**Jurnal Teknik Informatika (JUTIF)**
Vol. 5, No. 3, June 2024, pp. 799-810

DOI: https://doi.org/10.52436/1.jutif.2024.5.3.2002
p-ISSN: 2723-3863
e-ISSN: 2723-3871

# *PROTOTYPE OF CONTACTLESS PAYMENT SYSTEM WITH RFID AND BLOCKCHAIN TECHNOLOGY INTEGRATED WITH MOBILE APPLICATION*

**Danny Jiustian[1], Alfa Yohannis[*2]**

[1]Departement of Informatics, Faculty of Science and Technology, Pradita University, Tangerang, Indonesia
[2]Department of Computer Science, University of York, York, United Kingdom
Email: [1]danny.jiustian@student.pradita.ac.id, [2]alfa.ryano@gmail.com

(Article received: April 23, 2024; Revision: May 14, 2024; published: June 02, 2024)

***Abstract***

*The COVID-19 pandemic has led to a change in payment methods, with a shift towards cashless payments to avoid germs and viruses. Contactless payments, especially those using RFID technology, have become popular due to their convenience and no need to enter security codes. However, there is a risk of data manipulation in transaction records, leading to decreased trust and transparency. To solve this problem, this research develops an RFID contactless payment system connected with blockchain technology as the main goal of the research. Blockchain is known for its security and transparency, making it suitable for minimizing data manipulation that often occurs. This research will use the Sepolia Test Network of the Ethereum base network for development in terms of blockchain to serve as a security layer in this research. The Waterfall method will be used for application development, focusing on structured and linear stages such as requirements analysis, system design, implementation, testing, and maintenance. The application development process has shown positive results, with successful black box testing and the ability to track and validate transactions stored in the database and blockchain. This validation process is critical to ensure the integrity of transactions and detect any data manipulation.*

**Keywords**: *Blockchain, Contactless Payment, Ethereum, Internet of Things, RFID, Smart Contract.*

## 1. INTRODUCTION

Payment is a transaction process that will involve the exchange of financial value between two or more parties in return for goods, services, or obligations that have been provided or performed. The main purpose of payments is to facilitate economic exchange, ensure value for the recipient, and legally perfect the transaction [1].

The impact of the COVID-19 pandemic has made contactless payments increasingly popular, where payments can be made without touch. This change in consumer behavior can create new innovations. Contactless payment systems also help reduce direct contact between users and payment machines, avoiding the potential transmission of germs or viruses [2].

The payment system is inseparable from the recording of transactions connected to the computer system [1]. With the records connected to the system, it will not be separated from the problem of transaction data security. The development of the world now manipulates data or data falsification can not only be done with data written on paper media, but also with digital media such as data stored on computers or on the internet [3].

In the transaction process, it is inseparable from transaction security issues. Data Forgery is one of the security problems that can occur in transactions, because with data forgery, the perpetrator can make changes to data or falsify data that has been stored on the internet as if there has been a data typing error [3].

With blockchain technology, previously created transaction data can be secured properly. Blockchain is a decentralized system that records every transaction transparently and cannot be changed by anyone because every transaction is validated by blocks in the system [4]. The use of blockchain not only ensures data security, but also avoids potential manipulation or fraud in the payment process. Therefore, blockchain systems provide users with an additional level of confidence in financial security.

This research will use some help such as using the ESP32 Microcontroller as a component that controls other electronic devices. Microcontroller consists of CPU, memory, and input/output devices [5]. ESP32 is a versatile microcontroller with low power consumption that provides integrated Wi-Fi and Bluetooth Low Energy (BLE) for IoT projects and wireless applications [6].

This research will be RFID cards as a means of payment of the system built, RFID uses radio wave frequencies to activate RFID tags. RFID tags usually have EEPROM to store data such as product ID, quantity, and location. RFID tags themselves have various types such as card-shaped which are often used in security, access, and transaction applications. There are several frequencies used in RFID, including

Low Frequency (LF), High Frequency (HF) and Ultra-High Frequency (UHF). Each type of RFID frequency has different characteristics and is suitable for certain applications depending on the range, speed, and operational environment requirements [7].

In this study, we will use MQTT (Message Queuing Telemetry Transport), MQTT is an Internet of Things network protocol for machine-to-machine (M2M) communication at low bandwidth, the basis of this protocol is TCP / IP which can send messages to each other between the server and client on a publish-subscribe basis so that the process inside the server will be light [8].

This research will use the Smart Contract security system of the Ethereum network as a container for storing important data that has been created previously. Smart contracts are the result of the development of Ethereum which is useful for enhancing the capabilities of the blockchain. Smart contracts are written in the solidity programming language which is a specially designed Turing-complete. This smart contract will carry out certain actions when predetermined criteria are met, this feature provides transparency of transaction data that is trustworthy and secure without intermediaries [9]. In the use of smart contracts, there is a main code that can verify and check smart contract interactions [10]. Each smart contract used will incur gas fees, gas fees are described as labor wages that measure computational effort for operations and transactions. The gas system in Ethereum itself has two purposes, namely to provide prepaid incentives to miners who execute code and maintain network security, even in the event of failed executions. Then to prevent operations from exceeding their assigned computation time and handle the halting problem [11].

Smart contracts created will be assisted with the help of Hardhat, Hardhat provides features to create smart contracts, Hardhat also has the necessary features such as editing, compiling, debugging, and deploying smart contracts locally. Hardhat also has a main component, Hardhat Runner, which is useful for executing smart contracts [10][12].

From previous research, this research aims to integrate a payment system that uses contactless RFID with blockchain technology, so that every contactless transaction can be stored and validated in the blockchain. The main goal is to increase the security and transparency of transaction data, by utilizing smart contracts on the Ethereum network to automatically validate transactions. With the use of the blockchain system, it can overcome security problems, because the blockchain system implements a system that is transparent about the data that has been stored.

## 2. RESEARCH METHODS

The Waterfall Method [13] has several stages in the application development process, the process will be up to system maintenance or maintenance. The waterfall method has a systematic and sequential approach to the software system development process. The waterfall method starts from analyzing user needs then continues with the planning, modeling and construction stages [13].

The waterfall method was introduced around 1970 by Winston Royce [14], this method is used until now especially for the software application development process. It is called waterfall because the next stage requires confirmation of completion at the previous stage.

