Time & Project Management

Yanran Wang

H18A Team Fudge

LECTURES ON TIME

PROJECT TASKS ON TIME

BLOG POSTS ON TIME

WHAT'S IN THIS PART?

- Lectures & Tutorials
- Project Tasks
- Afterclass Learning
- Reflection

99%

TUTORIAL ATTENDANCE!

*miss 1 due to travel

Project tasks

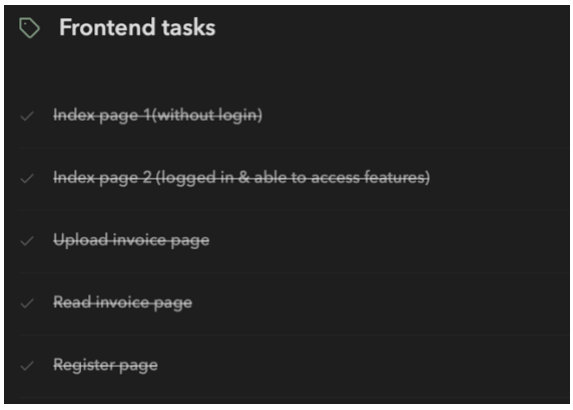
- **All lectures finished on time and attended most tutorials**

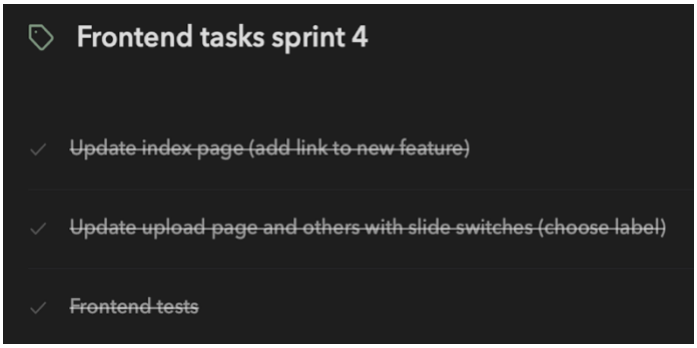
  Summary: always highlight important lectures and review based on project needs.

- **Always have specific planning before project tasks**

  Summary:
  1. For project part we always start the sprint with planning and evaluation.
  2. For myself, at the start of the term I usually started to write code first, but later I started to use todo list to manage my tasks, which is helpful with efficiency.

  [Project planning confluence page]

