

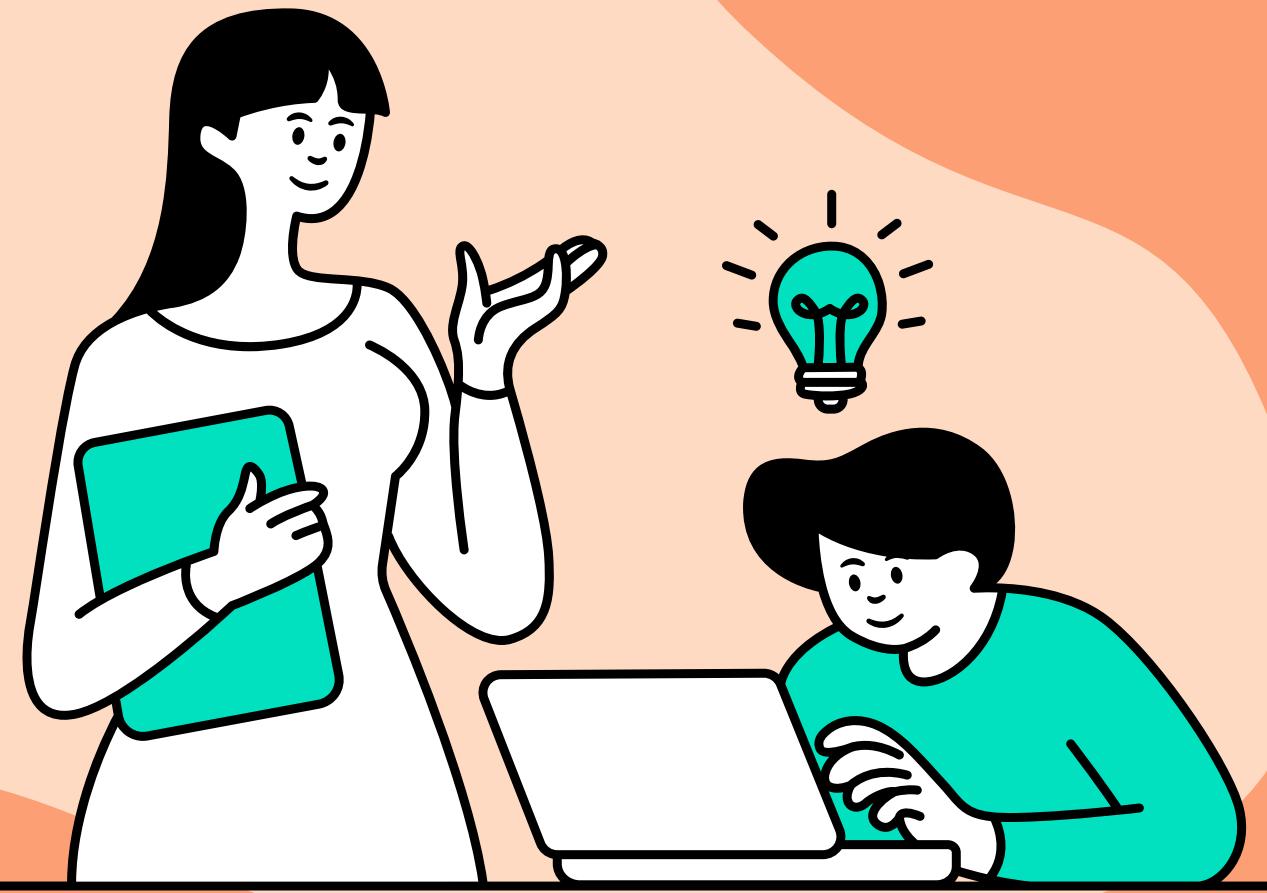
High.Teach

Demo 3



Contents:

- 1 The team
- 2 Video segments of 3 features
- 3 A demonstration of the application tests being run in GitHub actions
- 4 Future Elements
- 5 Git repository



The team:

David Cohen



Idan Rosenberg



Rany Whabi



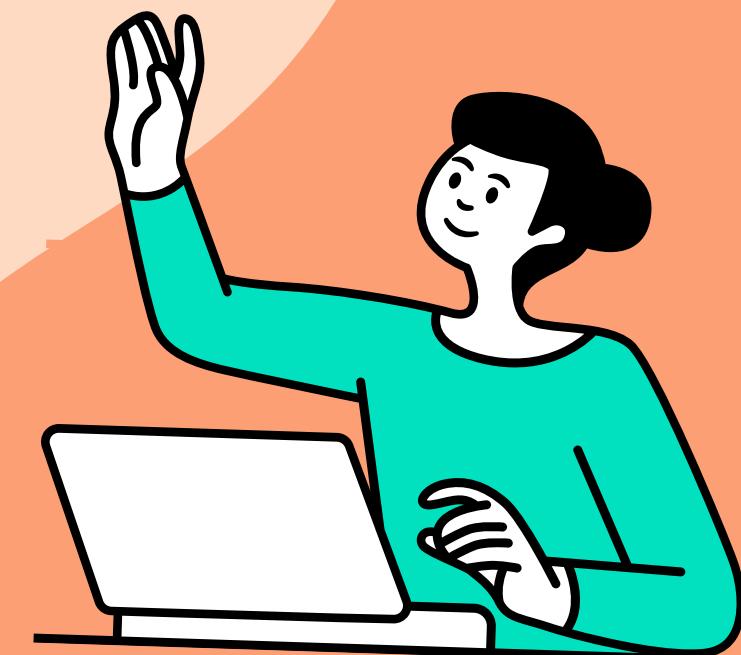
Yarden Gazit



Shiraz Yom Tov



Shachar Levy





In the previous presentation we presented the different models that our application contains. The understanding of the various entities necessary for the workflow of our system.

Now, we have moved to the UI stage, the implementation of the client interface and the communication between the user's actions and the information that the server provides us.





How we worked



First we created a base html files for the homepage and the screen after logging in. We chose an icon of light bulb indicating the possibility of sharing knowledge and ideas across the High.Teach platform.💡

All of these formed the first design infrastructure for the app.

We decided to start working on the features that represent the services provided on our platform: courses (which include reviews) and study groups.

The PRs planned for the coming week (toward the last demo) are PRs related to the social issue: user profile, chat and feed.





Home Page



High.Teach

[Register](#) [Login](#)

Welcome to High.Teach

High.Teach is a platform for connecting students and tutors.
It is a social network which enables tutors to expose themselves through dynamic connections with potential students, based on fields or special needs.



***We offer a temporary connection method using the Admin UI**

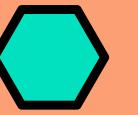


A base for the “logged in user” application



The screenshot shows the High.Teach application interface. At the top, there is a green header bar with the text "High.Teach" on the left and "Logout" on the right. Below the header, a blue sidebar on the left displays the user profile "teacher1" and "Private Account". The main content area contains a navigation menu with the following items:

- Feed
- Search
- Messages
- My Courses
- Study Groups



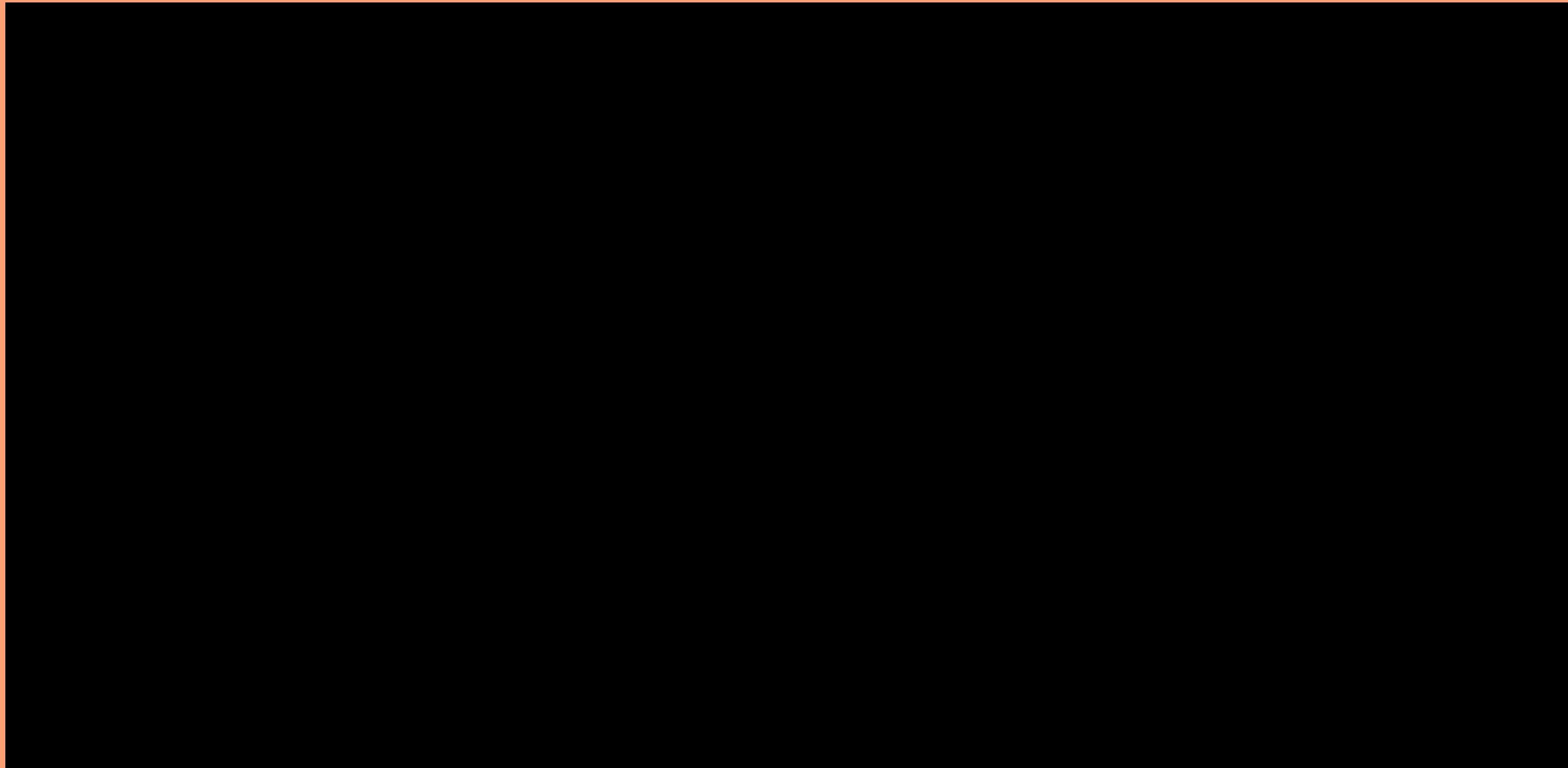
We will present now video segments showing how 3 features we've already implemented look and work from a user's point of view:





Management of Courses

The screenshot shows a web browser window with a green header bar. The header bar contains the text "High.Teach" on the left and "Register Login" on the right. The main content area has a white background. At the top center, the text "Welcome to High.Teach" is displayed in a green font. Below this, there is a descriptive paragraph: "High.Teach is a platform for connecting students and tutors. It is a social network which enables tutors to expose themselves through dynamic connections with potential students, based on fields or special needs." To the right of the text, there is a small cursor icon pointing towards the right edge of the screen. In the bottom right corner of the main content area, there is a logo featuring a person sitting cross-legged with a laptop, with the word "high.teach" written below it in a stylized font.





Study Group

High.Teach

127.0.0.1:8000/study-group/list/

Logout

idan
Private Account

Feed

Search

Messages

My Courses

Study Groups

My Study Groups

- Operating Systems Exc3**
Group for latest assignment...
1/3 Members
[Leave Group](#)
- Computability- Omer's group**
Computability course at MT...
1/25 Members
[Leave Group](#)
- Study group for Java**
This is a study group dedic...
1/10 Members
[Leave Group](#)
- Science**
This is a group for science I...
3/10 Members
[Leave Group](#)
- History**
This is a group for history b...
2/7 Members
[Leave Group](#)

Study Group Creation

Field:

Group description:

Capacity:

[Create Group](#)

Courses:

A demonstration of tests being run in GitHub actions

```
@pytest.mark.djangoproject
class TestCoursePageView:

    def test_courses_page_authorized(self, authorized_client, persist_course):
        course_response = authorized_client.get('/course/' + str(persist_course.course_id))
        assert course_response.status_code == 200
        expected_template_name = 'course_page.html'
        page_templates = course_response.templates[0].name
        assert expected_template_name in page_templates

    def test_courses_page_unauthorized(self, client, persist_course):
        course_response = client.get('/course/' + str(persist_course.course_id))
        assert course_response.status_code == 302

    def test_enroll_in_course(self, authorized_client, persist_course, persist_user):
        student_course = {'teacher_course_id': persist_course, 'student_id': persist_user}
        assert not StudentCourse.objects.filter(**student_course).exists()
        course_creation_response = authorized_client.post("/course/" + str(persist_course.course_id) + "/connect",
                                                          follow=True)
        assert course_creation_response.status_code == 200
        assert StudentCourse.objects.filter(**student_course).exists()
        expected_template_name = 'course_page.html'
        page_templates = course_creation_response.templates[0].name
        assert expected_template_name in page_templates

    def test_show_course_page(self, authorized_client, persist_course):
        response = authorized_client.get("/course/" + str(persist_course.course_id))
        assert response.status_code == 200
        course_in_page = response.context['course']
        assert persist_course.course_id == course_in_page.course_id
```

Tests - Courses

```
@pytest.mark.django_db
class TestAddCourseView:

    def test_courses_add_authorized(self, authorized_client):
        course_response = authorized_client.get('/course/add')
        assert course_response.status_code == 200
        assert 'add_course.html' in course_response.templates[0].name

    def test_courses_add_unauthorized(self, client):
        course_response = client.get('/course/add')
        assert course_response.status_code == 302

    def test_add_course(self, authorized_client):
        teacher_course = {"course_name": COURSE_NAME, "description": DESCRIPTION,
                          "difficulty_level": DIFFICULTY, "category": CATEGORY,
                          "price": PRICE, "years_of_experience": YEARS_OF_EXP}
        assert not TeacherCourse.objects.filter(**teacher_course).exists()
        course_creation_response = authorized_client.post("/course/add", teacher_course, follow=True)
        assert course_creation_response.status_code == 200
        assert 'courses.html' in course_creation_response.templates[0].name
        assert TeacherCourse.objects.filter(**teacher_course).exists()
```

Tests - Courses

```
@pytest.mark.djangoproject
class TestCourseTableView:

    def test_courses_list_authorized(self, authorized_client):
        course_response = authorized_client.get('/course/')
        assert course_response.status_code == 200
        assert 'courses.html' in course_response.templates[0].name

    def test_courses_list_unauthorized(self, client):
        course_response = client.get('/course/')
        assert course_response.status_code == 302

    def test_show_course_that_user_teaches(self, authorized_client, teacher_course_zero):
        response = authorized_client.get(reverse('show_courses'))
        assert response.status_code == 200
        courses_ids = [course.pk for course in response.context['courses']]
        assert teacher_course_zero.course_id in courses_ids

    def test_show_course_that_user_enrolled_in(self, authorized_client, persist_first_student_course):
        response = authorized_client.get(reverse('show_courses'))
        assert response.status_code == 200
        courses_ids = [course.pk for course in response.context['courses']]
        assert persist_first_student_course.student_course_id in courses_ids
```

Tests - Review

```
@pytest.mark.djangoproject_db
class TestReviewView:
    def test_create_review(self, client, persist_user, persist_review):
        client.force_login(persist_user)
        response = client.post(reverse('create_review', args=[persist_review.course_id]), {
            'rating': persist_review.rating,
            'content': persist_review.content
        })
        assert response.status_code == 302
        assert Review.objects.exists()

    def test_invalid_review_message(self, client, persist_user, persist_review):
        client.force_login(persist_user)
        response = client.post(reverse('create_review', args=[persist_review.course_id]), {
            'content': persist_review.content
        })
        messages = list(get_messages(response.wsgi_request))
        assert len(messages) == 1
        assert str(messages[0]) == "You must enter rating"

    def test_update_review(self, client, persist_user, persist_review):
        client.force_login(persist_user)
        response = client.post(reverse('update_review', args=[persist_review.pk]),
                               data={'rating': persist_review.rating, 'content': persist_review.content})
        assert response.status_code == 302
        assert Review.objects.get(pk=persist_review.pk).rating == 1
        assert Review.objects.get(pk=persist_review.pk).content == "Greate course"
```