The waterfall method is a System Development Life Cycle (SDLC) method that is used to develop systems or software [14]. The stages in this method will start from the design process to the maintenance process which is useful for keeping the developed system running properly.

The emphasis on the waterfall method emphasizes the structured and linear stages of system development. In this research, the greater emphasis on the waterfall method refers to the systematic approach in designing and developing the integration between contactless RFID payment systems and blockchain technology. This research will ensure that each stage of development, from the concept of integration to the implementation of the application, can be carried out carefully in a predetermined order. This process is important so that the integration of the system built between contactless RFID and blockchain can occur efficiently and without obstacles.

By emphasizing consistency with the waterfall methodology, this research will ensure that each step in the application development process can go according to plan, so that the results can be expected in accordance with the predetermined objectives.

This research will produce a Contactless Payment Prototype System with RFID and Blockchain Technology and Integrated with Mobile App. The design of this system is divided into two stages, namely the design of hardware devices then proceed with the design of software according to the needs of the hardware made. For the detailed process can be seen in the flowchart Figure 1.

In Figure 1 there are several stages that will be carried out, such as in the first stage, namely a literature study that seeks information related to previous research so that the system developed can be made optimally, the next stage is in the analysis of needs, at this stage will collect the components of the needs that will be used in the system. The next stage is to prepare tools that will help make the system. The next stage is database design which is a crucial stage because all data will be stored in the database, therefore a good design is needed. Next, namely designing or building hardware and software systems

which is a continuation stage so that the system built can run. In the last stage, system maintenance will be carried out, this system maintenance aims to ensure that the system that has been built can run continuously and avoid crucial system problems.

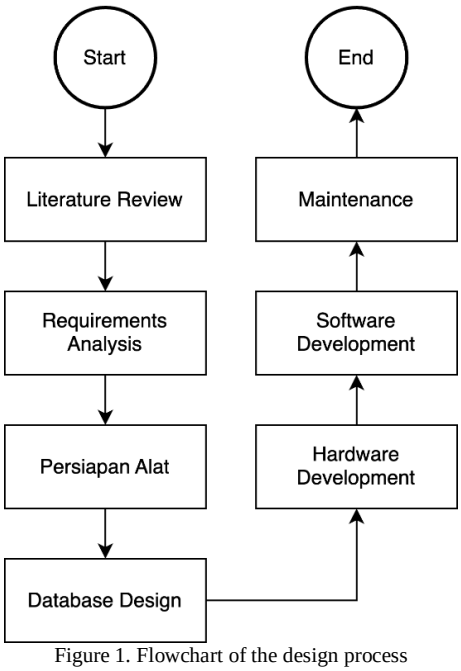


Figure 1. Flowchart of the design process

There are several preparations that need to be prepared, especially in the preparation of hardware tools so that the research process can run well, namely preparing the ESP32 microcontroller as the main component that can control other devices [5], other devices connected to the ESP32 microcontroller are RFID Reader [7] as an input tool for the ESP32 microcontroller. In Figure 2 provides details of the path related to the design of hardware devices and in Figure 3 displays the appearance of each device that is interconnected and will be used.

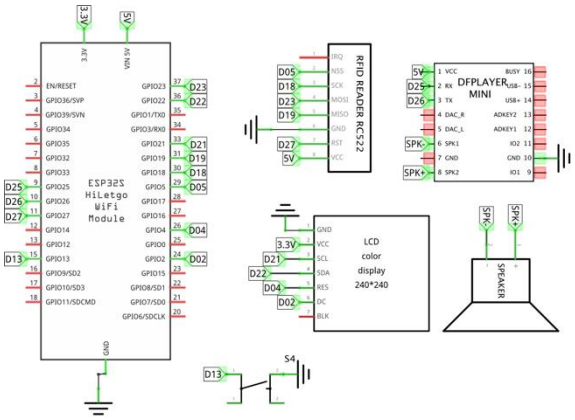


Figure 2. Schematic hardware design

The software system design will utilize MQTT as a means of communication between hardware devices and software [8]. The server will respond to the published results from the hardware which are then stored in the blockchain and database and will be displayed in the mobile application. Simulation of communication between systems is shown in Figure 4.

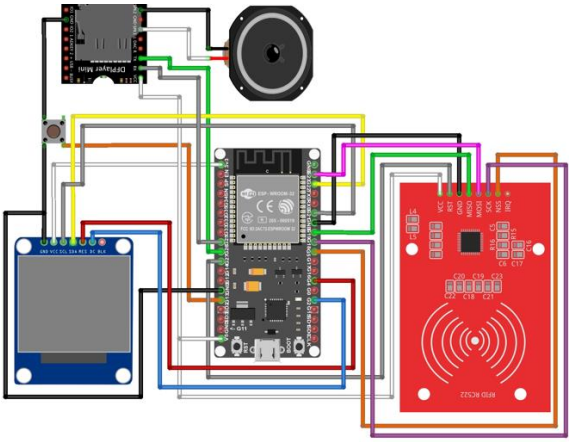

Figure 3. Hardware device display

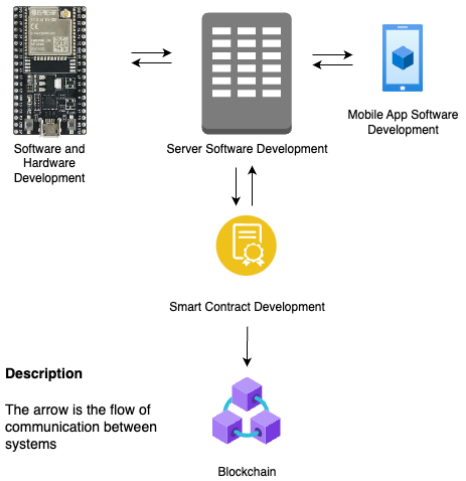


Figure 4. Simulation of communication between systems

All processes carried out will be tested using black box testing techniques [15]. The testing process will be carried out on the main functions in the designed application. This testing process is important so that the developed system avoids system failures that will affect the performance of the system and can provide satisfaction to users of the developed application.

## 3. RESULTS AND DISCUSSION

The developed system uses open source software, such as the use of PostgreSQL database. The application development process will follow the flow of Figure 1, which will begin by looking for similar research references, then designing a hardware system that is useful for the communication process between systems.

The system will be handled by a server designed using NodeJS technology. The server not only handles manual requests from the client but also makes requests made by the hardware system using the MQTT protocol. The process includes the

creation of smart contracts with the help of Hardhat, the blockchain validation process and also the process of storing data in the database.

