《生活消费管理系统》

系统设计与开发说明

课程名称 Java 语言与 WEB 应用程序开发

开课学期 2022 至 2023 学年 第 一 学期

年 级 2021 级 专业班级 网络工程

学 生 崔一含

学 号 222021321072052

第一章 绪论

1.1 系统的目的和意义

随着社会的发展，物质生活越来越充裕，大部分家庭当中均以独生子女居多，家长往往倾其所有来尽量满足小孩的各种需求，力争使他们得到最好的教育。基于此，节约意识和理财意识逐渐淡薄。为了让当代大学生对自己的每笔消费都能做到心中有数，从而培养自己的理财意识和自我管理能力，从而开发了此消费管理系统。

1. 使用该消费管理系统能够保持理智消费，规避消费风险。
2. 可以养成良好的生活习惯，可以促进消费安全，提高生活质量，避免浪费。
3. 可以帮助大学生合理规划消费，更好的实现自己的每一个消费目标，而不损害自己的生活消费品质。
4. 培养良好的消费习惯，搞清楚钱是怎样花出去的，才会避免大手大脚乱花钱。
5. 更直观地了解自己的消费与支出，对自己的日常经济生活进行合理的调整，从而能够成功在每个月存下一笔钱，投资自己，打造资产基础。

第二章 需求分析

2.1 用户需求分析

开发生活消费管理系统的第一步是进行需求分析。需求分析的好坏直接决定着系统能否真正满足用户的需要。分析需求是生活消费管理系统开发的第一步，也是最重要的一步。所以通过多方面的调查，搜集校内学生和老师的意见，通过分析比较和资料查询，最后所确定主要的用户需求包括：

（1）消费的管理

记录大学期间发生的各种消费事情，注册登录系统后可以进行日常消费事件的增删改查，可以进行搜索：可以按日期查询、也可以按某个关键词查询。

（2）消费情况的统计

可以对每个消费类型进行统计，统计每个消费类型花费的金额。

2.2 功能需求分析

理解需求是在问题及其最终解决方案之间架设桥梁的第一步。开发系统必须充分了解系统的面向对象和用户需求，否则，对需求定义的任何改进，设计上都必须大量的返工，而且系统的实用性会偏差。根据前面的用户需求分析，整理出了生活消费管理系统的功能需求。

