

Our first open source contribution.

Efthymia Kostaki
t8170055@aueb.gr

Stergios Sozos
t8170129@aueb.gr

May 2020

Abstract

The procedure of contributing to an open-source project was unique and unlike anything we have done before. All the phases for this assignment, from researching for the open-source project to selecting one and starting working on it, urged us to work more professionally and to create production-ready code.



Figure 1: AnitaB [1]



Figure 2: MentorshipSystem[2]

Contents

1	Project Selection Process	4
1.1	Troubleshooting	5
2	Mentorship Backend	7
3	Initial Communication with the community	7
4	Our contribution	8
4.1	Issue 1 - Create Quality Assurance table for Update Task API - <i>Merged #473</i>	8
4.1.1	Issue	8
4.1.2	Communication with the team	9
4.1.3	Our work	9
4.1.4	Testing	9
4.1.5	Code Reviews	10
4.1.6	Change	10
4.2	Issue 2 -Fix description messages on Mentorship Relation - <i>Merged</i> <i>#554</i>	10
4.2.1	Issue	10
4.2.2	Communication with the team	12
4.2.3	Our work	12
4.2.4	Testing	12
4.2.5	Code Reviews	12
4.2.6	Change	12
4.3	Issue 3 - ‘Complete task’ not including all the necessary informa- tion when request id is not in accepted mode - <i>Accepted #537</i> . .	12
4.3.1	Issue	12
4.3.2	Communication with the team	14
4.3.3	Our work	14
4.3.4	Testing	14
4.3.5	Code Reviews	14
4.3.6	Change	14
4.4	Issue 4 - Improve the description of the app on the Swagger UI docs - <i>Merged #482</i>	14
4.4.1	Issue	14
4.4.2	Communication with the team	16
4.4.3	Our work	16
4.4.4	Testing	16
4.4.5	Code Reviews	17
4.4.6	Change	17
4.4.7	Happy Moments!	19
4.5	Issue 5 - Optimise code for repeated code in ‘app/api/dao/user.py’ ‘get_user_dashboard’- <i>Accepted #418</i>	20
4.5.1	Issue	20

4.5.2	Communication with the team	21
4.5.3	Our work	21
4.5.4	Testing	22
4.5.5	Code Reviews	22
4.5.6	Change	23
4.6	Issue 6 - Change Test Structure for app/api/dao/task_comment.py - <i>Accepted</i> #553	24
4.6.1	Issue	24
4.6.2	Communication with the team	27
4.6.3	Our work	27
4.6.4	Testing	28
4.6.5	Code Reviews	28
4.6.6	Change	29
4.7	Issue 7 - Fix description messages on Backend Server - <i>Pending</i> #577	30
4.7.1	Issue	30
4.7.2	Communication with the team	31
4.7.3	Documentation	31
4.7.4	Our work	31
4.7.5	Testing	31
4.7.6	Code Reviews	31
4.7.7	Change	31
5	Our Legacy	33
5.1	Issue - Add only Python3 support in tests #595	33
6	Conclusion	35

Introduction

The methodology we used for this assignment can be split into two main parts. Firstly, in the selection of the open-source project we worked in a structured way, so we did research individually in projects of our interests and focused in specific programming languages. We categorized our findings and decided on the project. The second part is how we worked in the open-source project itself. We worked on already made issues and issues we identified. Each one of us was assigned to a specific issue, and he/ she undertook namely all the work on this issue, regarding branching, writing, communicating, opening a Pull Request and solving changes required using Github. For solving each of the issues mentioned we worked together, via Teamviewer, we agreed and implemented all the changes we would propose. All the changes we are suggesting and discussions are available on Github in the specific issues we are mentioning.

1 Project Selection Process

We utilized all the resources for finding an Open Source Project and we found more useful for us the site CodeTriage where we could explore different projects according to their programming language. Also, it was really important for us that the project we would choose would have an impact in the society. We organized the projects according to the frequency of updates, the variety of contribution from different nationalities, gender etc. and how often where pull requests accepted, issues generated, if there were issues for first-time contributors and the number of pull requests needing review versus the ones closed or accepted. These metrics helped us understand if a project was 'alive' and if it was interesting for us to work on it. We focused mainly in JAVA and Python.

We chose three of these OSS projects to install and build and also discussed with Mr Gkortzis our choices and their implementation. Due to operation system requirements (Windows) we disregarded one of them, Pretix, and focused on the other two. We successfully built Mentorship Backend and that's the project we worked on.

To sum up, the reasons why we chose Mentorship Backend are these:

- Python language
- Use of Google Python Style Guide,
- Guidance with resources for contributing and Commit Message Style Guide,
- Communication on a daily basis on zulip chat using different communication channels for each problem,
- Focus on issues for First Timers,
- Meaningful impact in mentoring women in tech,
- Interested in trying to contribute to the project also on the Android development part.

1.1 Troubleshooting

Of course, we were not able to build the projects immediately since we had some setup problems. We asked the community on Zulip to guide us through this process and if they knew how to solve our problems. Here is the questions we asked summarized:

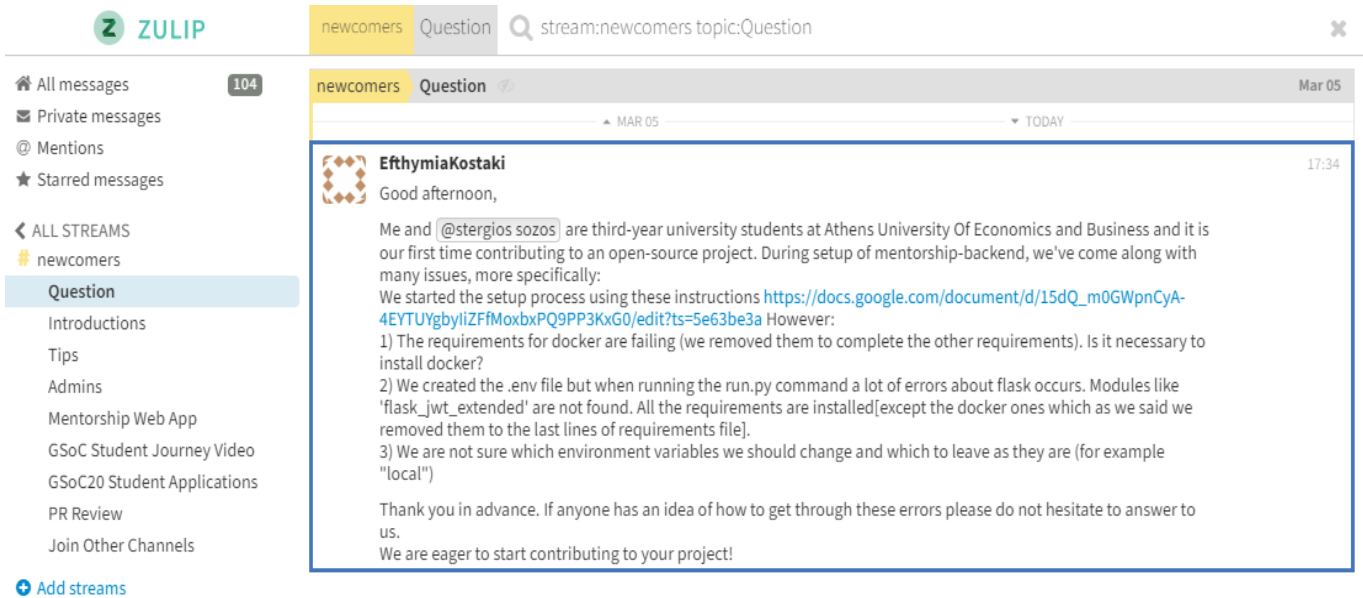



Figure 3: Setup Problems


After a vary insightful discussion with the community we were able to build the project and also share our insights in how we solved these problems if a new contributor wants to set up this project!



