

**2020年秋季学期**

**计算机学院大三**

**计算机系统安全课程**

**Lab 4实验报告**

**完整性访问控制系统设计与实现**

|  |  |
| --- | --- |
| 姓名 | 余涛 |
| 学号 | 1180300829 |
| 班号 | 1803202 |
| 电子邮件 | 1063695334@qq.com |
| 手机号码 | 15586430583 |

# 1. 实验目的

（1）配合第7章，为商业公司设计系统，提出针对该公司业务需求的应用系统安全策略。安全策略中要明确指明对公司的要求与约束,和对客户的要求与约束,区分各自的责任。（当出现商业公司与客户间意见分歧或法律纠纷时，安全策略可作为仲裁依据）

（2）配合第9章 为商业公司设计系统，应用系统满足完整性需求。需求中包含责任分离、功能分离、审计。

（3）具体指明是哪类应用系统，应用背景范围不限，可以是银行、股票等，符合商业系统完整性需求即可。

（4）4学时，每人独立完成。

# 2. 实验环境搭建

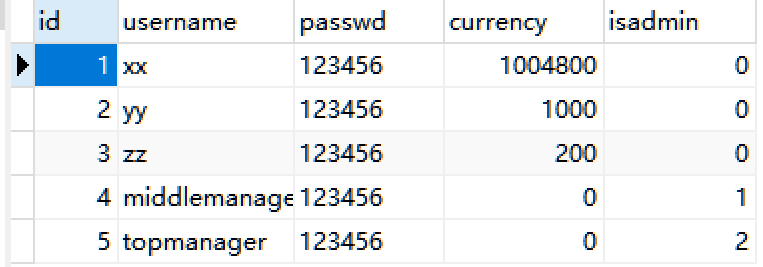
Windows10操作系统，MySQL关系数据库管理系统，web开发中python的Django框架。

# 3. 系统要求

## 3.1 给出应用系统的安全策略文档

首先是登录界面，可以进行登陆操作，登录用户包括普通用户、中级管理员和顶级管理员，系统通过用户名和密码然后查询数据库进行对比来判断用户类型，进而跳转到相应的不同类型用户的操作界面。在数据库中包含以下内容：

用户id，用户名，登录密码，存款余额，用户类型（0为普通用户、1为中级管理员、2位顶级管理员），如下所示：



对于普通用户来说，可以进行存款、取款和查看自己的余额。

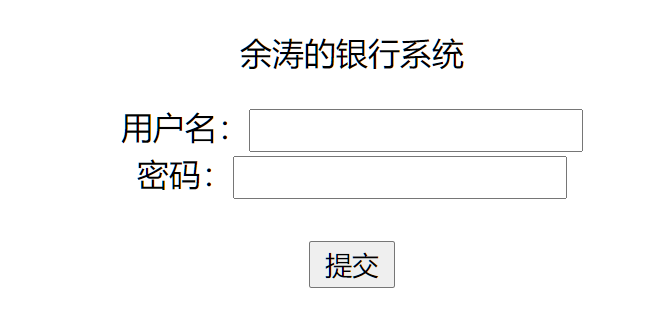
而对于中级管理员和顶级管理员来说，只能查看所有用户的余额和同意或拒绝用户的存取款操作。

当用户申请存款或取款后，首先需要输入相应的金额，选择操作类型，然后就会弹出管理员登录的界面，需要相应的管理员登录自己的账号进行确认，管理员决定是否同意用户的申请，中级管理员只有100000元以下金额操作的同意或拒绝权限，而顶级管理员有超过100000元的金额操作或拒绝权限。管理员完成操作后，返回到用户主页。