（1）记录大学期间发生的各种消费事情，登录系统后可以进行日常消费事件的增删改查。

（2）可以按日期查询、也可以按某个关键词查询，还可以统计各种关键词相关的消费情况。

（3）添加各种消费时先选择消费类型，方便后期的数据统计。

第三章 系统设计与技术

3.1 系统模块结构设计

【绘制系统结构模块图，展示功能组成部分，以及各个部分之间的关系】

3.1.1 功能模块设计

通过对系统的需求分析和可行性分析，本系统设计了以下几个模块，如图 3-1 所示

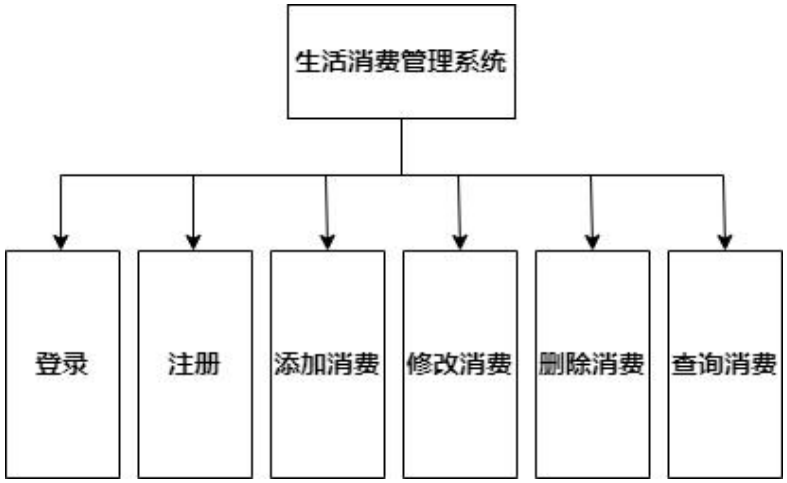

图 3-1 系统功能模块图

3.1.2 系统流程设计

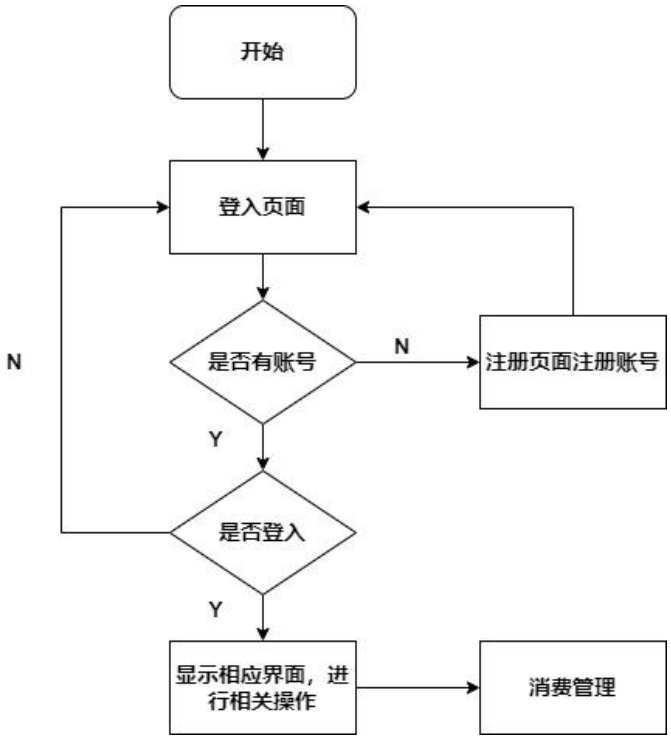

图 3-2 系统流程图

本系统的主要流程：用户打开系统进入登录页面，如果有账户可直接登录，如果没有先进行账号注册再登录系统，成功登录系统后就可以对消费进行管理。如图 3-2 所示。

3.1.3 业务逻辑层设计

“生活消费管理系统”主要是包括登录注册、添加消费、修改消费、删除消费、查询消费等。因此，可以将系统主要划分为登入注册和消费管理两个模块。

（1）登录注册模块设计

登陆注册账号，用户登录系统需要正确输入账号信息，没有账号需要先注册。

(2)消费管理模块设计

包括添加消费，修改消费，删除消费，查询消费，数据统计。

3.2 系统数据库结构设计

【系统实现所创建的数据库表结构，以及表与表之间的关系，最好用 ER 图说明】

数据库存储着所有的信息，数据库在一个管理系统中占有非常重要的地位，数据库结构设计的好坏将直接对应用系统的效率以及实现的效率产生影响。合理的数据库结构可以提高存储的效率，保证数据的完整性和一致性。

3.2.1 概念结构设计—E-R 图

本系统采用 E-R 图的方法进行数据库概念结构设计，系统部分 E-R 图，如图 3-3 所示。

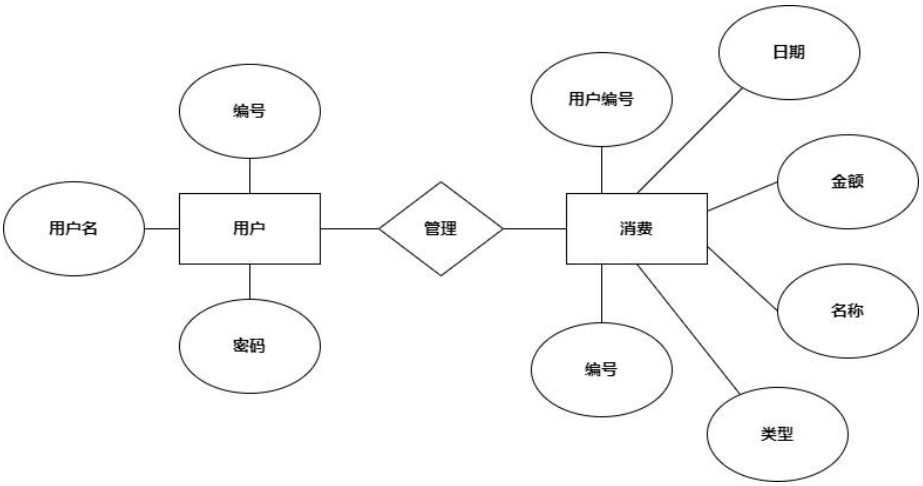

图 3-3 系统部分 E-R 图

2.3.2 逻辑结构设计—关系表

在系统的数据库设计中，首先要创建系统数据库，然后在数据库中创建需要的表和字段。在这个数据库管理系统中要建立 2 张数据表。这 2 张数据表的字段说明。

用户表，如表 3-1 所示。

表 3-1 用户表(t_user)

名称	类型	可否为空	说明	备注
id	int(11)	否	编号	主键
username	varchar(255)	否	用户名	
password	varchar(255)	否	密码	

