

T-rentor 汽车租赁管理系统 说明书

一. 需求分析

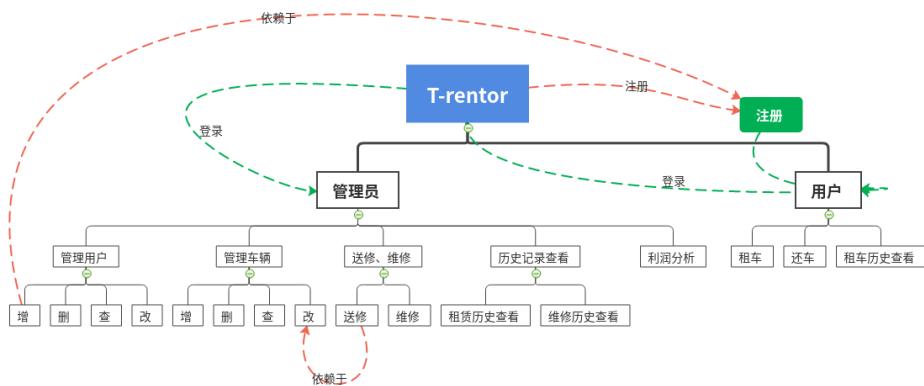
本系统是一个面向群众的本地汽车租赁系统。由于人们生活水平的提高，在外出旅游、婚宴喜事喝日常生活中租赁汽车的需求越来越旺盛，为了解决群众便捷安全的租车需求，我们上线了这款汽车租赁系统。

对于用户：在此系统中，新用户需要注册一个账号，为了保障用户租赁服务的安全，用户除了设置用户名和密码之外，还需要绑定真实姓名、电话号码和身份证号信息。登录系统后，在菜单栏中有租车、还车、租车历史以及作者四个模块。在首页中用户可以看到所有「可租」车辆，用户租车的完整流程为：点击租车（如果有未还车辆，不可租车）→ 点击想要租赁的汽车 → 即可租车。用户需要按时归还汽车，点击还车，再点击所要还的车辆，即可完成完整的租车流程。

对于管理员：功能有：注册用户管理、轿车管理、送修、维修、租车历史查看、车辆维修历史查看和利润分析。管理员可以「增删改查」用户、车辆信息，可以查看租车历史和维修历史，还能统计一段时间内的销售额和利润。

二. 功能结构

功能结构图 (使用 在线网站生成)



文件结构 (使用 Linux 插件 Tree 构建)

```
1 .
2 └── [4.0K Aug 30 14:54] graphs
3   └── [174K Aug 30 14:54] db_RC.png
4 └── [4.0K Aug 30 21:07] images
5 └── [4.0K Aug 30 15:14] jdbc
6   └── [2.3M Aug 20 21:30] mysql-connector-java-8.0.26.jar
```

```
7 └── [4.0K Sep 5 14:06] teko7a
8   ├── [4.0K Sep 5 13:32] DAOs
9     |   ├── [2.1K Sep 2 14:43] AdminDAO.java
10    |   ├── [7.0K Sep 4 22:28] CardDAO.java
11    |   ├── [4.0K Sep 5 13:22] MaintainDAO.java
12    |   ├── [3.3K Sep 4 21:58] RentedDAO.java
13    |   ├── [4.0K Aug 30 16:47] testClass
14    |   |   └── [3.2K Aug 30 16:47] Test_usrDao.java
15    |   └── [6.7K Sep 5 13:32] UsrDAO.java
16   └── [4.0K Sep 3 9:14] Frames
17     |   ├── [4.0K Sep 5 14:01] Admin
18     |   |   └── [114K Sep 5 14:01] MainFrm.java
19     |   ├── [4.0K Sep 1 14:53] testFrms
20     |   |   ├── [ 287 Sep 1 14:53] testAdminFrm.java
21     |   |   └── [ 287 Aug 30 21:33] testUsrMainFrm.java
22     |   ├── [4.0K Sep 2 20:03] Usr
23     |   |   └── [ 59K Sep 2 20:03] MainFrm.java
24     |   └── [4.0K Sep 5 13:44] Welcome
25       ├── [11K Sep 1 14:51] LoginFrm.java
26       └── [13K Sep 5 13:44] RegisterFrm.java
27   └── [4.0K Sep 4 22:21] Models
28     ├── [1.3K Aug 30 20:20] Admin.java
29     ├── [4.3K Sep 4 22:21] Car.java
30     ├── [2.5K Sep 4 20:55] Maintain.java
31     ├── [2.0K Aug 30 16:40] Person.java
32     ├── [2.4K Sep 2 15:23] Rented.java
33     └── [1.5K Aug 30 20:20] Usr.java
34 └── [ 271 Aug 30 19:37] T_rentor.java
35   └── [4.0K Sep 4 18:05] Utils
36     ├── [2.0K Aug 30 15:14] DataBaseUtil.java
37     ├── [2.2K Aug 30 15:18] DateUtil.java
38     ├── [1.0K Aug 25 20:40] FrmUtil.java
39     ├── [ 197 Sep 2 21:13] OrdersEnum.java
40     ├── [2.5K Sep 4 18:05] StringUtil.java
41     └── [1.5K Aug 30 15:13] UrlLinkUtil.java
42
43 13 directories, 68 files
```

三.类设计及类图

设计模式

DAO 设计，也即 Database Access Object. 其核心内涵是将数据库操作进行封装，同时数据库每一个表都有对应的实现类。

文件结构（使用 Linux Tree 插件构建）

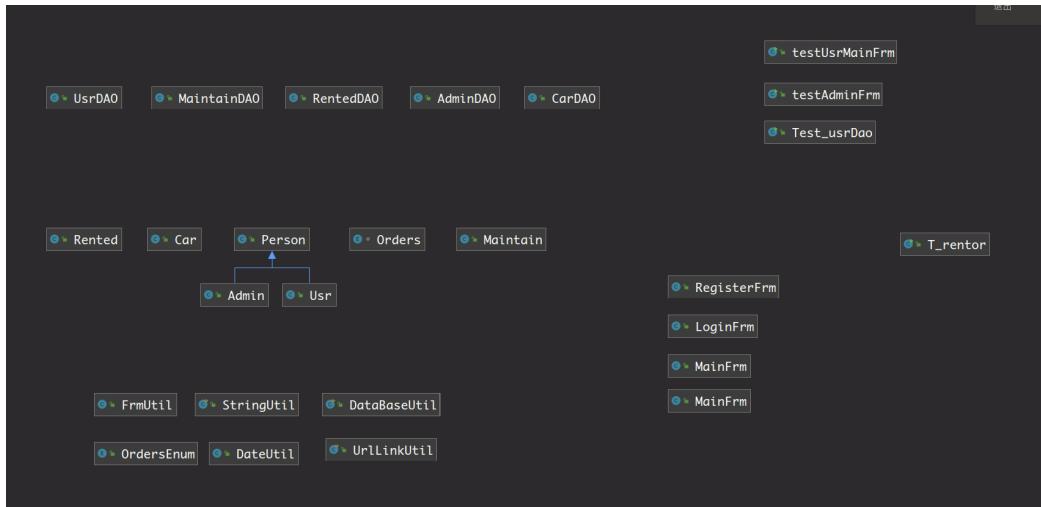
```
1 .
2   ├── graphs 存放所有的类图
3   ├── images 存放所有的图片
4   ├── jdbc jdbc 依赖 jar 包
5   └── teko7a 源代码
6     ├── DAOs dao层
7     |   └── testClass 测试类
8     ├── Frames 框架（Swing 与 所有功能实现）
9     |   ├── Admin 管理员界面
10    |   ├── testFrms 测试类
11    |   ├── Usr 用户界面
12    |   └── Welcome 注册登录界面
13    ├── Models 数据库表实体
14    └── Utils 工具类
```

类文件结构（使用 Linux Tree 插件构建）

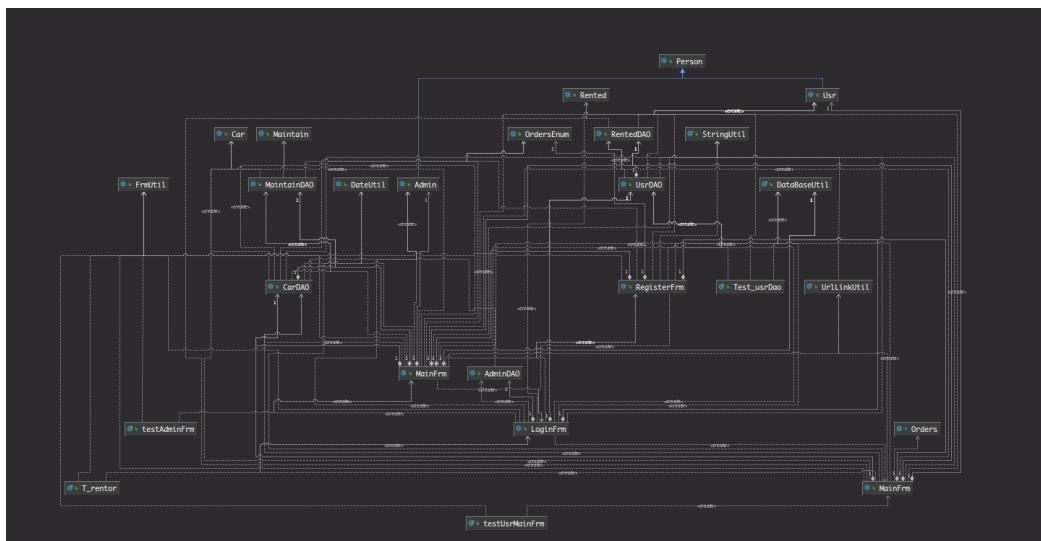
```
1 └── teko7a
2   ├── DAOs 「异常」
3   |   ├── AdminDAO.java 管理员 Dao 「异常」
4   |   ├── CarDAO.java 车 Dao 「异常」
5   |   ├── MaintainDAO.java 维修 Dao 「异常」
6   |   ├── RentedDAO.java 租赁 Dao 「异常」
7   |   ├── testClass 测试类
8   |   |   └── Test_usrDao.java
9   |   └── UsrDAO.java 用户 Dao 「异常」
10  └── Frames
11  |   ├── Admin
12  |   |   └── MainFrm.java 管理员主界面 「异常」 「集合」 「范型」 「注解」
13  |   ├── testFrms
14  |   |   ├── testAdminFrm.java 测试 管理员 主界面
15  |   |   └── testUsrMainFrm.java 测试 用户 主界面
16  |   ├── Usr
17  |   |   └── MainFrm.java 测试 用户 主界面 「注解」 「异常」 「集合」 「范型」
18  |   └── Welcome
19  |       ├── LoginFrm.java 登录 主界面 「异常」 「集合」
20  |       └── RegisterFrm.java 注册 主界面 「异常」 「集合」
21  └── Models
22  |   ├── Admin.java 管理员 实体 「继承」
23  |   ├── Car.java 车 实体
24  |   ├── Maintain.java 维修记录条 实体
25  |   ├── Person.java 人 实体，是 管理员 与 用户 的父类 「继承」
26  |   ├── Rented.java 租车条 实体
27  |   └── Usr.java 用户 实体 「继承」
28  └── T_rentor.java 系统入口
29  └── Utils
30  |   ├── DataBaseUtil.java 数据库 工具类 「反射」 「异常」
31  |   ├── DateUtil.java 日期 工具类 「Date类」
32  |   ├── FrmUtil.java 界面 工具类 「Swing」
33  |   └── OrdersEnum.java 辅助类，标记 Enum Class 「枚举」
```