The system process will be displayed in a mobile application designed with the help of Flutter. Application design includes the process of making application design, the connection process between the application and the server. The application will include the workflow of hardware and software systems, especially in the process of creating smart contracts and conducting blockchain transactions.

### 3.1. Hardware System Program Code

The process of connecting to the MQTT server requires coding into the microcontroller, the program code is in Figure 5.

```
while (!client.connect(clientId.c_str(),
mqtt_user, mqtt_password))
{
    delay(300);
}
json_active["sn_sensor"] = SN_SENSOR;
json_active["is_active"] = true;
json_active["timestamp"] =
timeClient.getEpochTime();
serializeJson(json_active, jsonString);
client.publish(topic_online,
strdup(jsonString.c_str()));
```

Figure 5. MQTT connection program code

In the program code, the connection process code will test the connection to MQTT by entering the username and password. If the connection process is successful, the publish process can be done by entering the topic and data to be sent.

Apart from the connection process to MQTT, it is necessary to code the RFID reader so that it can read RFID tags with the help of the MFRC522.h library. The program code for the RFID reader is in Figure 6.

```
if ((!rfid.PICC_IsNewCardPresent() ||
!rfid.PICC_ReadCardSerial()))
    return;

MFRC522::PICC_Type piccType =
rfid.PICC_GetType(rfid.uid.sak);

json_active["sn_sensor"] = SN_SENSOR;
json_active["type"] = switchMode;
json_active["rfid"] = GetUIDString(rfid.uid);
json_active["timestamp"] =
timeClient.getEpochTime();
serializeJson(json_active, jsonString);
client.publish(read_rfid,
strdup(jsonString.c_str()));
```

Figure 6. RFID reader program code

MQTT and RFID reader connection program code is the main code of the hardware system. This code functions for the process of reading RFID tags in the form of cards and the communication process to the server.

### 3.2. Software System Program Code

The process of creating software system code includes server creation, smart contract creation, blockchain transaction process, blockchain validation, connection with MQTT and application creation process as a means of interaction between developed systems.

The server program code will create a function to perform transactions, the function can be seen in Figure 7. The first process of this function is to check the allowed values, if it is appropriate, it will perform the process of storing data into the database with the help of NodeJS as an open-source server platform.

```
const store = async (req, res) => {
  response_error = {};
  const { error } = validateTransactionStore(req.body);
  if (error)
    error.details.forEach((err_msg) => {
      response_error[err_msg.path[0]] = err_msg.message;
    });
  if (Object.keys(response_error).length === 0) {
    const {
      type, total_payment, txn_hash, id_user, id_hardware,
      id_card, id_outlet, status,
    } = req.body;
    try {
      const result = await prisma.$transaction(
        async (prisma) => {
          return await prisma.transactions.create({
            data: {
              type: parseInt(type),
              total_payment: parseInt(total_payment),
              txn_hash, id_user, id_hardware, id_card,
              id_outlet, status,
            },
          });
        },
        {
          maxWait: 5000, timeout: 10000,
        }
      );
      responseServer200(res, "Successfully store
transaction!", result);
    } catch (error) {
      responseServer500(
        res,"Create new transaction failed!, check error",
        error
      );
    }
  } else {
    responseServer500(
      res, "Your request cannot run due to an error,
check",
JSON.parse(JSON.stringify(response_error).replace(/\\"/g,
""))
    );
  }
};
```

Figure 7. Transaction storage process function

The function that has been created will be included in the route so that the function that has been created can be run, the route will validate the function access. The validation function will be called first before the transaction storage function. Figure 8 shows how the code is written.

```
import { express, baseURL } from
"../configs/server.config.js";
const route = express.Router();
const basicURI = `/api/${baseURL}`;

route.post(`${basicURI}/transaction/save`,
checkVerifAccess, transaction.store);

export default route;
```

Figure 8. Program code for route function

The creation of the smart contract will use the help of Hardhat as the compilation engine of the smart contract file. The smart contract creation process will generate a smart contract address, as well as an ABI (Application Binary Interface) file. The smart contract address result has a hexadecimal string data type with a length of 42 characters. Address smart contracts are useful for identifying data on the blockchain. The smart contract creation process will require an Ethereum balance, the balance is for gas for the smart contract compilation process. The gas provided will fluctuate based on the busyness of the compilation process. The code form of the smart contract can be seen in Figure 9.

```
pragma solidity ^0.8.0;
contract SmartContract {
  event TransactionBlockchain(
     string id_transaction, string id_user, string id_hardware,
     string id_card, string id_outlet, uint256 total_payment,
     uint8 type_transaction, string status, string created_at,
     string updated_at
  );
  function saveTransactionHistory(
     string memory id_transaction, string memory id_user,
     string memory id_hardware, string memory id_card,
     string memory id_outlet, uint256 total_payment,
     uint8 type_transaction, string memory status,
     string memory created_at, string memory updated_at
  ) public payable {
     emit TransactionBlockchain(
        id_transaction, id_user, id_hardware, id_card, id_outlet,
        total_payment, type_transaction, status, created_at,
        updated_at
     );
  }
}
```

Figure 9. Smart contract code

The process of creating blockchain transactions will use the results of the address smart contract as a container to store each successful transaction. The blockchain transaction will provide a hash of the confirmation of each block in the blockchain, this hash has the same data type as the address smart contract but differs from the number of characters, the characters in the blockchain hash are 66 characters, more than the address smart contract which is only 42 characters.

The blockchain transaction will use the ABI result from the smart contract code compilation process to store the data. The use of the wallet address of the Ethereum Test Network is also required. The function of Test Network is to simplify the development process. The results of blockchains that use the Test Network can also be checked. In the transaction process, gas is needed so that the transaction can be executed, the gas from the transaction process is volatile like the smart contract compilation process.

