



Business Application Development Final Report
ParkingKu

Class: LH11

Group: 4

Major: Sistem Informasi

Campus: Alam Sutera

Member:

2401962384 - Devin Revel

2440033786 - Luis Jordy

2440053844 - Christiansen Julian Wijaya

2440010366 - Steven Taffanoto

2440066292 - Goeneari Haqim

2440045930 - Fellik Fernando

2440097234 - Felita Ailsa Nadhira

2440020436 - Marcello

2440005946 - Eunike Charista

2440015820 - Fayola

2440087416 - Evan Veda

Lecturer: Cahya Lukito, S.T., M.T. - D2950

UNIVERSITAS BINA NUSANTARA

2021

TABLE OF CONTENTS

| | |
|--|-----------|
| TABLE OF CONTENTS | 2 |
| CHAPTER I - PRELIMINARY | 3 |
| Background and Objective | 3 |
| Roles and Responsibilities | 3 |
| Timeline | 4 |
| CHAPTER II - BUSINESS APPLICATION | 5 |
| Use Case Diagram | 5 |
| Activity Diagram | 5 |
| GUI Design | 6 |
| DB Design | 7 |
| Coding explanation for DB Connectivity and GUI | 7 |
| Question and Answer | 13 |
| CHAPTER III - CONCLUSION | 15 |

CHAPTER I - PRELIMINARY

A. Background and Objective

With the growth of populations, malls are one of the popular destinations in large cities especially in Jabodetabek. Most people are having difficulty in finding a parking spot because there are limited spaces available, causing people to park wherever they can find a space or they might not be able to park their car. As a result, we came up with the idea to create an online-based parking application that allows customers to find parking spots easily by just making a reservation with their phone.

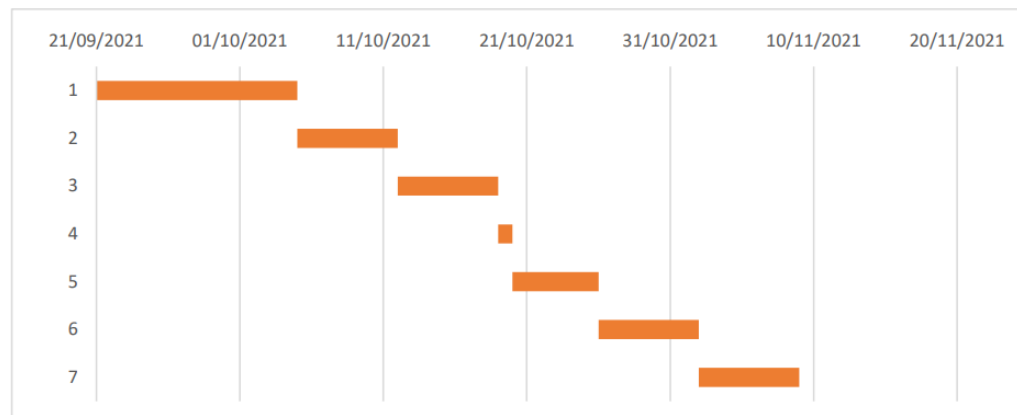
This project is focused on developing an application for reserving parking spots in crowded areas, specifically malls. The application is named Parking-Ku, and its main goal is to book parking spaces before you enter the mall as a visitor. All you have to do is download the application, create an account, log in, and then select the available parking space on the app's screen.

B. Roles and Responsibilities

| Roles | Name | Responsibilities |
|--------------|--|--|
| Leader | Devin Revel | takes responsibility in managing the whole project process, especially ensuring that the team member does every task completely and ensuring the process in doing the project on track with the schedule |
| Presenter | Steven Taffianoto Evan Veda | Presenting application and demo of the application |
| Coder | Fellik Fernando Devin Revel Fayola Marcello | Create application code Connect application to database |
| Designer | Christiansen Julian Wijaya Eunike Charista Felita Ailsa Nadhira | Create wireframe as a raw design of application interface Create & design powerpoint for presentation material Involve in the making of final report |
| Tester | Lois Jordy Goeneari Haqim | Create test case Involve in the making of final report |

