# **PRAKTIKUM**

**PEMROGRAMAN BERORIENTASI OBJEK**

**SEMESTER GENAP T.A 2019/2020**

**LAPORAN PROYEK AKHIR**

****

**DISUSUN OLEH :**

|  |  |  |  |
| --- | --- | --- | --- |
| **NAMA** | **:** | **ANANDA FUAD NUR F**  **MAHARDIKA RAHMAN** | **(123180158)**  **(123180178)** |
| **KELAS/PLUG** | **:** | **C** |  |
| **NAMA ASISTEN** | **:** | **MUHAMMAD AFIN FAUZI** |  |
|  |  | **GILBERT GIORALDO MAULANY** |  |

### PROGRAM STUDI TEKNIK INFORMATIKA

### JURUSAN TEKNIK INFORMATIKA

### FAKULTAS TEKNIK INDUSTRI

### UNIVERSITAS PEMBANGUNAN NASIONAL ”VETERAN”

#### YOGYAKARTA

#### 2020

**HALAMAN PENGESAHAN**

**LAPORAN PROYEK AKHIR**

****

Disusun oleh : Disusun oleh :

*Ananda Fuad Nur F Mahardika Rahman R*

123180158 123180178

Telah Diperiksa dan Disetujui oleh Asisten Praktikum Pemrograman Berorientasi Objek

Pada Tanggal : .................................

|  |  |
| --- | --- |
| **Asisten Praktikum**  **Gilbert Gioraldo Maulany**  **NIM. 124170022** | **Asisten Praktikum**  **Muhammad Afin Fauzi**  **NIM. 124170019** |

|  |
| --- |
| **Mengetahui,**  **Ka. Lab. Basis Data**  **Dessyanto Boedi Prasetyo,S.T.,M.T.**  **NIK. 2 7512 00 0229 1** |

**KATA PENGANTAR**

Puji syukur saya panjatkan kepada Tuhan Yang Maha Esa yang senantiasa mencurahkan rahmat dan hidayah-Nya sehingga saya dapat menyelesaikan praktikum Pemrograman Berorientasi Objek serta laporan proyek akhir praktikum yang berjudul Rental Kendaraan Adapun laporan ini berisi tentang proyek akhir yang saya pilih dari hasil pembelajaran selama praktikum berlangsung.

Tidak lupa ucapan terimakasih kepada asisten dosen yang selalu membimbing dan mengajari saya dalam melaksanakan praktikum dan dalam menyusun laporan ini. Laporan ini masih sangat jauh dari kesempurnaan, oleh karena itu kritik serta saran yang membangun saya harapkan untuk menyempurnakan laporan akhir ini.

Atas perhatian dari semua pihak yang membantu penulisan ini, saya ucapkan terimakasih. Semoga laporan ini dapat dipergunakan seperlunya.

Yogyakarta, 15 Mei 2020

Penyusun

**DAFTAR ISI**

**COVER………………………........................................................................................... i**

**HALAMAN PENGESAHAN .......................................................................................... ii DAFTAR ISI ................................................................................................................... iii**

BAB I JUDUL PROYEK AKHIR.......................................................................................1

BAB II PENDAHULUAN ..................................................................................................2

2.1 Latar Belakang...............................................................…...................................2

2.2 Tujuan ..........................................................................................…....................2

BAB III PEMBAHASAN………...................................................…................................3

3.1 Dasar Teori………………. ..............................................…................................3

3.2 Deskripsi Umum Project Akhir ..........................................…..............................6

3.3 Inti Pembahasan …................................................….........................…………...6

BAB IV JADWAL PENGERJAAN DAN PEMBAGIAN TUGAS .................................20

3.1 Jadwal Pengerjaan..............................................................................................20

3.2 Pembagian Tugas ...............................................................................................20

BAB V KESIMPULAN DAN SARAN ...........................................................................21

4.1 Kesimpulan........................................................................................................21

4.2 Saran .................................................................................................................21

**DAFTAR PUSTAKA................................................................................................….22**

**BAB I**

**RENTAL KENDARAAN**

**BAB II**

**PENDAHULUAN**

**2.1 Latar Belakang masalah**

Pada zaman modern ini setiap masyarakat lebih mengutamakan kemudahan dalam melakukan sesuatu pekerjaan, tak terkecuali dalam bidang bisnis. Oleh karena itu banyak muncul aplikasi-aplikasi untuk memudahkan dalam berbisnis. Jadi pelanggan dan penjual dapat melakukan traksaksi tanpa perlu bertatap muka secara langsung. Hal tersebut dinilai efektif dalam segi waktu maupun tenaga. Pelanggan dapat melakukan transaksi secara daring menggunakan aplikasi-aplikasi yang telah disediakan.

**2.2 Tujuan Project akhir**

Tujuan dari perancangan aplikasi tersebut adalah memudahkan pelanggan maupun admin dalam melakukan transaksi sewa kendaraan. pelanggan dapat menyewa kendaraan secara daring, ketersediaan barang yang disewakan pun dapat dilihat dengan jelas oleh pelanggan.

**BAB III**

**PEMBAHASAN**

**3.1. Dasar Teori**

Java adalah bahasa pemrograman yang paling populer digunakan untuk pengembangan perangkat lunak maupun aplikasi. Hal tersebut karena fungsionalitasnya yang memungkinkan java berjalan dibeberapa sistem operasi yang berbeda.

* 1. Object oriented programming

Inheritance adalah penurunan atribut dan method dari suatu kelas induk (superclass) ke kelas anak (subclass). Pewarisan bersifat menyeluruh dan beruntun, pernyataan yang digunakan untuk melakukan pewarisan adalah exstends.

Encapsulation atau pengkapsulan digunakan untuk menyembunyikan atribut atau method pada suatu objek, dari objek lainnya, atau dengan kata lain yakni *Information hiding.* Pengkapsulan dilakukan dengan memberikan modifier private pada atribut maupun method. Kemudian, untuk memberikan nilai pada atribut private, digunakan method *‘setter’*, dan untuk mengambil nilai dari atribut private, digunakan method *‘getter’*.

Polymorphism dapat diartikan sebagai kemampuan untuk membuat banyak objek berbeda dari satu sumber yang sama. Dalam Java, contoh yang mudah adalah membuat beberapa objek berbeda dari beberapa subclass yang mewarisi satu superclass yang sama.

