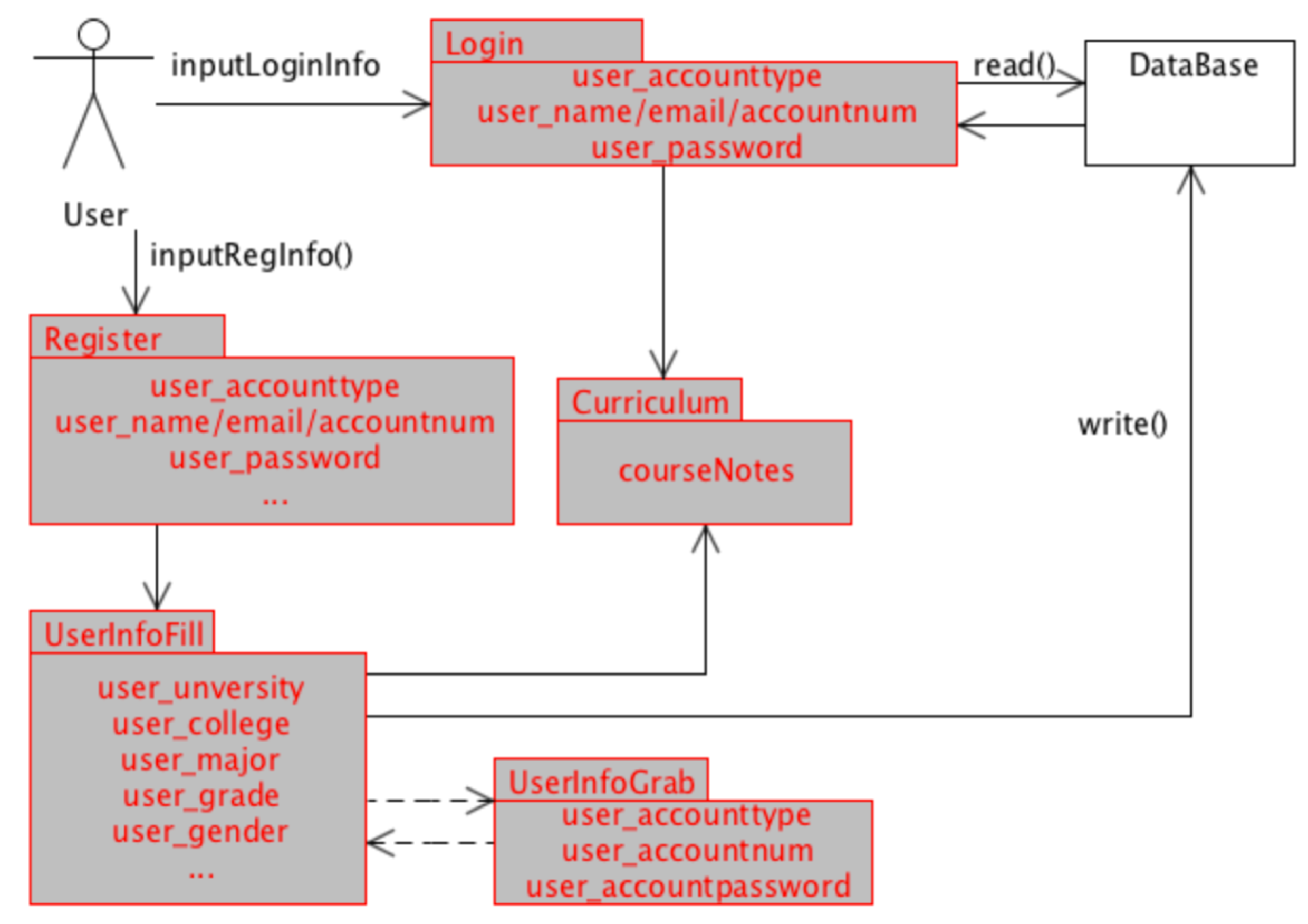
**后端原型设计**

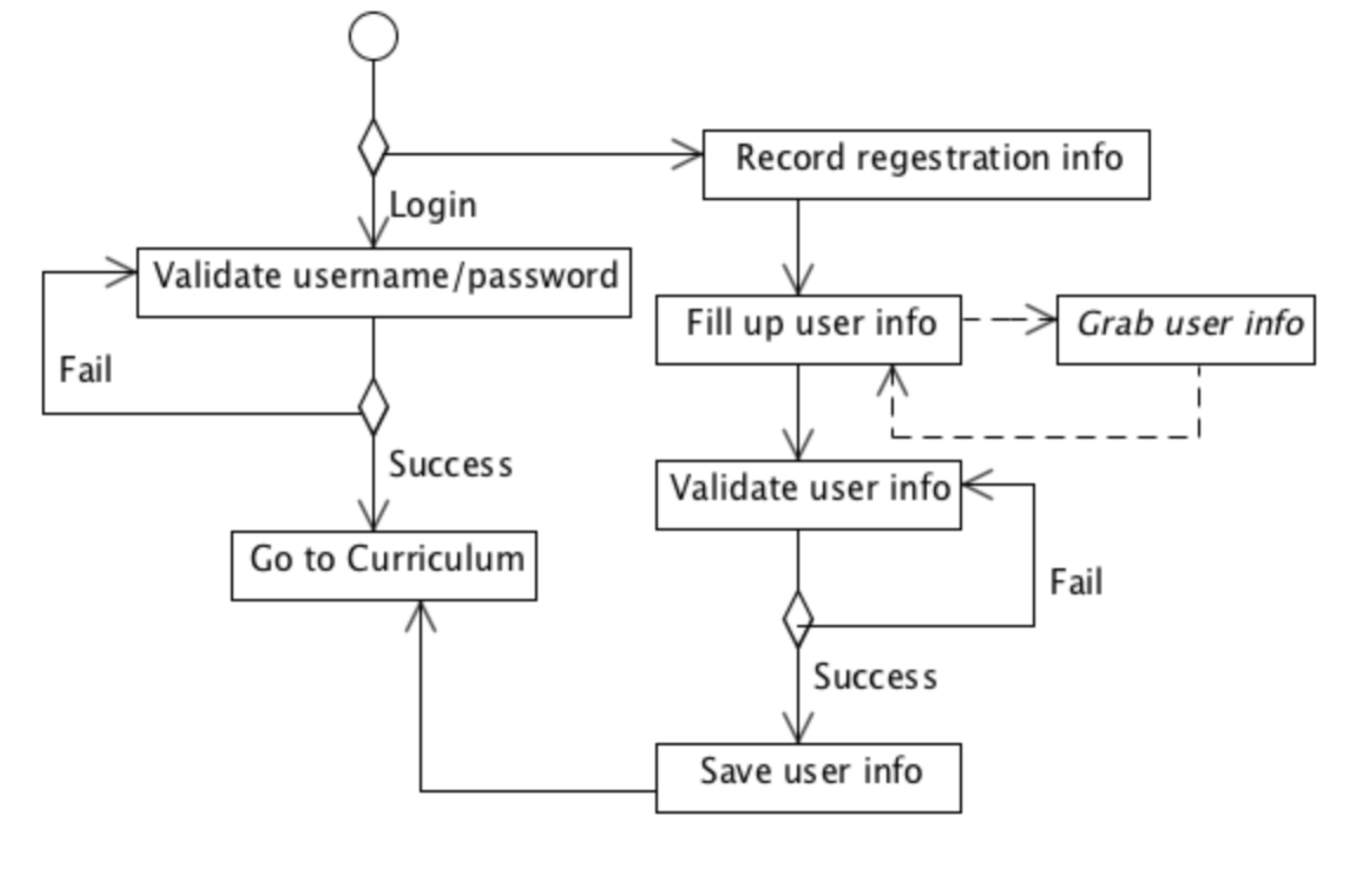
**功能设计（控制类）**

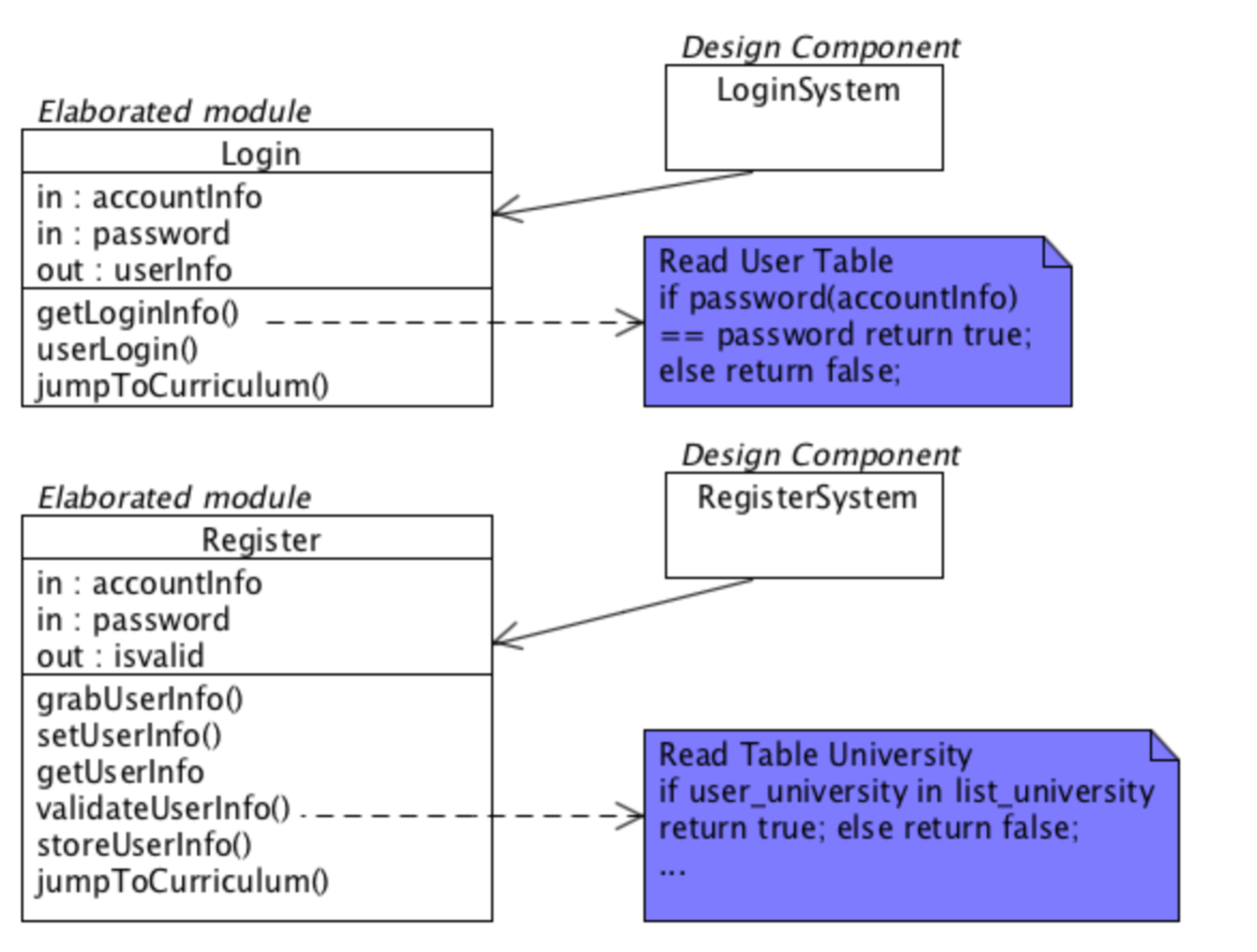
1. 用户注册与登录 (User{User})

直接注册需要用户提供邮箱、用户名、密码、重复密码、学校、院系、专业、学级，检查用户邮箱在数据库中的唯一性、密码的合法性；关联注册需要用户提供相关网站的账号及登录密码，系统尝试登录该社交网络，尝试获取用户名、学校、院系、专业、学级信息，登录出错需给出反馈，获取信息不完全需要给