重 庆 大 学 软 件 学 院

实 验 报 告

|  |  |
| --- | --- |
| 实验名称 | ATM |
| 课程名称 | 计算机网络与通信 |

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名 | 宋长峻  王镇东 | 成绩 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 学号 | 20221199  20221429 | 教师 | **胡海波** |

|  |  |  |  |
| --- | --- | --- | --- |
| 班级 | 软工04 | 日期 | 2023/4 |

**《计算机网络与通信》实验报告**

**开课实验室：DS1502 2023年 4月 1日**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 姓 名 | 宋长峻  王镇东 | | 年级、班级 | 2022软工04 | 成 绩 |  |
| 实验（项目）名称 | | | ATM | | 指导教师 | 胡海波 |
| 教师评语 | | 教师签名：  **年 月 日** | | | | |

**系统详细界面**

1.欢迎界面：

在用户启动ATM系统时，显示欢迎信息，包括系统名称和简短介绍。

2.主菜单界面：

提供用户各种操作选项的主菜单界面。

显示可用选项列表，如“查看余额”、“存款”、“取款”和“退出”。

用户通过键盘输入数字来选择相应的操作。

3.查看余额界面：

当用户选择“查看余额”时，显示用户当前账户余额。

提供返回主菜单选项。

4.存款界面：

当用户选择“存款”时，提示用户输入存款金额。

提示用户确认存款金额，并显示新的账户余额。

提供返回主菜单选项。

5.取款界面：

当用户选择“取款”时，提示用户输入取款金额。

检查余额是否充足，若充足则进行取款，否则显示余额不足信息。

显示取款成功或失败的消息，并显示更新后的余额。

提供返回主菜单选项。

6.退出界面：

当用户选择“退出”时，显示感谢信息并关闭系统。

7.错误处理界面：

处理用户输入错误的情况，如非法字符、超出范围等。

提示用户重新输入或返回主菜单。

**系统设计**

**SERVER程序设计思路（含源代码）**

数据库模块：处理客户端的余额查询和取款操作请求，更新数据库中的账户信息。

1.AccountDao.java：数据访问对象，负责与数据库进行交互，执行具体的数据操作，例如用户认证、账户查询等。

class ATM\_server {  
 private static final String *LOG\_FILE\_PATH* = "ATM\_record.txt";  
 public static void main(String[] args) {  
 try {  
 // 连接到数据库  
 String url = "jdbc:mysql://localhost:3306/atm? useSSL=true";  
 String user = "root";  
 String pass = "scj18360290868";  
  
 try (Connection conn = DriverManager.*getConnection*(url, user, pass)) {  
 // 创建表格  
 String createTableSQL = "CREATE TABLE IF NOT EXISTS accounts ("  
 + "id INT AUTO\_INCREMENT PRIMARY KEY,"  
 + "username VARCHAR(50),"  
 + "password VARCHAR(50),"  
 + "balance INT"  
 + ")";  
 try (Statement statement = conn.createStatement()) {  
 statement.execute(createTableSQL);  
 }  
 }  
  
 // 启动服务器  
 ServerSocket serverSocket = new ServerSocket(2525);  
 System.*out*.println("ATM Server started. Listening on port 2525...");  
  
 while (true) {  
 Socket clientSocket = serverSocket.accept();  
 System.*out*.println("Client connected: " + clientSocket.getInetAddress().getHostAddress());  
  
 new Thread(new ClientHandler(clientSocket)).start();  
 }  
 } catch (IOException | SQLException e) {  
 System.*err*.println("Error: " + e.getMessage());  
 }  
 }

2.OperLogDao.java：记录操作日志到数据库，例如账户的各种操作记录。

private void writeToLog(String logMessage) {  
 try (FileWriter writer = new FileWriter(*LOG\_FILE\_PATH*, true)) {  
 writer.write(logMessage + "\n");  
 } catch (IOException e) {  
 System.*err*.println("Error writing to log file: " + e.getMessage());  
 }  
 }  
  
 // 获取当前时间的字符串表示  
 private String getCurrentTime() {  
 LocalDateTime currentTime = LocalDateTime.*now*();  
 DateTimeFormatter formatter = DateTimeFormatter.*ofPattern*("yyyy-MM-dd HH:mm:ss");  
 return currentTime.format(formatter);  
 }  
}

1. DBConn.java:数据库连接管理。

try {  
 // 连接到数据库  
 String url = "jdbc:mysql://localhost:3306/atm? useSSL=true";  
 String user = "root";  
 String pass = "scj18360290868";  
  
 try (Connection conn = DriverManager.*getConnection*(url, user, pass)) {  
 // 创建表格  
 String createTableSQL = "CREATE TABLE IF NOT EXISTS accounts ("  
 + "id INT AUTO\_INCREMENT PRIMARY KEY,"  
 + "username VARCHAR(50),"  
 + "password VARCHAR(50),"  
 + "balance INT"  
 + ")";  
 try (Statement statement = conn.createStatement()) {  
 statement.execute(createTableSQL);  
 }  
 }

1. 查询取款等的相关代码

private int getBalance(String username) throws SQLException {  
 String url = "jdbc:mysql://localhost:3306/atm? useSSL=true";  
 String user = "root";  
 String pass = "scj18360290868";  
  
  
 try (Connection conn = DriverManager.*getConnection*(url, user, pass);  
 PreparedStatement statement = conn.prepareStatement("SELECT balance FROM accounts WHERE username=?")) {  
 statement.setString(1, username);  
 try (ResultSet resultSet = statement.executeQuery()) {  
 if (resultSet.next()) {  
 return resultSet.getInt("balance");  
 }  
 }  
 }  
 return 0; // 如果找不到账户，则返回0余额  
}  
  