In application development, especially in the smart contract compilation process, the gas contained in the compilation process will be borne by the application developer. In the blockchain transaction process, the gas is no longer borne by the developer but by the user. The code of transaction creation can be seen in Figure 10.

```
const options = {
     gasLimit: 3000000,
};
const transactionBlockchain = await callContract(
     resultOutlet.wallet_json, resultOutlet.smart_contract
  ).saveTransactionHistory(
     result.id_transaction, result.id_user, result.id_hardware,
     result.id_card, result.id_outlet, result.total_payment,
     result.type, result.status,
     formatDateTime(result.created_at),
     formatDateTime(result.updated_at),
     options
   ).catch((error) => {
     clientMqtt.publish(
     resultTransaction.id_transaction,
     JSON.stringify({
          status: false, message: "Error Create Blockchain",
          code: 500,
        })
     );
     return;
   });
```

Figure 10. Blockchain transaction code

The blockchain validation process uses a hash of the results of the blockchain transaction process. The hash will be used as an identifier on the blockchain network. The validation process will produce data that has been entered into the blockchain, the order of the data will be the same as the smart contract function. Figure 11 is the code to validate the blockchain hash.

```
const providerEthers =
    new ethers.providers.JsonRpcProvider(networkUrlEthers);
const getTransactionReceipt = async (txn_hash) => {
  const receipt =
    await providerEthers.getTransactionReceipt(txn_hash);
  const contractABI = "file abi";
  const iface = new ethers.utils.Interface(contractABI);
  const logs = receipt.logs.map(
    (log) => iface.parseLog(log));
  return logs;
};
```

Figure 11. Blockchain validation code

The MQTT connection on the server has a function to detect new information on the hardware. MQTT is a protocol that utilizes publish and subscribe, publish exists in the hardware system, in the software system both publish and subscribe are applied. The use of the MQTT protocol as a hardware and software communication tool can lighten the performance of the server.

The server will subscribe with the same topic as the hardware in order to connect. The function of the MQTT topic is useful as an identifier for an event that is sent. The event will be captured if it has the same topic between the two systems. The server publishes information to the mobile application, such as success or failure information.

The code used has almost the same typing logic as the code in the hardware system. There is a configuration to enter a username and password, and if it is connected to MQTT, the server system can receive information sent from the hardware.

Application development will use Flutter as the basis for application development. Flutter can not only run on android-based applications, but can also run on IOS, websites, and desktops, this is because flutter is a library that can run cross-platform.

```
import mqtt from "mqtt";
const mqttClientConfig = {
  host: hostMQTT, port: portMQTT, protocol: "mqtts",
  username: usernameMQTT, password: passwordMQTT,
};
const clientMqtt = mqtt.connect(mqttClientConfig);
clientMqtt.publish(
  resultTransaction.id_transaction,
  JSON.stringify({
    status: false, message: "Hardware Not Found",
    code: 404,
  })
);
clientMqtt.on("message", async (topic, message) => {})
```

Figure 12. MQTT server connection code

Cross-platform is a development technique that only writes the same code for each device used. Flutter provides such techniques in order to increase the flexibility of the system. Flutter uses the dart language for the code generation process.

Dart can be used to create a design view, as well as a server. In the process of creating an application design using Dart, it will apply material design. Material design is a technique for making each display component not from scratch, but already provides a template that is ready to use. Material design in dart is also called a widget.

Flutter provides state management that can be used. State management is a technique that separates the appearance of the program and the logic of the program process. This process will simplify development if the stage of the application being developed is large and there are many displays and logic functions behind the display.

```
import 'package:flutter/material.dart';

class IconAppbarCustomWidget extends StatelessWidget {
  IconAppbarCustomWidget({
    super.key, required this.iconType, required this.functionTap,
    this.backgroundColor, this.iconColor,
  });

  final IconData iconType;
  final Function() functionTap;
  final Color? backgroundColor, iconColor;

  @override
  Widget build(BuildContext context) {
    return Container(
      decoration: BoxDecoration(
        borderRadius: BorderRadius.circular(50),
        color: backgroundColor ?? Colors.grey.shade300,
      ),
      child: Padding(padding: const EdgeInsets.all(8),
        child: InkWell(onTap: functionTap,
          child: Ink(child: Icon(iconType,size: 20,
              color: iconColor ?? Colors.black,
            ),
          ),
        ),
      ),
    );
  }
}
```

Figure 13. Creation of a new widget

In application development, researchers use widgets that have been provided by Flutter. The widget is not used directly, there are some modifications to adjust the display. Flutter can create new widgets from the widgets already provided, as in Figure 13, the widgets provided by flutter are modified as needed and named with different widgets from flutter. This process can help create code that is neat, structured, and easy to read by future developers.

Flutter is not only used for display, but to perform server logic. In application development, researchers utilize the flutter feature to create a connection between the mobile application and the server. The use of this feature will create a model of the server response results which will then be displayed. The process of creating a model as in Figure 14 which creates a model for the server response when updating.

```
import 'package:json_annotation/json_annotation.dart';
part 'UpdateCard.g.dart';

@JsonSerializable(explicitToJson: true)
class DataUpdateCard {
  final int balance; final bool is_active;
  DataUpdateCard({
    required this.balance, required this.is_active,
  });
  factory DataUpdateCard.fromJson(Map<String, dynamic> json) =>
      _$DataUpdateCardFromJson(json);
  Map<String, dynamic> toJson() => _$DataUpdateCardToJson(this);
}
class UpdateCard {
  final bool status; final String message;
  final DataUpdateCard? data;
  UpdateCard({
    required this.status, required this.message, this.data,
  });
  factory UpdateCard.fromJson(Map<String, dynamic> json) =>
_$UpdateCardFromJson(json);
  Map<String, dynamic> toJson() => _$UpdateCardToJson(this);
}
```

Figure 14. Modeling the update

The model created will be used in the creation of the server connection. This process will use the http library to make the connection. The code that will be created can be given conditions that might occur if there is a system failure, the code can be seen in Figure 15.

```
Future updateCard(String token, String id_card, String balance) async {
  try {
    Uri uri = Uri.parse("${url}card/$id_card/update");
    final response = await http.put(
      uri, body: {"balance": balance,}, headers: _setHeaders(token),
    );
    Map<String, dynamic> data =
      (json.decode(response.body) as Map<String, dynamic>);
    if (response.statusCode == 200) {
      return UpdateCard.fromJson(data);
    } else if (response.statusCode == 401) {
      String? newToken = await refreshTokenAction();
      if (newToken != null) {
        return updateCard(newToken, id_card, balance);
      } else {
        return UpdateCard(
          status: false, message: "refresh token verification failed");
      }
    } else {
      return UpdateCard.fromJson(data);
    }
  } catch (e) {
    Map<String, dynamic> data =
      (json.decode('{"status": false, "message": "${e.toString()}"}')
            as Map<String, dynamic>);
    return UpdateCard.fromJson(data);
  }
}
```

Figure 15. Server connection code

### 3.3. System View

The system to be developed requires a display in order to interact with users, making the design will use wireframes as a framework for the application display. Wireframes consist of text, layout, and

images. Wireframe has a purpose so that the display design process can be made well, by applying user experience. All display ideas will be drawn first on the wireframe which will then be developed again in the user interface design.

