1

**UNIVERSITY OF RWANDA COLLEGE OF BUSINESS AND ECONOMICS**

**SCHOOL OF BUSINESS**



**BIT DEPARTMENT**

**LEVEL 2**

**GROUP 1**

**COURSE: SYSTEMS ENGINEERING, DATABASE AND JAVA PROGRAMMING**

**PROJECT PROPOSAL ON**

**USER INFORMATION DATABASE & PARKING TICKET INFORMATION SYSTEM**

**Names: IRAGUHA Aristide Registration number: 222008128**

**Submitted to: Dr BUGINGO Emmanuel (Lecturer)**

TABLE OF CONTENT:

CHAPTER1: SYSTEM ANALYSES 3

* 1. INTRODUCTION 3
  2. [PROBLEM STATEMENT 4](#_TOC_250008)
  3. GENERAL OBJECTIVE 4
  4. FUNDAMENTAL REQUIREMENTS 5
  5. [NON-FUNCTIONAL REQUIREMENTS 6](#_TOC_250007)
  6. [FEASIBILITY STUDY 7](#_TOC_250006)
  7. [DATA FLOW DIAGRAM (DFD) 8](#_TOC_250005)
     1. [DATA FLOW DIAGRAM LEVEL 0 8](#_TOC_250004)
     2. [DATA FLOW DIAGRAM LEVEL 1 9](#_TOC_250003)

1.10. CONCLUSSION 14

CHAPTER2: DATABASE DESIGN ... 15

* 1. INTRODUCTION 15
  2. SECTION1: 16
     1. ENTITIES 16
     2. ENTITY RELETIONSHIP DIAGRAM 22
  3. SECTION2:SQL 24
  4. SECTION3 26
     1. VIEWS 26
  5. CONCLUSION 39

CHAPTER3: JAVA PROGRAMMING 40

* 1. INTRODUCTION 40
  2. [TOOLS USED TO DEVELOP THIS SYSTEM IN JAVA PROGRAMMING 40](#_TOC_250001)
  3. [FORMS DESCRIPTION 41](#_TOC_250000)
  4. CONCLUSION 52

**TOPIC:**User information database and parking ticket information system



* 1. **INTRODUCTION**

**The parking ticket payment management system** is an information system designed to streamline

And automate the process of paying parking tickets. The system aims to provide a convenient

And user-friendlyplatform for individuals to pay their parking fines online, reducing the need

For manual paper-based processes and improving efficiency for both users and administrative staff.

The manual system for parking ticket payment management was burdened by manual processes, limited payment options, delays, and a lack of real-time information. It also lacked user-friendly features, multilingual support, and robust reporting capabilities, making it less efficient and inconvenient for both individuals and administrative staff.

Our system will provide a user-friendly interface for individuals to access their parking ticket information, make online payments conveniently, and submit appeals if necessary. It will ensure real-time updates, automated reminders, and robust reporting capabilities, making the entire process efficient, transparent, and accessible for both users and administrative staff.

## Problem statement

Today the manual system has become a pain for HR department as well as for the finance team as it need a lot of time, work and effort to make it successful every month. Here we present different problems of using manual payroll system which cost **The parking ticket payment management system:**

* + 1. Manual and inconvenient process
    2. Limited payment methods
    3. Limited real-time information
    4. Delays and inefficiency

## Solution

The proposed Parking Ticket Payment Management System will address the shortcomings of the existing system by introducing an automated and user-friendly platform

## 1.4 General Objective

the proposed system will provide a user-friendly interface for individuals to access their parking ticket information, make online payments conveniently, and submit appeals if necessary. It will ensure real-time updates, automated reminders, and robust reporting capabilities, making the entire process efficient, transparent, and accessible for both users and administrative staff.

## Specific objective

The overall objective of the Parking Ticket Payment Management System aims to simplify and streamline the process of paying parking fines, improving user experience and administrative efficiency while promoting compliance and transparency.

## Fundamental requirements Available technology to be used:

Languages:

1. HTML: as our system is web based we will need hypertext markup language to develop web based codes.
2. PHP and JavaScript: for assisting HTML developed code.
3. Java programming language: by mean of java IDE it will be used to create interfaces and logical connection to database.

Web server:

1. XAMPP server: for solution stack for Microsoft window operating system supporting MySQL database and PHP programming languages.

Development platform:

* 1. ECLIPSE IDE: for quickie creation and publish web pages for HTML, CSS and JavaScript.

## Available tools to be used:

This project will require tools like: editor which is ECLIPSE for both PHP, XAMPP server for MySQL, also it will require operating system which is window 10.

**Available hardware to be used: Processor: Lenovo i5**

RAM: 2GB

Hard disk: 700GB

## users of this system:

1. A system administrator
2. User / Customers

## Non-functional requirements

**Usability:** The system should have a user-friendly interface, providing a seamless and intuitive experience for users.

**Security:** The system should implement robust security measures to protect user data and ensure secure payment processing.

**Performance:** The system should be capable of handling a large number of users and transactions efficiently, with minimal response times.

**Accessibility:** The system should be accessible to users with disabilities, complying with accessibility guidelines and supporting features such as screen reader compatibility.