Abstract class adalah kelas yang tidak dapat diinstansiasi karena terlalu umum cakupannya. Method yang abstract adalah method yang belum memiliki implementasi (isi) dan menggunakan modifier abstract. Method ini diisi nilainya oleh kelas yang menjadi turunannya, hal ini disebut dengan overriding.

Interface merupakan suatu kelas yang berisi method-method tanpa implementasi, namun tanpa modifier abstract. Apabila terdapat atribut, maka atribut tersebut berlaku sebagai konstanta (static final). Interface bersifat seperti kontrak yang bisa digunakan oleh setiap kelas yang nyata, dan penggunaannya bukan diwariskan (extends) melainkan diimplementasikan (implements).

* 1. Exception

Exception merupakan suatu objek Error khusus yang dibuat ketika terjadi kesalahan pada suatu program. Code program exception handling secara umum terdiri dari 3 bagian yaitu try, catch dan finally. Kategori error ada 3 yaitu syntax error (kesalahan syntax), runtime error (kesalahan dalam input), dan logic error (kesalahan logika programmer). Jika sebuah metod memungkinkan terjadinya kesalahan exception tetapi exception tidak menangkapnya, maka di gunakan kewword throws.

* 1. Swing

Swing adalah kumpulan class–class yang digunakan untuk membuat GUI. GUI merupakan suatu metode untuk antar-muka komputer berbasis grafis. GUI digunakan dalam pembuatan program aplikasi dengan mempertimbangkan dua aspek yaitu keindahan tampilan dan kemudahan penggunaan program.

Dalam lingkup Java, sebuah interface window dikenal dengan sebutan frame. Frame adalah sebuah objek yang merepresentasikan suatu area di layar yang berisi sejumlah objek lain untuk menyampaikan informasi kepada user. Dengan adanya frame, berbagai komponen GUI dapat ditampilkan. Untuk menentukan ukuran frame yaitu dengan menggunakan method setSize(int x,int y). Contoh komponen-komponen swing adalah Jlabel, JtextField, JButton.

* 1. Event Handling

Event Handling merupakan tindakan yang harus dilakukan oleh komponen untuk dapat berinteraksi dengan program. Ada beberapa event handling, yaitu :

* ActionListener berfungsi untuk menangani reaksi atas perubahan pada suatu objek yang digunakan.
* AdjustmentListener berfungsi untuk menangani reaksi atas perubahan status pada objek yang digunakan pada saat penyesuaian.
* FocusListener berfungsi untuk menangani reaksi atas perubahan status akibat fokus berada pada objek tertentu.
* ItemListener berfungsi untuk menangani reaksi atas perubahan status pada pilihan yang terdapat pada objek tertentu.
* KeyListener berfungsi untuk menangani reaksi dari keyboard, misalkan penekanan tombol Ctrl, Alt, dan sebagainya.
* WindowListener berfungsi untuk menangani reaksi atas perubahan window akibat ukuran diperbesar, dikecilkan, dipindah, maupun ditutup.
* MouseMotionListener berfungsi untuk menangani reaksi akibat mouse digerakkan, misalkan melewati lokasi objek tertentu.
  1. Stream & File

Di dalam java, Stream dibedakan menjadi 2 jenis, yaitu Byte Stream (aliran data berupa Byte) dan Character Stream (aliran data berupa karakter). Operasi/aliran Stream dibedakan menjadi 2 yaitu input Stream dan output Stream.

Java menggunakan kata kunci Reader/Writer dalam kelas-kelas Stream yang berbasis karakter, menggunakan kata kunci Stream dalam kelas-kelas Stream yang berbasis Byte, menggunakan kata Buffered untuk kelas-kelas Pem-buffer dan menggunakan kata StreamReader/StreamWriter untuk kelas-kelas yang menjembatani Stream berbasis Byte dengan Stream berbasis karakter.

* 1. Java Database Connectivity

Aplikasi Java yang dibuat, agar dapat terhubung dengan database memerlukan driver JDBC yang sesuai. JDBC API tersedia dalam paket java.sql dan javax.sql. Langkah-langkah yang perlu dilakukan sebagai berikut :

* Memanggil Driver JDBC
* Melakukan koneksi database
* Membuat objek statement
* Melakukan query
* Memproses hasil
* Menutup koneksi

