**测试总结报告**

**B03-modeleval**

本项目使用ESlint完成代码风格测试,使用Jest进行单元测试,使用Cypress进行功能测试，使用Artillery进行性能测试。目前项目能够通过所有的功能测试，并产生性能测试报告。

**1.ESlint**

ESlint 是一个开源的 JavaScript 代码检查工具，被广泛用于发现代码错误、编程错误和不符合某些编码风格的代码。ESLint 的主要目标是提供一个插件化的架构，使得任何人都可以使用自定义的规则。

为了覆盖工作区尽可能多的代码,我们在代码风格测试中引入了如下扩展：

- eslint:recommend :ESlint推荐的规则,包含js的常见错误

- plugin:jest/recommended : Jest测试的推荐规则集

- plugin: cypress/recommended : Cypress测试的推荐规则集

- next: NextJS的推荐规则集

- prettier: Prettier代码检查工具的推荐规则集,避免ESlint与Prettier冲突

同时,为了编写方便,我们关闭了一些我们认为没那么必要的规则,包括不规则空白检查,JSX中为转义字符检查等。

**2.Jest**

**3. Cypress**

Cypress 是一个开源的端到端测试框架，专为现代 web 应用程序设计。它使开发者能够编写各种类型的测试，包括单元测试、集成测试和端到端测试，本次我们使用Cypress进行端到端测试，这涵盖了功能测试与部分性能测试。

Cypress不仅提供了较好的端到端测试页面,还提供了任意帧回退的能力,能够展示测试期间每一个状态的网页画面,方便我们对出现的问题进行调试。

我们拥有测试账户test@test.com，有测试用数据集333.csv与777.csv，在进行一些配置后，就可以开始运行测试。

我们共编写三个脚本，分别对网页的登录登出功能、测试任务运行功能与数据集的增删改查功能进行了测试，这涵盖了我们项目的绝大部分内容。这些脚本的主要内容是:

**sign.cy.js:测试登录登出功能,包含以下假设:**

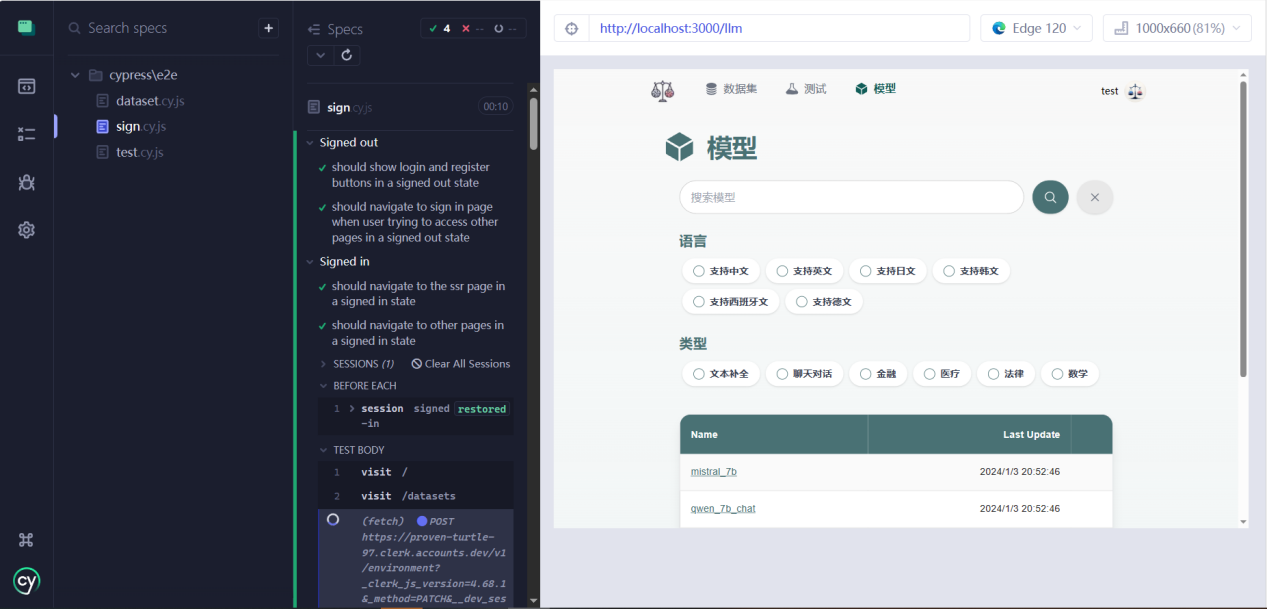
- should show login and register buttons in a signed out state:在登出状态时,检查页面是否有登录与注册按键,并检查登录才能展示的页面是否不能展示。