**Scalability:** The system should be designed to accommodate future growth and increased user demands without significant performance degradation.

**Integration:** The system should be capable of integrating with existing parking enforcement systems, payment gateways, and other relevant systems for seamless data exchange.

**Privacy:** The system should adhere to privacy regulations and protect user information from unauthorized access or misuse.

# FEASIBILITY STUDY

After identifying the scope of the project, the feasibility study is needed to be carried out. It is basically keeping the following points in mind.

**Building the software for meeting the scope:** This software has met the scope. As there is no data involved in the system, processing on the file, and the behavior of this project is already identified and bundled in quantitative manner. The processing of this software is very simple as it has been designed in php and it has been well divided into several functions according to the need.

**Technically feasible:** This software is very much technically feasible. This software is very much concerned with specifying equipment and the software will successfully satisfy almost all the admin’s requirements. The technical need for this system may vary considerably but might include:

System Architecture

Technology Stack

Security

Infrastructure

Development Resources

Maintenance and Support

c. Ability to process data at a particular speed.

Therefore, the basic input/output of data is identified. So, the project can easily be build up and it will also be technically feasible.

**State of Art:** The project is very much within the state of art since the project is a WINDOWS based; it uses very modern and common technique. Beside it is very much modern and user friendly. It also works as middleware i.e. only in between the user and the file. So, it is completely a state of art project.

**Financially Feasible:** The project is very much financially feasible. The implementation and development cost of this software under the reach of any college.

Moreover, it requires some training for the use. So, training cost can be neglected and the resources of this software are very much available. It also reduces the labor and extra cost to be paid for labor. So indeed, it is financially feasible.

**Resources:** As motioned earlier that the resources are easily available and the cost of training is almost negligible. Sometimes situations may arise when it may not be so much easy. For a person completely unaware of using a computer system could result in a training cost or for a very small organization the purchase of a computer, instalment of the system and other charges may lead to a difficult matter.

# DATA FLOW DIAGRAM (DFD)

A data-flow diagram is a way of representing a flow of data through a process or a system. The DFD also provides information about the outputs and inputs of each entity and the process itself

# DATA FLOW DIAGRAM LEVEL 0

In this diagram:

The User interacts with the Parking Ticket System by sending requests, such as viewing ticket information, making payments, or submitting appeals.

The Parking Ticket System processes user requests and interacts with the Payment Gateway to handle payment transactions securely.

The Payment Gateway facilitates payment processing and communicates the transaction status back to the Parking Ticket System, which in turn informs the User about the payment outcome.

***Figure that present level0 of User information database and The parking ticket payment management system***

******

# DATA FLOW DIAGRAM LEVEL 1

In this diagram:

The User interacts with the Parking Ticket System by sending requests for various actions.

The Parking Ticket System processes user requests and interacts with the Database to retrieve or store ticket information, user data, payment records, etc.

The Database serves as the central repository for storing and retrieving data for the Parking Ticket System and Administrator.

The Payment Gateway communicates transaction details with the Parking Ticket System for payment processing.

The Administrator interacts with the Parking Ticket System to manage system settings, user accounts, ticket data, and other administrative tasks.



## References:

1. System Engineering and Analysis third edition (Benjamin A & Walter j Fabrycky, 2013)
2. Agile System Engineering (Bruce Powel Douglass, 2022)
3. [www.incose.org>s](http://www.incose.org/)ystem-engineering



**DATA BASE OF THE SYSTEM 2.1INTRODUCTION:**

In this chapter we will be describing database of the system tables inside that database and the way those tables were created in, addition on that in this chapter we will show table views from original tables, the relationship between created tables that are in this database.

In database of this system be ready to look at operation used on some table in this database so that you can know what to and what not to do on given entity.

Note that to develop this system database there are some material which were used so that we can get on final output of system as was described in chapter one “system analyses”. Among these the important one is xampp saver MySQL, so now let together navigate this database system.

* 1. **​SECTION1:**
     1. **​ENTITIES**

1. Description of all entities and their corresponding attributes that are in this data base

***User information database structure***

This is table that will be only created by the system administrator and will hold other system users information including their first and last names, telephone numbers, address, vehicle ID, License plate and model of their vehicle.



|  |  |
| --- | --- |
| id-primary key | Id for the user |
| Firstname(varchar) | first name of the user |
| Lastname(varchar) | Last name of the user |
| Telephone(varchar) | Phone number of the user |
| Address(varchar) | Where the user lives |
| VehicleID | Vehicle identity number |
| License Plate | Vehicle s plate number |
| Model | Vehicle s model |

***Parking ticket information table database structure:***

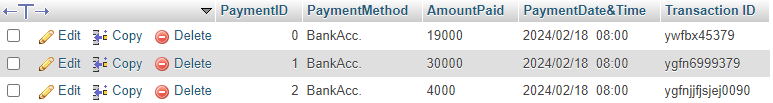
This table to contain Ticket ID, issue date and time, location where the vehicle is parked, Violation type and the fine amount if only there is any.



