COSC 4P02 Project Progress Report 1
Web Summarizer and Shortener

Index:

1. Team List:

Name	Brock ID	Email Address	Role(s)
Xiaobin Liu	6246789	xl17gl@brocku.ca	Scrum Master, developer team
Di Wu	6345912	dw17kf@brocku.ca	developer team
Daniel Yang	6937601	dy19hh@brocku.ca	developer team
Yuchen Ding	6677967	yd18jd@brocku.ca	developer team
William Li	6471254	wl18sc@brocku.ca	Product owner, develop team
Yuhao Cao	6594899	yc18do@brocku.ca	developer team

2. Our team has done so far(at the end of Sprint 3):

- Our team successfully implemented the function of short link creation and was able to make this function work properly on both the web and local sides. We successfully implemented the function of short link generation and were able to make it work properly on both the web and local sides. At the same time, the system will also check whether the user's input is standardized. If the wrong URL format is deliberately entered(e.g. 321.123.com), the system will notify the user of the input error in the form of a pop-up window. The principle is first to accept the URL entered by the user on the web page or the local front-end, then enter the URL into the database for storage, and then the database returns an indexed and numbered URL to the user. When building the server to run the web page, our team used *Nginx* and dynamic website development methods; when building the back-end database, our team used *Python/Flask* for development. Also, we finished the unit test for short link creation, and our code passed it successfully.
    - About *Nginx*: Nginx is a lightweight web server and reverse proxy server. Because of its small memory footprint, speedy startup, and high concurrency capabilities, it is widely used in Internet projects. Website: https://www.nginx.com/
    - About *Flask*: Flask is a lightweight web application framework written in Python. Its WSGI toolbox uses Werkzeug and its template engine uses Jinja2. Flask is licensed under BSD. It is also called a "micro-framework" since it uses a simple core and extensions to add additional functionality. Flask does not have a default database or form validation tool. Website: https://flask.palletsprojects.com/en/3.0.x/

- Our team successfully implemented free registration and login for users on both the web and local sides. Our system will accept the user's registration information (email address and password) through the register.html file and store it in the database. At the same time, the system will also detect whether the registration information entered by the user is blank and whether the email address entered by the user meets the specifications. In other words, if the user does not fill in all the registration information or uses the wrong email address, the system will prevent the user from registering and notify The user to refill the registration information. Of course, our system will also encrypt the stored user passwords. Here we use *Python/Flask-Bcrypt* to encrypt the passwords to ensure data security. Since the flask method does not provide effective and comprehensive form verification, we decided to use a third-party plug-in to handle verification which is *Validators with WTForms*. This allows us to verify that the email address and password entered by the user already exist in the database. In other words, it is used to check whether the user has successfully registered. Furthermore, we have also successfully implemented some social media integration through Google and FB authentication. Now users can directly share the generated new link to their Facebook account through the button at the bottom of the page after the short link is generated. Of course, this requires users to first log in to their Facebook account on other pages. Not only that, users can also log into our webpage with one click through their Google account by using the *OAuth 2.0 client and OAuth 2.0 protocol*.

- About *WTForms*, this is a form component that supports multiple web frameworks and is used to verify user request data. The *Validators* is a method used to verify the validity of data entered by the user. Website: https://flask-wtf.readthedocs.io/en/1.2.x/quickstart/#validating-forms
- About *Flask-Bcrypt*, this is a class container for hashing passwords and checking logic. This method is currently relatively simple and widely used. The main function of this method is to provide a simple interface to override Werkzeug's password-hashing function. Website: https://flask-bcrypt.readthedocs.io/en/1.0.1/
- About *OAuth 2.0 client and protocol*, this is an industry-standard three-party authorization protocol. It defines four authorization interaction modes that can be applied to various application scenarios: authorization code mode, application credit mode, user credit mode, and simplified mode. Currently, most third-party logins (including Google) and authorization are based on the standard or improved implementation of this protocol. The 2.0 protocol cancels the encryption process of all Tokens and simplifies the authorization process, but it is considered more secure than 1.0 because it mandates the use of the HTTPS protocol. Website: https://developers.google.com/identity/protocols/oauth2

3. What are we going to do:

- **Four features left:**
  - User Dashboard (pro-feature) - implementation during sprint 4
  - API Access (pro-feature) - implementation during sprint 5
  - Summarization by NLP or LLMs - implementation during sprint 6
  - Summarization Levels (pro-feature) - implementation during sprint 7

- **Test:**
  - The whole system test, including the unit test and user test - in the last week

4. Sprint Records:

- **Sprint 1(Jan 15 - Jan 28) - proposal and prepare:**
    - Sprint Review:
        - We held a sprint planning meeting in the first week of Sprint 1(Date: Jan 15, morning) to discuss and produce the user stories needed for the project. Then we analyzed the product backlog and broke it into sprint backlogs. During the meeting, we determined the theme of our project - Web Summarizer and Shortener and an initial timeline for the entire project and divided it into 7 sprints. Therefore, we can only focus on completing one feature and complete enough testing in each sprint and use the last sprint to do the final check and tests on the whole system. The advantage is this schedule would give us more time to solve problems we meet during the development process since no one in our team has experience developing actual software projects.

        - After the meeting, the scrum master organized and produced a project proposal based on the meeting notes. Other team members began searching and learning for technologies and tools we needed for the whole project.

        - During the second week of this sprint, our team members further discussed and analyzed the requirements, tasks, functions, steps and other details we needed to build the system(Date: Jan 22, afternoon), The scrum master finished and submitted the planning document based on team discussion.
    - Sprint Retrospective:
        - In Sprint 1, our team did an excellent job, and everyone actively put forward their opinions and suggestions when encountering problems or making plans. Everyone's communication and cooperation were also very active, so the preparations for the project went very smoothly.
    - So far, we have completed all preparations. After the review and retrospective, we determined our next goal is the first feature - Short Link Creation.