- should navigate to sign in page when user trying to access other pages in a signed out state：在登出状态时，检查点击其他链接，会重定向到登录页面。

- should navigate to the ssr page in a signed in state：检查登录后，应显示的组件是否显示。

- should navigate to other pages in a signed in state：检查登录后，是否能正确地跳转到其他页面。

测试结果：



**dataset.cy.js测试数据集的增删改查功能(模型的这些功能与之相仿)，测试前需要在测试账户中提前上传名为333.csv与777.csv两个数据集，由任意其他账户上传名为subjective1.csv数据集，该测试包含以下假设：**

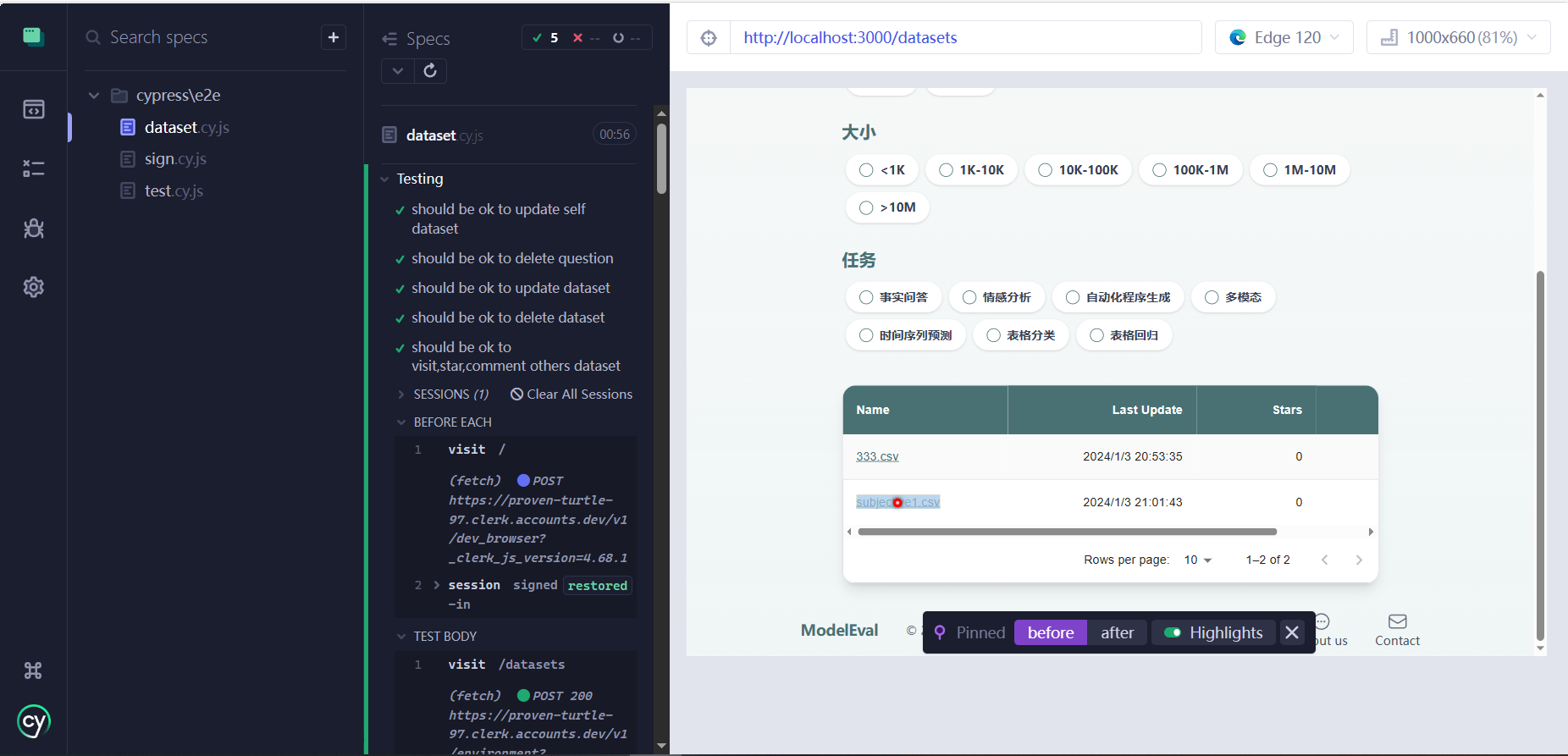
-should be ok to update self dataset，登录后，打开一个自己的数据集，点击添加按钮，输入特定信息，查看是否有新条目添加到该数据集中。