|  |  |
| --- | --- |
| Ticket ID(int)-primary key | Id of the Ticket |
| IssueDate&Time(varchar) | Date and time when ticket was issued |
| Location | Location of the parking lot |
| Violation type | Any violation committed |
| FineAmount | Fines to be paid |
| Status | Status of the fine |

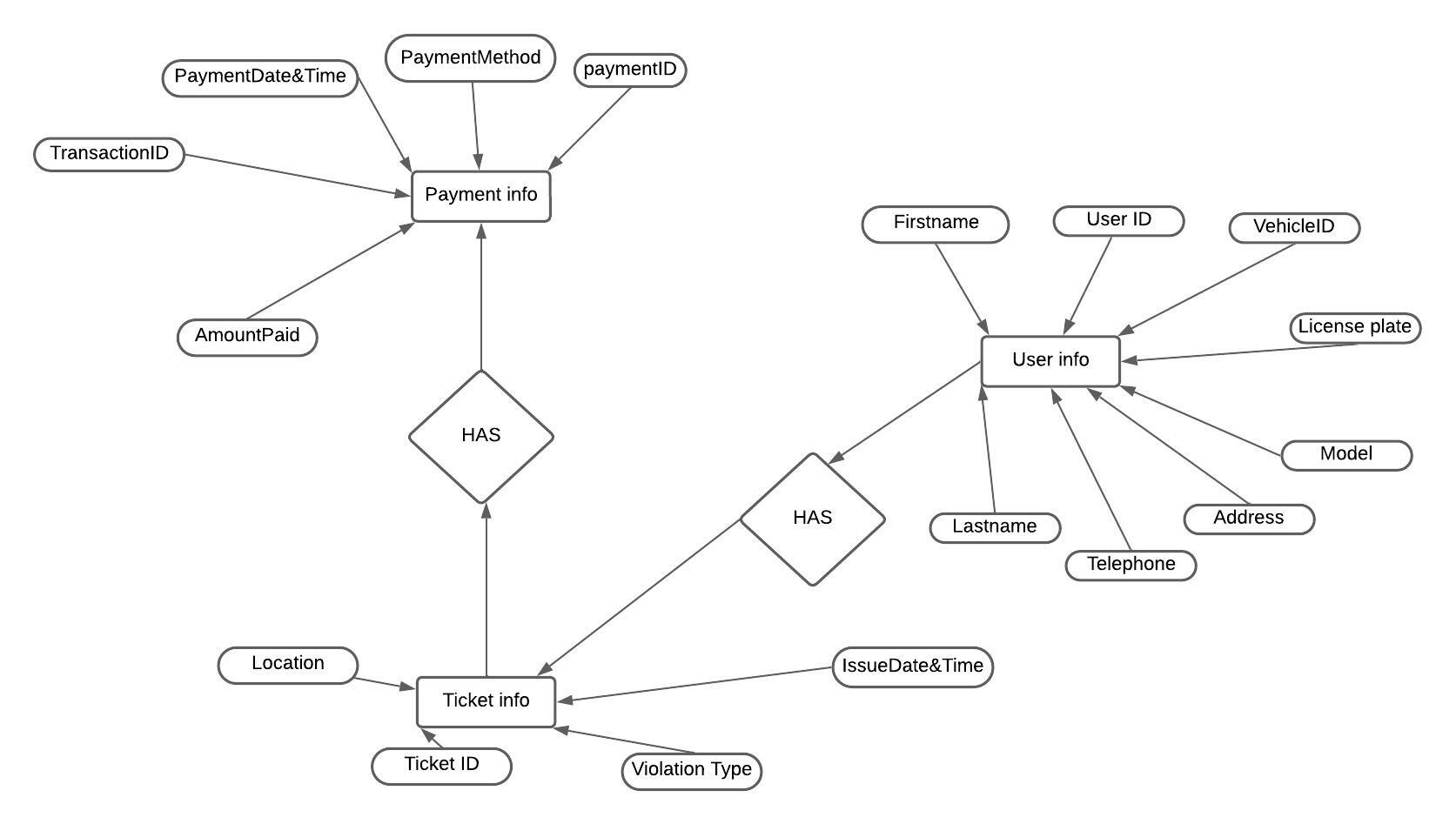
***Payment information table database structure***

This table to contains all tickets payment information including payment method, amount paid, payment date and time and the transaction ID.



|  |  |
| --- | --- |
| PaymentID(int)-primary key | Id of the payment |
| PaymentMethod(varchar) | Method used for Payment |
| AmountPaid(varchar) | Amount paid |
| PaymentDate&Time(varchar) | Date and Time when payment was carried out |
| Transaction ID(varchar) | Transaction identity |

* + 1. ENTITY RELETIONSHIP DIAGRAM



* 1. **​SECTION2:SQL**

In this sub unit we will be describing techniques especially SQL queries used to create, delete tables as well as inserting and deleting data in the tables in that database.

