ДОДАТОК А

Компонентно-функціональна структура для користувачів

Таблиця А.1 – Компонентно-функціональна структура для користувачів

Мета	Задачі	Процедури	Дії користувачів і системи
1	2	3	4
1			Повторення карток
	1.1		Отримати картку та відповісти на неї
		1.1.1	Система обчислює час до відправлення наступного повідомлення
		1.1.2	Користувач дає відповідь
			буде повторно відправлена через встановлений мінімальний інтервал і повернутися в пункт 1.1.1
		1.1.3	Користувачу виводиться правильна відповідь
		1.1.4	Користувач дає собі оцінку упродовж 15 секунд, інакше картка буде повторно відправлена через встановлений мін. інтервал і повернутися в пункт 1.1.1
		1.1.5	Система рахує оцінку
		1.1.6	Система перераховує час до відправки картки на основі виданої оцінки та поновлює план і повернутися в пункт 1.1.1
2			Перегляд карток у календарному плані

Продовження таблиці А.1

1	2	3	4
	2.1		Відображення циклічного (по колу) переліку карток
		2.1.1	Система отримує перелік карток з бази
		2.1.2	Система виставляє порожню картку (слугує для додавання нової картки та являє собою межу між зацикленим початком та кінцем календарного плану) у якості цільової картки
		2.1.3	Система поновлює календарний план
	2.2		Перегляд додаткової інформації та елементів управління по картці
		2.2.1	Користувач знаходить потрібну картку через скролінг по календарному плану
		2.2.2	Система виставляє обрану картку як цільову
		2.2.3	Система виводить додаткову інформацію та інструменти для управління карткою
	2.3		Відобразити картки за обраний проміжок часу
		2.3.1	Користувач обирає фільтрацію за проміжком часу
		2.3.2	Користувач переглядає календар та обирає початкову та кінцеві дати

Продовження таблиці А.1

1	2	3	4
		2.3.3	Система отримує дані про заплановані повторення та фільтрує їх за встановленим діапазоном
	2.4		Відобразити n карток одночасно
		2.4.1	Користувач обирає максимальну кількість карток, яка виводиться одночасно
		2.4.2	Система поновлює календарний план
3			Управління картками
	3.1		Додати нову картку
		3.1.1	Користувач вводить дані у відповідне поле при умові, що цільова картка не обрана, інакше користувач натискає на кнопку додавання і перейти в пункт 2.1.2
		3.1.2	Система отримує переклад та додаткову інформацію зі сторонніх сервісів
		3.1.3	Користувач додає картку у початок чи кінець календарного плану завдяки відповідним кнопкам або вказаним комбінаціям клавіш
	3.2		Редагувати картку
		3.2.1	Користувач оновлює інформацію у цільовій картці

Продовження таблиці А.1

1	2	3	4
		3.2.2	Система отримує переклад та додаткову інформацію зі сторонніх сервісів
		3.2.3	Користувач зберігає оновлену картку
	3.3		Видалити картку
		3.3.1	Користувач видаляє цільову картку
		3.3.2	Система видаляє дані з бази