出反馈；抓取相关学校、院系、专业及学级信息并格式化，用以检查用户输入的合法性；将用户信息存入数据库；接受用户登录信息，与数据库信息对比给予反馈。



注册登录功能整体类图（control entity design diagram）

注册登录功能行为流程图（activity diagram）



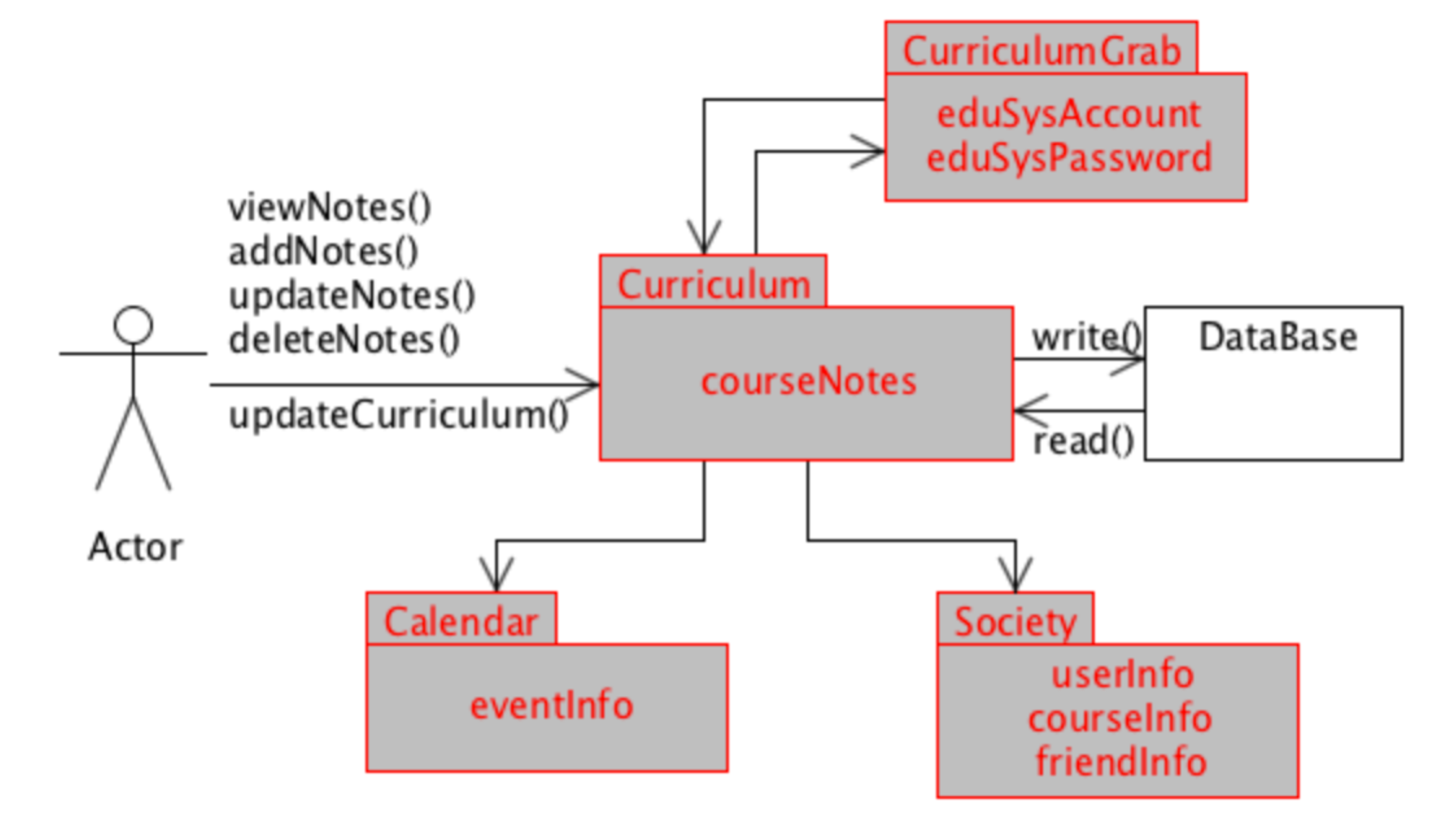
登录功能组件级设计图（component-level design diagram）

1. 学校信息抓取与保存 (Grab{University, College, Major, Grade, Course, Professor})

