International Journal of Sustainable Development and Planning
Vol., No., Month, Year, pp. **-**
Journal homepage: http://iieta.org/journals/ijsdp

Implementation of Housing Management Information System Using Agile Method with Extreme Programming Model (Case Study of Graha Nirmala Housing)

Nuril Anwar*, Yossan Sandi Rahmadi

[1] Informatics Department, Faculty of Industrial Technology Universitas Ahmad Dahlan, Daerah Istimewa Yogyakarta 55191
[2] Informatics Department, Faculty of Industrial Technology Universitas Ahmad Dahlan, Daerah Istimewa Yogyakarta 55191

Corresponding Author Email:

https://doi.org/10.18280/ijsdp.xxxxxx

Received:
Revised:
Accepted:
Available online:

Keywords:
Agile, Extreme Programming, Management Information System, Payment)

ABSTRACT

Graha Nirmala housing has admins that manage housing data and occupant's contribution payment. This management use Microsoft Excel to manage the housing data. However, this management still has several problems which are data inconsistency since there is no data validation on each column and most of the flows have dependency to the admin. Admin dependency increase the probability of human error and redundant actions that makes the management less efficient. Housing management information system is needed to tackle the problems and meet housing needs. The system able to manage data of the housing and handle contribution payment with automatic payment checking. In this research, a housing management information system developed using Agile method with Extreme Programming model as the development stages framework. This research successfully developed a housing management information system. In terms of functional testing with Blackbox testing, all the test cases that tested in every sprint are success, it means the system works as expected according to its functionality. This system able to manage housing data, ensure data type of each column, generate bill every month, and handle payment with automatic payment checking. It tackles pain points of previous system and accommodates the housing needs.

1. INTRODUCTION

The development of technology from time to time makes it easier for humans to do their jobs. One of the technological advances is the development of management information systems for various subjects that have managerial processes. This management information system is able to optimize the workflow and minimize the occurrence of human error. It developed along with technological advances to meet the needs of its users. One of the things that wants to be developed so that progress able to be realized is by increasing developments in the field of technology, one of which is in the housing that designed to be able to provide services and conveniences to managers and housing occupants in the housing area who initially still used manual management information systems. The main issue concerning the management information system for a group or individual is how to implement a management information system while at the same time utilizing it for the benefit of groups and individuals, in the form of integrated management and connected in a computer network called a management information system.

This information system can be defined as a system within an organization which is a combination of human resources, technology, facilities, media, procedures, and controls aimed at obtaining important communication lines, processing certain types of routine transactions, giving signals to management and others. on important internal and external events and provides an information basis for making good decisions [1]. The web-based housing management information system is a system designed for housing management needs in an effort to meet shared needs such as information related to housing and its occupants, as well as cash management for housing operational and maintenance costs. The purpose of implementing a web-based management information system implementation process is to support management function activities such as planning, organizing, and controlling in order to support the achievement of the goals and objectives of operational functions [10].

Graha Nirmala housing which is the case study in this research is everyone's dream housing. This housing has a safe and comfortable home, no exception for students to someone who wants to enjoy old age. With the spirit of helping to provide a safe and comfortable place, the owner of this housing tries to offer housing which is only a few minutes of drive to the city (Malioboro and Kraton).

Agile method is a specific approach that using in software development. This method response the uncertainty of feature development with iterative incremental update that called as sprint. Agile method response the uncertainty of feature development with iterative incremental update that called as sprint. Agile does not scale up to the requirements of the systems build by larger enterprises. Another way of working is needed that applies the power of Agile, but also leverages the more extensive knowledge supplies of lean product development and systems thinking [18]. Agile method has several models and Extreme Programming is one of them. Extreme Programming is a method of software development that able to make client add or change business process to the application throughout the development process. Management

Information System have different needs for a specific use case and may uncertainty of its feature. It is related with the basic concept of Agile with Extreme Programming model that able to handle changes as development progresses.

Currently, Graha Nirmala housing has also used a management information system using Microsoft Excel to assist housing management. This management way is carried out so that the occupants in the housing can achieve common goals and meet their needs. However, the current management information system which uses Excel still has several problems.

With the current system it's hard to maintain that admin input the correct types and follow the value rule. For example, phone number should have format country code followed by the number, so when the data needed, it will be consistent. With the current system, it cannot guarantee that the phone number follow that rule. To make it consistent in the current system, it also has pain point to input the data from the housing occupant. The pain point is we cannot expect the housing occupant will give their phone number with correct number, thus the admin has to redundantly make sure the format is correct. If the phone number format is not correct, the admin has to fix the format manually. This problem also applies in every fields that has specific rule.

