

作业要求

本作业要求使用javascript相关的WEB技术，编写一个交互界面更加友好的聊天机器人。基本要求如下：

- (1) 不需要接入ChatGPT后台，也不必采用人工智能技术进行自然语言处理。
- (2) 针对目前ChatGPT交互性为基本文本对话框的方式，进行改进。采用合适的javascript前端技术编写WEB页面，实现 自拖动、自调整布局，自动推荐补齐，历史聊天记录分类、分层次呈现，根据输入内容（一句话、一段话、一篇文章等）区分不同输入方式等，至少呈现两种前端效果。
- (3) 后端不要求必须使用javascript，要求使用MySQL数据库存放确切的聊天答案，能够根据输入，返回确切的输出答案。对于不符合输入要求的问题，能够做出判断并拒绝回答。对于大量的并发访问需求，能够判断连接数过多（如大于3、大于5），并拒绝连接。
- (4) 按照个人软件编程习惯，形成完整的软件流程文档，记录个人的软件开发过程。

加分项目：

- (1) 采用成熟的前端或者后端javascript框架，加速作业的完成。
- (2) 具有热点问题排名、最多提问者排名、最佳提问等热榜。
- (3) 后端能够调用简单的Python程序，处理分词等简单的人工智能任务。

总览

- 项目名称：Mini-ChatGPT
- 开发工具：IDEA+VS Code+Pycharm
- 后端：Spring Boot+JPA+MySQL, python
- 前端：HTML5+CSS+JavaScript, Vue+Vue-Router+Vue x+Element-UI
- 前后端连接：前端使用Axios发送请求，后端通过SpringBoot的原生Controller的RequestMapping拦截访问。
- 前端接口测试：Mock JS
- 后端接口测试：Postman
- 项目架构：B/S

一些前期想法

Chat or Cheat?

但鉴于ChatGPT经常无法保证回答安全性与正确性。项目创新性地采用**问题预设与用户监督**的方式来构建，并不一定是单方面的回答，用户可以提出问题与回答问题。

特别地，我们赋予用户的权限包括：

1. 提出新问题，以及删除所提出的问题。
2. 对挂载的问题进行回答。
3. 对问题与答案进行喜欢与取消喜欢。

前端

项目配置：

以Vue.js作为框架，生成Vue CLI快速开发，在之中大量使用npm方法对现有库进行安装与运行。

考虑到存在需要全局调用和修改的变量，项目采用了VueX进行对特殊参数进行管理。

```

import Vuex from 'vuex';
import Vue from 'vue';

Vue.use(Vuex);

const store = new Vuex.Store({
  state: {
    isAuthenticated: false, //记得修改回false
    username: "",
    userid: 0,
  },
  mutations: {
    setIsAuthenticated(state, isAuthenticated) {
      state.isAuthenticated = isAuthenticated
    },
    setUsername(state, username) {
      state.username = username
    },
    setUserId(state, id) {
      state.userid = id
    },
  },
});

export default store;

