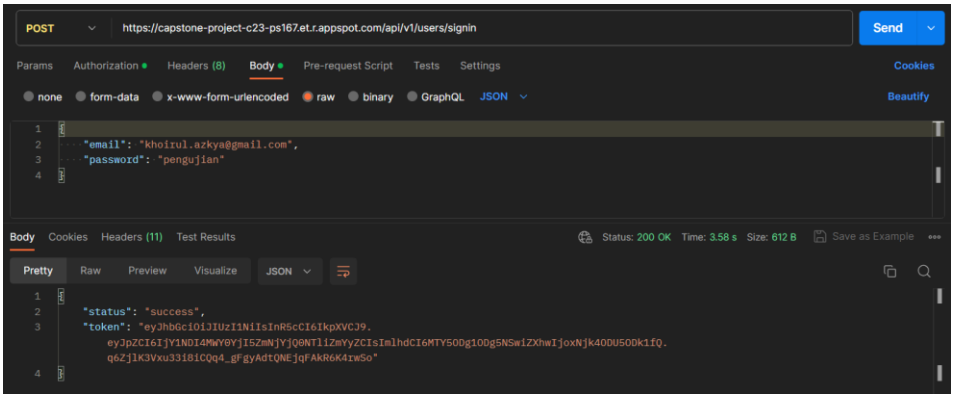

Figure 8. Result of API User Account Login

Testing

After the implementation stage, the next step is to test the API with the black box method. This stage aims to determine whether the function of the application has run well and following the expected output. Testing is carried out on each endpoint, with the results presented in the following Table 3.

Table 3. Results of Black Box Testing

Testing Activities	*Test Case*	*Expected Outputs*	*Results*	*Conclusion*

* Corresponding author

e-ISSN : 2541-2019
p-ISSN : 2541-044X

User logging in	Enter a value in the JSON key email and password	A success message and JSON Web Token as the user identifier are displayed.	Data Match	Valid
Registering user into the database	Enter values in the JSON keys name, email, and password.	A success message appears and the user data has been entered into the database.	Data Match	Valid
User uploads an image	Enter the value of the image file in the form-data.	A successful upload message appears and the image URL data on Cloud Storage.	Data Match	Valid

DISCUSSION

This research has focused on developing a Smart Farm application by integrating App Engine technology and Cloud Storage as a REST API. The API Endpoint design, involving URIs, HTTP methods, and processes, provides a clear structure for the application's interaction with the server and database. The implementation of App Engine and Cloud Storage has been successfully carried out, and testing shows that the Smart Farm application has been well designed, offering the potential for overall performance improvement. Despite the success, the research also identified some limitations that must be addressed. Application scalability can be a challenge if the number of users increases significantly. Data security and privacy, especially farmer data and plant images stored on Cloud Storage, as the buckets on Cloud Storage are public, thus requiring further attention. The use of APIs in this research is limited to the login process, user account registration, and image uploading, creating room for further development in future research. Comparing this research with previous research shows progress in endpoint design and cloud computing integration. JSON Web Token (JWT) for user authentication reflects a more enhanced security approach than traditional methods. The success of this research also positively contributes to supporting sustainable agriculture through the utilization of API technology and cloud computing and provides a basis for further development.

CONCLUSION

Based on the research results, implementing the App Engine and Cloud Storage on the Smart Farm Application can be designed and implemented correctly. The stages passed in this research include problem identification, data collection and literature study, design, implementation, and testing with the black box method. The use of APIs in this research allows the Smart Farm Application to exchange data with the database and store images on Cloud Storage so that user account data that register and upload images to the Smart Farm Application can run properly. As for the test results with the black box testing method, which shows that all functions of each API endpoint run according to the expected output, through the HTTP method, all endpoints can run well, starting from the user registering, the user logging in, and the user uploading the image with the POST method.

REFERENCES

Achyar, M. K. U., Zulhelmi, M. R., Sumanjayanti, R., Jatri, R. A. M., Sujarwo, A., & Fudholi, D. H. (2020). Aplikasi Pintar Bertani (SIPITA) Sebagai Solusi Efektif Mendapatkan Hasil Bertani yang Maksimal. *Jurnal Teknik Elektro, FT UGM*, 21–26.

Ariantara, I. G. M., Arwani, I., & Putra, W. H. N. (2020). Penerapan REST API dalam Pengembangan Aplikasi Pemesanan RentalMobil berbasis Web dan Mobile (Studi Kasus: CV. Dwi Cipta Rent Car). *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *4*(8), 2569–2576.

Bhaswara, F. A., Sarno, R., & Sunaryono, D. (2017). Perbandingan Kemampuan Database NoSQL dan SQL dalam Kasus ERP Retail. *Jurnal Teknik ITS*, *6*(2), 510–514. https://doi.org/10.12962/j23373539.v6i2.24031

Choirudin, R., & Adil, A. (2019). Implementasi Rest Api Web Service dalam Membangun Aplikasi Multiplatform untuk Usaha Jasa. *MATRIK : Jurnal Manajemen, Teknik Informatika Dan Rekayasa Komputer*, *18*(2), 284–293. https://doi.org/10.30812/matrik.v18i2.407

Cloud, G. (2023). *Understanding Data and File Storage | App Engine standard environment for Java 8*.

