DOI: 10.22146/jnteti.V13i1.10095

Fog Computing-Based System for Decentralized Smart Parking System by Using Firebase

Haposan Yoga Pradika Napitupulu[1], I Gde Dharma Nugraha[1]

[1] Department of Electrical Engineering, Faculty of Engineering, Universitas Indonesia, Depok, Indonesia

[Received: 5 October 2023, Revised: 29 November 2023, Accepted: 8 December 2023]
Corresponding Author: Haposan Yoga Pradika Napitupulu (email: haposan.yoga@ui.ac.id)

ABSTRACT — The growth of vehicle number is unavoidable whilst the availability of parking is not directly proportional with this condition. Nowadays, many shopping centers do not have sufficient parking spot, causing customers to have difficulty in finding available parking spots. Research has been conducted to tackle the issue of finding available parking spots. Much of this research proposed the narrowband-Internet of things (NB-IoT) as a fog node. For communication purposes, this NB-IoT-based fog node has some shortcomings, such as security and privacy, lower data rate, higher cost in development, dependency with wireless system, and only covers one area. In this research, the fog computing was proposed to decentralize smart parking system by using Firebase to cover several areas or malls in one system and interface. Instead of using NB-IoT, this research employed decentralized local server as a fog node to deliver a fast data exchange. Firestore database (Firebase) was also used to secure, manage, and analyze the data in the cloud. Conjunctively, the Android application was created as a user interface to book and find the availability of parking spots. The Android application was built using Android Studio and implemented authentication to keep the data access secure and private. The testing scenario was done following the design unified modeling language (UML). The research results confirmed that the fog computing system successfully supported the decentralized smart parking system and was able to be implemented for covering several areas or malls in one system.

KEYWORDS — Smart Parking, Fog Node, Firebase, MySQL.

I. INTRODUCTION

The growth of vehicle number today is inevitable. Unfortunately, the increasing number of vehicles does not align with the availability of parking spots. Every day, vehicles are being manufactured and produced to meet the demand. On the other hand, the land for parking spots is limited and not able to be produced. For instance, there are many shopping complexes and malls in many big cities which do not have enough parking spots. This condition can cause a significant inconvenience for customers, particularly those who visit these malls for the first time. It can be frustrating since they must spend a considerable amount of time and navigate a wide area to find available parking spots. This condition benefits none of the customers as they must sacrifice their time to do unintended work and is detrimental to the malls since they can get bad ratings from customers. In addition to customers, the parking spot problem will also affect parking staff working at that mall.

Today, by utilizing technology, people can easily navigate through a mall and find the right parking spot. It will allow them to make decisions by leveraging information and save their priceless time. One of the most common types of information technology used today is cloud computing. It allows people process and store data in one place in the cloud. It eliminates the requirement of separate servers and enables them to work seamlessly. The goal is to ease people in finding available parking spots. It will reduce congestion and improve the customer experience. This technology not only displays the availability of parking spots, but also provides customers with information regarding the rates of the facility. This information will be displayed on the building's website and in the mobile applications.

Aside from cloud computing, there is a great combination and arrangement to address the request of huge amount of connection and low latency, namely fog computing nodes. The Internet of vehicle (IoV) has emerged as a new resource management system that can improve utilization of system resources and vehicle service quality [1], [2]. Another study has shown a promising function of smart parking [3]. The result showed that vehicular fog computing was able to deliver ultra-low latency packages to the clients and vehicles [3].

There are potential applications for fog computing node such as smart home surveillance which existing system used home gateway for smart parking application, delivery service, and smart retail, it is able to be replaced by using fog node and narrowband-Internet of things (NB-IoT) system [4]. The system proposed in [4] used NB-IoT system in which each sensor of NB-IoT had a unique ID for each parking spot. Each car must be installed with the NB-IoT tracking sensors. The NB-IoT at the parking spot would be in sleep mode unless a car with NB-IoT came to the parking spot. A study has demonstrated that the use of a centralized management in the infrastructure of vehicle-based fog computing enables a quick decision-making process [5]. It can be inferred from prior studies that fog computing is essential and has big potential to support decentralized smart parking systems to cover several malls or areas.

Previous studies have used NB-IoT for intelligent parking systems [6]–[10]. These result studies have shown that using NB-IoT for fog computing is a promising way to solve parking problems since NB-IoT is similar to fog computing in a way it reduces communication delay. Additionally, NB-IoT has a wide range of coverage and can cover vertically, allowing it to cover every corner of the parking lot. It also has a low power consumption, high efficiency, and is easy to operate. An intelligent parking system enhances people's convenience as it can resolve the problem of locating available parking spots.

Other studies have built a decentralized smart parking with fog computing [11], [12]. The proposed idea is similar to other research findings which used IoT by placing the sensor in a parking lot and installing the sensor in the car. Building a gateway as a decentralized system for communication of each sensor in the parking area is intended for exchanging data and collecting the sensor's data that are physically collected in the parking area. Then, the data are stored in the cloud. However, the use of NB-IoT has disadvantages regarding security and privacy and lower data rate. Whereas a study has emphasized that the confidentiality of vital data must be protected [13]. Developing smart parking are also costly due to the requirement of equipping each parking lot in every mall with NB-IoT modules. Despite its high cost, NB-IoT still relies on connectivity, hence, the operation will likewise go down when the wireless connectivity is down. Looking at previous research, most of them still focused, most of them focused solely on the development of smart parking for one coverage area only.

