高级软件工程课程大作业

Project 1 课程管理

学生选课系统作为一种现代化的教学技术，越来越受到人们的重视。选课系统需要满足三类用户角色的需求，即学生用户、教师用户和管理员用户。学生对选课系统的使用大多数在于选课及查询，教师更多的注重成绩录入以及查询学生选课的信息，管理员更多的是安排课程和发布课程公告等。本次项目重点放在了学生角色上面，并从学生用户需求出发给出了几个需要实现的功能点，学有余力的同学可以在完成下面的功能后继续完善其他的功能。简易的原型系统参考 https://github.com/PENGZhaoqing/CourseSelect。

1.1 课程查询

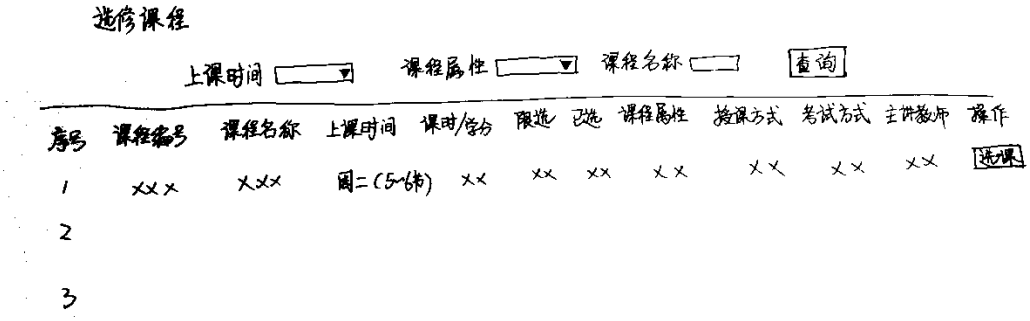
下载完以后将gemfile中的source 改为https://rubygems.org，然后运行bundle install 安装缺失的gem包

1.1.1 用户需求

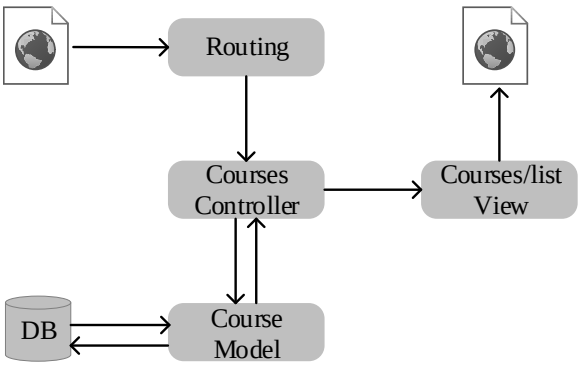
学生登录系统后，可以在“选修课程”界面键入或选择过滤条件，查询符合键入条件的课程，方便学生从众多的课程中筛选出想要选修或者是查看详细信息的课程。

1.1.2 用户界面

学生甲登录系统后，进入“选修课程”界面，选择上课时间、课程属性或者在课程名称搜索框中键入课程名称，点击查询按钮，系统返回符合查询条件的课程。



1.1.3 MVC 设计



1. 在 models/course.rb 中加入缺少的属性，如上课时间等;
2. 在视图 course/list.html.erb 中添加下拉框、搜索框、查询按钮等控件，并在用户点击查询按钮后获取相关参数传给 courses_controller.rb;
3. 在 courses_controller.rb 增加查询方法，根据查询请求携带的参数（上课时间、课程名称、课程属性等）在数据库中查询符合条件的课程并返还给视图 course/list.html.erb;
4. 视图 course/list.html.erb 接收查询结果并显示。

1.1.4 测试用例

1. 学生甲在上课时间下拉框中选择“周五”，点击查询按钮，页面显示所有上课时间为周五的课程及课程信息;
2. 学生甲在课程属性下拉框中选择“专业核心课”，点击查询按钮，页面显示所有课程属性为“专业核心课”的课程及课程信息;
3. 学生甲在课程名称搜索框中键入“软件”，点击查询按钮，页面显示所有名称中含“软件”的课程及课程信息;
4. 学生甲在上课时间下拉框中选择“周五”，在课程属性下拉框中选择“专业核心课”，点击查询按钮，页面显示所有上课时间为周五且课程属性为“专业核心课”的课程及课程信息;
5. 学生甲在上课时间下拉框中选择“周五”，在课程名称搜索框中键入“软件”，点击查询按钮，页面显示所有上课时间为周五且名称中含“软件”的课程及课程信息;
6. 学生甲在课程属性下拉框中选择“专业核心课”，在课程名称搜索框中键入“软件”，点击查询按钮，页面显示所有课程属性为“专业核心课”且名称中含“软件”的课程及课程信息;
7. 学生甲在上课时间下拉框中选择“周五”，在课程属性下拉框中选择“专业核心课”，在课程名称搜索框中键入“软件”，点击查询按钮，页面显示所有上课时间为周五，课程属性为“专业核心课”且名称中含“软件”的课程及课程信息。