The current flow of pay the housing bill is admin remind all housing occupant every month, then occupant pay the bill to admin bank account. The occupants need to send the payment proof to confirm to the admin that they already paid. In the other side, admin manually input the payment data. Two problems occurred in the admin side, the first problem payment proof guaranty, then the second payment amount guarantee. We cannot guarantee that the payment proof is valid one, because the payment proof photo can be faked. There is also possibility that the occupants pay the bill without its fine, so we cannot guarantee the occupant pay the bill with appropriate bill amounts. Thus, admin need to double check by its side whether the money already enter admin's bank account and its bill amount is appropriate. But at the same time, double-checking is redundant activity that cannot be guaranteed that it will be done by admin. When the worst scenario happens, it will harm the housing cash and lead to extra step to do by admin which is not efficient.

There are several ways to tackle pain points of the bill payment as mentioned above. One of the ways is using payment gateway that integrated with our system and set the rule logic for the bill amount in the system. Payment gateway is an intermediary for payment confirmation from the sales website to the online system by a third party or bank directly, which if the payment is rejected for certain reasons, it will be returned to our sales system and vice versa, if the transaction is successful, the payment will be immediately processed digitally/online [16]. Our system can forward the payment request to the payment gateway to be handled, and our system can reflect the payment status that will be send by the payment gateway. In that way, housing admin no need to check whether the bill already paid or not, and the rule logic can make sure the occupants pay with the appropriate amounts.

Therefore, hopefully the research able to develop a housing management information using agile method with Extreme Programming model. This research aims to tackling pain points that occurred in previous management system and accommodating the housing needs.

2. METHODS

2.1 Method of Collecting Data

There are two method of data collection method that used in this research which are literature study and interviews. Literature study is a type of research with a method that is a series of activities related to the methods of collecting library data, reading and taking notes, and processing research materials. This literature study was conducted by researchers between after determining the research topic and determining the formulation of the problem before going into the field to collect the necessary data [3]. Literature studies are conducted by each researcher with the main objective of finding the basis for obtaining and building theoretical foundations, frameworks of thought, and determining provisional assumptions or also known as research hypotheses.

So that writer can classify, allocate, organize, and use a variety of literature in their fields. An interview is a conversation carried out to find something. Interviews were conducted by two parties, namely the interviewer who asked the question and the interviewee who answered the question. According to Rosaliza (2015) interviews are an important process in conducting a research, especially in qualitative research. Interviews are used to obtain information related to facts, beliefs, feelings, desires, and so on that are needed to fulfill the research objectives [17].

2.2 System Development Stages

Several development stages are conducted to develop the system. Separation of this stages help the developer to have clear step to create system. When there is problem on the system, it easily able to identify from which stage the problem occurred and from that we can do iteration. Before entering the software development method, which in this research is Extreme Programming model, there is sprint plan stage.

Sprint plan stage is to identify the problem and feature as solution that able to develop and scheduling the development to make the development measurable.

After the sprint plan defined, Extreme Programming cycle conducted. The explanation of each stages has activity as below:

1. Planning is to defining functional and non-functional requirements. It also researches the implementation of design and coding as the addition to have context before implement it.
2. Design is to drawing the flow of the feature and its UI/UX design.
3. Coding is to implementing the design and makes it able to works as expected
4. Test is to making sure the system works as expected with Blackbox method.

3. RESULTS AND DISCUSSION

3.1 Sprint Plan

After define all the features, sprint plan created, so that the sprint can be structured by what features will be developed in that sprint. With the sprint plan, the overall development process can be measured by counting sum of each sprint duration. The first sprint activity is development preparation that prepare all the general things that used in feature development. After the first sprint, the activity is feature development that divided by related features. The duration calculation is using week calculation that in a week has 5 working days. So, 10 days equal to 2 weeks.

3.2 Development Preparation Sprint

This general preparation develops non-feature specific design and code. It means the design and code be reusable in the next development and even reusable for next sprint development explanation. There are no functional requirements because no feature development in this sprint, General design is the design that has same patterns that can be used in next sprint. General design is the design that has same patterns that can be used in next sprint. General coding is reusable implementation that used in many features. It includes Preparation, UI component and Utility function. There are several preparations that need to be done, which are: Setup Next.js project, Folder Structure, Database and Object-relational Mapper (ORM) configuration, API Handler Wrapper, Reusable Component, Controlled Form and its fields component, and CRUD API implementation. In General Testing, There are many test cases can be generalized for the purpose of avoid redundant explanation in the next sprint.

3.3 House & Account Sprint

As the sprint 2 plan, this sprint is feature development of transaction management and occupant transaction. Feature's user story translated into functional requirements. Design of House & Account Sprint contains Flowchart and Hi-Fi UI Design. The Coding of House & Account Sprint contains Backend, and Frontend. and Frontend. Testing of House & Account Sprint using Blackbox testing method to assess the functionality of House Management, Account Management, and Occupant Profile features.