This research proposes a system based on fog computing for decentralized smart parking systems by using Firestore (Firebase) data system to cover several areas or malls in one system and interface. This system is designed to be used in parking lots located at malls. Currently, malls are equipped with sensors which can identify the availability of parking spots. The sensors collect the data of available parking spots, which are subsequently delivered to the local server as a fog node for faster update. Therefore, each mall will send the data of the parking lot to its local server as a decentralized system which keeps the data secure since it is stored into MySQL at the local server. The proposed system serves not only one mall, but several malls which are listed into application. Then, the fog node will be linked into the cloud. In this research, Firebase was used to collect all the data and to do cloud computing. This cloud is also used to manage data. Firebase offers authentication in reading and writing data which can secure the data. The reservation used an Android application as a user interface, and it was connected into Firebase. This method can secure the data since there are two layers of authentication to access the data. The first authentication is in the fog node, while the second is in the cloud. In addition, the update will be faster since the fog node is in one network along with the sensor. The structure of the paper is as follows: Section 2 discusses fog computing and smart parking, Section 3 provides design and implementation, Section 4 provides the result and analysis, and Section 5 provides conclusion.

II. FOG COMPUTING AND SMART PARKING

A. SMART PARKING

Smart parking is a type of parking system that uses various technologies to manage the garage efficiently. A lot of research has been done and many cities are starting and developing smart parking projects. This concept aims to help drivers utilize information and communication technology to efficiently find a satisfactory parking spot.

This system is also described as a smart system that aids drivers to know the presence or absence of a vehicle in a parking spot. This system will find an available parking spot by using sensors and it will lead the drivers to that available spot. In addition, smart parking involves the utilization of a system to locate available parking spots remotely. The IoT and fog computing aim to reduce the time required to find the vacant parking spot by enabling a platform. In addition to time reduction, the implementation of this system will also result in a reduction in fuel consumption and carbon dioxide (CO_2) emissions, which are adverse effects of vehicles in urban areas and unmanaged parking lots [14].

The present development of smart parking primarily relies on fog computing. It can also be referred to as vehicular fog computing (VFC). This finding is emerging and promising. The use of fog computing for transportation can provide lightweight and ubiquitous computing at the vehicle network edge [15].

B. FOG COMPUTING

Nowadays, the trend of the revolution of information is currently surging. Technologies such as artificial intelligence (AI), IoT, cloud computing, and big data are permeating all areas of the social economy and significantly impacting human lives. With the technology of cloud computing, large amounts of data are kept, analyzed, and managed in the cloud. The pressure on cloud data centers is also increasing due to the surge of data applications. Therefore, the search for new models of computing became an important research topic.

There is a layer between cloud layer and the end device namely "fog layer" which has been introduced by fog computing. The distributed computing model is used by fog computing. Fog nodes are distributed to the edge of end devices to perform data processing, computation, and communication, thereby effectively relieving the computing pressure of cloud data centers. Fog computing is a cloud computing advanced service to address and solve problems such as lack of mobility and latency [14], [16]. There are three layers which are mainly used in fog computing: cloud layer, fog layer, and user layer. Fog servers are part of fog layer placed around a physical building.

The framework of fog computing enables end-to-end storage of critical data and computing services on cloud servers by creating a fog layer with precise storage and capabilities of computing between cloud servers and adding them to application terminals. The fog layer is then moved to a fog server near the device. To filter and process the data uploaded by the user layer, fog computing is fortified with many communication devices, data storage and computing. Thus, it is effectively reducing the computing load and storage pressure of cloud servers.

C. FIRESTORE DATABASE

Firebase is a development platform for web and mobile applications. It offers a variety of services: data management, data analysis, authenticating process, web hosting, storage system, and database called Firestore database. Firestore database allows users to save and sync between applications in real-time because this database system is based on cloud-hosted NoSQL. It provides detailed documentation and cross platform software development kits (SDKs) to help build and ships apps on Android. Firebase can synchronize with a local server and utilizes JSON data for real-time synchronization.

The Firestore database is more flexible since the data are saved in the form of a document with an arrangement inside collection and subcollection. It is quite different compared to MySQL. In MySQL, the column or table's structure must be described. In the Firestore database, every document inside this cloud can have its own structure. Thus, it offers greater flexibility. However, it is suggested that the structure of the data in each document should be defined as the same as others so that it can make querying easier [17].

Not only is flexible, but the Firestore database is also expressive querying, allowing the data to be filtered and sorted in a group using an expressive and simple query interface. The user can easily update the data in real-time due to the Firestore database's capability to do this by using the powerful SDKs. In a contradictory way, the Firestore database is also able to offline support. The query data that are actively used will be cached in the device and they will be able to be written, read, and detected. Upon reconnecting the device to the internet, the data will be synchronized directly [17]. In addition, the Firestore database provides robust security and privacy as it is able to implement authentication and authorization access rules easily by integrating identity and access management (IAM), Web Client SDKs, and Firebase Authentication into the cloud [17].

The authentication system in the Firebase is easy to use and only needs minimal modification to it. Firebase Authentication integrates directly into Firebase Database. Furthermore, cloud computing enables the management and processing of data over long periods of time, as well as the capability to undertake comprehensive data analysis [18], [19]. Thus, Firebase Database facilitates control access to the data. Firebase also has built-in security at the data node level.

D. MySQL

MySQL is an open-source relational database management system (RDBMS) which is the most widely used [20]. It is used as software as a service (SaaS) over the cloud and described as a RDBMS. This database management system was released in 1995 and is currently supported by Oracle Corporation. As the most well-known database management system, it can support hosting service providers for instance WebHost Manager (WHM), Bluehost, GoDaddy, and Rackspot.

