

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)
на тему:

«Симулятор аналізу та прогнозу результатів виборів»

Виконала студентка групи ІПЗ-22-1
КЛОСОВИЧ Ірина Андріївна

Керівник роботи:
САВІЦЬКИЙ Роман Святославович

Рецензент:
ВАКАЛЮК Тетяна Анатоліївна

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна Вакалюк

«28» лютого 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **КЛОСОВИЧ Ірина Андріївна**

Керівник роботи: **САВІЦЬКИЙ Роман Святославович**

Тема роботи: **«Симулятор аналізу та прогнозу результатів виборів»**,
затверджена Наказом закладу вищої освіти від **«27» лютого 2026 р., №92/с**

Вихідні дані для роботи: об'єктом дослідження є процес створення симуляції для політичного аналізу і прогнозу результатів виборів; предметом дослідження є методи і засоби для реалізації мети роботи; розробка алгоритмів та функціоналу системи визначення ймовірних результатів виборів з урахуванням характеристик електорату, зовнішніх факторів, виборчої системи.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Савіцький Р.С.	28.02.2026р.	28.02.2026р
2	Савіцький Р.С.	28.02.2026р	28.02.2026р
3	Савіцький Р.С.	28.02.2026р	28.02.2026р

РЕФЕРАТ

Випускна кваліфікаційна робота освітнього ступеня «бакалавр» складається із симулятора для аналізу та прогнозу результатів виборів і пояснювальної записки. Пояснювальна записка до кваліфікаційної роботи містить 82 сторінки, 79 ілюстрацій та 13 таблиць.

Метою роботи є розробка симулятора аналізу та прогнозування результатів виборів на основі характеристик представників електорату, зовнішніх чинників і виборчої системи. Систему реалізовано на основі гібридної архітектури з використанням Django для серверу, React для клієнтської частини та алгоритмів машинного навчання.

В межах роботи було проведено ретельний аналіз аналогів, обрано архітектуру системи, визначено варіанти використання, спроектовано основні алгоритми системи, обрано алгоритм машинного навчання, протестовано готовий результат.

КЛЮЧОВІ СЛОВА: ВИБОРЧІ СИСТЕМИ, ІДЕАЛЬНИЙ КАНДИДАТ, МАШИННЕ НАВЧАННЯ, ПРОГНОЗ.

					ІПЗ.КР.Б –121 – 26 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Клосович І.А.			Симулятор аналізу та прогнозу результатів виборів Пояснювальна записка	Літ.	Арк.	Аркушів
Керівник.		Савіцький Р.С.					3	82
Рецензент		Вакалюк Т.А.				Житомирська політехніка, група ІПЗ-22-1		
Зав. каф.		Вакалюк Т.А.						

ABSTRACT

The bachelor's qualification thesis consists of an election analysis and forecasting simulator and an explanatory report. The explanatory report contains 82 pages, 79 figures, and 13 tables.

The purpose of the thesis is to develop a simulator for analyzing and forecasting election results based on voter characteristics, external factors, and the electoral system. The system is implemented using a hybrid architecture with Django for the server side, React for the client side, and machine learning algorithms for prediction.

As part of the thesis, a comprehensive analysis of existing solutions was conducted, the system architecture was selected, use cases were defined, the core system algorithms were designed, a machine learning algorithm was chosen, and the final implementation was tested.

KEYWORDS: ELECTORAL SYSTEMS, IDEAL CANDIDATE, MACHINE LEARNING, FORECASTING.

		Клосович І.А.			ІПЗ.КР.Б –121 – 26 – ІПЗ	
		Савіцький Р.С.				
						4

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП	8
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ	10
1.1 Постановка задачі	10
1.2 Аналіз аналогів програмного продукту	13
1.3 Вибір архітектури	31
1.4 Обґрунтування вибору засобів для розробки.....	34
Висновки до першого розділу.....	37
РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ.....	38
2.1 Визначення варіантів використання та структури системи	38
2.2 Проєктування та реалізація алгоритмів	43
2.3 Створення набору даних	48
2.4 Вибір алгоритмів для регресійної задачі	52
Висновки до другого розділу	56
РОЗДІЛ 3. ІНТЕРФЕЙС ТА РОБОТА З СИСТЕМОЮ.....	58
3.1 Налаштування системи.....	58
3.2 Інтерфейс програми	61
3.3 Тестування програми	71
Висновки до третього розділу	75
ВИСНОВКИ.....	76
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	78
ДОДАТКИ.....	82

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API (Application Programming Interface) – прикладний програмний інтерфейс

CI (Continuous Integration) – неперервна інтеграція

CLI (Command-Line Interface) – інтерфейс командного рядка

CSS (Cascading Style Sheets) – каскадні таблиці стилів

E2E (End-to-End) – наскрізне тестування

EVS (Explained Variance Score) – оцінка поясненої дисперсії

FPTP (First-past-the-post) – переможець забирає все

HGBT – HistGradientBoostingRegressor

HTML (HyperText Markup Language) – мова розмітки гіпертексту

JS – JavaScript

JSON (JavaScript Object Notation) – запис об'єктів JavaScript

LLM (Large Language Model) – велика мовна модель

MAE (Mean Absolute Error) – середня абсолютна похибка

MSE (Mean Squared Error) – середньоквадратична похибка

MVP (Minimum Viable Product) – мінімально життєздатний продукт

R^2 (R-squared) – коефіцієнт детермінації

RC – RegressorChain

REST (Representational State Transfer) – передача репрезентативного стану

RF – RandomForest

SDLC (Software Development Life Cycle) – життєвий цикл проєкту

SOLID – аббревіатура з п'яти принципів побудови залежностей в системі

		Клосович І.А.			ІПЗ.КР.Б –121 – 26 – ПЗ	
		Савіцький Р.С.				
						6

TLS (Transport Layer Security) – захист на транспортному рівні

TRS (Two-Round System) – двотурова мажоритарна виборча система

UI (User Interface) – користувацький інтерфейс

VS Code – Visual Studio Code

БД – база даних

ПЗ – програмне забезпечення

		Клосович І.А.			ІПЗ.КР.Б –121 – 26 – ПЗ	
		Савіцький Р.С.				
						7

ВСТУП

Актуальність теми. Політика – це сфера життєдіяльності людей, пов’язана з реалізацією їх суспільних інтересів та ідеалів за допомогою державно-політичної влади [1, с. 8]. Вона є предметом політичної науки, а саме політології, яка була заборонена в Радянському Союзі і тривалий час табуованою у суспільстві, що спричинило недостатнє розуміння всіх процесів взаємодії народу та влади, роботи державних інституцій тощо. Окремим прикладним напрямом є політичний аналіз. Проблематика політичного аналізу і прогнозування обумовлюється потребами розвитку політичної теорії та її впровадження в сучасну політичну практику України [2, с. 11]. Українське суспільство обрало демократичні цінності і тому без аналізу минулого та поточних ситуацій рух може бути повільним, або взагалі неможливим. Відповідно обґрунтований аналіз і прогноз можуть забезпечити реалістичне бачення ситуації і допомогти прийняти оптимальне рішення.

Вибори – це загальноприйнятий демократичний механізм, в процесі якого відбувається формування представницьких органів державної влади та місцевого самоврядування. В багатьох країнах участь у виборах розглядається як право, в деяких вона є обов’язком громадян. Рішення за кого буде голосувати людина залежить від її соціально-культурного розвитку, політичної свідомості, цінностей, популярності політичних течій, впливів медіа тощо. Але крім рішень електорату на результат виборів впливає також і виборча система – сукупність установлених законом правил, принципів і критеріїв, за допомогою яких визначаються результати виборів представницьких органів [1, с. 116]. Розрізняють мажоритарну, пропорційну та змішані системи, і їхній вибір залежить від особливостей країни та стану розвитку політичної системи.

Таким чином, розробка симулятора аналізу дозволить на основі цінностей і переконань людини, її поглядів на життя та суспільство, зовнішніх чинників, які впливають на людину (до прикладу, медіа, кризи тощо) і виборчої системи передбачити ймовірний результат виборів і проводити експерименти. Розробка

таких інструментів актуальна і корисна для українського суспільства і політичної науки як з точки зору навчання та пізнання теми серед населення, так і дослідницької мети.

Мета випускної кваліфікаційної роботи бакалавра – розробити симулятор аналізу та прогнозування результатів виборів на основі характеристик представників електорату, зовнішніх чинників, а також таких елементів політичної системи як виборча система.

Завданнями для реалізації мети є:

1. Розгляд та аналіз аналогів системи, що розробляється, з метою дослідження функціоналу, переваг та недоліків.
2. Визначення архітектури системи.
3. Проектування системи та вибір засобів реалізації.
4. Розробка алгоритмів системи.
5. Реалізація функціоналу.
6. Тестування системи на відповідність поставлених вимог і задач.

Об'єктом дослідження є процес створення симуляції для аналізу і прогнозу результатів виборів.

Предметом дослідження є методи і засоби для реалізації мети роботи; розробка алгоритмів та функціоналу системи визначення ймовірних результатів виборів з урахуванням характеристик електорату, зовнішніх факторів і виборчої системи.

Практичне значення роботи полягає в розробці системи, що зможе бути використана в освітніх цілях під час вивчення політичних і соціальних дисциплін та стане прототипом складнішої системи для наукових досліджень в сфері соціології. В галузі розробки ПЗ результати можуть бути використанні для перевірки доцільності і якості розроблених рішень для прогнозування соціальних процесів з подальшим їх вдосконаленням.

Результати дослідження були апробовані на конференції та опубліковані тези [3, 4].

		Клосович І.А.			ІПЗ.КР.Б –121 – 26 – ПЗ	
		Савіцький Р.С.				
						9

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

1.1 Постановка задачі

Вибори є одними з найважливіших демократичних процедур, що забезпечують демократичні принципи. Вони розрізняються: за органом, що вибирається (президентські, парламентські, органів місцевого самоврядування тощо); за рівнем (загальнонаціональні, федеральні, місцеві); за видами (чергові або позачергові); за виборчими системами (мажоритарні, пропорційні, змішані) [1, с. 116]. Відповідно рано чи пізно дієздатні особи зіштовхуються з ними: або як виборці, або як кандидати, або як політичні аналітики. Для виборців корисно було би дізнатись про різні види виборчих систем і як вони працюють за допомогою інструментів, що можуть симулювати процес обрання, а для кандидатів і політичних аналітиків в швидшому режимі проаналізувати настрої та можливі варіанти подій за різних обставин.

Соціологічні опитування та дослідження важливі для визначення громадської думки на момент проведення цих опитувань, але вони потребують довгого часу опрацювання та визначення тенденцій у суспільстві. Також зазвичай питання сформульовані про конкретних кандидатів і партії в момент проведення опитування без врахування, що кандидата може замінити однопартієць, який матиме інакші рейтинги.

Метою роботи є розробка симулятора аналізу та прогнозу результатів виборів. Система для кінцевого користувача повинна забезпечити зрозуміле виведення полів для заповнення інформації про стан держави, інформації про виборців та кандидатів; відображення карти; можливість редагування інформації; копіювання країни; обробку окремо характеристик виборця, кандидатів та зовнішніх чинників; обрахунок результатів відповідно до виборчих систем; виведення результатів для аналізу та порівняння для кількох країн. Слова «держава» і «країна» в цьому випадку відносно розроблюваної системи – модель з поділом на частини подібні до «областей» чи «штатів», які містять різну кількість виборців, яких користувач може розподіляти на свій

		Клосович І.А.			ІПЗ.КР.Б –121 – 26 – ПЗ	
		Савіцький Р.С.				
						10

розсуд, для кращого сприйняття інформації. Інші визначення наведено у глосарії в додатку А. Копіювання країни з усіма даними допоможе менш затратно змінити лише деякі характеристики для зручнішого більш наочного подальшого порівняння. Відповідно на основі задач, які повинен виконувати застосунок, сформулюємо розподіл функціоналу по модулях за тим як передається інформація.

Реалізація модуля «Введення інформації, збору даних та виведення результатів» спрямована на зручне виведення полів для заповнення, налаштування роботи з картою та відокремлення інформації за типом виборів (президентські, парламентські), даних про виборців (вік, стать, рівень впливу релігії, розуміння та зацікавленість у різних процесах), даних про кандидатів і зовнішніх факторів (війна, кризи в економіці, медіа). Після опрацювання даних іншими модулями результати голосування кожного виборця та загальні підсумки виводяться у зручному форматі, а за потреби формується кілька турів голосування з окремим відображенням проміжних та фінальних результатів кожного етапу. Інтерфейс модуля забезпечує валідацію введених даних на рівні форм з підказками користувачу та можливістю редагування інформації перед запуском обчислень.