ДОДАТОК Б

Реалізація прототипу DDPG агенту, його середовища, тренування та тестів

```
import gymnasium as gym
from gymnasium import spaces
import numpy as np

def fsrs_mock(state):
    """Моделює FSRS, розраховує стабільність, складність та оптимальний інтервал."""
    stability = 5.0 + state['review_history_success'] * 2.0 - state['review_history_fail'] * 1.5
    difficulty = 3.0 - state['review_history_success'] * 0.5 + state['review_history_fail'] * 1.0
    return {
        'stability': max(1.0, stability),
        'difficulty': max(1.0, difficulty),
        'optimal_interval': max(0.01, stability * (np.power(0.9, -1) - 1))
    }

def simulate_user_recall(agent_interval, stability):
    """Симулює ймовірність пригадування користувачем."""
    recall_probability = np.power(1 + agent_interval / (9 * stability), -1)
    return np.random.rand() < recall_probability, recall_probability

class SpacedRepetitionEnv(gym.Env):

    metadata = {'render_modes': ['console']}

    def __init__(self, render_mode=None):
        super().__init__()
        self.render_mode = render_mode
```

```
        self.action_space    =    spaces.Box(low=-1,    high=1,
shape=(1,), dtype=np.float32)

        low = np.zeros(11, dtype=np.float32)
        high = np.array([1.0, 365.0, 10.0, 365.0, 50.0, 50.0,
365.0, 1.0, 1.0, 200.0, 1.0], dtype=np.float32)
        self.observation_space = spaces.Box(low=low, high=high,
dtype=np.float32)

        self.min_interval = 1 / (24 * 60)
        self.max_interval = 180

        self.max_episode_steps = 50
        self.current_step = 0

    def _generate_random_state(self):
        """Генерує стан для нової картки, що прийшла на
повторення."""
        return {
            'time_since_last_review':    np.random.uniform(0.1,
90.0),
            'review_history_success': np.random.randint(0, 15),
            'review_history_fail': np.random.randint(0, 5),
            'mean_interval': np.random.uniform(1.0, 30.0),
            'user_activity': np.random.randint(0, 2),
            'noise_level': np.random.rand(),
            'session_items': self.current_step,
            'session_time':         self.current_step         /
self.max_episode_steps
        }

    def _get_obs(self):
        """Створює вектор спостереження на основі поточного
стану картки."""
        fsrs = fsrs_mock(self.card_state)
```

```
        _,                  recall_prob_proxy                  =
simulate_user_recall(self.card_state['time_since_last_review']
, fsrs['stability'])

        return np.array([recall_prob_proxy, fsrs['stability'],
fsrs['difficulty'],

self.card_state['time_since_last_review'],
self.card_state['review_history_success'],

self.card_state['review_history_fail'],
self.card_state['mean_interval'],
                        self.card_state['user_activity'],
self.card_state['noise_level'],
                        self.card_state['session_items'],
self.card_state['session_time']], dtype=np.float32)

    def reset(self, seed=None, options=None):
        """Скидає середовище на початок нової сесії."""
        super().reset(seed=seed)
        self.current_step = 0
        self.card_state = self._generate_random_state()
        return self._get_obs(), {}

    def step(self, action):
        """Виконує один крок (повторення) в рамках сесії."""
        low, high = -1, 1
        log_min = np.log(self.min_interval)
        log_max = np.log(self.max_interval)
        scaled_action_log = log_min + (action[0] - low) *
(log_max - log_min) / (high - low)
        agent_interval = np.exp(scaled_action_log)

        fsrs = fsrs_mock(self.card_state)
        recalled,               recall_prob                =
simulate_user_recall(agent_interval, fsrs['stability'])
```

```
        if recalled:
            # Винагорода пропорційна довжині інтервалу, але
нормалізована до діапазону [0, 1]
            reward = agent_interval / self.max_interval
        else:
            # Штраф за забування
            reward = -1.0

        self.current_step += 1
        self.card_state = self._generate_random_state()
