Unit Test

Authentication & Authorization

```
def test_register(client, init_database):
    response = client.post('/api/auth/register',
                           data=json.dumps(dict(
                               username='testuser',
                               email='test@example.com',
                               password='password'
                           )),
                           content_type='application/json')
    assert response.status_code == 201
    data = json.loads(response.data)
    assert data['username'] == 'testuser'
    assert data['email'] == 'test@example.com'
    assert 'id' in data

def test_login(client, init_database):
    client.post('/api/auth/register',
                data=json.dumps(dict(
                    username='testuser',
                    email='test@example.com',
                    password='password'
                )),
                content_type='application/json')

    response = client.post('/api/auth/login',
                           data=json.dumps(dict(
                               email='test@example.com',
                               password='password'
                           )),
                           content_type='application/json')
    assert response.status_code == 200
    data = json.loads(response.data)
    assert 'token' in data

def test_login_invalid_credentials(client, init_database):
    response = client.post('/api/auth/login',
                           data=json.dumps(dict(
                               email='wrong@example.com',
                               password='wrongpassword'
                           )),
                           content_type='application/json')
    assert response.status_code == 401
    data = json.loads(response.data)
    assert data['message'] == 'Invalid credentials'
```

Comment & Image

```
def test_register(client, init_database):
    response = client.post('/api/auth/register',
                           data=json.dumps(dict(
                               username='testuser',
                               email='test@example.com',
                               password='password'
                           )),
```

```
                            content_type='application/json')
    assert response.status_code == 201
    data = json.loads(response.data)
    assert data['username'] == 'testuser'
    assert data['email'] == 'test@example.com'
    assert 'id' in data

def test_login(client, init_database):
    client.post('/api/auth/register',
                data=json.dumps(dict(
                    username='testuser',
                    email='test@example.com',
                    password='password'
                )),
                content_type='application/json')

    response = client.post('/api/auth/login',
                           data=json.dumps(dict(
                               email='test@example.com',
                               password='password'
                           )),
                           content_type='application/json')
    assert response.status_code == 200
    data = json.loads(response.data)
    assert 'token' in data

def test_login_invalid_credentials(client, init_database):
    response = client.post('/api/auth/login',
                           data=json.dumps(dict(
                               email='wrong@example.com',
                               password='wrongpassword'
                           )),
                           content_type='application/json')
    assert response.status_code == 401
    data = json.loads(response.data)
    assert data['message'] == 'Invalid credentials'
```

Ticket

```
def get_auth_token(client):
    client.post('/api/auth/register',
                data=json.dumps(dict(
                    username='testuser',
                    email='test@example.com',
                    password='password'
                )),
                content_type='application/json')
    response = client.post('/api/auth/login',
                           data=json.dumps(dict(
                               email='test@example.com',
                               password='password'
                           )),
                           content_type='application/json')
    return json.loads(response.data)['token']

def test_create_ticket(client, init_database):
    token = get_auth_token(client)
    response = client.post('/api/tickets',
                           headers={'Authorization': f'Bearer {token}'},
                           data={
                               'title': 'Test Ticket',
```

```
                                'description': 'This is a test ticket.',
                                'images': (io.BytesIO(b"some initial text data"), 'test.jpg')
                            },
                            content_type='multipart/form-data')
    assert response.status_code == 201
    data = json.loads(response.data)
    assert data['title'] == 'Test Ticket'
    assert data['description'] == 'This is a test ticket.'
    assert 'id' in data
    assert 'image_ids' in data
    assert len(data['image_ids']) == 1

def test_get_tickets(client, init_database):
    token = get_auth_token(client)
    client.post('/api/tickets',
                headers={'Authorization': f'Bearer {token}'},
                data={
                    'title': 'Test Ticket',
                    'description': 'This is a test ticket.'
                },
                content_type='multipart/form-data')

    response = client.get('/api/tickets', headers={'Authorization': f'Bearer {token}'})
    assert response.status_code == 200
    data = json.loads(response.data)
    assert isinstance(data, list)
    assert len(data) == 1
    assert data[0]['title'] == 'Test Ticket'

def test_get_ticket(client, init_database):
    token = get_auth_token(client)
    create_response = client.post('/api/tickets',
                                  headers={'Authorization': f'Bearer {token}'},
                                  data={
                                      'title': 'Test Ticket',
                                      'description': 'This is a test ticket.'
                                  },
                                  content_type='multipart/form-data')
    ticket_id = json.loads(create_response.data)['id']

    response = client.get(f'/api/tickets/{ticket_id}', headers={'Authorization': f'Bearer
{token}'})
    assert response.status_code == 200
    data = json.loads(response.data)
    assert data['title'] == 'Test Ticket'
    assert data['id'] == ticket_id

def test_update_ticket(client, init_database):
    client.post('/api/auth/register',
                data=json.dumps(dict(
                    username='adminuser',
                    email='admin@example.com',
                    password='password'
                )),
                content_type='application/json')

    with client.application.app_context():
        admin_user = Users.query.filter_by(email='admin@example.com').first()
        admin_user.role = 'admin'
        db.session.commit()

    response = client.post('/api/auth/login',
```