1.2 学分统计

1.2.1 用户需求

学生登录系统后，可以查看自己已选课程中“公共选修课”、“专业学位课”以及“总学分”的统计信息，了解已获取学分和规定修习学分之间的差距。

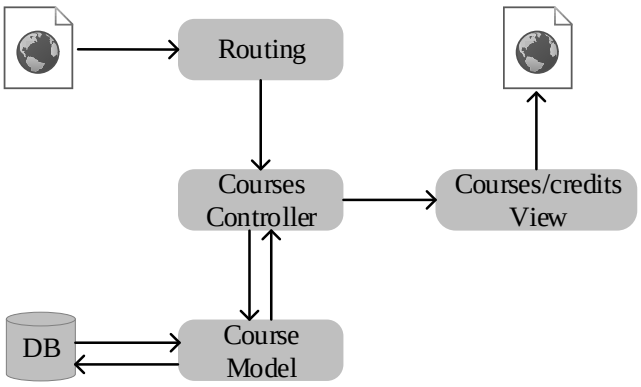
1.2.2 用户界面

学生甲登录系统后，进入“学分提示”界面，能够看到已选课程中每类课程的学分总计，“公共选修课”、“专业学位课”以及“总学分”的统计信息。

选课提示

类别	公共必修课程及学分	公共选修课程及学分	专业学位课学分要求	总学分要求
学习要求	人文系列讲座(1学分) XXX XXX	>= 2学分	3 >= 12学分	>=30学分
选课情况	XXX XXX	5学分	13学分	30学分
获取学分	XXX XXX	4学分	13学分	27学分

1.2.3 MVC 设计



1. 在 shared/_sidenav.html.erb 视图中增加“学分提示”列表项；
2. 在 courses_controller.rb 增加学分统计方法；
3. 增加“学分提示”视图，显示“公共选修课”、“专业学位课”以及“总学分”的已选学分统计信息。

1.2.4 测试用例

1. 学生甲进入“学分提示”界面，能够看到自己已选课程中课程的分类学分统计；
2. 学生甲进入“选修课程”界面，选择若干门课程加入课表，然后进入“学分提示”界面，能够看到更新后的已选课程中课程的分类学分统计。

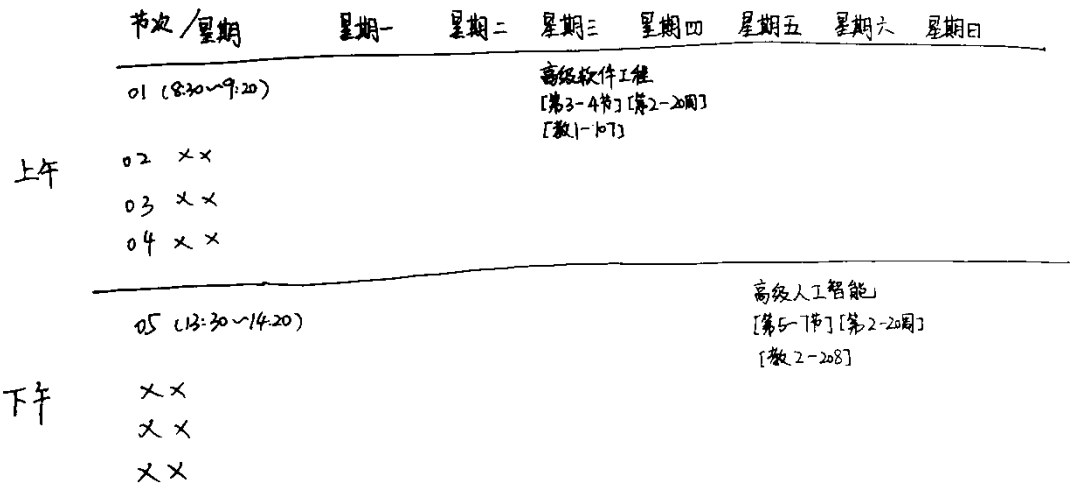
1.3 课表查看

1.3.1 用户需求

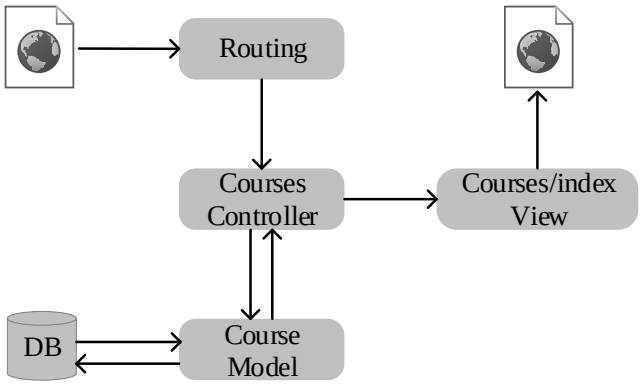
学生选完课之后，可以查看一周的课表，更加清晰地知道课程的时间排布。

1.3.2 用户界面

学生甲登录系统后，进入“已选课程”界面，点击“查看课表”按钮，能够看到课表信息。



1.3.3 MVC 设计



1. 在 courses/index.html.erb 视图中增加“查看课表”按钮；
2. 在 courses_controller.rb 增加生成课表的方法；
3. 增加“课表”视图，显示课表信息。

1.3.4 测试用例

1. 学生甲进入“已选课程”界面，点击“查看课表”按钮，能够看到当前课表信息；
2. 学生甲进入“选修课程”界面，选择若干门课程加入课表，然后进入“已选课程”界面，点击“查看课表”按钮，能够看到更新后的课表信息。