C. Timeline

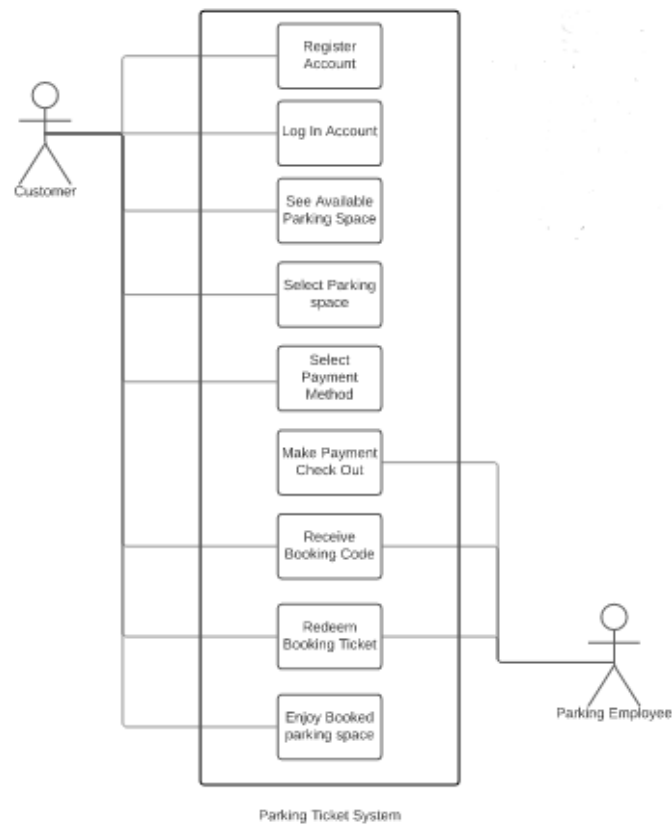
| Tasks | | Start Date | End Date | Duration (days) |
|-----------|-------------------------------------|------------|------------|-----------------|
| Session 1 | Decide Who doing What | 21/09/2021 | 04/10/2021 | 14 |
| Session 2 | Use Case Discovery | 05/10/2021 | 11/10/2021 | 7 |
| Session 3 | Use Case and Activity Diagram | 12/10/2021 | 18/10/2021 | 7 |
| Session 4 | Sprint #1 (First presentation) | 19/10/2021 | 19/10/2021 | 1 |
| Session 5 | Sprint #2 (Develop code and design) | 20/10/2021 | 25/10/2021 | 6 |
| Session 6 | Sprint #3 (Develop code and design) | 26/10/2021 | 01/11/2021 | 7 |
| Session 7 | Sprint #4 (Develop code and design) | 02/11/2021 | 08/11/2021 | 7 |



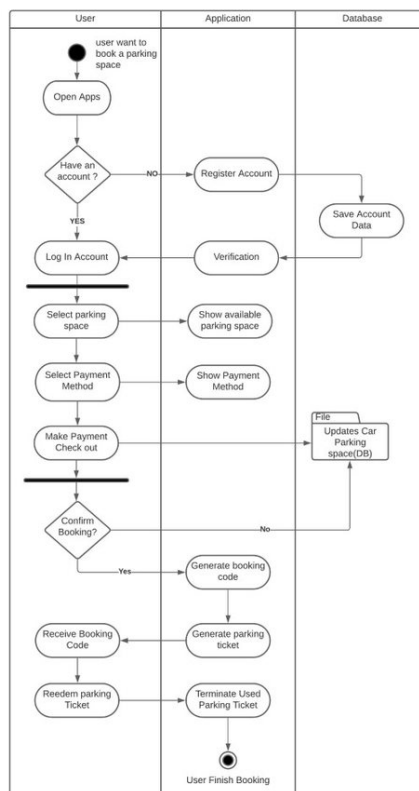
The work timeline is being followed in accordance with the information in the table above. We also had multiple meetings outside of class to talk about the project, the process and history can be seen on the Ms Teams.