Wireframes only display black and white designs accompanied by boxes and lines to organize each element that will be created. Wireframes are useful to facilitate application development, because if you already have a design framework, it will minimize changes. In Figure 16 provides the main view of the application, in this view there are three different main views according to user rights, namely payment, outlet, and counter.

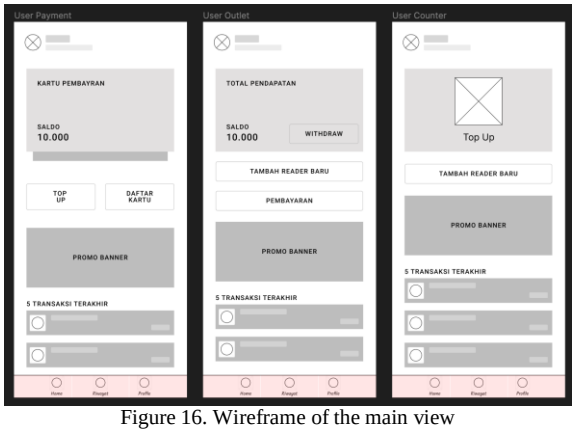


Figure 16. Wireframe of the main view

If the wireframe has been designed, then the next step will be to enter the application development stage, at the development stage will make the application based on the agreed design.

Figure 17 provides the main display that has been designed into the application. The display already has colors, functions of each button, and animations. This process not only changes the appearance of the wireframe, but provides a user experience of the application created, such as application speed, storage used, and security in the application in terms of appearance.

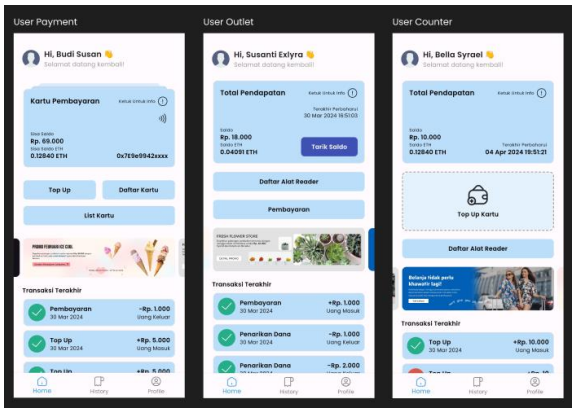


Figure 17. Main view of the application

The appearance of the hardware system is also designed to be small and there are not many cables installed, connections between components using connectors that can be easily removed, so that when damage occurs it can replace components. The designed system display can be seen in Figure 18 which displays different messages when first operated and when it is ready to operate. When the hardware system operates, it will provide feedback in the form of sound produced by the speaker.

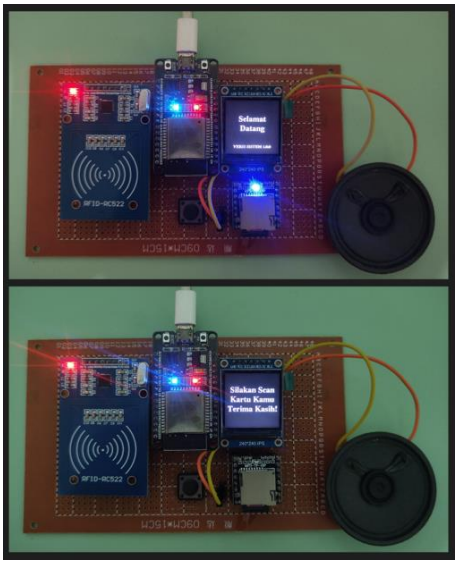


Figure 18. Hardware system view

### 3.4. System Simulation

Simulation of the results of system development that has been done can be run by opening the application installed on the mobile device, then preparing the card that has been registered in the user account in the application. The card used is an RFID card that has a balance on the account, the next process is tapping on the hardware device that has been activated by the outlet user, in the tapping process the hardware system will provide information regarding the card successfully scanned. The process will be completed and users can see a list of transactions that have been completed, the system simulation process is in Figure 19.


Gambar 19. Simulasi sistem

The simulation carried out has been connected to the database and blockchain system, the blockchain data can be checked by pressing the "Etherscan Blockchain" button which is already connected to the

blockchain checking service provider and when you want to validate the transaction data carried out, you can press the "Check Transaction Validation" button to check the blockchain system and database.

### 3.5. System Testing

System testing aims to determine whether the system developed is in accordance with the design or initial objectives, and is suitable for use. System testing in this research will use the black box testing method.

The system testing process in this study will be divided into several tests, including testing each main function of the application, the size of the application after being installed on the device, the time required when each function on the server, the costs incurred for each smart contract and blockchain transaction, as well as the time required to make smart contracts and blockchain transactions. The test process will be run and will be checked whether the system output is appropriate or not.

Table 1. Results of testing the main functions of the application with Black Box Testing

| System Function | Testing Scenario | Testing Data | Desired Result | Result of Testing |
|---|---|---|---|---|
| Register | Can create a new account | Enter name, username, email, role, password | New account successfully created marked server response code 200 | Aligned |
| Save New Reader | Can pair software to hardware, and save to database | Enter hardware serial number, Wi-Fi name, Wi-Fi password | Successfully pairing with hardware and hardware data stored in the database marked server response code 200 | Aligned |
| Save New Card | Can store new RFID cards into the database | Enter id_rfid | Successfully save the data into the database | Aligned |
| Tapping Card | Can make payments with RFID cards | Tapping the RFID card on the hardware | The transaction is successful and the balance is reduced marked with server response code 200 | Aligned |
| Create New Wallet Address | Can create a wallet address account | Entering Network API Key | Successfully created a wallet address account marked as providing a wallet address | Aligned |
| Save Transaction | Can save transactions created into the database | Enter id_card, id_outside device, id_user, total_payment, txn_hash, type, status | Transaction data saved successfully marked server response code 200 | Aligned |
| Create Smart Contract | Can compile the smart contract file and get the smart contract address | Enter Network API Key, Private Key, Network type | Successfully compile marked address smart contract | Aligned |
| Save Transaction Blockchain | Can save transactions created into the database | Enter id_card, id_hardware_id_outlet, id_user, total_payment, type, status | Successfully store data into the blockchain marked by the blockchain hash | Aligned |
| Validate Blockchain | Can validate data in the database and on the blockchain | Enter transaction id_transaction in database and txn_hash in blockchain validator | Provides a comparison result with the output value whether or not it matches | Aligned |

From the results of testing the main functions in Table 1, we have obtained results that are in accordance with what is expected and from the results of testing the developed system can run well, such as in the Register function, Save New Reader, Save New Card, Tapping Card, Create New Wallet Address, Save Transaction, Create Smart Contract, Save Transaction Blockchain, and Validate Blockchain.