表 3-2 消费表(t_consumption)

名称	类型	可否为空	说明	备注
id	int(11)	否	编号	主键
type	varchar(255)	否	类型	
name	varchar(255)	否	名称	
money	double(255)	否	金额	
date	datetime	否	日期	
user_id	int(11)	否	用户编号	

3.3 关键技术

本系统采用 MVC 三层架构模式 系统开发中使用了 Servlet、Jsp、JSEL、EL、Jdbc 等技术，其中用 Servlet 控制系统流程，Jsp、JSTL、EL 作为数据显示层，Jdbc 主要链接数据库，对数据进行增删改查。

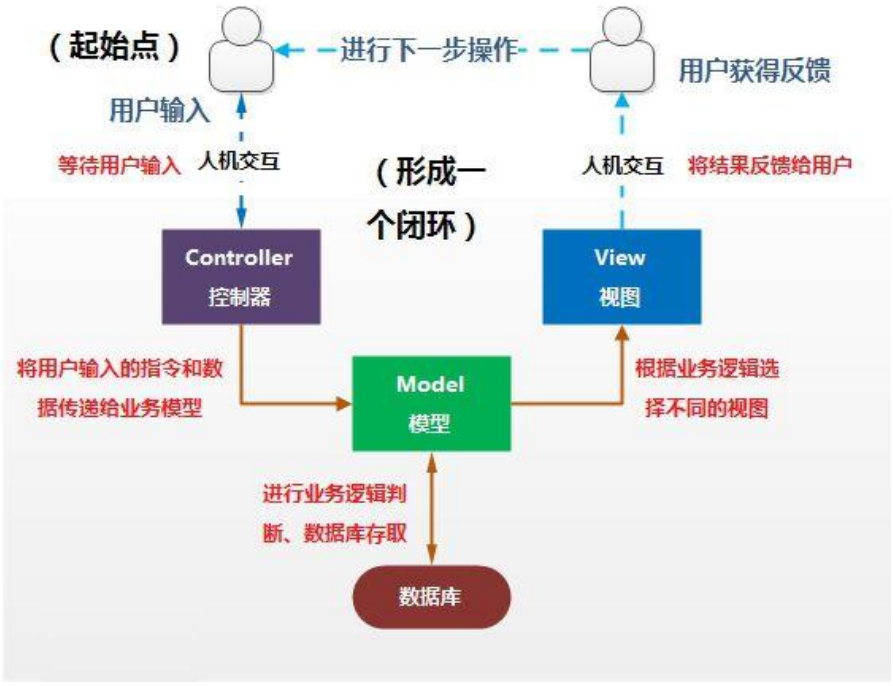

图 3-4 工作原理图

第四章 系统实现

4.1 数据库连接

该系统使用的数据库是MySQL，Web容器是Tomcat，使用的MySQL版本是MySQL5.6，Tomcat的版本是Tomcat 8.0。将两个软件分别进行安装、配置及测试。

数据采用Jdbc进行连接：

1. 加载驱动器

2. 创建connection对象

3. 创建Statement对象

4. Statement(executeQuery方法)执行sql语句

5. 创建处理结果集对象ResultSet

6. 处理异常，关闭所有JDBC对象资源(注意关闭顺序与声明顺序相反，先关结果集对象、后关statement对象、最后关connection对象)