Table 1. Sprint Two Blackbox testing using reusable test cases

BLACKBOX TESTING USING REUSABLE TEST CASES		
Test Case	Actual Result	Summary
Go to feature page (House Management, Account Management, Occupant Profile)	Page directed to feature page	Success
Go to protected feature page with invalid session (House Management, Account Management, Occupant Profile)	Page directed to login page with error message	Success
Edit data (house, account, occupant profile, family member)	Page directed back to manage feature page and show the updated data.	Success
Delete data (house, account, occupant family member)	Page directed back to manage feature page and show the updated data.	Success

As we can see on table above, there are test cases that can be used with reusable test case that already defined on Table 1. There is specific test case in sprint 2 feature that explained on the table below.

Table 2. Specific Occupant Management Blackbox testing

SPECIFIC OCCUPANT MANAGEMENT BLACKBOX TESTING					
Precondition	Test Case	Test Scenario	Expected Result	Actual result	Summary
Form page/component loaded In our database there are 2 houses with its occupant as described below. • A1 house: Owner: exist Renter: not exist • A2 house: Owner: not exist Renter: not exist	Select values of house is depend on selected value of role type	1. On "Tipe" field, select "Pemilik"	Select values of house is A2.	Select values of house is A2.	Success
		1. On "Tipe" field, select "Penyewa"	Select values of house are A1 and A2.	Select values of house are A1 and A2.	Success

3.4 Authentication Sprint

In the Plan of Authentication Sprint, Almost the API routes and web pages are protected, because some entity rules can only be interacted with one or some specific roles. So, in this sprint, Authentication feature developed. There are several strategies for system authentication. One of them is JSON Web Token (JWT) Authentication that used as SIMGN authentication strategy.

With JWT authentication, if the user login we give token as the API response, then the frontend store it in browser cookies. However, we need to set the token cookies to be "http-only" to avoid Cross-site scripting (XSS) attacks. Set token as "http-only" makes the token cannot be accessed via client-side script like JavaScript. But then, we need two strategies to set the token from the frontend since we cannot access it on the client.

As explained, Next.js has two environment types of rendering which are client-side and server-side. These types of

rendering need to handle differently since the cookies behavior is different from each type.
On the client-side rendering, we directly access our API, the default behavior is the request attach the cookies as request header. So, in the API we can directly access the token cookies.

On the other side, server-side rendering page requested by browser and the fetch API code will be executed on the server. The cookies sent as headers at the time browser request the page. Then, when execute the fetch API in the server, there is no behavior to send cookies with its request anymore. But, as the traditional way of JWT Authentication, we can assign token to the Authentication headers since we can access the token from server-side rendering.

Other than authentication feature, fetch wrapper function is need to be created to wrap logic above, so that it able to be reused every time frontend calling API request.
In the Design of Authentication Sprint, there is a Flowchart To describe the fetch strategy, flowchart of fetch strategy drawn that can be seen on Figure 1.

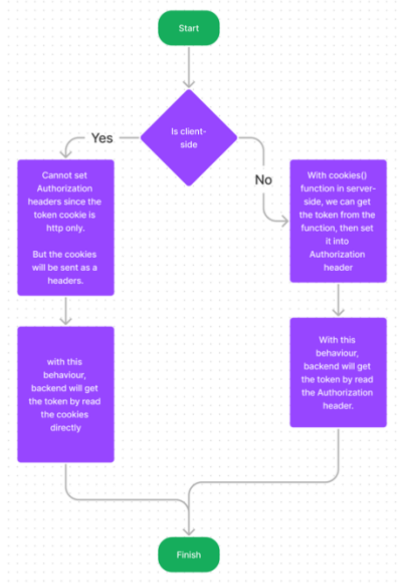

Start
Is client-side
Yes
No
Cannot set Authorization headers since the token cookie is http only.
But the cookies will be sent as a headers.
With cookies() function in server-side, we can get the token from the function, then set it into Authorization header
with this behaviour, backend will get the token by read the cookies directly
With this behaviour, backend will get the token by read the Authorization header.
Finish

Figure 1. Flowchart of Fetch Strategy

The admin login and occupant login has the same design. The login page design can be seen in Figure 2, below.

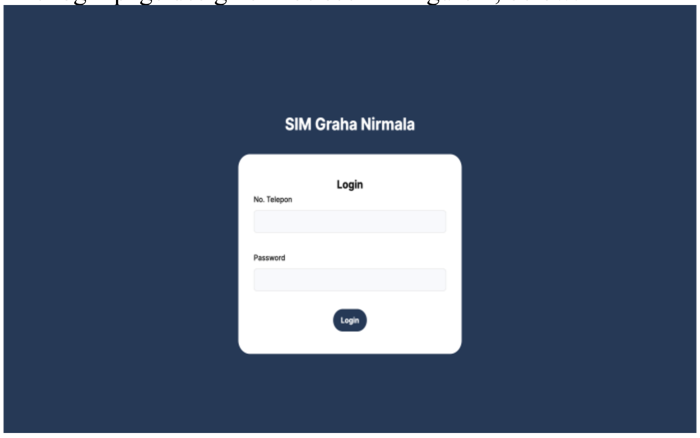


Figure 2. Hi-Fi Design of Login page

As the design drawn above, For the login there are two fields which are phone number and password. In the Coding of Authentication Sprint, for the Authentication for API Routes and Web Pages, in Backend for the token getter, planned strategy above is implemented, the token getter function implementation can be seen on Program Code

```
export function getToken(req: NextRequest): string {
  let token: string | undefined
  let authorization = req.headers.get("authorization")

  if (authorization) {
    token = authorization.split(" ")[1]
  } else {
    token = req.cookies.get("token")?.value
  }

  if (!token) {
    throwUnauthorized()
  }

  return token
}
```