Isabel Costa - GSoC20 Admin

(EDITED)

17:49



EfthymiaKostaki

said:

Good afternoon,

Me and [stergios sozos](#) are third-year university students at Athens University Of Economics and Business and it is our first time contributing to an open-source project. During setup of mentorship-backend, we've come along with many issues, more specifically:

We started the setup process using these instructions https://docs.google.com/document/d/15dQ_m0GWpnCyA-4EYTUYgbylIZFfMoxbxPQ9PP3KxG0/edit?ts=5e63be3a However:

(...)

Thank you in advance. If anyone has an idea of how to get through these errors please do not hesitate to answer to us. We are eager to start contributing to your project!

@EfthymiaKostaki

1) The requirements for docker are failing (we removed them to complete the other requirements). Is it necessary to install docker?

Its not mandatory to install Docker. For me it helps a lot, because I don't have to install on my local machine all the dependencies


2) We created the .env file but when running the run.py command a lot of errors about flask occurs. Modules like 'flask_jwt_extended' are not found. All the requirements are installed[except the docker ones which as we said we removed them to the last lines of requirements file].

my idea is that, for having errors like flask_jwt_extended not found, it must be because requirements.txt was not installed. What is the environment you are running in? Windows, Linux, ...? Did you create a virtual environment and run pip as this: `pip install -r requirements.txt` ?

3) We are not sure which environment variables we should change and which to leave as they are (for example "local")

For environment variables I would leave local. So you dont have to worry with setting up a remote database, for now.


[\[Condense message\]](#)



stergios sozos

(EDITED)

18:27



stergios sozos

said:


Hello,

1)We tried installing docker but it requires Windows 10 pro, so we dont have that option unfortunately

2)We are using windows 10, and after creating the environment we install the requirments except for the docker


3)We left local as it is but we are also unsure about the rest of the vaiable sas its not explained what should we changem and if we should change it.

**We are activating the environment like this venv\Scripts\activate, as the source command is not found in windows



Isabel Costa - GSoC20 Admin

18:36



stergios sozos

said:

Hello,

1)We tried installing docker but it requires Windows 10 pro, so we dont have that option unfortunately

2)We are using windows 10, and after creating the environment we install the requirments except for the docker

3)We left local as it is but we are also unsure about the rest of the vaiable sas its not explained what should we changem and if we should change it.

1) Its unfortunate that Docker does not support windows home ([as per this message on docker forum](#))


2) Can you please create an issue on github (mentorship-backend), and put there screenshots and these details, in this way other people can see the issue and comment there if that is the case for them. Also for me to see screenshots and understand better what is the error you are getting and what you did, the steps you followed. Once you create the issue can you put the link here?

3) here's a guide on what each environmnet variable means: <https://github.com/anitab-org/mentorship-backend/wiki/Environment-Variables> isn't that clarifying?

@stergios sozos


YESTERDAY

TODAY



EfthymiaKostaki

10:13



EfthymiaKostaki

Hello thank you very much for your help! I am sending also how we managed to solve the problems following these instructions so if someone has the same issues with us can try what we did:

- Delete all docker requirements from requirements.txt
- Run again the command pip install for the requirements
- Run python run.py

Thanks again!




Figure 4: Setup Discussion

2 Mentorship Backend

Mentorship System is an application that matches women in tech to mentor each other, on career development, through 1:1 relations during a certain period of time. The project is deployed in Heroku in this link: <https://mentorship-backend-temp.herokuapp.com/>. The backend is based on a REST API. There is also the Android client of the app for creating the interface for the users. [3]

The repository has the following permanent branches:

- **master:** This contains the code which has been released.
- **develop:** This contains the latest code. All the contributing PRs must be sent to this branch. When we want to release the next version of the app, this branch is merged into the master branch.

3 Initial Communication with the community

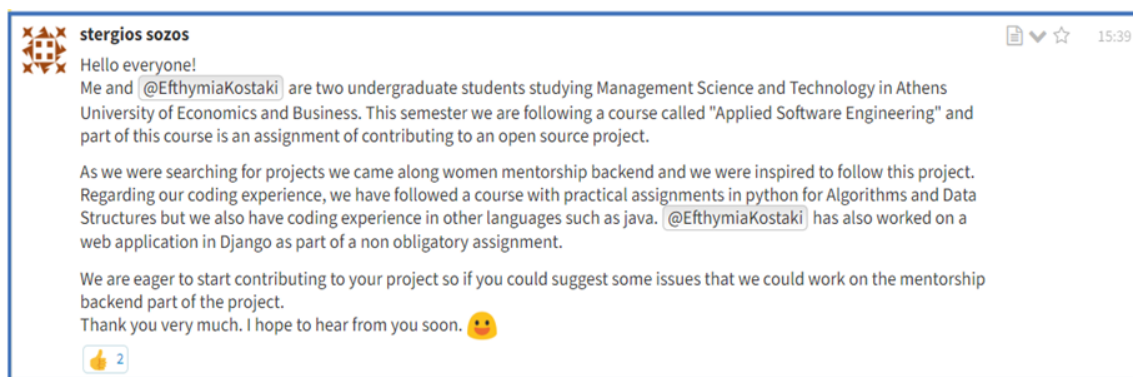


Figure 5: First Communication


4 Our contribution


Here is a full list of the parts of our contribution in a wide range of aspects such as quality assurance, user interface, code and testing.

4.1 Issue 1 - Create Quality Assurance table for Update Task API - *Merged* #473

4.1.1 Issue

Create Quality Assurance table for Update Task API #473

 **Closed** isabelcosta opened this issue on 3 Mar · 9 comments



isabelcosta commented on 3 Mar · edited ▾

Member

😊 ⋮

Description

As a developer,
I need to know what are the test cases for a certain API,
so that I can write new unit tests.

As a tester,
I need to know what are the test cases for a certain API,
so that I can perform manual tests on the API.

The task is to come up and add a quality assurance table to the docs in the `/docs` folder.

Acceptance Criteria

Update [Required]

☐ Add test cases for Update Task API


Definition of Done

☐ All of the required items are completed.
☒ Approval by 1 mentor.

Estimation

3 hours

Assignees

 StereiosSozos —unassign me

Labels

Category: Documentation/Training


First Timers Only

Milestone

No milestone

Linked pull requests

Successfully merging a pull request may close this issue.

 Create Quality Assurance table for ...

Notifications

Customize

🔊 Unsubscribe

You're receiving notifications because you were mentioned.

2 participants





Figure 6: Issue 473

8

4.1.2 Communication with the team

This issue was labeled for first-time contributors so we asked if we could work on this issue to understand the project better. The main maintainer was really helpful and answering our questions soon after we sent them. Firstly, we thought that new unit tests were required, combined with the quality assurance table, but after our communication it was clear to us that only the table had to be completed, as the tests already existed. While doing this table, by executing some tasks, we found out that there was a possible problem, and an issue had to be raised, as the maintainer agreed.

4.1.3 Our work

We worked in the default develop branch for this issue. We came to realize that this was a wrong strategy so in all other issues we created a specific branch for them. We tried to write meaningful messages in our commit messages always following the commit style guide suggested by the project. When we thought that the commit message was not understandable enough we wrote a body in our commit messages and documented further our change.