表 4-1 JDBCUtil 相关代码

```
1.public class JDBCUtil {
2.    private Connection conn;
3.    private Statement stmt;
4.    private PreparedStatement pstmt;
5.    private ResultSet rs;
6.
7.    //数据库连接操作
```

```
8.     public void init() throws Exception {
9.         Class.forName("com.mysql.jdbc.Driver");
10.          String url = "jdbc:mysql://localhost:3306/consumption";
11.          String user = "root";
12.          String pwd = "111111";
13.          conn = DriverManager.getConnection(url, user, pwd);
14.      }
15.
16.      //数据库关闭操作
17.      public void closeAll() throws SQLException {
18.          if (rs != null) {
19.              rs.close();
20.          }
21.          if (pstmt != null) {
22.              pstmt.close();
23.          }
24.          if (stmt != null) {
25.              stmt.close();
26.          }
27.          if (conn != null) {
28.              conn.close();
29.          }
30.      }
31.
32.
33.
34.      /*通用数据库所有 增删改 操作
35.       * sql 带? 号的 sql 语句
36.       * params ? 中具体的参数
37.       * Object... params  动态可变参数
38.       * @return 行数
39.       */
40.      public int executeNonQuery(String sql, Object... params) throws Exception
     {
41.          //获取一个连接
42.          init();
43.          //根据 sql 和连接，获取一个声明
44.          try {
45.              pstmt = conn.prepareStatement(sql);
46.              //加入所有参数
47.              for(int i =0;i<params.length;i++){
48.                  pstmt.setObject(i+1,params[i]);
49.              }
50.              int hang = pstmt.executeUpdate(); //返回影响的行数
51.              this.closeAll();
52.              return hang;
53.          } catch (SQLException e){
54.              System.out.println("执行通用 Sql 出错");
55.              e.printStackTrace();
56.          }
57.          return 0;
```

```
58.     }
59.
60.     /*自定义查询用户名是否存在
61.      * sql 带? 号的 sql 语句
62.      * params ? 中具体的参数
63.      * @return
64.      */
65.     public boolean getUsername(String sql,String username) throws Exception {

66.         //获取一个连接
67.         init();
68.         //根据 sql 和连接，获取一个声明
69.         try {
70.             pstmt = conn.prepareStatement(sql);
71.             pstmt.setObject(1,username);
72.             rs = pstmt.executeQuery();
73.             if(rs.next()){
74.                 this.closeAll();
75.                 return true;
76.             }
77.         }catch (SQLException e){
78.             System.out.println("执行查询用户名是否存在 Sql 出错");
79.             e.printStackTrace();
80.         }
81.         return false;
82.     }
83.
84.     /*自定义查询用户登入，根据用户名和密码
85.      * sql 带? 号的 sql 语句
86.      * params ? 中具体的参数
87.      * @return
88.      */
89.     public User getUser(String sql,String username,String password) throws Ex
    ception {
90.         //获取一个连接
91.         init();
92.         //根据 sql 和连接，获取一个声明
93.         try {
94.             pstmt = conn.prepareStatement(sql);
95.             pstmt.setObject(1,username);
96.             pstmt.setObject(2,password);
97.             rs = pstmt.executeQuery();
98.             if(rs.next()){
99.                 User user = new User();
100.                    user.setId(rs.getInt("id"));
101.                    user.setUsername(rs.getString("username"));
102.                    user.setPassword(rs.getString("password"));
103.                    this.closeAll();
104.                    return user;
105.                }
106.            }catch (SQLException e){
```

```
107.                 System.out.println("执行查询用户登录是否存在 Sql 出错");
108.                 e.printStackTrace();
109.             }
110.             return null;
111.         }
112.
113.         /*自定义查询 根据用户 id 查询消费
114.          * sql 带? 号的 sql 语句
115.          * params ? 中具体的参数
116.          * @return
117.          */
118.         public List<Consumption> getByUserIdConsumption(String sql,Integer id)
     throws Exception {
119.             //获取一个连接
120.             init();
121.             List<Consumption> list = new ArrayList<Consumption>();
122.             //根据 sql 和连接，获取一个声明
123.             try {
124.                 pstmt = conn.prepareStatement(sql);
125.                 pstmt.setObject(1,id);
126.                 rs = pstmt.executeQuery();
127.                 while (rs.next()){
128.                     Consumption consumption = new Consumption();
129.                     consumption.setId(rs.getInt("id"));
130.                     consumption.setType(rs.getString("type"));
131.                     consumption.setName(rs.getString("name"));
132.                     consumption.setMoney(rs.getDouble("money"));
133.                     consumption.setDate(rs.getDate("date"));
134.                     list.add(consumption);
135.                 }
136.                 this.closeAll();
137.                 return list;
138.             }catch (SQLException e){
139.                 System.out.println("执行查询消费(id)是否存在 Sql 出错");
140.                 e.printStackTrace();
141.             }
142.             return null;
143.         }
144.
145.         /*自定义查询 根据名称 查询消费
146.          * sql 带? 号的 sql 语句
147.          * params ? 中具体的参数
148.          * @return