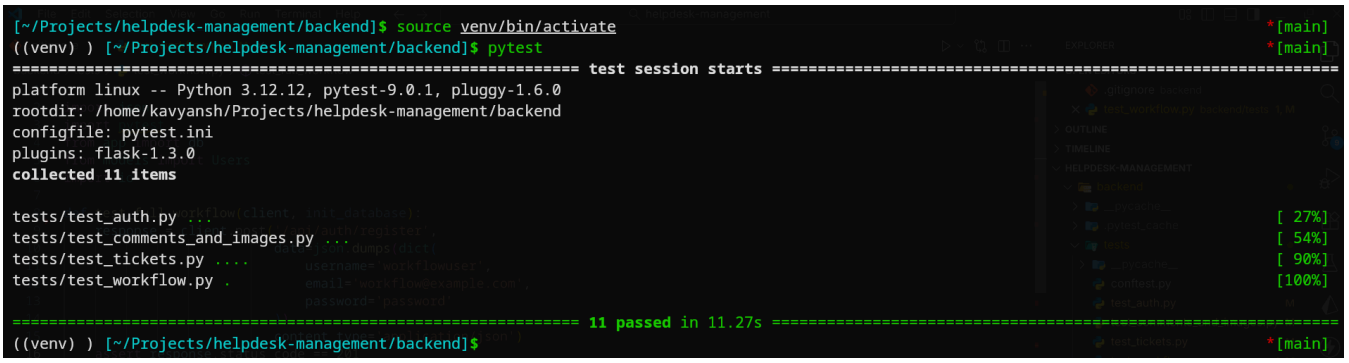
```
                            data=json.dumps(dict(
                                email='admin@example.com',
                                password='password'
                            )),
                            content_type='application/json')
    admin_token = json.loads(response.data)['token']

    user_token = get_auth_token(client)
    create_response = client.post('/api/tickets',
                                  headers={'Authorization': f'Bearer {user_token}'},
                                  data={
                                      'title': 'Test Ticket',
                                      'description': 'This is a test ticket.'
                                  },
                                  content_type='multipart/form-data')
    ticket_id = json.loads(create_response.data)['id']

    response = client.put(f'/api/tickets/{ticket_id}',
                          headers={'Authorization': f'Bearer {admin_token}'},
                          data=json.dumps(dict(status='closed')),
                          content_type='application/json')
    assert response.status_code == 200
    data = json.loads(response.data)
    assert data['status'] == 'closed'
```