Regarding the way we created the table, firstly we found the file that we had to change, which was easy as the creator of the issue had already mentioned which one it is. Then the difficult part was understanding what exactly we had to write on the table and how to test it. After going through the test cases to take some ideas on what we could test on the backend server, we tried to create an account on the server. We used temp mail, as it was suggested, and created 4 profiles, so we could test all the possibilities we wanted. For example, we needed a user with a non-verified email, and one with verified. Then, we struggled a lot to find out the parameters-arguments that were needed for completing the task we wanted. For example a "token" was required but there was nowhere any explanation about this token. After some searching, we realised that when a user is logging in the server, the success message also provides this token. But even the way that we needed to insert the token was unusual, as we had to write "Bearer {token here}" Finally, when we had our users ready, and new how to use the backend server, we finally tested all the scenarios and write them down in the table.

4.1.4 Testing

Since this change was a change in the UI we did not need to run local testing or write new tests. Instead, our changes were visible in github, as the file was written in markdown language and that's how we verified that our changes were visible and correct. Also, we verified that the continuous integration testing in Travis CI was passing.

4.1.5 Code Reviews

Our pull request received immediate feedback and reviews. our first PR needed some important changes: 1) We were checking multiple failing variables, but only one was required each time. For example we were testing the result of the task with arg1 and arg2 causing the task to fail but arg1 was enough and a separate test for arg2 was required. 2) We were describing a variable/argument as "wrong" but we did not specify why it was wrong. For example, a user could not exist, or he could have not accepted the terms, or even not verify his email address. These 3 cases should be separate and specified. 3) We referred the users with pronouns and we had to make it general ("the user" instead of "he"). 4) We had to fix merge conflicts as the PR was approved later. Also, other contributors viewed our changes and accepted them.

4.1.6 Change

4.2 Issue 2 -Fix description messages on Mentorship Relation - *Merged* #554

4.2.1 Issue

Fix description messages on Mentorship Relation #554

[Edit](#)[New issue](#)

StergiosSozos opened this issue 28 days ago · 5 comments



StergiosSozos commented 28 days ago · edited

Contributor



Description

As a developer, i need to see the same way for displaying messages on the backend server. so that i can follow the same pattern on the messages im writing. Right now the messages are shown in 2 ways:

Code	Description
200	Task comment was updated successfully.
400	Comment field is missing. Comment must be in string format. The comment field has to be shorter than 400 characters. This mentorship relation status is not in the accepted state. You have not created the comment and therefore cannot modify it.
401	The token has expired! Please, login again or refresh it. The token is invalid! The authorization token is missing! You are not logged in this mentorship relation.
404	User does not exist. Mentorship relation does not exist. Task does not exist. Task comment does not exist. Task comment with given task id does not exist.

Code	Description
400	{'message': 'User is already an Admin.'}
401	{'message': 'The token has expired! Please, login again or refresh it.'} {'message': 'The token is invalid.'} {'message': 'The authorization token is missing.'}
403	{'message': 'User is now an Admin.'}
404	{'message': 'User does not exist.'}

Assignees

StergiosSozos —unassign me

Labels

Category: Coding

Category: Documentation/Training

Category: User Interface

Milestone

No milestone

Linked pull requests

Successfully merging a pull request may close this issue.

[Fix Swagger UI http response docu...](#)

Notifications

[Customize](#)

Unsubscribe

You're receiving notifications because you were assigned.

Figure 7: Issue 554-1

- 4.2.2 Communication with the team
- 4.2.3 Our work
- 4.2.4 Testing
- 4.2.5 Code Reviews
- 4.2.6 Change
- 4.3 Issue 3 - ‘Complete task‘ not including all the necessary information when request id is not in accepted mode - *Accepted* #537
 - 4.3.1 Issue

"Complete task" not including all the necessary information when request id is not in accepted mode #537

Edit

New issue

Open

StergiosSozos opened this issue on 12 Apr · 5 comments · May be fixed by #549



StergiosSozos commented on 12 Apr

Contributor



Description

As a user,when i try to complete a task I need to know what I did wrong and why what I expected did not happen.

I observed that when a request id is not accepted and a user tries to complete a task (which does not exist, as a task cannot be created) the message is about the wrong task, but it should show the initial problem too, that the relationship is not accepted.

To Reproduce

Steps to reproduce the behavior:

1. Send a relationship request from user1 to a user2
2. Do not accept that request from user2
3. Try to complete a task (the number will be random as a task cannot be created)
4. See the error message, about the task not existing, but not about the relationship not having been accepted

Expected behavior

The message should inform the user trying to complete the task that the relationship is not accepted too.

Assignees

StergiosSozos —unassign me

Labels

Category: Coding

Type: Enhancement

Milestone

No milestone

Linked pull requests

Successfully merging a pull request may close this issue.

fix: Return error message when rela...

Notifications

Customize

Unsubscribe

You're receiving notifications because you were assigned.

Figure 8: Issue 537

4.3.2 Communication with the team

We had already talked about this issue with the maintainer in a PR, so after the maintainer's approval, we opened a new issue, and got assigned to it. We also clarified, that the solution suggested on this issue, is giving higher priority to a specific argument, which was the state of the relationship status, instead of the one that was then highest, which was the existence of the task.

4.3.3 Our work

We added an if statement before the existings one, so that it will be prioritized, and included the new possible responses on the responses board on the backend server. After that we changed the hardcoded status (i.e. 400,401 etc) to http status. Then, we changed the way messages were printed into f strings and added some blank spaces that were needed. Finally, we wrote a unit test for the added if statement.

After the original

4.3.4 Testing

Initially, we didn't have any unit testing for the specific change, and we tested it only on the backend server, and we also confirmed that the continuous integration testing in Travis CI was passing. After our PR was reviewed and after all the changes requested had been done, an additional test was asked, a unit test. To create this, we needed to also add a user to the database and create a new relation of this user with another user.

4.3.5 Code Reviews


The main part of the PR was approved, and only some minor changes were asked, regarding new standards(http status instead of hardcoded numbers, f strings for printing messages) and a blank line to separate visually validation sections. After these minor changes, a unit test was asked.

4.3.6 Change



4.4 Issue 4 - Improve the description of the app on the Swagger UI docs - *Merged #482*

4.4.1 Issue

This issue was referring to the description of the app which the developers view if they have cloned the repository locally or in the deployed app remotely in Heroku.



isabelcosta commented on 6 Mar • edited ▾

Member  

Description

As a contributor,
I need some more description on Swagger docs,
so that I can easily understand what the app is about.

Maybe you could add the link to the wiki, and the full description of the app, link to Android app.
These are just ideas.. I'm not sure what is the best, but I know this can definitely contain some more information.

Mocks

I tried this out myself, staying something like:

Mentorship System API ⁴²⁹

[Base URL: /]
<http://localhost:3000/swagger.json>

API documentation for the backend of Mentorship System blah blah
<https://github.com/isabel-org/mentorship-android> blah

default	Default namespace	>
Users	Operations related to users	>
Admins	Operations related to Admin users	>
Mentorship Relation	Operations related to mentorship relations between users	>
Models		>

Figure 9: Issue 482

4.4.2 Communication with the team

This issue was labeled for first-time contributors so we asked if we could work on this issue to understand the project better. The main maintainer was really helpful and answering out questions soon after we sent them. We realized while working on the issue that there was an issue with the namespaces since there was a default namespace which was not referring to any API. We decided to ask about this if this was intentional or it was a problem in the UI. In figure shown above this problem is visible in the ‘default namespace’ resource. Isabel (the main maintainer and creator of the project) told us that this was really an issue and that she did not know how to solve it. She suggested that if we solve this problem we can commit the change in the current pull request so as to further expand it so it includes the solution of this problem as well.