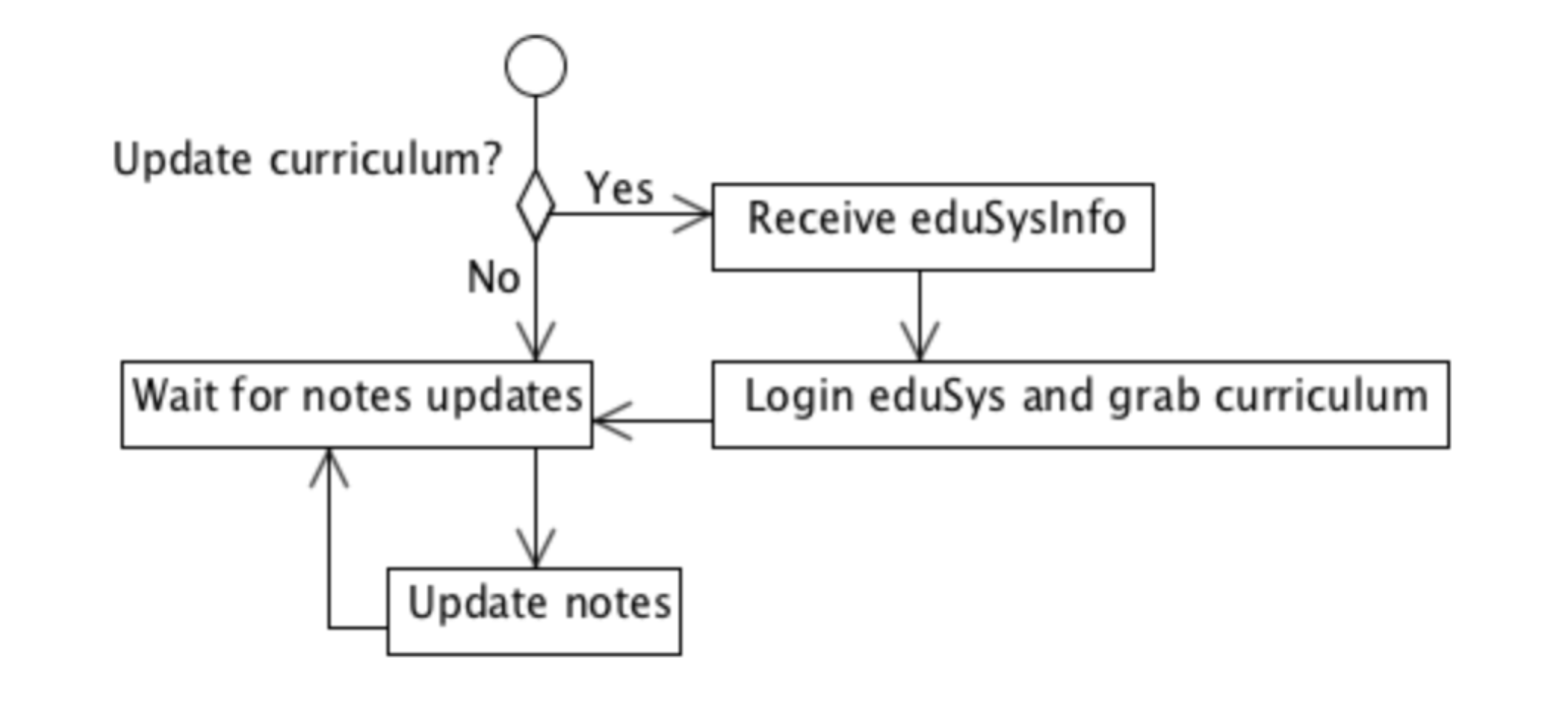
定时抓取更新学校教务系统网站中的学院、专业、年级、课程、教授信息。此模块是纯后台模块，不涉及与前端及用户交互，故不提供设计细节，仅提供数据接口信息。

1. 用户课表信息抓取与呈现 (Curriculum{Curriculum})

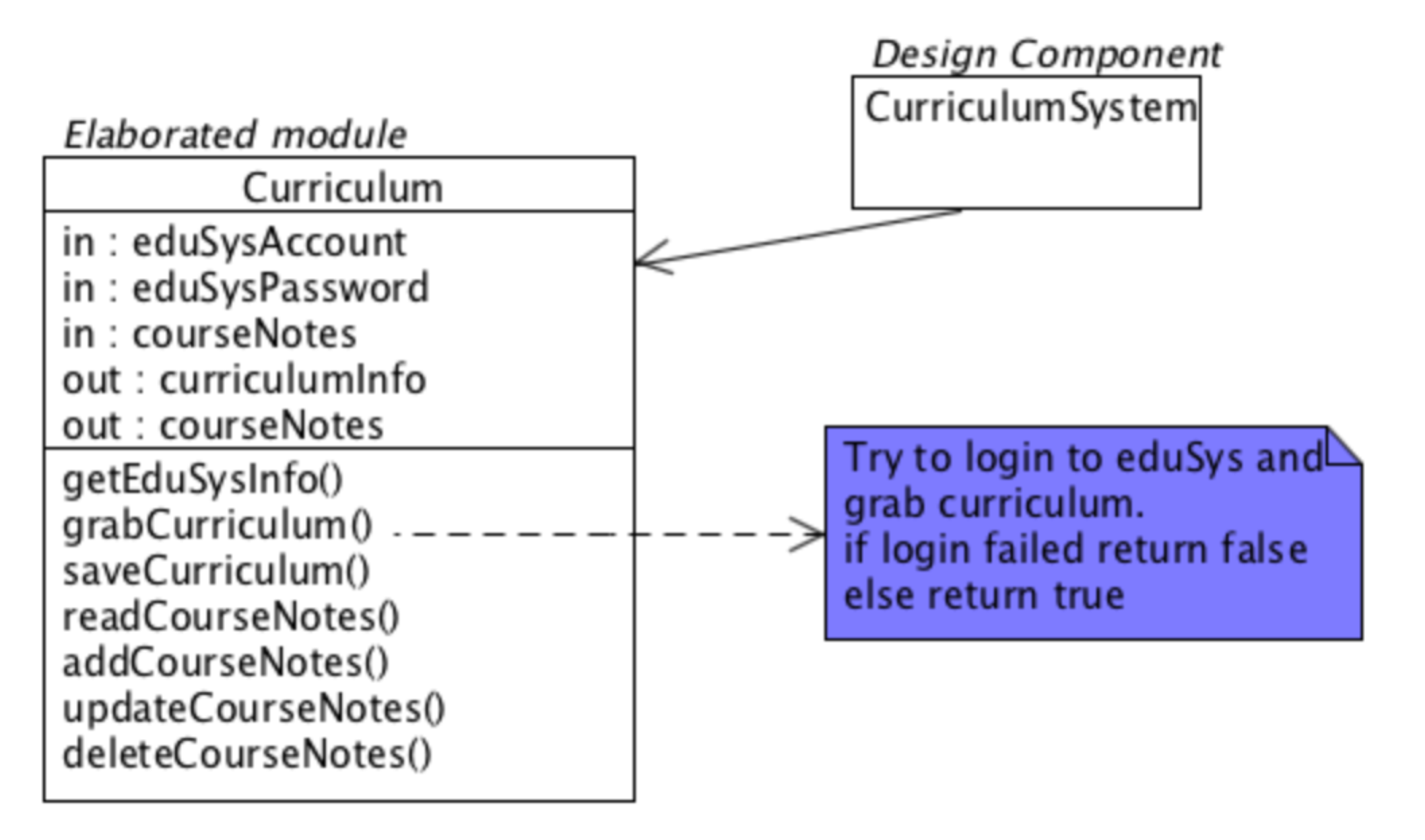
向用户索要教务系统账号及密码，加密传输；登录教务系统抓取课程信息；将课程信息格式化供前端使用；将课程信息存入数据库；从数据库读取笔记信息，供前端使用；接收用户提交的笔记，更新到数据库中。