1.4 选课冲突处理

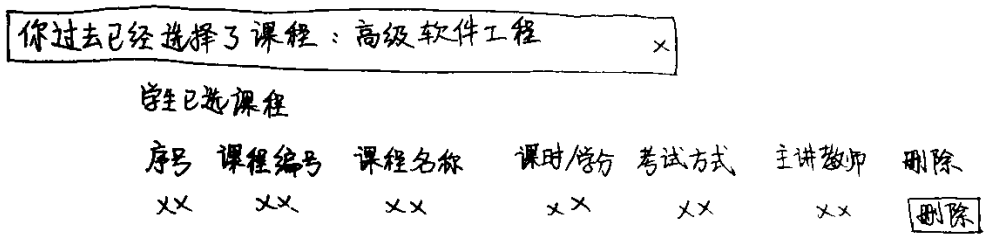
1.4.1 用户需求

学生在选修某门课程时，如果要选修的课程已经在课表中，或与已经选修的课程发生时间冲突，或者已经超过限选人数，则系统需要给出提示。

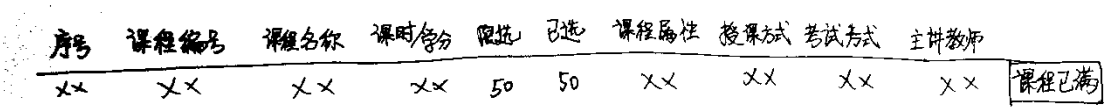
1.4.2 用户界面

学生甲登录系统后，进入“选修课程”界面，点击课程 A 的“加入课程”按钮，若甲已经选过 A 课程，则系统给出“请不要重复选课”的提示；若课程 A 的选课人数超过限选人数，则系统给出“课程 A 选课人数已满”的提示；若课程 A 的上课时间和甲已经选择的课程 B 上课时间冲突，则系统给出“课程 A 和课程 B 的上课时间冲突”的提示。

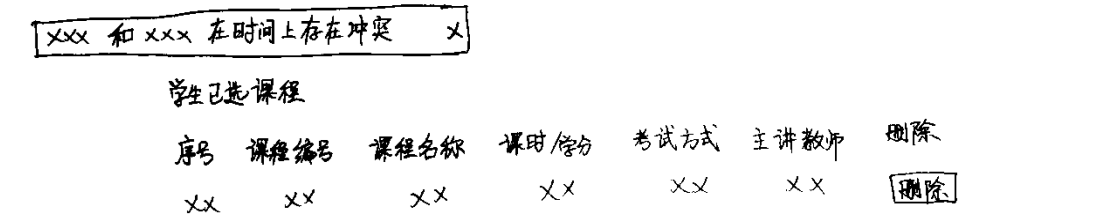
重复选择:



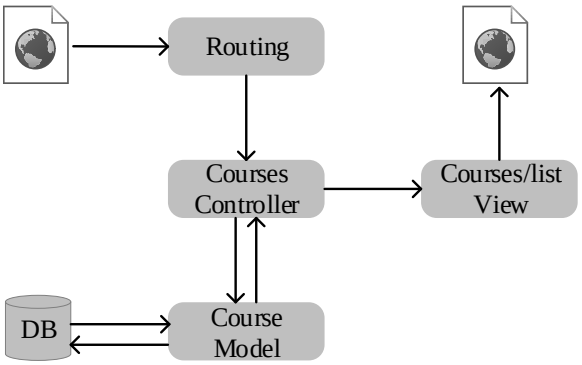
选课人数已满:



时间冲突:



1.4.3 MVC 设计



1. 在 models/course.rb 中添加相关属性如上课时间、限选人数等;
2. 在 courses_controller.rb 的 select 方法中，判断是否存在上述三种冲突情形，若存在，给出相关提示（可使用 flash[:danger]，flash[:notice]等）

1.4.4 测试用例

1. 学生甲（已经选择课程 A，课程 A 上课时间为周五）点击课程 A 的“加入课程”按钮，页面弹出“请不要重复选课”的提示；
2. 学生甲（已经选择课程 A，课程 A 上课时间为周五）点击课程 B（限选人数为 5，已选 5 人）的“加入课程”按钮，页面弹出“课程 B 选课人数已满”；
3. 学生甲（已经选择课程 A，课程 A 上课时间为周五）点击课程 C（上课时间为周五）的“加入课程”按钮，页面弹出“课程 A 和课程 C 的上课时间冲突”的提示。

1.5 其他功能

新增其他感兴趣的功能，给出用户需求、用户界面、MVC 设计和测试用例。

Project 2 聊天机器人

随着互联网的迅速普及，以及网络技术的不断发展，人们通过网络的交流方式不断发展。网络聊天室就是其中一种。聊天室的系统即时交流满足了网络中多人同时交流的需要。人工智能的飞速发展使得聊天机器人的智能程度越来越高，不仅可以智能对话，还可以查询公交线路，翻译句子，查询天气等。本项目着重实现给出两个基本聊天室的功能和接入聊天机器人的相关功能。简易的原型系统参考 https://github.com/PENGZhaoqing/RailsChat。