Модуль «Дослідження» отримує дані кандидатів і стану країни для подальшої передачі до великої мовної моделі з формуванням характеристик у числовому вигляді. Дані виборців аналізуються окремо за іншими алгоритмами без використання великої мовної моделі, що зменшує обчислювальне навантаження та зберігає прозорість обробки персональних характеристик. Для кожного виборця обраховується схожість з кожним кандидатом за обраним набором критеріїв з нормалізацією значень у єдину шкалу для подальшого зіставлення. Результати дослідження зберігаються у проміжній структурі даних та передаються до наступного модуля без втрати інформації про окремі компоненти схожості.

Модуль «Обрахування результатів» приймає вихідні дані модуля «Дослідження» та формує кінцеві результати відповідно до обраних виборчих

		Клосович І.А.			ІПЗ.КР.Б –121 – 26 – ПЗ	
		Савіцький Р.С.				
						11

систем. Логіка обчислень враховує особливості кожної системи, серед яких визначення переможця за відносною та абсолютною більшістю, попарне порівняння голосів за кожного кандидата, визначення переможців в регіонах. Остаточні результати передаються назад до модуля «Введення інформації, збору даних та виведення результатів» з відображенням розподілу голосів між кандидатами та статистичних показників явки.

Отже, застосунок, що розробляється, допоможе узагальнити теорію та наочно демонструє процес голосування з опорою не на простий наслідок «йду голосувати за кандидата А», а на причини вибору певного кандидата. Серед таких причин «це відповідає моїм цінностям і поглядам», «мені подобається або не подобається стан країни», «у мене активна або неактивна політична позиція», «мені байдуже до результатів виборів», «особисті цілі ставлю вище за державні». Кожен з мотивів отримує власну вагу в моделі прийняття рішення виборцем та впливає на підсумкову оцінку схожості з кожним кандидатом. Така структуризація дозволяє відійти від спрощеного бінарного підходу до моделювання вибору та врахувати багатовимірний характер реальної електоральної поведінки.

Подібний підхід відкриває нові можливості для аналізу електоральних процесів у навчальному та дослідницькому контекстах. Студенти політологічних та соціологічних спеціальностей отримують інструмент для перевірки теоретичних моделей на змодельованих сценаріях з можливістю змінювати параметри виборців, кандидатів та зовнішніх факторів. Дослідники можуть використовувати застосунок для побудови гіпотез про вплив окремих факторів на результат голосування з подальшою перевіркою на реальних статистичних даних. Гнучкість моделі також дозволяє адаптувати її до різних національних контекстів з урахуванням специфіки виборчих систем, культурних особливостей та поточної політичної ситуації в обраній країні.

1.2 Аналіз аналогів програмного продукту

Необхідно розглянути аналоги застосунку, що розробляється, з метою вивчення переваг, які можна додати у список вимог, і недоліків, яких можна уникнути.

Гра GerryMander [5] починається з інструкції про правила, у якій описується процес джеррімендерингу як змінювання меж території для збільшення кількості людей на цій території, які голосуватимуть за певну партію чи особу [6]. На рис. 1.1 зображено приклади картинок об'єднання округів та смугову діаграму з кількістю голосувальників за конкретну партію. Протягом гри необхідно об'єднувати голосувальників в округи, щоб перемогла партія за умовою рівня. Складність зростає поступово з додаванням нових обмежень на форму округів та розподіл виборців різних політичних поглядів. Ігровий процес побудований на просторовому мисленні та аналізі демографічних даних, що змушує гравця приймати рішення з урахуванням не лише поточної конфігурації округів, а й передбачуваної поведінки виборців.

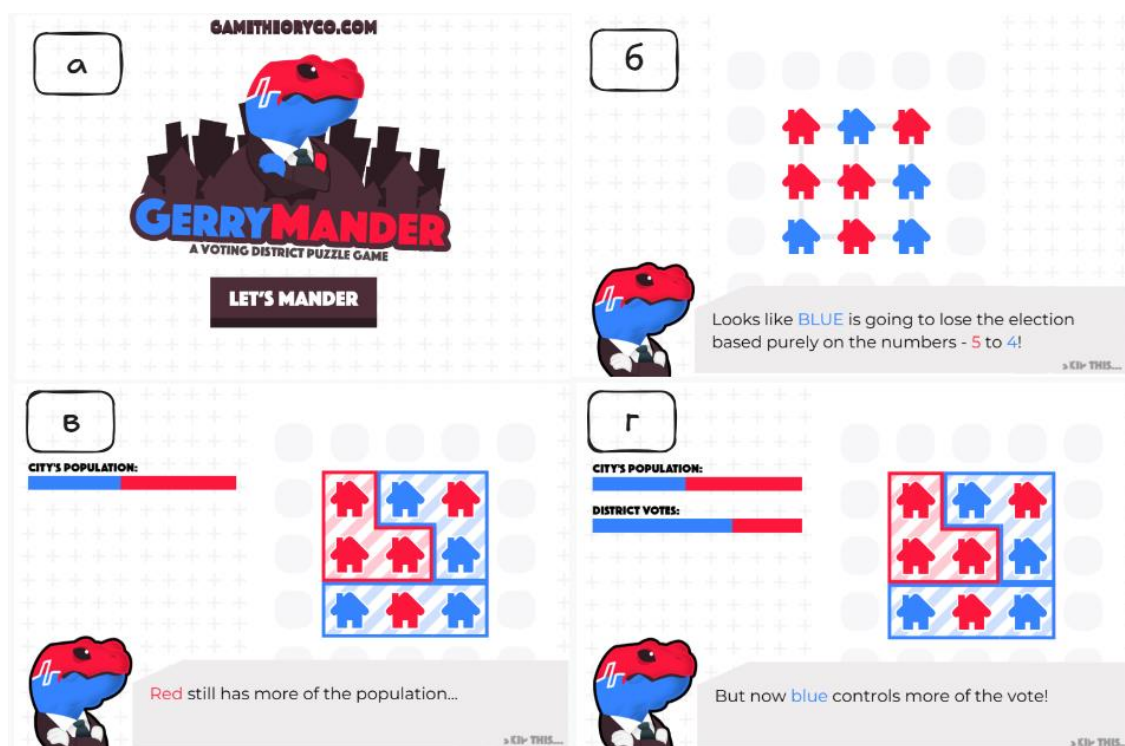


Рис. 1.1. Заставка до гри (а) та інструкція з поясненнями процесу джеррімендерингу: «синіх» менше, але вони перемогли «червоних» завдяки поділу на такі округи (б, в, г)

Наприкінці кожного рівня з'являється історична нотатка про цей процес (рис. 1.2). Нотатки містять реальні приклади джерррімендерингу з історії США та інших демократичних держав з посиланнями на конкретні політичні події. Такий освітній компонент перетворює гру з простого розважального продукту на інструмент підвищення політичної грамотності та критичного мислення щодо виборчих процесів.



Рис. 1.2. Історична нотатка про процес

Другий аналог – Election Lab [7]. За основу взято США та їхню виборчу систему. Першочергово користувача зустрічає форма з вибором режиму гри та складністю рівня, що зображено на рис. 1.3 та рис. 1.4 відповідно.

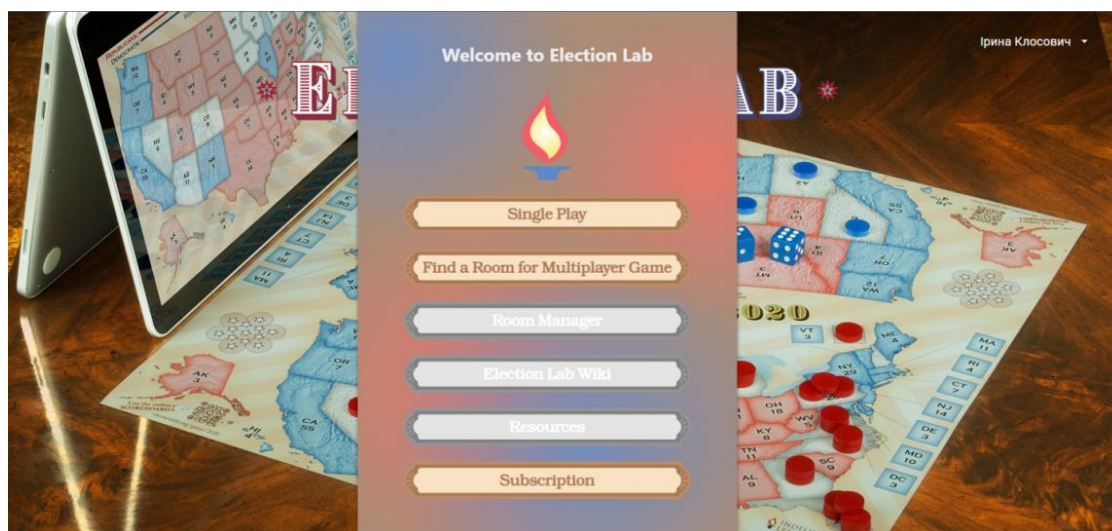


Рис. 1.3. Форма для вибору режиму гри

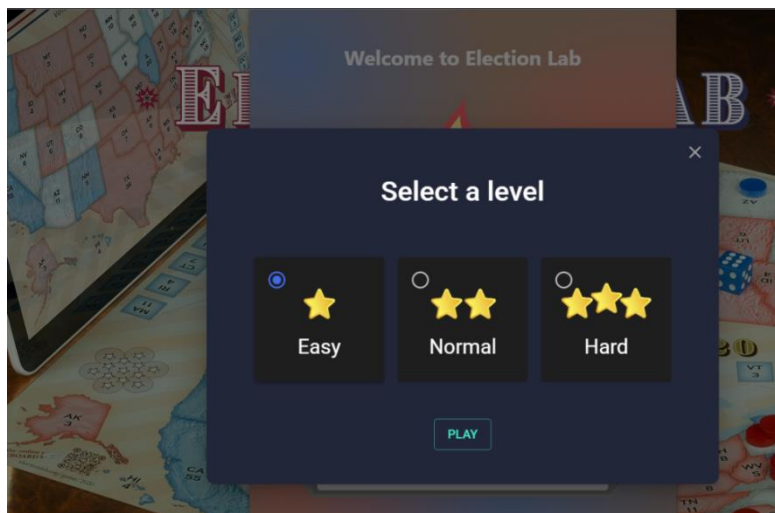


Рис. 1.4. Вибір рівня гри

Далі користувачу кажуть за яку партію він грає (рис. 1.5). Перед ним є карта США, де штати зафарбовані відповідно до партій-лідерів там, а також виокремлено штати, що не визначились (рис. 1.5 та рис. 1.6). Над картою розміщено смугові діаграми та кількість голосів за кожну партію з оновленням значень у реальному часі під час прийняття рішень гравцем. У правому верхньому куті розташована інструкція з правилами гри, що залишається доступною протягом усієї сесії. Колірне кодування штатів дозволяє швидко оцінити поточний баланс сил між партіями та зосередитися на штатах, де результат ще не визначено. Інтерактивність карти дає змогу натискати на окремі штати для перегляду детальної демографічної та електоральної інформації, що допомагає приймати обґрунтовані стратегічні рішення.



Рис. 1.5. Початок гри



Рис. 1.6. Інструкція, пояснення карти

Відповідно до інструкції пояснюється карта, як відбувається розподіл ресурсів передвиборчої кампанії та як іде боротьба за штати. Розподіл ресурсів на кампанію (рис. 1.7) відбувається за допомогою повзунків для кожного штату від 0 до 5, де 0 – найменше ресурсів, 5 – найбільше. Ресурси необхідно розподіляти відповідно до обмежень, не можна використати більше, ніж дано.