CHAPTER II - BUSINESS APPLICATION

A. Use Case Diagram



B. Activity Diagram



C. GUI Design

A login and registration screen for 'PARKING-KU'. It features a blue header with the title. Below it, a blue box contains 'Username' and 'Password' labels with corresponding input fields. The username field contains 'FF' and the password field contains '....'. A 'LOG IN' button is at the bottom of this box. Below the box is a 'REGISTER' button.

A registration form for 'ParkingKU'. It has a light gray background and a blue header. The form includes fields for 'Name' (filled with 'Felik Fernando'), 'Username' (filled with 'FF'), 'Password' (filled with '....'), and 'Confirm Password' (filled with '....'). A 'Register' button is at the bottom.

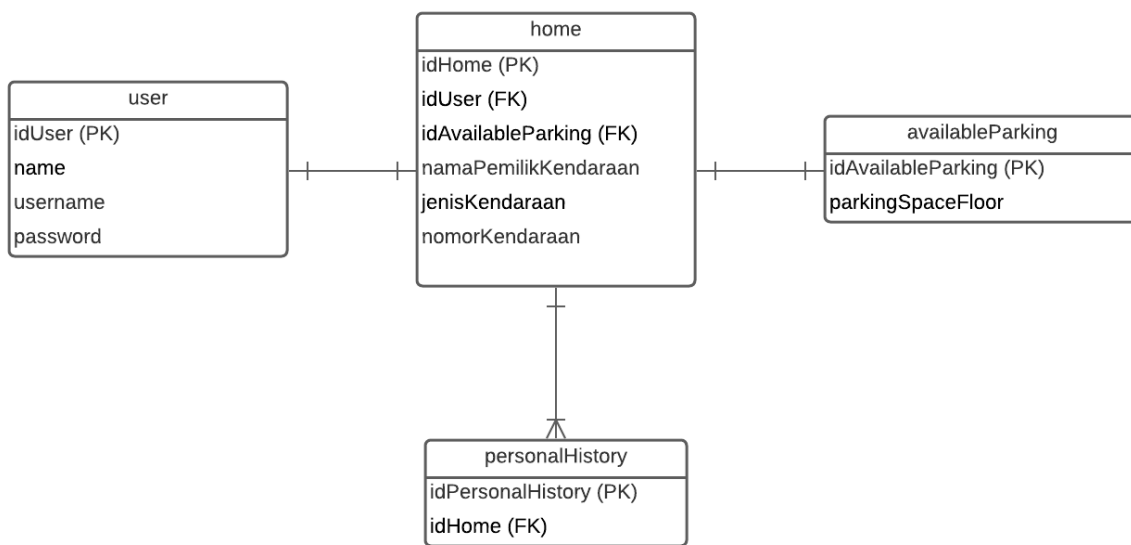
A vehicle registration form for 'ParkingKU'. It has a blue header and a light gray background. The form includes fields for 'Nama Pemilik Kendaraan' (filled with 'Felik Fernando'), 'Jenis Kendaraan' (filled with 'Mobil'), and 'Nomor Kendaraan' (filled with 'B 2000 OC'). A 'NEXT' button is at the bottom.

A screen showing available parking spots for 'ParkingKU'. It has a blue header with the title and subtitle 'Parking Available'. Below the header, there is a 3x3 grid of buttons labeled '1 - A', '1 - B', '1 - C', '2 - A', '2 - B', '2 - C', '3 - A', '3 - B', and '3 - C'. A mouse cursor is hovering over the '2 - B' button.

A confirmation screen for 'ParkingKU'. It has a light gray background and a blue header. The screen displays '2B' in large text, followed by 'Mobil' and 'Felik Fernando'. Below that is 'B 2000 OC'. At the bottom, it says 'Terimakasih sudah menggunakan layanan kami' and has a 'DONE' button.

A screenshot of a MySQL database structure diagram. It shows two tables: 'parkingku.account' and 'parkingku.historyparking'. The 'account' table has columns: username (varchar(30)), name (varchar(50)), and password (varchar(20)). The 'historyparking' table has columns: NoKendaraan (varchar(20)), Username (varchar(20)), Nama (varchar(30)), Floor (varchar(5)), and JenisKendaraan (varchar(10)).