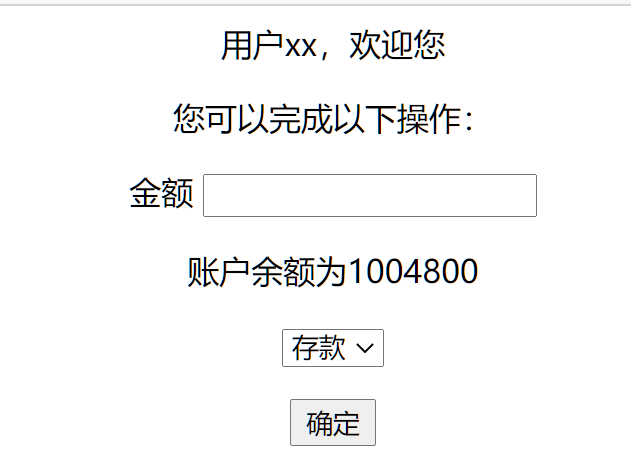
## 3.2 提供交互界面，能够完成录入、查询等功能

界面基于web：

（1）登录界面：



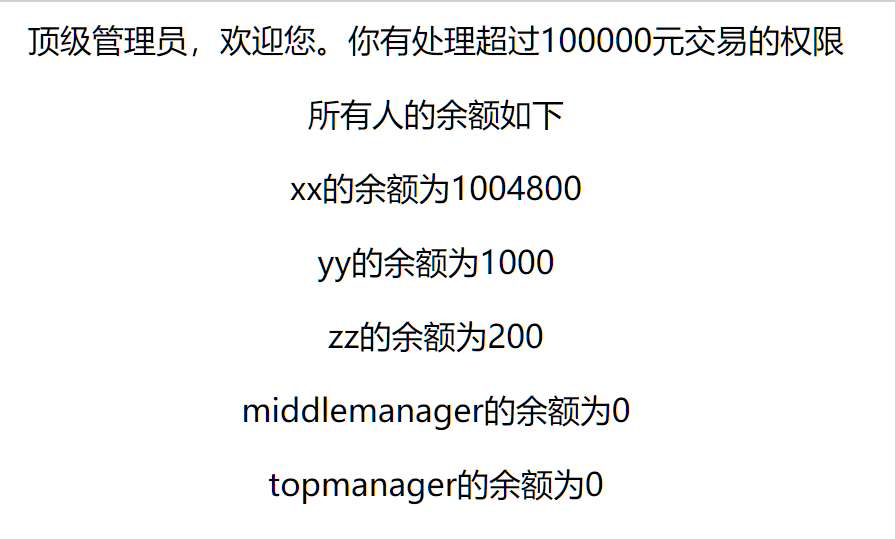
（2）普通用户登录：



（3）中级管理员登录：

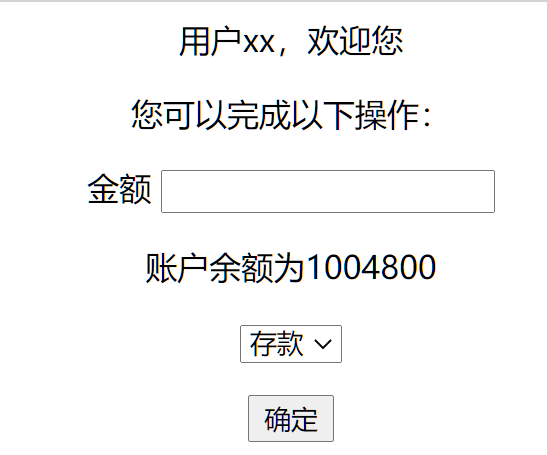


（4）顶级管理员登录：

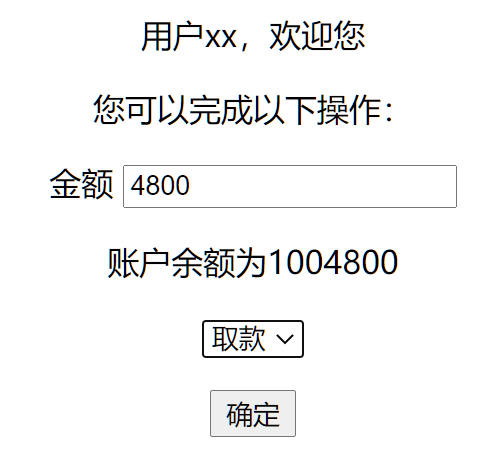


（5）普通用户进行金额小于100000元的取款操作：

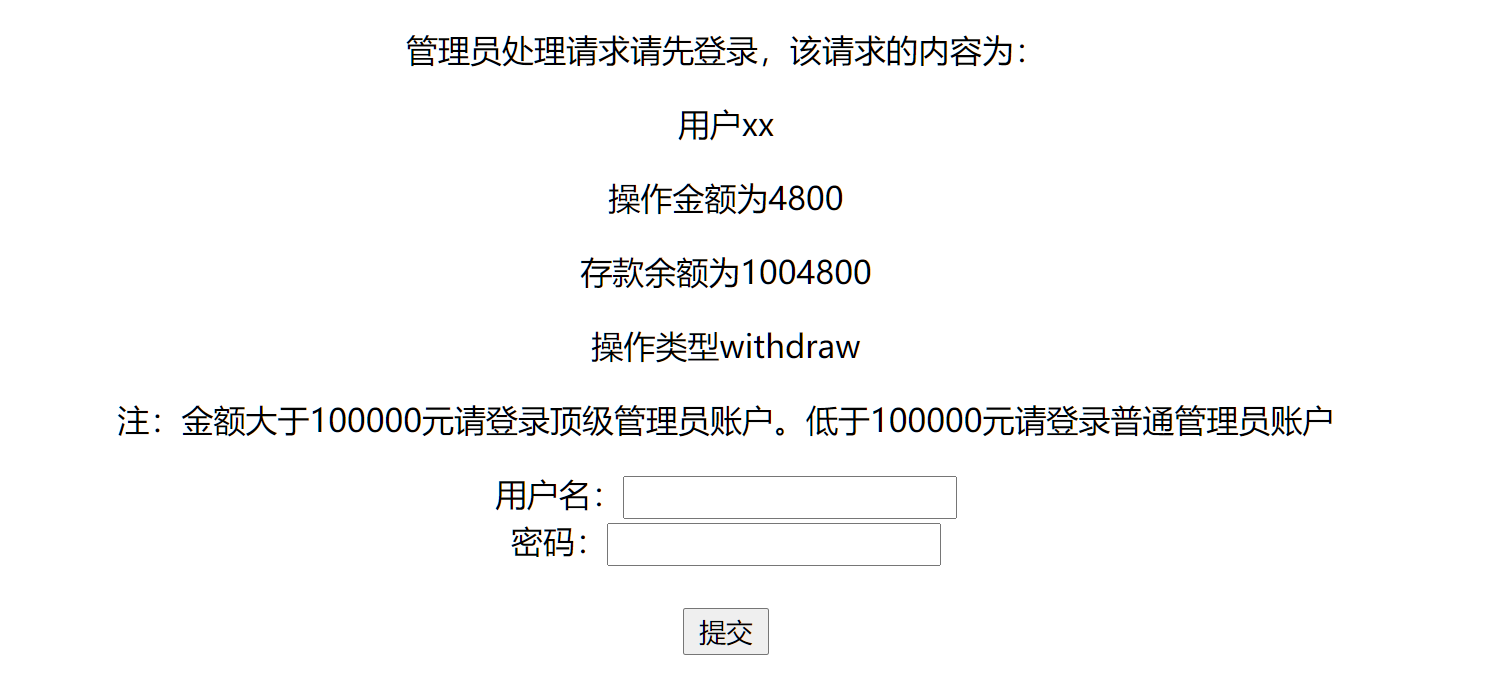
首先登录xx的账户：



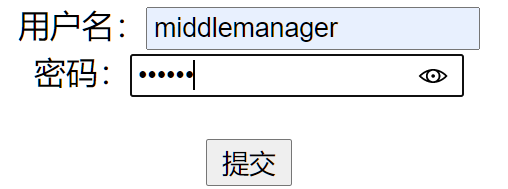
选择取款金额为4800并确定：



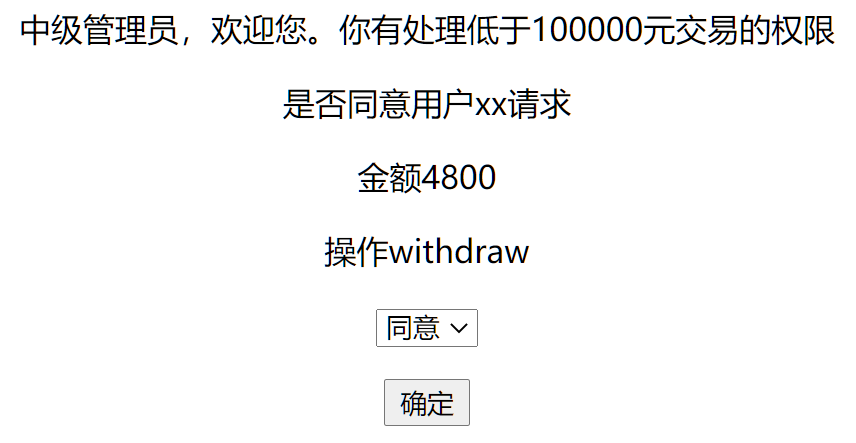
然后弹出管理员登录界面：



根据操作金额登录中级管理员账户：

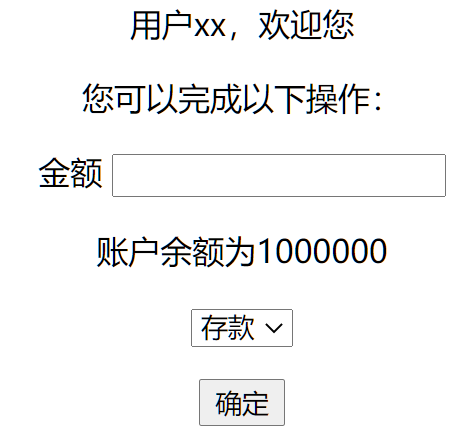


然后中级管理员选择同意或拒绝：