        observation = self._get_obs()
        terminated        =        self.current_step        >=
self.max_episode_steps

        return observation, reward, terminated, False, {}

import os
import numpy as np
from datetime import datetime
from stable_baselines3 import DDPG
from stable_baselines3.common.noise import NormalActionNoise
from stable_baselines3.common.env_checker import check_env
from stable_baselines3.common.callbacks import BaseCallback

# from spaced_repetition_env import SpacedRepetitionEnv

log_dir = "tensorboard_logs/"
os.makedirs(log_dir, exist_ok=True)
# os.makedirs(model_dir, exist_ok=True)

class TensorboardCallback(BaseCallback):
    def _on_step(self) -> bool:
        info = self.locals['infos'][0]
        if info and 'reward_breakdown' in info:
```

```
            self.logger.record("rewards/total",
info['reward_breakdown']['total'])
            self.logger.record("rewards/forget_penalty",
info['reward_breakdown']['forget_penalty'])
            self.logger.record("rewards/early_penalty",
info['reward_breakdown']['early_penalty'])
        return True


env = SpacedRepetitionEnv(render_mode='console')
# check_env(env)
print("Среда успішно створена та перевірена.")


n_actions = env.action_space.shape[-1]
action_noise   =   NormalActionNoise(mean=np.zeros(n_actions),
sigma=0.4 * np.ones(n_actions))

model = DDPG(
    "MlpPolicy",
    env,
    action_noise=action_noise,
    verbose=1,
    tensorboard_log=log_dir,
    learning_rate=1e-3,
    buffer_size=500000,
    learning_starts=0,
    batch_size=256,
    tau=0.005,
    gamma=0.99,
    train_freq=(1, "step"),
    gradient_steps=1,
)


# 4. Навчання моделі
```

```
total_timesteps = 10000
print(f"\n--- Початок навчання на {total_timesteps} кроках ---
")
model.learn(
    total_timesteps=total_timesteps,
    callback=TensorboardCallback(),
    log_interval=10
)
print("--- Навчання завершено ---")


# 5. Збереження моделі
model_dir = "saved_models"
os.makedirs(model_dir, exist_ok=True)

timestamp = datetime.now().strftime("%Y-%m-%d_%H%M%S")
model_basename = "ddpg_spaced_repetition"
model_filename = f"{model_basename}_{timestamp}.zip"
model_path = os.path.join(model_dir, model_filename)

model.save(model_path)

print(f"Модель збережена у: {model_path}")

env.close()

import numpy as np
from stable_baselines3 import DDPG
import os
import sys
import glob

MIN_INTERVAL = 1 / (24 * 60) # 1 хвилина
MAX_INTERVAL = 180  # ~6 місяців
MODEL_DIR = "saved_models"
MODEL_BASENAME = "ddpg_spaced_repetition"
```

```
def scale_action_to_interval(action: np.ndarray) -> float:
    """
    Масштабує дію моделі (від -1 до 1) до інтервалу повторення у днях,
    використовуючи логарифмічну шкалу для кращої чутливості.
    """
    low, high = -1, 1
    log_min = np.log(MIN_INTERVAL)
    log_max = np.log(MAX_INTERVAL)

    scaled_action_log = log_min + (action[0] - low) * (log_max - log_min) / (high - low)

    return np.exp(scaled_action_log)


def get_test_data() -> list[dict]:
    """
    Повертає список структурованих тестових випадків. Кожен випадок - це словник,
    що містить опис та відповідні дані спостереження (стану).
    """

    # 0: Ймовірність пригадування (ВКТ)
    # 1: Стабільність пам'яті (S) (FSRS)
    # 2: Складність елемента (D) (FSRS)
    # 3: Час з останнього повторення (днів)
    # 4: Історія: К-сть успішних повторень
    # 5: Історія: К-сть невдалих повторень
    # 6: Історія: Середній попередній інтервал (днів)
    # 7: Контекст: Поточна активність (0: відпочинок, 1: робота)
    # 8: Контекст: Рівень шуму (0.0 - тихо, 1.0 - гучно)
    # 9: Сесія: К-сть елементів у сесії
```

```
    # 10: Сесія: Час у поточній сесії (0.0 - початок, 1.0 - кінець)

    return [
        {
            "description": "1. Новий елемент, щойно вивчений, користувач відпочиває у тиші",
            "observation": np.array([0.95, 5.0, 3.0, 0.1, 1, 0, 0.1, 0, 0.1, 5, 0.1], dtype=np.float32)
        },
        {
            "description": "2. Той же новий елемент, але користувач на роботі у шумному офісі",
