ChatGPT

Доповідь і презентація для захисту бакалаврської роботи про мультиагентну симуляцію міського середовища

Executive summary

Для захисту на 5–7 хвилин оптимально не намагатися показати всю роботу як повний конспект пояснювальної записки, а побудувати виступ навколо одного сильного наративу: **місто як середовище, агенти як пояснювані цифрові мешканці, а симуляція як інструмент для відтворюваного аналізу поведінки**. У вашій КРБ найсильніші сторони — це прозора математична модель вибору дій, A*-маршрутизація, відтворюваний timeline, decision logs, SQLite-сховище метрик, а також аналітичні графіки, що показують не лише “рух точок”, а поведінкові патерни. Саме це і варто поставити в центр презентації. Практична цінність тут теж дуже сильна: система є локально розгортаним MVP, придатним для навчання, демонстрації, експериментів зі сценаріями та аналізу наслідків локальних правил на рівні всієї міської моделі. filecite turn0file2

Стиль подачі варто взяти не з класичних “технологічних” презентацій, а з наукових робіт на кшталт **MobileCity**, **CitySim** і **AgentSociety**: менше дрібного тексту, більше акценту на механізмі поведінки, архітектурі, графіках і тому, що саме система дозволяє дослідити. При цьому важливо чітко проговорити, що у вашій роботі **не використовується LLM**: рішення агентів формуються не мовною моделлю, а числовою потребнісною моделлю, persona-рисами, розкладом, правилами часу доби, utility-оцінкою та softmax-вибором. Це одразу знімає можливу плутанину комісії. filecite turn0file2 filecite turn0file1 filecite turn0file0

Нижче подано **основний план на 7 слайдів** і **один резервний слайд**. Якщо захист іде швидко, показуєте лише основні сім. Якщо комісія цікавиться валідацією метрик або “чому саме такі графіки”, відкриваєте резервний восьмий. Точну назву PDF у вашому формулюванні запиту не вказано; нижче я орієнтуюся на нумерацію рисунків усередині основної КРБ. filecite turn0file2

Логіка відбору матеріалу

Для короткого захисту я рекомендую тримати структуру **аналіз → проєктування → реалізація → тестування → практична цінність**. Саме такий порядок добре читається комісією як повний життєвий цикл ПЗ, але без перевантаження другорядними деталями на кшталт CSS або окремих UI-компонентів. У КРБ цей цикл справді простежується: від аналізу предметної області й формування вимог — до UML-моделей, реалізації рушія, SQLite-збереження, API, вебвідтворення та тестування. filecite turn0file2

Сильний акцент раджу зробити на чотирьох речах. По-перше, на **актуальності**: міську поведінку недостатньо показувати як випадковий рух; потрібні агенти зі станом, причинами рішень і просторовими обмеженнями. По-друге, на **математичній моделі**: саме вона робить роботу інженерною, а не “анімаційною”. По-третє, на **пояснюваності**: decision logs, utility, потреби,

альтернативи, маршрут. По-четверте, на **метриках і відтворюваності**: контрольний сценарій, фіксований seed, CSV-експорт, графіки, автоматизовані тести. Це повністю узгоджується і з вашою КРБ, і з лінією подачі в сучасних дослідженнях міських агентних систем. filecite turn0file2 filecite turn0file1 filecite turn0file0

План слайдів

Слайд про актуальність і практичну цінність

Заголовок слайду:
Мультиагентна симуляція міста як інструмент аналізу поведінки

Що розмістити на слайді:
Ліворуч — 3 короткі тези:

- міське середовище — це не випадковий рух, а взаємодія потреб, простору і часу;
- потрібна модель з пояснюваними рішеннями агентів;
- цінність системи — локальний, відтворюваний інструмент аналізу сценаріїв.