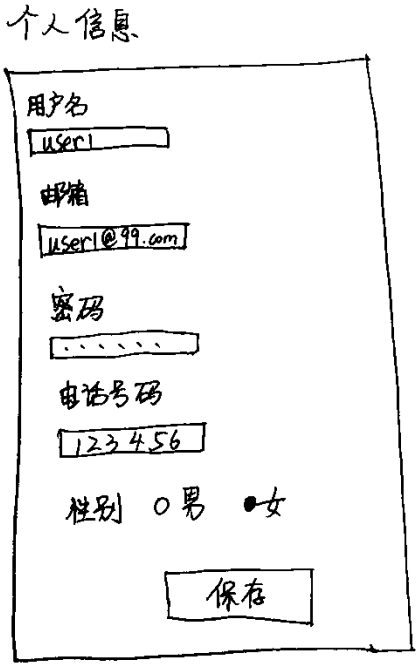
2.1 用户信息查看和修改

2.1.1 用户需求

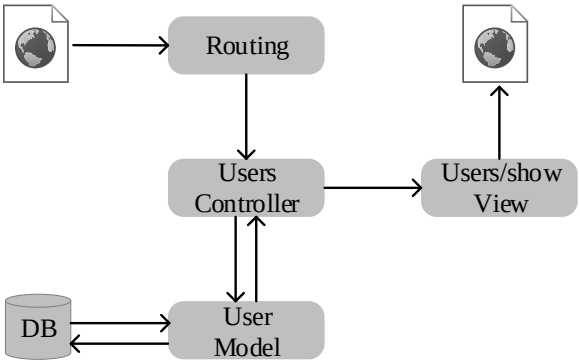
用户登录系统后，可以查看个人的账户信息，在账户信息有变动时，可以修改信息并保存更新后的信息。

2.1.2 用户界面

用户甲登录聊天室，点击右上角的“个人账户”按钮，可以看到和修改用户名、邮箱、电话、性别等个人信息，点击”保存“按钮，可以保存更新后的个人信息。



2.1.3 MVC 设计



1. 在 users_controller.rb 中补充 show 和 update 方法；
2. 增加 users/show.html.erb 视图，显示用户信息。

2.1.4 测试用例：

1. 用户甲登录聊天室，点击右上角的“个人账户”按钮，可以看到用户名、邮箱、电话、性别等个人信息；
2. 用户甲登录聊天室，点击右上角的“个人账户”按钮，修改用户名、邮箱、电话号码、性别，点击”保存“按钮，页面提示保存成功。

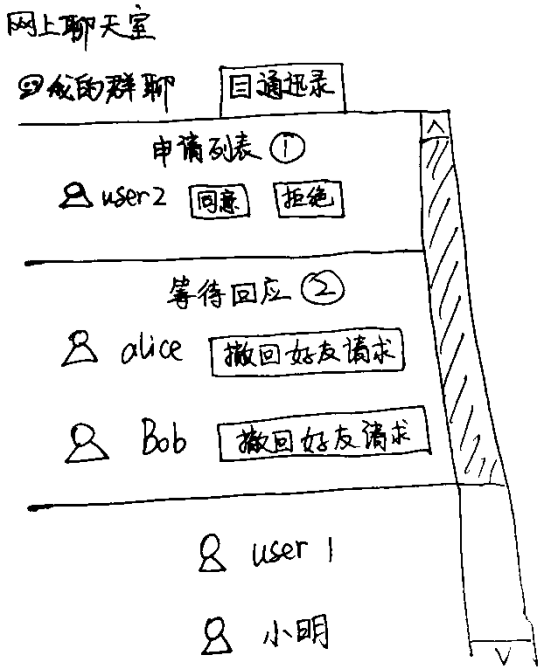
2.2 好友申请

2.2.1 用户需求

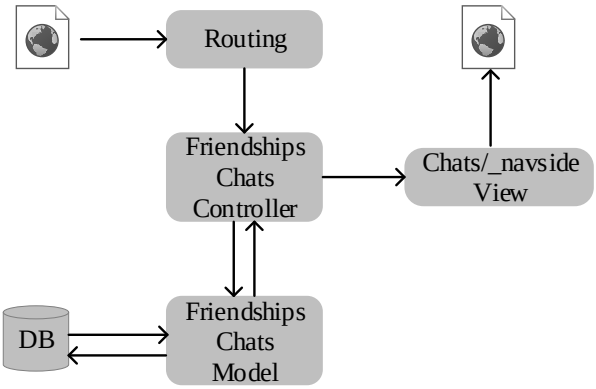
用户可以添加新的好友，搜索好友的账户名并发出好友申请，当好友登录账户后，看到好友申请后可以选择同意或拒绝。

2.2.2 用户界面

用户甲登录聊天室，进入通讯录，点击“添加好友“按钮，出现”查找用户“搜索框，键入查找好友的名称乙，点击查找，选择要添加的好友乙，点击”添加“按钮，则乙用户收到甲的添加好友请求，并可以在通讯录的申请列表中看到好友请求，乙可以通过点击”同意“按钮来同意添加甲为好友，或”拒绝“按钮来拒绝对方的请求。



2.2.3 MVC 设计



1. 在 friendships_controller.rb 中修改 create 和 destroy 方法，分别对应“同意“和”拒绝“两个按钮，通过判断两个用户的朋友集合中是否包含对方来确定是否建立朋友关系和销毁朋友关系；
2. 在 chats_controller.rb 中修改 show 和 index 方法，对每个 user 维护两个集合，一个是向自己发过好友请求的 user 集合，一个是自己发送过好友请求的 user 集合；
3. 在视图 chats/_navside.html.erb 中添加“好友申请“和”等待回应“两个列表，用以显示好友申请和自己发出的好友申请，每个”好友申请“有”同意“和”拒绝“两个操作按钮，每个”等待回应“的申请有一个”撤回“操作按钮。