149.          */
150.         public List<Consumption> getByNameConsumption(String sql,String name,I
    nteger userId) throws Exception {
151.             //获取一个连接
152.             init();
153.             List<Consumption> list = new ArrayList<Consumption>();
154.             //根据 sql 和连接，获取一个声明
155.             try {
```

```
156.                pstmt = conn.prepareStatement(sql);
157.                pstmt.setObject(1,name);
158.                pstmt.setObject(2,userId);
159.                rs = pstmt.executeQuery();
160.                while (rs.next()){
161.                    Consumption consumption = new Consumption();
162.                    consumption.setId(rs.getInt("id"));
163.                    consumption.setType(rs.getString("type"));
164.                    consumption.setName(rs.getString("name"));
165.                    consumption.setMoney(rs.getDouble("money"));
166.                    consumption.setDate(rs.getDate("date"));
167.                    list.add(consumption);
168.                }
169.                this.closeAll();
170.                return list;
171.            }catch (SQLException e){
172.                System.out.println("执行查询消费(name)是否存在 Sql 出错");
173.                e.printStackTrace();
174.            }
175.            return null;
176.        }
177.
178.        /*自定义查询 根据日期 查询消费
179.         * sql 带? 号的 sql 语句
180.         * params ? 中具体的参数
181.         * @return
182.         */
183.        public List<Consumption> getByNameConsumption(String sql,Date date,Int
    eger userId) throws Exception {
184.            //获取一个连接
185.            init();
186.            List<Consumption> list = new ArrayList<Consumption>();
187.            //根据 sql 和连接，获取一个声明
188.            try {
189.                pstmt = conn.prepareStatement(sql);
190.                pstmt.setObject(1,date);
191.                pstmt.setObject(2,userId);
192.                rs = pstmt.executeQuery();
193.                while (rs.next()){
194.                    Consumption consumption = new Consumption();
195.                    consumption.setId(rs.getInt("id"));
196.                    consumption.setType(rs.getString("type"));
197.                    consumption.setName(rs.getString("name"));
198.                    consumption.setMoney(rs.getDouble("money"));
199.                    consumption.setDate(rs.getDate("date"));
200.                    list.add(consumption);
201.                }
202.                this.closeAll();
203.                return list;
204.            }catch (SQLException e){
205.                System.out.println("执行查询消费(date)是否存在 Sql 出错");
```

```
206.                 e.printStackTrace();
207.             }
208.             return null;
209.         }
210.
211.
212.         /*自定义查询 根据类型 统计消费总额
213.          * sql 带? 号的 sql 语句
214.          * params ? 中具体的参数
215.          * @return
216.          */
217.         public Double statisticsType(String sql,String type,Integer id) throws
     Exception {
218.             //获取一个连接
219.             init();
220.             Double sumMoney=0.0;
221.             //根据 sql 和连接，获取一个声明
222.             try {
223.                 pstmt = conn.prepareStatement(sql);
224.                 pstmt.setObject(1,type);
225.                 pstmt.setObject(2,id);
226.                 rs = pstmt.executeQuery();
227.                 if (rs.next()){
228.                     sumMoney = rs.getDouble("sumMoney");
229.                 }
230.                 this.closeAll();
231.                 return sumMoney;
232.             }catch (SQLException e){
233.                 System.out.println("执行统计消费(type)是否存在 Sql 出错");
234.                 e.printStackTrace();
235.             }
236.             return null;
237.         }
238.
239.         /*自定义查询 根据消费 ID 查询消费
240.          * sql 带? 号的 sql 语句
241.          * params ? 中具体的参数
242.          * @return
243.          */
244.         public Consumption getByIdConsumption(String sql,Integer id) throws Ex
    ception {
245.             //获取一个连接
246.             init();
247.             //根据 sql 和连接，获取一个声明
248.             try {
249.                 pstmt = conn.prepareStatement(sql);
250.                 pstmt.setObject(1,id);
251.                 rs = pstmt.executeQuery();
252.                 while (rs.next()){
253.                     Consumption consumption = new Consumption();
254.                     consumption.setId(rs.getInt("id"));
```

```
255.                    consumption.setType(rs.getString("type"));
256.                    consumption.setName(rs.getString("name"));
257.                    consumption.setMoney(rs.getDouble("money"));
258.                    consumption.setDate(rs.getDate("date"));
259.                    consumption.setUserId(rs.getInt("user_id"));
260.                    this.closeAll();
261.                    return consumption;
262.                }
263.            }catch (SQLException e){
264.                System.out.println("执行查询消费(id)是否存在 Sql 出错");
265.                e.printStackTrace();
266.            }
267.            return null;
268.        }
269.    }
```