Size testing of the developed application can be seen in Table 2. The table provides details about the application created with flutter. Testing the application using a smartphone type M2007J20CG or POCO X3 NFC, the application installation testing process has been successful and can run well, but the device must prepare a minimum of 200 Mega Byte so that the application can run optimally.

Table 2. Test results of application installation

| Device Type | App Size | Size After Installed |
|---|---|---|
| M2007J20CG or POCO X3 NFC | 33.1 Mega Byte (MB) | 138 Mega Byte (MB) |

Testing the speed of the local server when providing the results of a given request can be seen in Table 3. Testing the speed of the local server is obtained less than 10 seconds for each function that is not connected to third party applications, this speed is good, if it exceeds the target time it can interfere with the user experience of the user. Therefore, data processing issues on the server are also a major consideration, not just the features provided.

Tabel 3. Hasil pengujian kecepatan *server* lokal

| Function | Processing Time |
|---|---|
| Login | 218 milliseconds |
| Save New Reader | 442 milliseconds |
| Save New Card | 323 milliseconds |
| Tapping Card | 416 milliseconds |
| Save Transaction | 867 milliseconds |

Testing gas costs when compiling smart contracts and conducting blockchain transactions can be seen in Table 4 and Table 5. Gas fee testing was carried out 5 times for each smart contract compilation process and blockchain transactions. The average gas fee test for smart contract compilation does not exceed 0.003 ETH or 9.2 USD and blockchain transactions do not exceed 0.001 ETH or 3 USD, the testing process uses the Sepolia Test

Network. Test Network Sepolia is a network used for the development process on Ethereum. The USD price contained in Table 4 and Table 5 is the price if using the Main Network Ethereum which was taken as of April 18, 2024 23.38 WIB, namely 1 ETH at the price of 3,083 USD.

Table 4. Testing results of smart contract compilation gas cost

| Date and Time | Gas Cost | USD Price |
|---|---|---|
| 27-03-2024 12:52:12 WIB | 0.0018 ETH | 5.5 $ |
| 27-03-2024 12:56:48 WIB | 0.0027 ETH | 8.3 $ |
| 28-03-2024 19:57:36 WIB | 0.0029 ETH | 8.9 $ |
| 28-03-2024 23:02:00 WIB | 0.0003 ETH | 0.9 $ |
| 29-03-2024 15:38:24 WIB | 0.0018 ETH | 5.5 $ |
| **Average** | **0.0019 ETH** | **5.8 $** |

Table 5. Testing results of blockchain transaction gas fees

| Date and Time | Gas Cost | USD Price |
|---|---|---|
| 30-03-2024 01:37:12 WIB | 0.0006 ETH | 1.8 $ |
| 30-03-2024 15:01:00 WIB | 0.0007 ETH | 2.1 $ |
| 30-03-2024 15:08:36 WIB | 0.0008 ETH | 2.4 $ |
| 30-03-2024 15:11:36 WIB | 0.0006 ETH | 1.8 $ |
| 30-03-2024 16:51:12 WIB | 0.0007 ETH | 2.1 $ |
| **Average** | **0.0007 ETH** | **2.1 $** |

Speed testing in compiling smart contracts and blockchain transactions is contained in Table 6 and Table 7. Speed testing also uses the data results of Table 4 and Table 5, the process aims to find out how long it takes to compile smart contracts and blockchain transactions when combined with the system. The testing process is carried out 5 times in each process. The average test result for smart contract compilation speed is 30 seconds and for blockchain transactions is 6 seconds. To track transactions made, you can copy the smart contract address and transaction hash which is then searched for on the Test Network Sepolia on the etherscan website.

Table 6: Test results of smart contract compilation speed

| Date and Time | Processing Time | *Smart Contract Address* |
|---|---|---|
| 27-03-2024 12:52:12 WIB | 30.1 seconds | 0x99f2544c90e4bb1c2d1b00be1a7d266949ba2976 |
| 27-03-2024 12:56:48 WIB | 32.6 seconds | 0x5f902038576ec7f410640f6aa56b29bb1f0f1128 |
| 28-03-2024 19:57:36 WIB | 31.4 seconds | 0xeeacabdaea6847985b7abd26e72ad82b6b32bd27 |
| 28-03-2024 23:02:00 WIB | 29.7 seconds | 0xcc04b4a8144f7c6dc0cd51a2e093f07d1f4439a4 |
| 29-03-2024 15:38:24 WIB | 30.0 seconds | 0xbfe2cd9802d1bd0f646215590417ad586cfa9083 |
| **Average** | **30.0 seconds** | |

Table 7. Test results of blockchain transaction speed

| Date and Time | Processing Time | *Transaction Hash* |
|---|---|---|
| 30-03-2024 01:37:12 WIB | 6.0 seconds | 0x4ac2c73835a41f813e74c6f0d67d317a5790a86bc768bbe0c8e279bc2c45eedc |
| 30-03-2024 15:01:00 WIB | 6.1 seconds | 0xb9bcfb5ce2f19bace54f4d50260aa59d4f084b14974202c928f0ce0617d6a90b |
| 30-03-2024 15:08:36 WIB | 5.9 seconds | 0x406214218c92603db87a2bcd6b5efe52dcce65d59763902a59194953e8179f65 |
| 30-03-2024 15:11:36 WIB | 6.1 seconds | 0x51545d2e2008d194df5481fa01ada2c7d5251c8d5b2673b6805209f7cc44a6a6 |
| 30-03-2024 16:51:12 WIB | 5.9 seconds | 0xeecf10c65ba1c2f00e7fe3b98a3866150d925123a946a774a893e0763186848e |
| **Avarage** | **6.0 seconds** | |

The test results that have been carried out in the application development process get good results. Every data that has been stored in the blockchain network can be tracked by the user and can be validated against the data that has been stored between the data contained in the database and that contained in the blockchain, the data can be stored due to the help of smart contracts.