4.4.3 Our work

We worked in the default develop branch for this issue. We came to realize that this was a wrong strategy so in all other issues we created a specific branch for them. We tried to write meaningful messages in our commit messages always following the commit style guide suggested by the project. When we thought that the commit message was not understandable enough we wrote a body in our commit messages and documented further our change.

Regarding the initial issue, it turned out to be hard for us to understand which part on the project we should change in order for our changes to be visible in the User Interface. We used Atom, the text editor, to find the parts of the software which had the initial description we wanted to change. They were two, the swagger.json and the api_extension.py file. Since we did not know which was the correct file we should change we decided to change both of them. The correct one turned out to be the api_extension.py file. We were able to view our changes locally.

Regarding the new part of the issue we identified and proposed, we needed to understand where the problem was coming from. It could have been from the call inside the specific resources or in some other part of the project. We searched a lot and found the documentation of the flask_restplus library which was used in the file we wanted to change. We found the method Api() and which parameters it could include and we were able to change the arguments and change the name of the ‘default namespace’. The documentation however, did not help us fix the actual issue which was to remove this resource. We searched again and found out that this library was actually open-source. In the issues of the Github repository we were able to find an issue about the same problem we had and a suggested solution to it. By implementing this code we were able to solve the issue.

4.4.4 Testing

Since this change was a change in the UI we did not need to run local testing or write new tests. Instead, our changes were visible in the localhost in our

browser and that's how we verified that our changes were visible and correct. Also, we verified that the continuous integration testing in Travis CI was passing.

4.4.5 Code Reviews

Our pull request received immediate feedback and reviews. Since the change was in the User Interface, Isabel asked us to include a screenshot of the change proposed in the description. Also, other contributors viewed our changes and accepted them.

4.4.6 Change

The merged change is this:

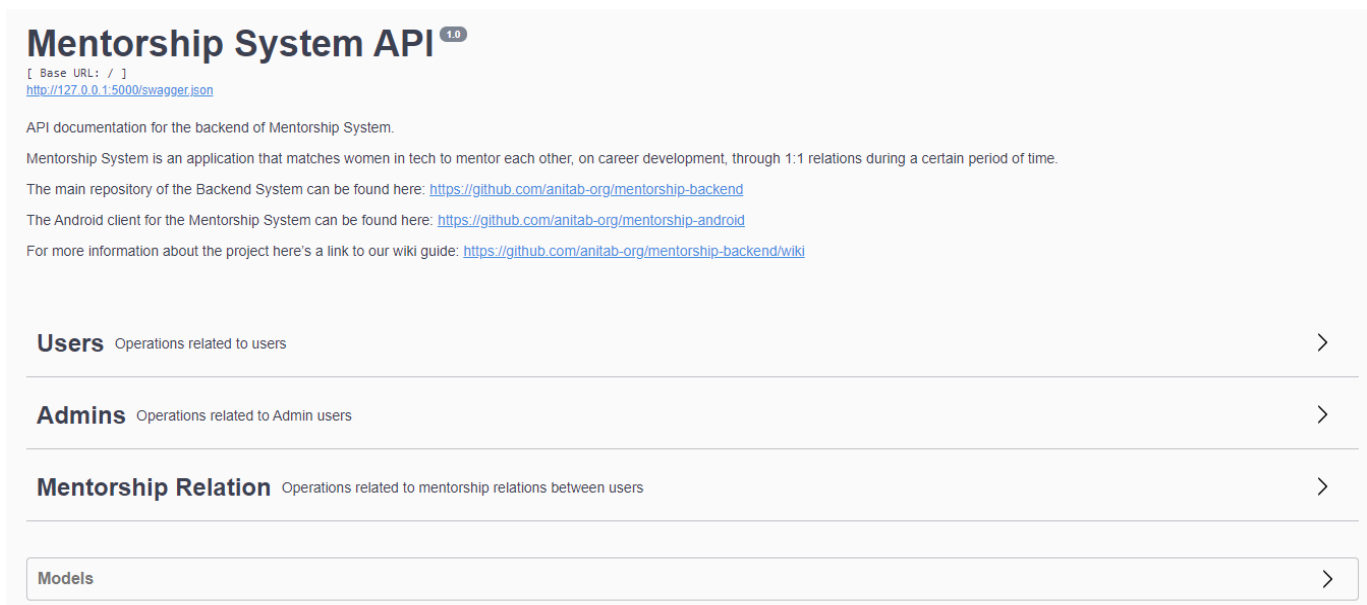
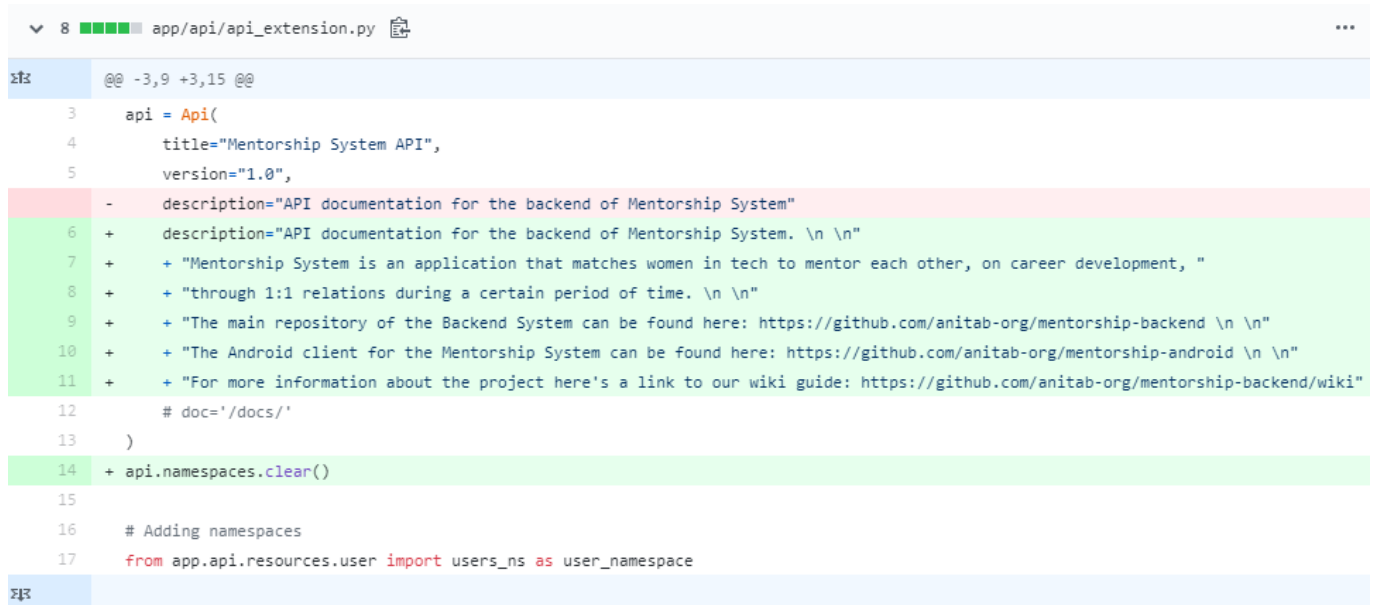


Figure 10: Change in the UI

This change required to change these files and add new code:



```
@@ -3,9 +3,15 @@
3     api = Api(
4         title="Mentorship System API",
5         version="1.0",
6         description="API documentation for the backend of Mentorship System"
7     +     description="API documentation for the backend of Mentorship System. \n \n"
8     +     + "Mentorship System is an application that matches women in tech to mentor each other, on career development, "
9     +     + "through 1:1 relations during a certain period of time. \n \n"
10    +     + "The main repository of the Backend System can be found here: https://github.com/anitab-org/mentorship-backend \n \n"
11    +     + "The Android client for the Mentorship System can be found here: https://github.com/anitab-org/mentorship-android \n \n"
12    +     + "For more information about the project here's a link to our wiki guide: https://github.com/anitab-org/mentorship-backend/wiki"
13    +     # doc='/docs/'
14    + )
15    + api.namespaces.clear()
16
17    # Adding namespaces
18    from app.api.resources.user import users_ns as user_namespace
```

Figure 11: Code change

4.4.7 Happy Moments!

The creator and main maintainer of the project congratulated us for our effort and change in the UI which urged us to find other aspects of the software in which we could contribute.