4.2 登录注册功能实现

4.2.1 登录模块的实现

图 4-1 登录界面

用户输入账号密码，账号密码通过 POST 请求传入 UserServlet 中进行校验，如果账号和密码都匹配，则登录成功。

表 4-2 index.jsp 部分代码

```
1.<div style="margin-right: auto;margin-left: auto;width: 30%;text-align: center;
    ">
```

```
<div class="animated fadeInRight" style="margin-right: auto;margin-left: auto;">
    <form action="user?action=login" method="post" id="loginForm" onsubmit="return dosubmit()">
        <h3 style="border-bottom:1px dashed #999;padding-bottom:10px;">用户登录</h3>
        <br>
        <p class="text-left">用户名:<input type="text" name="username" id="username"/></p>
        <p class="text-left">密 码:<input type="password" name="password" id="password"/></p>
        <p class="text-left"></p>
        <button type="submit" class="button-grey animated fadeInUp">登录</button>
          没有账号? <a class="link-contact" href="register.jsp">注册一个</a>
    </form>
  </div>
</div>
```

表 4-3 UserServlet 部分代码

```
if("login".equals(action)){
            try {
                user = userDao.login(username,password);
                if(user!=null){
                    session.setAttribute("user",user);
                    response.sendRedirect("pages/expenditure?action=list");
                }else {
                    response.setCharacterEncoding("utf-8");
                    response.setContentType("text/html; charset=utf-8");
                     PrintWriter out = response.getWriter();
                     out.println("<html>");
                     out.println("<head>");
                     out.println("</head>");
                     out.println("<body>");
                     out.println("<script type='text/javascript'>");
                     out.println("alert('用户名或密码错误! ');");
                     out.println("window.location.href='index.jsp';");
                     out.println("</script>");
                     out.println("</body>");
                     out.println("</html>");
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
```

4.2.2 注册模块的实现

注册需要填写账号和两次密码，只有两次密码都输入一致才能注册成功。

图4-2 注册界面

用户输入账号密码，账号密码通过 POST 请求传入 UserServlet 中进行校验，把数据插入到 t_user 表中。

表 4-4 Register.jsp 相关代码

```
<div style="margin-right: auto;margin-left: auto;width: 30%;text-align: center;
">
      <div class="animated fadeInRight" style="margin-right: auto;margin-left:
 auto;">
          <form action="user?action=register" method="post" id="loginForm" on
submit="return dosubmit()">
              <h3 style="border-bottom:1px dashed #999;padding-bottom:10px;">
用户注册</h3>
              <br>
              <p class="text-left">用户
名:<input type="text" name="username" id="username"/></p>
              <p class="text-left">
密 码:<input type="password" name="password" id="password"/></p>
              <p class="text-left">确认密
码:<input type="password" name="repeatPassword" id="repeatPassword"/></p>
              <p class="text-left"></p>
               <button type="submit" class="button-grey animated fadeInUp">注
册</button>
              已有账号? <a class="link-contact" href="index.jsp">去登录</a>
          </form>
      </div>
  </div>
```

表 4-5 UserServlet 部分代码

```
if("register".equals(action)){
    user.setUsername(username);
```

```
3.    user.setPassword(password);
4.    try {
5.        if(password.equals(repeatPassword)){
6.            if(userDao.register(user)==true){
7.                response.setCharacterEncoding("utf-8");
8.                response.setContentType("text/html; charset=utf-8");
9.                PrintWriter out = response.getWriter();
10.                out.println("<html>");
11.                out.println("<head>");
12.                out.println("</head>");
13.                out.println("<body>");
14.                out.println("<script type='text/javascript'>");
15.                out.println("alert('注册成功');");
16.                out.println("window.location.href='index.jsp';");
17.                out.println("</script>");
18.                out.println("</body>");
19.                out.println("</html>");
20.            }else {
21.                response.setCharacterEncoding("utf-8");
22.                response.setContentType("text/html; charset=utf-8");
23.                PrintWriter out = response.getWriter();
24.                out.println("<html>");
25.                out.println("<head>");
26.                out.println("</head>");
27.                out.println("<body>");
28.                out.println("<script type='text/javascript'>");
29.                out.println("alert('用户名已存在');");
30.                out.println("window.location.href='register.jsp';");
31.                out.println("</script>");
32.                out.println("</body>");
33.                out.println("</html>");
34.            }
35.        }else {
36.            response.setCharacterEncoding("utf-8");
37.            response.setContentType("text/html; charset=utf-8");
38.            PrintWriter out = response.getWriter();
39.            out.println("<html>");
40.            out.println("<head>");
41.            out.println("</head>");
42.            out.println("<body>");
43.            out.println("<script type='text/javascript'>");
44.            out.println("alert('两次输入的密码不一致');");
45.            out.println("window.location.href='register.jsp';");
46.            out.println("</script>");
47.            out.println("</body>");
48.            out.println("</html>");
49.        }
50.    } catch (Exception e) {
51.        e.printStackTrace();
52.    }
```