Program Code 1. Token getter function

Token also need to be checked whether it is valid or not. It uses “jsonwebtoken” library to verify the token.

```
export function verifyToken(token: string) {
  try {
    return jwt.verify(token, config.jwt.secret) as Claim
  } catch (error) {
    throwUnauthorized()
  }
}
```

Program Code 2. Verify token function

As we can see on Program Code above, if token cannot be verified, then the system will throw “unauthorized” error.
Then, auth function created that wrap token getter and token verification function in the server. Apart from those two functions, in this auth function, it checks whether the role is valid with the route role definition. The auth function can be seen on Program Code 3.

```
export function useAuth(req: NextRequest, ...roles:
Role[]) {
  const token = getToken(req)
  const claim = verifyToken(token)
  req.headers.set(claimSubHeader, claim.sub)

  if (roles.length === 0) return
  for (const role of roles) if (claim.role === role)
return

  throwUnauthorized()
}
```

Program Code 3. Server authentication function

In the Frontend, to protected route on the frontend, Next.js middleware is used to intercept all the request before the request forwarded. The example of middleware implementation can be seen on Program Code 4.

```
export function middleware(req: NextRequest) {
  …
  if (req.nextUrl.pathname.startsWith("/admin")) {
    const token = req.cookies.get("token")?.value

    if (token) {
      try {
        const { role_type } = parseJwt(token)
        if (role_type === "staff") return
NextResponse.next()
        else if (role_type === "occupant")
          return NextResponse.redirect(new
URL("/app/dashboard", req.url))
      } catch (error) {
        const response = NextResponse.redirect(new
URL("/admin/login", req.url))
        response.cookies.delete("token")
        return response
      }
    }

    return NextResponse.redirect(
      new URL("/admin/login?cause=invalid_session",
req.url),
    )
  }
  …
}
```

Program Code 4. Next.js Middleware implementation

Fetch wrapper created to wrap error handling and all the logic that drawn by Flowchart of Fetch Strategy on Program Code 5.

```
async function handleFetch<T>(
  url: string,
  requestOptions: RequestOptions = { ...requestOptionsDefaults
},
): Promise<[T, FetchError]> {
  try {
    let res: Response
    if (requestOptions.endpointProtected && isServer) {
      const { cookies } = await import("next/headers")
      res = await fetch(`${baseURL}${url}`, {
        headers: {
          Authorization: `Bearer ${cookies().get("token")?.value
?? ""}`,
        },
        cache: "no-store",
        ...requestOptions,
      })
    } else {
      res = await fetch(`${baseURL}${url}`, {
        cache: "no-store",
        ...requestOptions,
      })
    }
    const resJson = await res.json()
    return [resJson.data, null]
  } catch (error) {
    // Error handling
  }
}
```

Program Code 5. Fetch Wrapper Implementation

Testing of Authentication Sprint, Testing in sprint 3 is using Blackbox testing method to assess the functionality of Authentication feature. The Blackbox testing can be seen on table below.

Table 3. Blackbox Testing of Authentication feature

AUTHENTICATION BLACKBOX TESTING

Precondition	Test Case	Test Scenario	Expected Result	Actual result	Summary
-	Login as admin	- No.Telepon (085108510851) - Password (supersecret) - Click “Login” Button	Page directed to default admin page	Page directed to default admin page	Success
-	Login as Occupant	- No.Telepon (085108510851) - Password (supersecret) - Click “Login” Button	Page directed to default occupant page	Page directed to default occupant page	Success
	Logout	Click “Logout” on the sidebar	Page directed to login page	Page directed to login page	Success

3.5 Transaction Sprint

The transaction table has unique implementation since it has filter functionality. And also, custom implementation of cash overview UI component is created. The Transaction feature can be seen on Figure 3. Transaction table and cash information UI are reused in admin page and occupant page. Then the rest of UI components are reuse from reusable design that created on sprint 1.