1. **QUERIES TO CREATE ALL THE TABLES IN OUR DATABASE**

* **Query to create “userinfo” table and it ‘s attributes:**

# CREATE TABLE userinfo (

# id INT AUTO\_INCREMENT PRIMARY KEY,

# Firstname VARCHAR(255),

# Lastname VARCHAR(255),

# Telephone VARCHAR(20),

# Address VARCHAR(255),

# VehicleID VARCHAR(20),

# LicensePlate VARCHAR(20),

# Model VARCHAR(255)

# );

* **Query to create “ParkingTicketInfo” table and it ‘s attributes:**

# CREATE TABLE parkingticketinfo (

# TicketID INT AUTO\_INCREMENT PRIMARY KEY,

# IssueDateTime DATETIME,

# Location VARCHAR(255),

# ViolationType VARCHAR(255),

# FineAmount DECIMAL(10, 2),

# Status VARCHAR(50)

# );

# 

# **Query to create “PaymentInfo” table and it ‘s attributes:**

# CREATE TABLE paymentinfo (

# PaymentID INT AUTO\_INCREMENT PRIMARY KEY,

# PaymentMethod VARCHAR(255),

# AmountPaid DECIMAL(10, 2),

# PaymentDateTime DATETIME,

# TransactionID VARCHAR(255)

);

# QUERIES TO INSERT DATA INTO OUR DATABASE

# 

# **Insert data into userinfo table**

# INSERT INTO userinfo (Firstname, Lastname, Telephone, Address, VehicleID, LicensePlate, Model)

# VALUES

# ('Mugisha', 'Salomon', '+250782979568', 'Remera', '783gf920bcye', 'RAE380E', 'Hyundai Kona'),

# ('KAGABO', 'Methode', '+250788888888', 'Kayonza', '783gf920bzzzz', 'RAE888E', 'Hyundai Tucson'),

# ('Mugisha', 'Danny', '+25070000000', 'Kimironko', '783gbbvbfhd', 'RAE000E', 'Hyundai Santa Fe');

# Insert data into parkingticketinfo table

# INSERT INTO parkingticketinfo (IssueDateTime, Location, ViolationType, FineAmount, Status)

# VALUES

# ('2024-02-18 08:00:00', 'Parking lot 1', 'Double Parking Violation', 60000.00, 'pending'),

# ('2024-02-18 08:00:00', 'parking lot 2', 'Not Paying for metered parking', 57749.00, 'Active'),

# ('2024-02-18 08:00:00', 'Parking lot 8', 'Parking in One Spot for More than 24 Hours', 60000.00, 'Active');

# Insert data into paymentinfo table

# INSERT INTO paymentinfo (PaymentMethod, AmountPaid, PaymentDateTime, TransactionID)

# VALUES

# ('BankAcc.', 30000.00, '2024-02-18 08:00:00', '6498430'),

# ('BankAcc.', 4000.00, '2024-02-18 08:00:00', '650003438'),

# ('BankAcc.', 30000.00, '2024-02-18 08:00:00', '75973993'),

# ('BankAcc.', 19000.00, '2024-02-18 08:00:00', '648757839')

1. **queries to update two tables in our database**

UPDATE userinfo

SET Firstname = 'NewFirstname', Lastname = 'NewLastname'

WHERE id = 1; -- Update the record with id = 1

'

UPDATE parkingticketinfo

SET FineAmount = 65000.00, Status = 'Paid'

WHERE TicketID = 1; -- Update the record with TicketID = 1

'

## UPDATE paymentinfo

## SET AmountPaid = 25000.00, PaymentMethod = 'CreditCard'

## WHERE PaymentID = 1; -- Update the record with PaymentID = 1

* 1. **SECTION3**
     1. **VIEWS**
        1. ***view to insert data in given tables***

## CREATE VIEW UserInfoView AS

## SELECT u.id, u.Firstname, u.Lastname, u.Telephone, u.Address, u.VehicleID, u.LicensePlate, u.Model,

## p.TicketID, p.IssueDateTime, p.Location, p.ViolationType, p.FineAmount, p.Status,

## py.PaymentID, py.PaymentMethod, py.AmountPaid, py.PaymentDateTime, py.TransactionID

## FROM userinfo u

## LEFT JOIN parkingticketinfo p ON u.id = p.UserID

## LEFT JOIN paymentinfo py ON p.TicketID = py.TicketID;

## ('14','bignners','b','300000','30000','12000','63000','21000','987000','63000','51632')

This SQL code creates a view named UserInfoView, which joins data from the userinfo, parkingticketinfo, and paymentinfo tables based on their relationships. Adjust the join conditions and select columns according to your specific requirements.

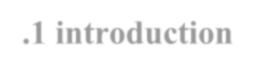
In this view:

Data from userinfo is selected directly.

Data from parkingticketinfo is left joined with userinfo using the UserID foreign key relationship.

Data from paymentinfo is left joined with parkingticketinfo using the TicketID foreign key relationship.

This view can be queried like a regular table, allowing you to retrieve consolidated information from multiple tables in your database**.**



* 1. **introduction**

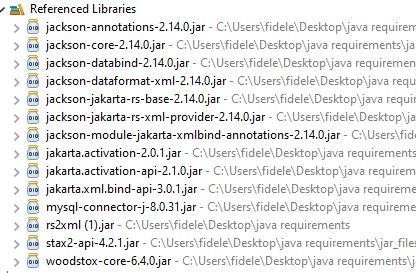
In this chapter I will be describing how powerful general-

purpose **programming** language was used to create the analyzed system. Under this chapter I will undergo full detail of how everything will function together with database that have been describe above and how it cope with full analyzed system.