课表功能整体类图



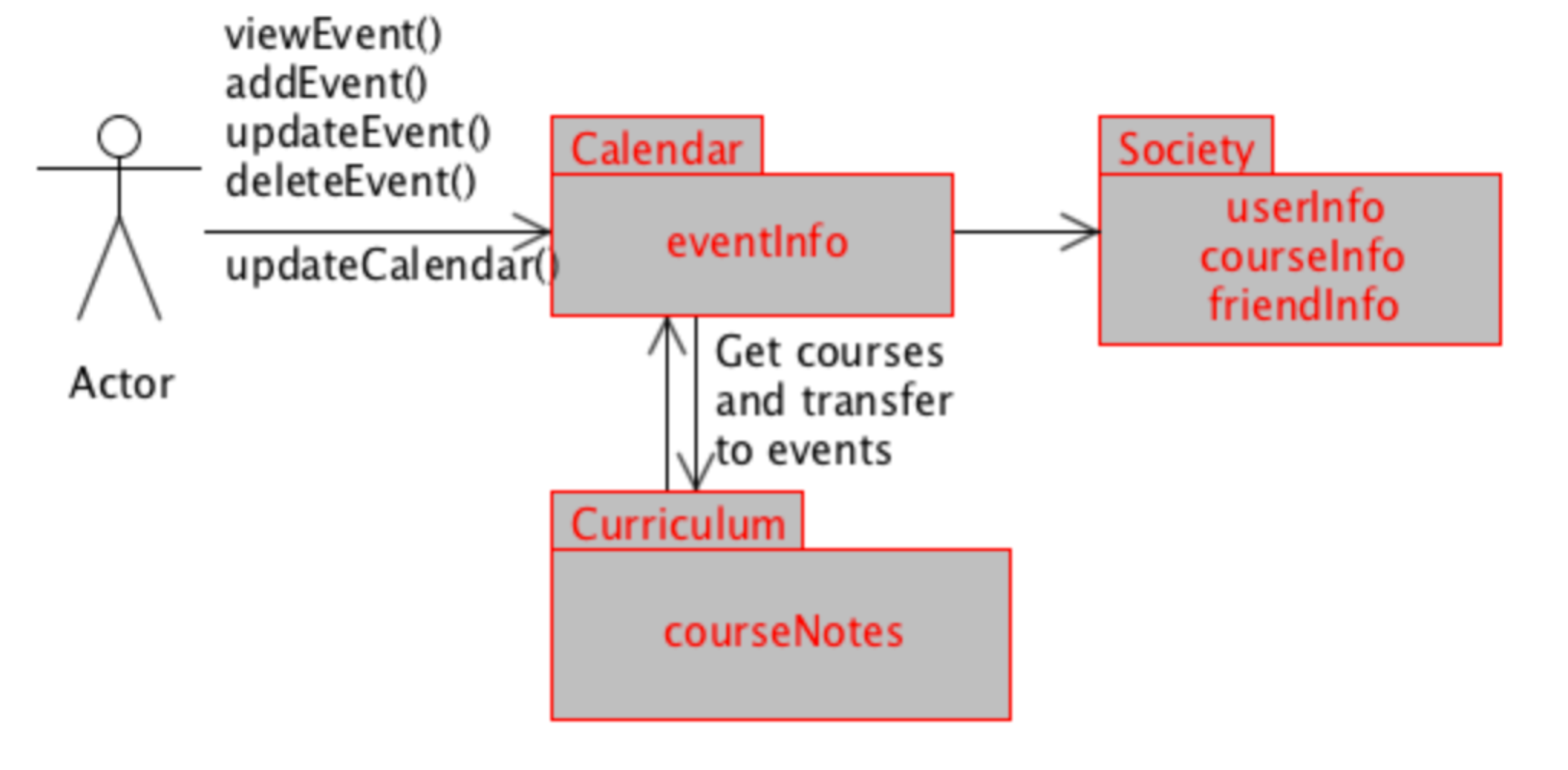
课表功能行为流程图



课表功能组件级设计图

1. 日历功能 (Calendar{Event, Calendar})

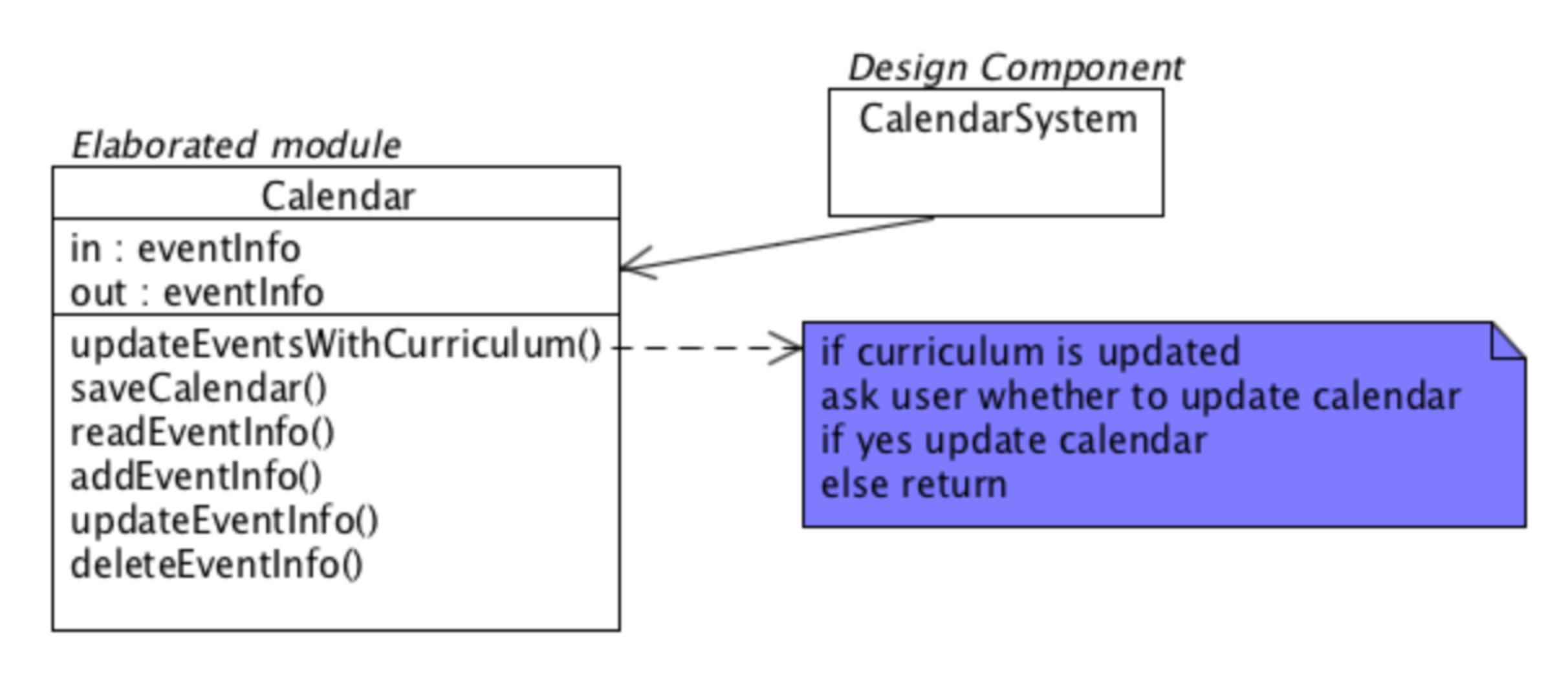
从数据库中读取课程及日历信息，进行整合更新后，供前端使用并存入数据库；从数据库中读取备忘录信息，供前端使用；从前端接受添加、更改、删除备忘录的命令及相关信息，在数据库中进行相应更改。