-should be ok to delete question，登录后，打开前面打开的数据集，找到刚刚加入的条目，点击删除按钮尝试删除之，测试是否成功。

-should be ok to update dataset，登录后，打开前面打开的数据集，点击“修改数据集属性”，输入指定内容，查看数据集的名字、描述、标签等是否被正确更改。

-should be ok to delete dataset，登录后，打开前面打开的数据集，点击“删除数据集”，查看数据集是否被成功删除。

-should be ok to visit,star,comment others dataset，登录后，打开数据集页面，点击一个他人已经创建的数据集，查看是否能看到指定内容；点击收藏按钮，查看是否能收藏之；点击评论按钮，输入指定内容，查看是否能正确评论。

测试结果：  


**test.cy.js用于测试评测任务是否能够正常进行，在评测前需要清空目前的所有评测任务，并上传上述的333.csv与777.csv。该测试包含以下假设：**

-should show testing page

，打开测试页面，应该能看到一些该页面该出现的组件。

-should available to create subjective new testing

，点击创建新测试按钮，选择333.csv与一个模型，选择主观评测方法，点击创建新测试，应该能看到新任务出现。

-should show task after creating

，在页面等待直到出现“等待主观评测”，这意味着该项任务已经完成，可以成功展示。

-should be available for evaluating subjective testing