2.2.4 测试用例

1. 用户甲登录聊天室，申请添加用户乙为好友，则用户乙出现在甲通讯录的“等待回应“列表中；用户乙登录聊天室，进入通讯录，看到申请列表中有甲，点击”同意“按钮，用户甲成为乙的好友，出现在乙的通讯录中，同时用户乙成为甲的好友，出现在用户甲的通讯录中。
2. 用户甲登录聊天室，申请添加用户丙为好友，则用户丙出现在甲通讯录的“等待回应“列表中；用户丙登录聊天室，进入通讯录，看到申请列表中有甲，点击”拒绝“按钮，用户甲从乙的申请列表中消失，同时用户甲收到用户丙拒绝好友请求的消息。

2.3 聊天机器人

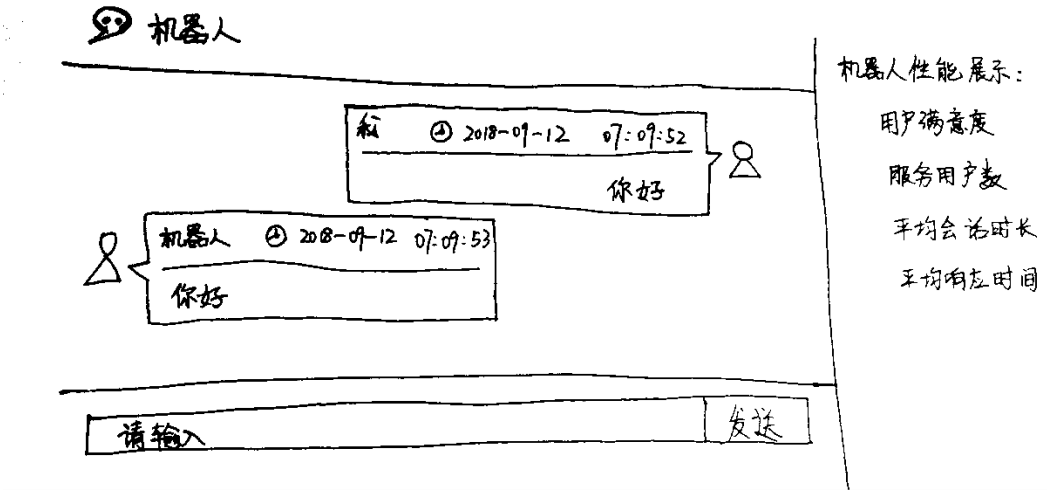
2.3.1 用户需求

使用现有的聊天机器人接口或者自己开发的聊天机器人(选择后者时可以考核时会降低互动能力的要求)，实现用户与聊天机器人的聊天互动，对互动的能力进行一个评估进行在线展示。

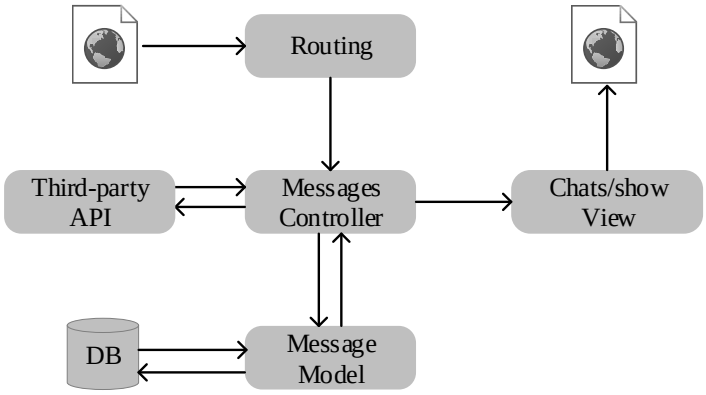
互动能力的评估可以使用用户满意度(可以在每次会话后让用户给出一个满意度分数)、服务用户数（与机器人有过至少一次会话的用户数目)、平均会话时长（用户与机器人的每次会话持续时长的均值)、平均响应时间（机器人应答用户消息的平均耗费时长）等。这些评估指标需要通过合适的表现形式在系统页面展现。

2.3.2 用户界面

用户甲登录聊天室，从通讯录中找到机器人好友，打开与它的聊天界面，甲可以发消息给机器人，机器人会对用户甲的聊天消息进行回复。用户可以在旁边侧栏查看机器人的互动性能。



2.3.3 MVC 设计



聊天机器人可以使用第三方 API 接口。Messages Controller 将用户消息传给第三方 API，第三方 API 返回 JSON 数据，Controller 解析 JSON 数据传给 Chats/show 视图。

图灵机器人是网上的一个第三方平台，提供了自动解析文字的功能，任何所有人、应用，包括微信、微博、人人等都可以方便的访问这个 API 接口。在 http://www.turingapi.com 上注册成为用户，每一个用户可以得到一个 key 值，作为访问 API 标识，注册的帐号申请免费版机器人每天可以给图灵机器人发送对话 1000 次，即这个 key 可以调用 1000 次接口。通过发送 POST 请求，来传输数据，请求参数格式如下所示：

```
{
	"reqType":0,
	"perception": {
		"inputText": {
			"text": "附近的酒店"
		},
		"inputImage": {
			"url": "imageUrl"
		},
		"selfInfo": {
			"location": {
				"city": "北京",
				"province": "北京",
				"street": "信息路"
			}
		}
	},
	"userInfo": {
		"apiKey": "",
		"userId": ""
	}
}
```