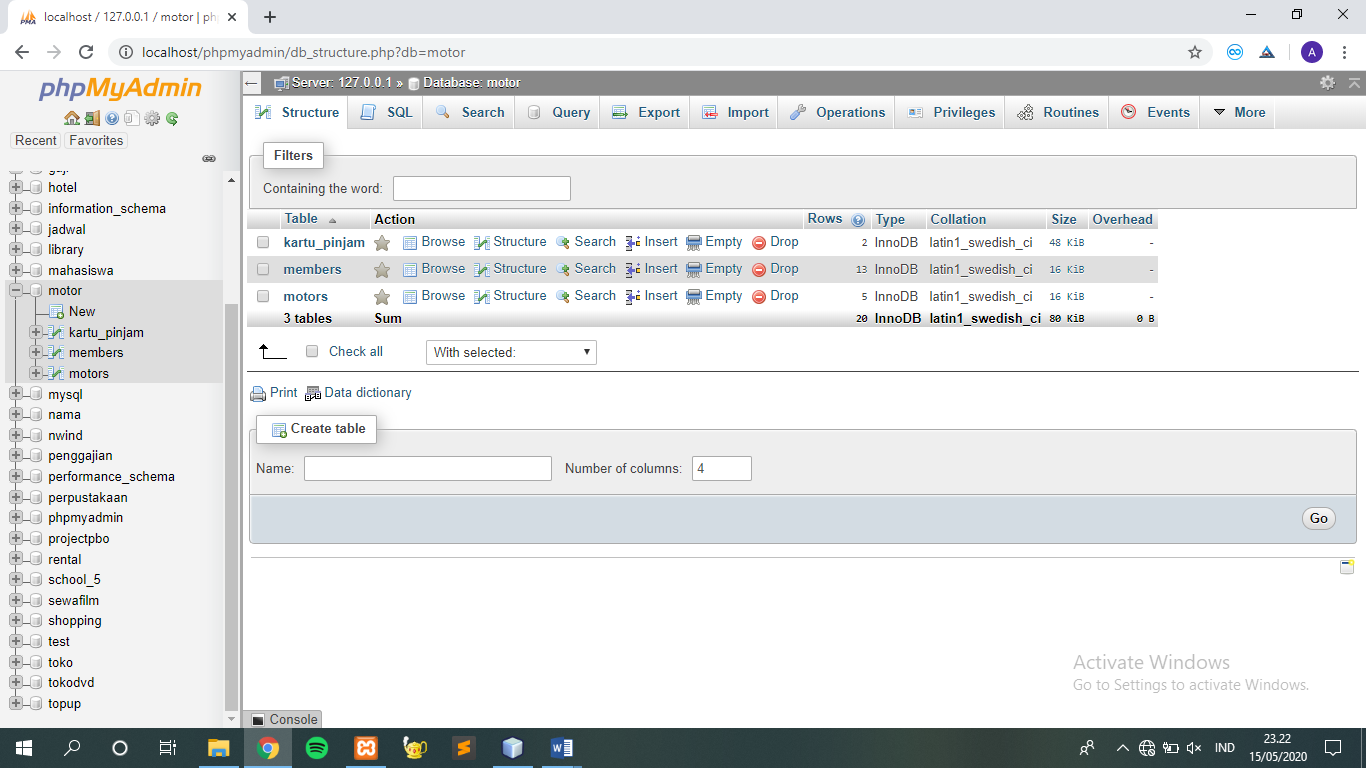
**3.2. Deskripsi Umum Project Akhir**

Aplikasi rental kendaraan ini memiliki 2 interface yaitu admin dan pelanggan yang sudah mendaftar (member). Diperlukan login untuk masuk kedalam aplikasi baik untuk pelanggan maupun admin. Jika pelanggan belum memiliki akun, dapat mendaftar terlebih dahulu. Pada interfate pelanggan, pelanggan dapat melihat kendaraan apa saja yang disewakan dan juga ketersediaan kendaraan tersebut. Pelanggan dapat menyewa dan mengembalikan kendaraan tersebut.

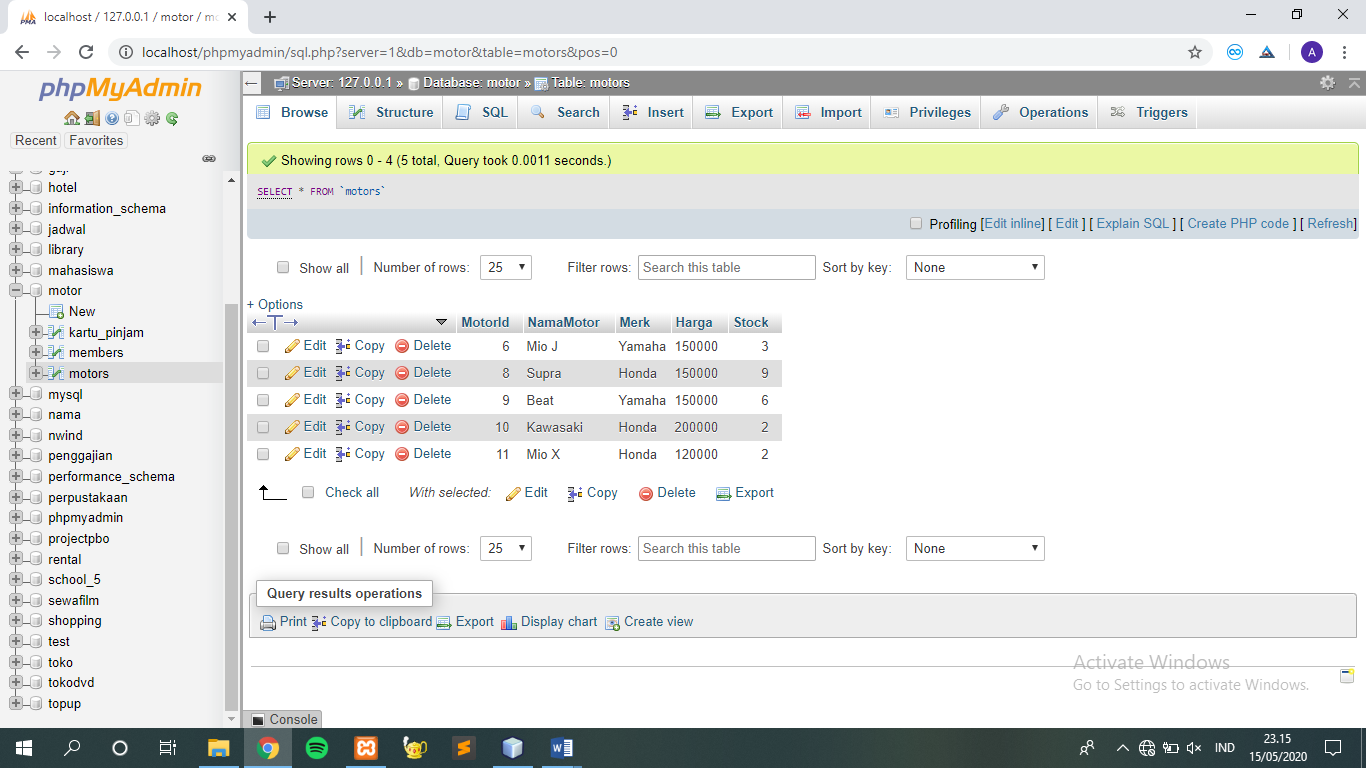
Pada interface admin, admin dapat menambahkan dan menghapus kendaraan yang disewakan, mengedit data pelanggan dan juga melihat riwayat penyewaan kendaraan.