，进入上述的主观评测任务，点击“进行主观评测”，为每一个问题都勾选“正确”，之后返回，界面应该出现“已完成评测”。

-should be available for changing subjective testing result

，再次进入上述的主观评测任务，点击“修改主观评测” ，将其中两个问题改为“错误”，退出后应该会发现评测得分同步变化。

-should correctly show the result of subjective testing and delete it

，回到测试的主页面，应该还是能看到这个已经人工评测完成的任务，并且可以点击删除按钮清除该任务。

-should be available for creating objective testing

，点击创建新测试按钮，选择777.csv与一个模型，选择评测方法客观评测，点击创建新测试，在测试主页面应该出现一个新的测试任务。

-should be available to see the result of objective testing

，等待上述的客观测试完成，进入该客观测试的具体页面，可以看到其已经评测完成的信息。

-should be available to omit objective testing

，返回测试主页面，可以看到上述客观测试已完成；可以删除该测试信息。

-should be available to create comparative testing

，点击创建新测试，选择333.csv（主观数据集），选择对抗评测，选择两个模型发起评测，评测应该能够成功发起。

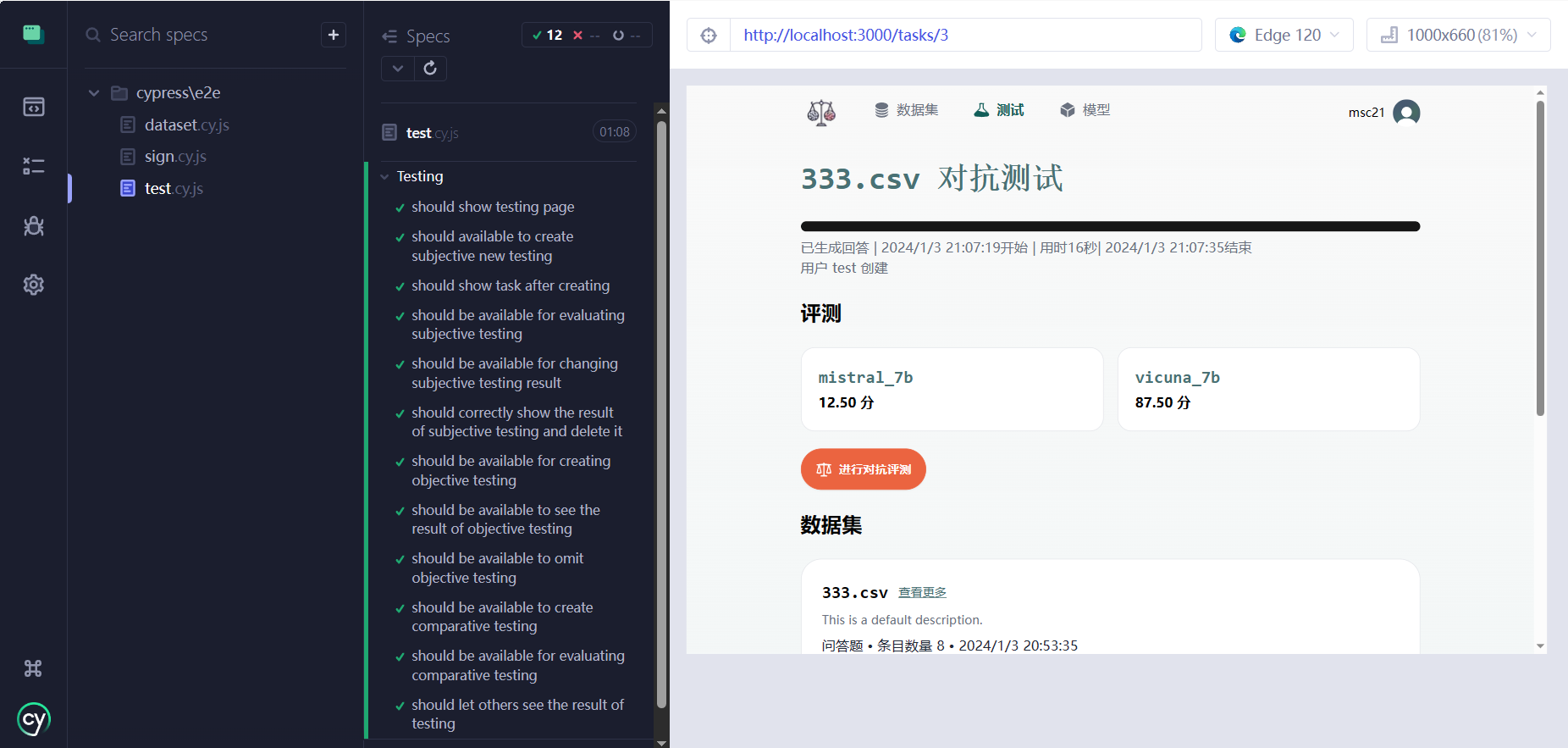
-should be available for evaluating comparative testing