Праворуч — **рисунок 3.3** з КРБ: *“Семантична структура тайлової карти міського середовища”*. Це дуже вдалий стартовий візуал: він одразу вводить комісію в тему і показує, що ви моделюєте не абстрактну сітку, а міські функціональні зони. Якщо цей рисунок на слайді виглядатиме дрібно, можна залишити лише центральну частину карти. filecite turn0file2

Що сказати:
«Моя кваліфікаційна робота присвячена розробці програмного забезпечення симуляції міського середовища на основі мультиагентного підходу. Актуальність теми в тому, що для аналізу міських процесів недостатньо показати просто рух об’єктів на карті. Потрібна модель, у якій кожний агент має власний стан, потреби, розклад, просторові обмеження, приймає рішення і залишає пояснюваний журнал своїх дій. Саме таку систему я і реалізував. Її практична цінність полягає в тому, що це локально розгортане MVP, яке можна використовувати як навчальний, демонстраційний і дослідницький інструмент: змінювати кількість агентів, тривалість, seed, сценарій середовища та отримувати відтворюваний запис симуляції, маршрути, decision logs і агреговані метрики». filecite turn0file2

Слайд про постановку задачі і місце роботи серед підходів

Заголовок слайду:
Що саме вирішує система і чому обрано саме такий підхід

Що розмістити на слайді:
У верхній частині — 1 речення з метою роботи:
“Розробити систему, яка генерує населення агентів, моделює їхні дії у місті, маршрутизує рух по пішохідній мережі, зберігає повний timeline та дозволяє аналізувати результати у браузері.”

У нижній частині — компактна таблиця на 3 рядки:

Підхід	Сильна сторона	Обмеження	Позиція вашої системи
Статистичні/ потокові моделі	швидкий макроаналіз	не видно індивідуальних рішень	не обрано як основу
LLM-агенти	природність, пам’ять, соціальність	дорожчі, важчі для локального повторення	використано як референс, не як ядро
Ваша числова модель	прозорість, тестованість, локальний запуск	менше когнітивної глибини, ніж у LLM	оптимально для бакалаврського MVP

Невеликим футером можна додати:
“У роботі не використовується LLM; поведінка формується числовою моделлю.”

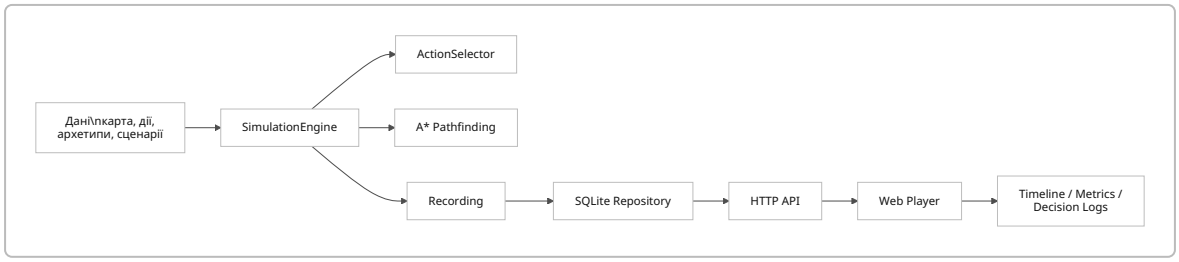
Для візуальної опори можна вставити маленьким блоком **рисунок 2.1** — діаграму варіантів використання, або не вставляти її зовсім, щоб не перевантажувати слайд. Основний акцент тут — на ясному формулюванні задачі й позиціонуванні. filecite turn0file2 filecite turn0file1 filecite turn0file0 filecite turn0file3

Що сказати:
«Постановка задачі в моїй роботі така: для множини агентів у тайлово описаному міському середовищі потрібно в кожний момент модельного часу визначати позицію агента, вектор його потреб, активну дію, маршрут до цілі та журнал прийнятих рішень. При цьому я свідомо не використовував LLM-модель як ядро поведінки. Сучасні роботи, такі як MobileCity, CitySim та AgentSociety, показують сильні сторони великих агентних систем — наприклад, пам’ять, довгострокові цілі або масштабування на тисячі агентів. Але для моєї задачі важливішими були локальне розгортання, відтворюваність, прозорість рішень і можливість автоматизованої перевірки. Тому я обрав пояснювану числову модель, яка займає проміжну позицію між простими rule-based системами і складними LLM-агентами». filecite turn0file2 filecite turn0file1 filecite turn0file0 filecite turn0file3

Слайд про архітектуру і життєвий цикл системи

Заголовок слайду:
Архітектура ПЗ і повний цикл роботи симуляції

Що розмістити на слайді:
Основний візуал тут краще **не брати з PDF**, а згенерувати чисту mermaid-схему. Вона буде значно читабельнішою за секвенс-діаграми 2.4–2.6 у короткій презентації.