## Tools used to develop this system in java programming:

Eclipse IDE: an integrated development environment used in computer programming. It contains a base workspace and an extensible plug-in system for customizing the environment. It is the second-most-popular IDE for Java development, and, until 2016, was the most popular.

**JAR stands for Java Archive**. It's a file format based on the popular ZIP file format and is used for aggregating many files into one. Although JAR can be used as a general archiving tool, the primary motivation for its development was so that Java applets and their requisite components.



**MySQL Connectors**. **MySQL** provides standards-based drivers for JDBC, ODBC, and

.Net enabling developers to build data

## **3.3 Tools used to develop this system in java programming :**

**Integrated Development Environment (IDE):**

# 1.Eclipse: A popular Java IDE that offers features like code completion, refactoring, debugging, and more.

# 2.Database Management System (DBMS):

# MySQL: Since you're working with a MySQL database, you'll need to have MySQL installed on your system.

# 3.Database Connector (JDBC Driver):

## You'll need the JDBC driver for your chosen database system. For MySQL, you can use mysql-connector-java. Make sure to include this driver in your project's dependencies.

**3.4 Forms Description**

**To create forms ,we are going to use Eclipse IDE s’ window builder.**

In our system, we have three forms. The one for user information interface which will help customers send requests to the database. We also have parking ticket information form as well as payment form to help an administrator or an employee to interact with the database and the system.

1. **Here is an example of a java code for ticket information ‘s form:**

import java.awt.EventQueue;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.border.EmptyBorder;

import javax.swing.JLabel;

import java.awt.Font;

import javax.swing.JTextField;

import javax.swing.JButton;

public class parkingticketinfo extends JFrame {

private static final long serialVersionUID = 1L;

private JPanel contentPane;

private JTextField textField;

private JTextField textField\_1;

private JTextField textField\_2;

private JTextField textField\_3;

private JTextField textField\_4;

private JTextField textField\_5;

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

parkingticketinfo frame = new parkingticketinfo();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the frame.

\*/

public parkingticketinfo() {

setTitle("PARKING TICKET INFORMATION");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setBounds(100, 100, 957, 566);

contentPane = new JPanel();

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

contentPane.setLayout(null);

JLabel lblNewLabel = new JLabel("Ticket ID:");

lblNewLabel.setFont(new Font("Tahoma", Font.BOLD, 14));

lblNewLabel.setBounds(76, 24, 143, 47);

contentPane.add(lblNewLabel);

JLabel lblIssueDateAnd = new JLabel("Issue Date and Time");

lblIssueDateAnd.setFont(new Font("Tahoma", Font.BOLD, 14));

lblIssueDateAnd.setBounds(76, 97, 143, 47);

contentPane.add(lblIssueDateAnd);

JLabel lblNewLabel\_1\_1 = new JLabel("Location");

lblNewLabel\_1\_1.setFont(new Font("Tahoma", Font.BOLD, 14));

lblNewLabel\_1\_1.setBounds(76, 173, 143, 47);

contentPane.add(lblNewLabel\_1\_1);

JLabel lblNewLabel\_1\_1\_1 = new JLabel("violation type");

lblNewLabel\_1\_1\_1.setFont(new Font("Tahoma", Font.BOLD, 14));

lblNewLabel\_1\_1\_1.setBounds(76, 238, 143, 47);

contentPane.add(lblNewLabel\_1\_1\_1);

JLabel lblNewLabel\_1\_1\_1\_1 = new JLabel("Fine Amount\r\n");

lblNewLabel\_1\_1\_1\_1.setFont(new Font("Tahoma", Font.BOLD, 14));

lblNewLabel\_1\_1\_1\_1.setBounds(76, 310, 143, 47);

contentPane.add(lblNewLabel\_1\_1\_1\_1);

JLabel lblNewLabel\_1\_1\_1\_1\_1 = new JLabel("Status");

lblNewLabel\_1\_1\_1\_1\_1.setFont(new Font("Tahoma", Font.BOLD, 14));

lblNewLabel\_1\_1\_1\_1\_1.setBounds(76, 380, 143, 47);

contentPane.add(lblNewLabel\_1\_1\_1\_1\_1);

textField = new JTextField();

textField.setBounds(353, 39, 361, 20);

contentPane.add(textField);

textField.setColumns(10);

textField\_1 = new JTextField();

textField\_1.setColumns(10);

textField\_1.setBounds(353, 112, 361, 20);

contentPane.add(textField\_1);

textField\_2 = new JTextField();

textField\_2.setColumns(10);

textField\_2.setBounds(353, 188, 361, 20);

contentPane.add(textField\_2);

textField\_3 = new JTextField();

textField\_3.setColumns(10);

textField\_3.setBounds(353, 253, 361, 20);

contentPane.add(textField\_3);

textField\_4 = new JTextField();

textField\_4.setColumns(10);

textField\_4.setBounds(353, 325, 361, 20);

contentPane.add(textField\_4);

textField\_5 = new JTextField();

textField\_5.setColumns(10);

textField\_5.setBounds(353, 395, 361, 20);

contentPane.add(textField\_5);