，等待评测成功，进入评测具体信息，点击进行对抗评测，应该能够对模型给出的答案进行排序并提交；完成全部问题后返回，应该可以看到测试完成，并看到每一个模型的分数。

-should let others see the result of testing

，登出当前账号，登录另一账号，在“他人创建的测试”一栏应该看到刚刚没删除的对抗评测。

测试结果：



除此之外，我们在cypress文件中插入了时间点，每一次测试时，**对以下时间进行了测试**：  
 - 加载登录页面的时间。

- 登录后，加载数据集页面的时间。

- 登录后，加载测试页面的时间。

- 登录后，加载模型页面的时间。

- 向数据集添加一个条目需要的时间。

- 删除一个数据集条目需要的时间。

- 更新数据集需要的时间。

- 删除数据集需要的时间。

- 收藏数据集需要的时间。

- 在数据集下留言并提交需要的时间。

- 进行一个8题的主观测试需要的大概时间。

- 人工评测一个8题的主观测试需要的大概时间。

- 修改2题主观测试结果并返回需要的大概时间。

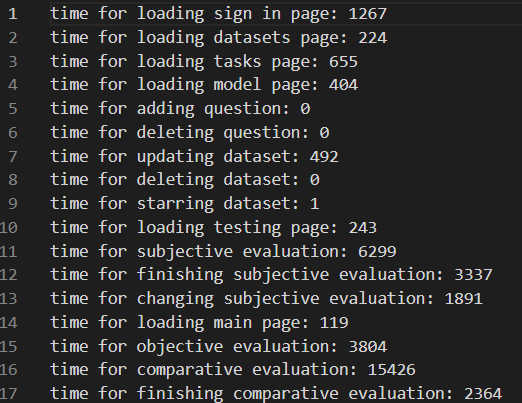
- 进行一个15题的客观测试我需要的大概时间。

- 进行一个8题的对抗测试需要的大概时间。

- 人工排序不同模型给出的该8题结果并确认，直至生成分数的时间。

根据cypress的配置，目前所有的信息会被输出到根目录以下的“testlog.txt文件”，可以在其中查看结果。

一个大致的时间结果如下（单位：ms）：



可以看出，由于初次加载，刚开始加载登录页面耗时较长，时间在一秒多一些；此后加载页面的时间减少，大概在0.5s左右，在有缓存后加载主页的时间仅需要0.1s左右。

对于数据集操作，由于目前体量较小，且端对端测试时服务器资源充足，增删改查操作几乎是瞬间完成，其中update dataset耗时492ms是因为计算了填写表单的时间。

对于8道题的主观测试大概是6s左右，但是对于同样的数据集对抗测试需要的时间几乎是其两倍，可能是因为使用了多个模型的原因；对于15题的客观题，其完成时间较快，约为3s。对于人工评测操作与修改评测结果操作，其完成时间很快，测试结果显示，人工完成8题的评测耗时3.3s，这包含了点击“正确”按钮的时间；而修改2题的评测耗时1.8s，两者相减可以估算出进入人工评测页面+确认结果+退出评测页面的时间约为1.3s。

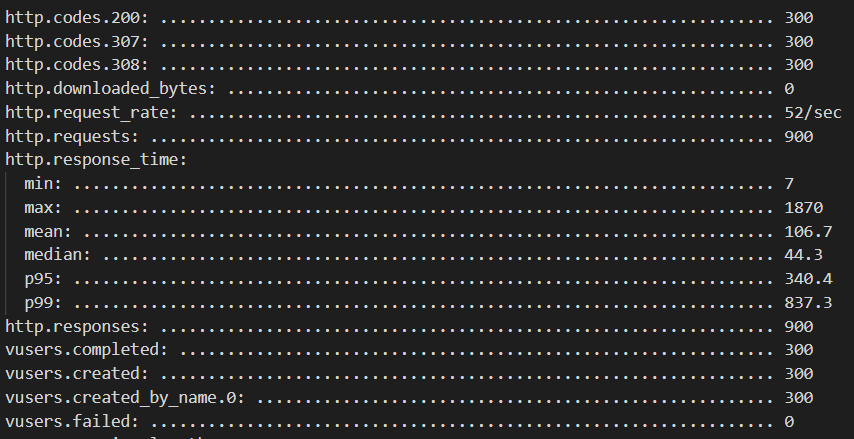
此后我们又进行了不同规模的数据集的测试，测试的时间不一定与数据集规模呈正比关系。例如，体量为195的客观数据集，耗时约为250s，这与15题3s的比例关系不太相符。这可能与大模型接口的性能不太稳定有关。