D. DB Design



E. Coding explanation for DB Connectivity and GUI

```
19
20
21 public Connect() {
22     try {
23         Class.forName("com.mysql.jdbc.Driver");
24         con = DriverManager.getConnection(
25             ("jdbc:mysql://localhost:3306/ParkingDB", "root", ""));
26         stat = con.createStatement();
27     } catch (Exception e) {
28         // TODO Auto-generated catch block
29         e.printStackTrace();
30     }
31 }
32
33 public ResultSet execQuery(String query){
34     try {
35         rs = stat.executeQuery(query);
36         rsm = rs.getMetaData();
37     } catch (Exception e) {
38         // TODO Auto-generated catch block
39         e.printStackTrace();
40     }
41     return rs;
42 }
43
```

To connect Java application with the MySQL database, we need to prepare

1. The driver: enable java applications to load and interact with database.
Interface from `java.sql.Driver` and the driver class syntax used for the mysql database is `com.mysql.jdbc.Driver`.
2. Connect to the database using interface connection.
Syntax using `con = DriverManager.getConnection(URL);`
The connection URL for the mysql database is `jdbc:mysql://localhost:3306/dbname`.
3. Make SQL statement for accessing database using syntax: `stat.conCreateStatement();`
4. Lastly, `ResultSet` that takes part in execution of SQL statement using syntax: `rs = stat.executeQuery(String query);`

To get execution result, we use method on ResultSet: rs.getMetaData();

5. The system will check whether the password and confirm password have the same input value during registration. If this is the case, use the statement object to execute the query when inserting new registers data.

```
}else if(InputPassword.getText().equals(InputConfirmPassword.getText())) {  
    name = textField_name.getText();  
    username = textField_Username.getText();  
    password = InputPassword.getText();  
    try {  
        String query = "INSERT INTO account(username, name, password) VALUES('%s', '%s', '%s')";  
        query = String.format(query, username, name, password);  
        con.stat.execute(query);  
    } catch (SQLException b) {  
        // TODO Auto-generated catch block  
        b.printStackTrace();  
    }  
}
```

6. For the login page, you only need to enter the username and password that were previously saved, which must be the same as the one entered by the user and will be checked in the database.

```
try {  
    String usernameTemp = nametext.getText();  
    String passwordTemp = new String(passwordField.getText());  
    if (usernameTemp.isEmpty() || passwordTemp.isEmpty()) {  
        JOptionPane.showMessageDialog(this, "username and Password cannot be Empty", "Message",  
            JOptionPane.INFORMATION_MESSAGE);  
        return;  
    }  
    boolean usernameIsFound = false;  
    boolean passwordIsFound = false;  
    String queryusername = "SELECT username FROM account";  
    con.rs = con.stat.executeQuery(queryusername);  
    while (con.rs.next()) {  
        String username = con.rs.getString("username");  
        if (usernameTemp.equals(username)) {  
            usernameIsFound = true;  
            break;  
        }  
    }  
    if (usernameIsFound) {  
        String queryPassword = "SELECT password FROM account WHERE username='%s'";  
        queryPassword = String.format(queryPassword, usernameTemp);  
        con.rs = con.stat.executeQuery(queryPassword);  
        con.rs.next();  
        String password = con.rs.getString("password");  
        if (passwordTemp.equals(password)) {  
            passwordIsFound = true;  
        }  
    }  
}
```



```

43
44 public void insertIntoHistoryParking(String txtNomorKendaraan, String username, String nametxt,
45     String jenisKendaraan, String floor ) {
46     try {
47         pStat = (PreparedStatement) con.prepareStatement("INSERT INTO historyparking VALUES(?, ?, ?, ?, ?)");
48         pStat.setString(1, txtNomorKendaraan);
49         pStat.setString(2, username);
50         pStat.setString(3, nametxt);
51         pStat.setString(4, floor);
52         pStat.setString(5, jenisKendaraan);
53         pStat.execute();
54     } catch (SQLException e) {
55         e.printStackTrace();
56     }

```

Then, create a PreparedStatement object to insert query into history parking that takes five input parameters - function: for sending SQL statements to the database.