- **Sprint 2(Jan 29 - Feb 11) - first feature - Short Link Creation:**
    - Sprint Review:
        - In the second sprint, our goal is to build a short link creation function. Fortunately, this seems easy, and many team members have come up with their ways of doing it. In the end, we decided to use the database as the backend to read the input from the webpage front-end as the main framework to implement this function. At this time a new point of disagreement emerged. After searching for information, studying, and group discussion(Date: Jan 30, afternoon), we found that implementing this function in Python is simpler and easier to understand than using JAVA. We finally decided to use Python although we decided to use JAVA in the sprint planning meeting.

        - However, we still encountered some problems when writing the code, which will be described in detail in the "Issues" part below. After hard work, we finally solved these difficulties and wrote unit tests for this feature. Going into detail, we wrote some unit tests to check our code. We decided to test our code based on methods: A *home page* method test, a *URL shortening* method test, a *redirect to URL* method test, a *database operation* method test, and a *submit invalid URL* method test. The reason why we divided our program like this is we want to make sure all methods in our code can work normally. After successfully passing all unit tests, all team members agreed that the short link creation function had been completed.
    - Sprint Retrospective:
        - In Sprint 2, our team encountered some difficulties in its work, but everyone in the team actively sought help and communicated. Everyone did not feel lost or frustrated because of these difficulties. Instead, they continued to discuss the causes of the problems and possible solutions with other team members. This allowed us to quickly resolve the issue in a short time.
    - After the review and retrospective, we encountered a problem. What should we do in this sprint? According to the project proposal schedule, we must start building custom summarization levels. But this is impossible. First, after group discussion, we found that we cannot allow users to customize the summary level without implementing NLP or LLMs to summarize web pages. Second, all group members needed to learn how to access and use NLP and LLM. So we decided to postpone the implementation of this feature and take Social Media Integration, a feature we currently have ideas to implement, as the next goal since we believe that we can get some advice and tips during Tutorial 3 (LLMs and Prompt Engineering for Software Engineers) on Friday, March 1st.

- **Sprint 3(Feb 12 - Feb 25) - second feature - Social Media Integration:**
    - Sprint Review:
        - After the new goal had been set. Our team started working on allowing users to register accounts and login freely and added social media integration through Google and FB authentication into the system.
    - Sprint Retrospective:
        - In Sprint 3, our team overturned and reset our project plan for the first time. This is a bad omen, which shows that our initial plan has loopholes and shortcomings to a certain extent. We did not fully consider the possible problems in actual code writing, nor did a comprehensive discussion of the technologies that each team member is good at and the technologies that they are not good at. In the next Sprint, we need to pay more attention to all details of the project, which will include the expected difficulty of implementing this feature, the general implementation method, whether the team members have ideas for the implementation method, etc. This ensures we don't make the same mistake again.
    - After the review and retrospective, we found that Regarding the Social Media Integration through Google and FB authentication function, our program currently still has a few problems which will be described in the "Issues" part below, and we will seek some help to solve them at the next meeting with TA. At the same time, in the next Sprint, implementing a User Dashboard for registered users to track their activity, manage links, and customize settings will be our main goal.

5. Issues:

- **Solved:**
    - When implementing the function of short link creation, we encountered a challenging problem: the function of short link creation can run normally on local files, but cannot work on the web page (server). We continuously debug the configuration files placed on the server, but once the web page front-end files and back-end database are placed in the cloud server, the program cannot run normally. After discussion and continuous attempts by the team members, the main cause of this problem was finally determined: dynamic development and static development of the website. Static web pages refer to HTML files stored in the server file system. Once each static web page content is published, it will be sent to the website server, regardless of whether users are accessing it. However, through searching and learning, we found that static web pages are not supported by a database. This means that we need to convert the web page to dynamic development to cooperate with the back-end

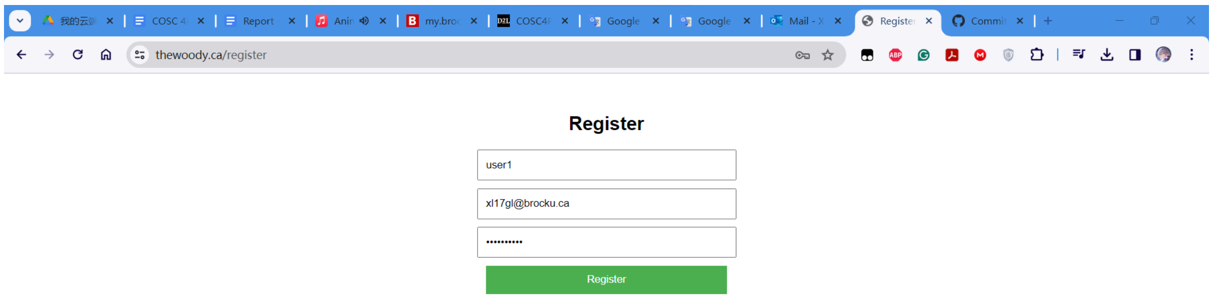
database. We began to try the following method: when the user enters the URL in the browser, the front end of the web page requests the server's response. The server dynamically generates an HTML page based on the current time, environmental parameters, database operations, etc., then returns it to the browser and the user. The good news is, we succeeded.

  - When a user enters an incorrect URL (e.g. 321.123.com) on the front end of the web page, the system cannot determine that the URL is incorrect and will give a prompt. This was a flaw, and after discussion among the group members, this feature was added and improved in the web page file(index.html).