Під схемою — 3 короткі пункти:

- двофазна архітектура: **генерація** і **відтворення**;
- backend формує **Recording**, frontend його лише програє;
- запис симуляції стає **відтворюваним артефактом експерименту**.

У правому нижньому куті можна додати невеликий код-блок з GitHub-репозиторієм:

```
github.com/but0wka/urban-agent-sim
```

Це посилання є в додатку Ж КРБ. filecite turn0file2

Що сказати:
«Система побудована за принципом поділу на дані, обчислювальне ядро, шар збереження, HTTP API та клієнтську частину. Ключове архітектурне рішення — це поділ на фазу генерації і фазу відтворення. Під час генерації backend завантажує карту, дії та сценарій, створює агентів, запускає SimulationEngine і зберігає готовий Recording у SQLite. Під час відтворення браузер уже не виконує складну симуляційну логіку, а лише отримує готовий запис через API, програє timeline, інтерполює рух і показує деталі. Це дає три переваги: по-перше, результат можна ставити на паузу і повторно аналізувати; по-друге, поведінка відтворюється при однакових параметрах; по-третє, decision logs і metrics можна досліджувати вже після завершення обчислень. Код системи опублікований у публічному репозиторії GitHub». filecite turn0file2

Слайд про математичну модель агента

Заголовок слайду:
Як агент приймає рішення

Що розмістити на слайді:
Це один із ключових слайдів. Я раджу зробити його як **центральний механізм усієї доповіді**.

Ліворуч — компактна схема: **Needs + Persona + Schedule + Time window + Cost → Utility → Softmax → Action → A* route**

Праворуч — короткий блок з формулами, без перевантаження:

- cosine similarity — формула **3.2**;
- needScore — формула **3.3**;
- utility — формула **3.4**;
- softmax — формула **3.5**;
- A *i Manhattan heuristic — формули 3.6–3.7*.

Як візуальну підтримку вставте **рисунок 3.5**: *“Послідовність вибору дії агентом у модулі ActionSelector”*. Якщо він виходить дрібним, залиште лише власну компактну схему, а рисунок 3.5 перенесіть у резерв. filecite turn0file2

Що сказати:
«Найважливіша частина моєї роботи — це математична модель вибору дії. Тут важливо підкреслити: це не LLM і не текстова генерація рішень. Кожен агент має вектор потреб, persona-риси, розклад і поточний часовий контекст. Спочатку система визначає домінантну поведінкову

категорію: наприклад, критичний голод, запланована дія, робоче вікно або вечірня рекреація. Далі кожна доступна дія отримує числову оцінку. Вона складається з трьох компонентів: відповідності persona до дії через cosine similarity, користі для незадоволених потреб і вартості дії. Після цього обчислюється utility, а остаточний вибір відбувається через softmax по top-k альтернативах. Тобто найкраща дія має найбільшу ймовірність, але поведінка не є повністю детермінованою. Коли ціль визначено, рух до неї будується алгоритмом A* тільки по прохідних тайлах — тротуарах і переходах. Саме так агент може, наприклад, не просто “випадково піти поїсти”, а вибрати їжу як реакцію на низький hunger, з урахуванням часу, характеру, вартості і доступності маршруту». filecite turn0file2