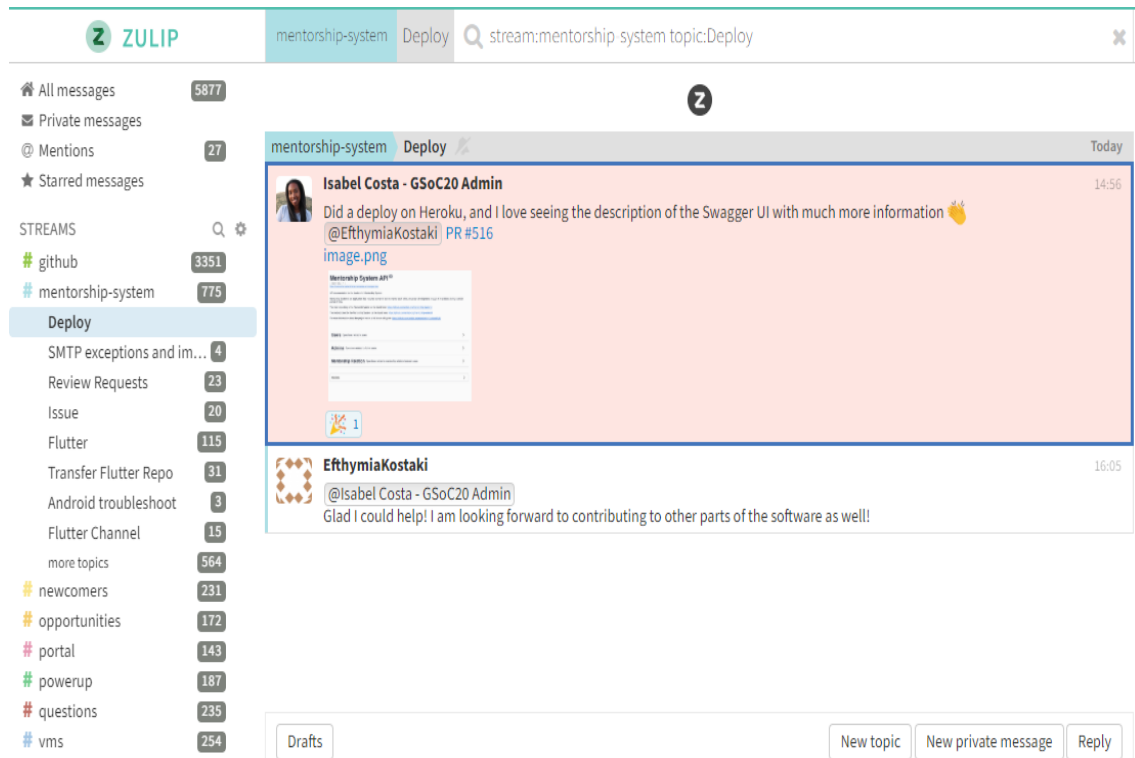


Figure 12: Happy Moments!

4.5 Issue 5 - Optimise code for repeated code in 'app/api/-dao/user.py' 'get_user_dashboard'- *Accepted* #418

4.5.1 Issue

This issue required simplifying a method which was around 200 lines in to less. We successfully removed 70 lines with repeated code.

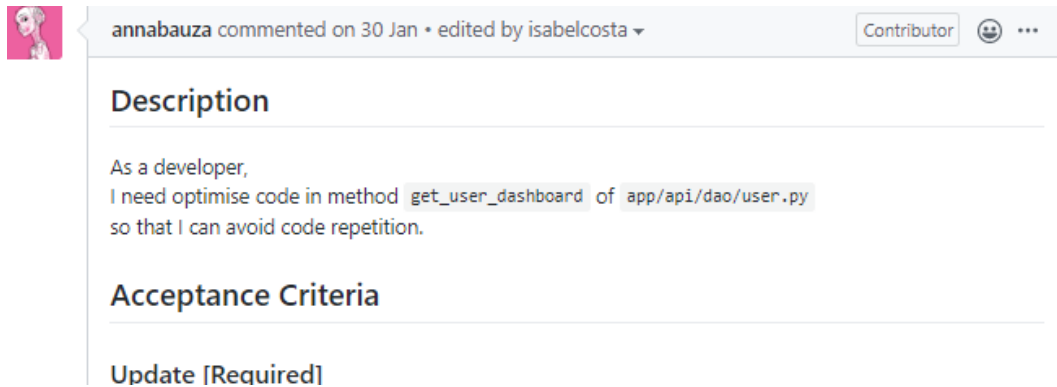


Figure 13: Issue 418

4.5.2 Communication with the team

It was crucial for us to understand how we could verify that our changes were correct and that the project was having the same functionality as before. So we asked in the issue how we could verify that or if it was necessary to write some new unit tests. The maintainer told us that we should just verify that the unit tests are working as before.

4.5.3 Our work

We created a specific branch for this issue first locally and then add this branch to our remote repository. We tried to write meaningful messages in our commit messages always following the commit style guide suggested by the project. When we thought that the commit message was not understandable enough we wrote a body in our commit messages and documented further our change.

Firstly, we realized that the `as_mentor` and `as_mentee` dictionaries were exactly the same and that we could delete the initialization of one and copy it to the other one. Since, these dictionaries had also subdictionaries we found out we should not copy them using the traditional `copy()` method since in some occasions when changing the subdictionary in the one dictionary could actually change the other one as well. To solve this problem, we found out the copy library which offers a method named `deepcopy()` which could solve our problem. We used this method and successfully deleted 15 lines of repeated code.

The second part of our work was to create a method which could remove further the repeated code. All the dictionary was assesed this way for more than 100 lines:

```
as_mentee["received"]["completed"] = [
    relation.response
    for relation in mentee_received_relations
    if relation.state == MentorshipRelationState.COMPLETED
]
as_mentee["received"]["cancelled"] = [
    relation.uresponse
    for relation in mentee_received_relations
    if relation.state == MentorshipRelationState.CANCELLED
]
```

After we understood which was the problem we tried to experiment and decide which parameters should this method have. We decided that these two were enough. First, the `mentor_or_mentee_received_or_sent` which would include the relations of the mentee or the mentor and if these relations are received or sent and the `mentorship_relation_state_wanted` which could take one of these values:

- PENDING
- CANCELLED

- COMPLETED
- REJECTED

This method is static as they are all the other methods in this file and has a for loop similar to the initial one taking use of all the arguments and returns a list to be added in each subdictionary.

4.5.4 Testing

Since the maintainer told us that we do not need to write unit tests about our change we verified that all the local testing was passing and of course that the continuous integration testing in Travis CI was passing.

4.5.5 Code Reviews

Unfortunately, this pull request has not been reviewed yet but we asked the main maintainer to do that as soon as possible. She said that she does not have much bandwidth and will try get back to us in that as soon as possible.