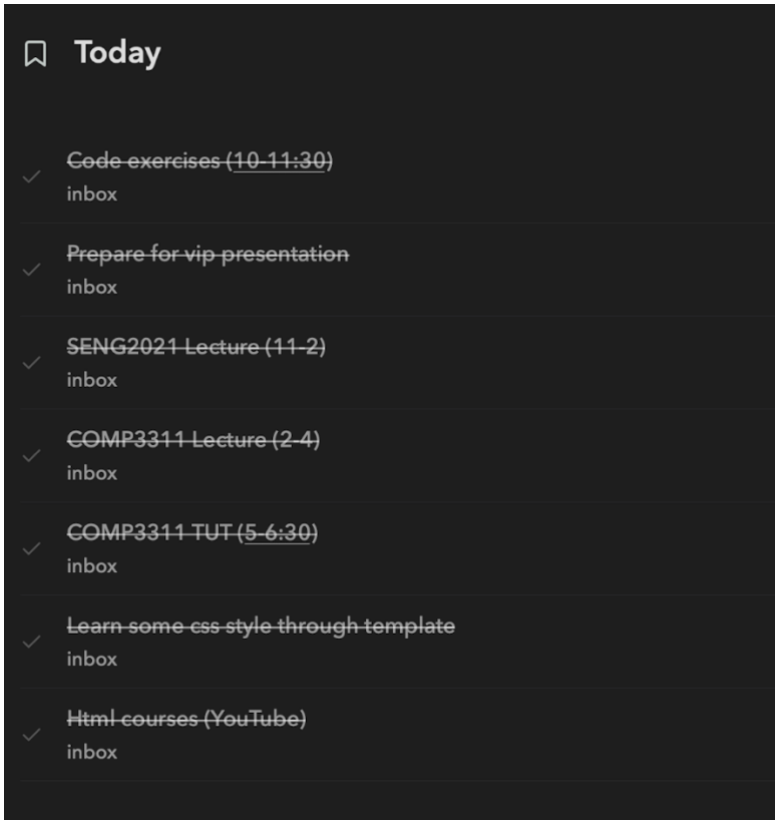




Afterclass Learning

- **regularly learn new knowledge based on project tasks (most frontend)**

  Summary: did research on backend & frontend learning at least **three times a week**. only do handwriting memos before but always made me feel confused without proper explanation, so I changed to **daily schedule** later.



Reflection

- **Time management is a challenge for the whole term**

  It's hard to find balance between lots of commitments during the whole term. I think my problem for this part is that I didn't make clear of all my courses' commitments, and this made me sometimes feel too stressful over the term. Fortunately, I still managed to find the balance. In the future I would like to do more research before enrolment.

- **It's better to use clear lists for tasks allocation**

  In the beginning I used hand-writing memos, however sometimes they couldn't remind me of my tasks, and I would forget about them, so I changed later to todo lists with alarm, which is helpful with finishing tasks on time.

Yanran Wang

H18A Team Fudge

CONTRIBUTIONS TO GROUP

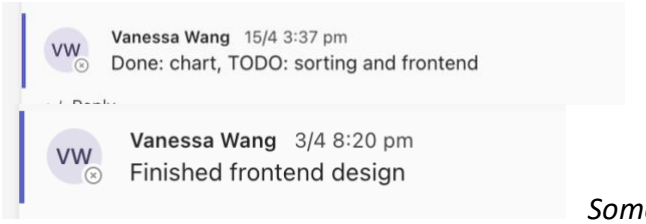
BLOG REPLY & DISCUSSION

WHAT HAVE I DONE?

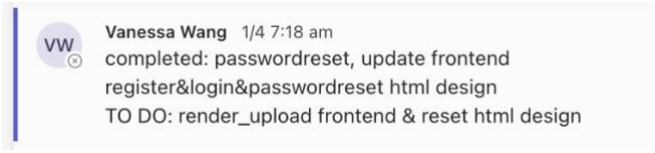
- Regular discussion with teammates using both Teams & Messenger
- Encourage others on the blog and ask questions

- **regular discussion with Teams and Messenger**

  Summary: my teammates and I always had regular stand-ups about our tasks, and we split into two subteams, which is helpful with task allocation.



Some standup screenshots



- **help group ask question on forum**

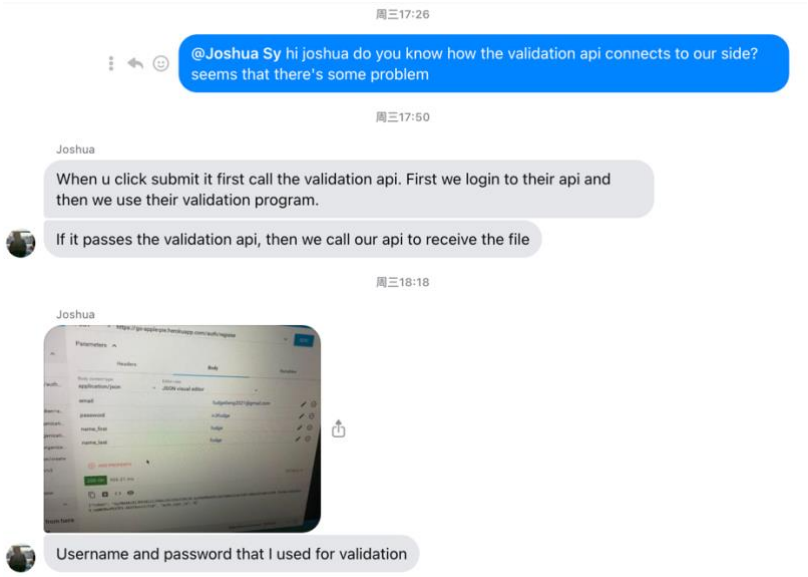
  Summary: asked many questions regarding to group's problems