Farizy, S., & Eriana, E. S. (2011). *Cloud Computing Komputasi Awan* (Issue 1).

Firdaus, A., & Ramadhan, D. A. (2021). Pengembangan Back End Berbasis REST API pada Sistem E-Partisipasi dan E-Inisiatif Patriot Pangan. *Jurnal Ilmu Komputer Dan Agri-Informatika*, *8*(1), 1–9. https://doi.org/10.29244/jika.8.1.1-9

Guntara, R. G., & Azkarin, V. (2023). Implementasi dan Pengujian REST API Sistem Reservasi Ruang Rapat dengan Metode Black Box Testing. *Jurnal Minfo Polgan*, *12*(1), 1229–1238. https://doi.org/10.33395/jmp.v12i1.12691

Hanifah, U., Alit, R., & Sugiarto. (2012). Penggunaan Metode Black Box Pada Pengujian Sistem Informasi Surat Keluar Masuk. *E-Journal UPN "Veteran" Jatim (Universitas Pembangunan Nasional)*, *XI*.

* Corresponding author

e-ISSN : 2541-2019
p-ISSN : 2541-044X

Harison, Putri, M., & Daratul, W. (2017). Perancangan Aplikasi Bercocok Tanam Padi dan Cabe Kriting Berbasis Android. *Jurnal Nasional Teknologi Dan Sistem Informasi*, *3*(2), 306–312. https://doi.org/10.25077/teknosi.v3i2.2017.306-312

Kurniawan, I., Humaira, & Rozi, F. (2020). REST API Menggunakan NodeJS pada Aplikasi Transaksi Jasa Elektronik Berbasis Android. *JITSI : Jurnal Ilmiah Teknologi Sistem Informasi*, *1*(4), 127–132. https://doi.org/10.30630/jitsi.1.4.18

Nugroho, A., & Mustofa, T. K. (n.d.). *Implementasi Komputasi Awan Menggunakan Teknologi Google App Engine (GAE) Dan Amazon Web Services (AWS)*.

Nurfauzia, A. R. (2021). *Implementasi Restful Web Service Pada Sistem Informasi Perpustakaan Berbasis Android*. http://repositori.unsil.ac.id/4343/

Perkasa, M. I., & Setiawan, E. B. (2018). Pembangunan Web Service Data Masyarakat Menggunakan REST API dengan Access Token. *Jurnal ULTIMA Computing*, *10*(1), 19–26. https://doi.org/10.31937/sk.v10i1.838

Sulistiani, W., & Sulistyo, W. (2020). *Implementasi Web Service dengan Metode REST Berbasis Golang pada Layanan Google Cloud Platform* (Issue 672015229).

Susanti, E., & Mailoa, E. (2020). RESTful API Implementation in Making a Master Data Planogram Using the Flask Framework (Case Study: PT Sumber Alfaria Trijaya, Tbk). *Journal of Information Technology and Computer Science*, *5*(3), 255–269. https://doi.org/10.25126/jitecs.202053189

Suzanti, I. O., Fitriani, N., Jauhari, A., & Khozaimi, A. (2020). REST API Implementation on Android Based Monitoring Application. *Journal of Physics: Conference Series*, *1569*(2). https://doi.org/10.1088/1742-6596/1569/2/022088

Triawan, A., & Prasetiyo, M. A. (2019). Penerapan Web Service (XML dan JSON) Untuk Meningkatkan Performance Pada Informasi Pembayaran Uang Kuliah. *Teknois : Jurnal Ilmiah Teknologi Informasi Dan Sains*, *8*(1), 78–93. https://doi.org/10.36350/jbs.v8i1.22

Warsito, A. B., Ananda, A., & Triyanjaya, D. (2017). Penerapan Data JSON Untuk Mendukung Pengembangan Aplikasi Pada Perguruan Tinggi Dengan Teknik Restfull Dan Web Service. *Technomedia Journal*, *2*(1), 26–36. https://doi.org/10.33050/tmj.v2i1.313

Wijaya, Y. D., & Astuti, M. W. (2021). Pengujian Blackbox Sistem Informasi Penilaian Kinerja Karyawan Pt Inka (Persero) Berbasis Equivalence Partitions Blackbox Testing of Pt Inka (Persero) Employee Performance Assessment Information System Based on Equivalence Partitions. *Jurnal Digital Teknologi Informasi*, *4*(1), 22–26. http://jurnal.um-palembang.ac.id/index.php/digital

Wiradiputra, M. R. D., Candiasa, I. M., & Divayana, D. G. H. (2021). Pengembangan dan Pengujian Sistem Informasi Manajemen Jalan Untuk Pemeliharaan Jalan Di Kabupaten Buleleng Menggunakan Standar Iso 9126. *Jurnal Ilmu Komputer Indonesia (JIK)*, *6*(1), 17–26.

Yusrizal, Dawood, R., & Roslidar. (2017). Rancangan Bangun Layanan Web (Web Service) untuk Aplikasi Rekam Medis Praktik Pribadi Dokter. *KITEKTRO: Jurnal Online Teknik Elektro*, *2*(1), 1–8. https://jurnal.usk.ac.id/kitektro/article/view/6803/5571

* Corresponding author