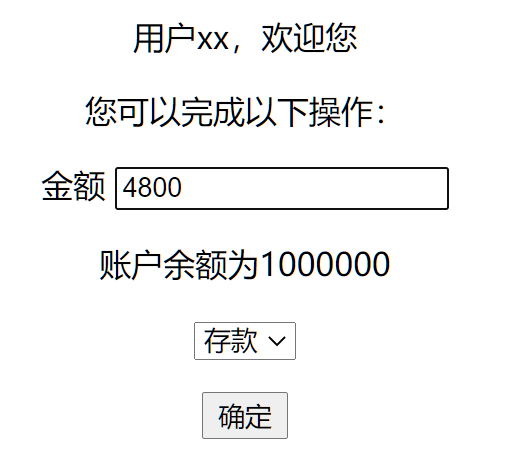




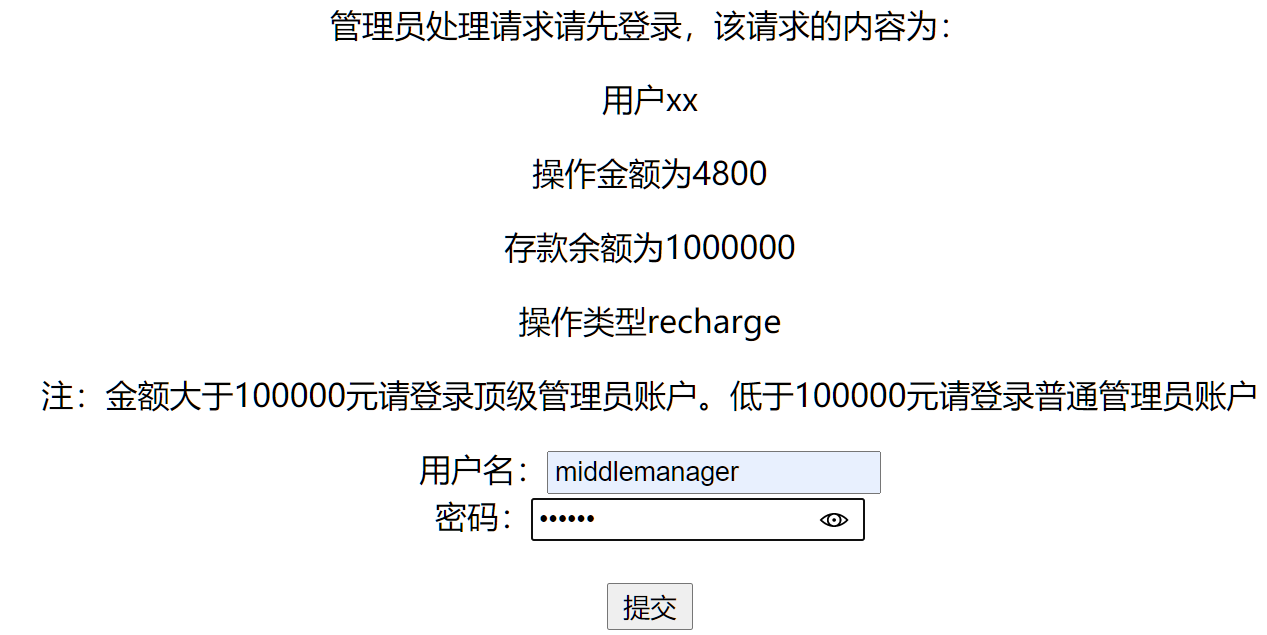
此时用户余额已减少4800：



（6）普通用户进行金额小于100000元的存款操作：



点击确认后弹出管理员确认页面：

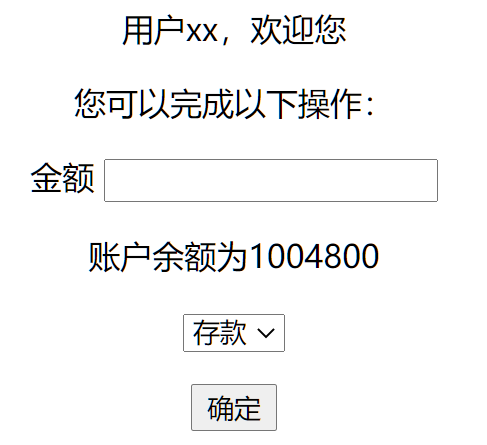


登录中级管理员账户后同意请求：





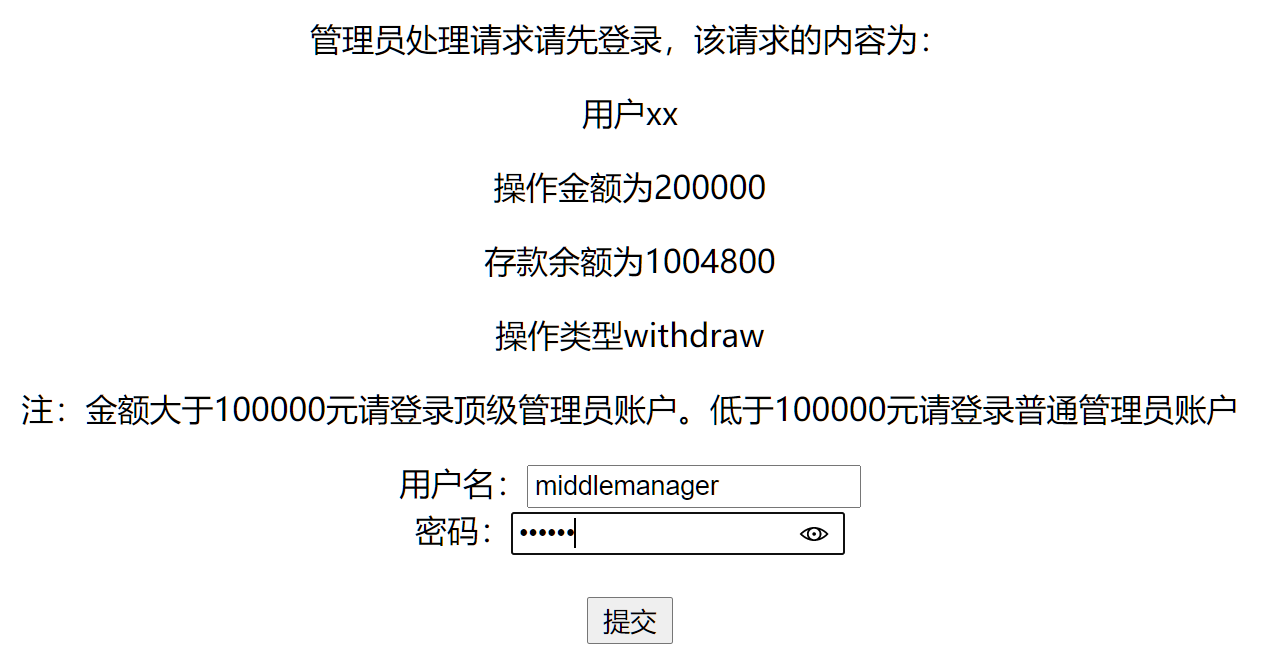
存款成功：



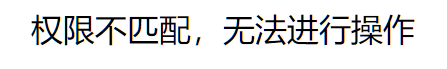
（6）用户进行金额超过100000元的存款或取款操作：

所有操作与前面相似，唯一区别是需要登录顶级管理员账户，此时中级管理员不再具有权限。

如果登录中级管理员此时会提示权限不匹配：



此时弹出的窗口如下：



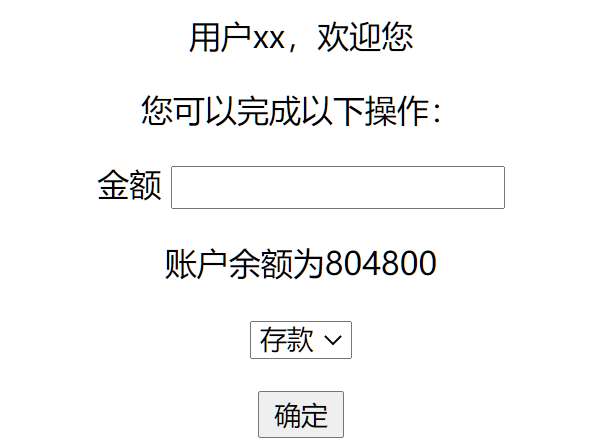
此时需要登录高级管理员用户：



管理员选择同意或拒绝请求：



以同意请求为例：



其他操作与前面相似。