34 ┌── StringUtil.java 字符串工具类
35 └── UrlLinkUtil.java 打开链接 工具类 「异常」

类图（使用 IntelliJ Idea 进行构建）



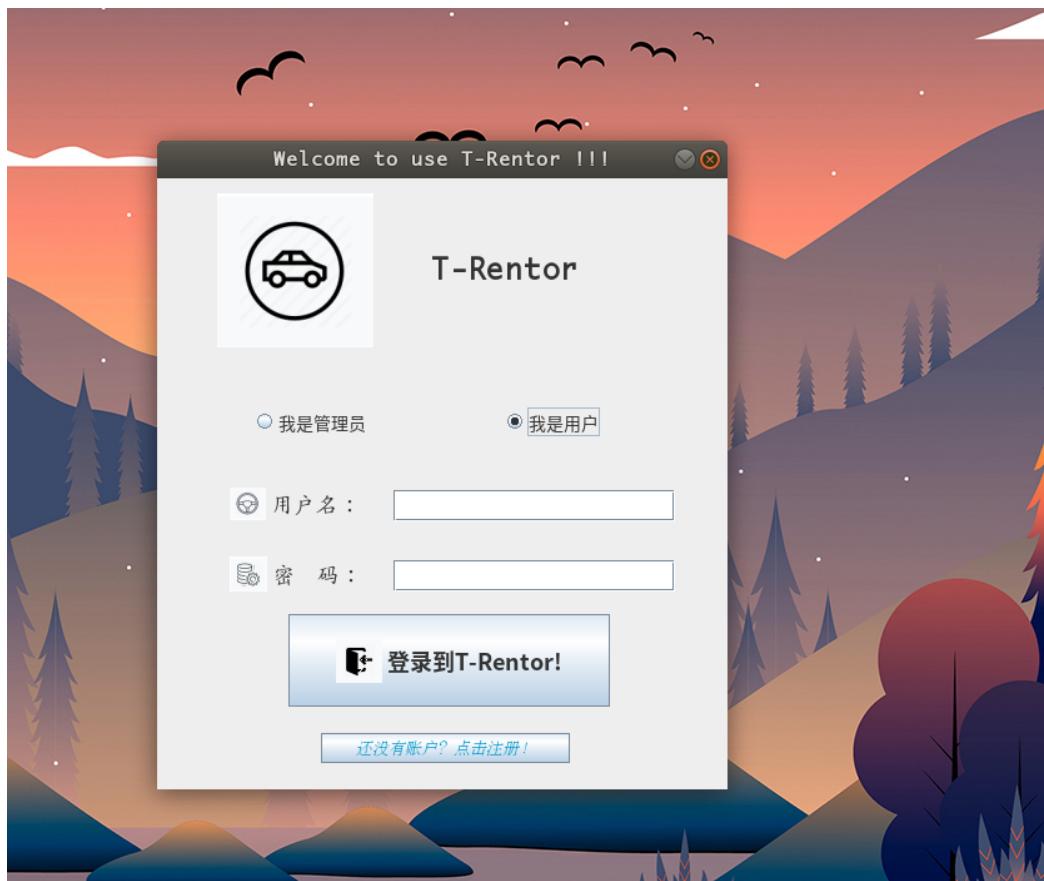
各类依赖关系



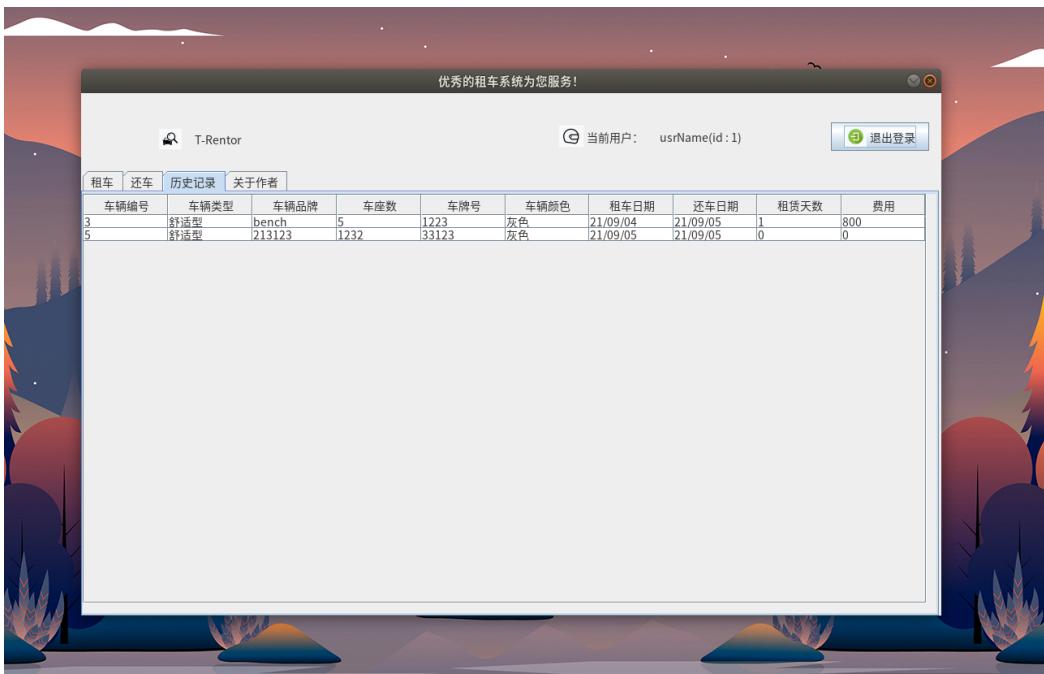
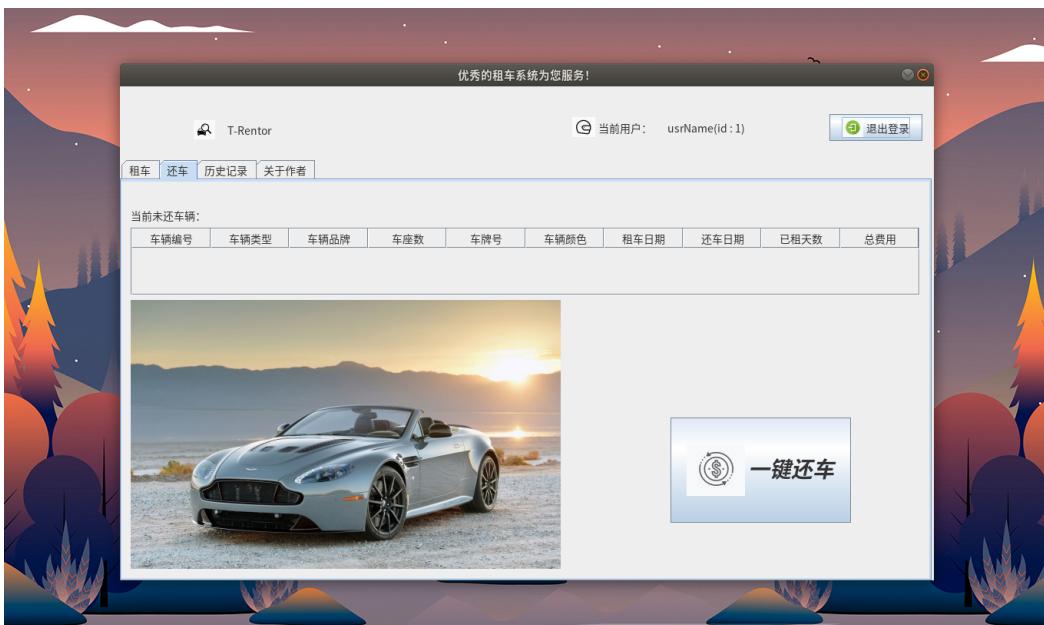
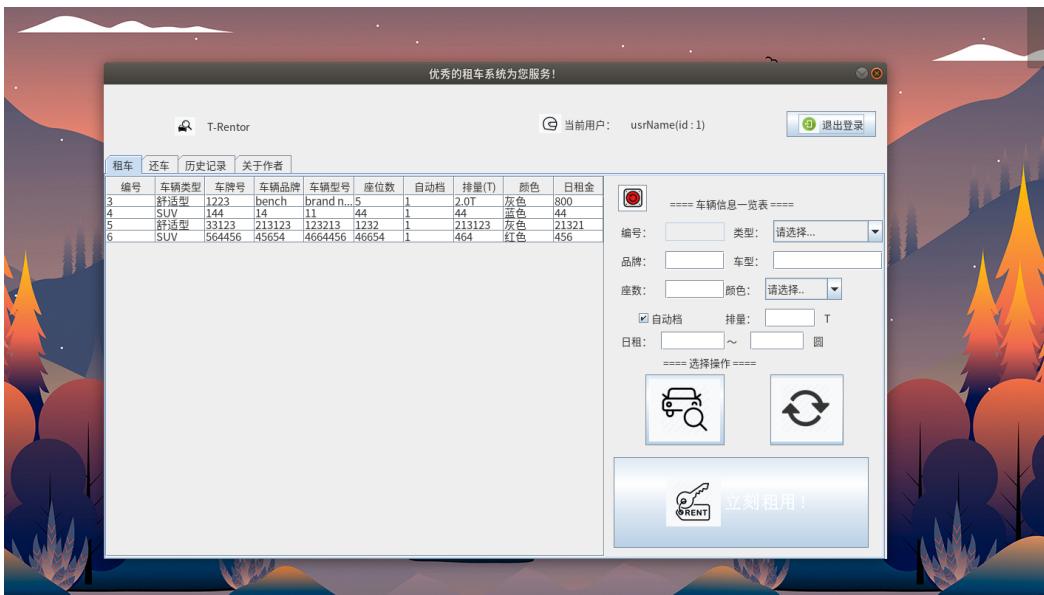
四. 界面设计