In addition, some renowned social media platforms are utilizing MySQL, such as YouTube and Facebook. Similarly, Amazon uses databases in MySQL in the cloud through hosting to provide service to its customers. Aside from having advantages, it also has limitations in terms of reliability, security, usability, and performance.

Some private organizations have database management systems for special purposes, despite there being many kinds of RDBMS which support the cloud in the market. However, research market study shows that MySQL is the most popular RDBMS in the cloud, with over 10 million installations.

MySQL also utilizes unencrypted connections between client and server by default [21]. This feature is essential for the safe and secure management of parking. It uses client name or IP address, username, and password. In the updated MySQL version 5.0 and higher even have the ability to check the complex string which is used as a password. However, MySQL has a vulnerability in regards to the security, including operation mode, padding way, and key derivation which is part of inner encryption principles [22]. MySQL has also been implemented in an integrated control system because MySQL is a useful database management system that will make documentation and reporting easier [23].

Therefore, not only did it implement MySQL, but this research also implemented the Firestore database to provide more layers of security. In addition, the real-time synchronization of MySQL to Firebase was connected using JsonNode, with the purposes of securing and protecting the data. The security of MySQL and Firestore database at the cloud layer can manage the parking database safer and more securely compared with other methods that only use NB-IoT.

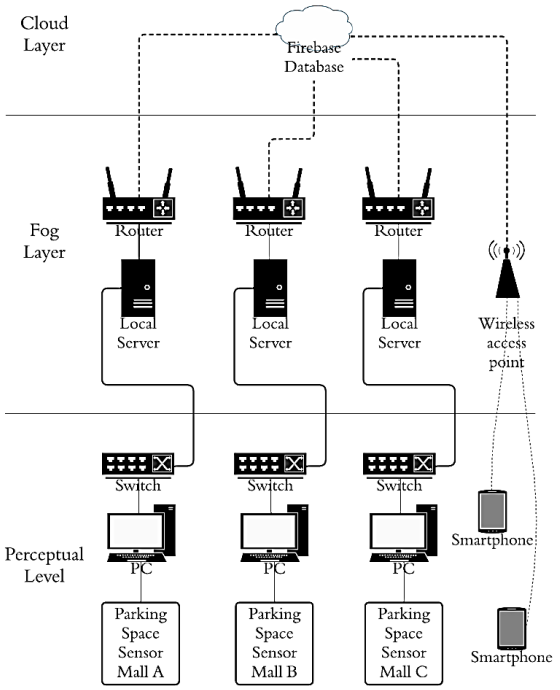


Figure 1. Architecture of fog computing for the built smart parking.

III. DESIGN AND IMPLEMENTATION

A. ARCHITECTURE

The built architecture system in this research is shown in Figure 1. The system was built based on decentralized fog computing, consisting of an existing parking system which in this research was assumed already exist. The data were stored into each local server with MySQL for each mall, meaning that every mall had a local server as a storage system based on MySQL. In this research, local servers were currently built in local computers with several databases as a representation of local servers at several malls. The fog node can reduce the latency of parking lot availability that will be updated in the cloud. Each local server was connected into the cloud by using Firebase Database. In the cloud layer of Firestore database, all the data were secured using authentication to prevent unauthorized access. All the data also can be managed and analyzed in the cloud layer, so it will make it easy to get data insight for future purposes.

The connection from the local server into Firestore database was implemented using Node.js. The built Android application, namely smart parking application, was then connected into the cloud as the user interface that allowed users to interact with the system and to obtain some information such as availability of parking spot, selection of mall, selection of parking lot, and reservations for desired parking spots. Then, the connection from the built smart parking application to the cloud layer of Firestore database was secured by implementing the authentication for the rule of reading and writing. In addition to authentication for reading and writing, authentication was implemented for user registration and login attempts.

B. SYSTEM DESIGN

The system design was built and separated into three different universal multi language (UML). It consists of flowchart of user, pseudocode of system, and class diagram, as shown in Figure 2, Figure 3, and Figure 4, respectively.

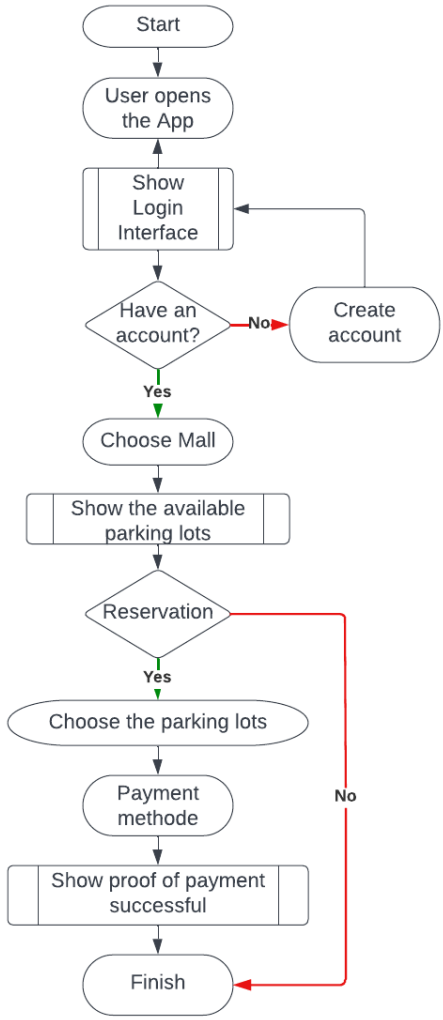


Figure 2. Flowchart of universal multi language (UML) system user.