```

页面设计

为了实现快速开发，采用常用的Element-UI的组件进行设计。

为了实现页面的有效切换，使用了Vue.js的官方路由Vue-Router作为切换工具。

```

JS main.js > ...
You, 3天前 | 1 author (You)
> import Vue from 'vue'; ...

Vue.config.productionTip = false;

Vue.use(ElementUI);
Vue.use(Vuex);
new Vue({
  store,
  render: h => h(App),
  router: router
}).$mount('#app');

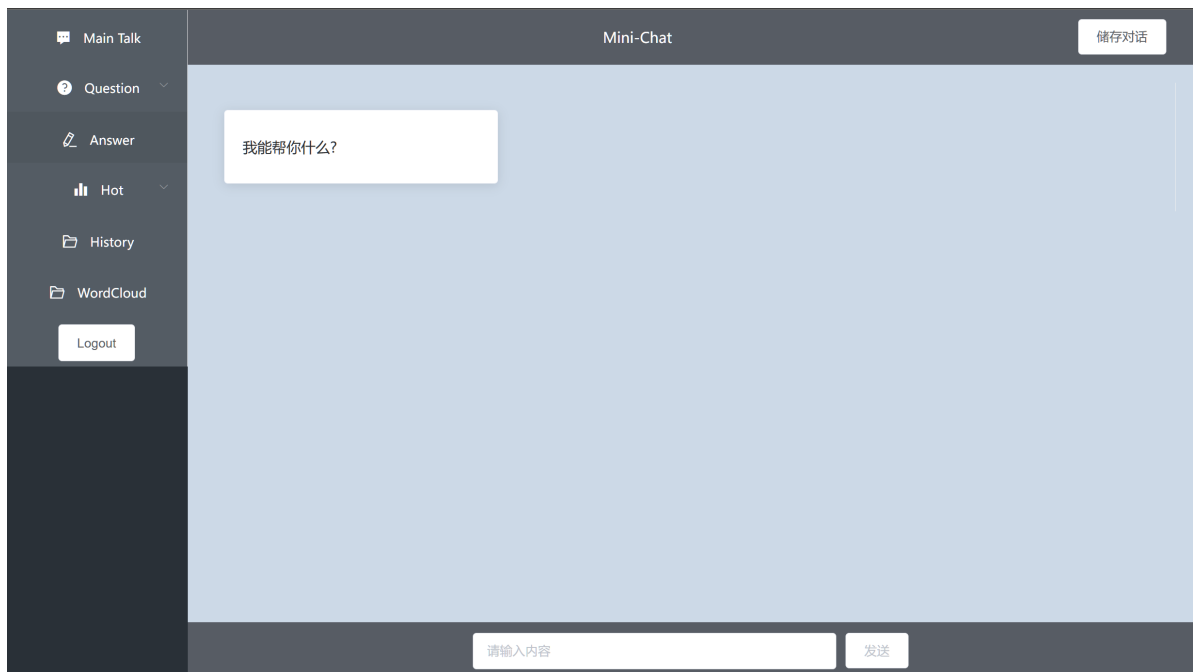
```

以MainPage.vue作为根页面，分为两个区域

```
App.vue x Input.vue M History.vue | ▶ HEX ◀
src > App.vue > Vue Language Features (Volar) > {} script > default
  You, 7天前 | 1 author (You)
1  <template>
2    <router-view></router-view>
3  </template>
4
5  <script>
6    import mainpage from './components/MainPage.vue'
7    export default {
8      components: { mainpage },
9    }
10 </script>
11
12 <style>
13
14 </style>
```

- 左侧的Aside侧栏区
 - 作为常显的部分。
 - 作为router的连接点，用于切换各个功能与板块。
- 右侧的操作区
 - 根据侧栏区router进行切换。
 - 分为若干个板块，包括：
 - Main Talk主要对话区
 - Question 问题区
 - 问题大厅：包括数据库的所有问题
 - 我的喜欢：包括用户所喜欢的所有问题
 - 我的提问：包括用户所提出所有问题。
 - 添加问题：可以在这个页面进行问题的添加。
 - Answer 回答区
 - 用户可以在这个区域进行回答问题
 - 同时，特别设计，
 - 在Main Talk页面提出一个没有答案的问题，我们可以点击一个按钮，直接到达回答区，并且自动读取所提出的问题。
 - 在Question页面的问题大厅，也可以直接点击到达回答区
 - Hot 热榜区
 - 包括对问题的排行：点赞排行
 - 包括对用户的排行：提问排行
 - 以上两个排行都是通过连接数据库，获取内容，并且返回给前端，进行动态调整的。
 - History 历史记录区
 - 在Main Talk区域的右上角有一个储存对话的功能，点击后，本次对话可以被存在数据库中。

- 在History区域可以访问查看该用户保存过的所有历史记录。
- Word Cloud 图云区



登录界面

Welcome to My App

* Username

Please enter username

* Password

Login

```
router.beforeEach((to, from, next) => {  
  // const isAuthenticated = localStorage.getItem('isAuthenticated');  
  if (to.name !== 'login' && (!store.state.isAuthenticated) || !isAuthenticated) {  
    next({name: "login"})  
  } else next()  
})  
export default router
```