**3.3. Inti Pembahasan**

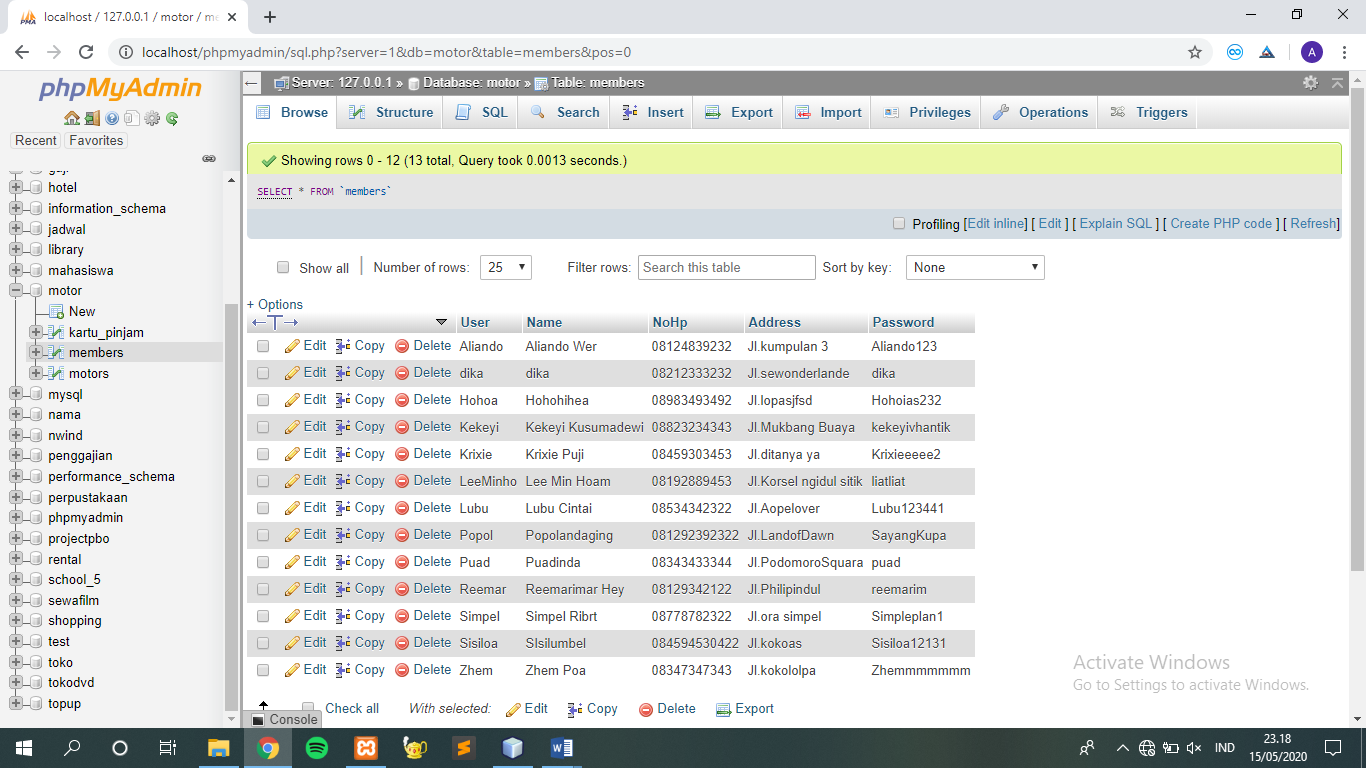
* 1. Database



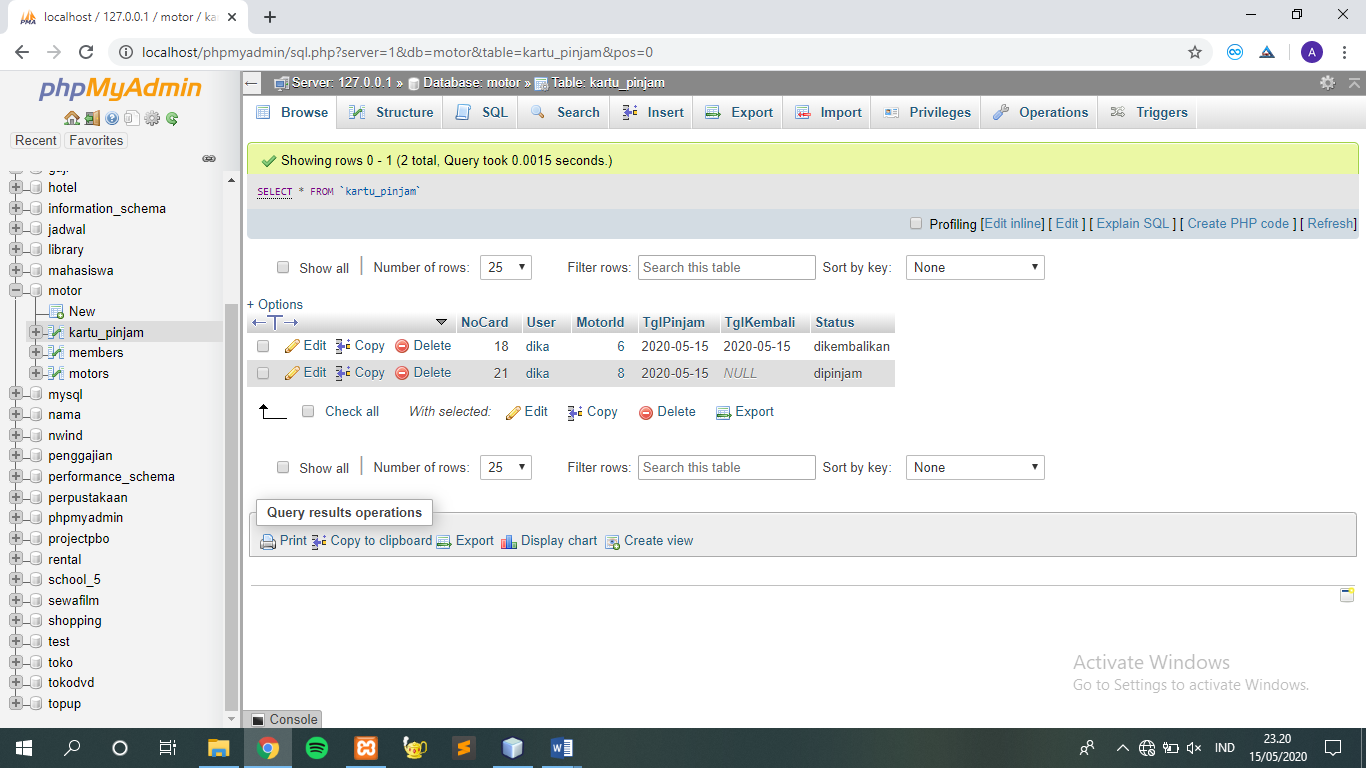
**Gambar 3.1.** Database motor



**Gambar 3.2.** Tabel motors



**Gambar 3.3.** Tabel members



**Gambar 3.4.** Tabel kartu\_pinjam

* 1. Login

|  |
| --- |
| public int insertMember(String user, String name, String noHp, String address, String password){  int count = 0;  try {  if (checkAlreadyMembers(user) == 0){  String query = "INSERT INTO members VALUES" +  "('" + user + "','" + name + "','" + noHp + "','" + address + "','" + password + "')";  statement = (Statement) Db.getConnection().createStatement();  statement.executeUpdate(query);  count = 1;  }  } catch (SQLException e) {  e.printStackTrace();  System.out.println(e.getMessage());  }  return count;  } |