日历功能整体类图



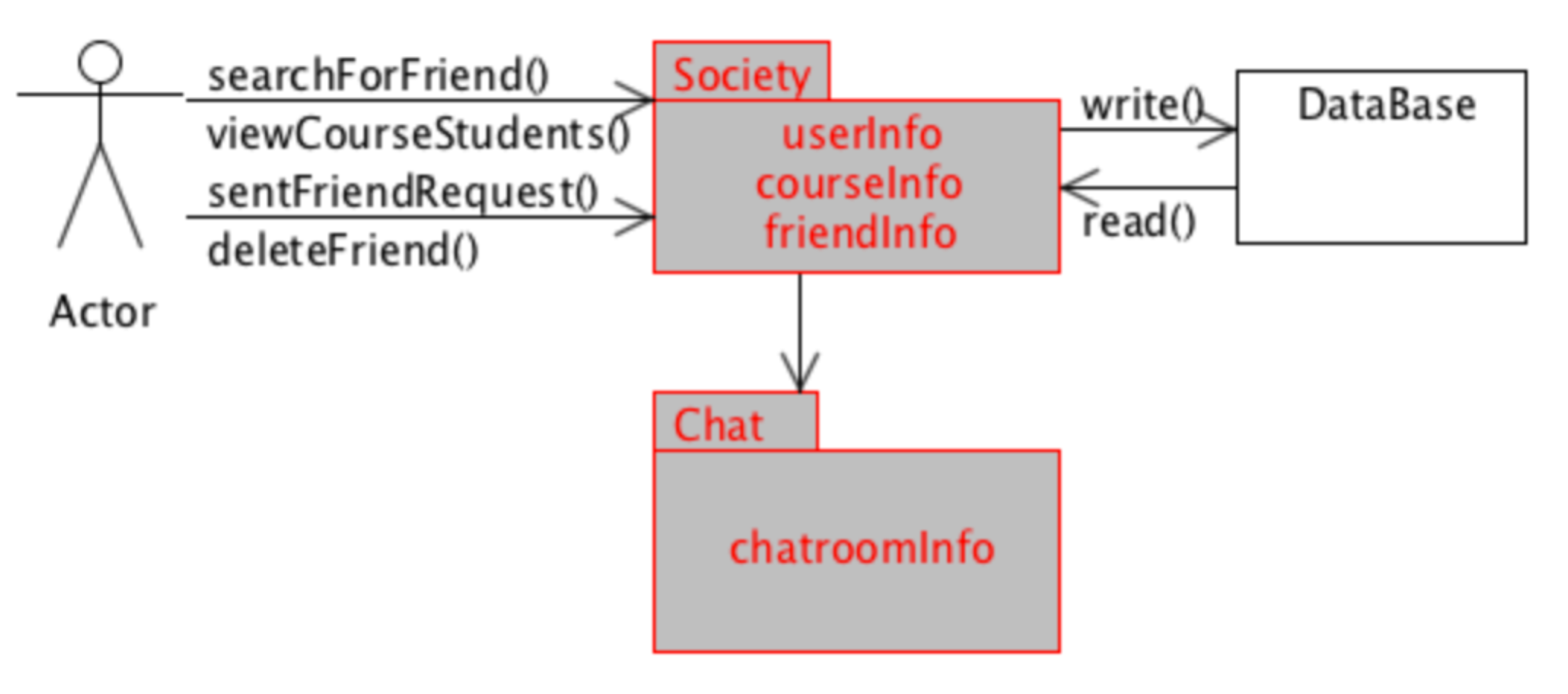
日历功能行为流程图



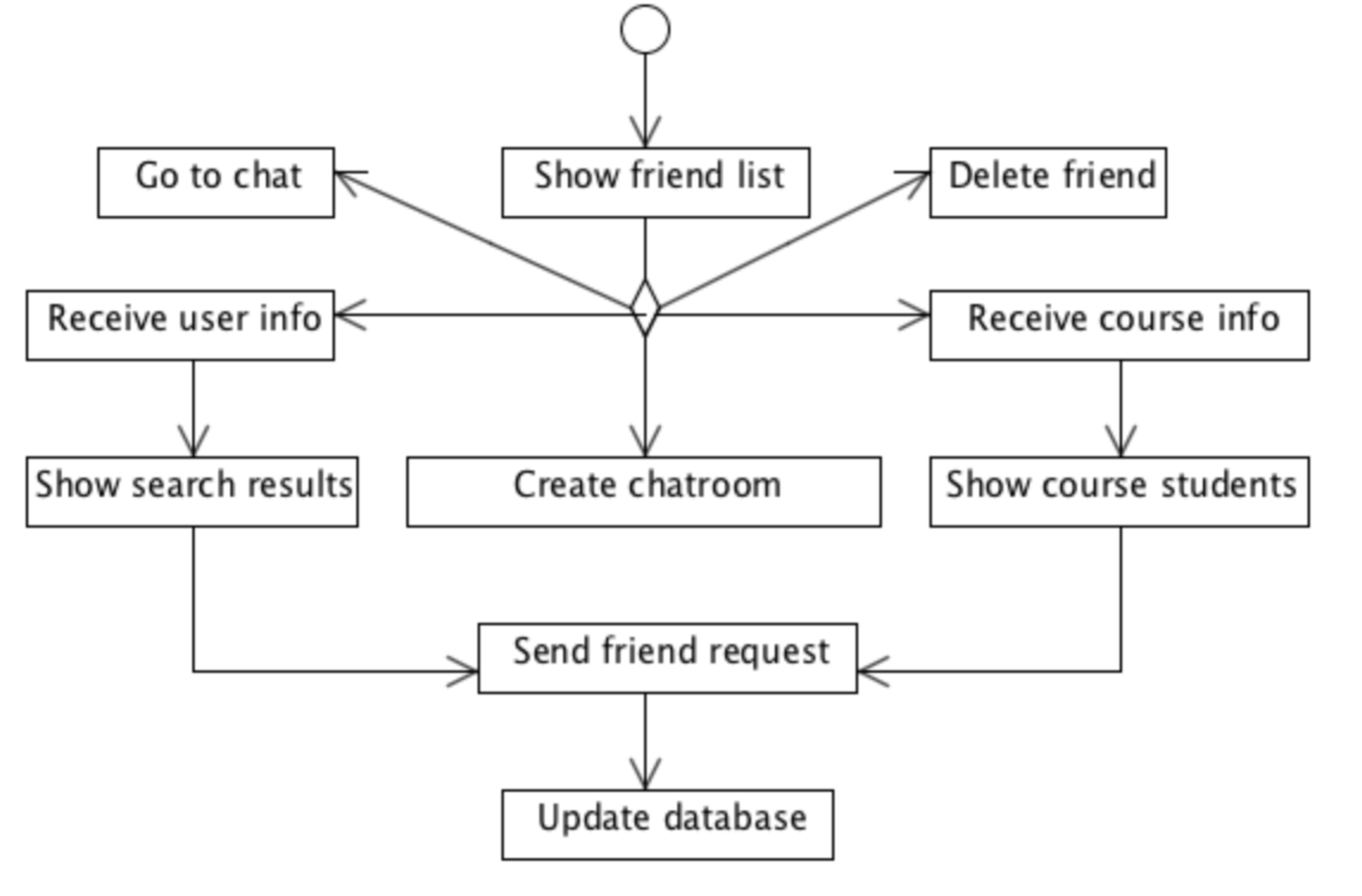
课表功能组件级设计图

1. 课程社区及好友功能 (Society{Student, Friend, Request})

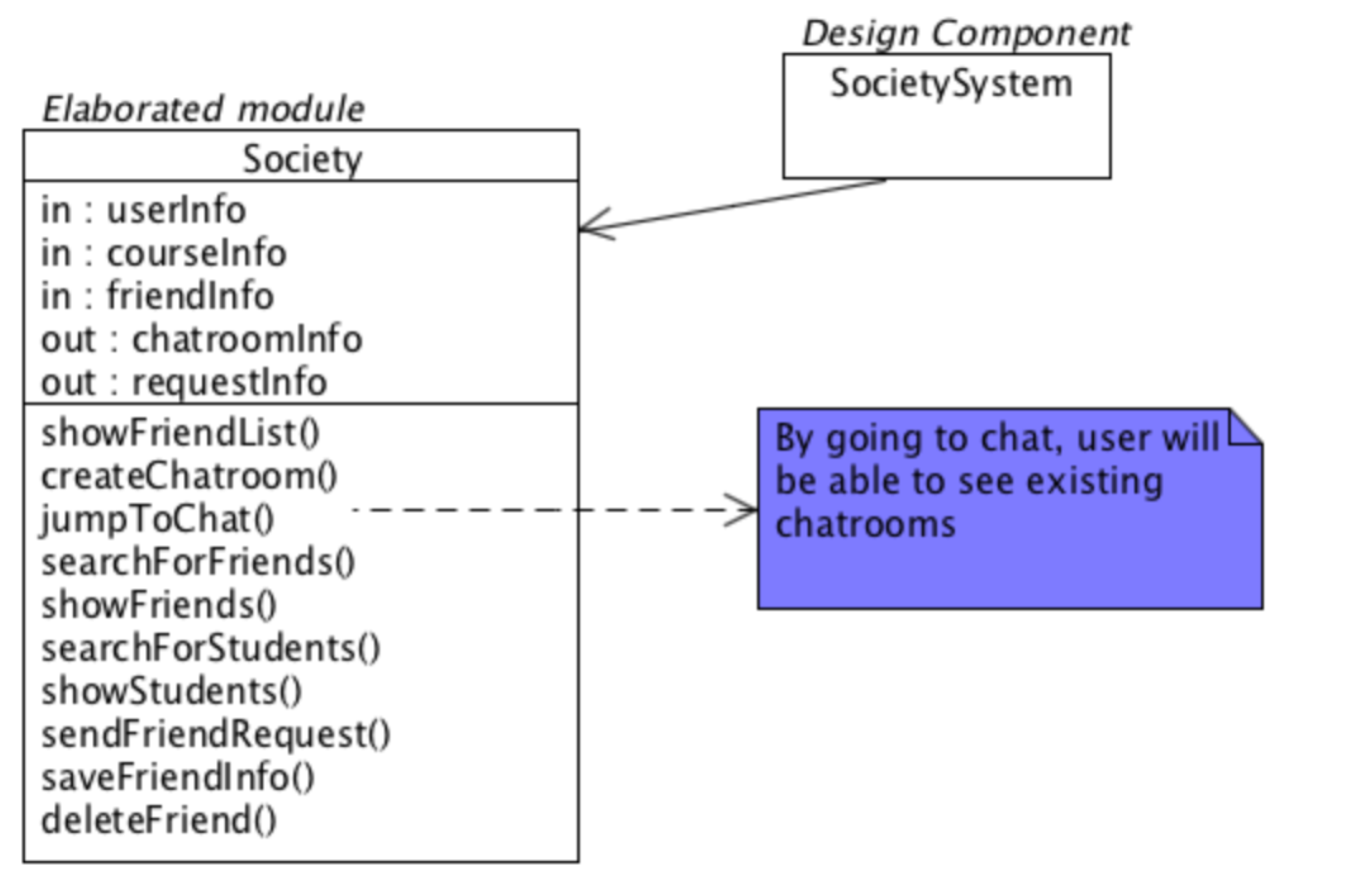
从数据库读取课程社区信息，供前端使用；接受用户加入或退出特定课程社区的要求，并在数据库中进行相应更改；随时可根据用户名\邮箱\关联账户名读取对应用户信息，供前端使用；接受用户添加好友的请求，并将该请求推送给相应用户；接受用户对好友请求的处理，并将结果存入数据库中；根据用户编号从数据库中获取相应用户的信息，供前端使用。