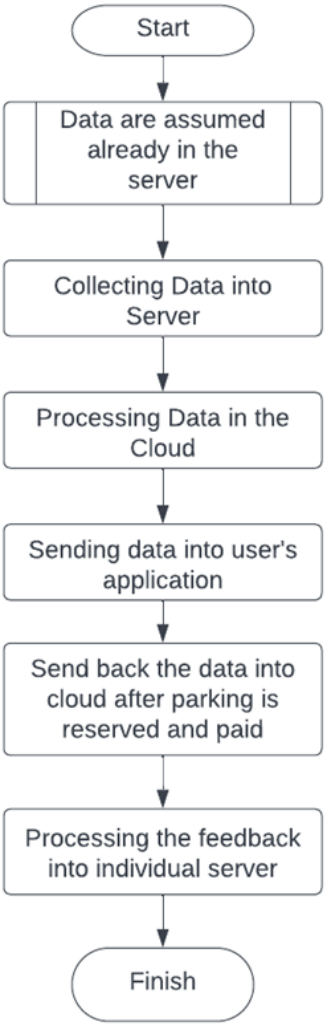


Figure 3. Flowchart of universal multi language (UML) system user.

Figure 2 shows a flowchart diagram at the user side. In the first stage, the user opened the smart parking application that had also been made in this research using Android Studio. After opening the smart parking application, a new page appeared, it showed the user login interface. At this stage, the user must input the registered name along with the password. If the user had not had a username yet, the user must be registered first, and they would be brought to the new page of registration interface. After completing the registration in registration interface, the smart parking app would bring the user into login interface again and it would allow the user to input and use the registered username and password. After successfully logging in, a new page would show a list of several malls. Here, the user must select a mall that they would visit. After selecting the mall, the smart parking application would provide a list of parking spots in the chosen mall along with the availability of those parking spots. The user could continue by choosing a parking spot or just looking at it. After selecting a parking spot and user would like to book an available parking spot, the smart parking application would lead the user into the payment page. Here, there were several payment methods which the user could use. After the user made the payment, a new page would show "payment successful" notification which notified the user that the booking of chosen parking spot had been already paid.

Figure 3 shows an activity diagram of the process in this smart parking system. The required data were assumed to already exist on the local server where the data came from an installed parking sensor on site. The data coming from the parking sensors were received and stored into MySQL on the local server. Furthermore, the data from the local server were processed to the cloud layer which was the Firestore database. Subsequently, the data were sent to the user smart parking application. If there were any data changes which could come from someone making a reservation and payment, the data were reverted to the cloud in order to update the availability of parking spot which had already been booked by the user. Then the cloud would send the instruction to the individual local server and notify administrators.

Figure 4 shows the relationship of the activity among the things, which is also called as a class diagram. Customers, as the user of the smart parking application, had a relationship with the mall. The consumers had the option to select a specific mall to view the availability of parking spots, and the mall offered the corresponding parking lot data. Customers also had relationship with Event Booking. This Event Booking would have a connection with required items, which the required item for this activity was the availability of parking spots that came from the individual local server from each mall to be executed during the booking and payment process.

C. GRAPHICAL USER INTERFACE (GUI)

This research also developed and built an Android application namely smart parking application. This application is a part of this research and has a function for interacting

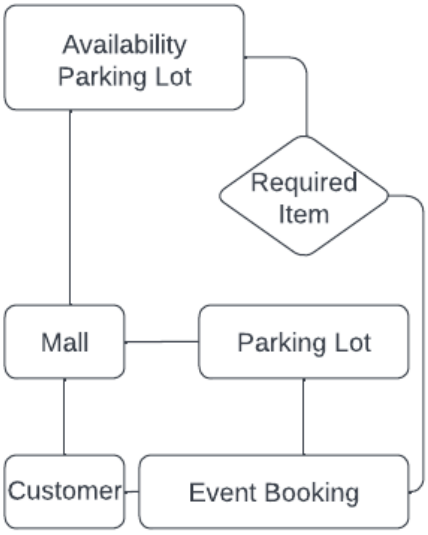


Figure 4. Universal multi language (UML) system class diagram.

between user and the system. The graphical user interface (GUI) in this research was built by using Android Studio, so the smart parking application could run on smart phone with Android operating system.

Figure 5(a) shows the registration page of the smart parking application. The user which has not registered, must register from this page. In this page, the user must input some required data such as email, password, user full name, license plate, vehicle type, and password confirmation. Password confirmation is done by retyping the password. It is essential in case the user unconsciously mistyped their password.

Figure 5(b) shows the list of the mall. In this page, the smart parking application showed every mall along with the amount of available parking spots. It is intended to allow users to know the availability of parking spots in each mall, so they can decide their destination and put the availability of parking spots as a part of their consideration. Here, the user can choose the preferred mall and press the "Select" button for further processing. The further processing is to select the parking spot.

Aside from two presented GUI in Figure 5, there are several GUI pages for supporting this smart parking system which was already built in this Android smart parking application using Android Studio.