这里在router.js里面定义了一个router.beforeEach的方法，可以对前端所有的跳转进行拦截。

在未登录状态，访问所有的非登录页面都会跳转到login界面，保证了前端的安全性。

```
const store = new Vuex.Store({
  state: {
    isAuthenticated: false, //记得修改回false
    username: "",
    userid: 0,
  },
  mutations: {
    setIsAuthenticated(state, isAuthenticated) {
      state.isAuthenticated = isAuthenticated
    },
    setUsername(state, username) {
      state.username = username
    },
    setUserId(state, id) {
      state.userid = id
    },
  },
})
```

这里的isAuthenticated是储存在由Vue.x管理的store.js里面的全局变量，初始化为false，在登陆后这个值被赋值为true。

主页面

```
1  <template>
2    <div id="app">
3      <el-container>
4  >    <el-aside width="200px"> ...
100   </el-aside>
101   <!-- <keep-alive> -->
102   <router-view></router-view>
103   <!-- </keep-alive> -->
104  </el-container>
105  </div>
106 </template>
```

主页面是一个el-container，左侧的el-aside作为侧栏区，右侧router.js渲染的router-view作为操作区。

侧栏区

侧边栏，采用了可折叠的动态组件，实现了侧边导航栏的可折叠与分层次。

完全打开后如下图：



使用了Vue.js的官方路由Vue-Router进行页面切换，以此更好的实现多层次的页面显示，使开发界面更加清晰明了。

```

import MyAskedQuestionList from '../components/MyAskedQuestion.vue'
import AskRank from '../components/AskRank.vue'
import History from '../components/History.vue'
import WordCloud from '../components/wordcloud.vue'
You, 22小时前 • 开发处理 ...

Vue.use(VueRouter)

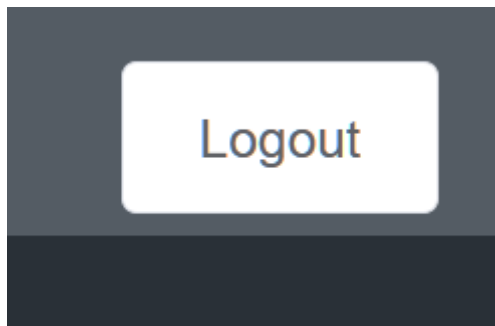
const router= new VueRouter({
  routes: [
    {path: '/',component:Login,},
    {path: '/mainpage',
      component:mainpage,
      children:[
        {path: '',component:MainTalks,name:'mainpage',},
        {path: '/talk',component:MainTalks},
        {path: '/quest',
          component:Quest,
          children:[
            {path: '/mylikedquestlist',component:MyLikedQuestion,},
            {path: '/myaskedquestlist',component:MyAskedQuestionList,},
            {path: '/allquestlist',component:AllQuestionList,},
            {path: '/addquestion',component:AddQuestion,},
          ]
        },
        {path: '/ans',component:AddAnswer},
        {path: '/likerank',component:LikeRank},
        {path: '/askrank',component:AskRank},
        {path: '/history',component:History},
        {path: '/image',component:WordCloud},
      ],
    },
    {path: '/login',component:Login,name:'login'}
  ],
});

router.beforeEach((to, from, next) => {
  // const isAuthenticated = localStorage.getItem('isAuthenticated');
  if (to.name!=='login'&&(!store.state.isAuthenticated))//||isAuthenticated))
    next({name:"login"})
  else next()
})

export default router

```

退出按钮，点击后用户退出，返回login页面。



操作区

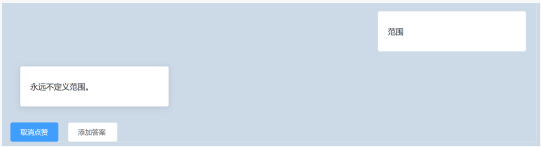
- 主对话

我能帮你什么?