**Listing 3.1.** Model Input member

|  |
| --- |
| public void signUp(){  String user = viewLogin.getUser1();  String name = viewLogin.getName();  String noHp = viewLogin.getNoHp();  String address = viewLogin.getAddress();  String password = viewLogin.getPassword1();  if (user.equals("") || name.equals("") || noHp.equals("") || address.equals("") || password.equals("") ){  JOptionPane.showMessageDialog(null, "Data Tidak Boleh Kosong");  } else {  if (modelSignIn.insertMember(user,name,noHp,address,password) == 1){  JOptionPane.showMessageDialog(null,"Berhasil Daftar!");  viewLogin.moveSignIn();  } else {  JOptionPane.showMessageDialog(null,"User Telah Terdaftar!");  }  }  } |

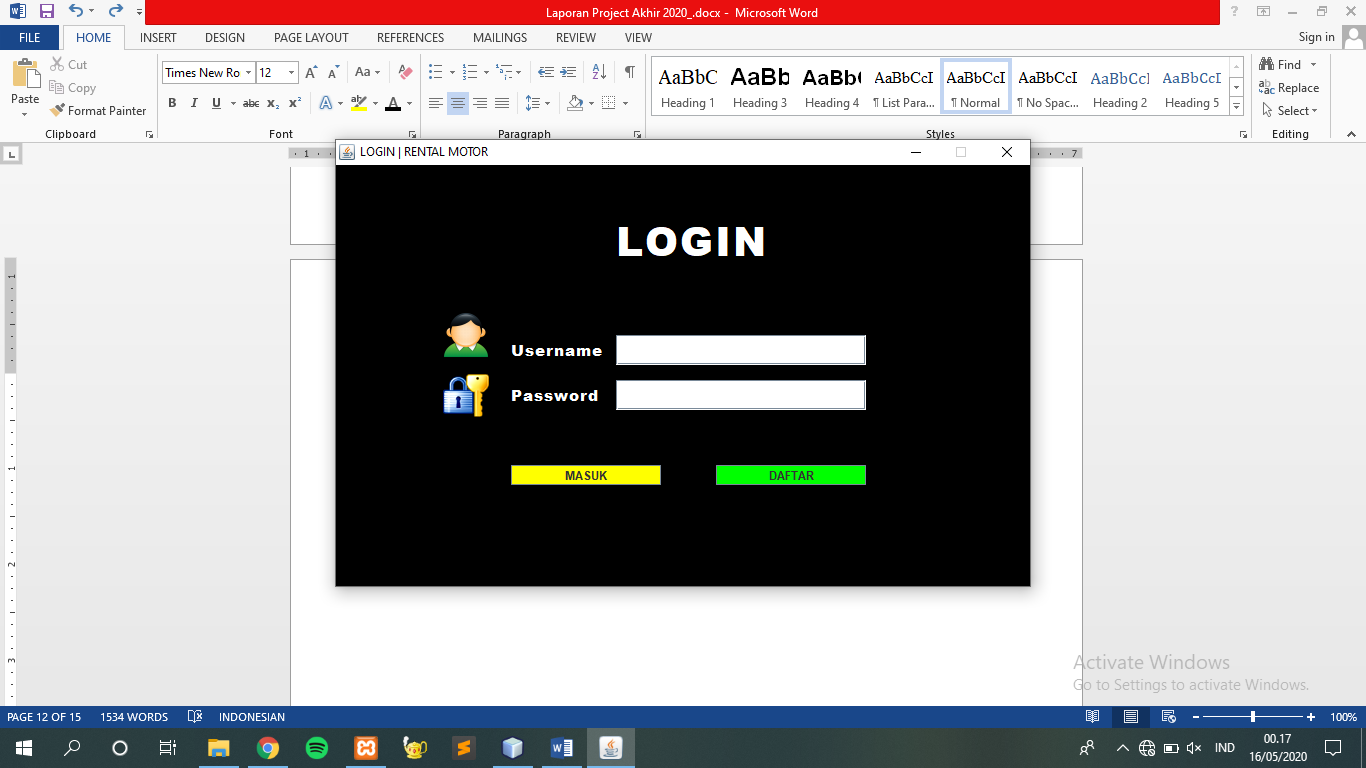
**Listing 3.2.** Controller Input member

|  |
| --- |
| public String[] getMember(String user, String password){  String data[] = new String[2];  try {  String query = "SELECT \* FROM members " +  "WHERE User = '" + user + "' AND Password = '" + password + "'";  statement = (Statement) Db.getConnection().createStatement();  ResultSet resultSet = statement.executeQuery(query);  if (resultSet.next()){  data[0] = resultSet.getString("User");  data[1] = resultSet.getString("Name");  }  } catch (SQLException e) {  } |