Слайд про діаграму класів і модель даних

Заголовок слайду:
Класова структура системи і база даних

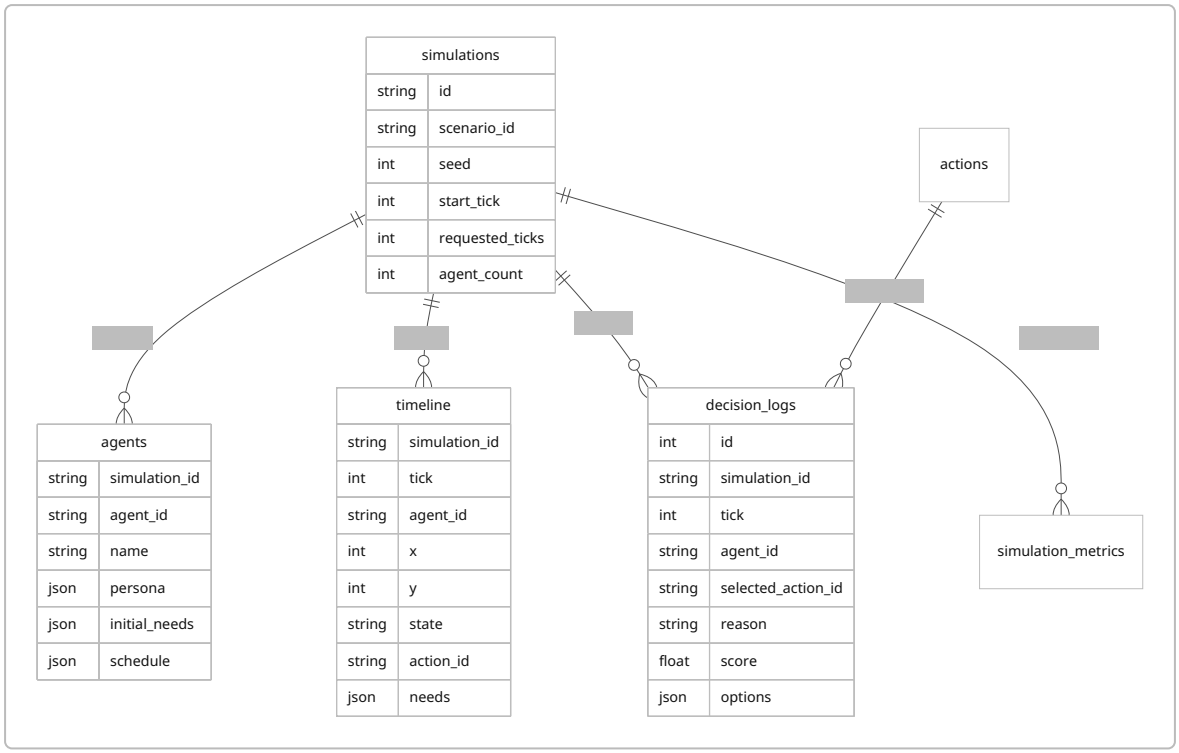
Що розмістити на слайді:
У лівій частині — **рисунок 2.2** з КРБ. Якщо він на екрані буде дрібним, краще взяти **повну UML-діаграму з додатка Б** як крупний crop по центру. Для короткого захисту вам достатньо, аби були добре видимі класи: `Agent`, `Needs`, `Persona`, `ScheduledTask`, `Action`, `AgentSnapshot`, `TimelineFrame`, `DecisionLog`, `Recording`. filecite turn0file2

У правій частині — короткий блок про БД:

- `simulations` — параметри запуску;
- `agents` — профілі агентів;
- `timeline` — снапшоти станів по tick;
- `decision_logs` — вибір дій, причина, score, alternatives;
- `simulation_metrics` — агреговані результати;
- `actions` — action space.

Унизу — маленька примітка: *“Ядро схеми наближене до 3НФ, але для MVP використано контрольовану денормалізацію через JSON-поля: persona, schedule, needs, options, metric_value.”*

Якщо хочете, замість текстового списку БД можна використати цю mermaid ER-схему:



Що сказати:
«На цьому слайді я коротко показую, як проєктна UML-модель імплементована в реальний застосунок. Центральний клас — це Agent. Він містить Needs, Persona, домашню позицію, поточну позицію і розклад у вигляді ScheduledTask. Action описує доступну дію: її категорію, цільову зону, required service, тривалість, ефекти на потреби і часові обмеження. Під час роботи рушія стан агента на кожному tick фіксується через AgentSnapshot, а послідовність snapshots формує TimelineFrame. Recording об’єднує весь результат симуляції: кадри, decision logs, метрики і журнали агентів.
Щодо БД, логічна модель побудована навколо simulations, agents, timeline, decision_logs, simulation_metrics та actions. Така схема добре розділяє “довідкові” сутності, параметри запуску і факти, що накопичуються під час симуляції. За змістом схеми ядро бази є близьким до третьої нормальної форми, але у межах MVP свідомо залишено JSON-поля для складних структур — наприклад persona, schedule, needs і options. Це компроміс між чистою нормалізацією та зручністю збереження аналітичних артефактів». filecite turn0file2

Слайд про тестування і результати

Заголовок слайду:
Що показало тестування і які метрики я отримав

Що розмістити на слайді:
Я рекомендую зробити цей слайд дворядковим.