输出参数格式如下：

```
{
    "intent": {
        "code": 10005,
        "intentName": "",
        "actionName": "",
        "parameters": {
            "nearby_place": "酒店"
        }
    },
    "results": [
        {
            "groupType": 1,
            "resultType": "url",
            "values": {
                "url": "http://m.elong.com/hotel/0101/nlist/#indate=2016-12-10&outdate=2016-12-11&keywords=%E4%BF%A1%E6%81%AF%E8%B7%AF"
            }
        },
        {
            "groupType": 1,
            "resultType": "text",
            "values": {
                "text": "亲，已帮你找到相关酒店信息"
            }
        }
    ]
}
```


具体使用信息见 https://www.kancloud.cn/turing/www-tuling123-com/718227
使用其他接口也可以，此处不作强制要求。

2.3.4 测试用例

用户甲新建与机器人的一个会话，向机器人发送消息并持续聊天会话一段时间（如两分钟）。机器人的互动性能评估指标会相应的发生变化，展示最新的互动性能。

2.4 其他功能

新增其他感兴趣的功能，给出用户需求、用户界面、MVC 设计和测试用例。

Project 3 股票价格预测

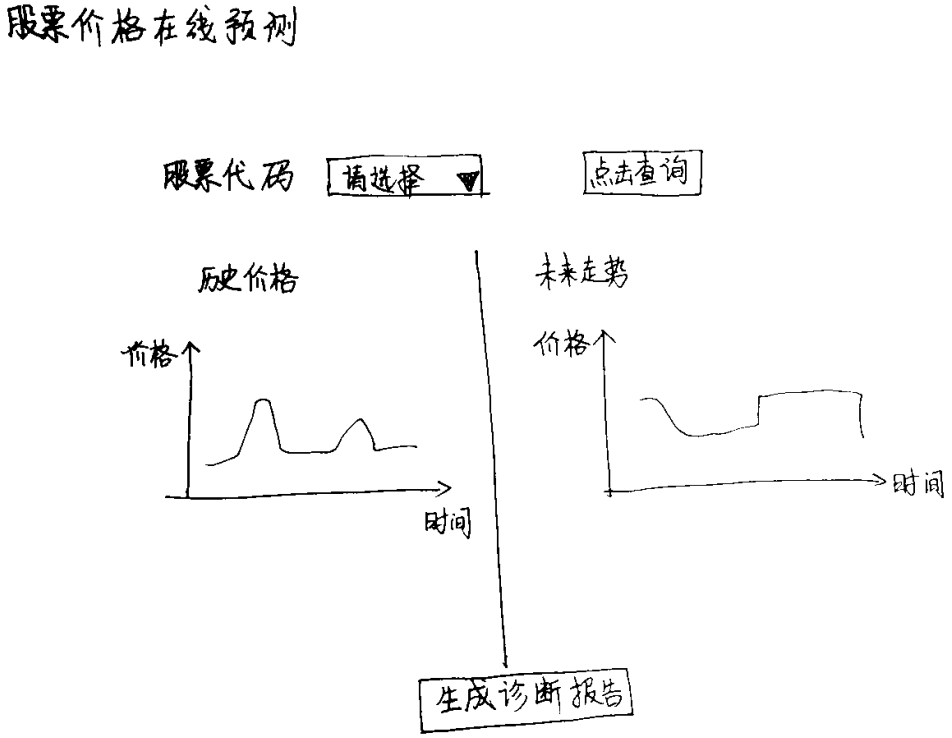
随着人们生活水平的提高，越来越多的人开始关注并参与到股票市场投资中去。股票具有高收益的同时也具有高风险，股票市场受多种因素的影响，价格令人无法捉摸。股票价格预测的研究具有巨大的价值，在人工智能飞速发展的今天，越来越多的研究者开始研究如何更加智能的预测股票价格。本项目旨在为投资爱好者搭建一个平台，可以查看感兴趣的股票价格未来预计走势和股票评估。