你喜欢什么样的音乐?
今天是几号?
今天可以完成吗?
范围

请输入内容

发送

前端效果	内容	效果
发送按钮禁用	在输入内容为空，对发送按钮进行禁用	
输入自动补全	获取后端的问题列表，通过前缀的方式进行问题的过滤，实现问题的自动补全	
问题回答逐字出现	对于问题的回答进行逐字地显示出来，类似chatgpt的打字机地效果	
页面动态滚动	由于回答呈现逐字逐行出现，页面检测回答正在生成时，会自动下拉对话框，跟随回答一同下降。	
对于没有的问题进行提出问题	点击后，可以直接添加问题。	
对于已有的问题进行点赞	在回答出现后，下侧会有点赞与取消点赞的按钮，可以动态改变喜爱状态。	
对于没有答案的问题进行答案的提出	在答案出现后，下侧会有添加答案的按钮，点击后，可以直接将问题注入到Answer界面，进行回答的添加。	
储存对话	在MainTalk的右上角，存在一个储存记录的按钮，用户可以根据所需，对当前talk进行储存。点击后，记录储存在后端。	

- Question
 - 问题大厅

前端效果	内容	效果图
基础列表	包含所有问题	
固定表头和列高度	限制了列表的高度，可以动态滚动。 并且，在滚动时，表头固定。	
展开行	每一个问题可以展开，看到更详细的数据。	
分页效果	可以对每页所显示的问题条数进行选择，下册的分页选择条也跟着动态改变，直接输入页数跳转到某一页。	
点赞按钮	在操作栏，可以对喜欢的问题进行点赞，收藏后的问题被收入我的喜欢之中。	
添加答案按钮	在操作栏，可以对想要回答的问题进行回答，点击后会载入到Answer页面，同时问题自动载入。	

◦ 我的喜欢

与问题大厅具有相同的表格效果。

点击取消点赞，这个问题会被立即移除。

	id	asker	title	操作
>	17	admin	你喜欢什么样的音乐?	取消点赞
∨	19	admin	今天可以完成吗?	取消点赞
<div><div>id</div><div>19</div></div> <div><div>asker</div><div>admin</div></div> <div><div>title</div><div>今天可以完成吗?</div></div> <div><div>content</div><div>今天可以完成吗?</div></div>				
>	21	admin	范围	取消点赞

共 3 条1条/页<123>前往1页

我的提问

同样的具有相同的前端效果。

特别地，存在一个KeyWords属性，这个属性是后端接收到问题标题后，调用python程序进行NPL语义分析而生成的。

存在一个删除按钮，点击后问题会被立即删除。

	id	asker	title	操作
∨	32	cp_cp	提问个问题测试下	编辑删除
<div><div>id</div><div>32</div></div> <div><div>asker</div><div>cp_cp</div></div> <div><div>title</div><div>提问个问题测试下</div></div> <div><div>content</div><div>我就想提问一下。</div></div> <div><div>keyWords</div><div>提问; 一下</div></div>				

共 1 条1条/页<1>前往1页

添加问题界面

热榜分为：问题排行榜（按照点赞数排行），用户排行榜（按照提问数排行）。

前端效果		效果图																					
带有斑马纹的表格	斑马纹，便于用户查阅，观感更佳。	<div>问题排行榜</div> <table><tr><th>排名</th><th>标题</th><th>点赞数</th></tr><tr><td>1</td><td>你喜欢什么样的音乐?</td><td>1</td></tr><tr><td>2</td><td>今天可以完成吗?</td><td>1</td></tr><tr><td>3</td><td>范围</td><td>1</td></tr><tr><td>4</td><td>今天是几号?</td><td>0</td></tr><tr><td>5</td><td>测试一下试试吗?</td><td>0</td></tr><tr><td>6</td><td>提问个问题测试下</td><td>0</td></tr></table>	排名	标题	点赞数	1	你喜欢什么样的音乐?	1	2	今天可以完成吗?	1	3	范围	1	4	今天是几号?	0	5	测试一下试试吗?	0	6	提问个问题测试下	0
排名	标题	点赞数																					
1	你喜欢什么样的音乐?	1																					
2	今天可以完成吗?	1																					
3	范围	1																					
4	今天是几号?	0																					
5	测试一下试试吗?	0																					
6	提问个问题测试下	0																					
排行榜自动更新	在其他页面对问题进行操作后，排行榜可以自动读取后端数据进行动态调整。	<div>提问排行榜</div> <table><tr><th>排名</th><th>姓名</th><th>提问数</th></tr><tr><td>1</td><td>admin</td><td>3</td></tr><tr><td>2</td><td>cp_cp</td><td>1</td></tr><tr><td>3</td><td>hello</td><td>0</td></tr></table>	排名	姓名	提问数	1	admin	3	2	cp_cp	1	3	hello	0									
排名	姓名	提问数																					
1	admin	3																					
2	cp_cp	1																					
3	hello	0																					

History

显示在MainTalk处储存的问题记录，实现历史记录功能。

	id	question
>	1	
>	2	
>	3	
√	52	今天是几号?
<div><div>id52</div><div>question今天是几号?</div><div>answer目前无答案。</div></div>		
>	53	今天是几号?