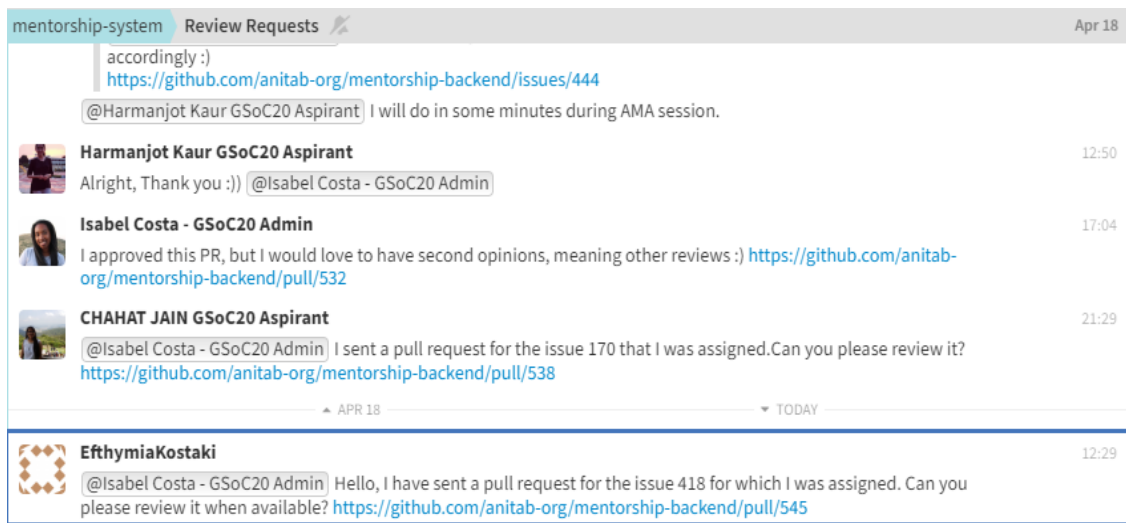


Figure 14: Issue 418 - Ask for review

4.5.6 Change

Our proposed change is this:

```
@staticmethod
def get_state(mentor_or_mentee_received_or_sent ,
              mentorship_relation_state_wanted):
    value = [relation.response
              for relation in mentor_or_mentee_received_or_sent
              if relation.state == mentorship_relation_state_wanted]
    return value

as_mentor = copy.deepcopy(as_mentee)
as_mentee["received"]["accepted"] =
    UserDAO.get_state(mentee_received_relations ,
                      MentorshipRelationState.ACCEPTED)
as_mentee["received"]["rejected"] =
    UserDAO.get_state(mentee_received_relations ,
                      MentorshipRelationState.REJECTED)
as_mentee["received"]["completed"] =
    UserDAO.get_state(mentee_received_relations ,
                      MentorshipRelationState.COMPLETED)
as_mentee["received"]["cancelled"] =
    UserDAO.get_state(mentee_received_relations ,
                      MentorshipRelationState.CANCELLED)
as_mentee["received"]["pending"] =
    UserDAO.get_state(mentee_received_relations ,
                      MentorshipRelationState.PENDING)
as_mentor["received"]["accepted"] =
    UserDAO.get_state(mentor_received_relations ,
                      MentorshipRelationState.ACCEPTED)
as_mentor["received"]["rejected"] =
    UserDAO.get_state(mentor_received_relations ,
                      MentorshipRelationState.REJECTED)
as_mentor["received"]["completed"] =
    UserDAO.get_state(mentor_received_relations ,
                      MentorshipRelationState.COMPLETED)
as_mentor["received"]["cancelled"] =
    UserDAO.get_state(mentor_received_relations ,
                      MentorshipRelationState.CANCELLED)
as_mentor["received"]["pending"] =
    UserDAO.get_state(mentor_received_relations ,
                      MentorshipRelationState.PENDING)
as_mentee["sent"]["accepted"] =
    UserDAO.get_state(mentee_sent_relations ,
                      MentorshipRelationState.ACCEPTED)
as_mentee["sent"]["rejected"] =
```

```

        UserDAO.get_state(mentee_sent_relations,
        MentorshipRelationState.REJECTED)
as_mentee["sent"]["completed"] =
        UserDAO.get_state(mentee_sent_relations,
        MentorshipRelationState.COMPLETED)
as_mentee["sent"]["cancelled"] =
        UserDAO.get_state(mentee_sent_relations,
        MentorshipRelationState.CANCELLED)
as_mentee["sent"]["pending"] =
        UserDAO.get_state(mentee_sent_relations,
        MentorshipRelationState.PENDING)
as_mentor["sent"]["accepted"] =
        UserDAO.get_state(mentor_sent_relations,
        MentorshipRelationState.ACCEPTED)
as_mentor["sent"]["rejected"] =
        UserDAO.get_state(mentor_sent_relations,
        MentorshipRelationState.REJECTED)
as_mentor["sent"]["completed"] =
        UserDAO.get_state(mentor_sent_relations,
        MentorshipRelationState.COMPLETED)
as_mentor["sent"]["cancelled"] =
        UserDAO.get_state(mentor_sent_relations,
        MentorshipRelationState.CANCELLED)
as_mentor["sent"]["pending"] =
        UserDAO.get_state(mentor_sent_relations,
        MentorshipRelationState.PENDING)

```

4.6 Issue 6 - Change Test Structure for app/api/dao/-task_comment.py - *Accepted* #553


4.6.1 Issue

We decided to take a look on the testing part and see if we could write new unit tests or fix any errors. We saw that the project did not have the codecoverage on the main repository so we could knw which classes were not tested enough so we decided to use the package coverage.py locally to find the coverage.

mentorship-system

Issue

Apr 26




Priyansh Kedia- GSoC20 Aspirant

Also @Isabel Costa - GSoC20 Admin what do you think about adding it to the wiki page?
What do others think about this?

10:31

▲ APR 26

▼ TODAY



EfthymiaKostaki

Hello, everyone ! I hope you are doing well during this difficult time.

I used coverage. py program to check the coverage of the project and I saw for the backend is 34%. Though many parts of the coverage of the project are based on the library folder, I would be really interested in working on new unit tests for this project.
@Isabel Costa - GSoC20 Admin Do you think I could open an issue for testing in which I could work on?

Thanks in advance!

Figure 15: Coverage Discussion

The community agreed that this was a really good idea to focus more on tests in this project and to write new tests or implement changes in the existing ones. They also wanted help into adding the badge of codecoov but since we did not have an account there and it was not our repository we decided that we could not help with that.

What we did find out what-so-ever was that the structure of all the tests was not consistent that's why we proposed this issue:

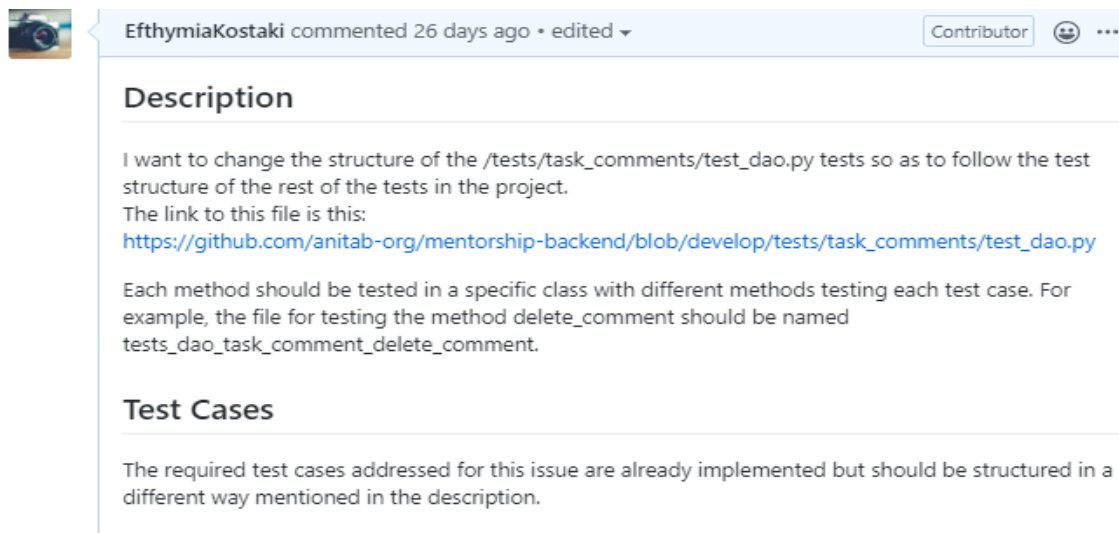


Figure 16: Issue 553

4.6.2 Communication with the team

Since the issue was suggested by us we were waiting to hear the response from the main maintainer Isabel about this issue.