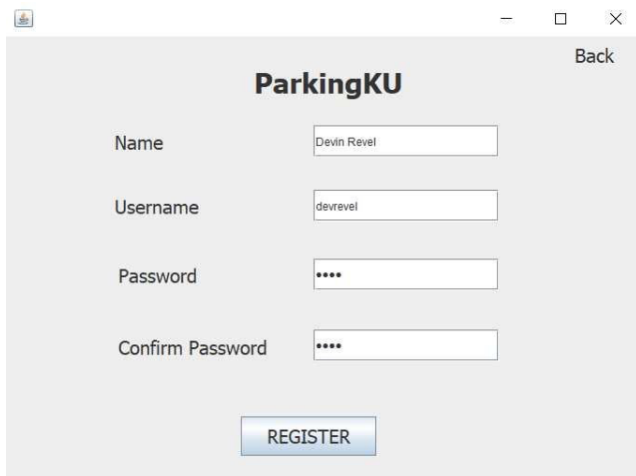
Before executing a PreparedStatement object, we must provide values in place of the question mark placeholders. Use a few of the setter methods defined in the PreparedStatement class to accomplish this.

The top screenshot shows the login page of the PARKING-KU application. It has a title bar with a maximize button and a close button. The main area has a blue header with the text 'PARKING-KU'. Below the header, there are two input fields: 'Username' with the text 'devrevel' and 'Password' with masked characters '.....'. Below the password field is a 'LOGIN' button. At the bottom of the window is a 'REGISTER' button.

The bottom screenshot shows the same login page, but with a pop-up message box. The message box has a title bar with a close button and contains the text 'Welcome to ParkingKU' with an 'OK' button.

When opening the application, the user will be directed to the login page to fill in the username/password that was previously created. If the user has already created a username/password, then he/she can fill in the username/password field. If not, the user can press the register button at the bottom of the page.

If the username and password are correct, a pop-up message which is shown in the image above will appear.



A screenshot of a web browser window displaying the "ParkingKU" registration page. The page has a light gray background. At the top right, there is a "Back" link. The form contains four input fields: "Name" (filled with "Devin Revel"), "Username" (filled with "devrevel"), "Password" (filled with four asterisks), and "Confirm Password" (filled with four asterisks). Below the fields is a blue "REGISTER" button.

On the register page, users can create a ParkingKu account. User must fill in their name, username, password, and confirm password. After all the data has been filled in, the user can press register to create an account and will be returned to the login page. If the user does not create an account, then the user can press the word "Back" in the upper right corner to return to the login menu.



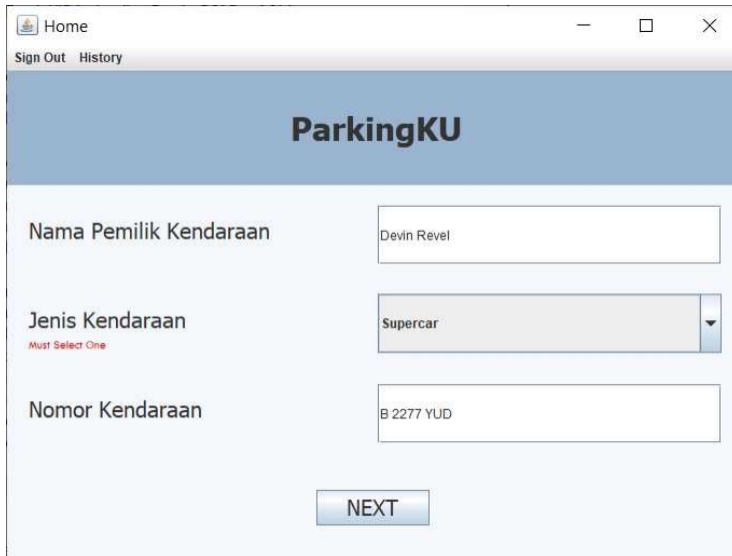
A screenshot of the "ParkingKU" registration page. A modal message box is displayed in the center, titled "Message" with a close button (X). The message contains an information icon (i) and the text "Password Did not Match", followed by an "OK" button. The background form shows the "Name" field filled with "Devin Revel", the "Username" field filled with "devrevel", the "Password" field filled with four asterisks, and the "Confirm Password" field filled with five asterisks. The "REGISTER" button is visible at the bottom.