社区功能整体类图



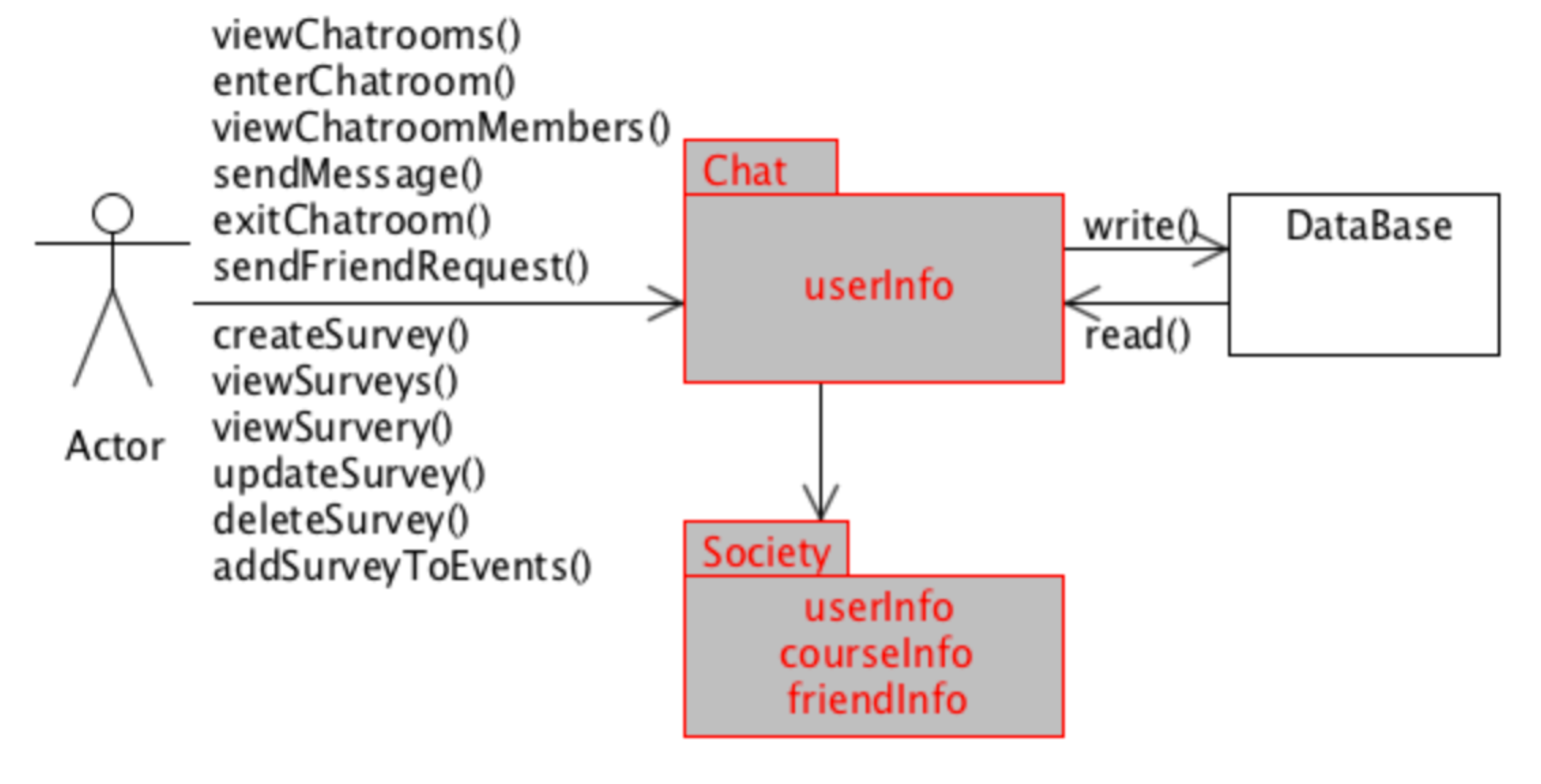
社区功能行为流程图



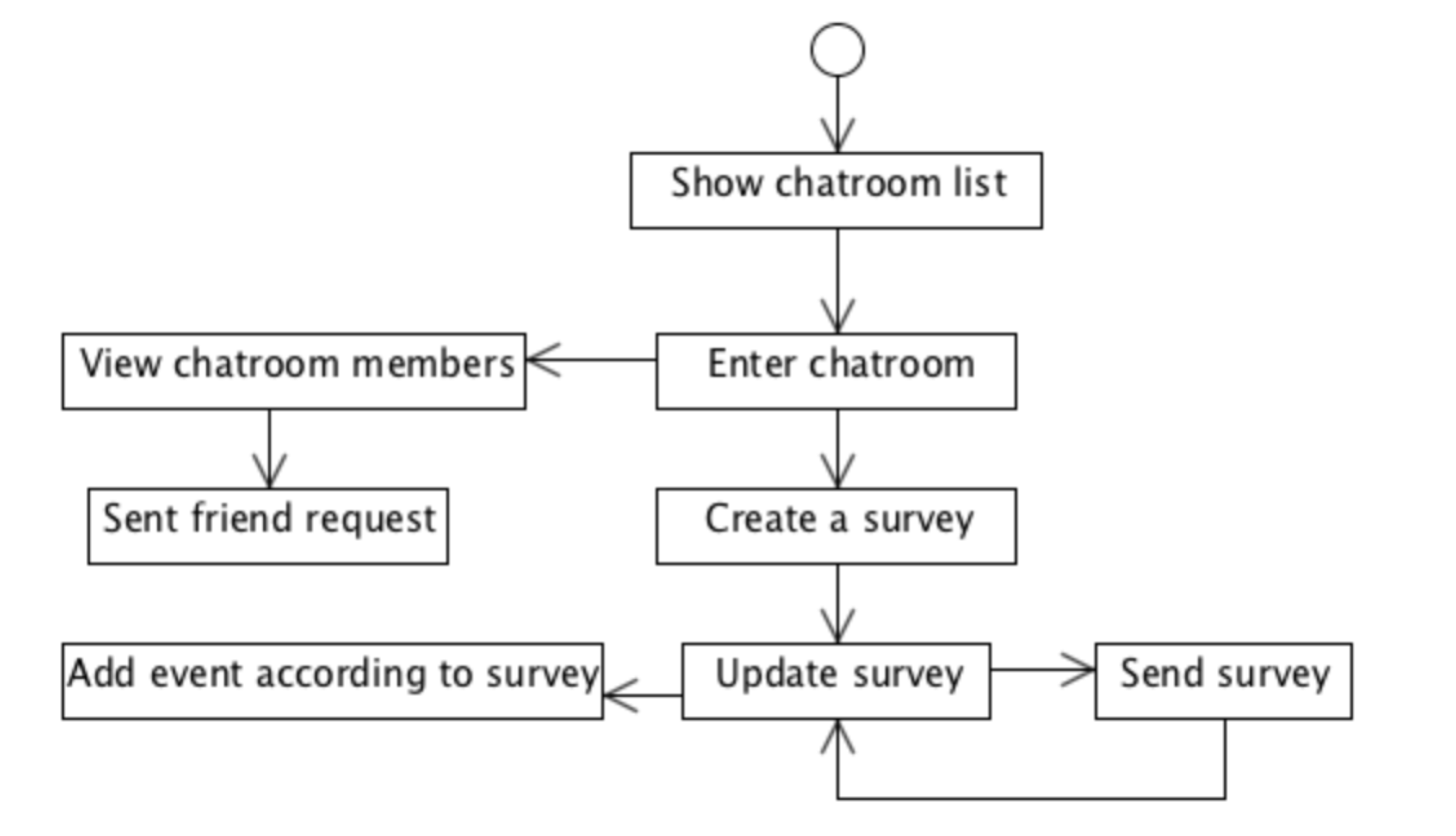
社区功能组件级设计图

1. 基于日历的问卷式邀约+实时聊天室功能 (Chat{Chatroom, Member, Survey})

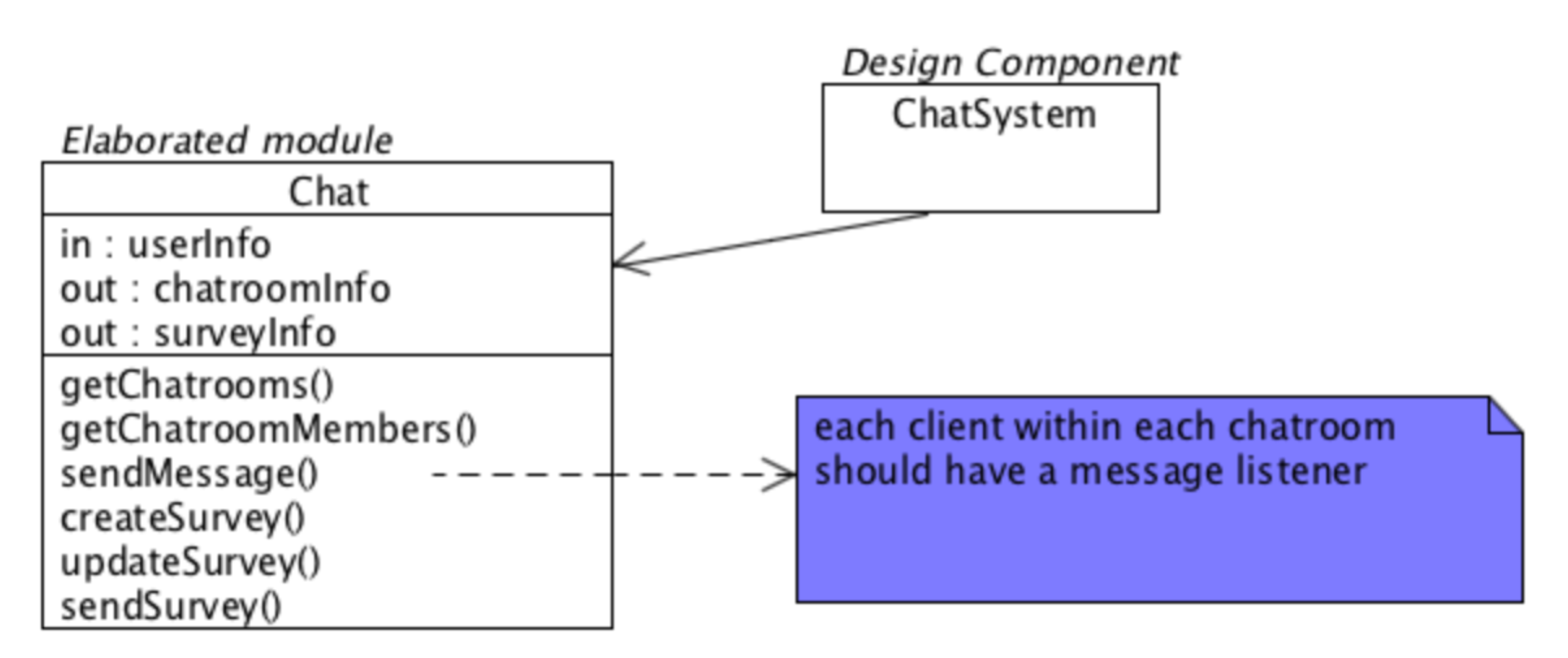
从数据库读取日历及好友信息，供前端使用；接受用户发送的被邀请人名单并建立对应聊天室，将聊天室数据推送给相关用户后存入数据库中；可随时读取、修改聊天室名称；接受聊天室中用户添加新成员及聊天室建立者删除已有成员的请求，将相关数据存入数据库中；接受聊天室中成员发起问卷的请求，读取该用户日历信息，自动生成格式化的空闲时间数据供前端使用；接受用户决定的问卷时间，推送给其他用户后存入数据库中；接受用户对问卷的填写结果，推送给其他用户，并存入数据库中；随时可读取问卷信息，供前端使用；接受问卷发起者结束问卷的指令，将问卷结果推送给所有聊天室中用户，征求用户意见是否将事件加入到日历，并将相关数据写入数据库；接受用户对是否将事件加入到日历的回复，将相关数据写入数据库；随时可读取聊天室成员信息，供前端使用；接受用户添加好友的请求，将请求POST到模块5；实时储存聊天信息；可根据聊天室编号及已获取条数按时间顺序每次获取20条聊天信息。



聊天功能整体类图



聊天功能行为流程图



聊天功能组件级设计图

# 2.数据接口设计（数据类）

## 2.1.URL列表

所有消息传递都用POST

|  |  |  |
| --- | --- | --- |
| 相对路径 | HTTP Variable | 备注 |
| /login | POST | 登录 |
| /logout | DELETE | 登出 |
| /register | POST | 注册 |
| /grab\_school\_info | POST | 请求服务器抓取学校信息 |
| /users/:id/notes | POST | 创建notes |
| /users/:user\_id/notes/:id | GET | 获取notes信息 |
| /users/:user\_id/notes/:id | POST | 更新notes信息 |
| /users/:user\_id/notes/:id | DELETE | 删除notes |
| /users/:user\_id/curriculums | POST | 抓取课程表信息 |
| /users/:user\_id/curriculums/:id | GET | 获取课程表信息 |
| /users/:user\_id/curriculumns/:id | POST | 更新课程表信息 |
| /users/:user\_id/curriculumns/:id | DELETE | 删除课程表信息 |
| /users/:user\_id/calendars | GET | 查看所有日历 |