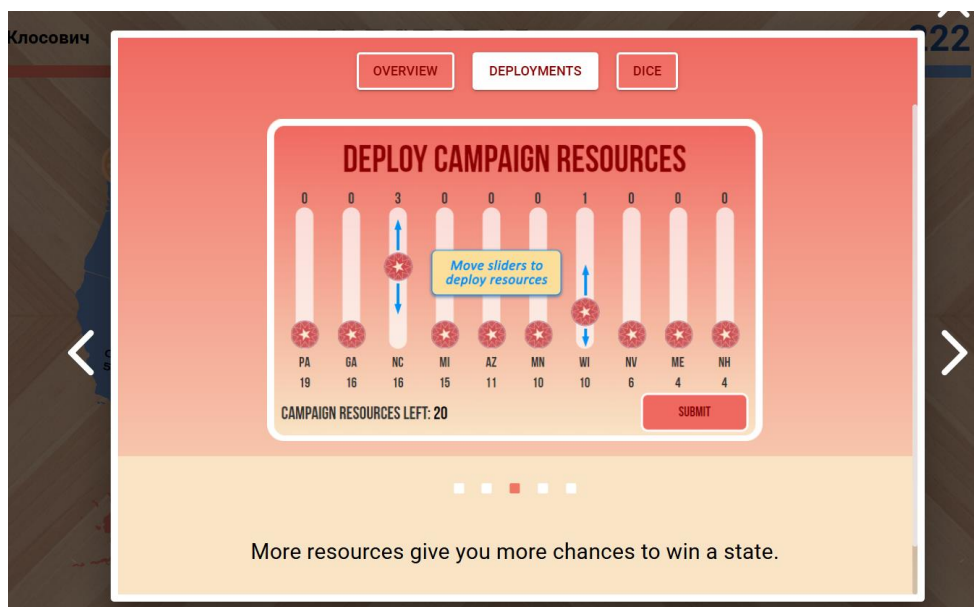


Рис. 1.7. Інструкція, повзунки для ресурсів кампанії

Після того як ресурси розподілені починається кампанія. Спочатку хтось один обирає штат для боротьби; відповідно до того скільки виділялось ресурсів на штат дається кількість гральних кубиків (0 ресурсів – нуль кубиків,

1 ресурс – один кубик і т. ін.). Далі вони «підкидаються» і таким чином визначається переможець в штаті як на рис. 1.8. На наступний раунд штат вибирає опонент.

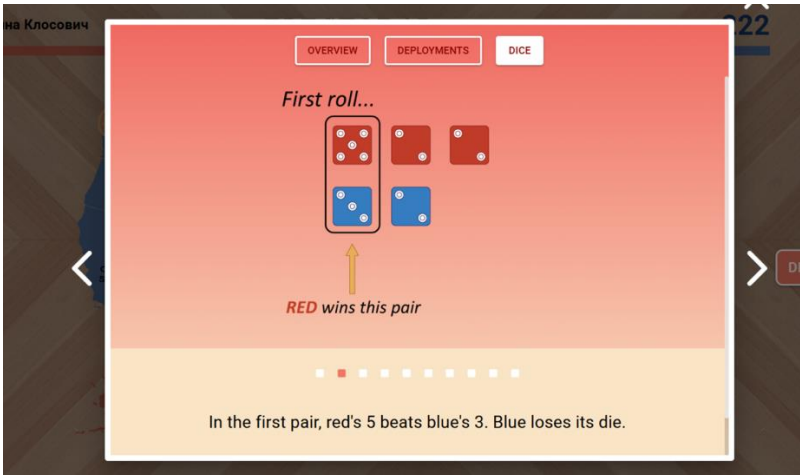


Рис. 1.8. Інструкція, як іде боротьба за штат

Після визначення переможця в штаті цей штат зафарбовується відповідним кольором, а також змінюється колір фону на вкладці з назвами штатів та ресурсами кампанії, що розташована праворуч від карти (рис. 1.9). Така візуальна синхронізація між картою та боковою панеллю дозволяє гравцеві миттєво відстежувати прогрес кампанії в кожному регіоні без переключення уваги між різними елементами інтерфейсу.

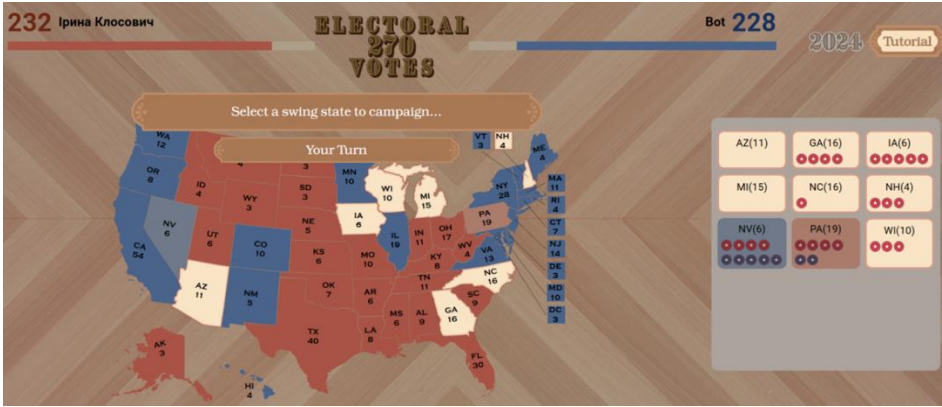


Рис. 1.9. Приклад як відображено забрані штати

Процес виборів відбувається поки всі штати не визначаться і один з кандидатів не набере необхідну кількість голосів. В кінці відображається вікно з результатом (рис. 1.10), а після цього можна переглянути стратегії і визначити яка з них була реалізована під час гри, коли розподілялись ресурси (рис. 1.11). Аналіз стратегій дозволяє гравцеві порівняти власні рішення з типовими

підходами до ведення виборчої кампанії, серед яких концентрація ресурсів у вирішальних штатах, рівномірний розподіл за всіма регіонами або фокус на штатах з невизначеною позицією виборців. Така рефлексія над прийнятими рішеннями перетворює гру з простого змагання у навчальний інструмент для розуміння реальних механізмів виборчих процесів. Повторне проходження кампанії з іншою стратегією дозволяє експериментально перевірити, як змінюється результат залежно від розподілу обмежених ресурсів між штатами.



Рис. 1.10. Результат гри

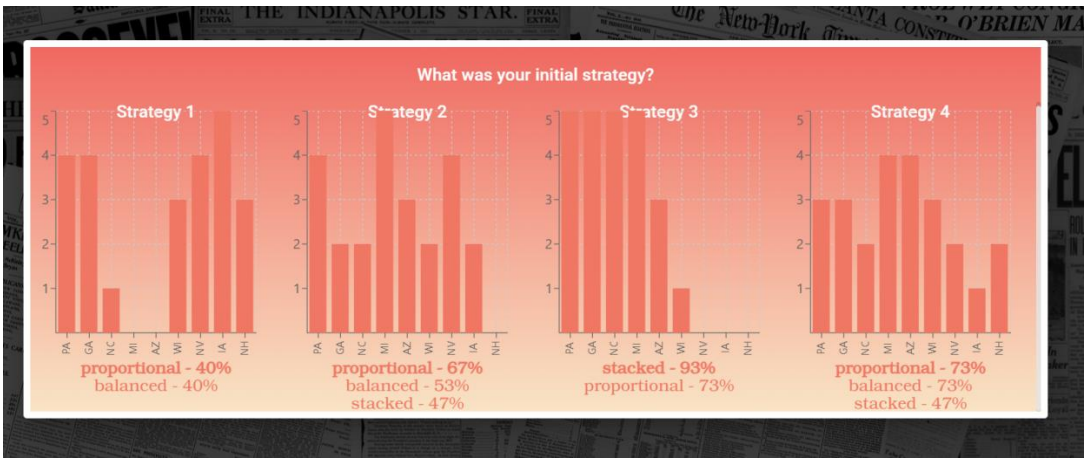


Рис. 1.11. Підбиття підсумків та аналіз помилок

Третій аналог було взято зі статті How ranked-choice voting could change the way democracy works новинного сайту The Washington Post [6]. У статті пояснюється що таке рейтингове (ранжоване) голосування (ranked-choice voting), його переваги та приклад на невеликій симуляції. Традиційно в бюлетені більшості країн позначають одного найбільш бажаного кандидата. В рейтинговому голосуванні кандидатів розподіляють за вподобаннями (рис. 1.12) [8].

In a traditional voting system, voters select just one candidate. With ranked-choice voting, they rank candidates in order of preference.

Traditional ballot		Ranked-choice ballot			
Voters select their favorite candidate.		Voters rank candidates from most to least favorite.			
Candidates	Choose one	Candidates	Choices		
			1st	2nd	3rd
● Alice Green	☒	● Alice Green	☒	☐	☐
● Bob Orange	☐	● Bob Orange	☐	☐	☒
● Carol Lavender	☐	● Carol Lavender	☐	☒	☐

Рис. 1.12. Різниця між традиційним і рейтинговим голосуванням

Відповідно до статті, в традиційному голосуванні в США перемагає той, хто отримав найбільше голосів (навіть, якщо це не більшість); в рейтинговому – спершу кандидати розглядаються за першим рівнем (пріоритетом), якщо жоден кандидат не набрав більшості, той, що набрав найменше голосів за першим пріоритетом вибуває, а голоси розподіляються за іншими кандидатами і цикл повторюється поки один з кандидатів не набере половину голосів.

Для пояснення процесу на симуляції взяли за основу систему координат, де на горизонтальній вісі позначили лібералів та консерваторів, а на вертикальній – рівень поміркованості (рис. 1.13). На цій системі хаотично розташовано 200 кружечків, що демонструють виборців. Також є великі круги, які можна рухати і визначати таким чином кому надають перевагу виборці (зону зафарбовує у відповідний колір кандидата).

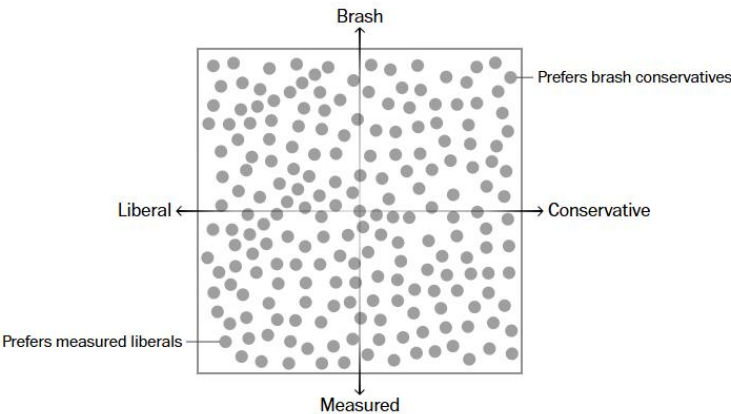


Рис. 1.13. Система поглядів для гри

Над системою знаходиться шкала, яка у відсотковому співвідношенні зафарбована до відповідних кандидатів (рис. 1.14) та кнопки з можливістю додати кандидатів та очистити, щоб повернутись до початкових даних. Динамічне оновлення шкали відбувається після кожної зміни ваг або параметрів виборців, що дозволяє користувачеві одразу бачити вплив змін на загальний результат. Поряд зі шкалою відображаються конкретні числові значення відсотків для кожного кандидата з підвищенням точності сприйняття результатів. Кнопка очищення повертає систему до стандартного набору параметрів без потреби перезапуску застосунку або повторного введення базових даних.



Рис. 1.14. Приклад розподілу голосів за кандидатами
у два тури (а) та один (б)

Під системою координат знаходиться ряд, в якому відображено всі раунди та як змінюється поділ після вибування одного з кандидатів. Кандидат, що вибув, зафарбовується в сірий колір, а його виборці розподіляються між іншими і виділяються яскравістю кольорів. Зміни в раундах відбуваються динамічно, одразу коли буде додано кандидата чи змінено пропорцію підтримки. На рис. 1.14 показано за замовчуванням два тури та один (висвічується повідомлення, що перший тур був вирішальний), на рис. 1.15 було додано кандидата і з'явився додатковий тур.

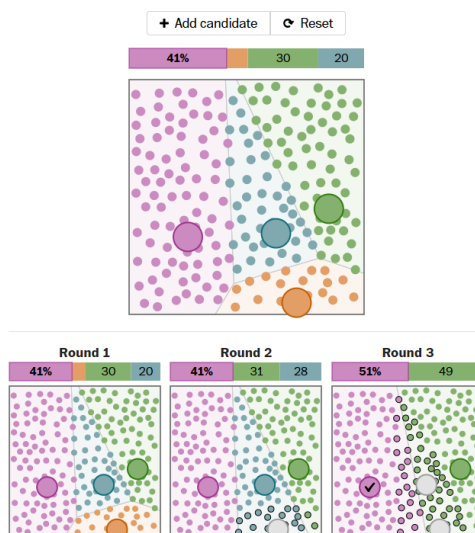
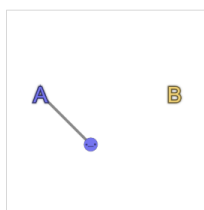


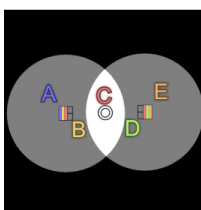
Рис. 1.15. Можливість додати кандидата і більше двох турів

Наступним аналогом виступає сайт із симуляціями про виборчі системи smartvotesim.com [9]. На головній сторінці розміщені посилання окремо для кожної теми (рис. 1.16) з описом і поясненням самих виборчих систем та симуляціями з відеоінструкціями щодо роботи з ними. Структура головної сторінки забезпечує швидку навігацію між різними виборчими системами без потреби попереднього ознайомлення з повним обсягом матеріалів сайту. Кожне посилання веде на окрему сторінку з інтерактивними елементами для безпосереднього експериментування з параметрами виборчої системи у браузері.