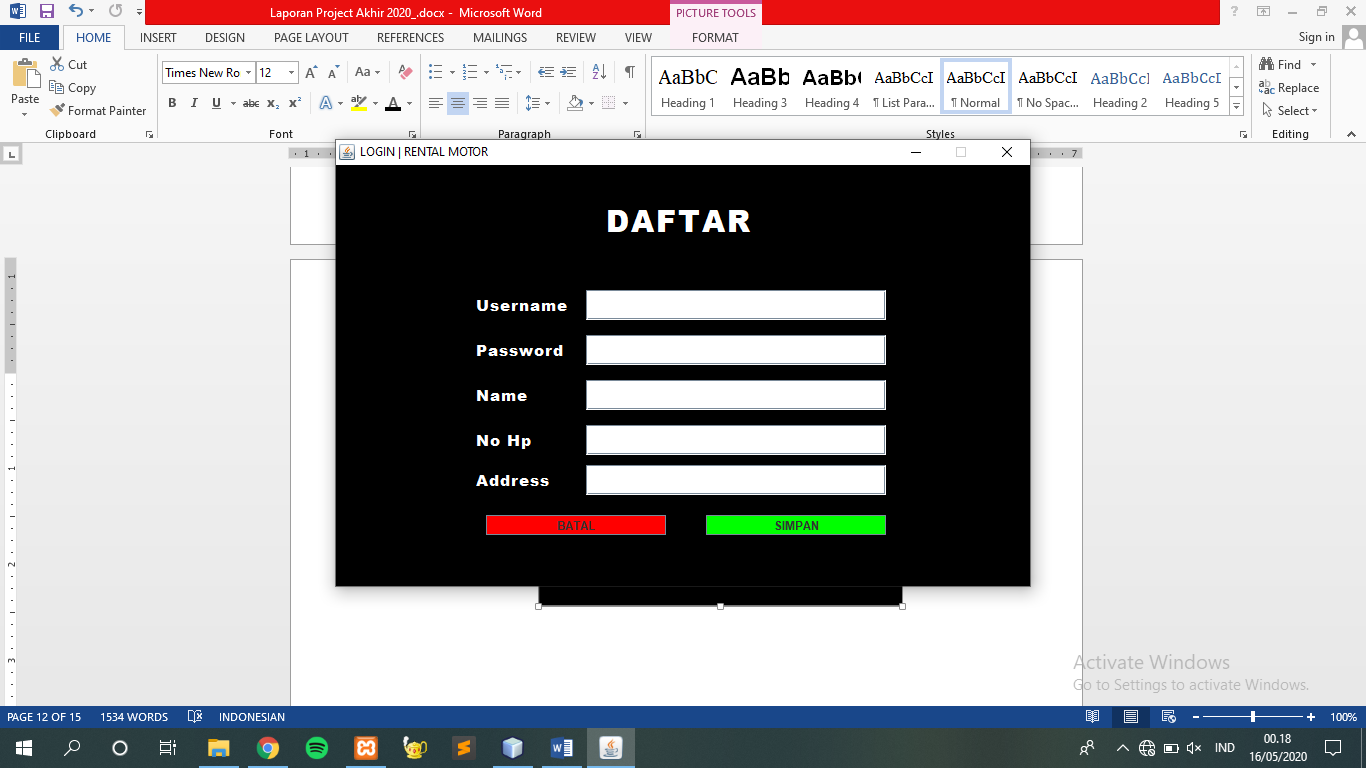
**Listing 3.3.** Model Login

|  |
| --- |
| public void signIn(){  String user = viewLogin.getUser();  String password = viewLogin.getPassword();  if (user.equals("admin") && password.equals("admin")){  JOptionPane.showMessageDialog(null, "Berhasil Masuk!");  viewLogin.setVisible(false);  MVC\_Menu\_Admin mvcMenuAdmin = new MVC\_Menu\_Admin();  } else {  if (user.equals("") || password.equals("")) {  JOptionPane.showMessageDialog(null, "Data Tidak Boleh Kosong");  } else {  if (modelSignIn.getCountForSignIn(user, password) == 1) {  JOptionPane.showMessageDialog(null, "Berhasil Masuk!");  viewLogin.setVisible(false);  String data[] = modelSignIn.getMember(user, password);  Pelanggan member = new Pelanggan(data[0], data[1]);  MVC\_Menu\_Pelanggan mvcMenuMember = new MVC\_Menu\_Pelanggan(member);  } else {  JOptionPane.showMessageDialog(null, "User / Password Salah");  }  }  }  } |

**Listing 3.4.** Controller Login



**Gambar 3.5.** Login



**Gambar 3.6.** Input Member

* 1. Pinjam



**Gambar 3.7.** Pinjam

|  |
| --- |
| public int pinjam(String motorId){  int result = 0;  try {  if (checkPinjam(motorId) == 1){  DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd");  LocalDateTime now = LocalDateTime.now();  String date = dtf.format(now);  String query = "INSERT INTO kartu\_pinjam VALUES " +  "(DEFAULT, '" + member.getUser() + "', " +  motorId + ", '" + date + "' , DEFAULT ,'dipinjam')";  statement = Db.getConnection().createStatement();  statement.executeUpdate(query);  updateMotor(motorId);  result = 1;  } |

**Listing 3.5.** Model pinjam

|  |
| --- |
| viewMemberPinjam.table.addMouseListener(new MouseAdapter() {  @Override  public void mouseClicked(MouseEvent e) {  super.mouseClicked(e);  int row = viewMemberPinjam.table.getSelectedRow();  String id = viewMemberPinjam.table.getValueAt(row,0).toString();  String motor = viewMemberPinjam.table.getValueAt(row,2).toString();  int confirm = JOptionPane.showConfirmDialog(null,  "Apakah Mau Meminjam Motor " + motor + " ?", "Konfirmasi" , JOptionPane.YES\_NO\_OPTION);  if (confirm == 0) {  if (modelMemberPinjam.pinjam(id) == 1)  JOptionPane.showMessageDialog(null,  "Berhasil Meminjam Motor " + motor);  else  JOptionPane.showMessageDialog(null,  "Gagal Meminjam Motor " + motor);  refreshTable();  } else  JOptionPane.showMessageDialog(null, "Tidak Jadi Dipinjam");  }}); |

**Listing 3.6.** Controller Pinjam

* 1. Pengembalian

|  |
| --- |
| public int kembali(String noCard){  int result = 0;  try {  DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd");  LocalDateTime now = LocalDateTime.now();  String date = dtf.format(now);  String query = "UPDATE kartu\_pinjam " +  "SET Status = 'dikembalikan', TglKembali = '" + date + "' " +  "WHERE NoCard = " + noCard;  statement = Db.getConnection().createStatement();  statement.executeUpdate(query);  updateBooks(noCard);  result = 1;  } catch (SQLException e) {  e.printStackTrace();  System.out.println(e.getMessage());  }  return result;  } |

**Listing 3.7.** Model Pengembalian

|  |
| --- |
| public void actionPerformed(ActionEvent e) {  if (e.getSource() == v\_p\_kembali.bSearch){  String book = v\_p\_kembali.getSearch();  String data[][] = modelMemberKembali.getBookSearch(book);  v\_p\_kembali.table.setModel(new JTable(data, v\_p\_kembali.coloumnName).getModel()); v\_p\_kembali.table.getColumnModel().getColumn(0).setMinWidth(0); v\_p\_kembali.table.getColumnModel().getColumn(0).setMaxWidth(0);  v\_p\_kembali.table.getColumnModel().getColumn(0).setWidth(0);  } else if (e.getSource() == v\_p\_kembali.bRefresh)  refreshTable();  } |

**Listing 3.8.** Controller Pengembalian



**Gambar 3.8.** Pengembalian

* 1. Pelanggan

|  |
| --- |
| public int updateMember(String user, String name, String noHp, String address, String password){  int result = 0;  try {  String query = "UPDATE members SET " +  "Name = '" + name + "', NoHp = '" + noHp + "', " +  "Address = '" + address + "', Password = '" + password + "' " +  "WHERE User = '" + user + "'";  statement = Db.getConnection().createStatement();  statement.executeUpdate(query);  result = 1;  } catch (SQLException e) {  e.printStackTrace();  System.out.println(e.getMessage());  }  return result;  } |