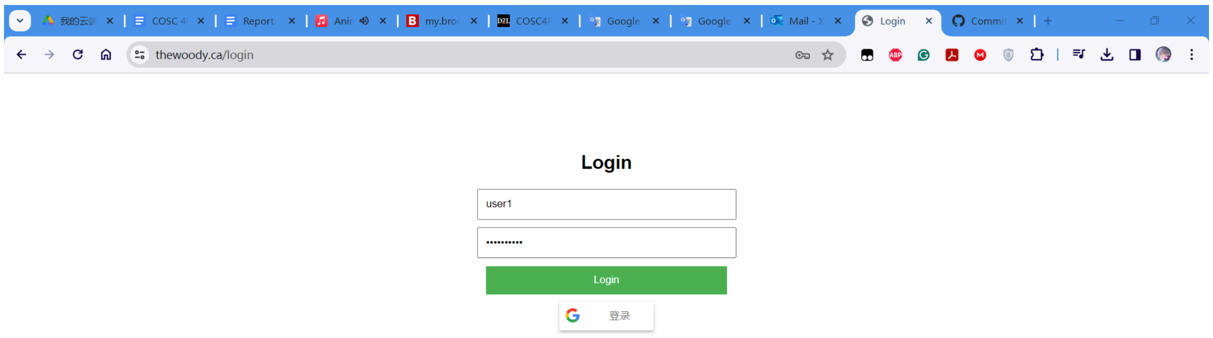
- **Still exist:**
  - We discussed whether API access could be implemented as a second function, but the team members had different opinions on the specific form of API access. It just displays a window on a third-party website that can call our website? Or make a button on a third-party website that can jump to our website? Or is it just a matter of simply giving an access port? Does allowing users to directly log in to our system through a third-party account also achieve the API access function? We decided to delay the implementation of this function since we would get some advice and help during the meeting with the TA.
  - In the Social Media Integration through Google and FB authentication function, we cannot log in to our system through a Facebook account. This problem is quite troublesome. Due to Facebook(Meta)'s policy, using a Facebook account to log in to third-party websites requires calling user data, and unofficial user data calls require qualification verification. We need to apply for certification from Facebook (Meta). Although we have applied as required, we are still waiting for a response from Facebook (Meta). Without obtaining certification qualifications, we cannot perform more operations. We can only allow users to share the generated short link to their Facebook account with one click after logging in to their Facebook account. Does this also count as realizing the function of Social Media Integration? We will get some advice and help during the meeting with the TA.

6. Screenshots:

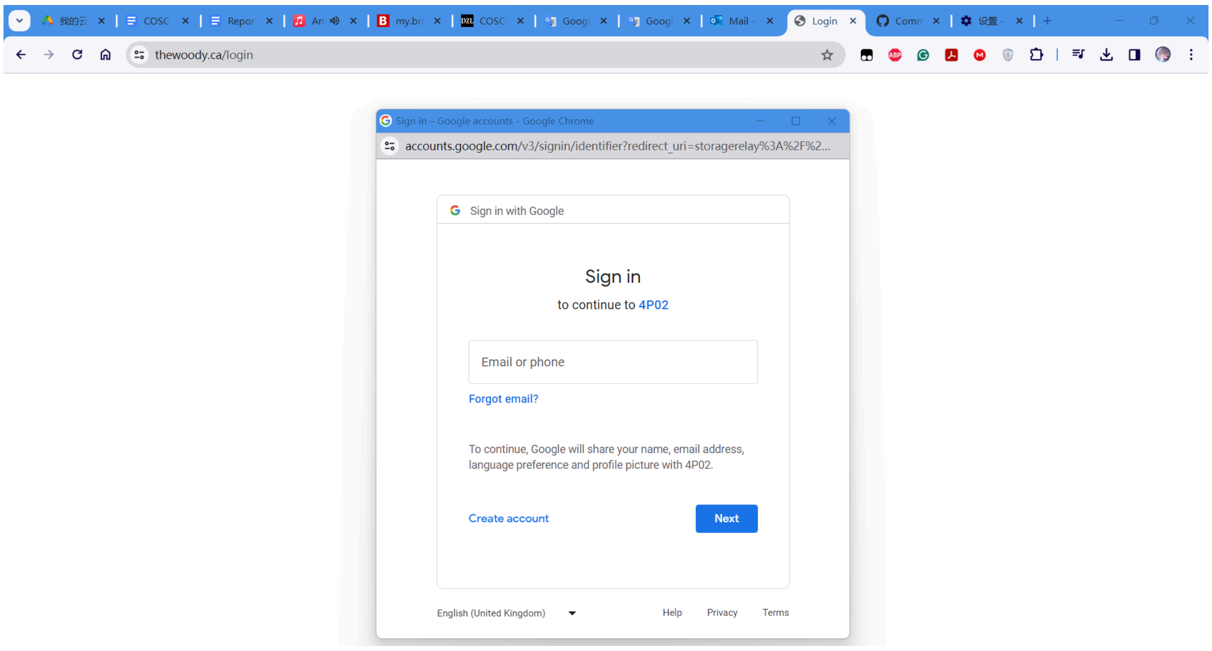
Register page:



Login page:



Login with Google:


thewoody.ca/login

Sign in – Google accounts – Google Chrome

accounts.google.com/v3/signin/identifier?redirect_uri=storagerelay%3A%2F%2...

Sign in with Google

Sign in

to continue to 4P02

Email or phone

Forgot email?

To continue, Google will share your name, email address, language preference and profile picture with 4P02.