            "observation": np.array([0.95, 5.0, 3.0, 0.1, 1, 0, 0.1, 1, 0.8, 25, 0.5], dtype=np.float32)
        },
        {
            "description": "3. 'Сильний' елемент (багато успіхів), давно не повторювався",
            "observation": np.array([0.8, 45.0, 2.0, 30.0, 15, 0, 25.0, 0, 0.2, 10, 0.2], dtype=np.float32)
        },
        {
            "description": "4. 'Складний' елемент: були невдачі, стабільність низька",
            "observation": np.array([0.7, 8.0, 7.0, 5.0, 4, 3, 4.0, 1, 0.7, 40, 0.8], dtype=np.float32)
        },
        {
            "description": "5. Дуже старий елемент, майже забутий (низька ймовірність ВКТ)",
            "observation": np.array([0.2, 20.0, 4.0, 90.0, 8, 1, 40.0, 0, 0.3, 15, 0.4], dtype=np.float32)
        },
        {
```

```
            "description": "6. Елемент середньої сили, але користувач втомився (кінець довгої сесії)",
            "observation": np.array([0.85, 15.0, 3.5, 10.0, 5, 0, 8.0, 0, 0.2, 150, 0.95], dtype=np.float32)
        },
        {
            "description": "7. Ситуація з попереднього тесту, але користувач тільки почав сесію (свіжий)",
            "observation": np.array([0.85, 15.0, 3.5, 10.0, 5, 0, 8.0, 0, 0.2, 10, 0.1], dtype=np.float32)
        },
        {
            "description": "8. Елемент з історією невдач, але зараз ймовірність пригадування висока",
            "observation": np.array([0.9, 10.0, 6.0, 4.0, 8, 4, 3.0, 1, 0.5, 33, 0.5], dtype=np.float32)
        },
        {
            "description": "9. Дуже простий елемент, який завжди успішно повторювався",
            "observation": np.array([0.95, 60.0, 1.0, 45.0, 20, 0, 40.0, 0, 0.1, 20, 0.3], dtype=np.float32)
        },
        {
            "description": "10. Гіпотетична ситуація одразу після невдачі",
            "observation": np.array([0.1, 5.0, 4.0, 0.1, 1, 1, 0.1, 1, 0.9, 50, 0.9], dtype=np.float32)
        },
    ]


def find_latest_model_path() -> str | None:
    """
    Знаходить шлях до останнього збереженого файлу моделі в директорії.
```

```
    """
    search_pattern = os.path.join(MODEL_DIR,
f"{MODEL_BASENAME}_*.zip")
    model_files = glob.glob(search_pattern)

    if not model_files:
        return None

    # Знайти найновіший файл за часом створення
    latest_model = max(model_files, key=os.path.getctime)
    return latest_model

def format_interval(days: float) -> str:
    """
    Перетворює інтервал у днях на зручний для читання формат
(хвилини, години, дні).
    """
    if days < 1 / 24:
        minutes = days * 24 * 60
        return f"{minutes:.1f} хвилин(а)"
    elif days < 1:
        hours = days * 24
        return f"{hours:.1f} годин(а)"
    else:
        return f"{days:.2f} днів"


def main():
    """
    Головна функція для завантаження моделі та виконання
прогнозів на тестових даних.
    """

    # 1. Пошук та завантаження останньої моделі
    print("Пошук останньої збереженої моделі...")
    latest_model_path = find_latest_model_path()
```

```
    if not latest_model_path:
        print(f"Помилка: Моделі не знайдено в директорії '{MODEL_DIR}'", file=sys.stderr)
        print("Будь ласка, спочатку запустіть скрипт навчання, щоб зберегти модель.", file=sys.stderr)
        sys.exit(1)

    print(f"Завантаження моделі: {os.path.basename(latest_model_path)}")
    try:
        model = DDPG.load(latest_model_path)
        print("Модель успішно завантажено.\n")
    except Exception as e:
        print(f"Сталася помилка під час завантаження моделі: {e}", file=sys.stderr)
        sys.exit(1)

    # 2. Отримання тестових даних
    test_data = get_test_data()

    # 3. Отримання та відображення прогнозів
    print("Результати тестування")
    for case in test_data:
        observation = case["observation"]
        description = case["description"]

        action, _ = model.predict(observation, deterministic=True)

        predicted_interval = scale_action_to_interval(action)

        print(f"## {description}")
        print(f"   Вхідний стан: {observation}")
        print(f"              => Прогнозований інтервал: {format_interval(predicted_interval)}\n")
```