Intro

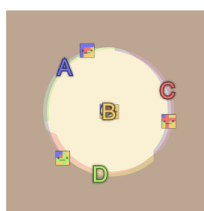


The Basics - #1 - Start here. See the 2D model we use for this site.

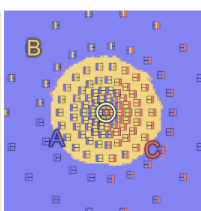


Finding Common Ground - #2 - Why do we vote? To make smarter group decisions.

In Depth



Condorcet Methods - the best way to find common ground



Approval Voting - a practical improvement - And it works with strategy.



STAR Voting - and a deeper dive into using strategy and the game of chicken.

Рис. 1.16. Приклади симуляцій на сайті

		Клосович І.А.			ІПЗ.КР.Б –121 – 26 – ПЗ	
		Савіцький Р.С.				
						21

Загалом усі наявні симуляції мають умовний поділ на дві колонки: в першій є місце з можливістю перетягування елементів, додавання тощо, а в другій даються короткі пояснення та діаграми. До прикладу, на рис. 1.17 в лівій колонці відбуваються перестановки кандидатів та змінюється колір фону від їхнього впливу; також внизу самого квадрата є інші інструменти, але відсутні підказки як ними користуватись і для чого вони; нижче є кнопки для очищення та збереження.

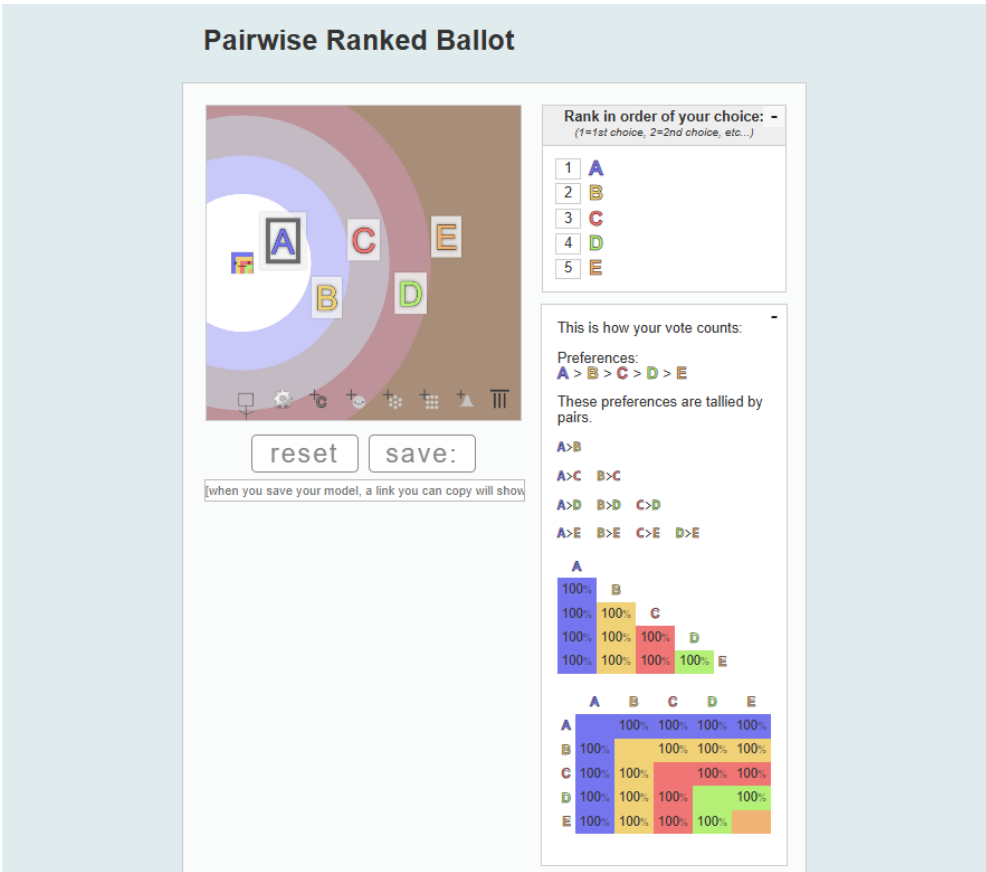


Рис. 1.17. Попарно ранжоване голосування

У правій колонці розміщені секції з виділенням кольорами та розмірами, які можна згортати для оптимізації простору, та які відображають положення кандидатів у списку вподобань виборця і у поточному випадку (рис. 1.17 – 1.18) попарне порівняння всіх кандидатів через знаки, колонки, матрицю, лінійні діаграми та пріоритети. Різноманітність форматів візуалізації дозволяє користувачеві обрати найзручніший спосіб сприйняття результатів залежно від конкретного аналітичного завдання.

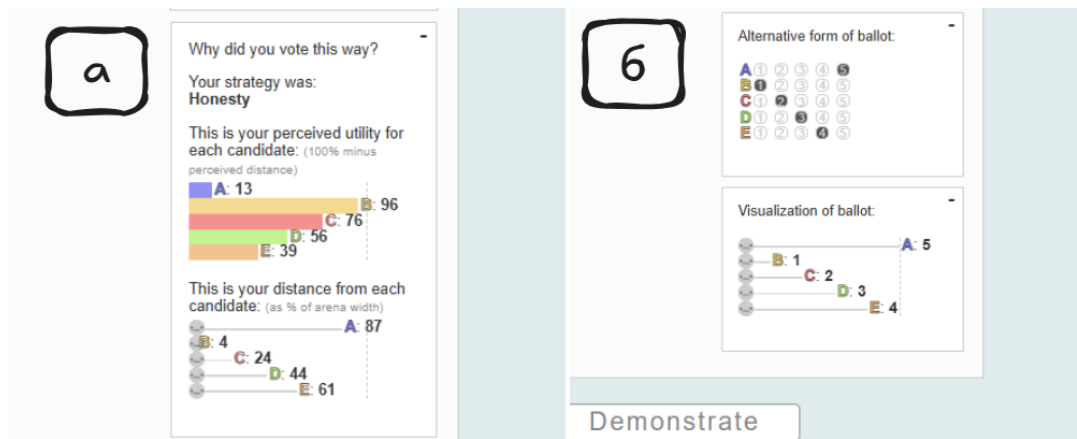


Рис. 1.18. Діаграми до попарного ранжованого голосування

В іншій симуляції, де є приклад FPTP системи та методу Кондорсе, є можливість за тих самих умов змінити ці системи одна одною. В лівій колонці для системи FPTP (рис. 1.19) на колі зображуються точками виборці та кандидати зафарбовують своїх прихильників у відповідний колір. Справа зображується діаграма з результатом хто переміг.

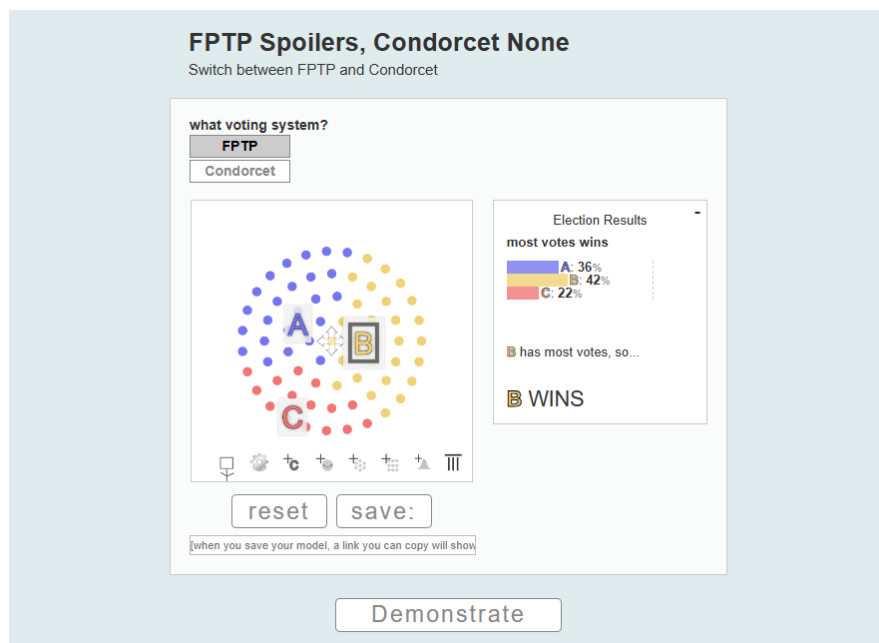


Рис. 1.19. FPTP

В методі Кондорсе (рис. 1.20) виборців зображують не суцільним кольором одного кандидата, а всіх з різним наповненням відповідно до симпатій. Справа є колонки та текстовий опис попарного порівняння кожного кандидата у відсотках та написано результат хто переміг.

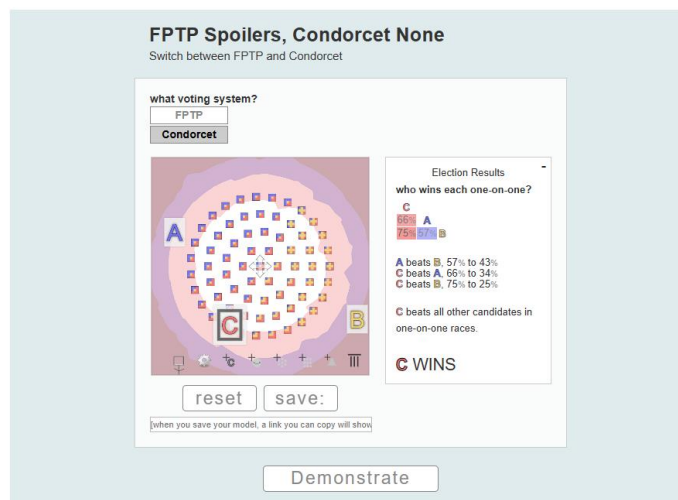


Рис. 1.20. Метод Кондорсе

Останній аналог взято з сайту perthirtysix.com [10]. В симуляції три кроки: налаштування кандидатів, запуск та дослідження результатів. В налаштуваннях (рис. 1.21) обирається кількість кандидатів та даються характеристики повзунком для рівня популярності та поляризації. Є можливість відредагувати ім'я. Наступний крок це запуск симуляції (рис. 1.22) натиском на кнопку.

1) Setup

You can set up the details about the candidates we're simulating here.
If you don't care about the details, you can skip to the Simulation section below!

How many candidates?
- 5 +

Randomize Their Values

Candidate 1

Popularity

Polarization

Candidate 2

Popularity

Polarization

Candidate 3

Popularity

Polarization

Candidate 4

Popularity

Polarization

Candidate 5

Popularity

Polarization

Рис. 1.21. Налаштування кандидатів

2) Simulate

Once you're done setting up candidates, you can run the simulation here.

Run It

Рис. 1.22. Запуск симуляції

Після того як симуляцію було запущено можна переглядати результати і досліджувати в третій секції. В перших двох рядках є вибір, щоб переглянути інформацію про голосувальників та самі типи виборчих систем (рис. 1.23). В комірках зазначається який кандидат здобув перемогу.



Рис. 1.23. Меню для дослідження

В інформації про голосувальників показано таблицю на 1000 голосувальників з їхніми пріоритетами щодо кожного кандидата в лівій частині екрану. В правій частині показано інформацію про кандидатів (популярність та поляризацію) та наскільки часто кандидати фігурували в тій чи іншій позиції серед виборців (рис. 1.24).