**Listing 3.9.** Model Update Pelanggan

|  |
| --- |
| public int deleteMember(String user){  int result = 0;  try {  String query = "DELETE FROM members WHERE User = '" + user + "'";  statement = Db.getConnection().createStatement();  statement.executeUpdate(query);  result = 1;  } catch (SQLException e) {  e.printStackTrace();  System.out.println(e.getMessage());  }  return result;  } |

**Listing 3.10.** Model Delete Pelanggan

|  |
| --- |
| if (user.equals(“”) || password.equals(“”) || name.equals(“”) || address.equals(“”) || noHp.equals(“”)){  JoptionPane.showMessageDialog(null, “Data Tidak Boleh Kosong”);  } else {  if (modelAdminMember.updateMember(user, name, noHp, address, password) == 1)  JoptionPane.showMessageDialog(null, “Update Berhasil”);  else  JoptionPane.showMessageDialog(null, “Update Gagal”);  }  refreshTable();  reset();  } else if (e.getSource() == viewAdminMember.bDelete){  String user = viewAdminMember.getUser();  if (user.equals(“”)){  JoptionPane.showMessageDialog(null, “User Tidak Boleh Kosong”);  } else {  if (modelAdminMember.deleteMember(user) == 1)  JoptionPane.showMessageDialog(null, “Delete Berhasil”);  else  JoptionPane.showMessageDialog(null, “Delete Gagal”);  } |

**Listing 3.11**. Controller Pelanggan



**Gambar 3.9.** Pelanggan

* 1. Data Motor

|  |
| --- |
| public int updateMotor(String motorId, String namaMotor, String merk, String harga, String stock){  int result = 0;  try {  String query = "UPDATE motors SET " +  "NamaMotor = '" + namaMotor + "', Merk = '" + merk + "', Harga = '" + harga + "', Stock = " + stock + " " +  "WHERE MotorId = " + motorId;  statement = Db.getConnection().createStatement();  statement.executeUpdate(query);  result = 1;  } catch (SQLException e) {  e.printStackTrace();  System.out.println(e.getMessage());  result = 0;  }  return result;  } |

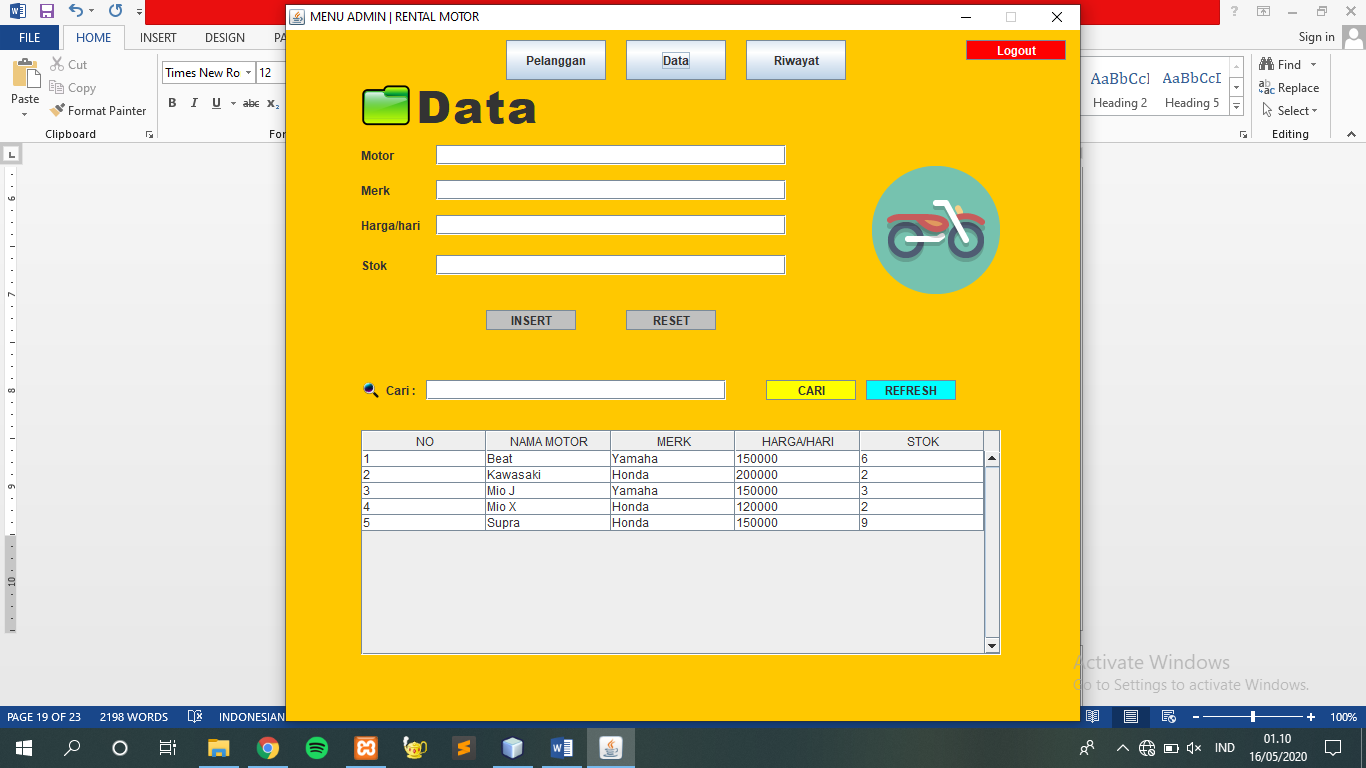
**Listing 3.12**. Model Update Data

|  |
| --- |
| public int deleteMotor(String motorId){  int result = 0;  try {  String query = "DELETE FROM motors WHERE MotorId = " + motorId;  statement = Db.getConnection().createStatement();  statement.executeUpdate(query);  result = 1;  } |

**Listing 3.13**. Model Data

|  |
| --- |
| public void actionPerformed(ActionEvent e) {  if (e.getSource() == v\_admin\_data.bInsert){  insertMotor();  } else if (e.getSource() == v\_admin\_data.bReset){  reset();  } else if (e.getSource() == v\_admin\_data.bSearch){  String motor = v\_admin\_data.getSearch();  String data[][] = modelAdminData.getMotorSearch(motor);  v\_admin\_data.table.setModel(new JTable(data, v\_admin\_data.coloumnName).getModel());  v\_admin\_data.table.getColumnModel().getColumn(0).setMinWidth(0);  v\_admin\_data.table.getColumnModel().getColumn(0).setMaxWidth(0);  v\_admin\_data.table.getColumnModel().getColumn(0).setWidth(0);  } else if (e.getSource() == v\_admin\_data.bRefresh){  refreshTable();  }  } |