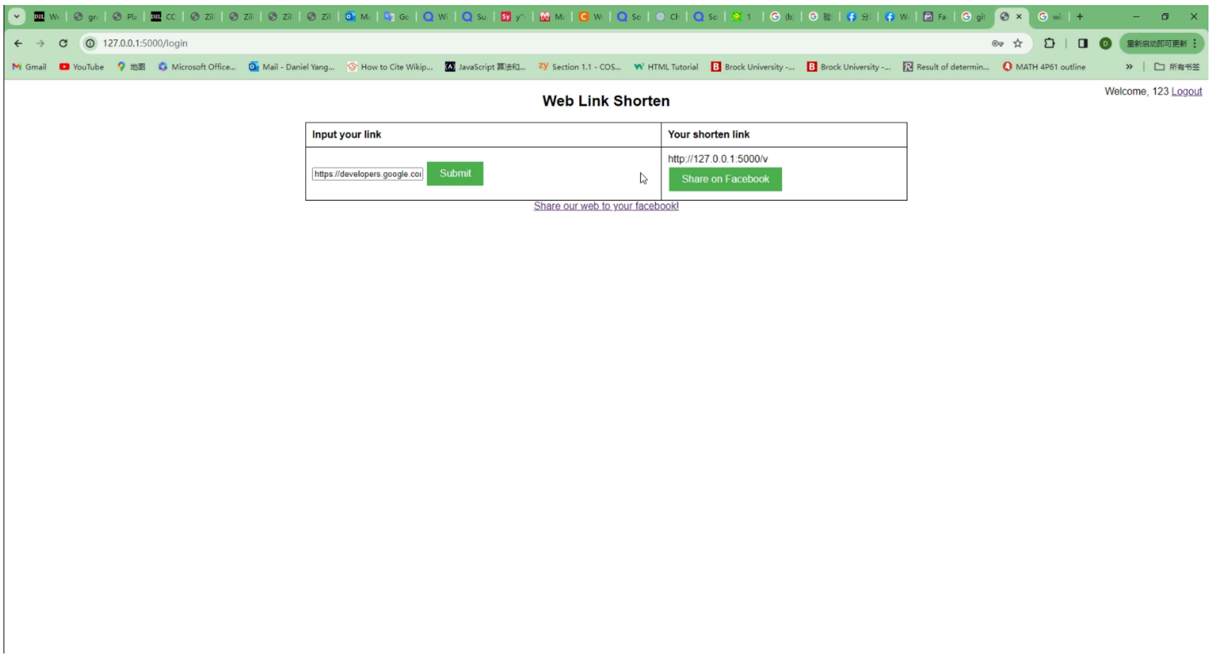
Create account

Next

English (United Kingdom)

Help Privacy Terms

Main page:


127.0.0.1:5000/login

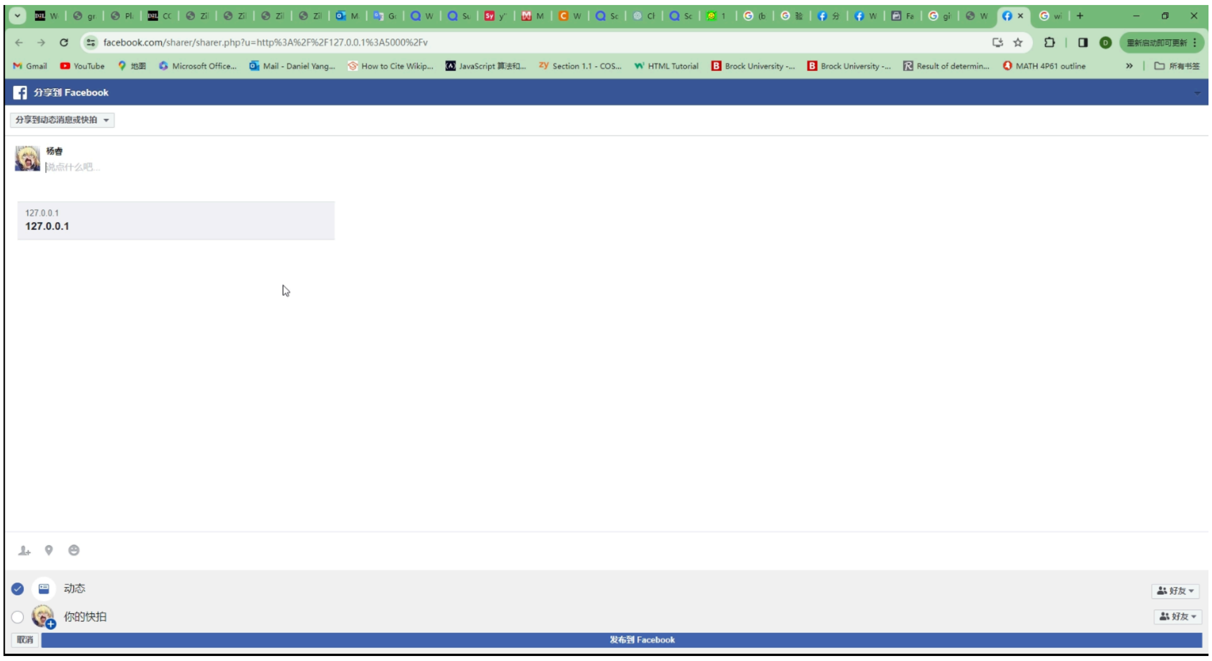
Welcome, 123 Logout

Web Link Shorten

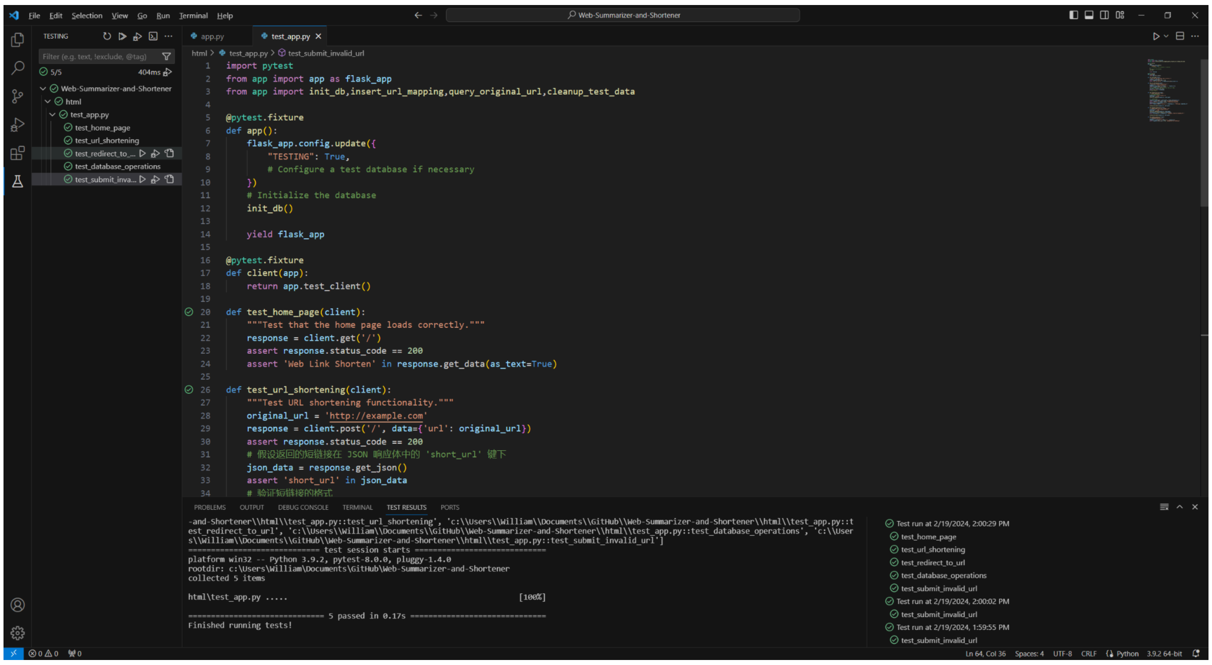
Input your link	Your shorten link
https://developers.google.co Submit	http://127.0.0.1:5000/v Share on Facebook

Share our web to your facebook!