1) HOMEPAGE DISPLAY OF THE APPLICATION

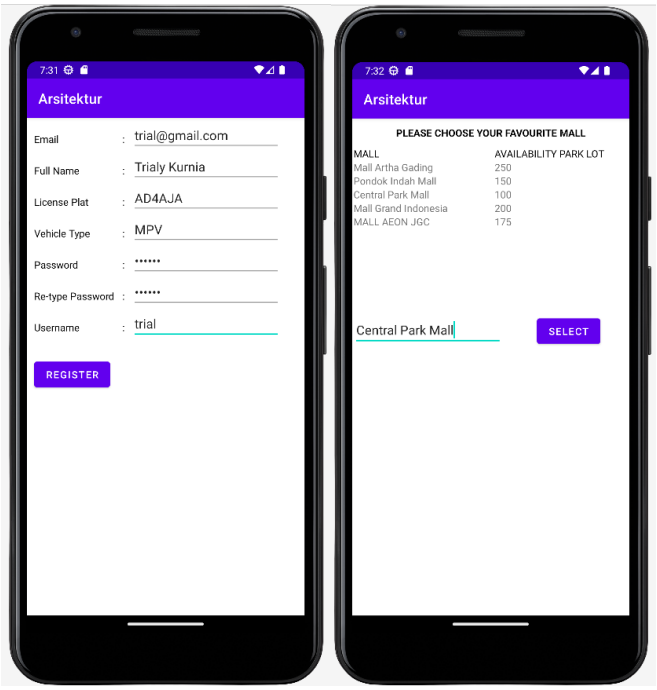
The homepage display includes columns for the username, password, login button, and registration button. This page is designed for the purpose of logging in. Once the user has an account, they can directly input the username and password then click the "Login" button. Although the user does not currently have an account, they can click on the register button, which will redirect them to the new registration page.

2) LIST MALL PAGE

The page consists of malls that have been registered into the smart parking system. Hence, this page shows several malls along with the number of available parking spot at each listed mall.

3) CHOOSE THE PARKING LOT PAGE

In this page, the user can choose an available parking spot. Each parking spot is provided with information regarding the type of vehicles that can be parked there, since several malls have parking spots specifically designed for sedans, due to the



(a) (b)

Figure 5. Graphical user interface (GUI) of the smart parking system, (a) registration page, (b) list of malls page.

lower ceilings. Therefore, choosing a preferred parking spot, the users must see the information of that parking spot.

4) BOOKING CONFIRMATION PAGE

In this page, the smart parking application shows the payment process, so the user can decide which payment method they want to use. In this page, there is a "Confirm" button and a "Cancel" button. The "Confirm" button functions to confirm the order, while the "Cancel" button functions to cancel the order and leads the user to the previous page of parking spot selection.

5) SUCCESSFUL PAGE

This page shows if the booking is successful. After successfully booking, the application sends feedback of updated data into the Firestore database cloud. The MySQL local server is updated subsequently.

D. RELATIONAL DATABASE MANAGEMENT SYSTEM (RDBMS)

RDBMS is based on the relational model, which is why it is called relational database management system. The most commonly used database is this system. It comprises numerous tables. Every table in RDBMS must implement a primary key. Data can be accessed easily in RDBMS since it consists of a collection of an organized set of tables. Everything in RDBMS is saved in the form of relations.

RDBMS was implemented for MySQL and stored into each local server of each mall for the availability of parking spot data. On the other hand, the user data, and its history of parking, were stored into a different database which was at the provider of the smart parking application. Tables were used by RDBMS to store data. A table is a collection of connected data elements that uses rows and columns to hold information. Each table represents a real-world entity, such as a person, location, or event, about which data are gathered. The logical perspective of the database is the orderly collection of data into a relational table.

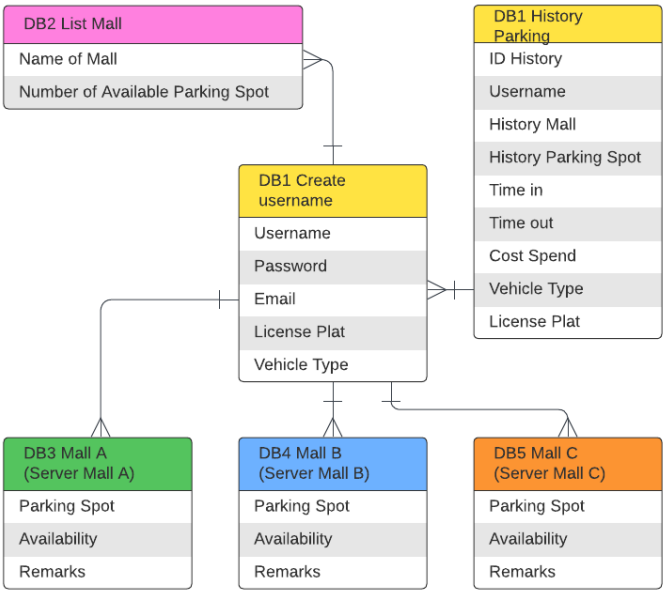


Figure 6. RDBMS of this research.

Figure 6 shows the RDBMS of this smart parking research project. This RDBMS shows the relationship between databases. One of the main concerns addressed in this research is the development of a smart parking system that can be implemented in many coverage areas. This smart parking system is designed to be connected to various local servers in different malls. This study utilized a total of six databases.

1) USER DATABASE (DB1 CREATE USERNAME)

This database contained users' data which was saved during registration. It consisted of username, password, email, license plate, and vehicle type. This database was stored into the Firestore database.

2) MALL DATABASE (DB2 LIST MALL)

This database contained registered malls along with the number of their available parking spots. Mall database was also stored into the Firestore database. The number of available parking spots was updated from MySQL local server from the mall.

3) PARKING DATABASE OF MALL A (DB3 MALL A)

This database contained the list of identity of parking spots along with their availability and remarks at mall A. The remarks indicated whether the parking spots were suitable for parking all types of vehicles or specific types of vehicles. This database was stored on the mall local server. Then, it was passed through the Firestore database.

4) PARKING DATABASE OF MALL B (DB4 MALL B)

This database contained the list of identity of parking spots along with their availability and remarks at mall B. The remarks indicated whether the parking spots were suitable for parking all types of vehicles or specific types of vehicles. This database was stored on the mall local server. Then, it was passed through the Firestore database.