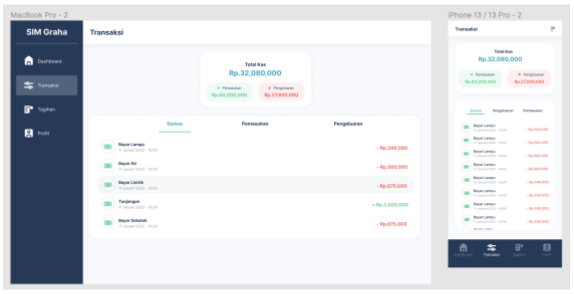


Figure 3. Transaction Table and Cash Overview UI

In the Coding of Transaction Sprint, Backend features only have basic Create, Read, Update, delete (CRUD) operation, that already explain on the sprint 1, while Frontend implementation of the design written in Program Code 6.

```
<>
  <div className="flex justify-evenly">
    <div onClick={() => handleTabClick("all")}>
      Semua
    </div>
    <div onClick={() => handleTabClick("income")}>
      Pemasukan
    </div>
    <div onClick={() => handleTabClick("outcome")}>
      Pengeluaran
    </div>
  </div>
  <div className="flex flex-col gap-5 px-4 sm:px-9 py-6">
    {transactions.length === 0 ? (
      <p>
        Tidak ada transaksi kas
      </p>
    ) : (
      transactions
        .filter((transaction) => {
          if (activeTab === "all") return true

          return transaction.movement === activeTab
        })
        .map((transaction, idx) => {
          return (
            <TransactionListItem />
          )
        })
    )}
  </div>
</>
```

Program Code 6. Transaction table

For each filter buttons, there is “handleClick” function that filter the transaction data based on the flow type. The UI will reflect when the transaction data updated.
Testing in Transaction sprint is using Blackbox testing method to assess the functionality of Transaction Management and Occupant Transaction feature. There are some test cases that can be using reusable test case that defined on sprint 1.
In these features also has some specific test cases. The test cases are specifically for the transaction table view, we need to test the filter button works as the expectation.

Table 4. Sprint Four Blackbox Testing using Reusable Test Cases

BLACKBOX TESTING USING REUSABLE TEST CASES

Test Case	Actual Result	Summary
Go to feature page (Transaction Management, Occupant Transaction)	Page directed to feature page	Success
Go to protected feature page with invalid session (Transaction Management, Occupant Transaction)	Page directed to login page with error message	Success
Add data (transaction)	Page directed back to feature page and show the added data.	Success
Submit data with empty required field (Transaction)	The required fields become red and show the error of the fields.	Success
Edit data (Transaction)	Page directed back to feature page and show the updated data.	Success
Delete data (Transaction)	Page directed back to manage feature page and show the updated data.	Success

Table 5. Specific Transaction Table Blackbox testing

SPECIFIC TRANSACTION TABLE BLACKBOX TESTING

Precondition	Test Case	Test Scenario	Expected Result	Actual result	Summary
In our database there are 2 transactions as described below 1. Transaction with income flow 2. Transaction with outcome flow	See all transactions	1. Go to transaction management page	Transaction table shows all the transactions The first transaction amount is green The second transaction amount is red	Transaction table shows all the transactions The first transaction amount is green The second transaction amount is red	Success
	See income transactions	1. Click “Pemasukan” button on table header	Transaction table shows all the income transactions All the transaction amounts are green	Transaction table shows all the income transactions All the transaction amounts are green	Success
	See outcome transactions	1. Click “Pengeluaran” button on table header	Transaction table shows all the outcome transactions All the transaction amounts are red	Transaction table shows all the outcome transactions All the transaction amounts are red	Success

3.6 Dashboard Sprint

In the Plan of Dashboard Sprint, As the sprint plan, this sprint is feature development of announcement management and occupant dashboard. Desgin of Dashboard Sprint include Flowchart, and Hi-Fi UI Design. There is no specific implementation in terms of functionality. All covered by general coding in sprint 1. In the Testing of Dashboard Sprint, Testing in sprint 5 is using Blackbox testing method to assess the functionality of Announcement Management and Occupant Dashboard features. There are some test cases that can using reusable test case that defined on sprint 1. The Blackbox testing can be seen on Table 6.

Table 6 Sprint Five Blackbox testing using reusable test case

BLACKBOX TESTING USING REUSABLE TEST CASE

Test Case	Actual Result	Summary
Go to feature page (Announcement Management, Occupant)	Page directed to feature page	Success
Go to protected route with invalid session (Announcement Management, Occupant)	Page directed to login page with error message	Success
Add data (announcement)	Page directed back to feature page and show the added data.	Success
Edit data (announcement)	Page directed back to manage feature page and show the updated data.	Success
Delete data (announcement)	Page directed back to manage feature page and show the updated data.	Success

3.7 Bill Sprint

As the sprint plan, this sprint is feature development features of Bill Management and Occupant Bill. This system generated bill every month to each house that can be pay by occupant whether the owner or renter. Thus, we need create a cron job that create billing data into database. Cron job is a scheduling rule for a command that is executed periodically (Setiawan, 2013). For the implementation, Go as the programming language is used. In Go, there is package namely "gocron" which is a Go based cron job package. To handle occupant that prefer offline payment, system need to provide a way to handle it. By give admin ability to change the bill status to be paid can solve the needs. In this sub-title, All the flowcharts, and All the hi-fi UI design covered by reusable design that explain on the sprint 1.