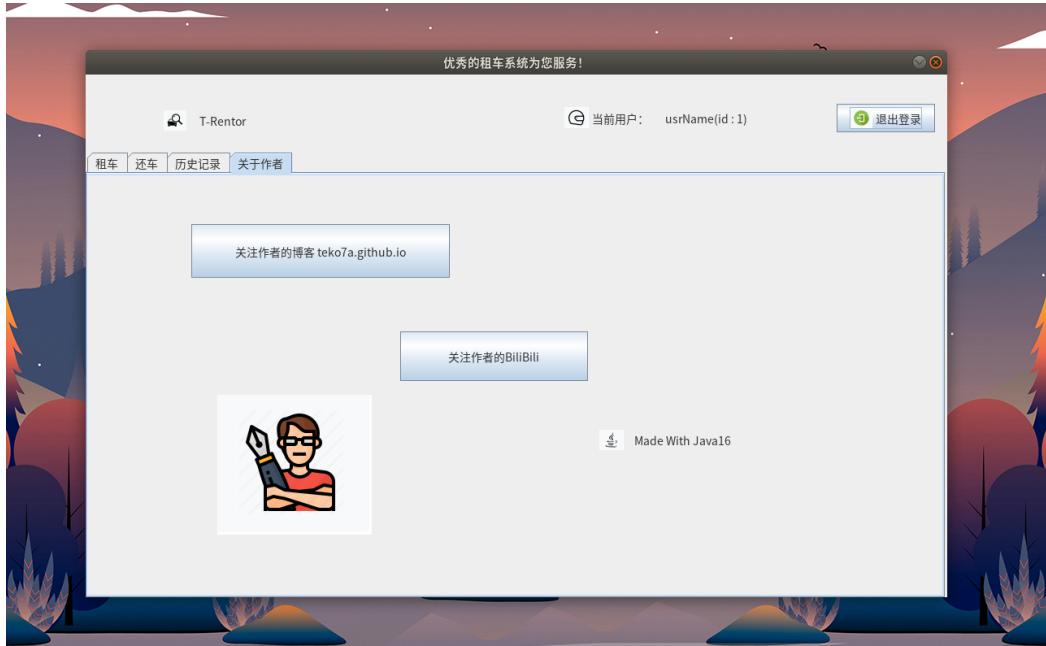
-
- 使用 Swing GUI

登录、注册界面

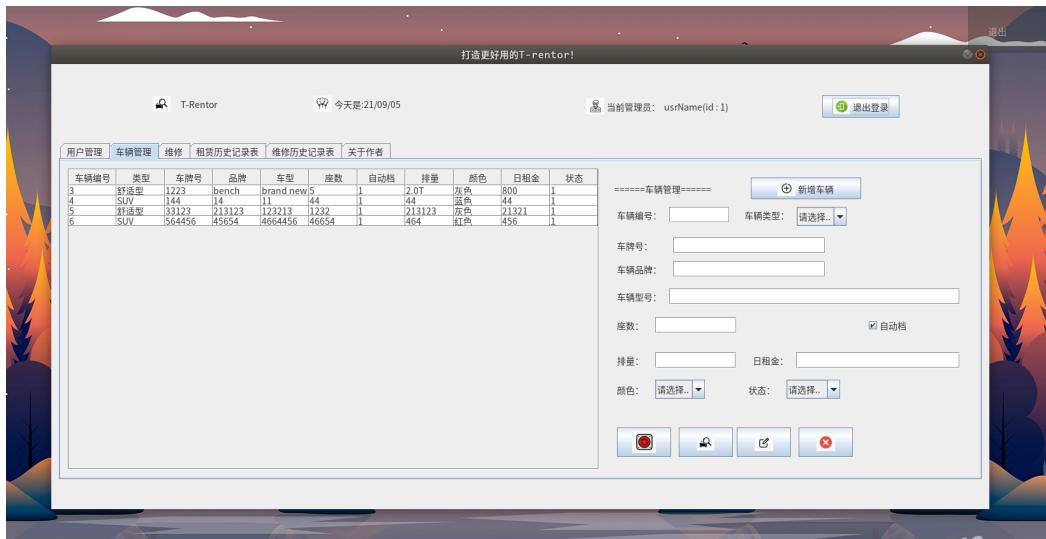
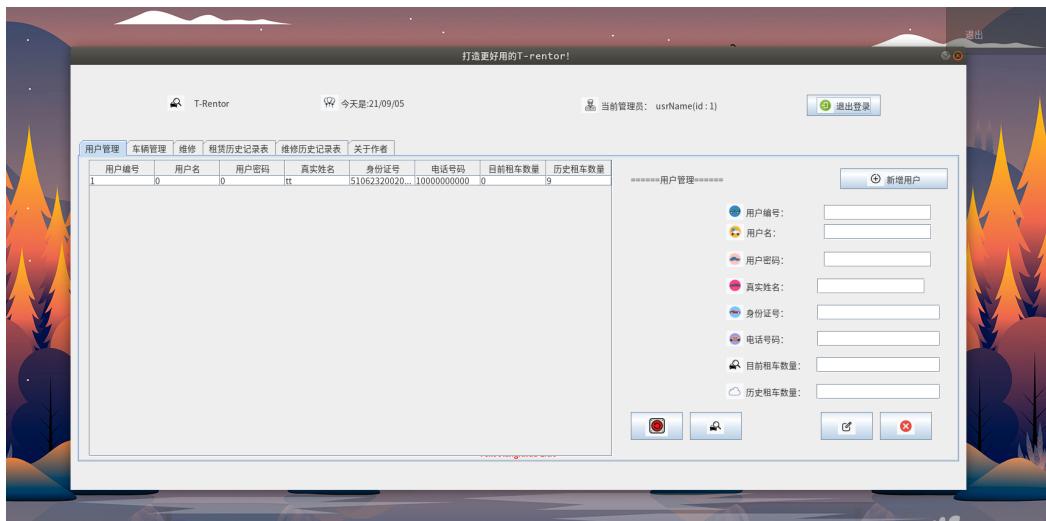


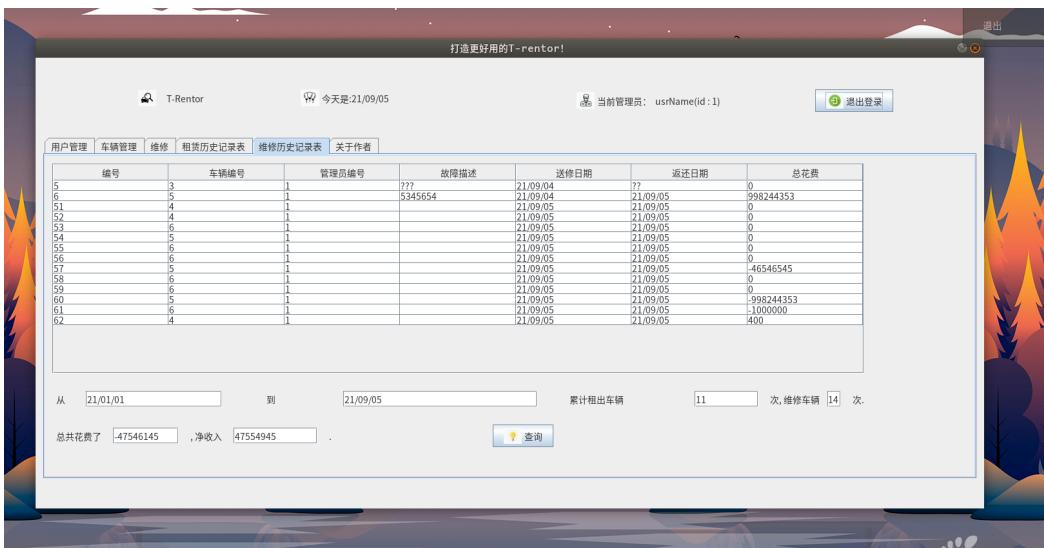
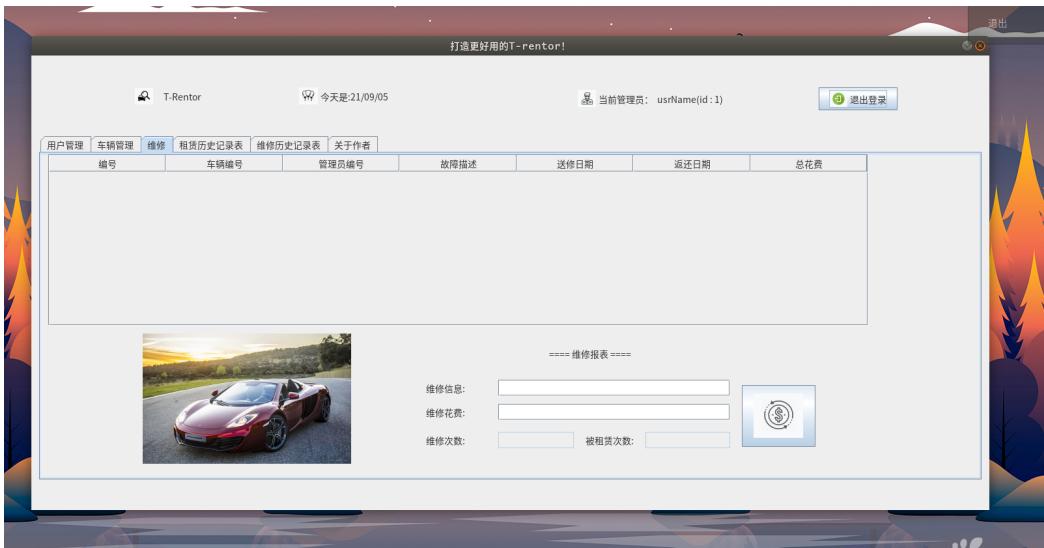
用户界面





管理员界面





五. 数据库表（使用终端输出）

```
mysql> desc t_usr;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra      |
+-----+-----+-----+-----+-----+
| id    | int(11) | NO   | PRI | NULL    | auto_increment |
| name  | varchar(30)| YES  |     | NULL    | |
| password | varchar(20)| YES  |     | NULL    | |
| realName | varchar(20)| YES  |     | NULL    | |
| idc   | varchar(18)| YES  |     | NULL    | |
| tel   | varchar(11)| YES  |     | NULL    | |
+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

```
mysql> desc t_car;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra      |
+-----+-----+-----+-----+-----+
| id    | int(11) | NO   | PRI | NULL    | auto_increment |
| type  | varchar(30)| YES  |     | NULL    | |
| license | varchar(10)| YES  |     | NULL    | |
| brand | varchar(20)| YES  |     | NULL    | |
| size  | varchar(20)| YES  |     | NULL    | |
| seatNum | int(11) | YES  |     | NULL    | |
| isAuto | int(11) | YES  |     | 1       | |
| tons  | varchar(20)| YES  |     | NULL    | |
| color | varchar(20)| YES  |     | NULL    | |
| cost   | int(11) | YES  |     | NULL    | |
| status | int(11) | YES  |     | NULL    | |
+-----+-----+-----+-----+-----+
11 rows in set (0.01 sec)
```

```
mysql> desc t_admin;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra      |
+-----+-----+-----+-----+-----+
| id    | int(11) | NO   | PRI | NULL    | auto_increment |
| name  | varchar(30)| YES  |     | NULL    | |
| password | varchar(20)| YES  |     | NULL    | |
| realName | varchar(20)| YES  |     | NULL    | |
| idc   | varchar(18)| YES  |     | NULL    | |
| tel   | varchar(11)| YES  |     | NULL    | |
+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

```
mysql> desc t_maintain;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra      |
+-----+-----+-----+-----+-----+
| id    | int(11) | NO   | PRI | NULL    | auto_increment |
| carID | int(11) | YES  | MUL | NULL    | |
| adminID | int(11) | YES  | MUL | NULL    | |
| errorDesc | varchar(50)| YES  |     | NULL    | |
| stDate | varchar(20)| YES  |     | NULL    | |
| edDate | varchar(20)| YES  |     | NULL    | |
| cost   | int(11) | YES  |     | NULL    | |
+-----+-----+-----+-----+-----+
7 rows in set (0.00 sec)
```

```

mysql> desc t_rented;
+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+
| id    | int(11)   | NO   | PRI | NULL    | auto_increment |
| carID | int(11)   | YES  | MUL | NULL    |                |
| usrID | int(11)   | YES  | MUL | NULL    |                |
| stDate | varchar(20) | YES  |      | NULL    |                |
| edDate | varchar(20) | YES  |      | NULL    |                |
| dates  | int(11)   | YES  |      | NULL    |                |
| cost   | int(11)   | YES  |      | NULL    |                |
+-----+-----+-----+-----+-----+
7 rows in set (0.00 sec)