Share to Facebook:



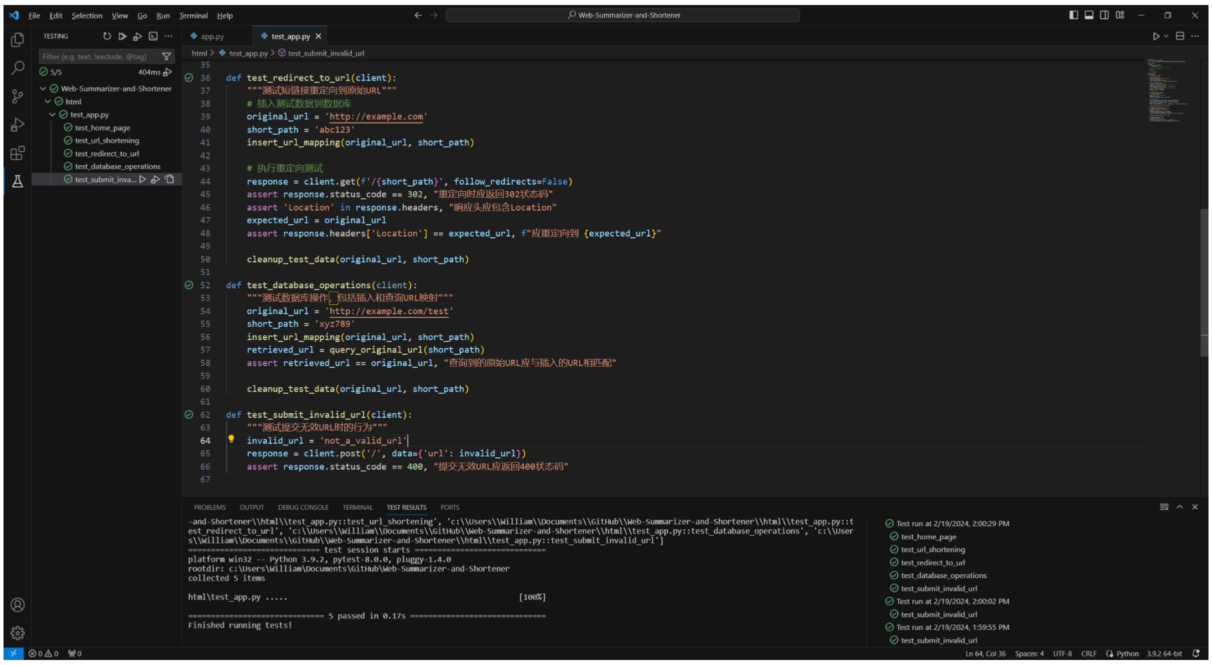
System unit test:



```python
import pytest
from app import app as flask_app
from app import init_db,insert_url_mapping,query_original_url,cleanup_test_data

@pytest.fixture
def app():
    flask_app.config.update({
        "TESTING": True,
        # Configure a test database if necessary
    })
    # Initialize the database
    init_db()

    yield flask_app

@pytest.fixture
def client(app):
    return app.test_client()

def test_home_page(client):
    """Test that the home page loads correctly."""
    response = client.get('/')
    assert response.status_code == 200
    assert 'Web Link Shorten' in response.get_data(as_text=True)

def test_url_shortening(client):
    """Test URL shortening functionality."""
    original_url = 'http://example.com'
    response = client.post('/', data={'url': original_url})
    assert response.status_code == 200
    # 假设返回的短链接在 JSON 响应体中的 'short_url' 键下
    json_data = response.get_json()
    assert 'short_url' in json_data
    # 验证短链接的格式
```

```
-and-Shortener\\html\\test_app.py::test_url_shortening', 'c:\\Users\\William\\Documents\\GitHub\\Web-Summarizer-and-Shortener\\html\\test_app.py::t
est_redirect_to_url', 'c:\\Users\\William\\Documents\\GitHub\\Web-Summarizer-and-Shortener\\html\\test_app.py::test_database_operations', 'c:\\User
s\\William\\Documents\\GitHub\\Web-Summarizer-and-Shortener\\html\\test_app.py::test_submit_invalid_url']
============================= test session starts ==============================
platform win32 -- Python 3.9.2, pytest-8.0.0, pluggy-1.4.0
rootdir: c:\Users\William\Documents\GitHub\Web-Summarizer-and-Shortener
collected 5 items

html\test_app.py .....                                                   [100%]

============================== 5 passed in 0.17s ===============================
Finished running tests!
```

```
✓ Test run at 2/19/2024, 2:00:29 PM
  ✓ test_home_page
  ✓ test_url_shortening
  ✓ test_redirect_to_url
  ✓ test_database_operations
  ✓ test_submit_invalid_url
✓ Test run at 2/19/2024, 2:00:02 PM
  ✓ test_submit_invalid_url
✓ Test run at 2/19/2024, 1:59:55 PM
  ✓ test_submit_invalid_url
```