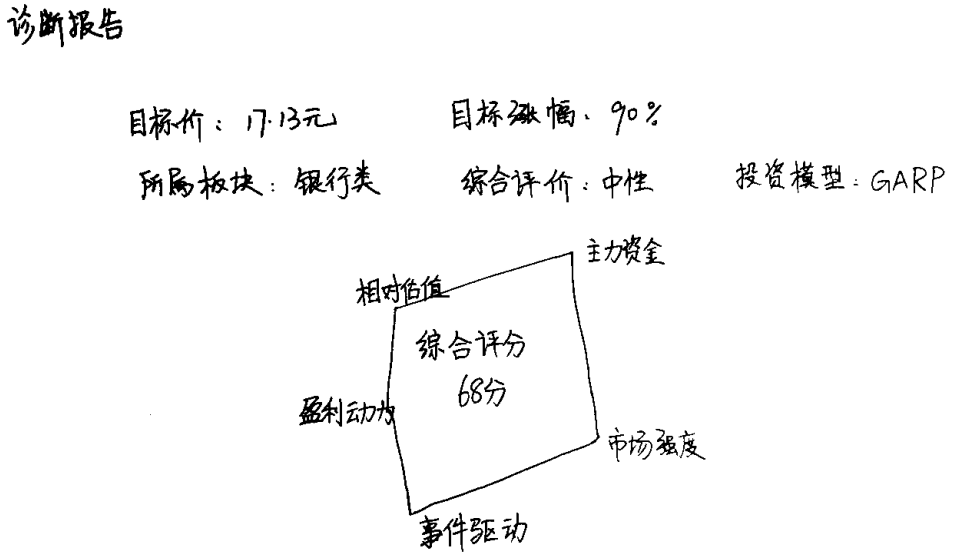
3.1 用户需求

对于投资爱好者，登录股票价格预测系统，输入或选择某只股票，可以查看其历史价格变化和未来预期价格估算（利用给定的数据集训练股票价格预测模型作为后台进行价格估算）。

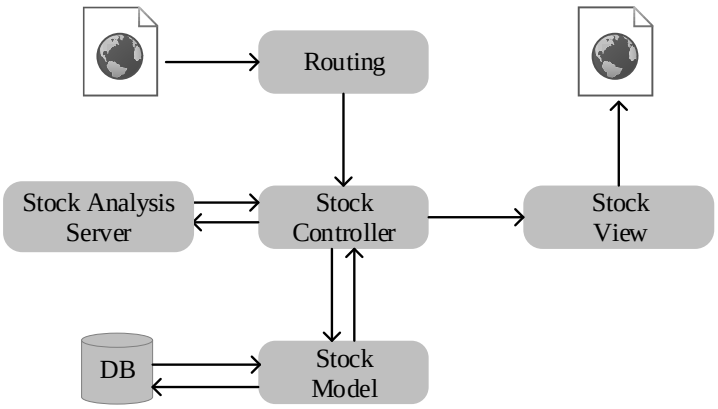
3.2 用户界面

投资爱好者甲登录系统后，输入或选择某只股票，可以查看其历史价格变化和未来预期价格估算，以及对这只股票的综合评估。





3.3 MVC 设计



股票价格预测和分析模型可以使用已有模型，在后台运行作为服务器。Stock Controller 将用户想要查询的股票代码传给服务器，服务器将相应股票的分析数据传给 Controller，Controller 解析数据传给 stock 视图，网页渲染股票的历史价格、未来走势和诊断报告等。

模型训练数据：

ftp://ftp.nyxdata.com/Historical%20Data%20Samples/

数据介绍：

http://www.nyxdata.com/Data-Products/NYSE-OpenBook-History

股票价格预测模型例子：

https://github.com/dzitkowskik/StockPredictionRNN

3.4 测试用例

投资爱好者甲登录系统，选择感兴趣的股票代码，点击“查询”按钮，可以得到股票的历史价格、未来走势和诊断报告。

附录

1. 测试说明

以上测试用例的给出可能未覆盖到所有情形，大家在进行测试的时候需要考虑测试的覆盖率。这里对代码测试进行一个补充说明，以便大家对此能够更快的掌握。以下基于 Project 1 进行说明。

1.1 Model 测试

以用户模型为例，位于 test/models/user_test.rb，首先生成一个@user 对象，然后 assert 用户是否有效，这里的调用 valid 方法会去检查你的模型中的相关的 validates 语句是否正确，若@user.valid?为 false，那么此 assert 会报错，代表"should be valid"这条测试没有通过，单独运行此测试文件使用 rake test test/models/user_test.rb

```
class UserTest < ActiveSupport::TestCase
  # test "the truth" do
  #   assert true
  # end

  def setup
    @user = User.new(name: "Example User", email: "user@example.com",
password: "password", password_confirmation: "password")
  end

  test "should be valid" do
    assert  @user.valid?
  end

  ...

end
```

1.2 View 和 Controller 测试

以用户登录为例，位于 test/integration/user_login_test.rb，首先同样生成一个@user 模型，这个@user 的用户名和密码可以在 test/fixtures/users.yml 中指定，然后我们用 get 方法到达登录页面（sessions_login_path），然后使用 post 方法提交这个@user 的账号密码来登录，如果登录成功，当前应该会跳转至 homes 控制器下的 index 方法进行处理，assert_redirected_to 能判断这个跳转过程是否发生，然后调用 follow_redirect！来紧跟当前的跳转，用 assert_template 来判读跳转后的视图文件是否为 homes/index，最后在这个视图

文件下做一些测试，比如判断这个视图下连接为 root_path 的个数等等（根据当前登录的角色不同，当前的页面链接会不同，比如 admin 用户就会有控制面板的链接 rails_admin_path，而普通用户没有，因此可以根据链接的个数来判断当前登录用户的角色）。

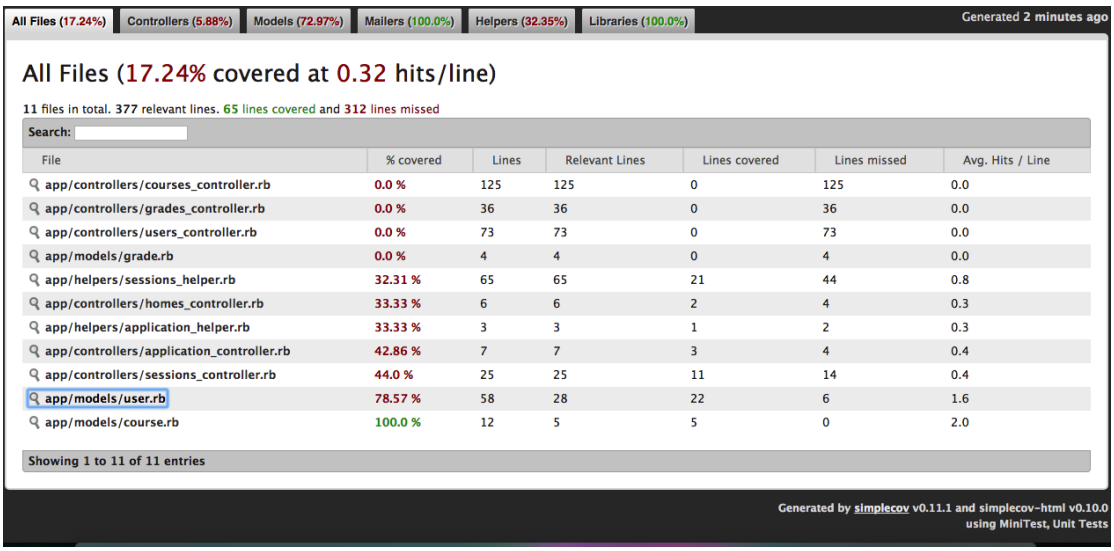
```
class UserLoginTest < ActionDispatch::IntegrationTest

  def setup
    @user = users(:peng)
  end

  test "login with valid information" do
    get sessions_login_path
    post sessions_login_path(params: {session: {email: @user.email, password: 'password'}})
    assert_redirected_to controller: :homes, action: :index
    follow_redirect!
    assert_template 'homes/index'
    assert_select "a[href=?]", root_path, count: 2
    assert_select "a[href=?]", rails_admin_path, count: 0
  end
end
```