If the confirm password field does not match the password that is set, then a warning like this will appear.



A screenshot of the "ParkingKU" registration page. A modal message box is displayed in the center, titled "Message" with a close button (X). The message contains an information icon (i) and the text "Data Cannot be Empty", followed by an "OK" button. The background form shows the "Name" field empty, the "Username" field filled with "devrevel", the "Password" field filled with four asterisks, and the "Confirm Password" field filled with five asterisks. The "REGISTER" button is visible at the bottom.

In addition, if one of the 4 data is empty, a pop-up message like the image above will appear.



Home Sign Out History

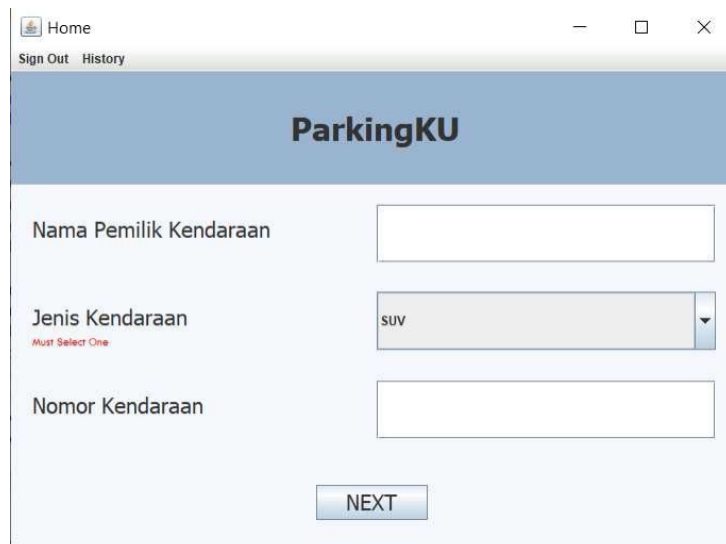
ParkingKU

Nama Pemilik Kendaraan

Jenis Kendaraan Must Select One

Nomor Kendaraan

After the user has successfully logged in, the user will be directed to the ParkingKu application home page. On this page the user must fill in the name of the vehicle owner, vehicle number, and vehicle type. In addition, users can also select the “History” tab to view parking history.



Home Sign Out History

ParkingKU

Nama Pemilik Kendaraan

Jenis Kendaraan Must Select One

Nomor Kendaraan

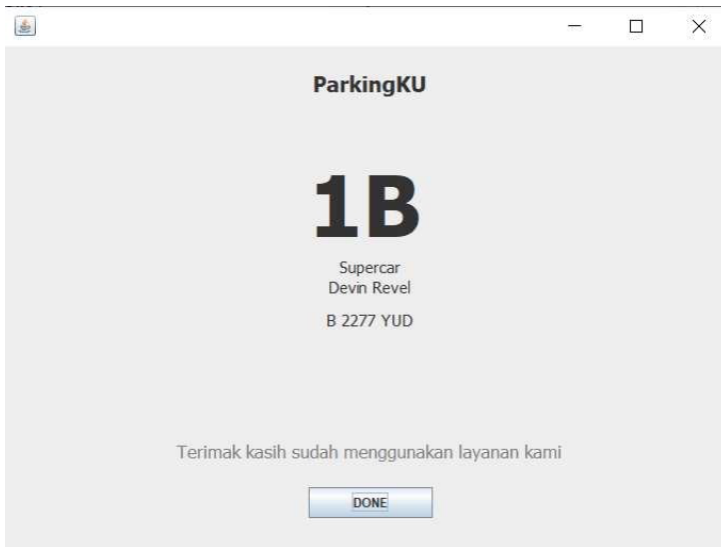
If the user does not fill in one of the three required data, a red warning like the image above will appear.