JButton btnNewButton = new JButton("Save");

btnNewButton.setFont(new Font("Tahoma", Font.BOLD, 14));

btnNewButton.setBounds(64, 462, 171, 47);

contentPane.add(btnNewButton);

JButton btnClear = new JButton("Update");

btnClear.setFont(new Font("Tahoma", Font.BOLD, 14));

btnClear.setBounds(371, 462, 171, 47);

contentPane.add(btnClear);

JButton btnNewButton\_1\_1 = new JButton("clear");

btnNewButton\_1\_1.setFont(new Font("Tahoma", Font.BOLD, 14));

btnNewButton\_1\_1.setBounds(670, 462, 171, 47);

contentPane.add(btnNewButton\_1\_1);

}

}

1. **The next is the code for the user information form s’ code in Eclipse IDE:**

package parkingticketsystem;

import java.awt.EventQueue;

import java.sql.\*;

import javax.swing.JFrame;

import javax.swing.JPanel;

import java.awt.Color;

import java.awt.Container;

import javax.swing.border.BevelBorder;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JTextField;

import java.awt.Font;

import java.awt.GridLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.JTable;

import javax.swing.JScrollPane;

import javax.swing.JButton;

public class UserInfo extends JFrame implements ActionListener {

String url = "jdbc:mysql://localhost/userinfo";

String UserN = "root";

String passD = "";

private static final long serialVersionUID = 1L;

private JPanel contentPane;

private JTextField tffirstname;

private JTextField tfsecondname;

private JTextField tftelephone;

private JTextField tfaddress;

private JTextField tfid;

private JTextField tflicense;

private JTextField tfmodel;

private JTextField tfusername;

private JTextField tfpassword;

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

UserInfo frame = new UserInfo();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the frame.

\*/

public UserInfo() {

setTitle("User Login");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setBounds(100, 100, 388, 525);

contentPane = new JPanel();

contentPane.setForeground(new Color(0, 0, 0));

contentPane.setBorder(new BevelBorder(BevelBorder.LOWERED, null, null, null, null));

setContentPane(contentPane);

contentPane.setLayout(null);

JLabel lblNewLabel = new JLabel("FIRSTNAME:");

lblNewLabel.setBounds(10, 11, 95, 30);

lblNewLabel.setFont(new Font("Tahoma", Font.BOLD, 11));

contentPane.add(lblNewLabel);

JLabel lblNewLabel\_1 = new JLabel("LASTNAME:");

lblNewLabel\_1.setBounds(10, 47, 95, 23);

lblNewLabel\_1.setFont(new Font("Tahoma", Font.BOLD, 11));

contentPane.add(lblNewLabel\_1);

tffirstname = new JTextField();

tffirstname.setBounds(153, 16, 205, 20);

contentPane.add(tffirstname);

tffirstname.setColumns(10);

tfsecondname = new JTextField();

tfsecondname.setBounds(153, 48, 205, 20);

contentPane.add(tfsecondname);

tfsecondname.setColumns(10);

JLabel lblNewLabel\_2 = new JLabel("Tel+:");

lblNewLabel\_2.setBounds(10, 81, 50, 40);

lblNewLabel\_2.setFont(new Font("Tahoma", Font.BOLD, 11));

contentPane.add(lblNewLabel\_2);

tftelephone = new JTextField();

tftelephone.setBounds(153, 91, 205, 20);

contentPane.add(tftelephone);

tftelephone.setColumns(10);

JLabel lblNewLabel\_3 = new JLabel("Address:");

lblNewLabel\_3.setBounds(10, 137, 95, 30);

lblNewLabel\_3.setFont(new Font("Tahoma", Font.BOLD, 11));

contentPane.add(lblNewLabel\_3);

tfaddress = new JTextField();

tfaddress.setBounds(151, 142, 207, 20);

contentPane.add(tfaddress);

tfaddress.setColumns(10);

JLabel lblNewLabel\_4 = new JLabel("Vehicle Information:");

lblNewLabel\_4.setBounds(11, 178, 117, 40);

lblNewLabel\_4.setFont(new Font("Tahoma", Font.BOLD, 11));

contentPane.add(lblNewLabel\_4);

JLabel lblNewLabel\_5 = new JLabel("Vehicle ID:");

lblNewLabel\_5.setBounds(21, 224, 89, 14);

lblNewLabel\_5.setFont(new Font("Tahoma", Font.BOLD, 11));

contentPane.add(lblNewLabel\_5);

JLabel lblNewLabel\_6 = new JLabel("License Plate:");

lblNewLabel\_6.setBounds(21, 264, 89, 14);

lblNewLabel\_6.setFont(new Font("Tahoma", Font.BOLD, 11));

contentPane.add(lblNewLabel\_6);