The main maintainer Isabel wrote that she believes that there are issues with higher priority but we could go ahead and work on this issue and propose a solution and how we believe it solves the problem. We thought it would be better to implement the change and propose the final version of it instead of submitting a Work In Progress Pull Review so the maintainers could check the whole width of our change.

4.6.3 Our work

We created a specific branch for this issue first locally and then add this branch to our remote repository. We tried to write meaningful messages in our commit messages always following the commit style guide suggested by the project. When we thought that the commit message was not understandable enough we wrote a body in our commit messages and documented further our change.

The problem stemmed from the fact that all the testing of the project was not following the same structure. In some cases the code included in each folder test for base setup and in others not, in others a method was tested in a specific file and in others not, so we thought it was crucial to implement a change which would expand the consistency of the project. We decided that the best strategy for changing the given file was this:

- Write a base setup for `tests/task_comments/test_dao.py`
- Split `tests/task_comments/test_dao.py` into four different files. In each one, every method would be tested separately.

To address the first bullet, we needed to understand how a base setup is created and how it is used. We wanted to understand how the setup method is used and how it should be structured to serve our needs. Since, the `create_task_comment` method was used in all the methods we decided that this was the method which should be used for the setup to avoid code repetition. In order to save each task comment we had to add it in the session of the database and also to commit in the database. In this way, the base setup class which created a new Test Case and all the data, could be used in the unit testing classes we implemented.

For the second bullet, we split the test methods into four different files, one for each method:

- `test_dao_create_task_comment.py`
- `test_dao_delete_comment.py`
- `test_dao_modify_comment.py`
- `test_dao_find_by_task_id.py`

4.6.4 Testing

Testing our addition in this case was harder than in previous points since we were altering the testing of the project. Firstly, we did not delete the class we were altering yet, so we saw the counter of how many tests were already running before our addition. Then, we included our tests and saw if all our tests were passing the testing locally. All the tests were successfully passing in the command line but we wanted to double-check so we run the `test_task_comments` folder separately through PyCharm. Once we were sure that all our changes were passing we deleted the initial file, committed and created the Pull Request.

4.6.5 Code Reviews

In the beginning our change received mixed reviews. Of course, there were suggested some minor changes to the code such as to remove unnecessary imports and repetitive code (which turned out to be from the initial testing) but also the argument that this change was not needed right now since there was not a problem.

Regarding the minor changes we addressed them all, since there were important for the correctness of our change. One suggested change about the repeated code turned out to have been from the primary testing so we thought that even though it did not seem so, that it was needed in the project. Turns out it was not so we fixed it.

Regarding the major arguments about the idea of the change and if it was really needed at this point we analyzed our thinking and said that:

- We started working on this project taking into consideration **the structure of the tests** in the other parts of the software. We saw that tasks and mentorship relation folder was structured like the solution we were proposing.
- Of course, we saw in some other parts of the project in the testing this structure wasn't followed, but the main idea was the **consistency** of the testing material so as everyone that needs to write some testing (for example to test a new method) will immediately understand where he/she should do so.
- **Expandability of the testing.** We think this may overlap with the previous point but for example, if we realize at some point that we need on the task comments further testing on the DAO part or if new functionality is added then, according to the other parts of the testing we highlighted in point 1, we shouldn't use the same class for testing the same method more than once and also test the other methods.
- **Base test for setup** We saw that all the methods in `task_comments_dao` were using the same method for the setup that is why we created a test base for the task comments so to avoid code repetition. Our thinking is that the setup test base should be further expanded so the `task_comments_api`

classes who test each method use this as well since they all have a setup and some common code. With the change suggested in this pull request, us or someone else interested would be able to intergrade these changes we just mentioned so that the test base covers both the API and the DAO part of the task comments.

The maintainer Sanket, who had his doubts about our change after our discussion wrote that he accepts our change and also suggested some change about supporting Python2 in our code. We altered our code and implemented this change but turns out that other parts of the testing still support Python2. We opened another issue for that-namely issue 8.

4.6.6 Change

As our change involves the same tests as before we will write in this report only the Test Case we wrote since all the other changes can be found on Github. This file is included in: tests/task_comments/task_comments_base_setup.py

```
1 + from tests.tasks.tasks_base_setup import TasksBaseTestCase
2 + from app.database.sqlalchemy_extension import db
3 + from app.api.dao.task_comment import TaskCommentDAO

This conversation was marked as resolved by EfthymiaKostaki Show conversation

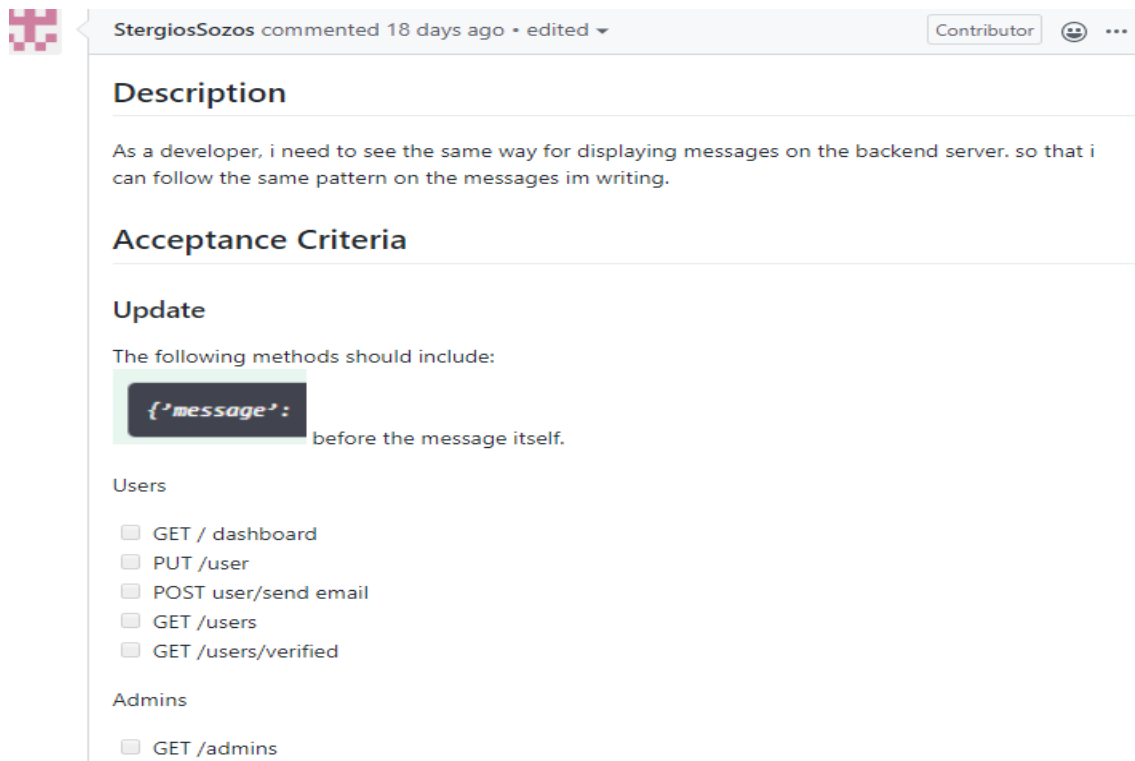
4 +
5 + class TaskCommentsBaseTestCase(TasksBaseTestCase):
6 +     def setUp(self):
7 +         super().setUp()
8 +
9 +         dao = TaskCommentDAO()
10 +         self.comment= dao.create_task_comment(user_id=1, task_id=1, relation_id=2,
comment="comment")
11 +         self.task_comment = TaskCommentDAO.get_task_comment(1, 1)[0]
12 +         db.session.add(self.task_comment)
13 +         db.session.commit()
```

Figure 17: Issue 553 Change

4.7 Issue 7 - Fix description messages on Backend Server - *Pending* #577

Similar to Issue 2 but addressed the whole project namely the resources Admins and Users.