## 4. DISCUSSION

From previous research conducted such as in research that developed a prototype RFID system designed to simplify the recording of administrative payment transactions, increase the speed of presenting payment reports, reduce recording errors [16].

In another study [17], RFID technology is not only used to make payments but also for security tool automation systems for door openers. The reading process of the RFID sensor can detect the card up to a distance of 5 cm, and after successfully detecting the card then the door will open automatically with the help of solenoid and servo.

In research conducted by L Ismanto1, H Suwito AR, A N Fajar, Sfenrianto and S Bachtiar [18] provided a solution to overcome data security. The solution uses blockchain technology, this technology promises data security and transparency.

In discussing the security of privacy data, a container is needed to guarantee these data, the container in question is the use of blockchain. Research [19] that has been conducted provides a solution to the problem, namely by using blockchain, because it has decentralized, immutable, and transparent properties which are the main characteristics of a reliable security system.

In another study [20] suggested the implementation of a medical crowdfunding system by integrating trust and data transparency using blockchain technology. The research conducted uses the Hardhat system architecture with the Sepolia Test Network in order to test blockchain technology.

Research from Min-Hyuk Jeong and Sang-Kyun Kim, proposed a system that provides video streaming services with the Internet of Things and utilizes blockchain technology and cryptocurrency (tokens) as a means of payment for video streaming services [21]. The payment process will be adjusted

by smart contracts that have been recorded in the blockchain system, all systems can be accessed by distributed applications (DApp).

Research from [22][23][24][25] provides a view of the PostgreSQL database which is useful for the data storage process in systems designed using NodeJS and other technologies, PostgreSQL is a database that is open source and free, not only PostgreSQL is free to use, but also Flutter technology developed by Google, Flutter is also an open source that can run on cross-platform systems and the process of making the system using the dart language which has the concept of widgets on its display.

User Interface or User Experience is the most important part when you want to build a system, the main role of UI / UX is so that users who use the system do not experience confusion or difficulty, and also the absence of system constraints that can affect UX from the user's side. In research [26][27] provides details on how to detail how to create a good system design from wireframe to implementation into the actual system.

From previous research that has been done, there are still several technologies that can be combined and made into a super application, the super application combines blockchain technology with an RFID contactless payment system which can reduce security problems from transactions that can be manipulated.

The results of research that have been carried out, researchers are developing new applications by combining blockchain technology into the RFID contactless payment system [28][29][30]. Application development prioritizes the security of transaction data that has been stored in the database so that it cannot be manipulated. Researchers use blockchain technology so that the security of transaction data that has been created can be stored safely and when you want to validate the transaction, it can be done. Transaction data stored in the blockchain is the same data in the database. Testing applications that have been built using the black box testing method, the tests carried out are appropriate and can run well. Researchers hope that this research can provide a choice of data security methods which can use blockchain technology as a form of data security, because blockchain technology provides data transparency and good security.

## 5. CONCLUSION

Based on the results of application development research that has been carried out, namely using blockchain technology to strengthen the security of data stored in the database, it is appropriate and can run well, characterized by successful partial system testing. The use of blockchain technology is expected to be the right choice for data security issues, the use of blockchain technology can not only be integrated in the same system as researchers but can also be done in other systems.

The use of blockchain technology can facilitate data tracking and data security, the data tracking process can be done using wallet addresses, smart contract addresses and hash transactions from the blockchain. Any data that has been stored in the blockchain will be difficult to manipulate.

The use of Ethereum as a blockchain transaction coin was successful with the help of the Sepolia Test Network. Test Network Sepolia is a container provided by Ethereum for development purposes that are different from the Ethereum main network. The use of Test Network Sepolia helps the development process of this application, especially in the data security system. This research is expected to be a lesson for further research, especially research on the topic of blockchain, so that it can develop applications that are more reliable in data security systems and data transparency.

There are suggestions that can help the further development of blockchain technology based on the results of this study, including it is hoped that it can speed up the time in compiling smart contracts and blockchain transactions, because these two processes are the most time-consuming processes, in terms of testing the speed of compiling smart contracts, it gets a speed result of 30 seconds, then in blockchain transactions it gets a result of 6 seconds per each transaction made. This process can reduce user experience when using the application. Another suggestion is the gas fee charged to users, if possible, other blockchain networks such as Binance Smart Chain, Polygon, Arbitrum and Harmony so that the gas fees used can be cheaper and more affordable for users.

## REFERENCES

[1] M. Syarif and W. Nugraha, "Pemodelan Diagram UML Sistem Pembayaran Tunai Pada Transaksi E-commerce," *JTIK (Jurnal Teknik Informatika Kaputama)*, vol. 4, no. 1, pp. 64–70, Jan. 2020, doi: 10.59697/jtik.v4i1.636.

[2] F. Hao, R. T. R. Qiu, J. Park, and K. Chon, "The Myth of Contactless Hospitality Service: Customers' Willingness to Pay," *Journal of Hospitality and Tourism Research*, vol. 47, no. 8, 2023, doi: 10.1177/10963480221081781.

[3] MOHD. Y. DM1, M. Agustantia, and S. Zulaiha, "Tindak Pidana Kejahatan Pemalsuan data (Data Forgery) dalam Bentuk Kejahatan Siber (Cyber Crime)," *Jurnal Pendidikan dan Konseling*, p. h.6638, 2022.

[4] T. W. E. Suryawijaya, "Memperkuat Keamanan Data melalui Teknologi Blockchain: Mengeksplorasi Implementasi Sukses dalam Transformasi Digital di Indonesia," *Jurnal Studi Kebijakan Publik*, vol. 2, no. 1, 2023, doi:

10.21787/jskp.2.2023.55-68.

[5] Z. Wu, K. Qiu, and J. Zhang, "A smart microcontroller architecture for the internet of things," *Sensors (Switzerland)*, vol. 20, no. 7, 2020, doi: 10.3390/s20071821.

[6] Z. Nurbekova, T. Tolganbaiuly, B. Nurbekov, A. Sagimbayeva, and Z. Kazhiakparova, "Project-based learning technology: An example in programming microcontrollers," *International Journal of Emerging Technologies in Learning*, vol. 15, no. 11, 2020, doi: 10.3991/IJET.V15I11.13267.