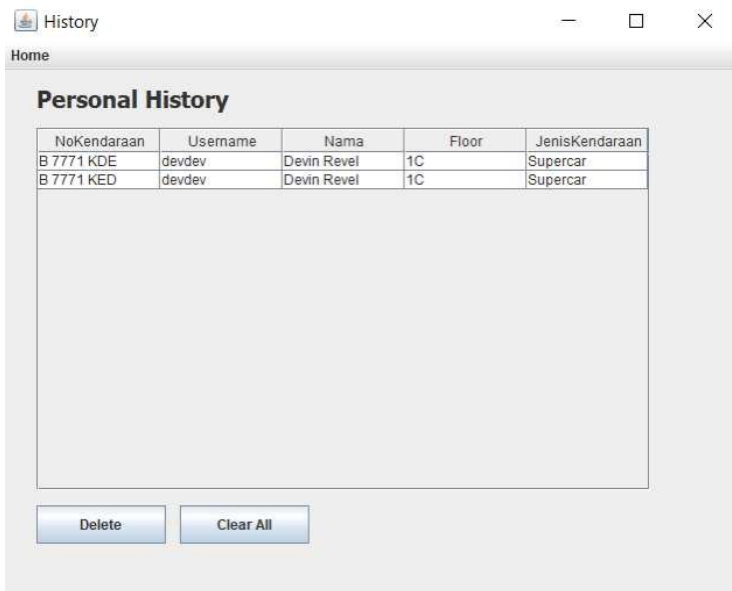
ParkingKu
Parking Available

| | | |
|-------|-------|-------|
| 1 - A | 1 - B | 1 - C |
| 2 - A | 2 - B | 2 - C |
| 3 - A | 3 - B | 3 - C |

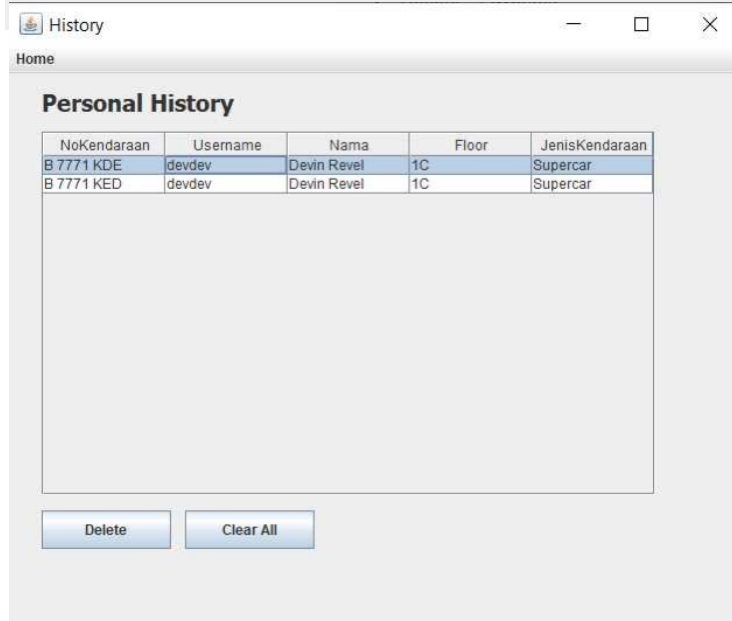
After filling in the data, the user will be directed to the parking page which shows the parking space that is available. On this page, the user must choose one of the available parking spaces.