## 3.3 满足责任分离、功能分离原则

责任分离要求：当需要多步完成一个事物时，需要两个以上人员共同完成。存款与取款后，都需要向管理员发出请求，此时管理员具有同意或拒绝请求的权利。对于金额小于100000元的请求由中级管理员进行确认，高于100000元的操作由顶级管理员进行确认。当只有管理员登录时，管理员无法完成存款或者取款操作，没有修改用户余额的权限，只能查看用户的余额，满足了责任分离原则，从而保证了数据的完整性。

功能分离原则：当开发数据时不应该污染产品数据，并且将数据库导出进行了备份，这样就防止了对产品数据的污染。

## 3.4 保存审计日志

功将系统的每一步操作都通过日志记录在journal.txt文件中，这样就完成了审计日志的保存：



## 3.5 遵循Clark-Wilson模型，定义应用系统的完整性限制条件

Clark-Wilson模型考虑如下几点：

（1）主体必须被识别和认证

（2）客体只能通过规定的程序进行操作

（3）主体只能执行规定的程序

（4）必须维护正确的审计日志

（5）系统必须被证明能够正确工作

具体如下：

1. 程序将中级和顶级管理员当做主体、普通用户为客体，此时管理员的登录过程就是识别和认证的过程，通过识别和认证的管理员才能进行管理员权限的操作，满足第一条。
2. 普通用户作为客体只能执行申请存款和取款和查询余额的操作，不能完成其他的操作，满足第二条。
3. 管理员作为主体只能够只能同意或者拒绝用户的请求，无法单独对用户完成存款或者取款的操作，满足第三条。
4. 用journal.txt文件来保存审计日志，满足第四条。
5. 系统正确运行，没有其他问题，满足第五条。

## 3.6 遵循Clark-Wilson模型的证明规则和实施规则

证明规则1：

当任意 IVP 运行时, 它必须保证所有的CDI处于有效状态

当用户登录时，在没有管理员同意的情况下，不能对数据库中的数据进行操作。只能提交账单申请，待管理员同意后，才能更改数据库。

证明规则2：

对相关联的CDI, 一个TP必须将这些CDI从一个有效状态转到另一个有效状态

在管理员同意后，账单就会被删除。这个交易过程，会将用户的申请状态，变为完成状态或错误状态（被管理员拒绝或读取/存入数据库失败）。

证明规则3：

系统执行操作时，符合责任分离原则。模型需要保证用户身份和执行代码身份一致。所以需要验证身份。这里设计的验证身份就是“登录”。

实施规则1：

系统要维护关联关系，保证经过验证的TP操作相应的CDI。在用户提出存取款申请后，管理员同意，就代表该账单已经被验证。被验证的这个账单可以对数据库中，相应的存款金额进行更改。