**4. Artillery**

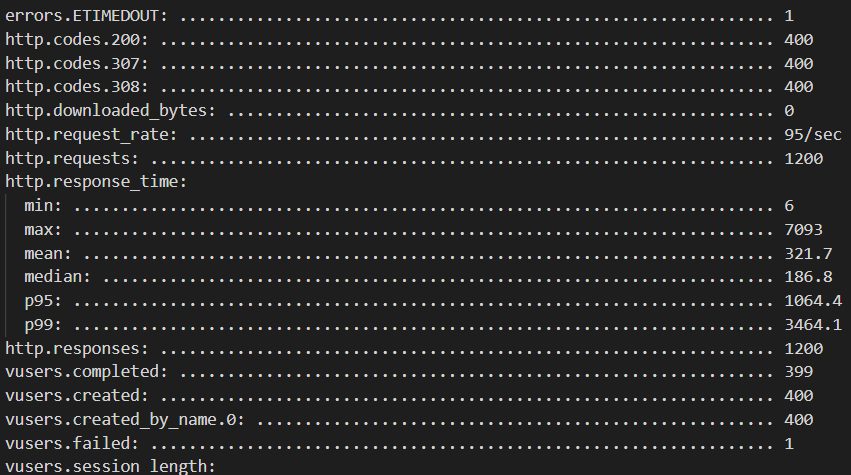
Artillery 是一个现代、强大的负载测试和功能测试框架，主要用于测试和监控你的代码在高负载下的性能。它可以帮助你发现应用程序在高并发用户访问下可能出现的性能瓶颈和问题。

本次我们主要对在服务器部署后的程序进行了该测试。测试包含两部分：并发用户访问测试，并发发起评测请求测试。代码包含在项目根目录下的artillery文件夹中。

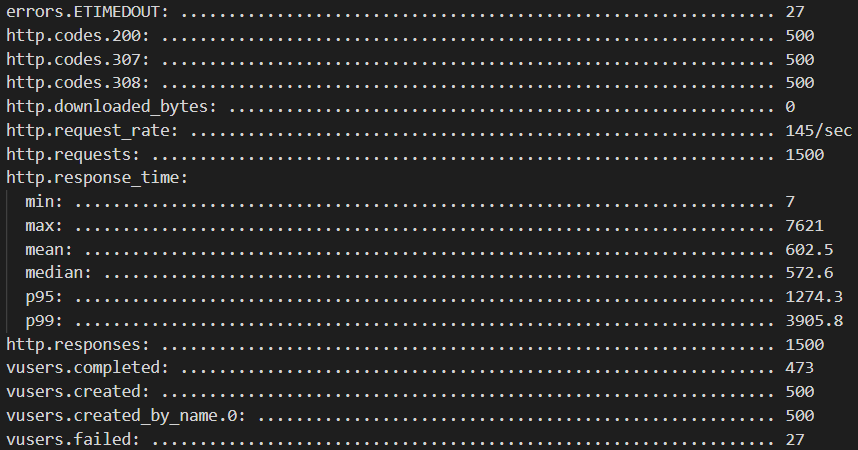
**对于用户访问，我们模拟了5s内300，400，500，2000用户的情况，结果如下：**



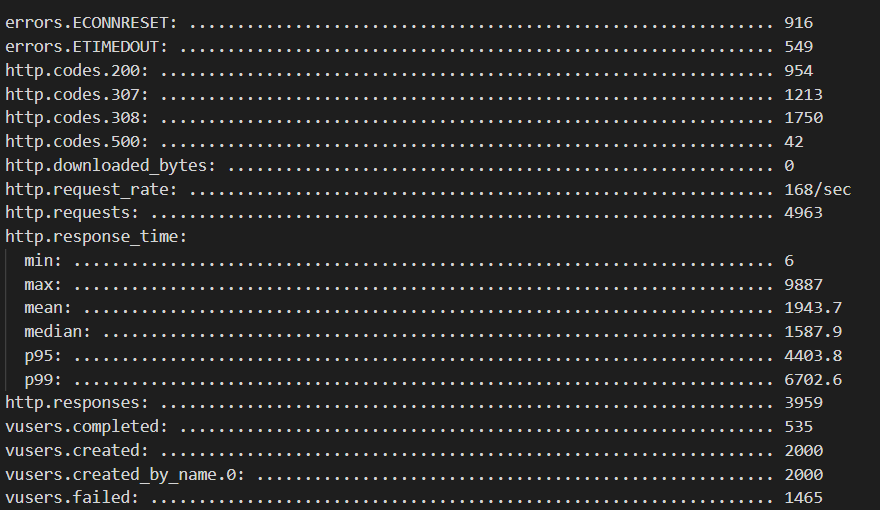
在300用户并发访问的情况下，服务器响应正常，所有用户都能接收到响应，平均响应时间为106ms，但是有大概3个的响应时间在1s左右。