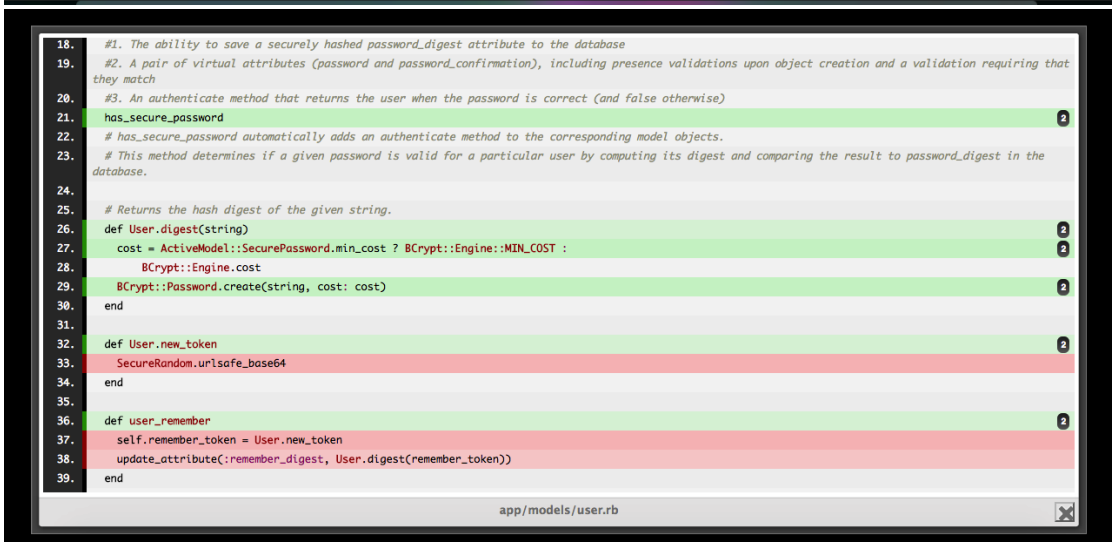
1.3 测试涵盖率检测

我们可以使用 simplecov 库来检测我们编写的测试对于我们的项目是否完整，步骤如下：
1. 在 Gemfile 文件中导入 simplecov 库：gem 'simplecov', :require => false, :group => :test，然后 bundle install 安装；
2. 在 test/test_helper.rb 的最前面加入 simplecov 的启动代码（这里默认使用 rails 自带的 test 框架，simplecov 也支持其他测试框架如 rspec，那么启动代码导入的位置请参考 simplecov 的官方文档）；

```
# 注意这里必须在 require rails/test_help 之前加入，否则不会生效
require 'simplecov'
SimpleCov.start 'rails'

ENV['RAILS_ENV'] ||= 'test'
require File.expand_path('../../config/environment', __FILE__)
require 'rails/test_help'

class ActiveSupport::TestCase
  # Setup all fixtures in test/fixtures/*.yml for all tests in alphabetical order.
  fixtures :all

  # Add more helper methods to be used by all tests here...
end
```

3. 运行 rake test,成功后会根目录的 coverage 下生成一个 index.html 文件，用浏览器打开能看到结果如下：





1.4 Travis CI 线上自动测试

上述为本地测试，我们可以使用 Travis CI 来实现自动测试，首先申请一个 Travis CI 的账号, 然后与自己的 github 连接起来, 接着在自己项目根目录中增加一个新的文件.travis.yml 如下, 这个文件中指定了测试需要的 ruby 版本, 数据库等配置以及一些测试前的脚本操作，当你的 github 发生更新后，Travis CI 会自动触发测试（需要你在 Travis CI 中自己设置自动/手动触发），然后读取你的.travis.yml 文件配置进行测试，其实也就是把本地测试拉到服务器上进行，测试成功后会在你的 github 项目给一个 buliding pass 的标签（见 CourseSelect 题目旁边），代表当前的代码是通过测试的。

```
language: ruby

rvm:
  - 2.2

env:
  - DB=pgsql

services:
  - postgresql

script:
  - RAILS_ENV=test bundle exec rake db:migrate --trace
  - bundle exec rake db:test:prepare
  - bundle exec rake

before_script:
  - cp config/database.yml.travis config/database.yml
  - psql -c 'create database courseselect_test;' -U postgres
```