У верхньому ряду:

- ліворуч — **рисунок 4.1**: автоматизовані тести;
- праворуч — короткий callout:
  61 tests passed / 0.941 s

У нижньому ряду:

- ліворуч — **рисунок 4.2**: теплова карта зайнятості зон;
- праворуч — **рисунок 4.3**: динаміка потреб агентів.

Якщо місця достатньо, у нижній вузькій смузі можна ще дати small inset з **рисунком 4.5** як “матриця переходів”. Але базово для усного захисту двох головних графіків достатньо.
filecite turn0file2

Що сказати:
«Якість такої системи визначається не тільки тим, що код запускається, а й тим, чи правдоподібною є поведінка агентів і чи відтворюються метрики. На рівні автоматизованої перевірки система пройшла 61 тест. Функціонально перевірено, що агенти рухаються лише по прохідних тайлах, входять у функціональні зони через entrance tiles, а API коректно проводить повний ланцюг від POST-запиту до готового Recording у SQLite.
Для аналізу результатів я використовував контрольну симуляцію на 100 агентів, seed 42, старт о 07:00 і тривалість 720 tick. Перша ключова метрика — це теплова карта зайнятості зон, яка показує зміну часток агентів між житловими, робочими, комерційними, освітніми, рекреаційними і медичними зонами впродовж дня. Друга — динаміка потреб. Вона дозволяє перевірити, що hunger змінюється поступово, stress зростає під час інтенсивної активності і спадає під час відпочинку, а продуктивність пов’язана зі станом агента. Тобто система дає не тільки візуалізацію, а й кількісну основу для інтерпретації поведінки». filecite turn0file2

Слайд про практичну цінність, GitHub і розгортання

Заголовок слайду:
Практична цінність проєкту і як його розгорнути

Що розмістити на слайді:
Ліворуч — три короткі блоки практичної цінності:

- навчальний інструмент для курсів із MAS, симуляції та ПІ;
- демонстраційний інструмент для пояснення міської поведінки;
- дослідницький інструмент для сценарних експериментів.

Праворуч — маленька покрокова схема: 1. клонувати репозиторій; 2. запустити через Docker Compose або локально через Python CLI; 3. відкрити вебплеєр, створити симуляцію, переглянути metrics і logs.

У нижній частині — код-блок:

```
git clone https://github.com/but0wka/urban-agent-sim
cd urban-agent-sim
docker compose up --build
# далі відкриваємо http://127.0.0.1:8000/
```

Під блоком — дрібний підпис: *“Посилання на репозиторій наведено в додатку Ж КРБ.”*
filecite turn0file2

Що сказати:
«Практична цінність цієї роботи полягає не лише в тому, що було реалізовано симуляцію, а в тому, що система придатна до повторного використання. Її можна застосовувати як навчальний інструмент для демонстрації мультиагентного підходу, як дослідницький інструмент для аналізу сценаріїв і як демонстраційний MVP для пояснення міської поведінки на рівні агентів і агрегованих метрик. Код опублікований у публічному GitHub-репозиторії. Система підтримує локальний запуск через Python CLI та контейнерний запуск через Docker Compose. Це важливо, тому що дозволяє відтворити експеримент на іншому комп'ютері без ручного налаштування складної серверної інфраструктури. Після генерації користувач отримує browser-based player, де може програвати timeline, переглядати журнали рішень, метрики і аналізувати конкретних агентів або зони. Тобто результат роботи — це не лише текст КРБ, а фактично готовий програмний продукт». filecite turn0file2

Резервний слайд про метрики у стилі сучасних досліджень

Заголовок слайду:
Чому саме такі метрики є переконливими

Що розмістити на слайді:
Цей слайд **не обов'язковий** для основних 5–7 хвилин, але дуже корисний на запитаннях.