  Blog links:

  1. Project Planning
  2. Communication Report
  3. Persistence Layer
  4. CI pytest
  5. Data Modelling
  6. Using other's API on Swagger
  7. CORS problem
  8. Late tutorial feedback problem

- **tackle problems with teammates**

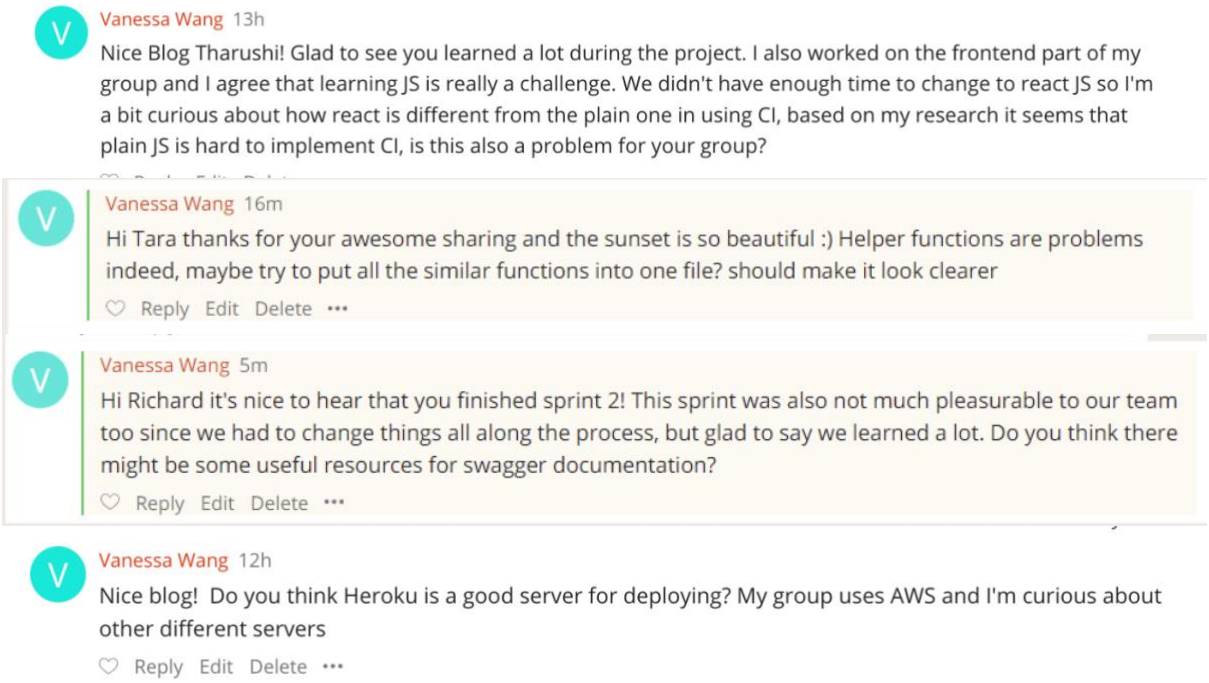
  Here is an example how my frontend teammates and I figured out problem. We used other team's validation API, but it seemed that there's some problems with interaction, so we had a meeting and sent posts on forum. Finally, we managed to contact the author group on forum and fixed the problem.



Messenger problem catchup

Blog Posts

- **always encourage others and discuss through replying forum posts**



Reflection

- **It’s important to communicate before starting to code**

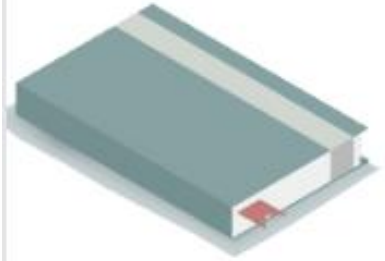
  Our team has a lot of helper functions in each part of code. So, we decided to put similar functions inside a file which could make it clearer. We didn't do well communication for the database part. We modified the database structure many times and always could not achieve the best way, next time I think we should spend more time communicating about each other's requirements.