In The Coding of Bill Sprint, at the Backend point, There is two parts of this cron job functionality which are the billing creation logic and worker function.

```
for _, occupant := range occupants {
err := w.db.Conn().NewSelect().
      Model(&entity.Billing{}).
      Where("house_id = ?", occupant.HouseId).
      Where("period >= ?", now.BeginningOfMonth()).
      Where("period <= ?", now.EndOfMonth()).
      Scan(ctx)

billing := &entity.Billing{
      HouseId:     occupant.HouseId,
      Period:      now.BeginningOfMonth(),
      Amount:      10_000,
      IsPaid:      false,
      ExtraCharge: 0,
}
}
```

Program Code 7. Billing creation function

The billing creation logic above is check for every occupant whether there is bill for this period or not. If not, add the billing to the system database. After create the billing creation, we need to run it at a period time as written in program code below.

```
type worker struct {
s  *gocron.Scheduler
db *db.Client
}

func NewWorker(s *gocron.Scheduler, db *db.Client) *worker {
return &worker{s, db}
}

func (w *worker) Do() {
w.s.Every(5).Seconds().Do(w.generateMonthlyBilling)
w.s.StartBlocking()
}
```

Program Code 8. Worker function

The code above state to run the billing creation for every 5 seconds. In the program code also written the "NewWorker" function that called on main function. And The frontend implementation covered by the general coding implementation in sprint 1.

Testing in Bill Sprint is using Blackbox testing method to assess the functionality of Bill Management and Occupant Bill features. There are some test cases that can using reusable test case that defined on sprint 1. The Blackbox testing can be seen on Table 7 and There are also specific test cases for Bill Management feature, we want to check the bill offline payment and the billing creation. The testing can be seen on table below.

Table 7. Sprint Six Blackbox Testing using Reusable Test Cases

BLACKBOX TESTING USING REUSABLE TEST CASE

Test Case	Actual Result	Summary
Go to feature page (Bill Management, Occupant Bill)	Page directed to feature page	Success
Go to protected route with invalid session (Bill Management, Occupant Bill)	Page directed to login page with error message	Success

Table 8. Specific Bill Management Blackbox Testing

SPECIFIC BILL MANAGEMENT BLACKBOX TESTING

Precondition	Test Case	Test Scenario	Expected Result	Actual result	Summary
In database there is a bill that is not paid yet	Bill offline payment	1. Select house 2. Select Occupant 3. Click "Bayar" button	Bill list show empty active bill	Bill list show empty active bill	Success
		4. Confirm	Bill history updated with paid bill	Bill history updated with paid bill	
-	Billing creation	1. Add an occupant 2. After ~5 seconds, go to Bill Management feature	New bill showed	New bill showed	Success

3.8 Payment Sprint

As the sprint plan, this sprint is feature development of Bill Payment via Payment Gateway. In the Design of Payment Sprint, at the Flowchart, As the functional requirement, flowchart drawn which describe how the payment via payment gateway works on the system. As the flowchart above, the system forwards the payment request if there is no pending payment in database. The payment status notified by the payment gateway system to this system.

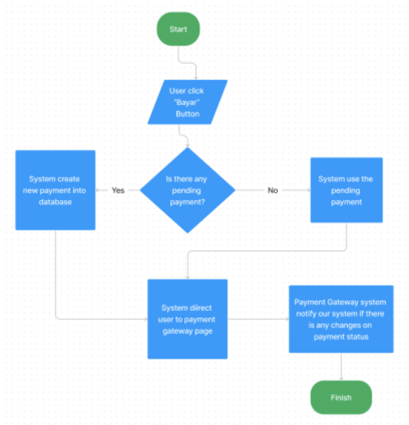

Start
User click "Bayar" Button
Is there any pending payment?
Yes
No
System create new payment into database
System use the pending payment
System direct user to payment gateway page
Payment Gateway system notify our system if there is any changes on payment status
Finish

Figure 4. Flowchart of Payment via Payment Gateway

In the Coding of Payment Sprint, at the Backend, When the occupant triggers the pay button, payment API will create payment session on Midtrans payment gateway and add the payment data into Payment table in the system's database. After the session created, backend will return the redirect URL that will be used as payment page.

```
const res = await snap().createTransaction(parameter)
const payment: TInsertPayment = {
      billingId: billing.id,
      amount: billing.amount,
      payerId: occupant.id,
      invoice: orderId,
      token: res.token,
      status: "pending",
      mode: "transfer",
      expiredAt: expiredAt,
      redirectUrl: res.redirect_url,
}

const [newPayment] = await
db().insert(Payment).values(payment).returning()
```

Program Code 9. Payment creation

The payment session creation using the "snap" library that provided by Midtrans. After handle the payment creation, system need to handle payment gateway callback to sync the payment status.

```
const res = await snap().transaction.status(payment.invoice)
  payment.status = res.transaction_status

  const [updatedPayment] = await db()
    .update(Payment)
    .set(payment)
    .where(eq(Payment.id, payment.id))
    .returning()
  if (updatedPayment.status === "settlement") {
    await db()
      .update(Billing)
      .set({ isPaid: true })
      .where(eq(Billing.id, updatedPayment.billingId))
  }
```