```

六.核心代码

Utils

DateBaseUtil

```

1 package teko7a.Utils;
2
3 import org.junit.Test;
4
5 import java.sql.Connection;
6 import java.sql.DriverManager;
7
8 /**
9  * 数据库工具类
10 *
11 * @author: tekola (teko7a@github.io)
12 * @create 2021/8/24
13 */
14 public class DataBaseUtil {
15
16 /**
17 * 获取数据库连接
18 *
19 * @return {@code Connection} 类
20 * @throws Exception 连接中的异常
21 * @author: tekola (teko7a@github.io)
22 */
23 public Connection getCon() throws Exception {
24     String jdbcName = "com.mysql.cj.jdbc.Driver";
25     Class.forName(jdbcName); // 动态加载 (反射)
26     String dbPassword = "root";
27     String dbUserName = "root";
28     String dbUrl = "jdbc:mysql://localhost:3306/db_RC";
29     return DriverManager.getConnection(dbUrl, dbUserName, dbPassword);
}

```

```

30    }
31
32    /**
33     * 关闭数据库连接
34     *
35     * @param con 要关闭的 {@code Connection}
36     * @throws Exception 关闭时的异常
37     * @author: tekola (teko7a@github.io)
38     */
39    public void closeCon(Connection con) throws Exception {
40        if (con != null) con.close();
41    }
42
43    /**
44     * 这是一个 junit 的测试单元， 可以模拟 Main 的效用
45     * 注意如果 {@code 连接失败}， 需要将 jdbc 中的 jar 包 buildPATH
46     * IDEA 操作步骤： {@code File -> Project Structure -> Libraries -> '+'}
47     *
48     * @author: tekola (teko7a@github.io)
49     */
50    @Test public void Test() {
51        DataBaseUtil DataBaseUtil = new DataBaseUtil();
52        Connection con = null;
53        try{
54            con = DataBaseUtil.getCon();
55            System.out.println("Connect Success");
56        } catch (Exception e) {
57            e.printStackTrace();
58            System.out.println("Connect Failed!");
59        } finally{
60            try{
61                DataBaseUtil.closeCon(con);
62                System.out.println("Close Success");
63            } catch (Exception e) {
64                e.printStackTrace();
65                System.out.println("Close Failed!");
66            }
67        }
68    }
69
70 }

```

StringUtil

```

1 package teko7a.Utils;
2
3 import org.junit.Test;
4
5 import javax.swing.*;

```

```
6 import java.util.regex.Pattern;
7
8 /**
9 * 字符串工具类
10 *
11 * @author: tekola (teko7a@github.io)
12 * @create 2021/8/24
13 */
14 public class StringUtil {
15 /**
16 * 判断字符串是否为空
17 *
18 * @param str 需要判断的字符串
19 * @return 如果本身为空或全为空格返回 true, 否则为 false
20 * @author: tekola (teko7a@github.io)
21 */
22 public static boolean isEmpty(String str) {
23     return str == null || "".equals(str.trim());
24 }
25
26 public static boolean allEmpty(String ... strings) {
27     boolean check = true;
28     for (String string : strings) check &= isEmpty(string);
29     return check;
30 }
31
32 public static boolean existEmpty(String ... strings) {
33     boolean check = false;
34     for (String string : strings) {
35         check |= isEmpty(string);
36     }
37     return check;
38 }
39
40 @Test public void test1() {
41     String[] s = { "", null, "451412", "451515", " " };
42     System.out.println(allEmpty(s));
43     System.out.println(existEmpty(s));
44 }
45
46 /**
47 * 依赖于 StringUtil.isEmpty(String str) , 用于判断 str 非空
48 *
49 * @author: tekola (teko7a@github.io)
50 */
51 public static boolean isNotEmpty(String str) {
52     return !isEmpty(str);
53 }
54
55 /**
56 * 因为这句话太长了, 容易影响单行代码长度。因此将其封装起来, 功能是显示提示
窗口。
```

```

57     *
58     * @param message 要显示的内容
59     * @author: tekola (teko7a@github.io)
60     */
61     public static void log(String message) {
62         JOptionPane.showMessageDialog(null, message);
63     }
64
65     private static final Pattern isPositiveDigitPat = Pattern.compile("[0-9]+(\\.\\.[0-
66     9]+)?");
66     /**
67      * 使用正则表达式 (REGEX) 来判断这是否是一个有效的数字字符串。
68      *
69      * @param str 要判断的字符串
70      * @return 是否是一个正数
71      * @author: tekola (teko7a@github.io)
72      */
73     public static boolean isPositiveDigit(String str) {
74         if (isEmpty(str)) return false;
75         if (str.charAt(0) == '0') return false;
76         str = str.trim();
77         return isPositiveDigitPat.matcher(str).matches();
78     }
79
80     public static boolean isInvalidIdc(String text) {
81         return text.length() != 18 || !isPositiveDigit(text.substring(0, 16));
82     }
83
84     public static boolean isInvalidTel(String tel) {
85         return tel.length() != 11 || !StringUtil.isPositiveDigit(tel);
86     }
87 }
88

```

Model.Person

```

1 package teko7a.Models;
2
3 /**
4  * 用户与管理员的公共父类。不难发现两者的数据库是基本一致的，因此不妨将两者作
5  * 为一个父类下的两个派生类。
6  * @author: tekola (teko7a@github.io)
7  * @create 2021/8/30
8  */
9  public class Person {
10     //==== Data ====
11     private int id;
12     private String name;
13     private String password;
14
15 }

```

```
13  private String realName;
14  private String idc;
15  private String tel;
16
17  public final String className = "person";
18
19  //==== Constructors ====
20
21  public Person() { super(); }
22
23  public Person(int id, String name, String password, String realName, String idc,
24    String tel) {
25    this.id = id;
26    this.name = name;
27    this.password = password;
28    this.realName = realName;
29    this.idc = idc;
30    this.tel = tel;
31  }
32
33  //==== Getter and Setter ====
34
35  public int getId() {
36    return id;
37  }
38
39  public void setId(int id) {
40    this.id = id;
41  }
42
43  public String getName() {
44    return name;
45  }
46
47  public void setName(String name) {
48    this.name = name;
49  }
50
51  public String getPassword() {
52    return password;
53  }
54
55  public void setPassword(String password) {
56    this.password = password;
57  }
58
59  public String getRealName() {
60    return realName;
61  }
62
63  public void setRealName(String realName) {
64    this.realName = realName;
```

```

64    }
65
66    public String getIdc() {
67        return idc;
68    }
69
70    public void setIdc(String idc) {
71        this.idc = idc;
72    }
73
74    public String getTel() {
75        return tel;
76    }
77
78    public void setTel(String tel) {
79        this.tel = tel;
80    }
81
82    //==== Override ====
83
84    @Override public String toString() {
85        return "{" +
86            "id=" + id +
87            ", name='" + name + '\'' +
88            ", password='" + password + '\'' +
89            ", realName='" + realName + '\'' +
90            ", idc='" + idc + '\'' +
91            ", tel='" + tel + '\'' +
92            '}';
93    }
94 }
95

```

Model.Rented

```

1 package teko7a.Models;
2
3 /**
4 * 租车信息条实体类
5 * @author: tekola (teko7a@github.io)
6 * @create 2021/8/30
7 *
8 * +-----+-----+----+----+-----+
9 * | Field | Type   | Null | Key | Default | Extra      |
10 * +-----+-----+----+----+-----+
11 * | id   | int(11) | NO  | PRI | NULL   | auto_increment |
12 * | usrID | int(11) | YES | MUL | NULL   |           |
13 * | carID | int(11) | YES | MUL | NULL   |           |
14 * | stDate| varchar(20)| YES |    | NULL   |           |

```

```
15 * | edDate | varchar(20) | YES |  | NULL  |      |
16 * | dates | int(11)  | YES |  | NULL  |      |
17 * | cost  | int(11)  | YES |  | NULL  |      |
18 * +-----+-----+-----+-----+-----+
19 */
20 public class Rented {
21     // ---- Data ----
22     private int id;
23     private int usrlID;
24     private int carID;
25     private String stDate;
26     private String edDate;
27     private int dates;
28     private int cost;
29
30
31     // ---- Constructors ----
32     public Rented() { super(); }
33
34     public Rented(int id, int usrlID, int carID, String stDate, String edDate, int dates,
35     int cost) {
36         this.id = id;
37         this.usrlID = usrlID;
38         this.carID = carID;
39         this.stDate = stDate;
40         this.edDate = edDate;
41         this.dates = dates;
42         this.cost = cost;
43     }
44
45     // ---- Getter and Setter ----
46
47     public int getId() {
48         return id;
49     }
50
51     public void setId(int id) {
52         this.id = id;
53     }
54
55     public int getUsrlID() {
56         return usrlID;
57     }
58
59     public void setUsrlID(int usrlID) {
60         this.usrlID = usrlID;
61     }
62
63     public int getCarID() {
64         return carID;
65     }
```

```
66     public void setCarID(int carID) {
67         this.carID = carID;
68     }
69
70     public String getStDate() {
71         return stDate;
72     }
73
74     public void setStDate(String stDate) {
75         this.stDate = stDate;
76     }
77
78     public String getEdDate() {
79         return edDate;
80     }
81
82     public void setEdDate(String edDate) {
83         this.edDate = edDate;
84     }
85
86     public int getDates() {
87         return dates;
88     }
89
90     public void setDates(int dates) {
91         this.dates = dates;
92     }
93
94     public int getCost() {
95         return cost;
96     }
97
98     public void setCost(int cost) {
99         this.cost = cost;
100    }
101
102
103 // ===== Override =====
104
105 }
```