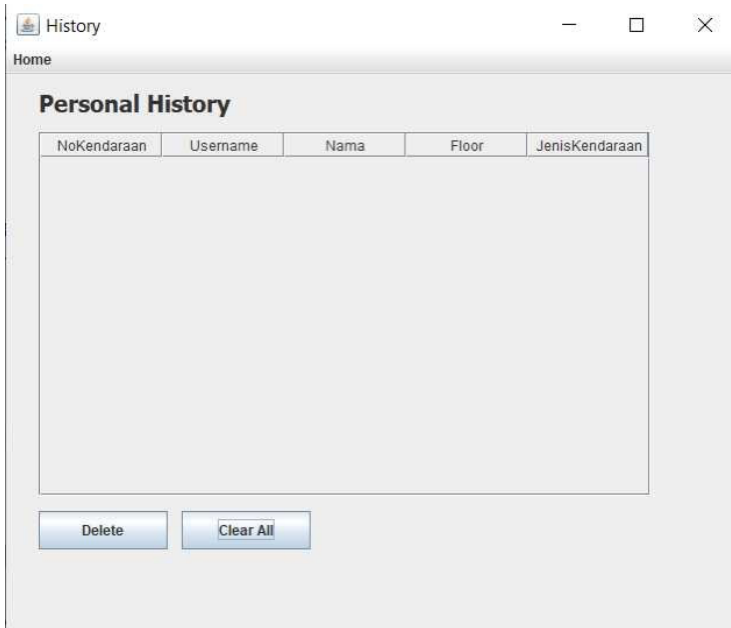
After selecting the parking space you want to choose, a confirmation screen like the image above will appear. Then, the user can press the "Done" button at the very bottom of the page and the user will be redirected back to the home page.



On the history page, users can see parking transactions that have occurred previously. On this page there are two features, namely "Delete" and "Clear all".



Users can select the data they want to delete by pressing the mouse on the data they want to delete. After that, all rows of that data will be highlighted in blue. The user can then press the delete button and the selected row will be deleted from the history page and database.



Users can delete all data by pressing the "Clear" button. After that, all data in the History page and database will be deleted.

F. Question and Answer

Based on the presentation, we collected the questions and answers as follows.

1. How much income can be obtained from launching this application?

Answers:

According to several related parties and those who have experience with this kind of parking reservation application, the income that we can get starts from 20 million in one month to 700 million. It depends on how large the mall with which the application has collaborated, how much of the population is usually in the mall, and also how advanced technology is in an area that results in people being sophisticated in using technology. The greater the probability, the greater the possibility for greater income.

2. What is the use of the history feature that is connected to the database?

Answers:

In terms of security, the history feature can be a strategy for detecting fraud and auditing. An unwanted event that can be avoided is allocating two vehicles to the same parking number. It is because, if the system does not have a history of parking locations that have been ordered by the user, the system will still display all the parking space to other including the number of parking space that has been ordered and has not been ordered before, so that it has a big possibility for duplicate orders to occur. In addition, the feature can be used to analyze how many orders are made in a certain period of time for bookkeeping and auditing purposes.

3. What kind of developments can be made to the ParkingKu application in the future?

Answers:

Applications can be developed to be more complex, with payment features using various methods and further detailed information. In addition, cooperating with e-money companies can be a solution in making it easier for customers to make parking booking payments. Also, in increasing revenue and expanding business, it is possible to establish cooperation with other malls in Indonesia.

In terms of IoT, applications can be developed such as connecting the user's internet to the sensor lights at the parking location to show a green parking lot to make it easier for customers to find a parking space. Another development that can be made is in redeeming parking tickets, which customers can scan the QR Code and pay using the scanned QR Code that has been displayed after ordering tickets. For example, after reserving a parking space, the customer will get a digital parking ticket, including a QR Code that will be on the ticket. When they want to enter the mall to park their vehicle, the customer scans the QR Code on the ticket, so that the system in the application and also the system at the mall will track the time when they enter the mall. However, same thing with entering the mall process, when they want to leave the mall, they also scan the QR Code on the scanner to pay parking fees on the application according to the history of how long they have parked in the mall.

CHAPTER III - CONCLUSION

In Conclusion, our team has managed to create an Application that can help users to order/book parking space that will be useful in the Mall. However, the application we made isn't fully developed. Why? Because we only focused on specific features that we mention in the use case. Furthermore, our group members have made a great contribution and do their best in finishing their task throughout the project.