Program Code 10. Payment status callback

As can be seen on program code above, system will update the billing status to paid if the payment status is "settlement". To be able see the settlement payment in transaction, the result of transaction data need to combine database table between the cashflow table and payment table.

```
const results = await db().execute(
    sql`
    SELECT
        cashflow.title,
        cashflow.created_at,
        cashflow.amount,
        cashflow.movement
    FROM
        cashflow
    UNION
    SELECT
        'Pembayaran iuran',
        payment.created_at,
        payment.amount,
        'income'
    FROM
        payment
    WHERE
        payment.status = 'settlement'
    ORDER BY
        created_at desc`,
  )
```

Program Code 11. Combine cashflow table with transaction Table

As the program code shown above, it uses "UNION" to multiple select. It also has a condition where the "payment.status" is equal to "settlement. Testing in sprint 7 is using Blackbox testing method to assess the functionality of Bill Payment via Payment Gateway feature that can be seen on Table 9.

Table 9. Bill Payment via Payment Gateway Blackbox testing

PAYMENT BLACKBOX TESTING

Precondition	Test Case	Test Scenario	Expected Result	Actual result	Summary
For development purpose, this test case using Midtrans sandbox environment to mock the payment There is billing on the occupant account	Pay occupant's bill	1. Click "Bayar" button 2. Select BCA as payment method 3. Pay with the BCA virtual account simulator that provided by Midtrans	Bill status change to be paid.	Bill status change to be paid.	Success

4. CONCLUSION

This research successfully developed a housing management information system. In terms of functional testing with Blackbox testing, all the test cases that tested in every sprint are success, it means the system works as expected according to its functionality. This system able to manage housing data, ensure data type of each column, generate bill every month, and handle payment with automatic verification checking. It tackles pain points of previous system and accommodates the housing needs.

5. REFERENCE

[1] Alviana, S., & Kurniawan, B. (2021). Penerapan Sistem Informasi Iuran Warga Griya Pataruman Asri Berbasis Website. *Jurnal Pengabdian Masyarakat Indonesia*, 1(6), 343-350.

[2] Gunawan, R. (2018). Perancangan Dan Implementasi Sistem Informasi Manajemen Iuran Anggota Di Pb. Perkemi. *Infotronik: Jurnal Teknologi Informasi dan Elektronika*, 3(2), 76-84.

[3] Kartiningrum, E. D. (2015). *Panduan Penyusunan Studi Literatur*. Mojokerto: LPPM Poltekkes Majapahit.

[4] Maulana, A., Sadikin, M., & Izzuddin, A. (2018). Implementasi Sistem Informasi Manajemen Inventaris Berbasis Web Di Pusat Teknologi Informasi Dan

Komunikasi-BPPT. *Setrum: Sistem Kendali-Tenaga-Elektronika-Telekomunikasi-Komputer*, 7(1), 182-196.

[5] Mukhtar, M. (2017). SISTEM INFORMASI IURAN KEAMANAN WARGA RW. 04 KEL. TAMPAN KEC. PAYUNG SEKAKI BERBASIS WEB. *Jurnal Intra Tech*, 1(2), 10-17.

[6] Mulyati, S. (2020). (Sistem Pembayaran dan Manajemen Tagihan Iuran Perumahan Berbasis Web) (Studi kasus: Desa Bumiharjo, Moilong, Banggai, Sulawesi Tengah).

[7] Purnama, C., & SE, M. (2016). Sistem Informasi Manajemen. Mojokerto: Insan Global.

[8] Rosyida, E. (2020). *Sistem Informasi Manajemen*. Banyuwangi.

[9] Siregar, W., Irvan, I., & Rahayu, E. (2020). Sistem Informasi Pembayaran Iuran Keamanan Dan Kebersihan Pada Perumahan Berbasis Website Menggunakan Metode Design Thinking. *JITEKH*, *8*(2), 50-58.

[10] Wahyudi, A., Sowiyah, S., & Ambarita, A. (2015). Implementasi Sistem Informasi Manajemen Akademik Berbasis Web. *Jurnal Manajemen Mutu Pendidikan*, 3(1), 2-6.

[11] Nidhra, S., & Dondeti, J. (2012). Black box and white box testing techniques-a literature review. *International Journal of Embedded Systems and Applications (IJESA)*, 2(2), 29-50.

[12] Nugroho, F. E. (2016). Perancangan Sistem Informasi Penjualan Online Studi Kasus Tokoku. *Simetris: Jurnal Teknik Mesin, Elektro Dan Ilmu Komputer*, 7(2), 717-724.

[13] Raharjana, I. K. (2017). *Pengembangan Sistem Informasi Menggunakan Metodologi Agile*. Deepublish.