Daos.usrDao

```
1 package teko7a.DAOs;
2
3 import teko7a.Models.Rented;
4 import teko7a.Models.Usr;
5 import teko7a.Utils.StringUtil;
```

```
6
7 import java.sql.Connection;
8 import java.sql.PreparedStatement;
9 import java.sql.ResultSet;
10 import java.sql.SQLException;
11
12 /**
13 * @author: tekola (teko7a@github.io)
14 * @create 2021/8/30
15 */
16 public class UsrDAO {
17     RentedDAO rentedDao = new RentedDAO();
18
19 /**
20 * 用户登录验证
21 *
22 * @param con 连接
23 * @param user 登入用户
24 * @return 有效用户， 无效则返回 null
25 * @throws Exception 中途可能出现的异常
26 * @author: tekola (teko7a@github.io)
27 */
28     public Usr login(Connection con, Usr user) throws Exception {
29         Usr resultUser = null;
30         String sql = "SELECT * FROM t_usr WHERE name = ? AND password = ?";
31         // 使用预处理的方法以增强代码可读性。
32         PreparedStatement pstmt = con.prepareStatement(sql);
33         pstmt.setString(1, user.getName());
34         pstmt.setString(2, user.getPassword());
35
36         ResultSet rs = pstmt.executeQuery();
37         if (rs.next()) {
38             resultUser = new Usr();
39             resultUser.setId(rs.getInt("id"));
40             resultUser.setName(rs.getString("name"));
41             resultUser.setPassword(rs.getString("password"));
42         }
43         return resultUser;
44     }
45
46 /**
47 * 判断是否出现重复注册
48 *
49 * @param con 链接
50 * @param usrName 要检查的用户名
51 * @return 没有重复用户名出现
52 * @throws Exception 异常
53 */
54     public boolean hasTwoSameNameUsr(Connection con, String usrName) throws
Exception {
55         PreparedStatement pstmt = con.prepareStatement("SELECT * FROM t_usr
WHERE name = '" + usrName + "'");
```

```
56     return pstmt.executeQuery().next();
57 }
58
59 /**
60 * 用户注册类
61 *
62 * @param con 连接
63 * @param user 用户
64 * @return 0 / 1 --> 0 表示注册失败, 1 表示注册成功
65 * @throws Exception 可能出现的异常
66 */
67 public int register(Connection con, Usr user) throws Exception {
68     String sql = "INSERT INTO t_usr VALUES(NULL,?, ?, ?, ?, ?)";
69     PreparedStatement pstmt = con.prepareStatement(sql);
70     pstmt.setString(1, user.getName());
71     pstmt.setString(2, user.getPassword());
72     pstmt.setString(3, user.getRealName());
73     pstmt.setString(4, user.getIdc());
74     pstmt.setString(5, user.getTel());
75     return pstmt.executeUpdate();
76 }
77
78 /**
79 * 根据用户名查询用户具体信息模块
80 * @param con 链接
81 * @param usrName 用户名, 注册时保证其唯一, 因此查询是安全的。
82 * @return 该用户的信息
83 * @throws Exception 异常
84 */
85 public Usr detail(Connection con, String usrName) throws Exception {
86     String sql = "SELECT * FROM t_usr WHERE name = ?";
87     PreparedStatement pstmt = con.prepareStatement(sql);
88     pstmt.setString(1, usrName);
89     ResultSet rs = pstmt.executeQuery();
90
91     boolean hasNext = rs.next();
92     assert hasNext;
93
94     int id = rs.getInt("id");
95     String userPassword = rs.getString("password");
96     String userRealName = rs.getString("realName");
97     String userIDCN = rs.getString("idc");
98     String userTel = rs.getString("tel");
99
100    return new Usr(id, usrName, userPassword, userRealName, userIDCN,
101                  userTel);
102 }
103
104 /**
105 * 根据用户 id 查询用户具体信息模块
106 * @param con 链接
107 * @param usrid 用户 id 唯一, 查询是安全的。
```

```
107     * @return 该用户的信息
108     * @throws Exception 异常
109     */
110    public Usr detail(Connection con, int usrid) throws Exception {
111        String sql = "SELECT * FROM t_usr WHERE id = ?";
112        PreparedStatement pstmt = con.prepareStatement(sql);
113        pstmt.setInt(1, usrid);
114        ResultSet rs = pstmt.executeQuery();
115
116        boolean hasNext = rs.next();
117        assert hasNext;
118
119        String usrName = rs.getString("name");
120        String userPassword = rs.getString("password");
121        String userRealName = rs.getString("realName");
122        String userIDCN = rs.getString("idc");
123        String userTel = rs.getString("tel");
124
125        return new Usr(usrid, usrName, userPassword, userRealName, userIDCN,
126                      userTel);
127    }
128
129    public ResultSet detail(Connection con, Usr usr) throws Exception {
130        assert usr != null;
131        StringBuilder sb = new StringBuilder("SELECT * FROM t_usr");
132        if (usr.getId() != -1) sb.append(" AND id = ").append(usr.getId()).append("'");
133        if (StringUtil.isNotEmpty(usr.getName())) sb.append(" AND name = ");
134        if (StringUtil.isNotEmpty(usr.getPassword())) sb.append(" AND password = ");
135        if (StringUtil.isNotEmpty(usr.getRealName())) sb.append(" AND realName = ");
136        if (StringUtil.isNotEmpty(usr.getIdc())) sb.append(" AND idc = ");
137        if (StringUtil.isNotEmpty(usr.getTel())) sb.append(" AND tel = ");
138        // System.out.println(sb.toString().replaceFirst("AND", "WHERE"));
139        return con.prepareStatement(sb.toString().replaceFirst("AND",
140                                "WHERE")).executeQuery();
141    }
142
143    /**
144     * 删除用户
145     * @param con 连接
146     * @param usr 要删除的用户
147     * @return 改变记录数
148     * @throws SQLException 数据库异常
149     */
150    public int remove(Connection con, Usr usr) throws SQLException {
151        con.prepareStatement("SET FOREIGN_KEY_CHECKS = 0;").executeQuery();
152        String sql = "DELETE FROM t_usr WHERE id = " + usr.getId() + "'";
153
```

```

152     con.prepareStatement("SET FOREIGN_KEY_CHECKS = 1").executeQuery();
153     return con.prepareStatement(sql).executeUpdate();
154 }
155
156 public ResultSet list(Connection con, Usr usr) throws SQLException {
157     String sql = "SELECT id, name, password, realName, idc, tel " +
158         "FROM t_usr";
159     PreparedStatement pstmt = con.prepareStatement(sql);
160     return pstmt.executeQuery();
161 }
162
163 public int modify(Connection con, Usr usr) throws SQLException {
164     String sql = "UPDATE t_usr SET " +
165         + "name = ?, password = ?, realName = ?, idc = ?, tel = ?" +
166         + "WHERE id = ?";
167     PreparedStatement pstmt = con.prepareStatement(sql);
168     set_5_data(pstmt, usr);
169     pstmt.setInt(6, usr.getId());
170     return pstmt.executeUpdate();
171 }
172
173 private void set_5_data(PreparedStatement pstmt, Usr usr) throws
SQLException {
174     pstmt.setString(1, usr.getName());
175     pstmt.setString(2, usr.getPassword());
176     pstmt.setString(3, usr.getRealName());
177     pstmt.setString(4, usr.getIdc());
178     pstmt.setString(5, usr.getTel());
179 }
180 }
181

```

Frms.Admin.MainFrm 部分代码

```

1 package teko7a.Frames.Admin;
2
3 import teko7a.DAOs.CarDAO;
4 import teko7a.DAOs.MaintainDAO;
5 import teko7a.DAOs.RentedDAO;
6 import teko7a.DAOs.UsrDAO;
7 import teko7a.Frames.Welcome.LoginFrm;
8 import teko7a.Frames.Welcome.RegisterFrm;
9 import teko7a.Models.Admin;
10 import teko7a.Models.Car;
11 import teko7a.Models.Maintain;
12 import teko7a.Models.Usr;
13 import teko7a.Utils.*;
14
15 import javax.swing.*;

```

```
16 import javax.swing.table.DefaultTableModel;
17 import java.awt.*;
18 import java.awt.event.ActionEvent;
19 import java.awt.event.MouseAdapter;
20 import java.awt.event.MouseEvent;
21 import java.sql.Connection;
22 import java.sql.ResultSet;
23 import java.sql.SQLException;
24 import java.util.Objects;
25 import java.util.Vector;
26
27 /**
28 * @author: tekola (teko7a@github.io)
29 * @create 2021/8/30
30 */
31 public class MainFrm extends JFrame {
32     private Admin curAdmin = null;
33
34     DataBaseUtil dbUtil = new DataBaseUtil();
35     CarDAO carDao = new CarDAO();
36     RentedDAO rentedDao = new RentedDAO();
37     UsrDAO usrDao = new UsrDAO();
38     MaintainDAO maintainDao = new MaintainDAO();
39
40     public MainFrm(Admin admin) {
41         this();
42         this.curAdmin = admin;
43         this.name_idJL.setText("usrName(id :" + admin.getId() + ")");
44         this.todayJL.setText("今天是:" + DateUtil.getToday());
45
46         this.fillRentedHisJT();
47         this.fillFixHisJT();
48         this.updateProfitJBActionPerformed();
49
50         this.fillUsrJT(new Usr());
51         this.fillCarJT(new Car());
52         this.fillFixJT();
53     }
54
55     private void fillFixJT() {
56         Connection con = null;
57         try {
58             con = dbUtil.getCon();
59             ResultSet rs = maintainDao.list(con, OrdersEnum.ADD);
60             fillTableHelper4(rs, this.toFixJT);
61         } catch (Exception e) {
62             e.printStackTrace();
63         } finally {
64             try {
65                 dbUtil.closeCon(con);
66             } catch (Exception e) {
67                 e.printStackTrace();
68             }
69         }
70     }
71 }
```

```
68     }
69 }
70 }
71
72 private void fillTableHelper4(ResultSet rs, JTable toFixJT) throws SQLException
{
73     DefaultTableModel dtm = (DefaultTableModel) toFixJT.getModel();
74     dtm.setRowCount(0);
75     while (rs.next()) {
76         Vector<String> v = new Vector<>();
77         v.add(rs.getString("id"));
78         v.add(rs.getString("carID"));
79         v.add(rs.getString("adminID"));
80         v.add(rs.getString("errorDesc"));
81         v.add(rs.getString("stDate"));
82         v.add(rs.getString("edDate"));
83         v.add(rs.getString("cost"));
84         dtm.addRow(v);
85     }
86 }
87
88 private void fillFixHisJT() {
89     Connection con = null;
90     try {
91         con = dbUtil.getCon();
92         ResultSet rs = maintainDao.list(con, OrdersEnum.QUERY);
93         fillTableHelper4(rs, this.hisFixJT);
94     } catch (Exception e) {
95         e.printStackTrace();
96     } finally {
97         try {
98             dbUtil.closeCon(con);
99         } catch (Exception e) {
100            e.printStackTrace();
101        }
102    }
103 }
104 ...
105 private void logoutJBActionPerformed() {
106     int confirm = JOptionPane.showConfirmDialog(null, "确认退出系统吗? \n再检
查下还有没有要做的事哦~");
107     if (confirm == 0) {
108         this.dispose();
109         FrmUtil.openFrm(new LoginFrm());
110     }
111 }
112
113 /**
114 * 表格 {@code usrJT} 行点击事件: 点击某一行, 会在右侧显示出详细数据。
115 *
116 * @author: tekola (teko7a@github.io)
117 */
```

```

118 private void usrJTMousePressed() {
119     int row = this.usrJT.getSelectedRow();
120
121     this.idJTF.setText((String) usrJT.getValueAt(row, 0));
122     this.nameJTF.setText((String) usrJT.getValueAt(row, 1));
123     this.passwordJTF.setText((String) usrJT.getValueAt(row, 2));
124     this.rNameJTF.setText((String) usrJT.getValueAt(row, 3));
125     this.idcJTF.setText((String) usrJT.getValueAt(row, 4));
126     this.telJTF.setText((String) usrJT.getValueAt(row, 5));
127     this.curJTF.setText((String) usrJT.getValueAt(row, 6));
128     this.hisJTF.setText((String) usrJT.getValueAt(row, 7));
129 }
130
131 private void carJTMousePressed() {
132     int row = this.carJT.getSelectedRow();
133
134     this.carIDJTF.setText((String) carJT.getValueAt(row, 0));
135     String carTypeName = (String) carJT.getValueAt(row, 1);
136     int carTypeOrder = carTypeJCB.getItemCount();
137     for (int i = 0; i < carTypeOrder; i++) {
138         if (carTypeJCB.getItemAt(i).equals(carTypeName)) {
139             carTypeJCB.setSelectedIndex(i);
140         }
141     }
142     this.carLicenseJTF.setText((String) carJT.getValueAt(row, 2));
143     this.carBrandJTF.setText((String) carJT.getValueAt(row, 3));
144     this.carSizeJTF.setText((String) carJT.getValueAt(row, 4));
145     this.carSeatNumJTF.setText((String) carJT.getValueAt(row, 5));
146
147     this.carIsAutoJChB.setSelected("1".equals(carJT.getValueAt(row, 6)));
148     this.carTonsJTF.setText((String) carJT.getValueAt(row, 7));
149     int carColorOrder = carColorJCB.getItemCount();
150     String carColor = (String) carJT.getValueAt(row, 8);
151     for (int i = 0; i < carColorOrder; i++) {
152         if (carColorJCB.getItemAt(i).equals(carColor)) {
153             carColorJCB.setSelectedIndex(i);
154         }
155     }
156     this.carFeeJTF.setText((String) carJT.getValueAt(row, 9));
157     String carStatus = (String) carJT.getValueAt(row, 10);
158     switch (Objects.requireNonNull(carStatus)) {
159         case "1" -> carStatusJCB.setSelectedIndex(1);
160         case "0" -> carStatusJCB.setSelectedIndex(2);
161         case "-1" -> carStatusJCB.setSelectedIndex(3);
162     }
163 }
164 ...
165 private JTable usrJT;
166 private JTextField idJTF;
167 private JTextField nameJTF;
168 private JTextField passwordJTF;
169 private JTextField rNameJTF;