- **Should have more discussions on forum posts**

  I didn’t spend much time checking and replying to forum posts in the beginning weeks since I thought they were not important. However, I found there were a lot of useful advice, and the forum is good place to communicate with other group (e.g., we solved validation problem through forum)

Yanran Wang

H18A Team Fudge



ALWAYS WRITE CLEAR CODE

ALWAYS WRITE TESTS

ALWAYS LEARN NEW SKILLS & DEBUG

WHAT HAVE I DONE?

- Clear code and tests for all the tasks
- Tests before coding
- Improve debug skills

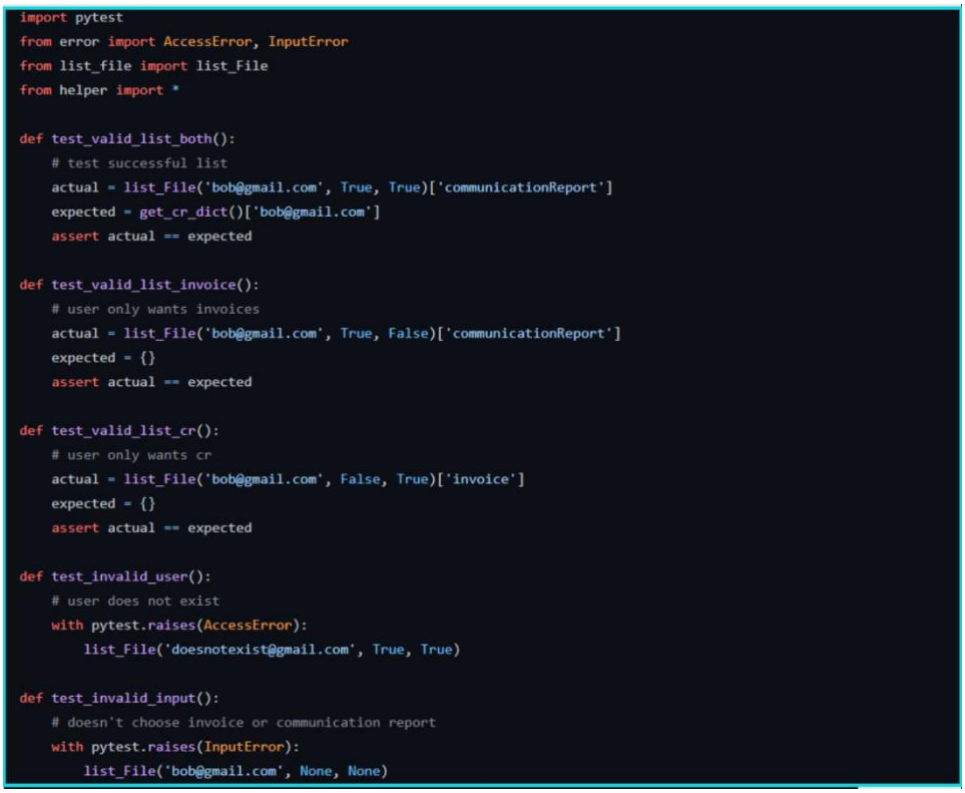
Improve

CODE & TESTS ALL THE TIME

Code & Tests

From code below it could be seen that I considered

- *helper functions to avoid long code*
- *errors for possible situations (AccessError & InputError)*



```python
import pytest
from error import AccessError, InputError
from list_file import list_File
from helper import *

def test_valid_list_both():
    # test successful list
    actual = list_File('bob@gmail.com', True, True)['communicationReport']
    expected = get_cr_dict()['bob@gmail.com']
    assert actual == expected

def test_valid_list_invoice():
    # user only wants invoices
    actual = list_File('bob@gmail.com', True, False)['communicationReport']
    expected = {}
    assert actual == expected

def test_valid_list_cr():
    # user only wants cr
    actual = list_File('bob@gmail.com', False, True)['invoice']
    expected = {}
    assert actual == expected

def test_invalid_user():
    # user does not exist
    with pytest.raises(AccessError):
        list_File('doesnotexist@gmail.com', True, True)

def test_invalid_input():
    # doesn't choose invoice or communication report
    with pytest.raises(InputError):
        list_File('bob@gmail.com', None, None)
```