| /users/:user\_id/calendars | POST | 创建日历表 |
| /users/:user\_id/calendars/:id | GET | 获取日历表 |
| /users/:user\_id/calendars/:id | POST | 更新日历表 |
| /users/:user\_id/calendars/:id | DELETE | 删除日历表 |
| /calendars/:calendar\_id/events | GET | 查看所有events |
| /calendars/:calendar\_id/events | POST | 创建events |
| /calendars/:calendar\_id/events/:id | GET | 查看events |
| /calendars/:calendar\_id/events/:id | POST | 更新events |
| /calendars/:calendar\_id/events/:id | DELETE | 删除events |
| /search\_user?q= | GET | 搜索用户 |
| /users/:user\_id/friends | GET | 获取好友列表 |
| /users/:user\_id/friends/:id | POST | 发送加好友请求 |
| /users/:user\_id/friends/:id | DELETE | 删除好友 |
| /search\_course?q= | GET | 搜索课程 |
| /courses/:id | GET | 获取课程信息 |
| /courses/:id/students?page= | GET | 获取某门课  所有学生 |
| /chatrooms/:id | POST | 创建聊天室 |
| /chatrooms/:id | GET | 获取聊天室  基本信息 |
| /chatrooms/:id/enter | POST | 进入聊天室 |
| /chatrooms/:id/exit | POST | 进入聊天室 |
| /chatrooms/:id/messages | GET | 异步获取  聊天信息 |
| /chatrooms/:id/messages | POST | 发送聊天信息 |
| /chatrooms/:id/members | GET | 获取聊天室  成员信息 |
| /surveys?page= | GET | 查看所有Survey |
| /surveys | POST | 创建Survey |
| /surveys/:id | GET | 获取Survey |
| /surveys/:id | POST | 更新Survey |
| /surveys/:id | DELETE | 删除Survey |
| /surveys/:survey\_id/add\_to\_events/:id | POST | 添加Survey  到事件 |