System & Integration Testing

```
def test_full_workflow(client, init_database):
    response = client.post('/api/auth/register',
                           data=json.dumps(dict(
                               username='workflowuser',
                               email='workflow@example.com',
                               password='password'
                           )),
                           content_type='application/json')
    assert response.status_code == 201
    user_data = json.loads(response.data)
    user_id = user_data['id']

    response = client.post('/api/auth/login',
                           data=json.dumps(dict(
                               email='workflow@example.com',
                               password='password'
                           )),
                           content_type='application/json')
    assert response.status_code == 200
    user_token = json.loads(response.data)['token']

    response = client.post('/api/tickets',
                           headers={'Authorization': f'Bearer {user_token}'},
                           data={
                               'title': 'Workflow Ticket',
                               'description': 'This is a ticket created during a workflow
test.',
                               'images': (io.BytesIO(b'workflow image'), 'workflow.jpg')
                           },
                           content_type='multipart/form-data')
    assert response.status_code == 201
    ticket_data = json.loads(response.data)
    ticket_id = ticket_data['id']
    assert ticket_data['title'] == 'Workflow Ticket'
```

```
    assert len(ticket_data['image_ids']) == 1
    image_id = ticket_data['image_ids'][0]

    response = client.post(f'/api/tickets/{ticket_id}/comments',
                           headers={'Authorization': f'Bearer {user_token}'},
                           data=json.dumps(dict(text='This is a workflow comment.')),
                           content_type='application/json')
    assert response.status_code == 201
    comment_data = json.loads(response.data)
    assert comment_data['text'] == 'This is a workflow comment.'

    response = client.post('/api/auth/register',
                           data=json.dumps(dict(
                               username='workflowadmin',
                               email='workflowadmin@example.com',
                               password='password'
                           )),
                           content_type='application/json')
    assert response.status_code == 201

    with client.application.app_context():
        admin_user = Users.query.filter_by(email='workflowadmin@example.com').first()
        admin_user.role = 'admin'
        db.session.commit()

    response = client.post('/api/auth/login',
                           data=json.dumps(dict(
                               email='workflowadmin@example.com',
                               password='password'
                           )),
                           content_type='application/json')
    assert response.status_code == 200
    admin_token = json.loads(response.data)['token']

    response = client.get(f'/api/tickets/{ticket_id}', headers={'Authorization': f'Bearer
{admin_token}'})
    assert response.status_code == 200
    assert json.loads(response.data)['title'] == 'Workflow Ticket'

    response = client.put(f'/api/tickets/{ticket_id}',
                          headers={'Authorization': f'Bearer {admin_token}'},
                          data=json.dumps(dict(status='in_progress')),
                          content_type='application/json')
    assert response.status_code == 200
    assert json.loads(response.data)['status'] == 'in_progress'

    response = client.get(f'/api/tickets/{ticket_id}', headers={'Authorization': f'Bearer
{user_token}'})
    assert response.status_code == 200
    assert json.loads(response.data)['status'] == 'in_progress'

    response = client.get(f'/api/tickets/{ticket_id}/comments', headers={'Authorization':
f'Bearer {admin_token}'})
    assert response.status_code == 200
    assert len(json.loads(response.data)) == 1

    response = client.get(f'/api/images/{image_id}')
    assert response.status_code == 200
    assert response.data == b'workflow image'
```

Output


[~/Projects/helpdesk-management/backend]$ source venv/bin/activate *[main]
((venv)) [~/Projects/helpdesk-management/backend]$ pytest *[main]
test session starts
platform linux -- Python 3.12.12, pytest-9.0.1, pluggy-1.6.0
rootdir: /home/kavyansh/Projects/helpdesk-management/backend
configfile: pytest.ini
plugins: flask-1.3.0
collected 11 items
tests/test_auth.py ...
tests/test_comments_and_images.py ...
tests/test_tickets.py ....
tests/test_workflow.py .
[27%]
[54%]
[90%]
[100%]
11 passed in 11.27s
((venv)) [~/Projects/helpdesk-management/backend]$ *[main]