Tests - Review

```
def test_delete_review(self, client, persist_user, persist_review):
    client.force_login(persist_user)
    response = client.post(reverse('delete_review', args=[persist_review.pk]))
    assert response.status_code == 302
    assert not Review.objects.filter(pk=persist_review.pk).exists()

def test_show_reviews_view(self, client, persist_user, persist_review):
    client.force_login(persist_user)
    response = client.get(reverse('reviews', args=[persist_review.course_id]))
    assert response.status_code == 200
    assert any(persist_review.pk == review.pk for review in response.context['reviews_by_course'])

def test_show_without_reviews_view(self, client, persist_user, persist_course):
    client.force_login(persist_user)
    response = client.get(reverse('reviews', args=[persist_course.course_id]))
    assert response.status_code == 200
    assert response.context['reviews_by_course'] == []
    assert b"No reviews available" in response.content
```

Tests - Study Group

```
@pytest.mark.djangoproject_db
class TestStudyGroupUpdateView:
    def test_study_group_update_view_loaded_authorized(self, authorized_client, persist_group):
        group_detail_response = authorized_client.get(f"/study-group/update/{persist_group.pk}/")
        assert group_detail_response.status_code == 200
        assert 'study_group_update.html' in group_detail_response.templates[0].name

    @pytest.mark.parametrize("new_field, new_desc", [("new field", "this is a new group description")])
    def test_update_group_details_not_owner(self, make_study_group_of_varied_size, authorized_client, new_field,
                                            new_desc):
        group_not_owner_of = make_study_group_of_varied_size(1, 5)
        response = authorized_client.post(f"/study-group/update/{group_not_owner_of.pk}/",
                                           {"field": new_field, "group_description": new_desc})
        assert response.status_code == 403

@pytest.mark.djangoproject_db
class TestJoinLeaveGroupButton:
    def test_leave_group(self, client, persist_user, persist_group):
        client.force_login(persist_user)
        persist_group.join_group(persist_user)
        assert persist_group.is_user_in_group(persist_user)
```

Tests - Study Group

```
@pytest.mark.djangoproject_db
class TestJoinLeaveGroupButton:
    def test_leave_group(self, client, persist_user, persist_group):
        client.force_login(persist_user)
        persist_group.join_group(persist_user)
        assert persist_group.is_user_in_group(persist_user)

        post_response = client.post(f"/study-group/join_leave/{persist_group.pk}/")

        assert post_response.status_code == 302
        assert post_response.url == reverse("study_groups_list_for_user")
        assert not persist_group.is_user_in_group(persist_user)

    def test_join_group(self, client, persist_user, persist_group):
        client.force_login(persist_user)
        assert not persist_group.is_user_in_group(persist_user)

        post_response = client.post(f"/study-group/join_leave/{persist_group.pk}/")

        assert post_response.status_code == 302
        assert post_response.url == reverse("study_groups_list_for_user")
        assert persist_group.is_user_in_group(persist_user)
```

Tests - Study Group

```
def test_join_leave_non_existent_group(self, authorized_client, persist_group):
    StudyGroup.objects.get(pk=persist_group.pk).delete()
    post_response = authorized_client.post(f"/study-group/join_leave/{persist_group.pk}/")

    assert post_response.status_code == 404

@pytest.mark.parametrize("non_full, capacity", [(0, 5)])
def test_join_group_button_text(self, make_study_group_of_varied_size, authorized_client,
                               non_full, capacity):
    non_empty_group = make_study_group_of_varied_size(non_full, capacity)
    response = authorized_client.get(f"/study-group/detail/{non_empty_group.pk}/")
    assert b"Join Group" in response.content

@pytest.mark.parametrize("full, capacity", [(1, 1)])
def test_full_group_button_text(self, make_study_group_of_varied_size, authorized_client,
                               full, capacity):
    non_empty_group = make_study_group_of_varied_size(full, capacity)
    response = authorized_client.get(f"/study-group/detail/{non_empty_group.pk}/")
    assert b"Group Full" in response.content
```

Tests - Study Group

```
assert b"Group Full" in response.content

@pytest.mark.parametrize("num_of_members, capacity", [(1, 2)])
def test_leave_group_text_button(self, make_study_group_of_varied_size, num_of_members, capacity,
                                 persist_user, client):
    study_group = make_study_group_of_varied_size(num_of_members, capacity)
    study_group.join_group(persist_user)

    client.force_login(persist_user)
    response = client.get(f"/study-group/detail/{study_group.pk}/")
    assert b"Leave Group" in response.content
```