实施规则2：

TP操作CDI时，保证操作用户有权对相应CDI做操作，TP所代表的用户是CDI的真实用户。经过验证的账单，即管理员同意后，可以对数据库中的 CDI（即用户的存款金额）进行更改。

实施规则3：

系统执行操作时，符合责任分离原则。模型需要保证用户身份和执行代码身份一致。满足责任分离原则，用户和管理员都不能单独对存款金额进行更改，只有用户申请，管理员同意后，才能进行操作。

实施规则4：

只有可以授予TP访问规则的主体才能修改列表中相应的表项，授权主体不能执行TP操作。只有用户提出申请，才能进行 TP 操作。授权的管理员，没有执行 TP 操作的能力。

# 4. 系统设计

设计了一个数据库表：bankuser，表结构如下：

|  |  |
| --- | --- |
| id | 用户唯一id |
| username | 用户名 |
| passwd | 用户密码 |
| currency | 用户账户余额 |
| isadmin | 用户类型 |

# 5. 源代码

由于使用的是web开发中的django框架进行开发，所以只粘贴后端的处理前端请求的代码：

1. **import** re
3. **import** pymysql
4. **from** django.shortcuts **import** render, redirect
5. **import** time
6. **from** appdata.models **import** bankuser

9. **def** signin(request):
10. # 用于登录
11. **if** request.method == "POST":
12. username = request.POST.get("username")
13. passwd = request.POST.get("passwd")
14. **if** username == "manager" **and** passwd == "yutao19981119":  # 分配权限的管理员
15. # request.session['username'] = username
16. # request.session['info'] = ""
17. **return** redirect("/bank/admin")
18. **else**:
19. user = bankuser.objects.get(username=username)
20. **if** user.passwd == passwd:
21. **if** user.isadmin == 0:
22. journal = open('journal.txt', 'a')  # 写日志
23. localtime = time.asctime(time.localtime(time.time()))
24. journal.write(localtime+" 用户"+user.username+"从主登录页面登录\n")  # 写入日志
25. **return** redirect("/bank/users/"+username)
26. **if** user.isadmin == 1:
27. journal = open('journal.txt', 'a')  # 写日志
28. localtime = time.asctime(time.localtime(time.time()))
29. journal.write(localtime+" 中级管理员" + user.username + "从主登录页面登录\n")  # 写入日志
30. **return** redirect("/bank/middlemanager\_look/"+username)
31. **if** user.isadmin == 2:
32. journal = open('journal.txt', 'a')  # 写日志
33. localtime = time.asctime(time.localtime(time.time()))
34. journal.write(localtime+" 顶级管理员" + user.username + "从主登录页面登录\n")  # 写入日志
35. **return** redirect("/bank/topmanager\_look/"+username)
36. **else**:
37. **return** render(request, "appdata/signin.html")