共 7 条5条/页 < 1 2 > 前往 1 页

WordCloud

对于不同的输入需求，对于短问题是以MianTalk回答问题的方式进行回答，而长文本需要额外的考虑，本项目，采用对长文本提取关键词并制作云图的形式来应对特殊的输入方式。

输入文本后，点击立即创建按钮，后端调用另一个python程序，自动分析文本，返回在服务器本地生成一个png，前端通过访问接口，获取图片。


```

public void addQuest(Question question)
{
    if (question.getKeyWords() == null) {
        question.setKeyWords()= new ArrayList<>();
    }
    List<String> ls=processString(question.getContent());
    for (String l : ls) {
        KeyWord keyWord=new KeyWord();
        keyWord.setWords(l);
        System.out.println(keyWord.getWords());
        keyWordRepository.save(keyWord);
        question.getKeyWords().add(keyWord);
    }
    questionRepository.save(question);
}

1 个用法  cp-cp
public List<String> processString(String input) {
    String output = "";
    List<String> ls=new ArrayList<>();
    try {
        ProcessBuilder pb = new ProcessBuilder( ...command: "python", "F:\\pythonProject\\chatbot\\main.py", input);
        Process p = pb.start();

        BufferedReader in = new BufferedReader(new InputStreamReader(p.getInputStream()));
        String line;
        while ((line = in.readLine()) != null) {
            ls.add(line);
        }

        p.waitFor();
        in.close();
    } catch (IOException | InterruptedException e) {
        e.printStackTrace();
    }
    return ls;
}

```

这个python程序在Java的QuestionService.class之中进行调用。

```

import jieba
import sys
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# 读取文本文件
# with open('text.txt', 'r', encoding='utf-8') as f:
# text =
text = sys.argv[1]
# 进行分词
words = jieba.lcut(text)

# 生成词云
wordcloud = WordCloud(font_path='msyh.ttc', width=800, height=800, background_color="white", min_font_size=10).generate(" ".join(words))

# 显示词云
plt.figure(figsize=(8,8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
wordcloud.to_file('F:\\pythonProject\\chatbot\\wordcloud.png')