ДОДАТОК В

docker-compose.yml

```
---
services:


  broker:
    image: confluentinc/cp-server:7.9.0
    hostname: broker
    container_name: broker
    ports:
      - "9092:9092"
      - "9101:9101"
    environment:
      KAFKA_NODE_ID: 1
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP:
'CONTROLLER:PLAINTEXT,PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAIN
TEXT'
      KAFKA_ADVERTISED_LISTENERS:
'PLAINTEXT://broker:29092,PLAINTEXT_HOST://localhost:9092'
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
      KAFKA_GROUP_INITIAL_REBALANCE_DELAY_MS: 0
      KAFKA_CONFLUENT_LICENSE_TOPIC_REPLICATION_FACTOR: 1
      KAFKA_CONFLUENT_BALANCER_TOPIC_REPLICATION_FACTOR: 1
      KAFKA_TRANSACTION_STATE_LOG_MIN_ISR: 1
      KAFKA_TRANSACTION_STATE_LOG_REPLICATION_FACTOR: 1
      KAFKA_JMX_PORT: 9101
      KAFKA_JMX_HOSTNAME: localhost
      KAFKA_CONFLUENT_SCHEMA_REGISTRY_URL: http://schema-
registry:8081
      KAFKA_METRIC_REPORTERS:
io.confluent.metrics.reporter.ConfluentMetricsReporter
      CONFLUENT_METRICS_REPORTER_BOOTSTRAP_SERVERS:
broker:29092
```

```
      CONFLUENT_METRICS_REPORTER_TOPIC_REPLICAS: 1
      KAFKA_PROCESS_ROLES: 'broker,controller'
      KAFKA_CONTROLLER_QUORUM_VOTERS: '1@broker:29093'
      KAFKA_LISTENERS:
'PLAINTEXT://broker:29092,CONTROLLER://broker:29093,PLAINTEXT_
HOST://0.0.0.0:9092'
      KAFKA_INTER_BROKER_LISTENER_NAME: 'PLAINTEXT'
      KAFKA_CONTROLLER_LISTENER_NAMES: 'CONTROLLER'
      KAFKA_LOG_DIRS: '/tmp/kraft-combined-logs'
      CONFLUENT_METRICS_ENABLE: 'true'
      CONFLUENT_SUPPORT_CUSTOMER_ID: 'anonymous'
      # Replace CLUSTER_ID with a unique base64 UUID using
"bin/kafka-storage.sh random-uuid"
      # See https://docs.confluent.io/kafka/operations-
tools/kafka-tools.html#kafka-storage-sh
      CLUSTER_ID: 'MkU3OEVBNTcwNTJENDM2Qk'


  schema-registry:
    image: confluentinc/cp-schema-registry:7.9.0
    hostname: schema-registry
    container_name: schema-registry
    depends_on:
      - broker
    ports:
      - "8081:8081"
    environment:
      SCHEMA_REGISTRY_HOST_NAME: schema-registry
      SCHEMA_REGISTRY_KAFKASTORE_BOOTSTRAP_SERVERS:
'broker:29092'
      SCHEMA_REGISTRY_LISTENERS: http://0.0.0.0:8081


  connect:
    image: cnfldemos/cp-server-connect-datagen:0.6.4-7.6.0
    hostname: connect
```

```
    container_name: connect
    depends_on:
      - broker
      - schema-registry
    ports:
      - "8083:8083"
    environment:
      CONNECT_BOOTSTRAP_SERVERS: 'broker:29092'
      CONNECT_REST_ADVERTISED_HOST_NAME: connect
      CONNECT_GROUP_ID: compose-connect-group
      CONNECT_CONFIG_STORAGE_TOPIC: docker-connect-configs
      CONNECT_CONFIG_STORAGE_REPLICATION_FACTOR: 1
      CONNECT_OFFSET_FLUSH_INTERVAL_MS: 10000
      CONNECT_OFFSET_STORAGE_TOPIC: docker-connect-offsets