JLabel lblNewLabel\_7 = new JLabel("Model:");

lblNewLabel\_7.setBounds(21, 308, 49, 14);

lblNewLabel\_7.setFont(new Font("Tahoma", Font.BOLD, 11));

contentPane.add(lblNewLabel\_7);

tfid = new JTextField();

tfid.setBounds(153, 221, 205, 20);

contentPane.add(tfid);

tfid.setColumns(10);

tflicense = new JTextField();

tflicense.setBounds(153, 261, 205, 20);

contentPane.add(tflicense);

tflicense.setColumns(10);

tfmodel = new JTextField();

tfmodel.setBounds(154, 305, 205, 20);

contentPane.add(tfmodel);

tfmodel.setColumns(10);

JLabel lblNewLabel\_8 = new JLabel("Login Credentials:");

lblNewLabel\_8.setBounds(10, 357, 89, 14);

lblNewLabel\_8.setFont(new Font("Tahoma", Font.BOLD, 11));

contentPane.add(lblNewLabel\_8);

JLabel lblNewLabel\_9 = new JLabel("Username:");

lblNewLabel\_9.setBounds(106, 357, 104, 14);

lblNewLabel\_9.setFont(new Font("Tahoma", Font.BOLD, 11));

contentPane.add(lblNewLabel\_9);

JLabel lblNewLabel\_10 = new JLabel("Password:");

lblNewLabel\_10.setBounds(106, 404, 125, 14);

lblNewLabel\_10.setFont(new Font("Tahoma", Font.BOLD, 11));

contentPane.add(lblNewLabel\_10);

tfusername = new JTextField();

tfusername.setBounds(195, 354, 163, 20);

contentPane.add(tfusername);

tfusername.setColumns(10);

tfpassword = new JTextField();

tfpassword.setBounds(195, 401, 163, 20);

contentPane.add(tfpassword);

tfpassword.setColumns(10);

JButton btnNewButton = new JButton("Update");

btnNewButton.setBounds(134, 429, 95, 43);

contentPane.add(btnNewButton);

btnNewButton.addActionListener(this);

JButton btnSave = new JButton("Save");

btnSave.setBounds(10, 429, 95, 43);

contentPane.add(btnSave);

btnSave.addActionListener(this);

JButton btnreset = new JButton("Reset");

btnreset.setBounds(263, 429, 95, 43);

contentPane.add(btnreset);

btnreset.addActionListener(this);

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getActionCommand().equals("Save")){

String firstName = tffirstname.getText();

String lastName = tfsecondname.getText();

String phoneNumber = tftelephone.getText();

String address = tfaddress.getText();

// String password = tfpassword.getText();

String identity = tfid.getText();

String license = tflicense.getText();

String model = tfmodel.getText();

// String username = tfusername.getText();

String inSql = "INSERT INTO userinfo (Firstname, Lastname, Telephone, Address, VehicleID, LicensePlate, Model) VALUES (?, ?, ?, ?, ?, ?, ?)";

try (

Connection conn = DriverManager.getConnection(url, UserN, passD);

PreparedStatement Stmt = conn.prepareStatement(inSql, Statement.RETURN\_GENERATED\_KEYS)

) {

Stmt.setString(1, firstName);

Stmt.setString(2, lastName);

Stmt.setString(3, phoneNumber);

Stmt.setString(4, address);

// Stmt.setString(5, password);

Stmt.setString(5, identity);

Stmt.setString(6, license);

Stmt.setString(7, model);

// Stmt.setString(9, username);

int affectedRows = Stmt.executeUpdate();

if (affectedRows > 0) {

JOptionPane.showMessageDialog(null, "Signup Successful!");

// Optionally, you can close the signup frame or perform other actions

} else {

JOptionPane.showMessageDialog(null, "Failed to insert login credentials!");

}

}

catch (SQLException ex) {

//JOptionPane.showMessageDialog(null, "Enter all fields");

ex.printStackTrace();

}

}else if (e.getActionCommand().equals("Update")) {

String sql = "SELECT \* FROM userinfo ORDER BY id DESC LIMIT 1";

try (Connection conn = DriverManager.getConnection(url, UserN, passD);

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(sql)) {

if (rs.next()) {

int id = rs.getInt("id");

String fname = rs.getString("Firstname");

String lname = rs.getString("Lastname");

String phoneNumber = rs.getString("Telephone");

String address = rs.getString("Address");

String identity = rs.getString("VehicleID");

String license = rs.getString("LicensePlate");

String model = rs.getString("Model");

JFrame viewFrame = new JFrame("Update");

viewFrame.getContentPane().setLayout(new GridLayout(0,2));

// viewFrame.setLocationRelativeTo(null);

viewFrame.setBounds(150, 50, 400, 300);

addLabelAndData(viewFrame, "ID:", String.valueOf(id));

addLabelAndData(viewFrame, "First Name:", fname);

addLabelAndData(viewFrame, "Last Name:", lname);

addLabelAndData(viewFrame, "Telephone:", phoneNumber);

addLabelAndData(viewFrame, "Model:", model);

addLabelAndData(viewFrame, "Address:", address);

addLabelAndData(viewFrame, "identity:", identity);

addLabelAndData(viewFrame, "license:", license);

viewFrame.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

// viewFrame.pack();

viewFrame.setVisible(true);

} else {

JOptionPane.showMessageDialog(null, "No data found!", "Error", JOptionPane.ERROR\_MESSAGE);