```

这个python代码调用了jieba与wordcloud库，对前端发送到后端的长文本进行分词与生成图云。


```

@GetMapping("/image")
public ResponseEntity<byte[]> getImage() throws IOException {
    File file = new File(pathname: "F:\\pythonProject\\chatbot\\wordcloud.png");
    FileInputStream fis = new FileInputStream(file);

    byte[] bytes = new byte[(int)file.length()];
    fis.read(bytes);
    fis.close();

    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.IMAGE_PNG);

    return new ResponseEntity<>(bytes, headers, HttpStatus.OK);
}

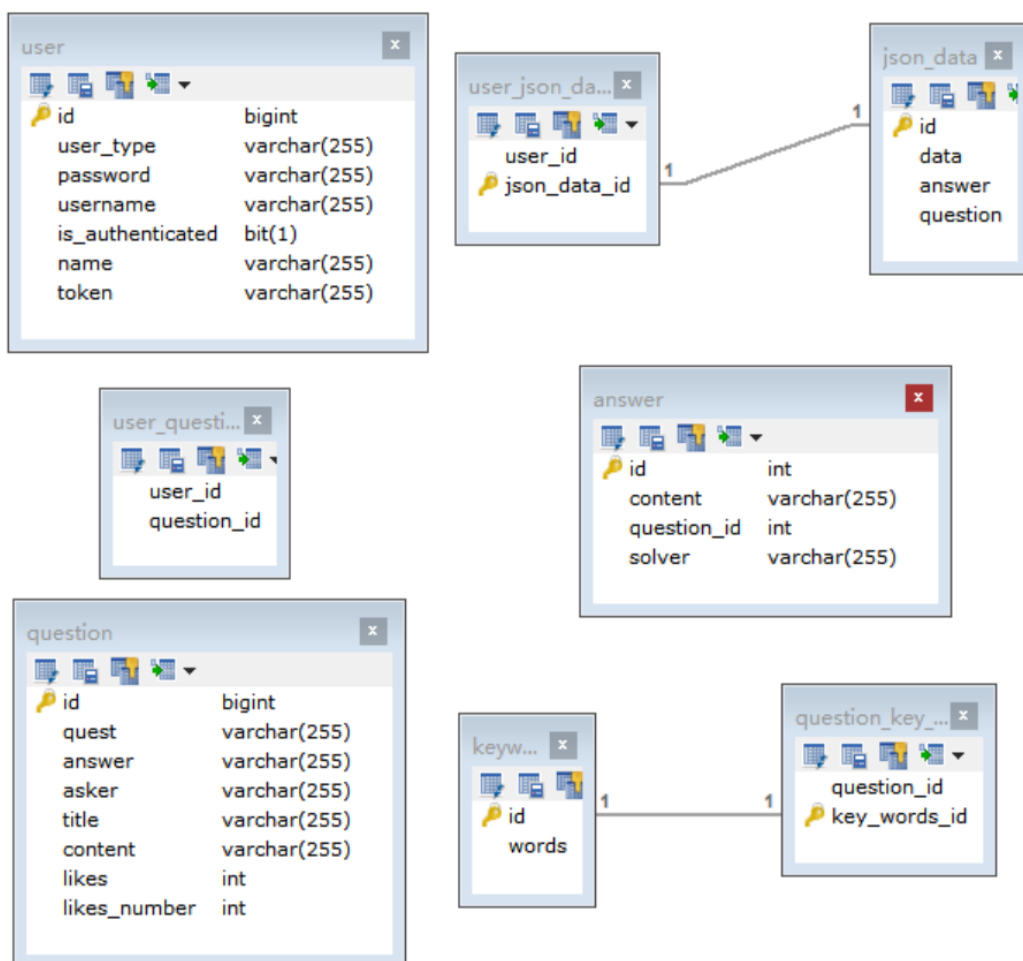
@PostMapping("/image/create")
public ResponseEntity generateImage(@RequestBody String text) {
    try {
        // System.out.println(text);
        String singleLineText = text.replace(target: "\n", replacement: " ");
        // call Python script with text as argument
        Process process = Runtime.getRuntime().exec(command: "python " + PYTHON_SCRIPT_PATH + " " + singleLineText);
        int exitCode = process.waitFor();

        if (exitCode == 0) {
            return new ResponseEntity<>(HttpStatus.OK);
        } else {
            return new ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR);
        }
    } catch (IOException | InterruptedException e) {
        return new ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR);
    }
}

```

在ImageController.class之中调用。

数据库设计



在数据库中，

- user与question存在多对多的关系连接
- user与jsonData通过外键id一对多的进行连接。
- question与keyword通过外键id一对多的进行连接。

前后端连接

并发访问需求限制

后端

```
#server.tomcat.max-connections=1
management.endpoints.web.exposure.include=tomcat
spring.datasource.hikari.maximum-pool-size=20
# ??????????10?
spring.datasource.hikari.connection-timeout=10000
```

Spring Boot使用的Hikari 连接池，通过设置连接池中最大连接数的属性配置，可以有效地处理并发请求。超过这个数量的连接请求会被放入等待队列中等待其他连接释放后再去获取。