5) PARKING DATABASE OF MALL C (DB5 MALL C)

This database contained the list of identity of parking spots along with their availability and remarks at mall C. The remarks indicated whether the parking spots were suitable for parking all types of vehicles or specific types of vehicles. This database was stored on the mall local server. Then, it was passed through the Firestore database.

6) DATABASE OF PARKING HISTORY (DB1 PARKING HISTORY)

This database contained the history of parking of the user. It consisted of the username, the history of the mall, the history of the parking spot, the time of entry, the time of exit, cost spent, vehicle type, and the license plate. This history parking was used for business analytics purposes. This database was stored in the Firestore database since the Firestore database has an analytics function which can support this purpose.

E. IMPLEMENTING AUTHENTICATION IN FIREBASE

One of the issues addressed in this research is security and privacy concern. This fog computing-based decentralized smart parking system utilizing Firestore database (Firebase) implemented an authentication system based on a blocking function that allowed and blocked users when registered into the application. It also allowed and blocked users when signed into the application. Below is the code of this authentication.

```
  const functions = require('firebase-
functions');

exports.beforeCreate =
functions.auth.user().beforeCreate((user,
context) => {
});

exports.beforeSignIn =
functions.auth.user().beforeSignIn((user,
context) => {
});
```

In addition to implementing authentication for registration and login, this research also incorporated the utilization of IP address tracking to monitor any suspicious activity within the smart parking system. This approach monitors the users' IP address during the login process and compares it with the IP address of subsequent requests. The code is implemented as shown below.

```
  exports.beforeSignIn =
functions.auth.user().beforeSignIn((user,
context) => {
  return {
    sessionClaims: {
      signInIpAddress: context.ipAddress,
    },
  };
});
```

In the condition if the requestor attempted to access resources which authentication required with Firebase Authentication, the smart parking system compared the requestor IP address with the IP used to try to login. Below is the code of this method.

```
  app.post('/getRestrictedData', (req, res) =>
{

  // Obtain the past ID token.
  const idToken = req.body.idToken;

  // ID token verification, if revoked then
check and its payload will be decoded.
  admin.auth().verifyIdToken(idToken,
true).then((claims) => {

    // IP address during requesting
    const requestIpAddress =
req.connection.remoteAddress;
```

```
    // IP address when signing-in
    const signInIpAddress =
claims.signInIpAddress;

    // Determine whether the request IP
address origin is suspicious in comparison to
the session IP addresses. The current request
timestamp and the auth_time of the ID token
can provide additional indications of abuse,
particularly if the IP address changes
abruptly. If there is a sudden geographical
shift in a short period of time, it will give
stronger indications of suspected abuse.
    if
(!isSuspiciousIpAddressChange(signInIpAddress,
requestIpAddress)) {

      // Unusual IP address change. Re-
authentication is required. You can also call
admin.auth(.revokeRefreshTokens(claims.sub) to
revoke all user sessions.
      res.status(401).send({error:
'Unauthorized access. Please login again!'});
    } else {

      // Invalid Access. Attempt to return
data.
      getData(claims).then(data => {
        res.end(JSON.stringify(data);
      }, error => {
        res.status(500).send({ error: 'Server
error!' })
      });
    }
  });
});
```

IV. RESULTS AND DISCUSSION

A. DATABASE IN MySQL

This research used MySQL as the platform to create a database. The database was divided into three databases: parking_mall_a, parking_mall_b, and parking_mall_c. In the parking_mall_a, parking_mall_b, and parking_mall_c databases, there is only one table containing parking_spot, availability, and remarks as shown in Table I. In the parking_spot column, there was an identification for every parking spot in the mall. Meanwhile, the availability column indicated the availability of parking spots, whether they are available.

Last, column remarks showed the information of parking spots, whether they could be used for certain vehicles or all types of vehicles. Remark “none” means that parking spots can be used for all types of vehicles. In this research, it was assumed that the data had been inputted into the server.

B. FIREBASE

Firebase is a backend as a service (BaaS) offered by Google to make it easier for application developers to develop an application (web and mobile). In this smart parking research project, Firebase was used as a cloud. The necessary data from fog nodes were stored into the cloud and the cloud also had capability to authenticate the activity. In this Firestore database, data can be managed, secured, and analyzed.

Figure 7 presents all databases in Firebase such as db_customer, db_customer containing username, password, and other things that were previously registered into the application. The detailed contents of db_customer consist of name, license_plat, password, username, and vehicle_type.

TABLE I
DATABASE OF PARKING LOT OF MALL A

parking_spot	Availability	Remarks
LotA1	TRUE	sedan
LotA2	TRUE	none
LotA3	FALSE	none
LotA4	TRUE	none
LotA5	FALSE	sedan

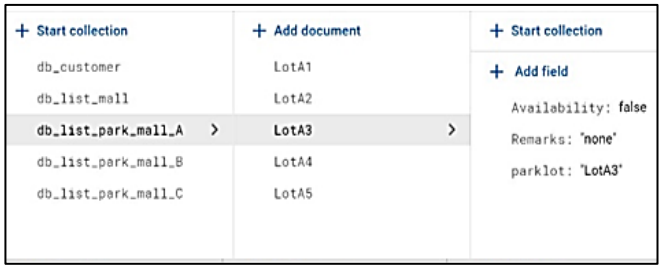


Figure 7. All databases which are listed in the Firebase and mall’s parking spot database.