40. **def** admin(request):
41. # 权限管理员页面的后端处理
42. **if** request.method == "POST":
43. username = request.POST.get("username")
44. rignt = request.POST.get("right")
45. table = request.POST.get("table")
46. operate = request.POST.get("operate")
47. # request.session['username'] = "manager"
48. **if** operate == "check":
49. conn = pymysql.connect(
50. host='localhost',
51. port=3306,
52. user="manager",
53. password="yutao19981119",
54. db='bank',
55. charset='utf8'
56. )
57. whom = []
58. **if** (username != ""):  # 分隔开读取的用户字符串
59. **if** (',' **in** username):
60. whom = username.split(',')
61. **else**:
62. whom.append(username)
63. temp1 = ""
64. **for** w **in** whom:
65. cursor = conn.cursor()  # 得到数据库的一个游标对象
66. sql = "show grants for " + w + "@'localhost'" + ';'
67. **print**(sql, "\n")  # 打印信息
68. cursor.execute(sql)  # 数据库执行该sql语句
69. # print(cursor)
70. priv = []
71. **for** i **in** cursor:
72. priv.append(tuple(re.split(r' TO ', str(\*i))[0].split(r' ON ')))
74. **print**(''.center(80, '~'))
75. # print("用户", username)
76. temp = "用户" + w + ":\n"
77. **for** j **in** priv:
78. privs = j[0].replace('GRANT', '')
79. privs\_info = j[1]
80. temp = temp + '{0} {1:<20} {2} {3}'.format('库(表):', privs\_info, '权限:', privs) + "\n"
81. **print**('{0} {1:<20} {2} {3}'.format('库(表):', privs\_info, '权限:', privs))
82. **print**(''.center(80, '~'))
83. **print**('\n')
84. localtime = time.asctime(time.localtime(time.time()))
85. journal = open('journal.txt', 'a')  # 写日志
86. journal.write(sql + "        " + localtime + "\n")  # 写入日志
87. conn.commit()  # 提交
88. cursor.close()
89. temp1 = temp1 + temp
90. # request.session['info'] = temp1
91. # return redirect("../admin")
92. **return** render(request, "appdata/admin.html", {'username': "manager", 'data': temp1})
94. **if** operate == "give":
95. conn = pymysql.connect(
96. host='localhost',
97. port=3306,
98. user="manager",
99. password="yutao19981119",
100. db='bank',
101. charset='utf8'
102. )
103. whom = []
104. allrignt = []
105. **if** (username != ""):  # 分隔开读取的用户字符串
106. **if** (',' **in** username):
107. whom = username.split(',')
108. **else**:
109. whom.append(username)
110. **if** (rignt!= ""):  # 分隔开读取的用户字符串
111. **if** (',' **in** rignt):
112. allrignt = rignt.split(',')
113. **else**:
114. allrignt.append(rignt)
115. temp = ""
116. **for** w **in** whom:
117. **for** r **in** allrignt:
118. cursor = conn.cursor()  # 得到数据库的一个游标对象
119. sql = 'grant ' + r + ' on bank.' + table + ' to ' + w + "@'localhost'" + ';'  # 为每个用户赋予权限
120. **print**(sql, "\n")  # 打印信息
121. cursor.execute(sql)  # 数据库执行该sql语句
122. journal = open('journal.txt', 'a')  # 写日志
123. localtime = time.asctime(time.localtime(time.time()))
124. journal.write(sql + "        " + localtime + "\n")  # 写入日志
125. conn.commit()  # 提交
126. cursor.close()
127. temp = temp + sql + "\n"
128. # request.session['info'] = temp
129. # return redirect("../admin")
130. **return** render(request, "appdata/admin.html", {'username': "manager", 'data': temp})
132. **if** operate == "withdraw":
133. conn = pymysql.connect(
134. host='localhost',
135. port=3306,
136. user="manager",
137. password="yutao19981119",
138. db='bank',
139. charset='utf8'
140. )
141. whom = []
142. allrignt = []
143. **if** (username != ""):  # 分隔开读取的用户字符串
144. **if** (',' **in** username):
145. whom = username.split(',')
146. **else**:
147. whom.append(username)
148. **if** (rignt != ""):  # 分隔开读取的用户字符串
149. **if** (',' **in** rignt):
150. allrignt = rignt.split(',')
151. **else**:
152. allrignt.append(rignt)
153. temp = ""
154. **for** w **in** whom:
155. **for** r **in** allrignt:
156. cursor = conn.cursor()  # 得到数据库的一个游标对象
157. sql = 'revoke ' + r + ' on bank.' + table + ' from ' + w + "@'localhost'" + ';'
158. **print**(sql, "\n")  # 打印信息
159. cursor.execute(sql)  # 数据库执行该sql语句
160. journal = open('journal.txt', 'a')  # 写日志
161. localtime = time.asctime(time.localtime(time.time()))
162. journal.write(sql + "        " + localtime + "\n")  # 写入日志
163. conn.commit()  # 提交
164. cursor.close()
165. temp = temp + sql + "\n"
166. # request.session['info'] = temp
167. # return redirect("../admin")
168. **return** render(request, "appdata/admin.html", {'username': "manager", 'data': temp})
169. **return** render(request, "appdata/admin.html", {'username': "manager", 'data': ""})

172. **def** users(request, username):
173. # 用户和管理员页面的后端处理
174. **if** request.method == "POST":
175. money = request.POST.get("money")
176. operate = request.POST.get("operate")
177. request.session["money"] = money
178. request.session["operate"] = operate
179. user = bankuser.objects.get(username=username)
180. journal = open('journal.txt', 'a')  # 写日志
181. localtime = time.asctime(time.localtime(time.time()))
182. journal.write(localtime+" 用户" + user.username + "请求为" + operate + ",金额为" + money + "\n")  # 写入日志
183. **return** redirect("/bank/rightsignin/"+username)
184. user = bankuser.objects.get(username=username)
185. currency = user.currency
186. request.session["currency"] = currency
187. **return** render(request, "appdata/users.html", {'username': username, 'currency': currency})