4.3 消费管理功能实现

4.2.2 消费管理模块的实现

当用户成功登入系统，效果如图 4-3，图 4-4 所示，代码如表 4-6, 表 4-7，表 4-8 所示。

图 4-3 管理所有消费界面

图 4-4 消费统计界面

表 4-6 ConsumptionServlet 代码

```
1.@WebServlet("/pages/expenditure")
2.public class ConsumptionServlet extends HttpServlet {
3.
```

```
4.     protected void doPost(HttpServletRequest request, HttpServletResponse resp
   onse) throws UnsupportedEncodingException {
5.         request.setCharacterEncoding("UTF-8");
6.         response.setContentType("text/html;charset=utf-8");
7.         response.setCharacterEncoding("utf-8");
8.
9.         ConsumptionDao consumptionDao = new ConsumptionDao();
10.          HttpSession session = request.getSession();
11.          User user = (User) session.getAttribute("user");
12.          String action = request.getParameter("action");
13.          String searchType = request.getParameter("searchType");
14.          String key = request.getParameter("key");
15.
16.          List<Consumption> list = new ArrayList<Consumption>();
17.
18.          String id = request.getParameter("id");
19.          String type = request.getParameter("type");
20.          String money = request.getParameter("money");
21.          String name = request.getParameter("name");
22.          String createTime = request.getParameter("date");
23.          Integer userId = user.getId();
24.          //修改消费前
25.          if ("modify".equals(action)) {
26.              try {
27.                  Consumption consumption = consumptionDao.getByIdConsumption(I
   nteger.valueOf(id));
28.                  request.setAttribute("consumption", consumption);
29.                  request.getRequestDispatcher("expenditureEdie.jsp").forward(r
   equest, response);
30.              } catch (Exception e) {
31.                  e.printStackTrace();
32.              }
33.          }
34.          //修改消费后
35.          if ("modifyAfter".equals(action)) {
36.              Consumption consumption = new Consumption(Integer.valueOf(id), ty
   pe, name, Double.parseDouble(money), Date.valueOf(createTime), userId);
37.              try {
38.                  if (consumptionDao.updateConsumption(consumption)) {
39.                      response.setCharacterEncoding("utf-8");
40.                      response.setContentType("text/html; charset=utf-8");
41.                      PrintWriter out = response.getWriter();
42.                      out.println("<html>");
43.                      out.println("<head>");
44.                      out.println("</head>");
45.                      out.println("<body>");
46.                      out.println("<script type='text/javascript'>");
47.                      out.println("alert('修改成功');");
48.                      out.println("window.location.href='expenditure?action=lis
   t';");
49.                      out.println("</script>");
50.                      out.println("</body>");
51.                      out.println("</html>");
```

```
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        //根据消费日期查询
        if ("search".equals(action) && "date".equals(searchType)) {
            try {
                list = consumptionDao.getByDateConsumption(Date.valueOf(key), userId);
                request.setAttribute("consumptionList", list);
                request.getRequestDispatcher("statistics.jsp").forward(request, response);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        //根据消费名称查询
        if ("search".equals(action) && "name".equals(searchType)) {
            try {
                list = consumptionDao.getByNameConsumption(key,userId);
                request.setAttribute("consumptionList", list);
                request.getRequestDispatcher("statistics.jsp").forward(request, response);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        if ("del".equals(action)) {
            try {
                if (consumptionDao.delByConsumptionId(Integer.valueOf(id))) {

                    response.setCharacterEncoding("utf-8");
                    response.setContentType("text/html; charset=utf-8");
                    PrintWriter out = response.getWriter();
                    out.println("<html>");
                    out.println("<head>");
                    out.println("</head>");
                    out.println("<body>");
                    out.println("<script type='text/javascript'>");
                    out.println("alert('删除成功');");
                    out.println("window.location.href='expenditure?action=list';");
                    out.println("</script>");
                    out.println("</body>");
                    out.println("</html>");
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        if ("list".equals(action)) {
            try {
```

```
                    list = consumptionDao.getByUserIdConsumption(user.getId());

                    request.setAttribute("consumptionList", list);
                    request.getRequestDispatcher("expenditure.jsp").forward(re
quest, response);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
            if ("add".equals(action)) {
                Consumption consumption = new Consumption(0, type, name, Doubl
e.parseDouble(money), Date.valueOf(createTime), userId);
                try {
                    if (consumptionDao.addConsumption(consumption)) {
                        response.setCharacterEncoding("utf-8");
                        response.setContentType("text/html; charset=utf-8");
                        PrintWriter out = response.getWriter();
                        out.println("<html>");
                        out.println("<head>");
                        out.println("</head>");
                        out.println("<body>");
                        out.println("<script type='text/javascript'>");
                        out.println("alert('添加成功');");
                        out.println("window.location.href='expenditure?action=
list';");
                        out.println("</script>");
                        out.println("</body>");
                        out.println("</html>");
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }


        }

        protected void doGet(HttpServletRequest request, HttpServletResponse r
esponse) throws UnsupportedEncodingException {
            this.doPost(request, response);
        }

    }
```