detailed tests for listing (deleted later since feature deletion)

```python
import os
from error import AccessError, InputError
from helper import read_file, get_cr_dict, get_user_dict

def list_File(email, is_invoice, is_report):

    total_list = {}

    # check if the email is valid
    users = get_user_dict()
    if email not in users.keys():
        raise AccessError("Email is not in the system!")
    # check if the user logs in
    if users[email]["is_active"] == False:
        raise AccessError("The user has not logged in!")
    # check if user chooses which one to get
    if is_invoice is None or is_report is None:
        raise InputError("Please confirm if invoice/communication report is needed!")

    total_list['invoice'] = list_invoice(email, is_invoice)
    total_list['communicationReport'] = list_cr(email, is_report)

    return total_list
```

```javascript
function analysis() {

    anychart.onDocumentReady(function() {
        var fd = new FormData()
        var token = localStorage.getItem('token');

        var org_name = localStorage.getItem("analysis_org")
        if (org_name == "Select organisation") {
            return
        }


        fd.append("token", token)
        fd.append("org_name", org_name)


        let config = {
            headers: {
                'Content-Type': 'multipart/form-data'
            }
        }

        axios.put("http://127.0.0.1:3000/analysis", fd, config)
        .then(function(response) {


            // FIRST CHART
            paid = response.data['paid']
            unpaid = response.data['unpaid']
            var data = [
                {x: "Paid", value: paid},
                {x: "Unpaid", value: unpaid}
            ];
            var chart = anychart.pie();
            chart.title("Invoice Payable Amount Paid or Unpaid");
            chart.title().fontColor("Black")
            chart.legend().fontColor("Black")
            chart.data(data);
            chart.container('firstChart');
```

```
function reset_send() {
    var email = document.getElementById('email').value;
    axios.post('http://127.0.0.1:3000/auth/passwordreset/request/v1', {
        email: email
      }
    ).then(function(){
        alert("code has been sent!")
        window.location.href = "reset.html"
    })
    .catch(err => console.log(err));
}

function reset_password() {
    var new_password = document.getElementById('new_password').value;
    var reset_code = document.getElementById('reset_code').value;

    axios.post('http://127.0.0.1:3000/auth/passwordreset/reset/v1', {
        reset_code: reset_code,
        new_password: new_password
    ).then(function(){
        alert("change success!");
        window.location.href = "login.html"
    })
    .catch(err => console.log(err));
}
```


some frontend coding

Reflection

- **still need more knowledge about API / Frontend**

  For both my functions and documentation I spent lots of time doing the research, which is inefficient. I think for the next part I would choose **a more structed approach to learning**. Also, for frontend part, I didn't have enough time to learn everything about JS and HTML, so some codes weren't well-designed. In the future I would have better time management.

- **Codes and tests are clear**

  One of the most important things is that I have written my code and tests clearly all the time, which is easy to understand. The only thing I want to improve is the code style including **passing pylint** and **improving the quality**.

Appendix

Weekly Forum Post

Week 1: https://edstem.org/au/courses/7693/discussion/715408

Week 2: https://edstem.org/au/courses/7693/discussion/723451

Week 3: https://edstem.org/au/courses/7693/discussion/736271

Week 4: https://edstem.org/au/courses/7693/discussion/749976

Week 6: https://edstem.org/au/courses/7693/discussion/775501

Week 7: https://edstem.org/au/courses/7693/discussion/797705

Week 8: https://edstem.org/au/courses/7693/discussion/810651

Week 9: https://edstem.org/au/courses/7693/discussion/820058

Forum comment

https://edstem.org/au/courses/7693/discussion/822750

https://edstem.org/au/courses/7693/discussion/822668

https://edstem.org/au/courses/7693/discussion/818673

https://edstem.org/au/courses/7693/discussion/816651

https://edstem.org/au/courses/7693/discussion/752156

https://edstem.org/au/courses/7693/discussion/749910