在400用户并发访问的情况下，服务器出现异常，有一个用户的请求被拒绝，所有用户的平均响应时长增加，达到321.7ms。除此外，有约20个用户的响应时长超过了1s，最长的响应时间达到了7s。

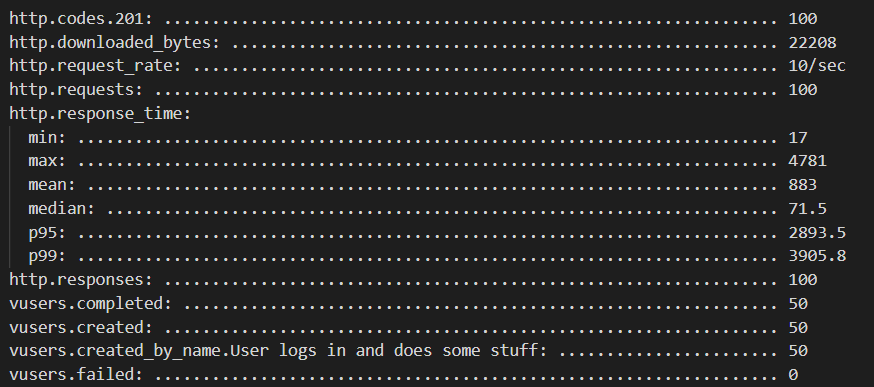


在500用户并发访问的情况下，服务器异常率上升，异常人数达到27。并且，平均响应时长进一步上升，达到602.5s。

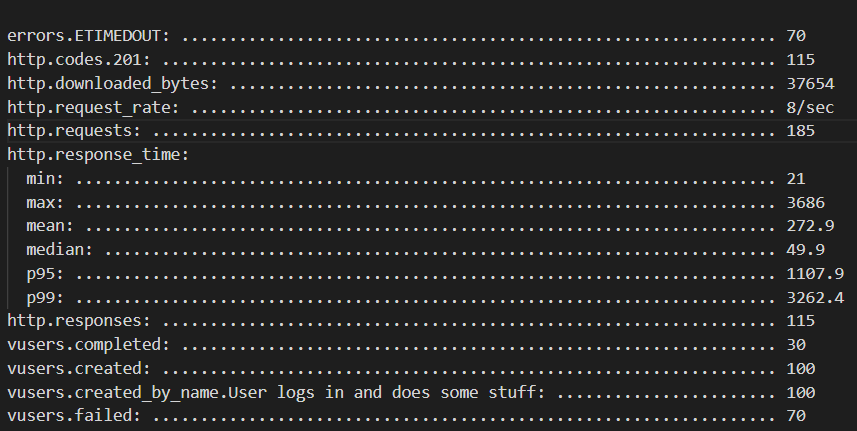


在2000用户并发访问的情况下，服务器几乎崩溃，异常人数达到1465，平均响应时间达到1.5s。

**经测试，部署后项目能够允许的并发访问人数大致为300。**

**对于并发评测测试，我们分别模拟了5s内创建50个任务与100个任务的情况**，结果大致如下：  
 

对于50个任务的情况，创建任务全部成功，并且打开页面可以发现所有任务都在正常运行，平均响应时间大概在1s之内。



对于100个任务的情况。创建任务失败率达到70%，有70个任务无法被创建并运行。同时，由于并发量过大，导致了服务器直接崩溃需要重启。

此后我们又测试了70个任务的情况，虽然全部成功，但是打开页面后显示有的任务已经无法被正常的执行，其进度停留在0%的状态。

**经测试，部署后的项目服务器，能够接受的并发测试任务数量约为50。**

1.5 测试结论

本项目代码规范能够通过ESlint风格测试，重要单元通过Jest单元测试，各功能通过Cypress测试。性能方面，其各功能运行时间，最大客户负载量见上述Cypress与Artillery部分。