**Listing 3.14**. Controller Data



**Gambar 3.10.** Data

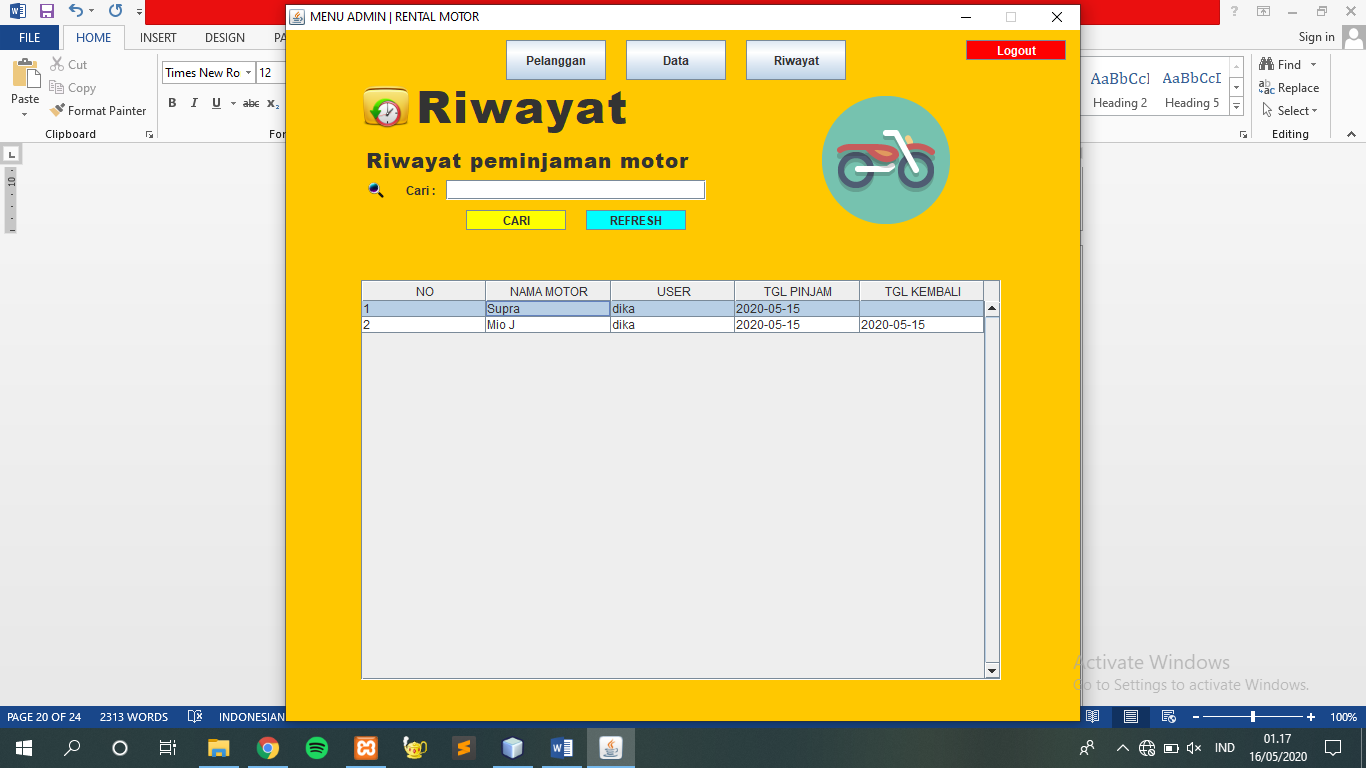
* 1. Riwayat

|  |
| --- |
| public String[][] getAllMotor(){  String[][] data = new String[getCountBooks()][5];  try{  String query = "SELECT \* FROM (kartu\_pinjam k JOIN motors b ON k.MotorId = b.MotorId) JOIN members m ON k.User = m.User " +  "ORDER BY k.NoCard DESC";  statement = Db.getConnection().createStatement();  ResultSet resultSet = statement.executeQuery(query);  int i = 0;  while (resultSet.next()){  data[i][0] = "" + (i+1);  data[i][1] = resultSet.getString("NamaMotor");  data[i][2] = resultSet.getString("User");  data[i][3] = resultSet.getString("TglPinjam");  data[i][4] = resultSet.getString("TglKembali");  i++;  } |

**Listing 3.15**. Model Riwayat

|  |
| --- |
| public C\_Admin\_History(V\_Admin\_History viewAdminHistory, M\_Admin\_History modelAdminHistory) {  this.viewAdminHistory = viewAdminHistory;  this.modelAdminHistory = modelAdminHistory;  if (modelAdminHistory.getCountBooks() != 0)  refreshTable();  viewAdminHistory.bSearch.addActionListener(this);  viewAdminHistory.bRefresh.addActionListener(this);  } |

**Listing 3.16**. Controller Riwayat



**Gambar 3.11.** Riwayat

**BAB IV**

**JADWAL PENGERJAAN DAN PEMBAGIAN TUGAS**

Tabel 1 Jadwal Pengerjaan

|  |  |  |
| --- | --- | --- |
| **Tanggal** | **Keterangan** | **Pembagian** |
| 8 MEI 2020 | Perancangan Awal | Fuad & Dika |
| 9 Mei 2020 | Pembuatan SQL | Dika |
| 10 Mei 2020 | Pembuatan Program (admin) | Dika |
| 10 Mei 2020 | Pembuatan Program (user) | Fuad |
| 15 Mei 2020 | Pembuatan Laporan | Fuad |

**BAB V**

**KESIMPULAN & SARAN**

1. **Kesimpulan**

Aplikasi tersebut berguna untuk memudahkan bisnis antara pemilik rental dan pelanggan yang ingin menyewa kendaraan. pelanggan dapat melihat ketersediaan kendaraan sewaan dan menyewa kendaraan tersebut secara daring. Jadi tidak perlu lagi datang ketempat penyewaan kendaraan.

1. **Saran**

Diharapkan pelanggan dan admin dapat menggunakan aplikasi tersebut sesuai dengan tujuan awal pembuatan aplikasi

**DAFTAR PUSTAKA**

2020.*Modul Praktikum Pemrograman Berorientasi Objek.*Yogyakarta. UPN “Veteran” Yogyakarta