// 取款操作  
private boolean withdraw(String username, int amount) throws SQLException {  
 String url = "jdbc:mysql://localhost:3306/atm? useSSL=true";  
 String user = "root";  
 String pass = "scj18360290868";  
  
  
 try (Connection conn = DriverManager.*getConnection*(url, user, pass)) {  
 // 查询当前余额  
 int balance = getBalance(username);  
 if (balance >= amount) {  
 // 如果余额充足，执行取款操作  
 balance -= amount;  
 // 更新数据库中的余额信息  
 try (PreparedStatement statement = conn.prepareStatement("UPDATE accounts SET balance=? WHERE username=?")) {  
 statement.setInt(1, balance);  
 statement.setString(2, username);  
 statement.executeUpdate();  
 }  
 return true; // 取款成功  
 } else {  
 return false; // 余额不足，取款失败  
 }  
 }  
}

1. 与客户端连接代码

try (  
 BufferedReader inFromClient = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));  
 DataOutputStream outToClient = new DataOutputStream(clientSocket.getOutputStream())  
) {  
 String response;  
 String username = null;  
 String password = null;  
  
 while (true) {  
 String request = inFromClient.readLine();  
 System.*out*.println(request);  
  
 if (request.equals("BYE")) {  
 outToClient.writeBytes("BYE" + '\n');  
 writeToLog("[" + username + "] BYE command received at " + getCurrentTime());  
 }  
  
 if (request.startsWith("HELO")) {  
 username = request.substring(5);  
 System.*out*.println(username);  
 // 检查用户ID是否存在  
 if (checkAccountExists(username)) {  
 response = "500 AUTH REQUIRE";  
 } else {  
 response = "401 ERROR!";  
 }  
 outToClient.writeBytes(response + '\n');  
 writeToLog("[" + username + "] HELO command received at " + getCurrentTime());  
 } else if (request.startsWith("PASS")) {  
 password = request.substring(5);  
 System.*out*.println(password);  
 // 检查密码是否正确  
 if (authenticate(username, password)) {  
 response = "525 OK!";  
 } else {  
 response = "401 ERROR!";  
 }  
 outToClient.writeBytes(response + '\n');  
 writeToLog("[" + username + "] PASS command received at " + getCurrentTime() + ". Authentication status: " + (response.equals("525") ? "Successful" : "Failed"));  
 } else if (request.startsWith("WDRA")) {  
 int amount = Integer.*parseInt*(request.substring(5));  
 boolean success = withdraw(username, amount);  
 if (success) {  
 response = "525 OK!"; // 取款成功  
 } else {  
 response = "401 ERROR!"; // 取款失败  
 }  
 outToClient.writeBytes(response + '\n');  
 writeToLog("[" + username + "] WDRA command received at " + getCurrentTime() + ". Withdrawal amount: " + amount + ". Status: " + (success ? "Successful" : "Failed"));  
 } else if (request.equals("BALA")) {  
 int balance = getBalance(username); // 传递正确的用户名参数  
 outToClient.writeBytes("AMNT:" + balance +'\n');  
 writeToLog("[" + username + "] BALA command received at " + getCurrentTime() + ". Balance inquiry result: " + balance);  
 }  
 }  
} catch (IOException | SQLException e) {  
 System.*err*.println("Error handling client request: " + e.getMessage());  
} finally {  
 try {  
 clientSocket.close();  
 } catch (IOException e) {  
 System.*err*.println("Error closing client socket: " + e.getMessage());  
 }  
}

**CONSTOMER程序（含源代码）**

主界面实现代码（.java）：

功能实现代码（其他功能代码逻辑类似，此处只以取款为例）：

**数据库设计**

**数据库设计说明**

**本数据库设计涉及以下实体：**

账户（account）

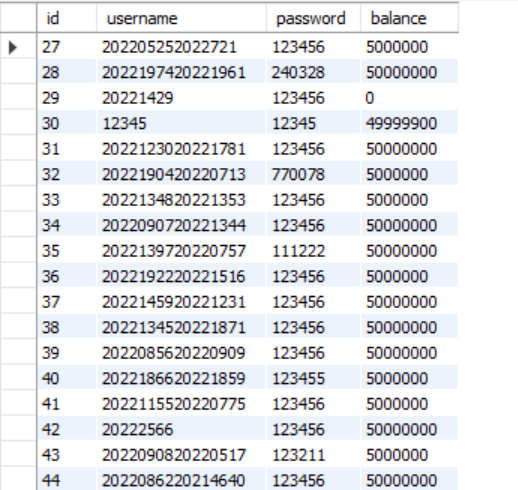
**实体属性和关系：**

账户（account）：

| username | INT | 主键，唯一标识用户的账户，自增长整数类型。

| amount | FLOAT | 存款数额，表示用户存款的初始金额。 |

| password | VARCHAR(255) | 账户密码，表示用户的登录密码。 |



**测试计划**

**1.启动和欢迎界面测试：**

**验证ATM系统启动后是否显示正确的欢迎信息以及主菜单是否显示正确的选项列表。**

**2.密码功能测试：**

**确保用户能够成功验证密码。**

**若密码输入错误，系统会提示“密码输入错误”，如图所示（后同）：**

**3.查看账号查询功能测试：**

**确保用户能够成功查看账户余额和操作记录。**

1. **取款功能测试：**

**测试用户能否成功取款，验证取款成功后余额是否正确更新。**

**如果输入的金额超过余额，系统会提示“余额不足”。如下图所示：**

1. **退出功能测试：**

**确保用户选择退出后系统能够正确关闭**。

1. **日志功能，记录客户端每一次操作情况，其中包括操作名，操作时间，与相应的操作情况等**

（下述图片为一部分示例）