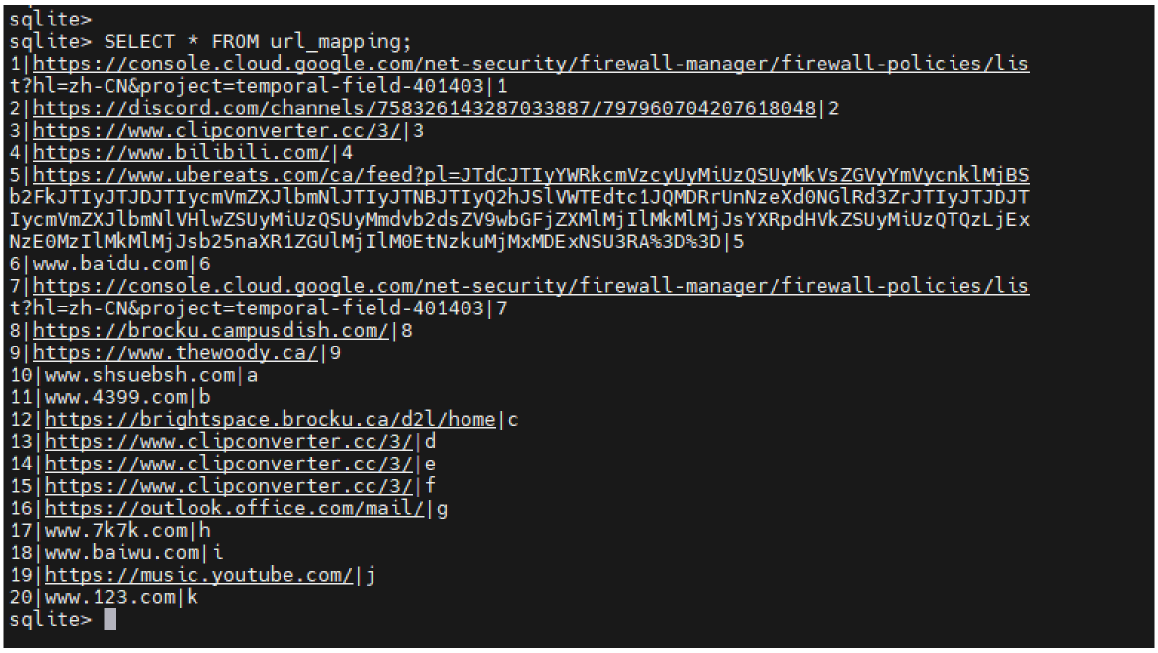


```python
def test_redirect_to_url(client):
    """测试短链接重定向到原始URL"""
    # 插入测试数据到数据库
    original_url = 'http://example.com'
    short_path = 'abc123'
    insert_url_mapping(original_url, short_path)

    # 执行重定向测试
    response = client.get(f'/{short_path}', follow_redirects=False)
    assert response.status_code == 302, "重定向时应返回302状态码"
    assert 'Location' in response.headers, "响应头应包含Location"
    expected_url = original_url
    assert response.headers['Location'] == expected_url, f"应重定向到 {expected_url}"

    cleanup_test_data(original_url, short_path)

def test_database_operations(client):
    """测试数据库操作，包括插入和查询URL映射"""
    original_url = 'http://example.com/test'
    short_path = 'xyz789'
    insert_url_mapping(original_url, short_path)
    retrieved_url = query_original_url(short_path)
    assert retrieved_url == original_url, "查询到的原始URL应与插入的URL相匹配"

    cleanup_test_data(original_url, short_path)

def test_submit_invalid_url(client):
    """测试提交无效URL时的行为"""
    invalid_url = 'not_a_valid_url'
    response = client.post('/', data={'url': invalid_url})
    assert response.status_code == 400, "提交无效URL应返回400状态码"
```

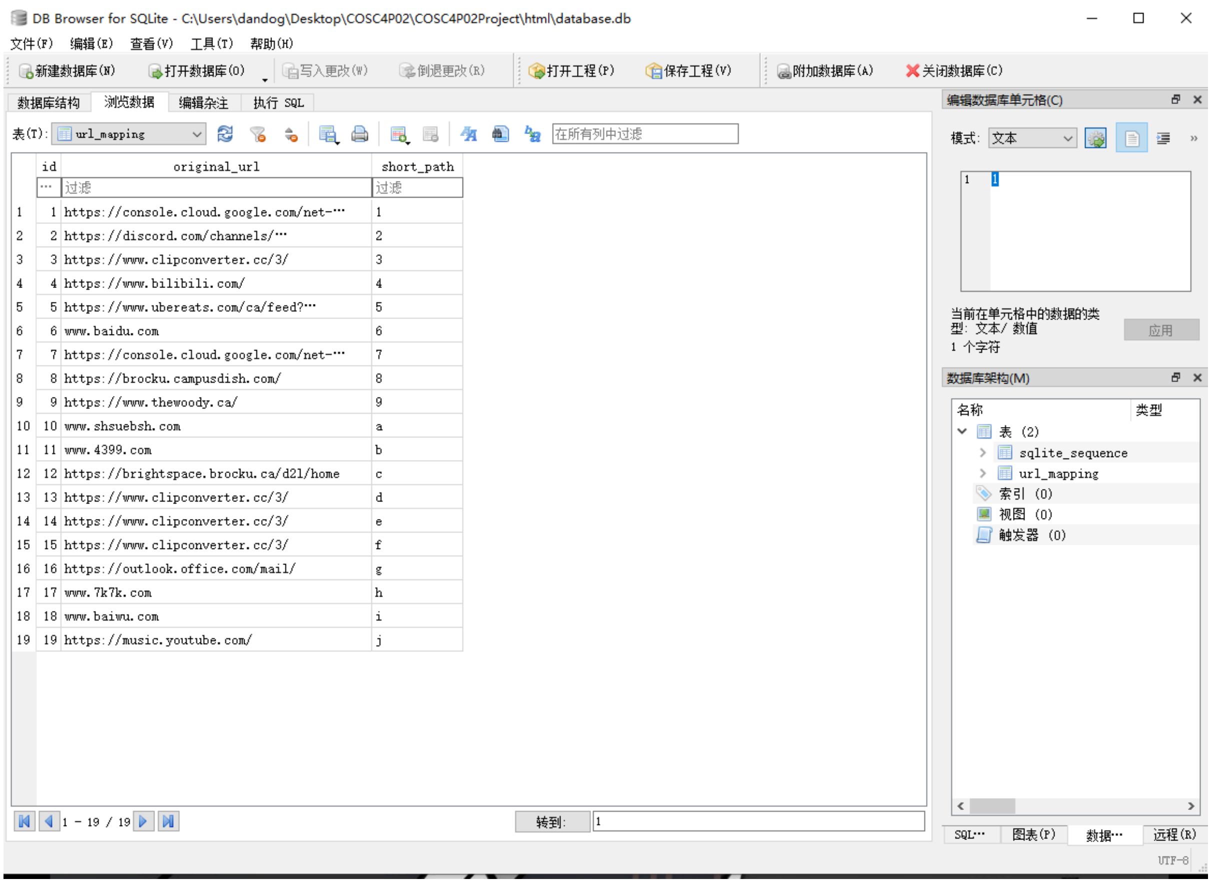
```
-and-Shortener\\html\\test_app.py::test_url_shortening', 'c:\\Users\\William\\Documents\\GitHub\\Web-Summarizer-and-Shortener\\html\\test_app.py::t
est_redirect_to_url', 'c:\\Users\\William\\Documents\\GitHub\\Web-Summarizer-and-Shortener\\html\\test_app.py::test_database_operations', 'c:\\User
s\\William\\Documents\\GitHub\\Web-Summarizer-and-Shortener\\html\\test_app.py::test_submit_invalid_url']
============================= test session starts ==============================
platform win32 -- Python 3.9.2, pytest-8.0.0, pluggy-1.4.0
rootdir: c:\Users\William\Documents\GitHub\Web-Summarizer-and-Shortener
collected 5 items

html\test_app.py .....                                                   [100%]

============================== 5 passed in 0.17s ===============================
Finished running tests!
```

```
Test run at 2/19/2024, 2:00:29 PM
  test_home_page
  test_url_shortening
  test_redirect_to_url
  test_database_operations
  test_submit_invalid_url
Test run at 2/19/2024, 2:00:02 PM
  test_submit_invalid_url
Test run at 2/19/2024, 1:59:55 PM
  test_submit_invalid_url
```