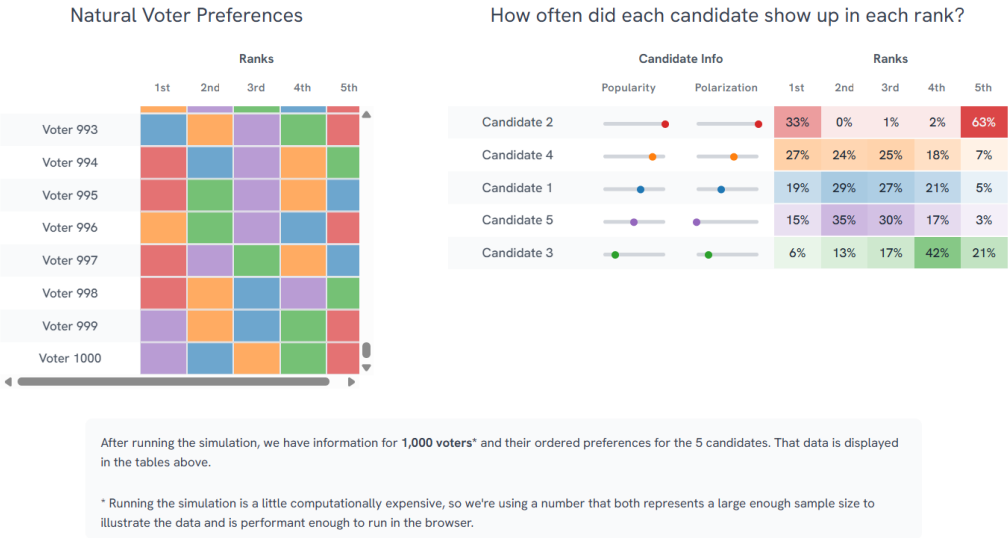


Рис. 1.24. Демонстрація виборців та їхніх вподобань

Відповідно до кожної системи демонстрація результатів різна. Щодо спільного, то в лівій частині буде таблиця виборців хто як голосував з акцентами відповідно до виборчої системи. В правій частині можуть бути таблиці або графіки. Наприклад, для системи відносної більшості береться до

уваги стовпчик (виділяється яскравістю) з першим пріоритетом серед кандидатів (рис. 1.25).

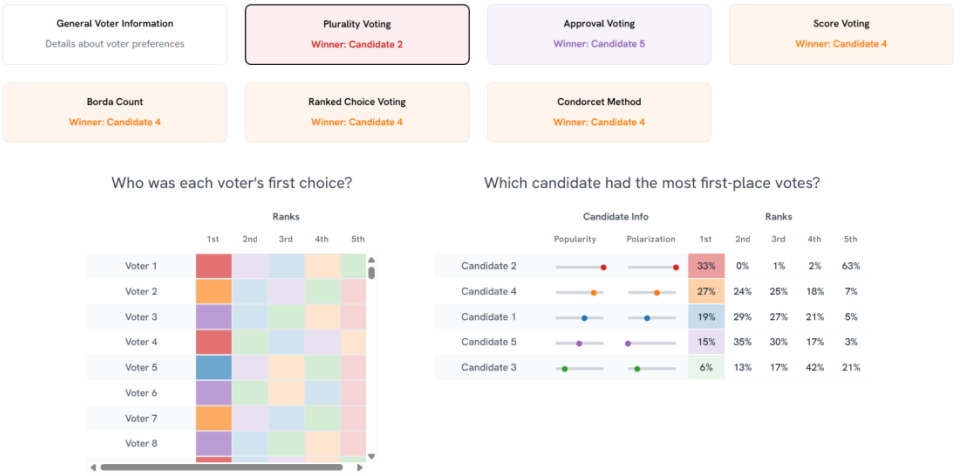


Рис. 1.25. Система відносної більшості

У системі голосування схваленням виборець обирає декількох кандидатів, тому на результатах у цій вкладці відображаються позначки конкретних схвалених виборцем кандидатів та сумарний підрахунок голосів за кожного з них (рис. 1.26). Сумарний підрахунок дозволяє швидко визначити кандидата з найбільшою кількістю схвалень, який і стає переможцем у такій виборчій системі. Візуалізація схвалень для кожного виборця окремо допомагає простежити структуру електоральних вподобань та виявити кандидатів з широкою базою підтримки.

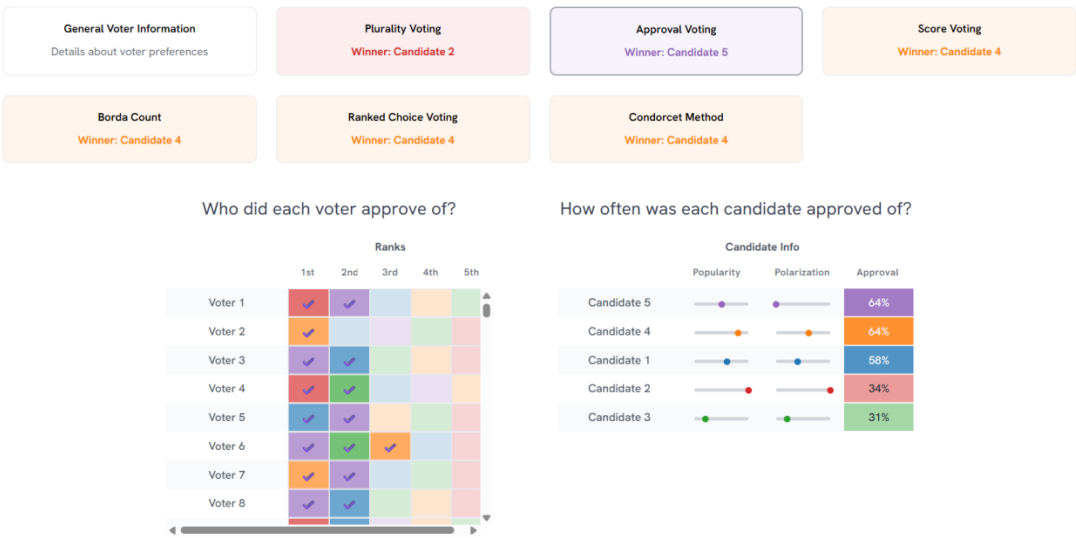


Рис. 1.26. Голосування схваленням

Для всіх систем є опис хто переміг, що це за система, переваги та недоліки (рис. 1.27).

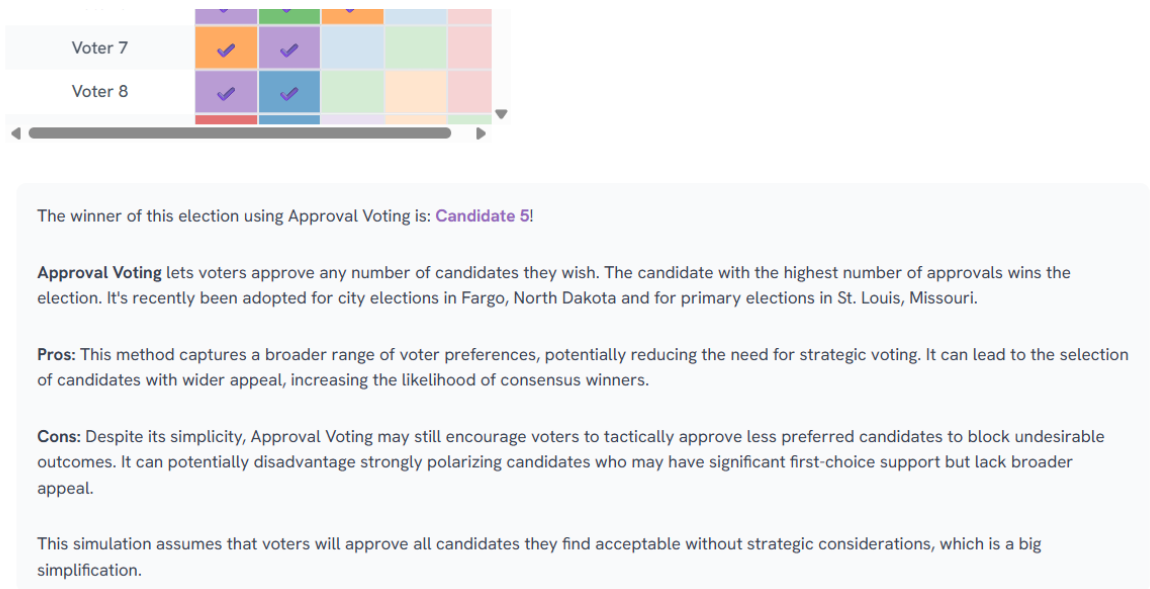


Рис. 1.27. Нотатка про виборчу систему

Рейтингове голосування було проаналізовано в третьому аналізі. Якщо там була система координат поглядів та покроково саме на ній позначали всі зміни, то в цьому аналізі кроки показані на діаграмах, зверху є можливість переключатись між ними вперед або назад (рис. 1.28-1.30). На діаграмі є позначка до якої потрібно, щоб дійшли; якщо кандидат вибуває, про це зазначається в наступній діаграмі, а його голоси розподіляються між іншими зберігаючи колір першочергового кандидата.

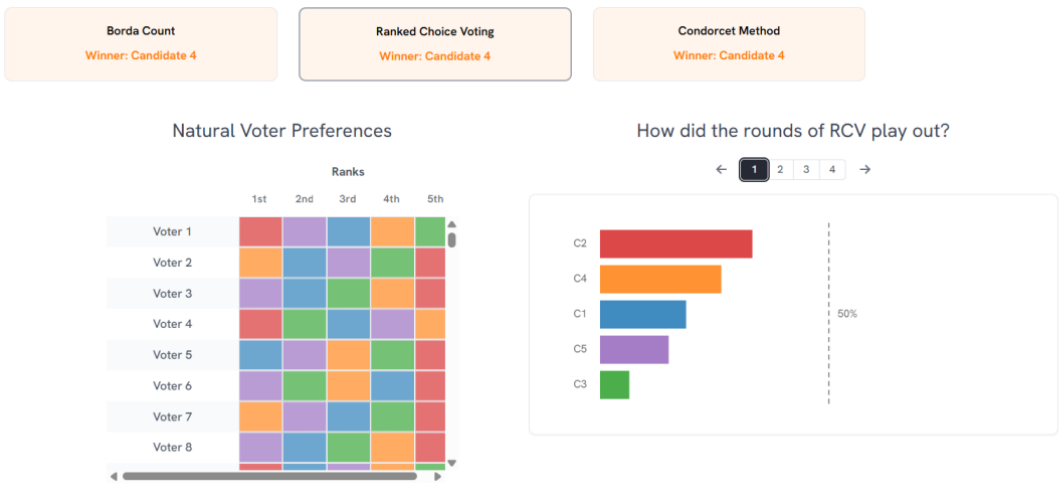


Рис. 1.28. Рейтингове голосування крок 1

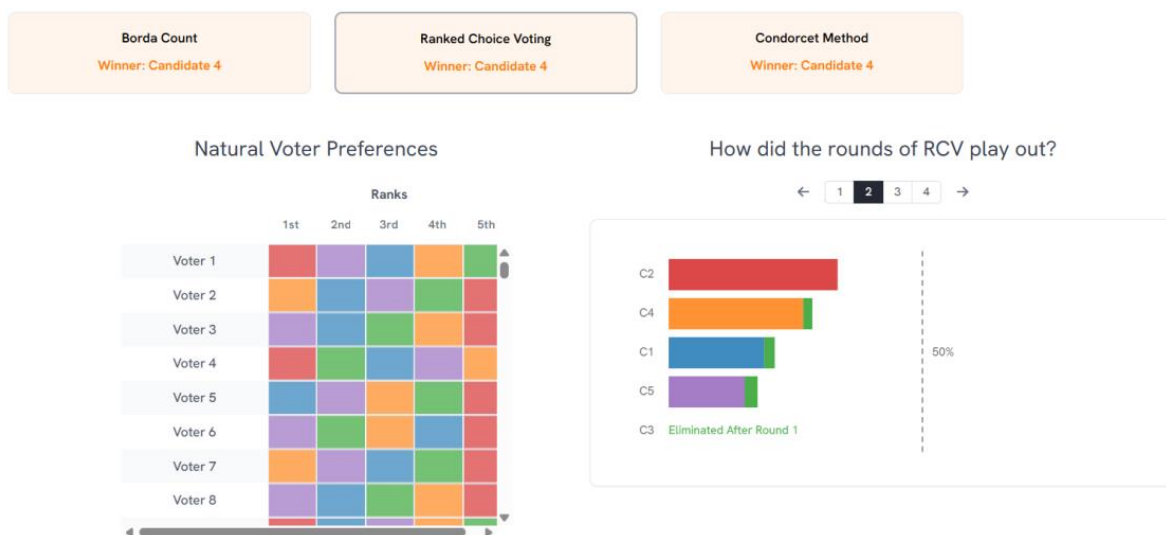


Рис. 1.29. Рейтингове голосування крок 2

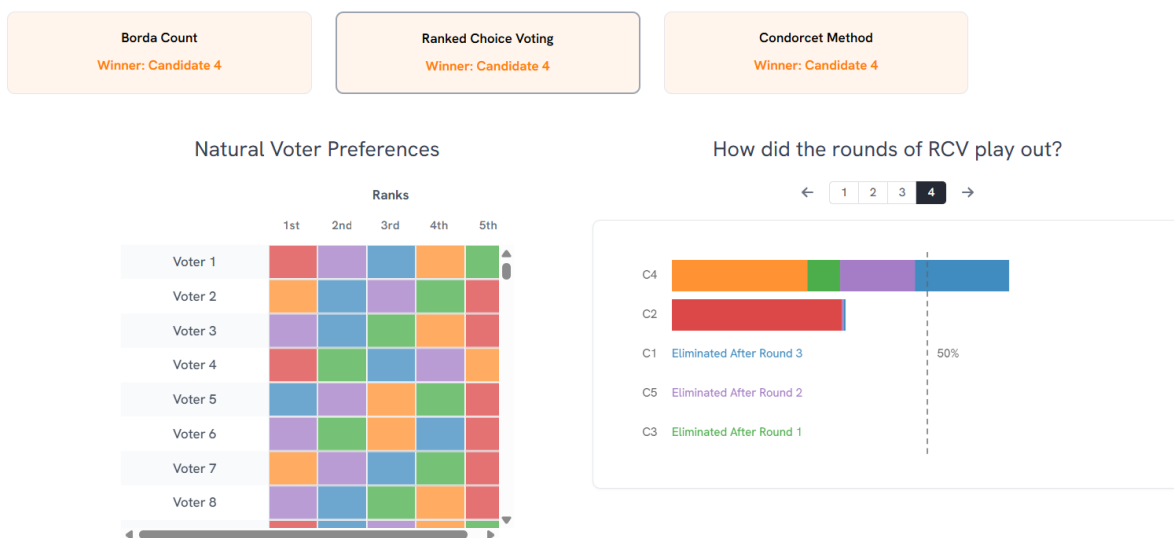


Рис. 1.30. Рейтингове голосування останній крок

Система Кондорсе розглядалась в четвертому аналізі: пари, які порівнювались, було описано текстом із зазначенням відсотків. В цьому аналізі для демонстрації використано матрицю, в якій брали рядок, що відображає кандидата і по стовпчиках порівнювали з іншими. Наприклад, з рис. 1.31 перший рядок це перший кандидат, з першим кандидатом не порівнюється, другого переміг, третього переміг, четвертому програв, п'ятому програв. Такий пробіг робиться по всіх кандидатах. Перемагає той, у чиєму рядку всі перемоги.