```

```

170  private JTextField idcJTF;
171  private JTextField telJTF;
172  private JTextField curJTF;
173  private JTextField hisJTF;
174  private JTable carJT;
175  private JTextField carIDJTF;
176  private JTextField carLicenseJTF;
177  private JTextField carBrandJTF;
178  private JTextField carSizeJTF;
179  private JTextField carSeatNumJTF;
180  private JCheckBox carIsAutoJChB;
181  private JComboBox<String> carStatusJCB;
182  private JComboBox<String> carColorJCB;
183  private JComboBox<String> carTypeJCB;
184  private JTextField carTonsJTF;
185  private JTextField carFeeJTF;
186  private JTable toFixJT;
187  private JTextField maintainMessageJTF;
188  private JTextField maintainCostJTF;
189  private JTextField maintainCntJTF;
190  private JTextField rentedCntJTF;
191  private JTable rentedHisTable;
192  private JTextField totRentedCarsJTF;
193  private JTextField totFixedCarsNumJTF;
194  private JTextField totCostJTF;
195  private JTextField totProfitJTF;
196  private JTable hisFixJT;
197  private JTextField stDateJTF;
198  private JTextField edDateJTF;
199  private JLabel name_idJL;
200  private JLabel todayJL;
201 }

```

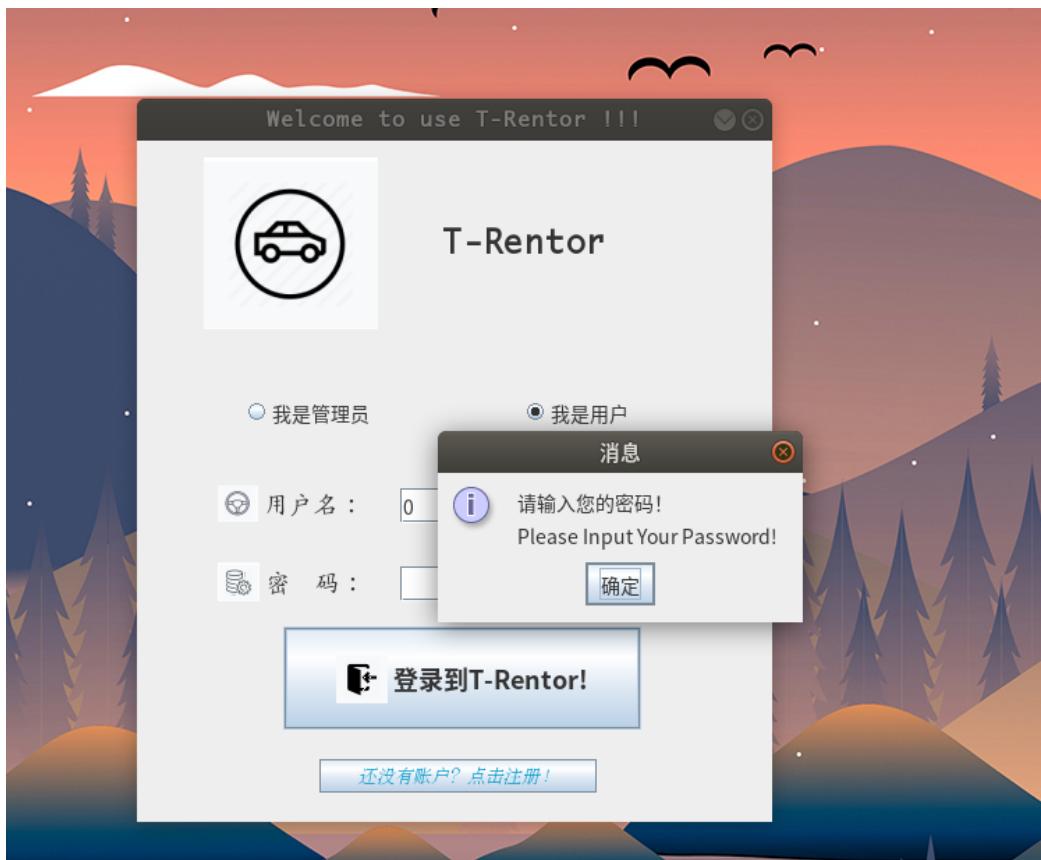
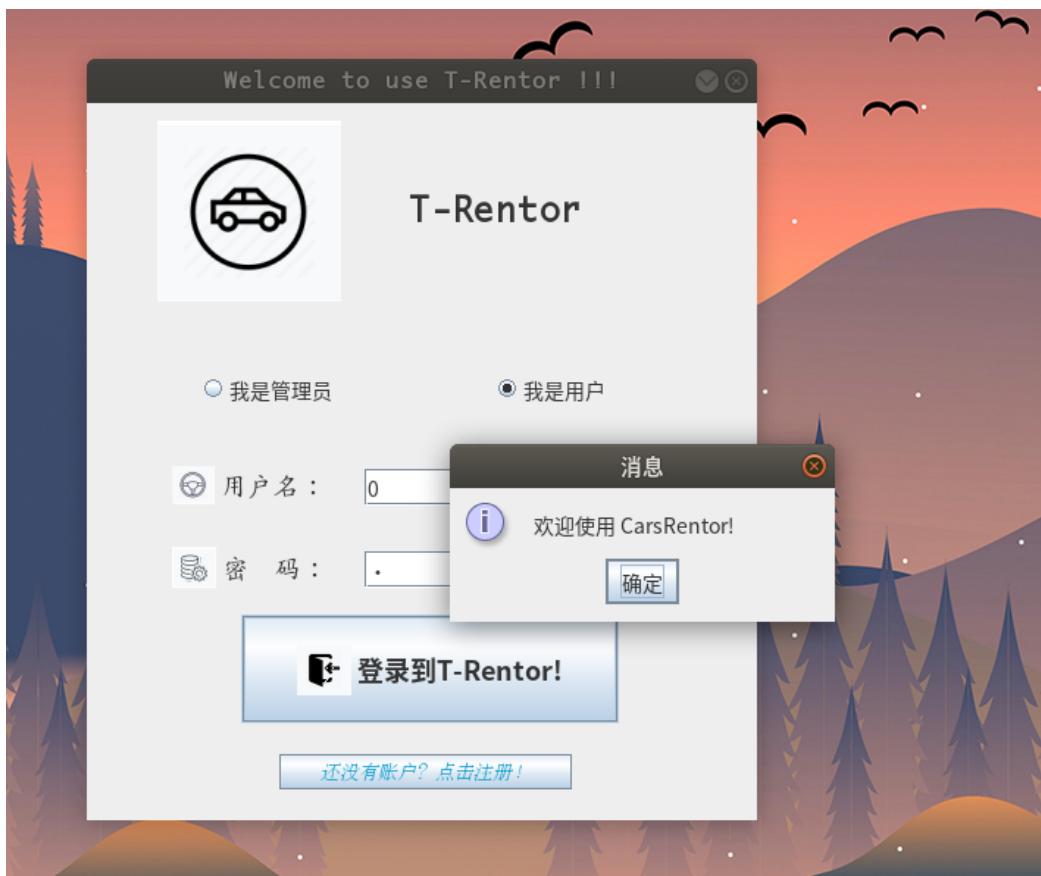
七.系统测试(包含案例以及结果图示)

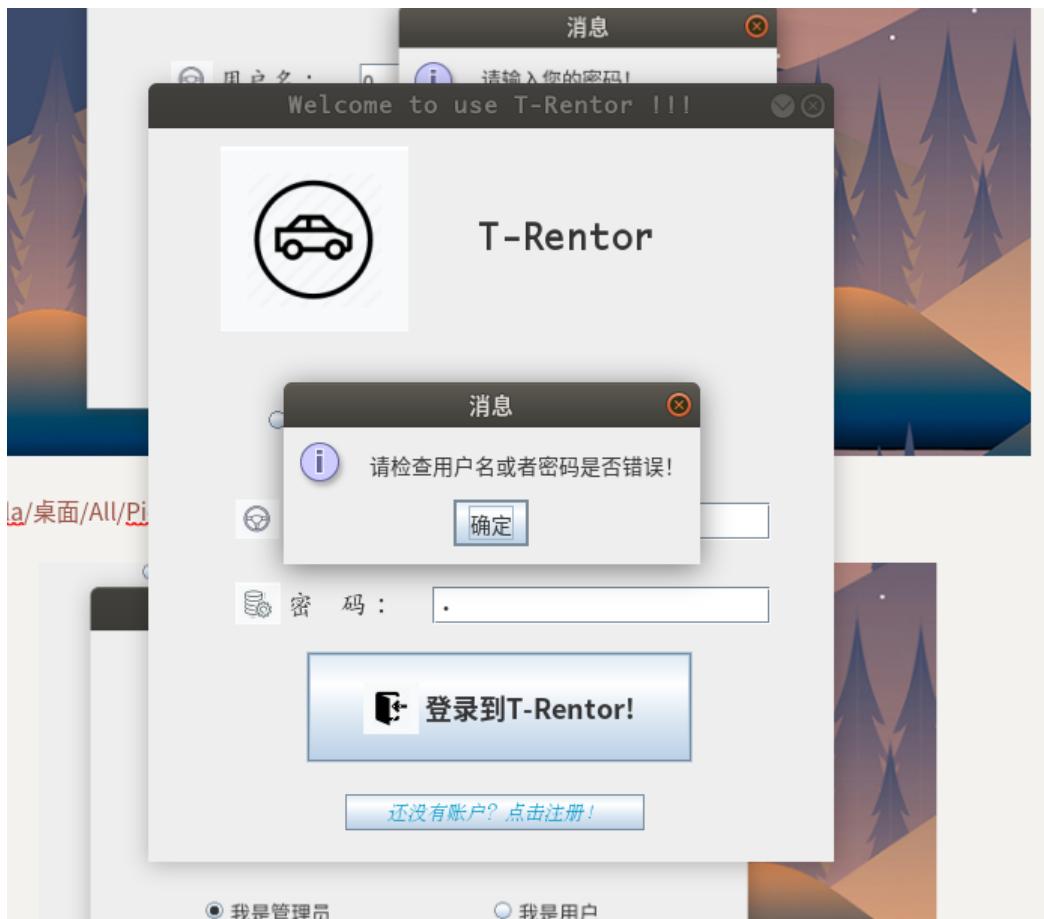
登录

用户登录

所属模块	测试用例编号	用例标题	输入数据	预期结果
用户登录	L01	正确输入所有信息	0, 0	登录成功
	L02	未输入所有信息	0,	登陆失败
	L03	输入所有信息，但是错误	0, 1	登陆失败