      CONNECT_OFFSET_STORAGE_REPLICATION_FACTOR: 1
      CONNECT_STATUS_STORAGE_TOPIC: docker-connect-status
      CONNECT_STATUS_STORAGE_REPLICATION_FACTOR: 1
      CONNECT_KEY_CONVERTER:
org.apache.kafka.connect.storage.StringConverter
      CONNECT_VALUE_CONVERTER:
io.confluent.connect.avro.AvroConverter
      CONNECT_VALUE_CONVERTER_SCHEMA_REGISTRY_URL:
http://schema-registry:8081
      # CLASSPATH required due to CC-2422
      CLASSPATH: /usr/share/java/monitoring-
interceptors/monitoring-interceptors-7.9.0.jar
      CONNECT_PRODUCER_INTERCEPTOR_CLASSES:
"io.confluent.monitoring.clients.interceptor.MonitoringProduce
rInterceptor"
      CONNECT_CONSUMER_INTERCEPTOR_CLASSES:
"io.confluent.monitoring.clients.interceptor.MonitoringConsume
rInterceptor"
      CONNECT_PLUGIN_PATH:
"/usr/share/java,/usr/share/confluent-hub-components"
```

```
  control-center:
    image: confluentinc/cp-enterprise-control-center:7.9.0
    hostname: control-center
    container_name: control-center
    depends_on:
      - broker
      - schema-registry
      - connect
      - ksqldb-server
    ports:
      - "9021:9021"
    environment:
      CONTROL_CENTER_BOOTSTRAP_SERVERS: 'broker:29092'
      CONTROL_CENTER_CONNECT_CONNECT-DEFAULT_CLUSTER:
'connect:8083'
      CONTROL_CENTER_CONNECT_HEALTHCHECK_ENDPOINT:
'/connectors'
      CONTROL_CENTER_KSQL_KSQLDB1_URL: "http://ksqldb-
server:8088"
      CONTROL_CENTER_KSQL_KSQLDB1_ADVERTISED_URL:
"http://localhost:8088"
      CONTROL_CENTER_SCHEMA_REGISTRY_URL: "http://schema-
registry:8081"
      CONTROL_CENTER_REPLICATION_FACTOR: 1
      CONTROL_CENTER_INTERNAL_TOPICS_PARTITIONS: 1
      CONTROL_CENTER_MONITORING_INTERCEPTOR_TOPIC_PARTITIONS:
1
      CONFLUENT_METRICS_TOPIC_REPLICATION: 1
      PORT: 9021


  ksqldb-server:
    image: confluentinc/cp-ksqldb-server:7.9.0
    hostname: ksqldb-server
    container_name: ksqldb-server
```

```
    depends_on:
      - broker
      - connect
    ports:
      - "8088:8088"
    environment:
      KSQL_CONFIG_DIR: "/etc/ksql"
      KSQL_BOOTSTRAP_SERVERS: "broker:29092"
      KSQL_HOST_NAME: ksqldb-server
      KSQL_LISTENERS: "http://0.0.0.0:8088"
      KSQL_CACHE_MAX_BYTES_BUFFERING: 0
      KSQL_KSQL_SCHEMA_REGISTRY_URL: "http://schema-
registry:8081"
      KSQL_PRODUCER_INTERCEPTOR_CLASSES:
"io.confluent.monitoring.clients.interceptor.MonitoringProduce
rInterceptor"
      KSQL_CONSUMER_INTERCEPTOR_CLASSES:
"io.confluent.monitoring.clients.interceptor.MonitoringConsume
rInterceptor"
      KSQL_KSQL_CONNECT_URL: "http://connect:8083"
      KSQL_KSQL_LOGGING_PROCESSING_TOPIC_REPLICATION_FACTOR: 1
      KSQL_KSQL_LOGGING_PROCESSING_TOPIC_AUTO_CREATE: 'true'
      KSQL_KSQL_LOGGING_PROCESSING_STREAM_AUTO_CREATE: 'true'


  ksqldb-cli:
    image: confluentinc/cp-ksqldb-cli:7.9.0