Вставити два рисунки **не з вашої КРБ, а з MobileCity**:

- *"The crowd distribution across different urban venues, on weekdays (top) and weekends (bottom)"* — **Figure 1** з MobileCity;
- *"Residents with different employment status have different fluctuations in agent needs during the day"* — **Figure 2** з MobileCity. filecite turn0file1

Під ними — одна фраза: *"Ці типи графіків задають стандарт подання макрорезультатів; у моїй КРБ їм відповідають рисунки 4.2 і 4.3."*

Що сказати:
«Якщо говорити про форму подання результатів, я орієнтувався на сучасні дослідження міських агентних систем. Наприклад, у MobileCity дуже добре показано два класи метрик: по-перше, зміну розподілу натовпу між типами міських просторів упродовж дня; по-друге, різну динаміку потреб для груп населення. Це важливо, бо комісія бачить не просто красиву анімацію, а спосіб перевіряти, чи породжує модель осмислені макропатерни. У моїй роботі аналогічну роль виконують теплова карта зайнятості зон і динаміка потреб за профілями. Тобто я свідомо будував систему так, щоб вона завершувалась не візуалізацією, а аналітичними артефактами, придатними для подальшого дослідження». filecite turn0file1 filecite turn0file2

Розгортання з GitHub

Нижче — інструкція, яку варто тримати окремо як технічний додаток і яку можна частково проговорити на фінальному слайді. Вона спирається на те, що в КРБ явно описані два варіанти запуску: **через Python CLI** і **через Docker Compose**, а також наведено типові команди `init-db`, `generate`, `serve` і контейнерний запуск `docker compose up --build`. Точний файл залежностей у тексті КРБ не вказано, тому цей крок треба звірити безпосередньо з репозиторієм. filecite turn0file2

Варіант через Docker

```
git clone https://github.com/but0wka/urban-agent-sim
cd urban-agent-sim
docker compose up --build
```

Після старту:

```
http://127.0.0.1:8000/
```

Що сказати комісії, якщо спитають:
«Рекомендований варіант розгортання — Docker Compose, тому що він одразу піднімає застосунок в ізольованому середовищі з потрібними залежностями. У контейнерному режимі SQLite-файл зберігається у volume, тому записи симуляцій не зникають після перезапуску контейнера». filecite turn0file2

Варіант локально через Python

```
git clone https://github.com/but0wka/urban-agent-sim
cd urban-agent-sim

python -m venv .venv
source .venv/bin/activate
# Windows PowerShell:
# .venv\Scripts\Activate.ps1

# Далі встановити залежності з файлу проєкту.
# У КРБ точну назву dependency-файла не наведено,
# тому це треба звірити в репозиторії:
pip install -r requirements.txt
# або:
# pip install -e .
```

Потім типова послідовність запуску, прямо описана в КРБ:

```
python -m urban_sim init-db
python -m urban_sim generate
python -m urban_sim serve
```

Якщо ви хочете показати приклад параметризованого запуску:

```
python -m urban_sim generate --ticks 720 --start-tick 420 --seed 42 --agent-
count 100
```

Системні вимоги, які можна безпечно озвучувати:

- Python;
- веббраузер;
- доступ до командного рядка;
- для контейнерного запуску — Docker і Docker Compose. filecite turn0file2

Що показати після презентації у відео-демо

Після доповіді, у відео або live-демо, логічно показати такий маршрут: 1. головну сторінку вебплеєра; 2. створення нової симуляції; 3. запуск playback; 4. вибір одного агента; 5. offcanvas з decision log; 6. метрики і heatmap; 7. коротку згадку, що ті самі записи лежать у SQLite і можуть бути експортовані в CSV. filecite turn0file2

Фраза для озвучення під час відео: «На відео я коротко покажу вже не інтерфейс заради інтерфейсу, а повний цикл роботи системи: генерацію симуляції, відтворення timeline, перегляд конкретного агента, decision logs, теплову карту і метрики. Це дає змогу побачити, що система не завершується на рівні моделі, а має завершений контур від генерації даних до їх аналізу». filecite turn0file2

Опорні таблиці й діаграми

Порівняння математичних моделей і підходів

Підхід	Суть моделі	Переваги	Обмеження	Як позиціонувати на захисті
Статистичні або потокові міські моделі	Агрегують потоки, завантаження, переходи без глибокої індивідуальної логіки	Добрі для швидкого макроаналізу	Погано пояснюють індивідуальні маршрути й причини рішень	Пояснити, що вони корисні, але не дають behavior trace окремого агента. filecite turn0file2
Класичні rule-based ABM	Набір правил або евристик для агентів	Прозорість, швидкість, локальний запуск	Ризик надмірної жорсткості і одноманітності	Ваша система успадковує прозорість, але додає utility та softmax для неоднорідності поведінки. filecite turn0file2