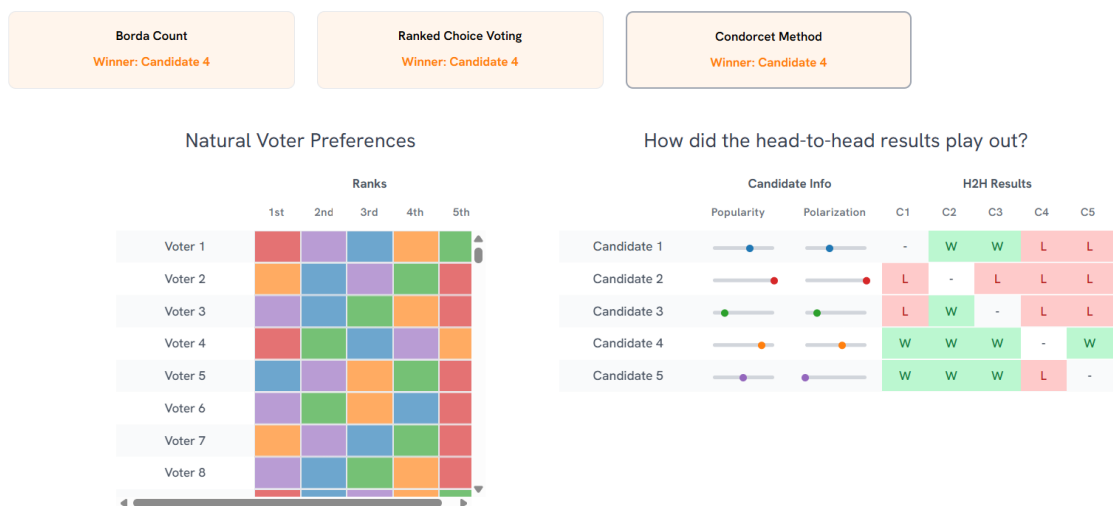


Рис. 1.31. Метод Кондорсе

Щоби змінити дані необхідно натиснути кнопку, що починає симуляцію спочатку, всі попередні дані втрачаються (рис. 1.32).

1) Setup

If you want to change the details of the election, you can reset the simulation and start over.

How many candidates?

Candidate 1

Popularity

Polarization

Candidate 2

Popularity

Polarization

Candidate 3

Popularity

Polarization

Candidate 4

Popularity

Polarization

Candidate 5

Popularity

Polarization

Рис. 1.32. Початкові налаштування

Характеристику аналогів та розроблюваної системи можна подати у вигляді таблиці 1.1. Номери аналогів дано відповідно до послідовності, в якій вони розглядались (GerryMander це I, Election Lab це II, стаття The Washington Post це III і т.д).

Порівняння характеристик

Характеристика	Непрямі аналоги		Прямі аналоги			Розроблювана система
	I	II	III	IV	V	
Допомога в загальному розумінні процесу	+, частина з територіальним поділом	+, процес передвиборчої кампанії	+, система поглядів виборців і виборча система	+, погляди виборців і виборчі системи	+, опис кандидатів та виборчі системи	звертається увага на територіальний поділ, цінності окремих виборців, виборчу систему, опис кандидатів та медіа
Візуалізація результатів	робоча область, діаграми	карта, графіки	система координат	робоча область, графіки	графіки, таблиці	карта, графіки, таблиці
Можливість редагування після спрацювання симуляції	-	-	+	+	-	+
Можливість порівняння результатів	-	+/-, стратегії передвиборчих кампаній	-	+/-	+/-, в цілому добре	+
Відсутність бажання голосування певного виборця	-	-	-	-	-	+
Демонстрація покроковості (за необхідності)	-	-	+	-	+	+

Після аналізу аналогів було вирішено, що найкраще орієнтуватись на останній з чіткістю демонстрації процесу голосування. З першого та другого аналогів взято ідею роботи з картою. Третій і четвертий підштовхнули до формування системи цінностей кожного з виборців, але не програмно як в них,

а власноруч. Також важливо було врахувати аспект відсутності бажання голосувати.

В результаті розроблюваний симулятор аналізу та прогнозу результатів виборів має низку переваг в порівнянні з вже наявними аналогами, поєднує найкращі функції та рішення для кращого розуміння і пізнання процесу виборів.

1.3 Вибір архітектури

У книзі Роберта Мартіна Clean Architecture можна прочитати: «Архітектура програмної системи – це форма, яку надають цій системі ті, хто її створює. Ця форма визначається поділом системи на компоненти, їхнім розташуванням та способами, якими ці компоненти взаємодіють між собою. Метою цієї форми є полегшення розробки, розгортання, експлуатації та обслуговування програмної системи, що міститься в ній». Далі автор продовжує, що архітектура не має великого впливу на працездатність системи, але стверджує, що її роль критична, хоч і є пасивною та косметичною. Основною її метою називає підтримку життєвого циклу системи. Гарна архітектура робить систему легкою для розуміння, легкою для розробки, легкою в обслуговуванні та легкою в розгортанні. Основна ціль – мінімізувати вартість життєвого циклу системи та максимізувати продуктивність розробника [11, частина 5, розділ 15].

Відповідно можна зробити висновок, що архітектура системи базується на конкретних задачах, що поставлені перед розробниками, і тому незважаючи на різноманітну кількість архітектурних підходів, патернів та стилів, рідко використовують суто один з них. Натомість в проєктах поєднують елементи з різних патернів для задоволення конкретних вимог, створюючи гібридні підходи, що адаптовані до унікального контексту завдань [12, 13].

В цій роботі ми маємо одну задачу, яка поділена на кілька підзадач. Далі ми розглянемо варіанти архітектури для підзадач спираючись на життєвий цикл проєкту (SDLC), а саме етапи розробки, тестування та розгортання [14].

Для розробки найзручніше і найпростіше використовувати монолітну архітектуру, оскільки вона легка в реалізації та економічно ефективна [15]. Але

при її масштабуванні виникають труднощі, тому доцільно використати багат шарову архітектуру для розподілу залежностей. У шарі для ядра системи можна розмістити сутності та реалізацію функцій для прогнозу і обрахунку результатів, а в шарі з бізнес-логіки викликати їх у потрібній послідовності та з перевірками. Такий поділ зумовлений принципом єдиної відповідальності, що кожен модуль виконує свою задачу. Це полегшує внесення змін та розширення функціоналу, а також допомагає уникнути сильної зв'язаності.

Важливо згадати, що усі залежності повинні спрямовуватись лише в середину: шар API залежить від бізнес-логіки, бізнес-логіка у свою чергу – від ядра, яке незалежне від зовнішніх компонентів.

Для етапу тестування важливо, щоб усе було логічно розподілено для простоти ізольованої перевірки окремих частин системи, тому доречно виділити у шарі ядра модуль для дослідження, де будуть передбачатись уподобання виборців, та модуль для обрахування, де буде здійснено обчислення за відповідними виборчими системами. Такий поділ дає змогу замінити алгоритми машинного навчання без внесення змін в алгоритми обрахування результатів виборчих систем. Окремий модуль для виборчих систем надає можливість додавати нові алгоритми підрахунку голосів.

Якщо думати про розгортання, то на даний момент не передбачено використання однієї чи кількох БД і оскільки достатньо виділення модулів, а не сервісів, то можна задовільнитися монолітною, а не мікросервісною архітектурою. До того ж, розглядаючи фізичний рівень, завданням не обмежено, що це буде локальна розробка. З написаного можна припустити, що необхідна клієнт-серверна архітектура, де клієнтська частина відповідає за презентаційний рівень, а сервер є монолітним. Для зв'язку клієнта з сервером в цьому випадку не завадить використовувати REST API [16].

Перевагою REST API є його простота використання, оскільки базується на знайомих HTTP протоколах; гнучкість, бо працює з будь-кими даними та форматами; масштабованість, бо легко розширюється під потреби; незалежність клієнта та сервера, що розвиваються окремо.

З вище написаного можна зробити висновок, що найкращим варіантом буде використати клієнт-серверну архітектуру, у якій клієнтська частина буде поділена на компоненти, відповідатиме презентаційному шару і зв'язуватиметься з сервером використовуючи REST API. У свою чергу сервер буде монолітний, поділений на три шари, де шар API приймає запити від клієнта та надсилає відповіді, шар бізнес-логіки використовуючи реалізовані функції і сутності в шарі ядра обраховує результати, а шар ядро поділений на модулі, де перший відповідає за вподобання виборців, а другий за обрахунок відповідно до виборчих систем. Саме такий поділ є оптимальним з точки зору розробки, тестування та розгортання такого рівня проєкту. Якщо правильно все реалізувати з дотриманням усіх принципів залежностей та SOLID, то ми отримаємо чисту архітектуру, в основі якої лежить незалежність від обраного фреймворку, UI, БД, інших зовнішніх сервісів і яку можна легко протестувати [11, частина 5, розділ 22]. Описану архітектуру зображено на рис. 1.33.

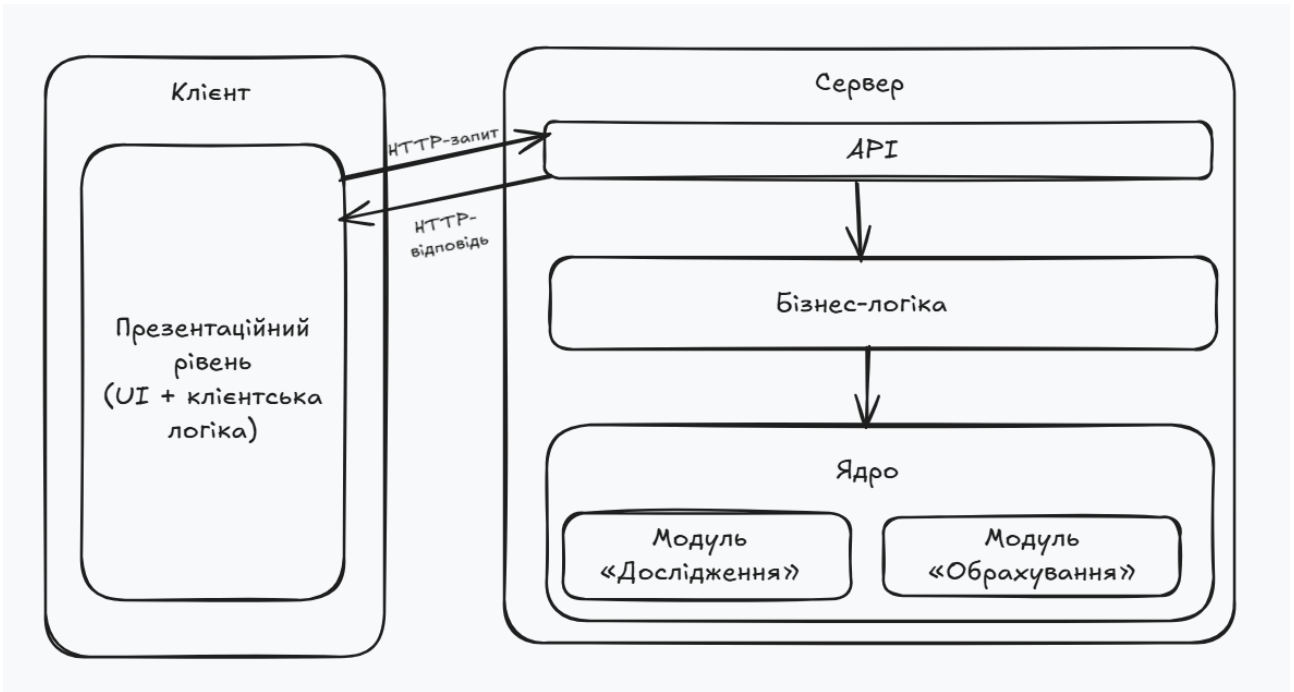


Рис. 1.33. Обрана архітектура проєкту

Отже, розглядаючи спроектовану архітектуру системи запит від користувача проходить наступним чином. Клієнтська частина формує HTTP-запит і надсилає його до REST API сервера. Шар API виконує первинну

валідацію вхідних даних та передає керування шару бізнес-логіки. Бізнес-логіка визначає необхідну послідовність виконання операцій, звертаючись до модуля прогнозування електоральних уподобань і модуля обрахунку результатів виборів. Після завершення обчислень сформований результат повертається до шару API, який серіалізує відповідь у формат JSON та надсилає її клієнтській частині для відображення користувачу.