190. **def** rightsignin(request, username):
191. # 管理员登录的处理
192. **if** request.method == "POST":
193. name = request.POST.get("name")
194. passwd = request.POST.get("passwd")
195. money = request.session.get("money")
196. user = bankuser.objects.get(username=name)
197. **if** user.passwd == passwd:
198. **if** user.isadmin == 1:  # 中级管理员
199. **if** int(money) <= 100000:
200. journal = open('journal.txt', 'a')  # 写日志
201. localtime = time.asctime(time.localtime(time.time()))
202. journal.write(localtime+" 中级管理员登录了处理请求页面\n")  # 写入日志
203. **return** redirect("/bank/middlemanager/" + username)
204. **else**:
205. **return** redirect("/bank/error/" + username)
206. **if** user.isadmin == 2:  # 顶级管理员
207. journal = open('journal.txt', 'a')  # 写日志
208. localtime = time.asctime(time.localtime(time.time()))
209. journal.write(localtime + " 顶级管理员登录了处理请求页面\n")  # 写入日志
210. **return** redirect("/bank/topmanager/" + username)
211. money = request.session.get("money")
212. operate = request.session.get("operate")
213. currency = request.session.get("currency")
214. **return** render(request, "appdata/rightsignin.html", {'username': username, 'money': money, 'operate': operate, 'currency': currency})
216. **def** error(request, username):
217. **return** render(request, "appdata/error.html")
219. **def** middlemanager(request, username):
220. # 中级管理员处理请求页面
221. **if** request.method == "POST":
222. money = request.session.get("money")
223. operate = request.session.get("operate")
224. manageroperate = request.POST.get("manageroperate")
225. **if** manageroperate == "agree":  # 管理员同意
226. **if** operate == "recharge":
227. user = bankuser.objects.get(username=username)
228. money = int(money)
229. user.currency = user.currency + money
230. user.save()
231. journal = open('journal.txt', 'a')  # 写日志
232. localtime = time.asctime(time.localtime(time.time()))
233. journal.write(localtime+" 中级管理员同意了" + user.username + "请求为" + operate + ",金额为" + str(money) + "的请求，请求已执行\n")  # 写入日志
234. **return** redirect("/bank/users/" + username)
235. **if** operate == "withdraw":
236. user = bankuser.objects.get(username=username)
237. money = int(money)
238. user.currency = user.currency - money
239. user.save()
240. journal = open('journal.txt', 'a')  # 写日志
241. localtime = time.asctime(time.localtime(time.time()))
242. journal.write(localtime+" 中级管理员同意了" + user.username + "请求为" + operate + ",金额为" + str(money) + "的请求，请求已执行\n")  # 写入日志
243. **return** redirect("/bank/users/" + username)
244. **elif** manageroperate == "deny":  # 管理员拒绝
245. user = bankuser.objects.get(username=username)
246. journal = open('journal.txt', 'a')  # 写日志
247. localtime = time.asctime(time.localtime(time.time()))
248. journal.write(localtime+" 中级管理员拒绝了" + user.username + "请求为" + operate + ",金额为" + money + "的请求\n")  # 写入日志
249. **return** redirect("/bank/users/" + username)
250. money = request.session.get("money")
251. operate = request.session.get("operate")
252. **return** render(request, "appdata/middlemanager.html", {'username': username, 'money': money, 'operate': operate})

255. **def** topmanager(request, username):
256. # 顶级管理员处理请求页面
257. **if** request.method == "POST":
258. money = request.session.get("money")
259. operate = request.session.get("operate")
260. manageroperate = request.POST.get("manageroperate")
261. **if** manageroperate == "agree":  # 管理员同意
262. **if** operate == "recharge":
263. user = bankuser.objects.get(username=username)
264. money = int(money)
265. user.currency = user.currency + money
266. user.save()
267. journal = open('journal.txt', 'a')  # 写日志
268. localtime = time.asctime(time.localtime(time.time()))
269. journal.write(localtime+" 顶级管理员同意了" + user.username + "请求为" + operate + ",金额为" + str(money) + "的请求，请求已执行\n")  # 写入日志
270. request.session.clear()
271. **return** redirect("/bank/users/" + username)
272. **if** operate == "withdraw":
273. user = bankuser.objects.get(username=username)
274. money = int(money)
275. user.currency = user.currency - money
276. user.save()
277. journal = open('journal.txt', 'a')  # 写日志
278. localtime = time.asctime(time.localtime(time.time()))
279. journal.write(localtime+" 顶级管理员同意了" + user.username + "请求为" + operate + ",金额为" + str(money) + "的请求，请求已执行\n")  # 写入日志
280. request.session.clear()
281. **return** redirect("/bank/users/" + username)
282. **elif** manageroperate == "deny":  # 管理员拒绝
283. user = bankuser.objects.get(username=username)
284. journal = open('journal.txt', 'a')  # 写日志
285. localtime = time.asctime(time.localtime(time.time()))
286. journal.write(localtime+" 顶级管理员拒绝了" + user.username + "请求为" + operate + ",金额为" + money + "的请求\n")  # 写入日志
287. **return** redirect("/bank/users/" + username)
288. money = request.session.get("money")
289. operate = request.session.get("operate")
290. **return** render(request, "appdata/topmanager.html", {'username': username, 'money': money, 'operate': operate})

293. **def** middlemanager\_look(request, username):
294. # 中级管理员登录后页面
295. journal = open('journal.txt', 'a')  # 写日志
296. localtime = time.asctime(time.localtime(time.time()))
297. journal.write(localtime + " 中级管理员查看了所有用户余额\n")  # 写入日志
298. currency\_list = bankuser.objects.all()
299. **return** render(request, "appdata/middlemanager\_look.html", {'username': username, 'currency\_list': currency\_list})

302. **def** topmanager\_look(request, username):
303. # 顶级管理员登录后页面
304. journal = open('journal.txt', 'a')  # 写日志
305. localtime = time.asctime(time.localtime(time.time()))
306. journal.write(localtime + " 顶级管理员查看了所有用户余额\n")  # 写入日志
307. currency\_list = bankuser.objects.all()
308. **return** render(request, "appdata/topmanager\_look.html", {'username': username, 'currency\_list': currency\_list})

# 6. 心得体会

通过本次实验，我理解了责任分离原则和功能分离原则，知道了Clark-Wilson模型和应用系统的完整性限制条件，对系统安全有了更深的理解。