[7] W. C. Tan and M. S. Sidhu, "Review of RFID and IoT integration in supply chain management," *Operations Research Perspectives*, vol. 9, 2022, doi: 10.1016/j.orp.2022.100229.

[8] A. Kurnianto, J. Dedy Irawan, and F. Xaverius Ariwibisono, "Penerapan IoT (Internet of Things) Untuk Controlling Lampu Menggunakan Protokol MQTT Berbasis Web," *JATI (Jurnal Mahasiswa Teknik Informatika)*, vol. 6, no. 2, 2023, doi: 10.36040/jati.v6i2.5393.

[9] V. Buterin, "A next-generation smart contract and decentralized application platform," *Etherum*, no. January, 2014.

[10] L. Dimnik, *Smart contract based decentralized voting system*. theseus.fi, 2023. [Online]. Available: https://www.theseus.fi/handle/10024/793771

[11] W. Zou *et al.*, "Smart Contract Development: Challenges and Opportunities," *IEEE Transactions on Software Engineering*, vol. 47, no. 10, 2021, doi: 10.1109/TSE.2019.2942301.

[12] B. Erden, "Automated Unit Testing of Solidity Smart Contracts in an Educational Context," *wwwmatthes.in.tum.de*, [Online]. Available: https://wwwmatthes.in.tum.de/file/12ob34pjkeux8/Sebis-Public-Website/Student-Theses-Guided-Research/Current-Theses-Guided-Researches/Master-s-Thesis-Batuhan-Erden/Erden%20Master's%20Thesis.pdf

[13] R. Maulana and I. H. Ikasari, "Literature Review: Implementasi Perancangan Sistem Informasi Perpustakaan Sekolah Berbasis Web dengan Pendekatan Metode Waterfall," *JRIIN: Jurnal Riset Informatika dan Inovasi*, vol. 01, no. 01, 2023.

[14] A. A. Wahid, "Analisis Metode Waterfall Untuk Pengembangan Sistem Informasi," *Jurnal Ilmu-ilmu Informatika dan Manajemen STMIK*, no. November, 2020.

[15] O. Loyola-Gonzalez, "Black-box vs. White-Box: Understanding their advantages and weaknesses from a practical point of view," *IEEE Access*, vol. 7. 2019. doi: 10.1109/ACCESS.2019.2949286.

[16] Gunawan Wibisono, Vivi Kumalasari Subroto, and Danang Danang, "Analisa dan Perancangan Sistem Aplikasi Pembayaran Administrasi Menggunakan RFID Berbasis Client Server," *Kompak :Jurnal Ilmiah Komputerisasi Akuntansi*, vol. 13, no. 1, 2020, doi: 10.51903/kompak.v13i1.201.

[17] M. Yusup, "Teknologi Radio Frequency Identification (RFID) Sebagai Tools System Pembuka Pintu Outomatis pada Smart House," *Jurnal Media Infotama*, vol. 18, no. 2, 2022.

[18] L. Ismanto, H. S. Ar, A. N. Fajar, Sfenrianto, and S. Bachtiar, "Blockchain as E-Commerce Platform in Indonesia," in *Journal of Physics: Conference Series*, 2019. doi: 10.1088/1742-6596/1179/1/012114.

[19] T. Huynh-The *et al.*, "Blockchain for the metaverse: A Review," *Future Generation Computer Systems*, vol. 143, 2023, doi: 10.1016/j.future.2023.02.008.

[20] D. N. K. Raja *et al.*, "Medtrust - A Decentralized Model for Medical Crowdfunding Based on Smart Contract Using Etherium," *Int J Res Appl Sci Eng Technol*, vol. 11, no. 6, 2023, doi: 10.22214/ijraset.2023.53967.

[21] M. H. Jeong and S. K. Kim, "Video Streaming Based on Blockchain State Channel with IoT Camera," *Journal of Web Engineering*, vol. 21, no. 3, 2022, doi: 10.13052/jwe1540-9589.2134.

[22] A. Makris, K. Tserpes, G. Spiliopoulos, D. Zissis, and D. Anagnostopoulos, "MongoDB Vs PostgreSQL: A comparative study on performance aspects," *Geoinformatica*, vol. 25, no. 2, 2021, doi: 10.1007/s10707-020-00407-w.

[23] A. Rimal, *Developing a web application on NodeJS and MongoDB using ES6 and beyond*. theseus.fi, 2019. [Online]. Available: https://www.theseus.fi/bitstream/handle/10024/159951/Rimal_Aashis.pdf

[24] A. P. Pratama and M. Kamisutara, "Pengembangan Sistem Informasi Akademik Berbasis Mobile Menggunakan Flutter Di Universitas Narotama Surabaya," *Network Engineering Research Operation*, vol. 6, no. 2, 2021, doi: 10.21107/nero.v6i2.238.

[25] S. Faust, "Using Googles Flutter Framework for the Development of a Large-Scale Reference Application," *Tribal*, 2020.

[26] H. Herfandi, Y. Yuliadi, M. T. A. Zaen, F. Hamdani, and A. M. Safira, "Penerapan

Metode Design Thinking Dalam Pengembangan UI dan UX," *Building of Informatics, Technology and Science (BITS)*, vol. 4, no. 1, 2022, doi: 10.47065/bits.v4i1.1716.

[27] S. Nurbaiti Oktaviani, C. Fikri Aziz, and B. Maula Sulthon, "Analisa UI/UX Sistem Informasi Penjualan Berbasis Mobile Menggunakan Metode Prototype," *KLIK: Kajian Ilmiah Informatika dan Komputer*, vol. 2, no. 6, 2022, doi: 10.30865/klik.v2i6.401.

[28] D. Jiustian, "System-Server-RFID-Blockchain," GitHub. Accessed: Mar. 30, 2024. [Online]. Available: https://github.com/dannyjiustian/System-Server-RFID-Blockchain

[29] D. Jiustian, "System-Mobile-App-RFID-Blockchain," GitHub. Accessed: Mar. 30, 2024. [Online]. Available: https://github.com/dannyjiustian/System-Mobile-App-RFID-Blockchain

[30] D. Jiustian, "System-Hardware-RFID-Blockchain," GitHub. Accessed: Mar. 30, 2024. [Online]. Available: https://github.com/dannyjiustian/System-Hardware-RFID-Blockchain.