1.4 Обґрунтування вибору засобів для розробки

Етап планування, прототипування та проєктування – це важлива стадія, на якій можна передбачити частину помилок, щоби уникнути під час розробки та зекономити час і ресурси. Під час цієї фази було використано векторний онлайн-сервіс розробки інтерфейсів Figma [17]. Також було здійснено підготовку до наступного етапу і налаштовано Visual Studio Code – безкоштовний відкритий редактор коду, у якому можна використовувати різні мови програмування та розширення для зручного користування [18]. Для проєктування системи у вигляді діаграм було використано розширення у VS Code PlantUML і Mermaid, що мають простий синтаксис для написання та детальні інструкції [19, 20].

Етап розробки. В першу чергу під час створення проєкту було налаштовано Git для контролю та управління версіями вихідного коду [21]. Репозиторій розміщено на університетській платформі git.ztu.edu.ua (GitLab Community Edition) відповідно до вимог [22]. Додатково було створено дзеркало на GitHub [23].

Для клієнтської частини було використано локальний сервер для розробки Vite [24]. Він швидко запускається та миттєво оновлюється після внесення змін. Також з його переваг є підтримка мови TypeScript та різних фреймворків.

Клієнтська частина не може обійтись без мови розмітки HTML, стилів CSS та динаміки. Для динамічних змін станів елементів та передачі даних з клієнта на сервер було обрано TypeScript, оскільки він підтримує типізацію, що спрощує розробку та пошук помилок і є більш надійним варіантом у середніх

та великих проєктах, на відміну від JavaScript, де динамічна типізація несе ризики.

В наш час існує безліч фреймворків та бібліотек, що збільшують ефективність розробки вебзастосунків. Бібліотека React має широку спільноту підтримки; невисокий поріг для вивчення, якщо є гарне розуміння HTML та JS; гарну документацію. Також вона використовує функціональний підхід до написання компонентів і взагалі тут усе є компонентами, які можна перевикористовувати. У наявності є широкий вибір бібліотек під різні задачі [25]. Angular потребує більше знань аби почати писати, має велику складність, процес розробки триваліший, але проєкти є більш гнучкими і легше масштабуються [26]. Vue є поєднанням двох попередніх технологій. Має близький до React поріг входу, гарну документацію, але спільнота є невеликою, що є недоліком для підтримки самого фреймворку.

Для проєкту важлива стабільність технологій та яка буде подальша вартість його підтримки, тому всі характеристики було узагальнено в таблиці 1.2 і на основі цього для розробки обрано бібліотеку React.

Таблиця 1.2

Порівняння характеристик фреймворків та бібліотек для клієнту

	React	Angular	Vue
Поріг входу для розробки	Невисокий	Високий	Невисокий
Стабільність підтримки	+	+	-
Спільнота	Широка	Широка	Невелика
Швидкість розробки	Висока	Тривала	Висока
Ринок розробників (вплив на вартість)	Найбільший	Менший	Менший
Рівень володіння командою (1 ос.)	+	-	-

Проект передбачає залежності між елементами та їх станами, що змінюються, а також часту передачу даних з одного компонента в інший. Щоб уникнути в React таких частих передач від батьківського елемента до дочірнього і навпаки (англ. Prop Drilling), необхідно використовувати менеджер станів. Найкращим рішенням є Zustand, бо він простий, швидкий і дозволяє створювати кілька сховищ (англ. Store) [27].

Оскільки задачі проекту містять вимогу обрахунків та прогнозу, найкращим варіантом для написання серверної частини є мова Python. На цьому етапі створення MVP не передбачається збереження даних в БД для користувачів і гарним варіантом був би фреймворк FastAPI, оскільки має високу продуктивність, є інтуїтивним та легким [28]. Але цей засіб не надає вбудовані механізми для роботи з користувачами чи БД, що могло би ускладнити подальший розвиток продукту не як однієї симуляції, а набору симуляцій з можливістю збереження даних. Тому було обрано фреймворк Django, який є стабільним, має всі ці вбудовані механізми та широку підтримку спільноти [29].

NumPy є основою для роботи з математичними функціями [30]. Scikit-learn є бібліотекою для машинного навчання, яка надає прості та ефективні засоби для прогнозного аналізу даних [31].

Імпорти та залежності є вагомою частиною розробки, тому для того, щоб їх було легко читати, а код виглядав більш зрозумілим використовувались такі лінтери та форматори коду як Ruff для Python і ESLint для TypeScript [32, 33]. Зокрема, другий лінтер було налаштовано на дотримання правил виклику хуків та автоматичне впорядкування імпортів у визначеному порядку: зовнішні бібліотеки, внутрішні модулі системи, побічні імпорти, відносні імпорти, файли стилів та статичні ресурси. Такий підхід підвищує читабельність коду та його підтримку.

Безперервна інтеграція (CI) – практика розробки ПЗ, коли кожен учасник інтегрує роботу до спільного репозиторію, після чого автоматично запускаються збірка і тести. Це є досить зручним, оскільки завжди відомий стан

коду і вартість виправлення помилок є мінімальною. Контейнери Docker забезпечують ізольоване середовище (для роботи було встановлено Docker Desktop, його складність мінімальна), що в подальшому полегшує розгортання [34]. Додатково було використано Google Colab для машинного навчання.

Висновки до першого розділу

У першому розділі було сформовано задачі необхідні для реалізація застосунку. До них належать зручний інтерфейс для роботи з необхідним функціоналом, що допоможе розгорнути тему; перетворення характеристик країни та кандидатів у числовий вигляд; аналіз виборців; обрахування схожостей кандидатів і виборців; обрахування результатів за виборчими системами.

Також у розділі проаналізовано п'ять аналогів до розроблюваної системи. З них було виокремлено шість характеристик пов'язаних з різними аспектами виборчого процесу та запозичено найкращі рішення, які доповнено власними. Серед таких запозичених рішень є робота з картою, але з модернізованою структурою.

Далі було описано вибір гібридної архітектури спираючись на цикл розробки. Як результат найкращим рішенням є клієнт-серверна архітектура, у якій сервер монолітний зв'язується з клієнтом, використовуючи REST API. Сам же сервер поділений на шари.

В останньому підпункті описано вибір засобів для розробки симулятора. Оптимальними для розробки стали такі засоби як React і Zustand для клієнту, Django для серверу, Scikit-learn для машинного навчання.

РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Визначення варіантів використання та структури системи

Після ретельного дослідження області можна сформувати різні способи зобразити виборчий процес та зміну характеристик виборця. Перший варіант системи є науково-дослідницьким: зібрати демографічні дані реальних виборців з внутрішніми характеристиками; створити алгоритм, що зможе аналізувати тенденції та передбачати результат; показувати на реальній карті зміни по роках. Але фізично реалізувати це важко через відсутність таких даних.

Другий спосіб є більш навчальним та орієнтованим на одного користувача, який задає характеристики виборцям, кандидатам та країні. Після цього формуються пріоритети виборців, які використовуватимуться для обрахунків за виборчими системами (саме підрахунок голосів без врахування, які є списики як на рис. 2.1-а).

Третій спосіб орієнтований на отримання результатів за виборчими системами, спираючись на різні види бюлетенів для голосування як на рис. 2.1-б. Таку систему доцільно робити багатокористувацькою і з можливістю організатору формувати типи бюлетенів, але вона не дає змогу прогнозувати тенденції суспільства.

Спираючись на вище наведені роздуми, було вирішено реалізовувати другий спосіб; відповідно було виявлено одного актора – користувача.

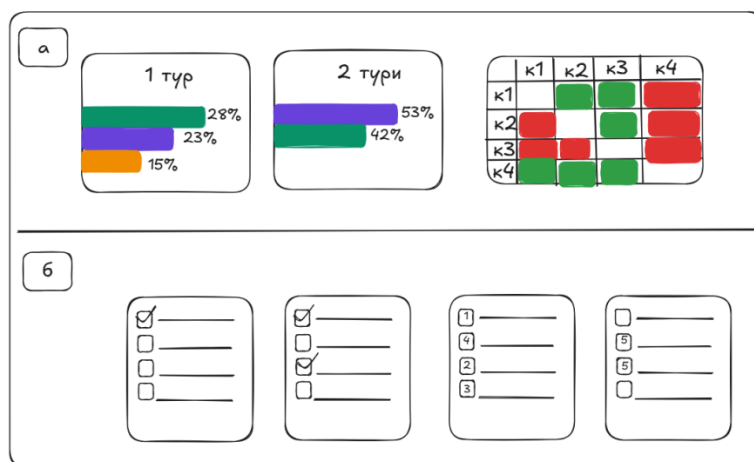


Рис. 2.1. Варіанти реалізації симулятора виборчих систем: а – загальний підрахунок, б – орієнтація на формування бюлетеня

У таблиці 2.1 описано виявленого актора.

Таблиця 2.1

Виявлення акторів

Актор	Короткий опис
Користувач	Людина, що використовує симулятор для кращого пізнання і розуміння виборчого процесу при різних характеристиках. Користувач здатний створювати країну, редагувати її характеристику та виборців, додавати та видаляти різних кандидатів тощо, тобто повністю керує нюансами.

У таблиці 2.2 наведено варіанти використання системи.

Таблиця 2.2

Виявлені варіанти використання

Актор	Найменування	Формулювання
1	2	3
Користувач	Створення країни	Створення порожнього шаблону країни, який необхідно редагувати
	Редагування даних країни	Можливість змінити назву країни та її опис, попрацювати з картою і виборцями на ній, задати кандидатів
	Робота з картою	Включає переміщення виборців та зміну рівня небезпеки в регіонах
	Додавання виборця	Додавання виборця до карти
	Редагування виборця	Оновлення даних виборця
	Видалення виборця	Видалення виборця загалом
	Переміщення виборців	За необхідності зміна положення виборців на карті
	Додати кандидата	Створити нового кандидата з можливістю подальшого редагування. Залежно від типу виборів або одна людина, або партія, або кандидат від партії
	Редагувати кандидата	Задавати характеристики кандидату
	Видалити кандидата	Вилучити кандидата з усіма характеристиками

1	2	3
	Зміна рівня небезпеки регіону	Обирається рівень безпечності регіону
	Надсилення даних на обробку	Можна виконати після заповнення даних, коли створено або відредаговано країну
	Перегляд результатів	Переглянути у графічній формі результати (можливо після опрацювання даних)
	Перегляд графіків населення	Зображення статево-вікової структури країни загалом та по регіонах
	Копіювання країни	Створення копії країни з тими характеристиками, що вже є
	Видалити країну	Видалення країни з усіма характеристиками

На рис. 2.2-2.3 зображено діаграму варіантів використання. Також було деталізовано варіанти використання у вигляді 25 історій користувачів з пріоритезацією, що містяться у додатку Б.