表 4-7 expenditure.jsp 代码

```
<div class="col-xs-10">
    <form action="expenditure?action=add" method="post">
    <div class="add">
        <label class="add-label">添加消费:</label>
        <select class="add-select" id="type" name="type">
            <option selected hidden disabled value="">选择消费类型</option>
```

```
            <option value="餐饮美食">餐饮美食</option>
            <option value="家居服装">家居服装</option>
            <option value="文化休闲">文化休闲</option>
              <option value="交通出行">交通出行</option>、
              <option value="美容美发">美容美发</option>
        </select>
        <input class="add-money" placeholder="金额
" id="money" name="money">
        <input class="add-money-remark" placeholder="名称
" id="name" name="name">
        <input class="add-money-remark" onClick="WdatePicker({el:this})" plac
eholder="日期" id="date" name="date" readonly="readonly">
        <button class="add-button" type="submit">添加</button>
        <label class="add-label" style="margin-left: 70px">用户:
<%=user.getUsername()%></label>
    </div>
    </form>


    <div>
        <table class="table table-bordered">
            <thead>
                <tr>
                    <th>日期</th>
                    <th>消费名称</th>
                    <th>消费金额</th>
                    <th>消费类型</th>
                    <th>操作</th>
                </tr>
            </thead>
            <tbody>
                <c:forEach items="${consumptionList}" end="5" var="consumptio
n">
                    <tr>
                        <td>${consumption.date}</td>
                        <td>${consumption.name}</td>
                        <td>${consumption.money}</td>
                        <td>${consumption.type}</td>
                        <td>
                            <a href="expenditure?action=modify&id=${consumpti
on.id}">修改</a>
                            <a href="expenditure?action=del&id=${consumption.
id}">删除</a>
                        </td>
                    </tr>
                </c:forEach>
            </tbody>
        </table>
    </div>

</div>
```

表 4-8 statics.jsp 代码

```
<div class="col-xs-10">
    <form action="expenditure?action=search" method="post">
        <div class="selectAndSearch">
            <select class="select" id="searchType" name="searchType">
                <option selected hidden disabled value="">选择消费类型
</option>
                <option value="name">名称</option>
                <option value="date">日期</option>
            </select>
            <input name="key" class="searchInput" placeholder="搜索对象">
             <button class="add-button" type="submit">搜索</button>
        </div>
    </form>
    <div class="total">
        <label class="total-label1">餐饮美食: </label>
        <label class="total-money1"><%=consumptionDao.statisticsType("餐饮美食
",user.getId()) %>元</label>
        <label class="total-label2">家居服装: </label>
        <label class="total-money2"><%=consumptionDao.statisticsType("家居服装
",user.getId()) %>元</label>
        <label class="total-label2">文化休闲: </label>
        <label class="total-money2"><%=consumptionDao.statisticsType("文化休闲
",user.getId()) %>元</label>
        <label class="total-label2">交通出行: </label>
        <label class="total-money2"><%=consumptionDao.statisticsType("交通出行
",user.getId()) %>元</label>
        <label class="total-label2">美容美发: </label>
        <label class="total-money2"><%=consumptionDao.statisticsType("美容美发
",user.getId()) %>元</label>
    </div>
    <div class="borrowTable">
        <table class="table table-bordered">
            <thead>
            <tr>
                <th>日期</th>
                <th>消费名称</th>
                <th>消费金额</th>
                <th>消费类型</th>
                <th>操作</th>
            </tr>
            </thead>
            <tbody>
            <c:forEach items="${consumptionList}" end="5" var="consumption">

                <tr>
                    <td>${consumption.date}</td>
                    <td>${consumption.name}</td>
                    <td>${consumption.money}</td>
                    <td>${consumption.type}</td>
```

```
43.                     <td><a href="expenditure?action=del&id=${consumption.id}"
    >删除</a></td>
44.                 </tr>
45.             </c:forEach>
46.             </tbody>
47.         </table>
48.     </div>
49. </div>
```

4. 消费管理系统实现的不足

不足之处在于对于前端传过来的数据没有很好的进行验证。