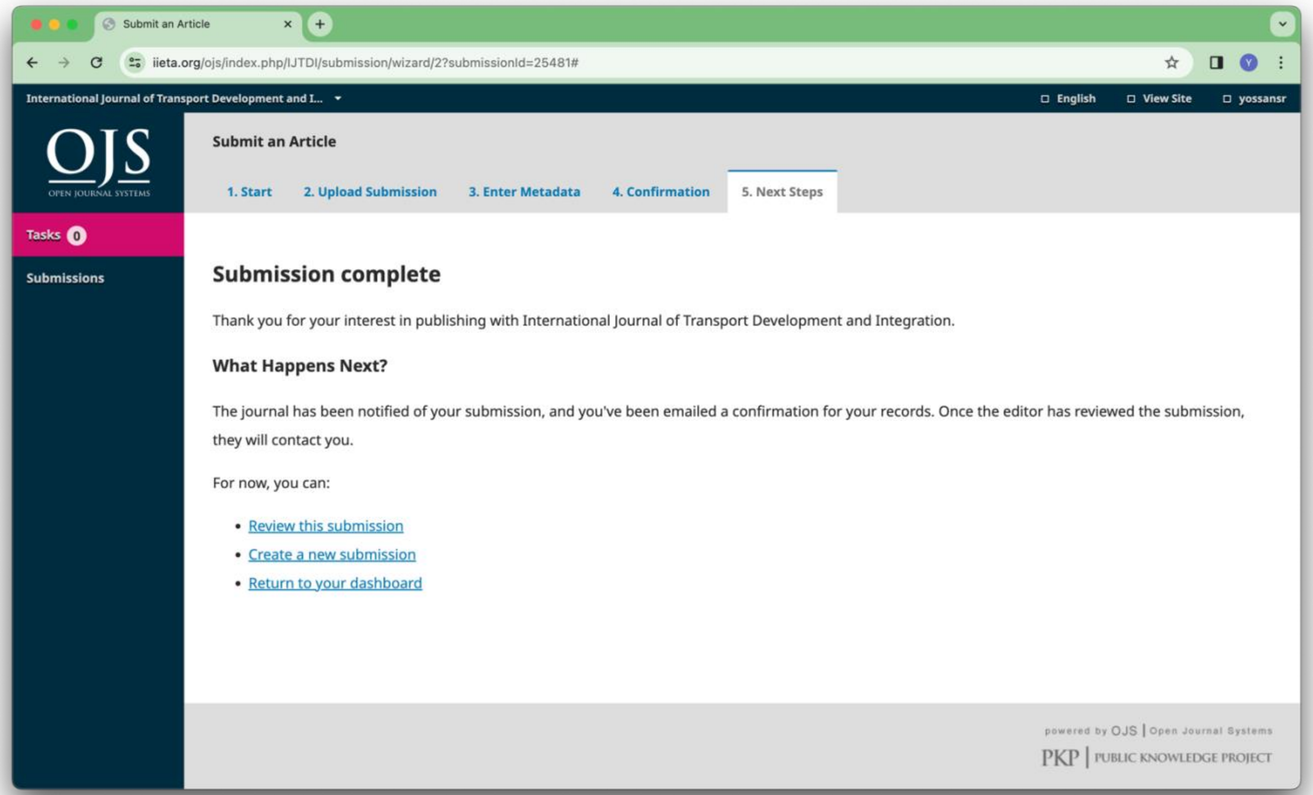
[14] Shore, J., & Warden, S. (2021). *The art of agile development*. " O'Reilly Media, Inc.".

[15] Setiawan, A. (2013). *Rancang Bangun Sistem Monitoring Ruangan Menggunakan Webcam Berbasis OpenWrt* (Doctoral dissertation, UIN Sunan Kalijaga).

[16] Puspitasari, T. M. M., & Maulina, D. (2019). Implementasi payment gateway menggunakan midtrans pada marketplace travnesia. com. *Mobile and Forensics*, 1(1), 22-29.

[17] Rosaliza, M. (2015). Wawancara, Sebuah interaksi komunikasi dalam penelitian kualitatif. *Jurnal Ilmu Budaya*, 11(2), 71-79.

[18] Ismail, A., & Ali, S. M. Agile software development: Implementation perspective Agile software development: Implementation perspective.


Submit an Article
iieta.org/ojs/index.php/IJTDI/submission/wizard/2?submissionId=25481#
International Journal of Transport Development and I...
English
View Site
yossansr
OJS
OPEN JOURNAL SYSTEMS
Tasks 0
Submissions

Submit an Article

1. Start | 2. Upload Submission | 3. Enter Metadata | 4. Confirmation | 5. Next Steps

Submission complete

Thank you for your interest in publishing with International Journal of Transport Development and Integration.

What Happens Next?

The journal has been notified of your submission, and you've been emailed a confirmation for your records. Once the editor has reviewed the submission, they will contact you.

For now, you can:

- Review this submission
- Create a new submission
- Return to your dashboard

powered by OJS | Open Journal Systems
PKP | PUBLIC KNOWLEDGE PROJECT