4.7.1 Issue



The screenshot shows a GitHub issue page for issue #577. At the top, a comment by StergiosSozos is visible, dated 18 days ago. The issue title is "Issue 7 - Fix description messages on Backend Server - Pending #577". The description states: "As a developer, i need to see the same way for displaying messages on the backend server. so that i can follow the same pattern on the messages im writing." The acceptance criteria section is empty. The update section states: "The following methods should include:" followed by a code snippet:

```
{ 'message': }
```

 before the message itself. Below this, there are two sections: "Users" and "Admins". The "Users" section lists five methods with checkboxes: GET / dashboard, PUT /user, POST user/send email, GET /users, and GET /users/verified. The "Admins" section lists one method with a checkbox: GET /admins.

StergiosSozos commented 18 days ago • edited ▾ Contributor 😊 ...

Description

As a developer, i need to see the same way for displaying messages on the backend server. so that i can follow the same pattern on the messages im writing.

Acceptance Criteria

Update

The following methods should include:

```
{ 'message': }
```

before the message itself.

Users

- ☐ GET / dashboard
- ☐ PUT /user
- ☐ POST user/send email
- ☐ GET /users
- ☐ GET /users/verified

Admins

- ☐ GET /admins

Figure 18: Issue 577

4.7.2 Communication with the team

Since we were the ones who created the issue we did not want to start working in it before we were assigned. We asked siddhipandare if we could be assigned to this issue and said that yes we could start working on it.

4.7.3 Documentation

This change required a change in the documentation of the resources. This was labeled in the issue as a label by the maintainers.

4.7.4 Our work

Since, we were already familiar in the way this issue should be solved we utilized the knowledge from issue 2 to correct the messages in the resources of Users and Admins, change to fstrings and create new specific messages in the messages.py. For more specifics on how we worked on that issue refer to issue 2.

4.7.5 Testing

- While running the local server and checking the UI the changes are visible.
- All Travis CI tests have passed

4.7.6 Code Reviews

Unfortunately, this review has not received a code review yet probably because of the limit of the bandwidth of the maintainers.

4.7.7 Change

We changed three files so we present here only our changed code.

- app/api/resources/admin.py

```
@admin_ns.response(HTTPStatus.OK,
    f"{messages.GENERAL_SUCCESS_MESSAGE}" ,
    public_admin_user_api_model)
@admin_ns.doc(
    responses={
        HTTPStatus.UNAUTHORIZED: f"{messages.TOKEN_HAS_EXPIRED}<br>"
        f"{messages.TOKEN_IS_INVALID}<br>"
        f"{messages.AUTHORISATION_TOKEN_IS_MISSING}"
    }
)
```
- app/api/resources/user.py

```

@users_ns.doc(
  responses={
    HTTPStatus.UNAUTHORIZED:
      f" {messages.TOKEN_HAS_EXPIRED}<br>"
      f" {messages.TOKEN_IS_INVALID}<br>"
      f" {messages.AUTHORISATION_TOKEN_IS_MISSING}"
  }
)

@users_ns.response(HTTPStatus.CREATED,
  f" {messages.GENERAL_SUCCESS_MESSAGE}"
  , public_user_api_model)

@users_ns.response(HTTPStatus.BAD_REQUEST,
  f" {messages.INVALID_INPUT}")

@users_ns.doc(
  responses={
    HTTPStatus.UNAUTHORIZED:
      f" {messages.TOKEN_HAS_EXPIRED}<br>"
      f" {messages.TOKEN_IS_INVALID}<br>"
      f" {messages.AUTHORISATION_TOKEN_IS_MISSING}"
  }
)

@users_ns.response(HTTPStatus.OK,
  f" {messages.EMAIL_VERIFICATION_MESSAGE}")
@users_ns.response(HTTPStatus.BAD_REQUEST,
  f" {messages.INVALID_INPUT}")
@users_ns.response(HTTPStatus.FORBIDDEN,
  f" {messages.USER_ALREADY_CONFIRMED_ACCOUNT}")
@users_ns.response(HTTPStatus.NOT_FOUND,
  f" {messages.USER_DOES_NOT_EXIST}")

@users_ns.response(HTTPStatus.OK, f" {messages.SUCCESSFUL_REFRESH}" ,
  refresh_response_body_model)

@users_ns.response(HTTPStatus.OK, f" {messages.SUCCESSFUL_LOGIN}" ,
  login_response_body_model)

@users_ns.response(HTTPStatus.OK, f" {messages.SUCCESSFUL_RESPONSE}" ,
  home_response_body_model)

@users_ns.response(HTTPStatus.OK, f" {messages.GENERAL_SUCCESS_MESSAGE}" ,
  dashboard_response_body_model)

```



```
@users_ns.response(HTTPStatus.NOT_FOUND,  
                    f"{messages.USER_NOT_FOUND}")
```

- app/messages.py

```
INVALID_INPUT = {"message": "Invalid input."}  
GENERAL_SUCCESS_MESSAGE = {  
    "message": "Success."  
}  
  
SUCCESSFUL_REFRESH = {  
    "message": "Successful refresh."  
}  
  
SUCCESSFUL_RESPONSE = {  
    "message": "Successful response."  
}  
  
SUCCESSFUL_LOGIN = {  
    "message": "Successful login"  
}
```

5 Our Legacy

5.1 Issue - Add only Python3 support in tests #595



EfthymiaKostaki commented 11 days ago • edited ▼

Contributor



Description

I was working on an issue to alter the test structure of the project and discussing with a maintainer we found out that the project should only support python 3 but in some tests, this is not achieved. For example, in `tasks_base_setup` this is happening:

```
class TasksBaseTestCase(BaseTestCase):  
  
    # Setup consists of adding 2 users into the database  
    # User 1 is the mentorship relation requester = action user  
    # User 2 is the receiver  
    def setUp(self):  
        super(TasksBaseTestCase, self).setUp()
```

The setup method should be written like this

```
super.setUp()
```

Acceptance Criteria

- ☐ Refactor all the test bases which support Python2 to Python3 as presented
- ☐ All local testing passing

Figure 19: Issue 595

6 Conclusion

Contributing to an open-source project enriched our understanding of program development and urged us to try to establish possible improvements. We were able to express our opinion with facts and use the terminology we learned during the course. We researched a lot with or without the guidance of the open-source community and offered feasible solutions. In the future, we want to continue contributing to other open-source projects.

References

- [1] <https://github.com/anitab-org>
- [2] <https://github.com/anitab-org/mentorship-android>
- [3] <https://github.com/anitab-org/mentorship-backend>