所属模块	测试用例编号	用例标题	输入数据	预期结果
	L04	选成了管理员登陆	1, 0	登陆失败

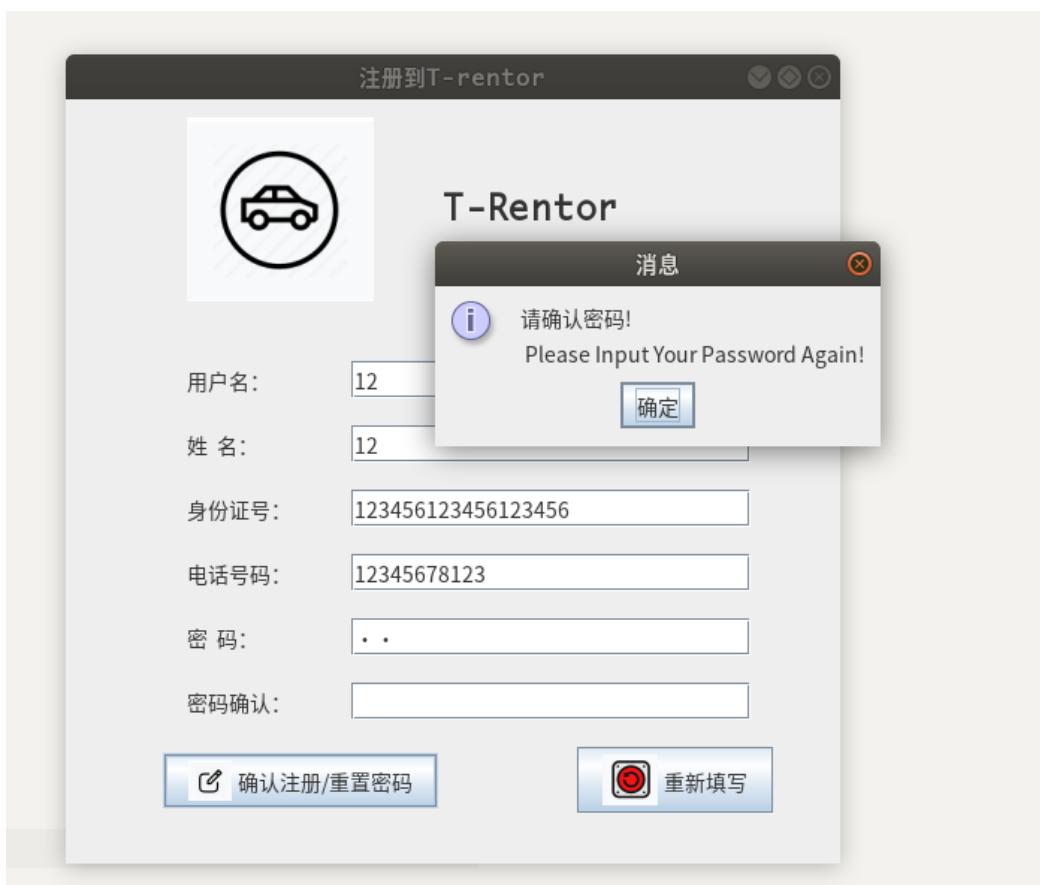




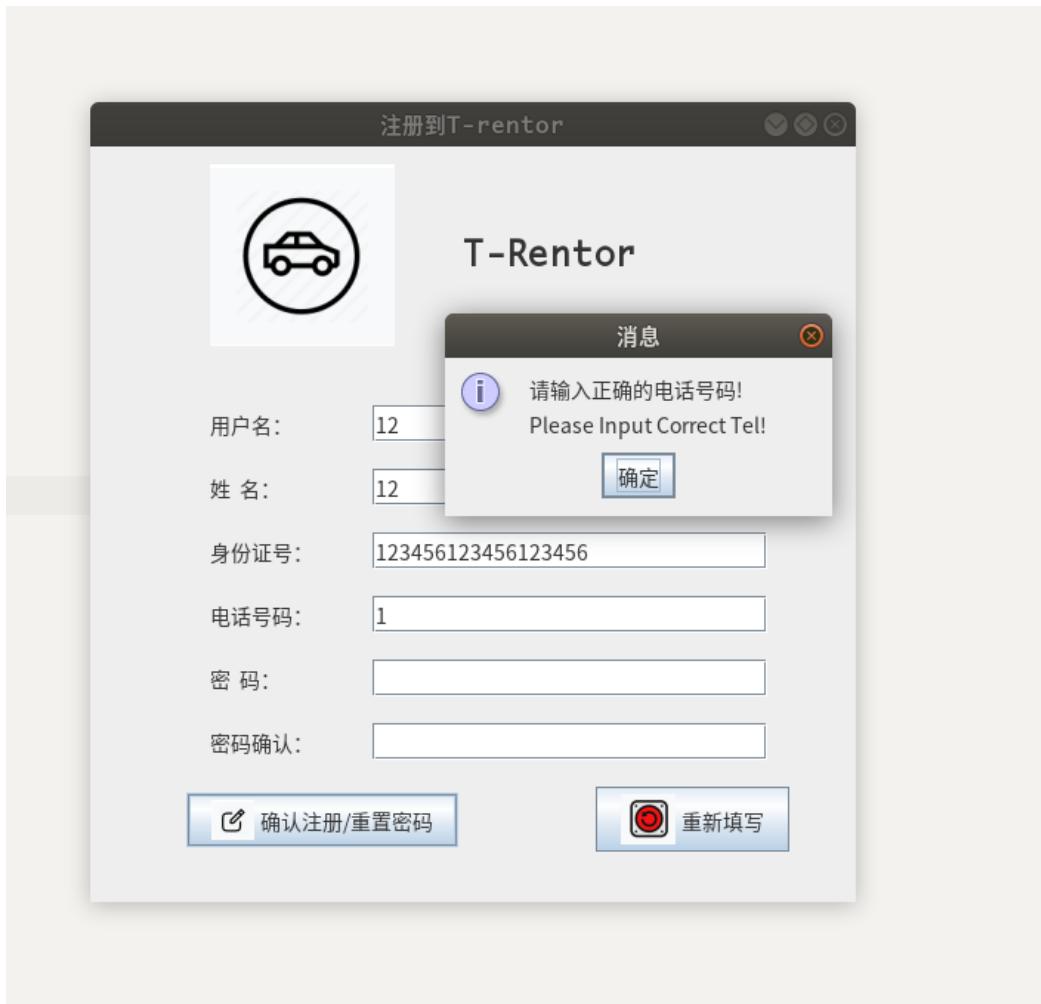
注册

所属模块	测试用例编号	用例标题	输入数据	预期结果
注册测试	L05	正确输入所有信息	如图所示	注册成功
	L06	输入重复信息	如图所示	注册失败
	L07	密码两次不一致	如图所示	注册失败
	L08	没有输入某信息	如图所示	注册失败
	L09	身份证号 / 电话号	如图所示	注册失败

The screenshot shows the T-Rentor registration interface. On the left, there is a sidebar with a '登录到T-Rentor!' button and a '没有账户? 点击注册!' link. The main area has a '注册到T-rentor' title, a car icon, and the 'T-Rentor' logo. A modal dialog box titled '消息' displays the message: '创建新用户 'test_usr' 成功' and '现在就去登录!' with a '确定' (Confirm) button. To the right of the dialog, there are input fields for '用户名' (test_usr), '姓 名' (name), '身份证号' (123456200001011234), '电话号码' (15922222222), '密 码' (empty field), and '密码确认' (empty field). At the bottom are two buttons: '确认注册/重置密码' (Confirm Registration/Reset Password) and '重新填写' (Re-enter).







用户

租车

所属模块	测试用例编号	用例标题	输入数据	预期结果
租车	I101	直接租车	点击	租车成功
	I102	有未还车辆	点击	租车失败
	I103	未点击	null	租车失败

还车

所属模块	测试用例编号	用例标题	输入数据	预期结果
还车	I201	直接还车	点击	还车成功

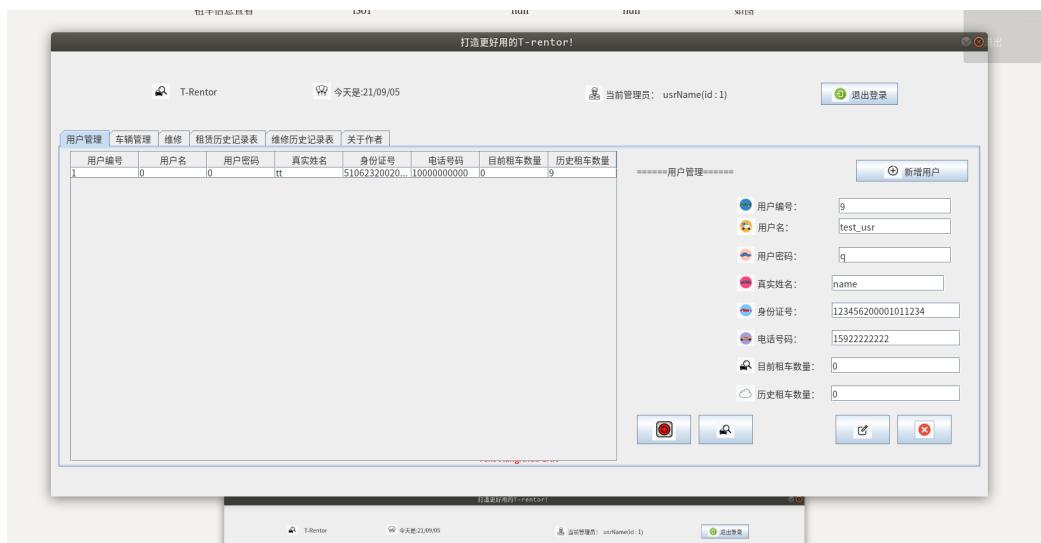
租车信息查看

所属模块	测试用例编号	用例标题	输入数据	预期结果
租车信息查看	I301	null	null	如图

管理员

管理用户

所属模块	测试用例编号	用例标题	输入数据	预期结果
增	I401	增	如图	成功
删	I402	删	如图	成功
查	I403	查	如图	成功
改	I404	改	如图	成功



打造更好用的T-rentor!