## 2.2.CRUD接口

下面给出一个CRUD接口范例（日历表）

**2.2.1.查看所有日历 POST /users/:user\_id/calendars**

请求数据格式：无

返回数据格式：

HTML：对应HTML视图

JSON：{

calendars: [CalendarArray {

id: integer,

name: string

}]

}

**2.2.2.创建日历表 POST /users/:user\_id/calendars**

请求数据格式：

{

name: string

}

返回数据格式：

HTML: redirect\_to “/users/:user\_id/calendars/:id”

JSON: {

status: string

}

**2.2.3获取日历表 GET /users/:user\_id/calendars/:id**

请求数据格式：无

返回数据格式：

HTML：对应的HTML视图

JSON: {

id: integer,

user\_id: integer,

calendar\_name: string

event\_ids: [integer array]

}

**2.2.4更新日历表 POST /users/:user\_id/calendars/:id**

请求数据格式：

{

id: integer,

user\_id: integer,

calendar\_name: string

}

返回数据格式：

HTML: redirect\_to “/users/:user\_id/calendars/:id”

JSON: {

status: “success” / [failure message]

}

**2.2.5删除日历表 DELETE /users/:user\_id/calendars/:id**

请求数据格式：无

返回数据格式：

HTML: redirect\_to “/users/:user\_id/calendars”

JSON: {

status: “success” / [failure message]

}

**以下对于某一实体的创建、查看、更新、删除(CRUD)接口并不具体说明，具体参数参见数据模型。其中，Curriculumns, Calendars, Notes使用了Nested Model。关于这方面的常见接口，参见Rails文档**[**http://guides.rubyonrails.org/index.html**](http://guides.rubyonrails.org/index.html)**。**

## 2.3.非CRUD接口

1. **用户验证**

采用public+private key，在注册的时候给定，private key存在服务器，客户端需要上传一个ID（待定），然后得到public key，之后用它加密

1. **聊天室信息实时获取**

**用socket方式实现**

1. **推送消息**

参考APNS苹果消息同步方式；如果用户处于登录状态，直接由socket推送，如果未登录，暂定为加上设备ID后push给苹果服务器；此功能还需讨论！

**2.3.1.POST /login**

请求参数格式：

{

user\_name: string,

password: string,

password\_confirmation: string

}

返回数据格式

Html: redirect\_to “/users/:id/calendars”

JSON: {

status: “success” / [failure message]

}

**2.3.2.DELETE /logout/:id**

返回数据格式

Html: redirect\_to “/”

JSON: {

Status: “success” / [failure message]

}

**2.3.3.POST /register**

请求参数格式：

{

user\_accounttype: “user\_name” or “email” or “accountnum”,

user\_name: string,

password: string,

password\_confirmation

}

返回数据格式

Html: redirect\_to “/users/:user\_id/curriculums”

JSON: {

status: “success” / [failure message]

}

**2.3.4.POST /grab\_school\_info**

此接口的意义在于强制服务器去抓取并更新学校信息。

请求参数格式：

{

school\_id: integer

}

返回数据格式：

JSON: {

status: “success” / [failure message]

}

**2.3.5.GET /search\_user?q=**

此接口用于搜索用户，q=后跟查询字符串

返回数据格式：

JSON: {

user\_ids: [int array]

}

**2.3.6.GET /search\_course?q=**

类似的，此接口用于搜索课程，q=后跟查询字符串

返回数据格式：

JSON: {

couse\_ids: [int array]

}

**2.3.7.POST /surveys/:survey\_id/add\_to\_events/:id**

返回数据格式：

JSON: {

status: “success” / [failure message]

}

# 所需数据表项

## 3．1 基础类

## 1．用户信息类 - table User

(user\_name varchar(20),

user\_id int,

user\_password varchar(20),

user\_accounttype varchar(10),

user\_accountnum varchar(20),

user\_email varchar(20),

user\_university varchar(20),

user\_college varchar(20),

user\_major varchar(20),

user\_grade varchar(20),

user\_gender int,

primary key (user\_id))

## 2．学校类 - table University

(university\_id int,

university\_name varchar(20),

primary key (university\_id))

## 3．学院类 - table College

(college\_id int,

college\_name varchar(20),

primary key (college\_id)

foreign key (university\_id) references university (university\_id))

## 4．专业类 - table Major

(major\_id int,

major\_name varchar(20),

college\_id int,

primary key (major\_id),

foreign key (college\_id) references college(college\_id))

## 5．年级类 - table Grade

(grade\_id int,

grade\_name varchar(20),

primary key (grade\_id))

## 6．课程类- table Course

(course\_id int,

course\_num varchar(10),

course\_name varchar(10),

course\_weekday varchar(10),

course\_daytime varchar(10),

course\_credit numeric(12,2),

course\_college int,

course\_professor int,

course\_description varchar(100),

course\_rate numeric(12,2),

course\_ratecount int,

primary key (course\_id),

foreign key (course\_college) references college (course\_id),

foreign key (course\_professor) references professor (professor\_id))

## 7．教授类 - table Professor

(professor\_id int,

professor\_name varchar(20),

professor\_college int.

professor\_status varchar(10),

professor\_description varchar(100),

professor\_rate numeric(12,2),

professor\_ratecount int,

primary key (professor\_id),

foreign key (professor\_college) references college (college\_id))

## 8．事件类 - table Event

(event\_id int,

event\_name varchar(20),

event\_startdatetime datetime,

event\_enddatetime datetime,

primary key (event\_id))

## 9．学生类 - table Student

(student\_id int,

student\_course int,

student\_user int,

student\_status int,

primary key (student\_id),

foreign key (student\_course) references course (course\_id),

foreign key (student\_user) references user (user\_id))

## 10．聊天室类 - table Chatroom

(chatroom\_id int,

chatroom\_name varchar(20),

chatroom\_membercount int,

chatroom\_requestcount int,

primary key (chatroom\_id))

## 15．调查类 - table Survey

(survey\_id int,

survey\_chatroom int,

survey\_user int,

survey\_timespancount int,

survey\_status int,

survey\_startdatetime datetime,

survey\_enddatetime datetime,

primary key (survey\_id),

foreign key (survey\_user) references user (user\_id))

## 16．时间段类 - table Timespan

(timespan\_id int,

timespan\_user int,

timespan\_survey int,

timespan\_status int,

timespan\_startdatetime datetime,

timespan\_enddatetime datetime,

primary key (timespan\_id),

foreign key (timespan\_user) references user (user\_id),

foreign key (timespan\_survey) references survey (survey\_id))

## 17．聊天记录类 - table Message

(message\_id int,

message\_user int,

message\_chatroom int,

message\_num int,

message\_datetime datetime

primary key (message\_id),

foreign key (message\_user) references user (user\_id),

foreign key (message\_chatroom) references chatroom (chatroom\_id))

## 3．2 关系类

## 1．table Curriculum

(curriculum\_id int,

curriculum\_user int,

curriculum\_course int,

curriculum\_notes varchar(100),

primary key (curriculum\_id)

foreign key (curriculum\_user) references user (user\_id),

foreign key (curriculum\_course) references course (course\_id))

## 2．table Calendar

(calendar\_id int,

calendar\_user int,

calendar\_event int,

primary key (calendar\_id),

foreign key (calendar\_user) references user (user\_id),

foreign key (calendar\_event) references event (event\_id))

## 3．table Friend

(friend\_id int,

friend\_user1 int,

friend\_user2 int,

primary key (friend\_id),

foreign key (friend\_user1) references user (user\_id),

foreign key (friend\_user2) references user (user\_id))

## 4．table Request

(request\_id int,

request\_user1 int,

request \_user2 int,

primary key (friend\_id),

foreign key (request \_user1) references user (user\_id),

foreign key (request \_user2) references user (user\_id))

## 5．table Member

(member\_id int,

member\_user int,

member\_chatroom int,

primary key (member\_id),

foreign key (member\_user) references user (user\_id),

foreign key (member\_chatroom) references chatroom (chatroom\_id))