System database(two different vision) :



```
sqlite>
sqlite> SELECT * FROM url_mapping;
1|https://console.cloud.google.com/net-security/firewall-manager/firewall-policies/lis
t?hl=zh-CN&project=temporal-field-401403|1
2|https://discord.com/channels/758326143287033887/797960704207618048|2
3|https://www.clipconverter.cc/3/|3
4|https://www.bilibili.com/|4
5|https://www.ubereats.com/ca/feed?pl=JTdCJTIyYWRkcmVzcyUyMiUzQSUyMkVsZGVyYmVycnklMjBS
b2FkJTIyJTJDJTIycmVmZXJlbmNlJTIyJTNBJTIyQ2hJSlVWTEdtc1JQMDRrUnNzeXd0NGlRd3ZrJTIyJTJDJT
IycmVmZXJlbmNlVHlwZSUyMiUzQSUyMmdvb2dsZV9wbGFjZXMlMjIlMkMlMjJsYXRpdHVkZSUyMiUzQTQzLjEx
NzE0MzIlMkMlMjJsb25naXR1ZGUlMjIlM0EtNzkuMjMxMDExNSU3RA%3D%3D|5
6|www.baidu.com|6
7|https://console.cloud.google.com/net-security/firewall-manager/firewall-policies/lis
t?hl=zh-CN&project=temporal-field-401403|7
8|https://brocku.campusdish.com/|8
9|https://www.thewoody.ca/|9
10|www.shsuebsh.com|a
11|www.4399.com|b
12|https://brightspace.brocku.ca/d2l/home|c
13|https://www.clipconverter.cc/3/|d
14|https://www.clipconverter.cc/3/|e
15|https://www.clipconverter.cc/3/|f
16|https://outlook.office.com/mail/|g
17|www.7k7k.com|h
18|www.baiwu.com|i
19|https://music.youtube.com/|j
20|www.123.com|k
sqlite>
```


DB Browser for SQLite - C:\Users\dandog\Desktop\COSC4P02\COSC4P02Project\html\database.db

表(T): url_mapping

	id	original_url	short_path
1	1	https://console.cloud.google.com/net-…	1
2	2	https://discord.com/channels/…	2
3	3	https://www.clipconverter.cc/3/	3
4	4	https://www.bilibili.com/	4
5	5	https://www.ubereats.com/ca/feed?…	5
6	6	www.baidu.com	6
7	7	https://console.cloud.google.com/net-…	7
8	8	https://brocku.campusdish.com/	8
9	9	https://www.thewoody.ca/	9
10	10	www.shsuebsh.com	a
11	11	www.4399.com	b
12	12	https://brightspace.brocku.ca/d2l/home	c
13	13	https://www.clipconverter.cc/3/	d
14	14	https://www.clipconverter.cc/3/	e
15	15	https://www.clipconverter.cc/3/	f
16	16	https://outlook.office.com/mail/	g
17	17	www.7k7k.com	h
18	18	www.baiwu.com	i
19	19	https://music.youtube.com/	j

1 - 19 / 19

数据库架构(M): 表 (2): sqlite_sequence, url_mapping; 索引 (0); 视图 (0); 触发器 (0)