前端

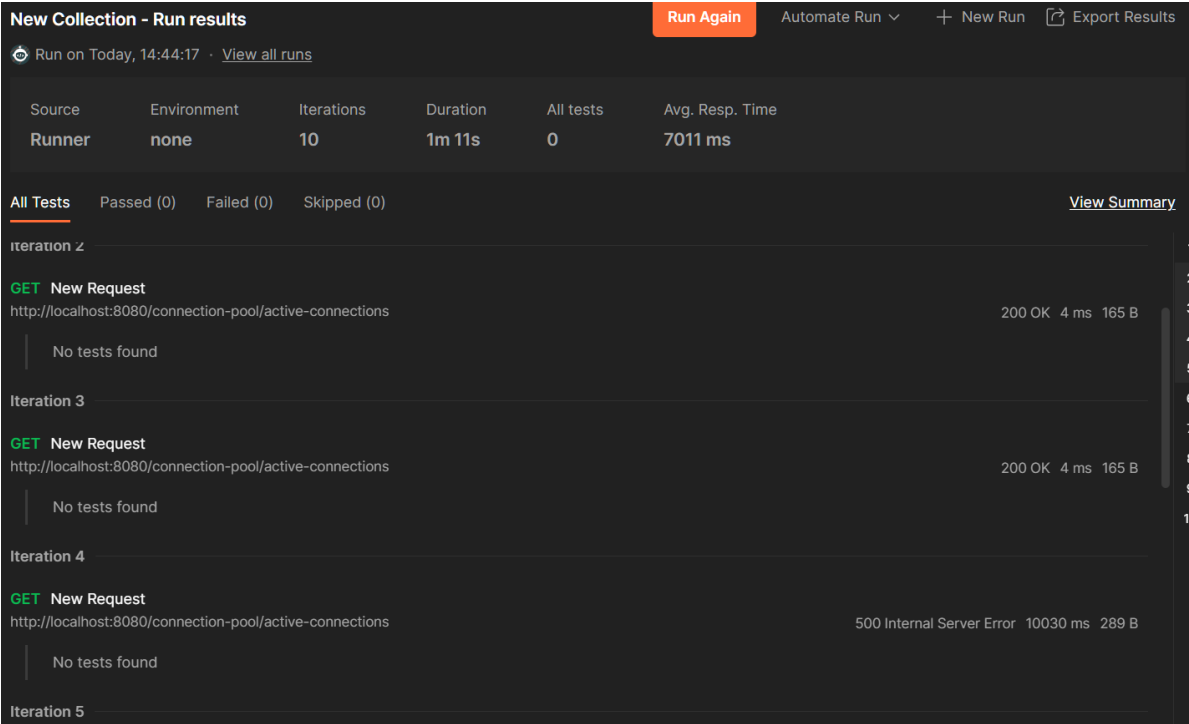
那么前端可以设置一个请求时限，一旦超过一定的限度，那么就说明后端到达了最大连接数，前端需要等待后再次连接。

```
axios.post('http://localhost:8080/user/login', this.form, { timeout: 5000 })
  .then(response => {
    if (response.status === 200) {
      // 登录成功，更新isAuthenticated状态并跳转到主页
      //localStorage.setItem('isAuthenticated', 'true');
      this.setIsAuthenticated(true);
      this.setUserName(this.form.username);
      this.setUserId(response.data.userId);
      this.$router.push({ name: 'mainpage' });
    } else {
      // 登录失败，弹出提示框
      alert("登陆失败");
      this.$message.error(response.message);
    }
  })
  .catch(error => {
    if (axios.isCancel(error)) {
      console.log('Request canceled:', error.message);
    } else {
      console.log(error);
      this.$message.error('Failed to login, please try again later.');
```