Tests - Study Group

```
@pytest.mark.djangoproject_db
class TestStudyGroupListView:
    def test_study_group_list_view_loaded_authorized(self, authorized_client):
        study_group_list_response = authorized_client.get('/study-group/list/')
        assert study_group_list_response.status_code == 200

    def test_study_group_list_view_loaded_unauthorized(self, client):
        study_group_list_response = client.get('/study-group/list/')
        assert study_group_list_response.status_code == 302

@pytest.mark.parametrize("field, group_description, capacity",
                      [ ("Test Group", "Description", 5)])
def test_create_new_study_group(self, authorized_client, field, group_description, capacity):
    study_group_creation_form_args = {"field": field, "group_description": group_description, "capacity": capacity}

    assert not StudyGroup.objects.filter(**study_group_creation_form_args).exists()
    group_creation_response = authorized_client.post("/study-group/list/", study_group_creation_form_args)

    assert group_creation_response.status_code == 302
    assert group_creation_response.url == reverse("study_groups_list_for_user")
    assert StudyGroup.objects.filter(**study_group_creation_form_args).exists()
```

Tests - Study Group

```
@pytest.mark.djangoproject_db
class TestStudyGroupDetailView:
    def test_study_group_detail_view_loaded_authorized(self, authorized_client, persist_group):
        group_detail_response = authorized_client.get(f"/study-group/detail/{persist_group.pk}/")
        assert group_detail_response.status_code == 200
        assert 'study_group_detail.html' in group_detail_response.templates[0].name

    def test_study_group_detail_view_loaded_unauthorized(self, client, persist_group):
        study_group_list_response = client.get(f"/study-group/detail/{persist_group.pk}/")
        assert study_group_list_response.status_code == 302

    def test_display_edit_group_when_owner(self, client, persist_user, persist_group):
        client.force_login(persist_user)
        response = client.get(f"/study-group/detail/{persist_group.pk}/")
        assert b>Edit Group info" in response.content

    def test_not_display_edit_group_when_not_owner(self, client, persist_second_user, persist_group):
        client.force_login(persist_second_user)
        response = client.get(f"/study-group/detail/{persist_group.pk}/")
        assert b>Edit Group info" not in response.content
```

A demonstration of the application tests being run in GitHub actions

```
89%]
tests/study_group/test_study_group_views.py::TestStudyGroupUpdateView::test_study_group_update_view_loaded_authorized PASSED
[ 90%]
tests/study_group/test_study_group_views.py::TestStudyGroupUpdateView::test_update_group_details_not_owner[new field-this is a new group description] PASSED [ 91%]
tests/study_group/test_study_group_views.py::TestJoinLeaveGroupButton::test_leave_group PASSED [ 92%]
tests/study_group/test_study_group_views.py::TestJoinLeaveGroupButton::test_join_group PASSED [ 93%]
tests/study_group/test_study_group_views.py::TestJoinLeaveGroupButton::test_join_leave_non_existent_group PASSED [ 94%]
tests/study_group/test_study_group_views.py::TestJoinLeaveGroupButton::test_join_group_button_text[0-5] PASSED [ 95%]
tests/study_group/test_study_group_views.py::TestJoinLeaveGroupButton::test_full_group_button_text[1-1] PASSED [ 96%]
tests/study_group/test_study_group_views.py::TestJoinLeaveGroupButton::test_leave_group_text_button[1-2] PASSED [ 97%]
tests/study_group/test_study_group_views.py::TestStudyGroupListView::test_study_group_list_view_loaded_authorized PASSED [ 98%]
tests/study_group/test_study_group_views.py::TestStudyGroupListView::test_study_group_list_view_loaded_unauthorized PASSED [ 99%]
tests/study_group/test_study_group_views.py::TestStudyGroupListView::test_create_new_study_group[Test Group-Description-5] PASSED [100%]
```

Future Features

User Profile:

High.Teach

Sign Up Logout

David COH
Private Account

Feed

Search

Messages

My Courses

Study Groups

Home / User / User Profile

David_COH
personal tutor

Private Account Business Account

Website http://127.0.0.1:8000/
Github github.com/redhat-beyond/HighTeach

Full Name David Cohen

Email Davidco94@gmail.com

Phone 0502320055

About Me computer science student

Address Jaffo

Edit

subject Web App dev

Web Design

Website Markup

One Page

Mobile Template

Backend API

subject Algorithms

data structures

computability

Graph theory

complexity

Subject	Skills
Web App dev	Web Design (progress ~80%), Website Markup (progress ~70%), One Page (progress ~90%), Mobile Template (progress ~60%), Backend API (progress ~75%)
Algorithms	data structures (progress ~85%), computability (progress ~70%), Graph theory (progress ~90%), complexity (progress ~65%)

Future Features



Edit User Profile:

High.Teach

Logout

david_co
Business Account



Feed

Search

Messages

My Courses

Edit Profile

Username: david_co

Email: david@gmail.com

Bio: hello

Profession: student

City: Jaffo

Phone number: 0502220055

Account type: Teacher

Meeting method: Live and Online

Update

This screenshot shows the 'Edit User Profile' page of a platform called High.Teach. At the top, there's a navigation bar with 'High.Teach' on the left and 'Logout' on the right. Below the navigation bar, a blue sidebar on the left displays the user's profile picture, name (david_co), and account type (Business Account). The main content area is titled 'Edit Profile'. It contains several input fields for updating personal information: Username (david_co), Email (david@gmail.com), Bio (hello), Profession (student), City (Jaffo), Phone number (0502220055), Account type (set to Teacher), and Meeting method (Live and Online). At the bottom of the form is a blue 'Update' button.



Future Features



Sign Up:

High.Teach

Register Log In

Sign Up

Create New Profile



Username: Required. 150 characters or fewer. Letters, digits and @//+/-_ only.

Password:

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation: Enter the same password as before, for verification.

Bio:

Professional photo:

Profession:

City:

Phone number:

Account type:

Meeting method:

Future Features

Chat:

The screenshot shows a user interface for a messaging application. At the top, there's a green header bar with the text "High.Teach" on the left and "Logout" on the right. Below the header is a sidebar on the left containing account information ("teacher1 Private Account") and links to "Feed", "Search", "Messages", "My Courses", and "Study Groups". The main area is a conversation window. On the left side of the window, there's a search bar with the placeholder "Search...". A message from "teacher1" to "MEGAN LEIB" is displayed, showing the message "_TEST_MESSAGE" and a timestamp of "1 week". At the bottom of the window, there's a text input field with a smiley face icon and the placeholder "Type your message here".

Future Features

Feed:

The screenshot shows a user interface for a 'Feed' feature. On the left, there is a sidebar with a user profile icon and the name 'yarden' followed by 'Private Account'. Below this are five menu items: 'Feed', 'Search', 'Messages', 'My Courses', and 'Study Groups'. The 'Feed' item is highlighted with a green border. The main content area has a title 'Feed' and a sub-section 'New post:' with a large text input field labeled 'Content:'. Below this is a dropdown menu labeled 'Course id: -----' and a green 'Create' button. Two posts are visible in a list below: Post ID: 1 with the content 'fff' and Post ID: 2 with the content 'blala'.

Feed

New post:

Content:

Course id: -----

Create

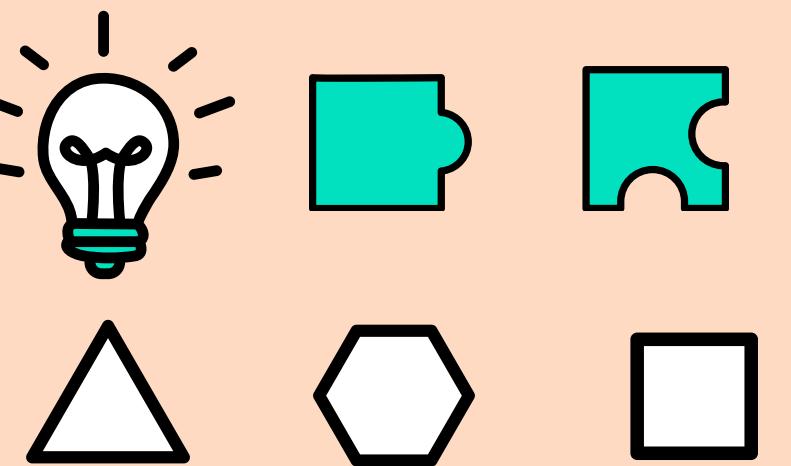
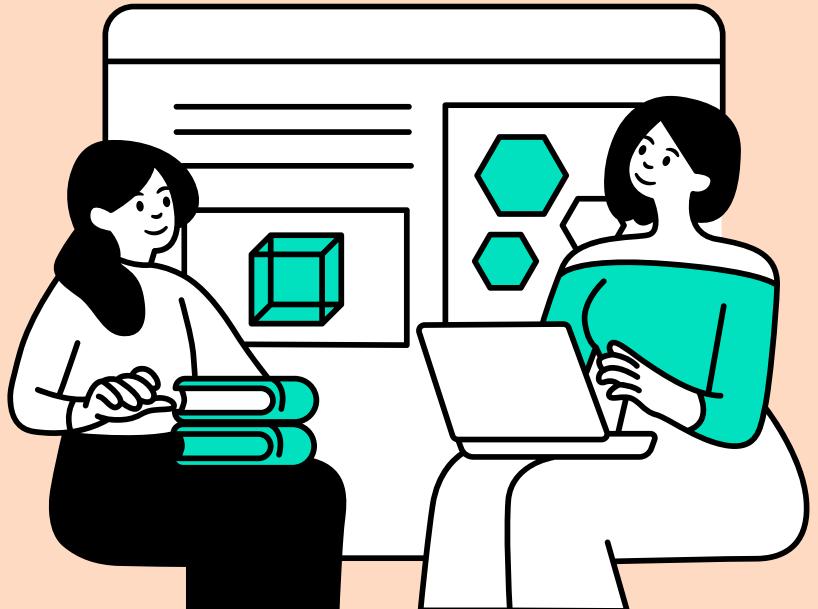
Post ID: 1
fff

Post ID: 2
blala

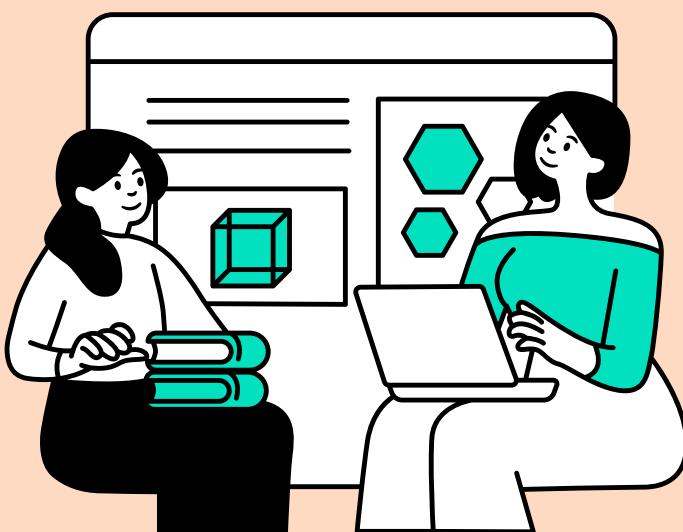
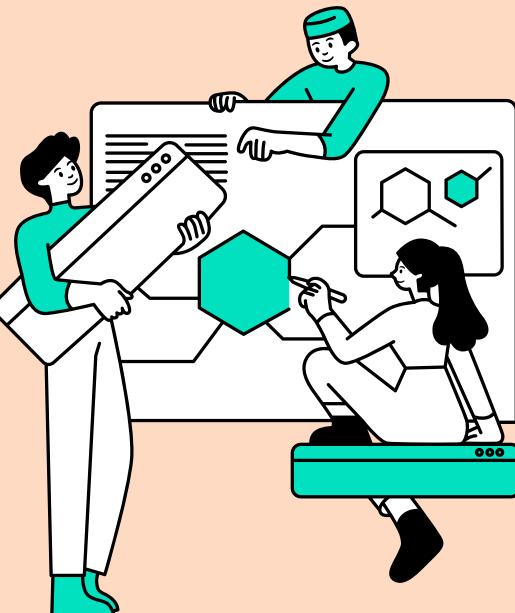
Git Repository

<https://github.com/redhat-beyond/HighTeach>





The road to success has never been easier...





Thank You!

Do you have any question?