Підхід	Суть моделі	Переваги	Обмеження	Як позиціонувати на захисті
MobileCity	Потреби, звички, обов’язки, контекст середовища, асинхронні LLM-виклики, 4 000 агентів	Реалістичні макрометрики, сильна візуальна аналітика, людяність поведінки	Вищі вимоги до інфраструктури і складніша система	Використати як стильовий і методичний референс подачі метрик. filecite turn0file1
CitySim	Персони, потреби, beliefs, temporal/ reflective/spatial memory, долгострокові цілі, belief-aware place selection	Вища когнітивна глибина, сильна адаптивність	Значно складніше відтворювати локально; залежність від LLM	Пояснити, що це верхня межа складності, але для бакалаврського MVP важливіша пояснюваність. filecite turn0file0
AgentSociety	Реалістичні societal environments і паралелізований рушій, моделювання до 30 000 агентів	Масштаб, середовище, продуктивність, parallel execution	Висока системна складність	Використати як аргумент, що масштабування є окремою науковою задачею, не ціллю вашого MVP. filecite turn0file3
Ваша модель	Needs + Persona + Schedule + Time windows + Utility + Softmax + A* + Decision logs	Локальний запуск, відтворюваність, тестованість, пояснюваність	Менша когнітивна глибина, ніж у LLM-систем	Це інженерно виправдане рішення для бакалаврської роботи і практичного MVP. filecite turn0file2

Перелік інструментів і технологій

Інструмент / технологія	Роль у системі
Python	Основна мова реалізації backend-логіки, рушія симуляції, CLI та API. filecite turn0file2
`urban_sim`	Основний пакет, де реалізовано доменні моделі, ActionSelector, A*, SimulationEngine, repository, API і CLI. filecite turn0file2
SQLite	Локальне реляційне сховище для simulations, timeline, decision logs і metrics. filecite turn0file2
Tiled JSON	Формат опису тайлової карти міського середовища. filecite turn0file2

Інструмент / технологія	Роль у системі
Canvas API	Вебвізуалізація карти, агентів, траєкторій, heatmap та overlay-елементів. filecite turn0file2
Fetch API	Завантаження JSON-даних з API без перезавантаження сторінки. filecite turn0file2
HTML dialog	Модальне вікно для створення симуляції. filecite turn0file2
Docker	Контейнерний запуск системи в ізольованому середовищі. filecite turn0file2
Docker Compose	Оркестрація контейнерного запуску, порту, volume і змінних середовища. filecite turn0file2
CSV-експорт	Підготовка відтворюваних даних для дипломних графіків і подальшого аналізу. filecite turn0file2
GitHub	Публічний репозиторій коду та артефактів проєкту. filecite turn0file2

Рекомендовані типи читабельних діаграм

Для самої презентації я рекомендую не відтворювати всі наявні рисунки один в один, а використати такі типи візуалів:

- **семантична карта міста** — для контексту предметної області;
- **процесна схема рішення агента** — для математичної моделі;
- **велика class diagram з 6–9 підписаними класами** — для проектування;
- **спрощена ER-схема** — для БД і нормалізації;
- **один stacked bar / heatmap** — для макроповедінки;
- **один line chart** — для динаміки потреб;
- **один tiny evidence block** з тестами — для інженерної якості. filecite turn0file2 filecite turn0file1

Відповіді на питання комісії

Актуальність, аудиторія і конкурентоспроможність

Чи є потреба в такій розробці і чому?
Так, тому що міські процеси залежать не лише від геометрії карти, а від поведінки багатьох автономних учасників. Моя система дозволяє бачити не тільки агреговані потоки, а й пояснювані рішення окремих агентів, що важливо для навчання, дослідження і демонстрації сценаріїв. filecite turn0file2

Хто користувач системи?
Користувач — це викладач, студент, дослідник або розробник, який хоче запускати міські симуляції, змінювати параметри й аналізувати поведінку агентів через timeline, decision logs і метрики. У термінах КРБ система задумана як навчальний, демонстраційний і дослідницький інструмент. filecite turn0file2