测试

```
spring.datasource.hikari.maximum-pool-size=3
```

设置最大连接数为3时，用Postman的Runner模拟前端高并发的数据请求。



很明显地发现，前三个请求可以轻松获取，而第四个请求超过10s也没有成功获取。

总结

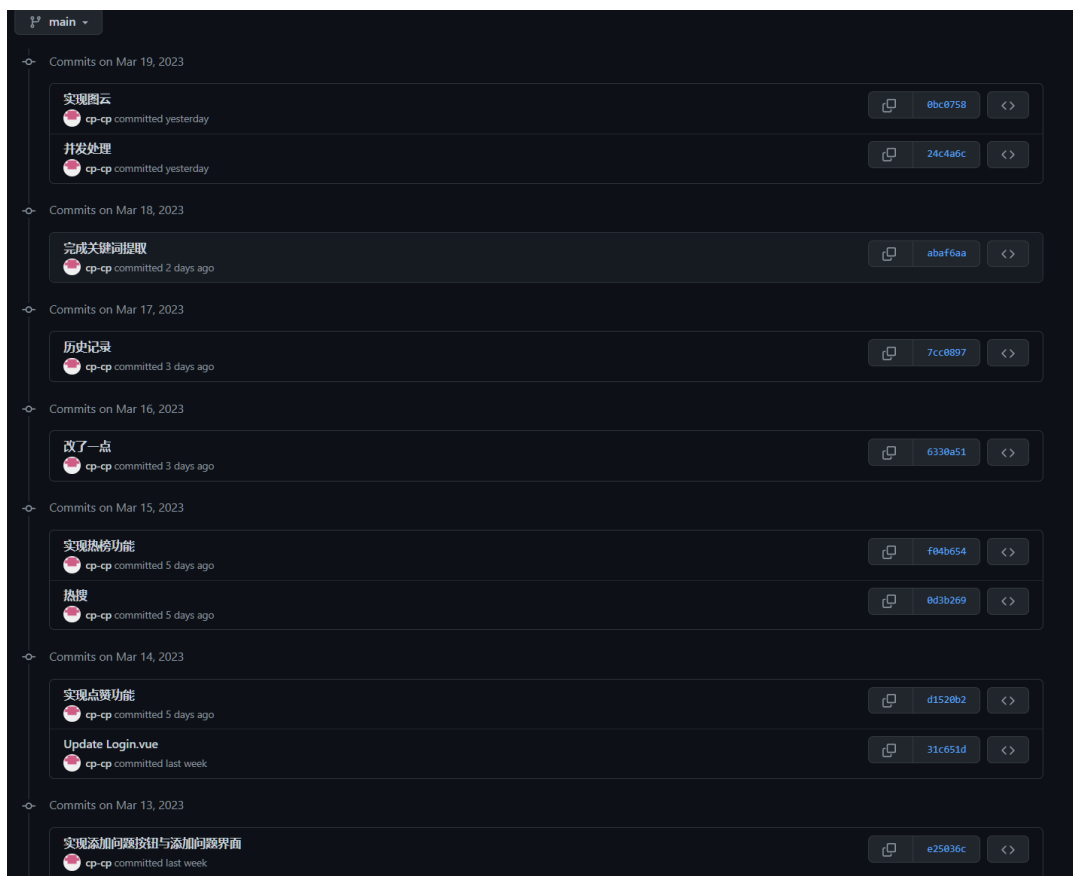
基础要求的实现

要求	完成情况
自动推荐补齐	在Input.vue组件中实现
自调整布局	在Answer与Quetsion添加时候实现
历史聊天记录分类	在History.vue组件中实现
回答文本的动态显示	在Card.vue组件中实现
区分不同输入方式	短文本在MianTalk中回答，长文本通过WordCloud进行分析
使用 MySQL 数据库存放确切的聊天答案	数据库中各个表与SpringBoot的类相对应，并且存在映射关系
根据输入要求输出确切的答案	区分了有回答的问题、无回答的问题。
不符合输入要求的问题做出判断并拒绝回答	区分了已存在的问题、未存在的问题
并发访问需求限制	前端检测超时，后端设置连接池最大连接数
软件流程文档	本文
采用成熟的 JavaScript 框架	Vue.js+Vue.router+VueX+ELement-ui
热榜	用户、问题热榜
后端调用 Python 程序 NLP	对问题进行分词、对长文本分析生成云图

数据统计

本项目于3月9日在github上进行开发。

- 前端
 - commit数量25次
 - 代码组成
 - 19个vue组件
 - mock.js
 - store.js
 - router.js
- 后端
 - commit数量10次
 - 代码组成
 - 7个controller
 - 5个bean
 - 3个service
 - 5个repository



特色内容

- 较为成熟的问答系统，不仅仅是预设回答，每个人都是提问者，每个人都是回答者，每一个人都可以为这个chat系统做出贡献。
- 可拓展性的后端程序，使用了相对重型SpringBoot开发后端，虽然增加了后端逻辑与编写复杂性，但也提高了代码的可读性与可重复利用性。
- 图云功能，一个很独特的小功能，能够清晰明了的了解文章的重点内容。

未来展望

- 安全认证：可以接入JWT或者Session对用户进行管理，增强系统的安全性。
- 接入ChatGPT：本次作业没有要求接入AI，但如果能在之后接入AI的话，会进一步提高人机交互的体验感。