Figure 7 also depicts a database of parking spots in mall A. It was assumed that parking spots were already present on the server. There were five parking spots in mall A. Each parking spot contained information on its availability, name, and information on whether it could b e used for all types of vehicles or only certain vehicles. For example, the remarks in LotA3 indicate “none”, suggesting that the parking spots are suitable for all types of vehicles. Conversely, remarks in the LotA5 parking spot say “sedan”, indicating that this spot is only designated for sedans and not other vehicles. These arrangements are also implemented into the database of parking spots in mall B and mall C.

C. TESTING

After everything was built and compiled together, starting from creating a database on the local server, connecting the database on the Firestore database, and connecting the Android Application, then testing was carried out following the UML design. The results of the testing are shown in Table II.

Table II shows the testing results. There were several test scenarios that represented how the system worked based on the UML, starting with a test scenario of users registered as new users and ending up with connecting Firebase Cloud to the app. The results of all scenarios showed PASS, suggesting that the system works well.

This research proposed fog computing in decentralized smart parking systems by using Firebase to cover several areas or malls in one system and interface. It means that the system can accommodate multiple parking lots from several malls into one system along with security and privacy by implementing authentication in Firebase. The system also offers a higher data rate since it is connected to internet network infrastructure that allows people to see and book the available parking spots in advance at their desired malls. This feature is beneficial for them since they do not need to search for parking spots upon their arrival. However, there are disadvantages associated with this parking system, particularly when the internet connection is down, the parking system is not able to be used. However, to address this issue, the manual system can serve as a viable alternative.

TABLE II
TESTING DOCUMENTATION

Test Scenario	Test Case	Test Step	Result	Status
Users register as new user.	Registering new user.	Input all data, press button register, ensure data registered into firebase.	Data stored into firebase.	PASS
Check log in.	Checking validity log in.	Input right and wrong data.	If result right, then will go to next GUI. If wrong, it will tell that username or password is not recognised.	PASS
Users open the app.	User open app and click available button.	1. user open the app.	It will show the app.	PASS
User open mall option.	User open mall option with click choose mall option.	1. User choose the mall.	User is on the selected mall.	PASS
User open parking spot option.	User open parking mall option at the selected mall.	1. Users choose the mall. 2. Users choose the parking spot at the selected mall.	User can choose parking spot.	PASS
Load data from database.	Load database from firebase so we can see the parking availability.	Users open the parking spot and will show the parking availability.	User can book parking spot.	PASS
User open payment method.	User open payment method.	User open payment method after choosing the parking spot.	User can process the payment method and will show "payment success."	PASS
Connecting Local MySQL into Google Cloud.	Test reading the database, table, and data via Google Cloud.	Open terminal in Google Cloud.	Google Cloud can read the data from Local MySQL.	PASS
Connecting Firebase Cloud to the App.	Test reading and writing into database Firebase Cloud.	Test on App and Firebase Cloud.	App can write and read data from Firebase.	PASS

V. CONCLUSION

The smart parking system commonly uses NB-IoT, and it has already been proposed by some researchers. However, it has some disadvantages such as the matter of security and privacy, lower data rate, higher cost in the development, dependencies with wireless systems, and only covers one area. This research proposes fog computing based on a decentralized smart parking system using Firebase to cover several different areas or malls. This smart parking research project used a decentralized local server as a fog node to deliver a fast data exchange. Aside from that, this smart parking research used the Firestore database (Firebase) to authenticate, secure, manage, and analyze data in the cloud. Whilst the smart parking Android application was used as a GUI to interact with users, show the information, find the parking spot's availability, and book the available parking spot. This smart parking application was built using Android Studio. The result of this research demonstrates that the built and proposed system works well and can be implemented for covering several areas or malls in one system. For future works, it is suggested to implement and connect the system directly to the malls' servers, test execution times, and conduct comparative analysis. In addition, it is essential to determine an economic feasibility study for the real implementation, given that the local servers are built into local computers using several databases in this research.

CONFLICTS OF INTEREST

During conducting, designing, testing, and providing the outcome of this research, the authors state that there is no conflict of interest.

AUTHORS' CONTRIBUTIONS

Conceptualization, Haposan Yoga Pradika Napitupulu; methodology, Haposan Yoga Pradika Napitupulu; software, Haposan Yoga Pradika Napitupulu; validation, Haposan Yoga Pradika Napitupulu; analysis, Haposan Yoga Pradika Napitupulu. writing, Haposan Yoga Pradika Napitupulu; visualization, Haposan Yoga Pradika Napitupulu; supervision, I Gde Dharma Nugraha.

REFERENCES

[1] A.J. Kadhim and J.I. Naser, "Proactive load balancing mechanism for fog computing supported by parked vehicles in IoV-SDN," *China Commun.*, vol. 18, no. 2, pp. 271–289, Feb. 2021, doi: 10.23919/JCC.2021.02.019.

[2] M. Kong, J. Zhao, X. Sun, and Y. Nie, "Secure and efficient computing resource management in blockchain-based vehicular fog computing," *China Commun.*, vol. 18, no. 4, pp. 115–125, Apr. 2021, doi: 10.23919/JCC.2021.04.009.

[3] Y. Zhang, C.-Y. Wang, and H.-Y. Wei, "Parking reservation auction for parked vehicle assistance in vehicular fog computing," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3126–3139, Apr. 2019, doi: 10.1109/TVT.2019.2899887.

[4] C. Tang *et al*., "Towards smart parking based on fog computing," *IEEE Access*, vol. 6, pp. 70172–70185, Nov. 2018, doi: 10.1109/ACCESS.2018.2880972.

[5] F.H. Rahman *et al*., "Street parked vehicles based vehicular fog computing: TCP throughput evaluation and future research direction," *2019 21st Int. Conf. Adv. Commun. Technol. (ICACT)*, 2019, pp. 26–31, doi: 10.23919/ICACT.2019.8701912.