У чому перевага над існуючими аналогами?
Головна перевага — не “найбільша складність”, а поєднання локального запуску, прозорої моделі, збереження повного запису симуляції, пояснюваності рішень і підтримки аналітики. Порівняно з LLM-підходами система простіше відтворюється, тестується і розгортається локально. filecite turn0file2 filecite turn0file1 filecite turn0file0

Архітектура, проєктування і вибір технологій

Чому обрана саме така архітектура?
Тому що поділ на генерацію і відтворення зменшує навантаження на клієнт, зберігає повний Recording у SQLite і дозволяє повторно аналізувати ту саму симуляцію як артефакт експерименту. Це архітектурно добре відповідає вашій предметній області. filecite turn0file2

Чи не краще було б використати іншу архітектуру?
Для промислового або великомасштабного рішення можлива мікросервісна або розподілена архітектура. Але для бакалаврського MVP правильніше було обрати просту, локально відтворювану архітектуру, у якій чітко видно доменні сутності, алгоритми і потік даних. filecite turn0file2

Як діаграма класів імплементована в реальний застосунок?
Практично один до одного: `Agent`, `Needs`, `Persona`, `ScheduledTask`, `Action`, `AgentSnapshot`, `TimelineFrame`, `DecisionLog`, `Recording` присутні як доменні моделі, проходять через рушій, зберігаються в Recording і віддаються в API. filecite turn0file2

Технічні вимоги, розгортання і інтеграція

Що потрібно, щоб розгорнути систему?
Мінімально — Python, браузер і командний рядок. Альтернативно — Docker і Docker Compose. Базова послідовність: `init-db`, `generate`, `serve`, або `docker compose up --build`. filecite turn0file2

Чи розгорнуто проєкт зараз, і як саме?
У роботі передбачено локальний запуск і контейнерний запуск. Якщо говорити на захисті, найкраща відповідь: “Так, проєкт підготовлено до локального розгортання; рекомендований спосіб — Docker Compose, а для розробки доступний запуск через Python CLI”. filecite turn0file2

Як можна інтегрувати систему в інше ПЗ?
Через два механізми: по-перше, через HTTP API, яке видає карту, дії, сценарії, список симуляцій і конкретний Recording; по-друге, через експорт JSON/CSV для зовнішнього аналізу. filecite turn0file2

ШІ, LLM і математична модель

Це LLM-поведінка чи ні?
Ні. У вашій системі рішення агента формуються числовою моделлю: потреби, persona, розклад, часові вікна, utility-оцінка, softmax-вибір і A*-маршрутизація. Саме це треба чітко проговорити, щоб комісія не сприйняла симуляцію як “чорну скриньку”. filecite turn0file2

Звідки беруться рішення на кшталт “піти поїсти” або “купити їжу”?
Із дефіциту потреб, доступності дій у поточному часовому вікні, узгодженості з persona та вартості дії. Наприклад, при низькому hunger активуються харчові дії, після чого система обирає конкретну альтернативу на основі utility і прокладає маршрут до відповідної зони або сервісу. filecitturn0file2

Чим це краще за повністю rule-based систему?
Тим, що поведінка не є жорстко детермінованою. Softmax додає варіативність, а utility поєднує характер агента, його поточні потреби та вартість дій. Тобто два схожі агенти у схожій ситуації можуть поводитися не абсолютно однаково. filecitturn0file2

Якість коду та інженерія

Як ви боретеся з “макаронним кодом”?
Через розділення на шари: окремо дані, окремо доменні моделі, окремо ActionSelector і pathfinding, окремо SimulationEngine, repository, API, CLI і web player. Це зменшує зв’язність, покращує тестованість і робить код підтримуваним. filecitturn0file2

У чому суть вашої методології розробки?
Якщо відповідати по суті, це ітеративна інженерна розробка MVP: спочатку вимоги і моделі, потім ядро симуляції, далі збереження та API, після цього visualization layer, тести і аналітичні метрики. Календарний план КРБ також відображає поетапну реалізацію цього циклу. filecitturn0file2

Як довести, що система не просто “малює агентів”?
Трьома фактами: є формальна математична модель вибору дій; є decision logs з причинами, оцінками й альтернативами; є аналітичні метрики й автоматизовані тести. Це ознаки інженерної системи, а не просто анімації. filecitturn0file2