T-Rentor 今天是:21/09/05 当前管理员: usrName(id:1) 退出

用户管理							
用户编号	用户名	用户密码	真实姓名	身份证号	电话号码	目前租车数量	历史租车数量
1	0	0	tt	51062320020302631X	1000000000	0	9

=====用户管理=====

用户编号:
 用户名:
 用户密码:
 真实姓名:
 身份证号:
 电话号码:
 目前租车数量:
 历史租车数量:

管理车

所属模块	测试用例编号	用例标题	输入数据	预期结果
增	1501	增	如图	成功
删	1502	删	如图	成功
查	1503	查	如图	成功
改	1504	改	如图	成功

打造更好用的T-rentor!

T-Rentor 今天是:21/09/05 当前管理员: usrName(id:1) 退出

车辆管理							
车辆编号	类型	车牌号	品牌	车型	座数	自动档	排量
3	舒适型	1223	bench	brand new	5	1	2.0T
4	SUV	144	14	1	44	1	44
5	舒适型	313123	0131323	023213	44	1	213123
6	SUV	45654	45654	4664456	46654	0	4664456
7	舒适型	1223	bench	brand new	5	1	2.0T

=====车辆管理=====

新增车辆
 车辆编号: 车辆类型:
 车牌号: 车辆品牌:
 车辆型号:
 座数: 自动档
 排量: 日租金:
 颜色: 状态:

打造更好用的T-rentor!

T-Rentor 今天是:21/09/05 当前管理员: usrName(id:1) 退出

管理用户							
车辆编号	类型	车牌号	品牌	车型	座数	自动档	排量
6	SUV	564456	45654	4664456	46654	1	464

=====车辆管理=====

新增车辆
 车辆编号: 车辆类型:
 车牌号: 车辆品牌:
 车辆型号:
 座数: 自动档
 排量: 日租金:
 颜色: 状态:

送修、修理

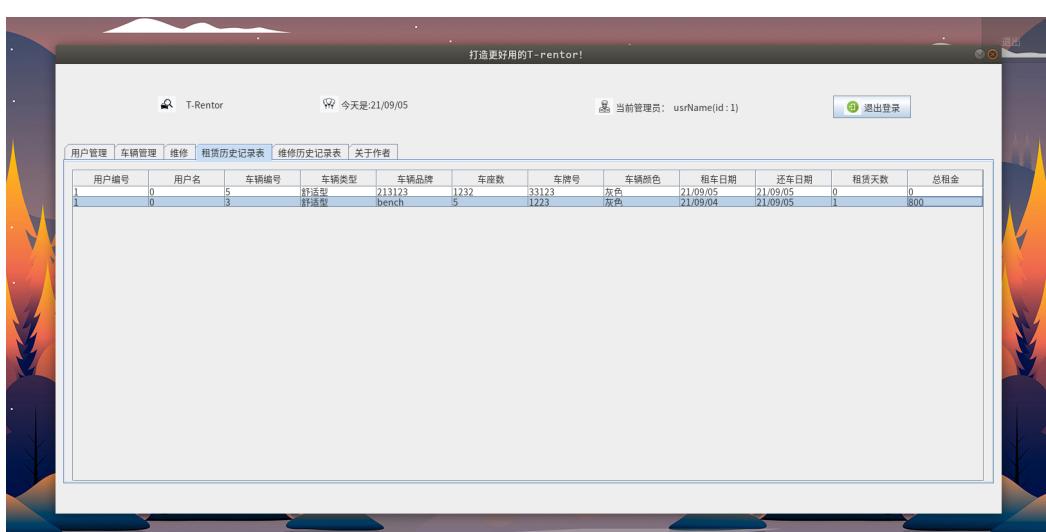
所属模块	测试用例编号	用例标题	输入数据	预期结果
修理	1601	null	如图	成功



The screenshot shows the T-Rentor software interface. At the top, there is a header bar with tabs for '所属模块' (Module), '测试用例编号' (Test Case ID), '用例标题' (Case Title), '输入数据' (Input Data), and '预期结果' (Expected Result). Below the header, a sub-header bar shows '所属模块: 增', '测试用例编号: 1401', '用例标题: 增', '输入数据: 如图', and '预期结果: 成功'. A central message box displays the text '提交表单成功!' (Form submission successful!) with a '确定' (Confirm) button. Below the message box, there is a section titled '维修信息' (Repair Information) containing fields for '维修信息' (Repair Information), '维修花费' (Repair Cost), and '维修次数' (Repair Times). A preview image of a red sports car is also visible.

租车历史查看

所属模块	测试用例编号	用例标题	输入数据	预期结果
租车历史查看	1701	null	null	null



The screenshot shows the T-Rentor software interface. At the top, there is a header bar with tabs for '用户管理' (User Management), '车辆管理' (Vehicle Management), '维修' (Repair), '租赁历史记录表' (Rental History Record Table), '维修历史记录表' (Repair History Record Table), and '关于作者' (About Author). Below the header, a sub-header bar shows '所属模块: 租车历史查看', '测试用例编号: 1701', '用例标题: null', '输入数据: null', and '预期结果: null'. A central table displays a list of rental records with columns: 用户编号 (User ID), 用户名 (User Name), 车辆编号 (Vehicle ID), 车辆类型 (Vehicle Type), 车辆品牌 (Vehicle Brand), 车座数 (Number of Seats), 车牌号 (License Plate Number), 车辆颜色 (Vehicle Color), 租车日期 (Rent Date), 退车日期 (Return Date), 租赁天数 (Number of Days), and 总租金 (Total Rent). One record is shown in the table:

用户编号	用户名	车辆编号	车辆类型	车辆品牌	车座数	车牌号	车辆颜色	租车日期	退车日期	租赁天数	总租金
1	0	5	舒适型	213123	1232	33123	灰色	21/09/05	21/09/05	0	0
1	0	3	舒适型	bench	5	1223	灰色	21/09/04	21/09/05	1	800

维修历史查看、利润分析

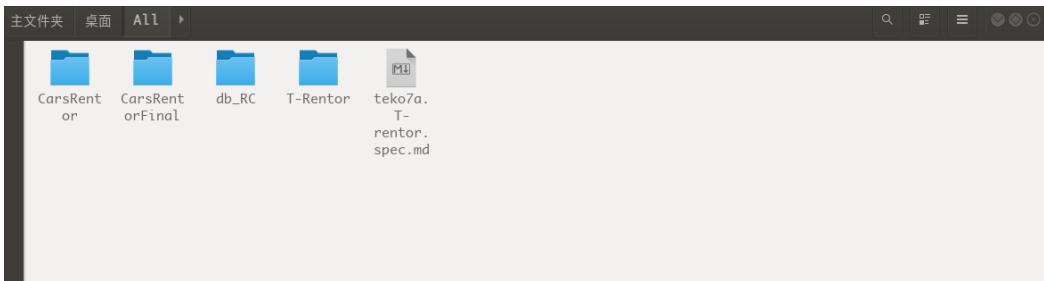
所属模块	测试用例编号	用例标题	输入数据	预期结果
维修历史查看	1801	null	null	null



八.项目总结

在完成整个项目的过程中，我收获良多。在大一刚入学不久后便自学了C语言，加入ACM团队之后，在团队的前辈的带领下又掌握了不少有关C++的知识点。在这个大一后半学年，我自己尝试进行了Java的学习，由于有C++ OOP (Object Oriented Programming) 的基本知识，如鱼得水很快就对其有了一定的了解。暑假开始我便尝试构建一个项目，便恰好有了这样一个机会。

本项目使用了J2SE(也即Java Standard Edition v 16), Swing GUI框架以及 JDBC (Java Database Connectivity) 的知识点。实现了对数据库·表的「增删查改」功能以及对诸多信息的检查功能。涉及到Java中继承、集合、异常、范型、反射、注解以及多态等知识，同时，数据库采用MySQL数据库。除终端开发外，我还使用了IntelliJ Idea与Data Grip软件。



收获与感受：这次项目实践经历让我加深了对Java的理解，在写下第一个、第二个版本之后，第三个版本中我更能体会到「代码结构」的重要性，不断构思不断增强了我的思考能力以及代码能力。同时，在开发中，我还遇到了一个巨大的问题：我常常写出「Null Pointer Exception」的代码，但我总能通过调试将其解决。不仅如此，我同时还学习到了「MySQL」与「Swing GUI」的相关知识，这些虽然已经过时，但对于我之后学习Qt与C3P0有了一定的基础。相信自己一定能在之后的大学生活中掌握更多实用的知识！

```
mysql -u root -p
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H) 退出

mysql> select * from t_maintain WHERE stDate >= '21/01/01' AND edDate IS not null
   AND edDate <= '21/09/05';
+----+-----+-----+-----+-----+-----+
| id | carID | adminID | errorDesc | stDate | edDate | cost |
+----+-----+-----+-----+-----+-----+
| 6 | 5 | 1 | 5345654 | 21/09/04 | 21/09/05 | 998244353 |
| 15 | -1 | 1 | NULL | 21/09/04 | 21/09/04 | 0 |
| 16 | -1 | 1 | NULL | 21/09/04 | 21/09/04 | 0 |
| 17 | -1 | 1 | NULL | 21/09/04 | 21/09/04 | 0 |
| 18 | -1 | 1 | NULL | 21/09/04 | 21/09/04 | 0 |
| 19 | -1 | 1 | NULL | 21/09/04 | 21/09/04 | 0 |
| 20 | -1 | 1 | NULL | 21/09/04 | 21/09/04 | 0 |
| 51 | 4 | 1 | NULL | 21/09/05 | 21/09/05 | 0 |
| 52 | 4 | 1 | NULL | 21/09/05 | 21/09/05 | 0 |
| 53 | 6 | 1 | NULL | 21/09/05 | 21/09/05 | 0 |
| 54 | 5 | 1 | NULL | 21/09/05 | 21/09/05 | 0 |
| 55 | 6 | 1 | NULL | 21/09/05 | 21/09/05 | 0 |
| 56 | 6 | 1 | NULL | 21/09/05 | 21/09/05 | 0 |
| 57 | 5 | 1 | NULL | 21/09/05 | 21/09/05 | -46546545 |
+----+-----+-----+-----+-----+-----+
14 rows in set (0.00 sec)

mysql> select * from t_maintain WHERE stDate >= '21/01/01' AND edDate IS not null
   AND cost != 0 AND edDate <= '21/09/05';
+----+-----+-----+-----+-----+-----+
| id | carID | adminID | errorDesc | stDate | edDate | cost |
+----+-----+-----+-----+-----+-----+
| 6 | 5 | 1 | 5345654 | 21/09/04 | 21/09/05 | 998244353 |
| 57 | 5 | 1 | NULL | 21/09/05 | 21/09/05 | -46546545 |
+----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

* 作者 :Teko7a, 邮箱 :tecola@qq.com, 个人网站 :<https://teko7a.github.io>

*2021/09/05