    container_name: ksqldb-cli
    depends_on:
      - broker
      - connect
      - ksqldb-server
    entrypoint: /bin/sh
    tty: true
```

```
  ksql-datagen:
    image: confluentinc/ksqldb-examples:7.9.0
    hostname: ksql-datagen
    container_name: ksql-datagen
    depends_on:
      - ksqldb-server
      - broker
      - schema-registry
      - connect
    command: "bash -c 'echo Waiting for Kafka to be ready...
&& \
                        cub kafka-ready -b broker:29092 1 40 &&
\
                        echo Waiting for Confluent Schema
Registry to be ready... && \
                        cub sr-ready schema-registry 8081 40 &&
\
                        echo Waiting a few seconds for topic
creation to finish... && \
                        sleep 11 && \
                        tail -f /dev/null'"
    environment:
      KSQL_CONFIG_DIR: "/etc/ksql"
      STREAMS_BOOTSTRAP_SERVERS: broker:29092
      STREAMS_SCHEMA_REGISTRY_HOST: schema-registry
      STREAMS_SCHEMA_REGISTRY_PORT: 8081


  rest-proxy:
    image: confluentinc/cp-kafka-rest:7.9.0
    depends_on:
      - broker
      - schema-registry
    ports:
      - 8082:8082
```

```
    hostname: rest-proxy
    container_name: rest-proxy
    environment:
      KAFKA_REST_HOST_NAME: rest-proxy
      KAFKA_REST_BOOTSTRAP_SERVERS: 'broker:29092'
      KAFKA_REST_LISTENERS: "http://0.0.0.0:8082"
      KAFKA_REST_SCHEMA_REGISTRY_URL: 'http://schema-
registry:8081'


  flink-sql-client:
    image: cnfldemos/flink-sql-client-kafka:1.19.1-scala_2.12-
java17
    depends_on:
      - flink-jobmanager
    hostname: flink-sql-client
    container_name: flink-sql-client
    environment:
      FLINK_JOBMANAGER_HOST: flink-jobmanager


  flink-jobmanager:
    image: cnfldemos/flink-kafka:1.19.1-scala_2.12-java17
    hostname: flink-jobmanager
    container_name: flink-jobmanager
    ports:
    - 9081:9081
    command: jobmanager
    environment:
    - |
      FLINK_PROPERTIES=
      jobmanager.rpc.address: flink-jobmanager
      rest.bind-port: 9081


  flink-taskmanager:
```

```
image: cnfldemos/flink-kafka:1.19.1-scala_2.12-java17
hostname: flink-taskmanager
container_name: flink-taskmanager
depends_on:
- flink-jobmanager
command: taskmanager
scale: 1
environment:
- |
  FLINK_PROPERTIES=
  jobmanager.rpc.address: flink-jobmanager
  taskmanager.numberOfTaskSlots: 10
```

ДОДАТОК Г

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	125 с.
	Інші документи	
2.	Презентаційні матеріали	__ плакатів

.	.	Прізвище та ініціали.	Підп.	Дата			
Розробив		Свєркунов В.Р.			Застосування інтелектуальних агентів для адаптивного фонового вивчення мов	Шифр групи	Код напр./спец.
Перевірив		Мірошниченко Т. О.				ІТШІ-21-2	122
Н.контр.		Малєєва І.А.				ХНУРЕ кафедра ШІ	
Затв.		Золотухін О.В.					