}

} catch (SQLException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(null, "Failed to view data!", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

}

private void addLabelAndData(Container container, String labelText, String data) {

container.add(new JLabel(labelText));

container.add(new JLabel(data));

}

}

1. **The last form is for paymentInfo form which will help in entry of new data into the database. So it ‘s java code is below:**

package parkingticketsystem;

import java.awt.EventQueue;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.border.EmptyBorder;

import javax.swing.JLabel;

import java.awt.Font;

import javax.swing.JTextField;

import javax.swing.JButton;

public class PaymentInfo extends JFrame {

private static final long serialVersionUID = 1L;

private JPanel contentPane;

private JTextField tfid;

private JTextField tfmethod;

private JTextField tfpaid;

private JTextField tftime;

private JTextField tftransaction;

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

PaymentInfo frame = new PaymentInfo();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the frame.

\*/

public PaymentInfo() {

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setBounds(100, 100, 743, 643);

contentPane = new JPanel();

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

contentPane.setLayout(null);

JLabel lblNewLabel = new JLabel("PaymentID");

lblNewLabel.setFont(new Font("Tahoma", Font.BOLD, 14));

lblNewLabel.setBounds(38, 38, 188, 52);

contentPane.add(lblNewLabel);

JLabel lblPaymentmethod = new JLabel("PaymentMethod");

lblPaymentmethod.setFont(new Font("Tahoma", Font.BOLD, 14));

lblPaymentmethod.setBounds(38, 131, 188, 52);

contentPane.add(lblPaymentmethod);

JLabel lblNewLabel\_1\_1 = new JLabel("AmountPaid");

lblNewLabel\_1\_1.setFont(new Font("Tahoma", Font.BOLD, 14));

lblNewLabel\_1\_1.setBounds(38, 217, 188, 52);

contentPane.add(lblNewLabel\_1\_1);

JLabel lblNewLabel\_1\_1\_1 = new JLabel("PaymentDate&Time");

lblNewLabel\_1\_1\_1.setFont(new Font("Tahoma", Font.BOLD, 14));

lblNewLabel\_1\_1\_1.setBounds(38, 304, 188, 52);

contentPane.add(lblNewLabel\_1\_1\_1);

JLabel lblNewLabel\_1\_1\_1\_1 = new JLabel("TransactionID");

lblNewLabel\_1\_1\_1\_1.setFont(new Font("Tahoma", Font.BOLD, 14));

lblNewLabel\_1\_1\_1\_1.setBounds(38, 392, 188, 52);

contentPane.add(lblNewLabel\_1\_1\_1\_1);

tfid = new JTextField();

tfid.setBounds(287, 56, 369, 34);

contentPane.add(tfid);

tfid.setColumns(10);

tfmethod = new JTextField();

tfmethod.setColumns(10);

tfmethod.setBounds(287, 149, 369, 34);

contentPane.add(tfmethod);

tfpaid = new JTextField();

tfpaid.setColumns(10);

tfpaid.setBounds(288, 235, 369, 34);

contentPane.add(tfpaid);

tftime = new JTextField();

tftime.setColumns(10);

tftime.setBounds(288, 322, 369, 34);

contentPane.add(tftime);

tftransaction = new JTextField();

tftransaction.setColumns(10);

tftransaction.setBounds(288, 410, 369, 34);

contentPane.add(tftransaction);

JButton btnclear = new JButton("Clear");

btnclear.setFont(new Font("Tahoma", Font.BOLD, 15));

btnclear.setBounds(114, 506, 170, 52);

contentPane.add(btnclear);

JButton btnSave = new JButton("Save");

btnSave.setFont(new Font("Tahoma", Font.BOLD, 15));

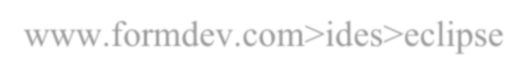
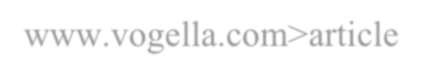
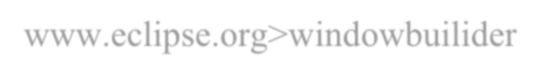
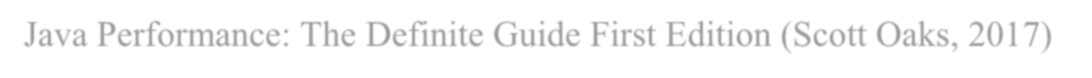
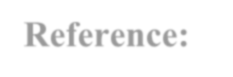
btnSave.setBounds(449, 506, 170, 52);

contentPane.add(btnSave);

}

}

3.4 Conclusion: By concluding this chapter concerns with java programming especially in my developed system, we can say that I have final product that I was expecting to have it, the manipulation of data is going well the design is there with special appearance, but there much to go on and that need to be improved will be gained from external view apart from system developer.



**Reference:**

1. Java Performance: The Definite Guide First Edition (Scott Oaks, 2017)
2. [www.eclipse.org>w](http://www.eclipse.org/)indowbuilider
3. [www.vogella.com](http://www.vogella.com/)>article
4. [www.formdev.com>id](http://www.formdev.com/)es>eclipse