[6] X. Lin *et al*., "Application research of NB-IoT technology based on fog computing in intelligent parking system," *2019 IEEE 3rd Adv. Inf. Manag. Commun. Electron. Automat. Control Conf. (IMCEC)*, 2019, pp. 1496–1503, doi: 10.1109/IMCEC46724.2019.8984053.

[7] Y.-C.P. Chang, S. Chen, T.-J. Wang, and Y. Lee, "Fog computing node system software architecture and potential applications for NB-IoT industry," *2016 Int. Comput. Symp. (ICS)*, 2016, pp. 727–730, doi: 10.1109/ICS.2016.0150.

[8] M.A. Hoque and R. Hasan, "Towards an analysis of the architecture, security, and privacy issues in vehicular fog computing," *2019 SoutheastCon*, 2019, pp. 1–8, doi: 10.1109/SoutheastCon42311.2019.9020476.

[9] S. Nguyen, Z. Salcic, and X. Zhang, "Big data processing in fog - smart parking case study," *2018 IEEE Int. Conf. Parallel Distrib. Process. Appl.*

Ubiquitous Comput. Commun. Big Data Cloud Comput. Soc. Comput. Netw. Sustain. Comput. Commun. (ISPA/IUCC/BDCloud/SocialCom/SustainCom), 2018, pp. 127–134, doi: 10.1109/BDCloud.2018.00031.

[10] B. Cheng, J. Fuerst, G. Solmaz, and T. Sanada, "Fog Function: Serverless fog computing for data intensive IoT services," *2019 IEEE Int. Conf. Serv. Comput. (SCC)*, 2019, pp. 28–35, doi: 10.1109/SCC.2019.00018.

[11] E.C. Anderson, K.C. Okafor, O. Nkwachukwu and D.O. Dike, "Real time car parking system: A novel taxonomy for integrated vehicular computing," *2017 Int. Conf. Comput. Netw. Inform. (ICCNI)*, 2017, pp. 1–9, doi: 10.1109/ICCNI.2017.8123788.

[12] M. Celaya-Echarri *et al*., "Building decentralized fog computing-based smart parking systems: From deterministic propagation modeling to practical deployment," *IEEE Access*, vol. 8, pp. 117666–117688, Jun. 2020, doi: 10.1109/ACCESS.2020.3004745.

[13] J.-E. Park and Y.-H. Park, "Fog-based file sharing for secure and efficient file management in personal area network with heterogeneous wearable devices," *J. Commun. Netw.*, vol. 20, no. 3, pp. 279–290, Jun. 2018, doi: 10.1109/JCN.2018.000040.

[14] A.M.S. Maharjan and A. Elchouemi, "Smart parking utilizing IoT embedding fog computing based on smart parking architecture," *2020 5th Int. Conf. Innov. Technol. Intell. Syst. Ind. Appl. (CITISIA)*, 2020, pp. 1–9, doi: 10.1109/CITISIA50690.2020.9371848.

[15] X. Huang, D. Ye, R. Yu, and L. Shu, "Securing parked vehicle assisted fog computing with blockchain and optimal smart contract design," *IEEE/CAA J. Automatica Sinica*, vol. 7, no. 2, pp. 426–441, Mar. 2020, doi: 10.1109/JAS.2020.1003039.

[16] A.M. Farooqi, M.A. Alam, S.I. Hassan, and L. Ansari, "Approaches of shared smart parking model in fog and roadside cloud environment: A detailed survey," *2021 6th Int. Conf. Commun. Electron. Syst. (ICCES)*, 2021, pp. 914–924, doi: 10.1109/ICCES51350.2021.9489178.

[17] Y. Sukmana and Y. Rosmansyah, "The use of Cloud Firestore for handling real-time data updates: An empirical study of gamified online quiz," *2021 2nd Int. Conf. Electron. Commun. Inf. Technol. (CECIT)*, 2021, pp. 1239–1244, doi: 10.1109/CECIT53797.2021.00220.

[18] K.S. Awaisi *et al*., "Towards a fog enabled efficient car parking architecture," *IEEE Access*, vol. 7, pp. 159100–159111, Nov. 2019, doi: 10.1109/ACCESS.2019.2950950.

[19] J. Tuvakov and K. Park, "On the fog node model for multi-purpose fog computing systems," *2018 IEEE 9th Annu. Inf. Technol. Electron. Mob. Commun. Conf. (IEMCON)*, 2018, pp. 1211–1214, doi: 10.1109/IEMCON.2018.8614845.

[20] H.-L. Shieh, W.-S. Chang, S.-F. Lin, and S.-B. Jhang, "A motorcycle parking lot management system based on RFID," *2013 Int. Conf. Fuzzy Theory Appl. (iFUZZY)*, 2013, pp. 268–272, doi: 10.1109/iFuzzy.2013.6825448.

[21] I. Zoratti, "MySQL security best practices," *2006 IET Conf. Crime Secur.*, 2006, pp. 183–198.

[22] L. Zhang, J. Fan, and Y. Zhou, "The security analysis of MySQL's encryption functions," *2015 Int. Conf. Comput. Sci. Mech. Automat. (CSMA)*, 2015, pp. 5–8, doi: 10.1109/CSMA.2015.8.

[23] H.Y.P. Napitupulu, "Design and realization of integrated control system based on Microsoft Visual Basic .Net and Mitsubishi's programmable logic controller (PLC) through ethernet cable," *2023 Int. Conf. Comput. Sci. Inf. Technol. Eng. (ICCoSITE)*, 2023, pp. 290–295, doi: 10.1109/ICCoSITE57641.2023.10127819.