7. Contributions by each member:

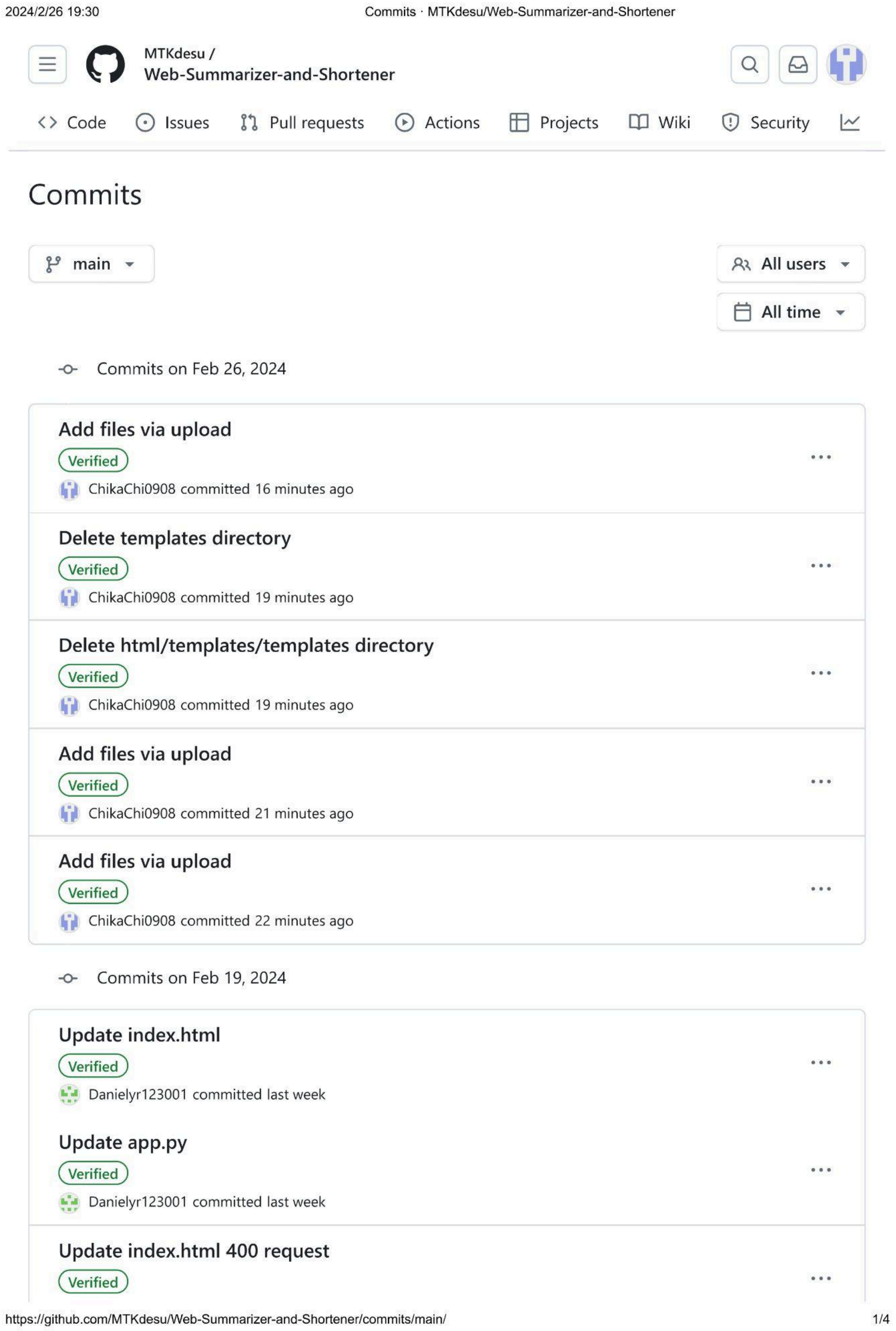
- **Front-End:**
  - Website home page, accept user inputs, login, logout, register, online servers and website page design are implemented by Di Wu and Daniel Yang.

- **Back-End:**
  - Read&save user inputs, database functions and return alert information are implemented by Yuchen Ding and Yuhao Cao.

- **Overall:**
  - Project total planning, project total management, Jira and Github page updates and system unit tests are carried out by William Li.

  - Information summary, problem record and summary, team adviser, and all document writing are carried out by Xiaobin Liu.

8. GitHub log of project activities:



2024/2/26 19:30 Commits · MTKdesu/Web-Summarizer-and-Shortener

MTKdesu /
Web-Summarizer-and-Shortener

Code Issues Pull requests Actions Projects Wiki Security

Commits

main All users All time

Commits on Feb 26, 2024

Add files via upload
Verified
ChikaChi0908 committed 16 minutes ago

Delete templates directory
Verified
ChikaChi0908 committed 19 minutes ago

Delete html/templates/templates directory
Verified
ChikaChi0908 committed 19 minutes ago

Add files via upload
Verified
ChikaChi0908 committed 21 minutes ago

Add files via upload
Verified
ChikaChi0908 committed 22 minutes ago

Commits on Feb 19, 2024

Update index.html
Verified
Danielyr123001 committed last week

Update app.py
Verified
Danielyr123001 committed last week

Update index.html 400 request
Verified

https://github.com/MTKdesu/Web-Summarizer-and-Shortener/commits/main/ 1/4

Danielyr123001 committed last week

Update test_app.py
Verified
MTKdesu committed last week

Commits on Feb 13, 2024

Update test_app.py
Verified
MTKdesu committed 2 weeks ago

pytest update
MTKdesu committed 2 weeks ago

Merge branch 'main' of https://github.com/MTKdesu/Web-Summarizer-and-Shortener
MTKdesu committed 2 weeks ago

upload pytest file
Verified
MTKdesu committed 2 weeks ago

update
MTKdesu committed 2 weeks ago

Add files via upload
Verified
Danielyr123001 committed 2 weeks ago

README
Verified
Danielyr123001 committed 2 weeks ago

Commits on Feb 12, 2024

html folder
Verified
Danielyr123001 committed 2 weeks ago

Commits on Feb 10, 2024

Delete html directory
Verified
MTKdesu committed 2 weeks ago

update

MTKdesu committed 2 weeks ago

Commits on Jan 26, 2024

Rename COSC 4P02 Projct Requirements Document(Draft).pdf to COSC 4P02 Projct Requirements Document.pdf

Verified

MTKdesu committed last month

Add files via upload

Verified

MTKdesu committed last month

Commits on Jan 17, 2024

Update README.md

Verified

Asukadd committed last month

Update README.md

Verified

Valorrrrr committed last month

Commits on Jan 15, 2024

Update README.md

Verified

ChikaChi0908 committed last month

Delete Document/Proposal.docx

Verified

MTKdesu committed last month

Add files via upload

Verified

MTKdesu committed last month

Delete Document/Proposal

Verified

MTKdesu committed last month

Add files via upload

Verified

MTKdesu committed last month

Delete Document/Project Proposal.docx
Verified
MTKdesu committed last month

Add files via upload
Verified
MTKdesu committed last month

Update README.md
Verified
MTKdesu committed last month

Create Proposal
Verified
MTKdesu committed last month

Update README.md
Verified
MTKdesu committed last month

Commits on Jan 14, 2024

Update README.md
Verified
Danielyr123001 committed last month

Update README.md
Verified
hao777777 committed last month

Previous Next