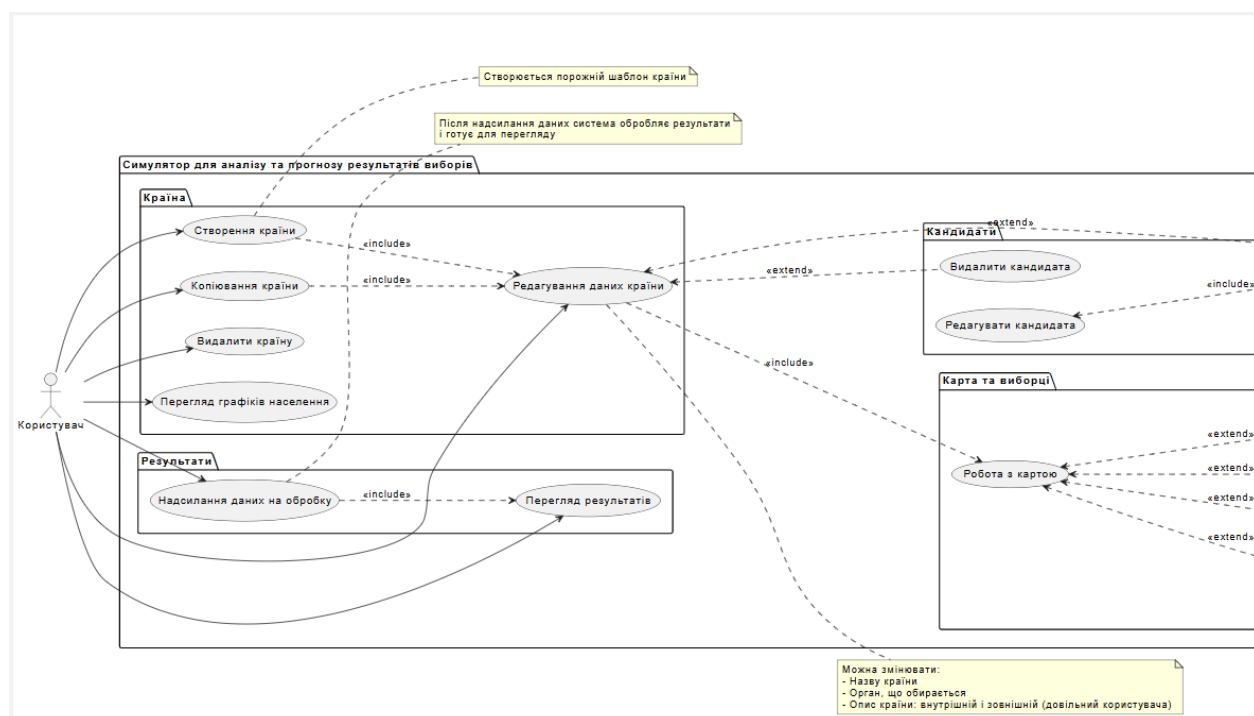


Рис. 2.2. Діаграма варіантів використання, частина перша

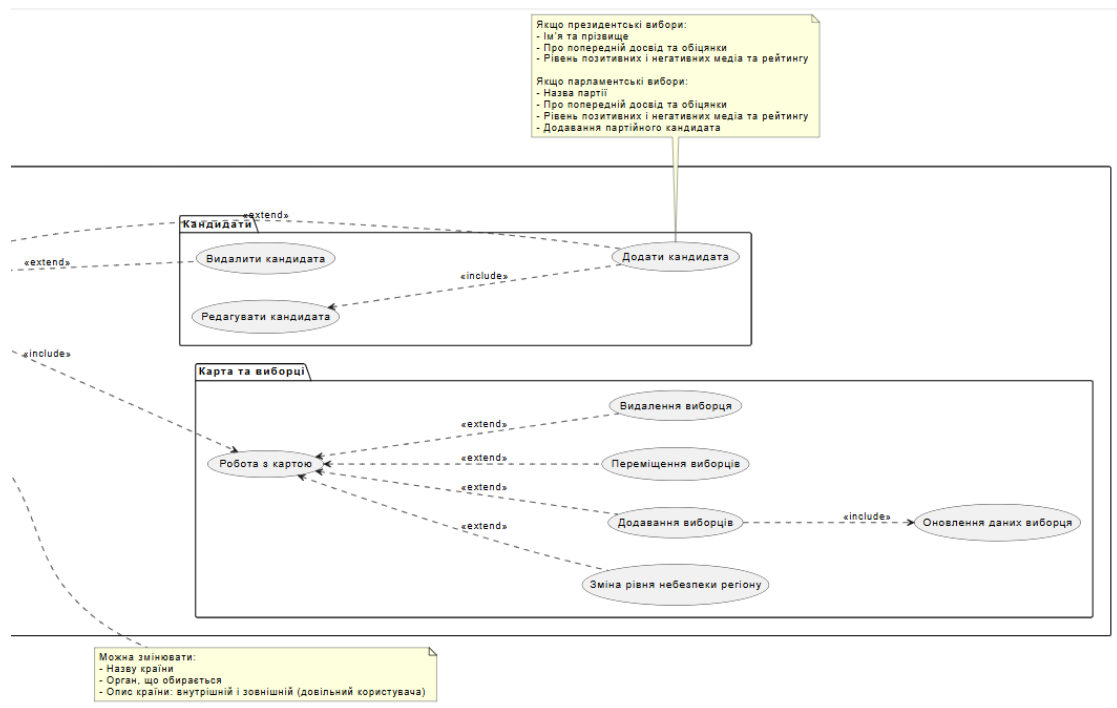


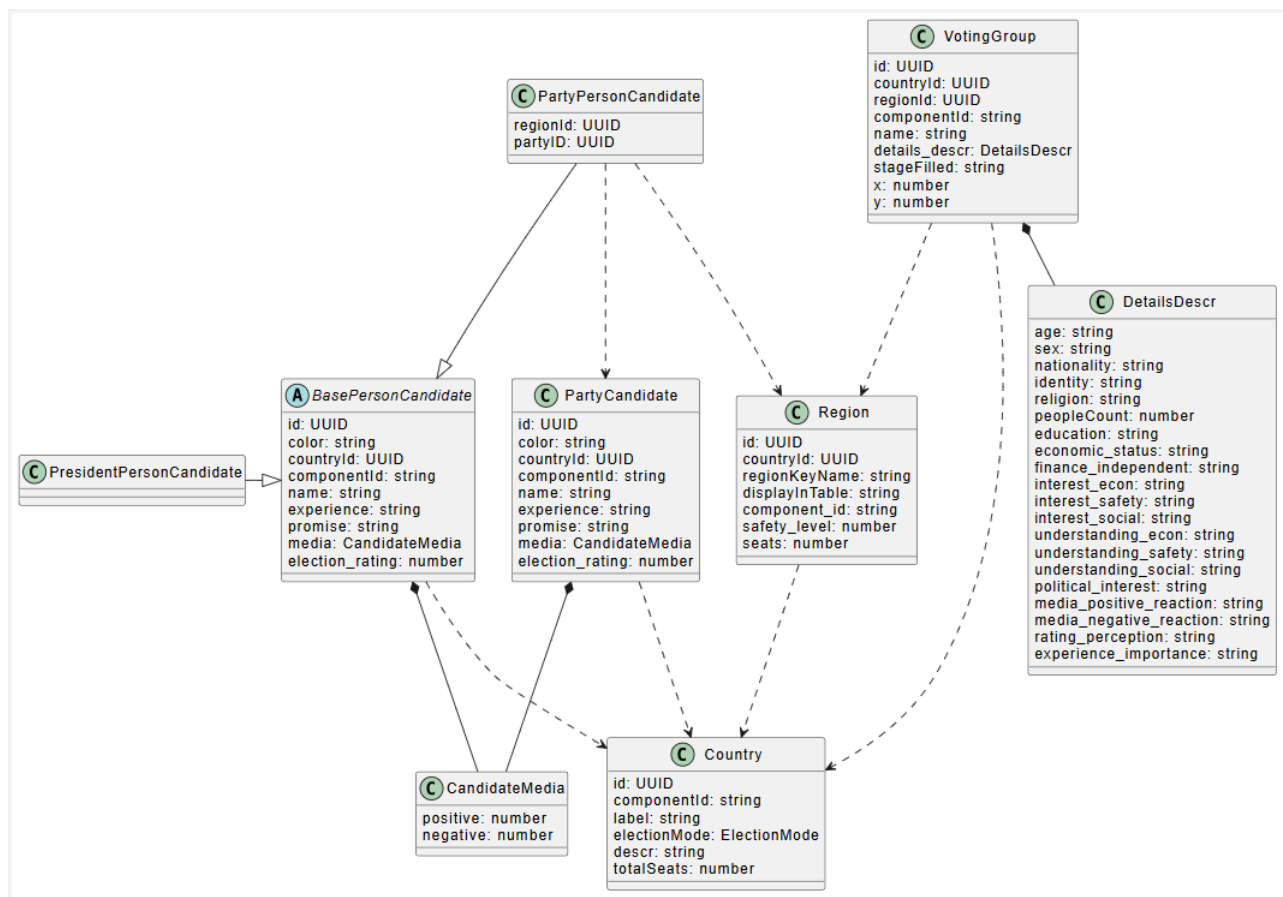
Рис. 2.3. Діаграма варіантів використання, частина друга

Створення країни дає базовий шаблон, з яким можна далі працювати. До країни є прив'язка регіонів, виборців та кандидатів. Виборці обов'язково мають зв'язок з регіоном, оскільки у системі колегії виборників (використовується в США) рахують забрані голоси кандидатів по регіонах.

Також залежно від типу виборів (президентські чи парламентські) форми кандидатів є різними. Для парламентських виборів важливо розглядати змішані системи, де частина обирається за пропорційно системою (тобто партія), а інша – за мажоритарною (тобто представник партії). Відповідно треба мати можливість додати партійного кандидата до партії та ще й з прив'язкою до регіону, в якому його обиратимуть.

Окрім цього на результат ще може вплинути виборчий округ (в системі це регіон). Округи поділяються на одномандатні та багатомандатні. Партія не може висунути більше кандидатів, ніж є в окрузі, тому необхідно забезпечити таку перевірку. В реальності кожна країна у своєму законодавстві закріплює або стає число мандатів за округом, або визначає пропорційно до кількості населення. Для симулятора перший варіант перекриває розгляд усіх варіантів як може іти підрахунок, а другий буде громіздким для користувача: чим більша кількість мандатів, тим більше кандидатів треба створити і заповнити для

На основі наведеної інформації можна зобразити залежності між основними елементами системи: країна, регіони, виборці та кандидати (рис. 2.4).



Структура серверної частини складається з арі, де послідовно викликаються функції бізнес-логіки; елементи модуля «Дослідження» поділено на підключення до LLM для аналізу текстового повідомлення країни та кандидатів у відповідні числові характеристики та прогнозування ідеального вектора кандидата виборця; модуль «Обрахування результатів» містить окремо кожен виборчу систему. Класи поділено на схеми країни, регіонів, кандидатів та виборців; також є класи для перевірки чи прийшов запит відповідно до заданої структури. Також було виокремлено допоміжні функції в окрему папку, як-от перетворення об'єктів з результатами у JSON.

На клієнтській частині всі React-компоненти поділено відповідно до їх функціоналу та секцій інтерфейсу (панелі налаштування, карти з таблицями, графіки, блок з кандидатами, панель результатів). Стилі виокремлено від компонентів та поділено у відповідні до них секції. Відокремлено знаходяться типи, фабрики для створення та копіювання, константи, повідомлення для користувачів, хуки, сервіси, сховища та утиліти.

Для основних елементів системи було додано чотири сховища (тобто для країни, регіонів, виборців та кандидатів) і ще одне для результатів, що прийшли з серверу.

Сервіси містяться об'єднання логіки при роботі з певним елементом. До прикладу, при переміщенні виборця необхідно зберегти нові координати та зробити перерахунок розподілу голосів (мандатів) по регіонах. Також у таблицях по регіонах переміщений виборець має оновити своє положення. Другим прикладом є додавання кандидата, де необхідно генерувати новий колір партії, щоб не повторювався в межах країни для одного типу виборів. Під час копіювання країни важливо було забезпечити зв'язок одне з одним всіх скопійованих елементів, надаючи для кожного новий унікальний ідентифікатор та змінюючи індекс активної країни, яка відображається користувачу. Саме тут найбільше допомогли фабрики, які можна було викликати в циклі при копіюванні кожного внутрішнього елемента.

2.2 Проєктування та реалізація алгоритмів

Загальний алгоритм роботи за стосунку полягає у тому, що користувач, створивши власну країну, заповнює необхідні характеристики та відправляє на сервер (рис. 2.5). На сервері спочатку здійснюється перетворення тексту країни в числові характеристики (табл. 2.3) та аналогічно характеристики усіх кандидатів (табл. 2.4). Далі характеристики країни та рівень безпеки регіону додаються до характеристик виборця, таким чином створюючи контекст, в якому є виборець. Ці дані формують значення, які відповідають рисам ідеального кандидата виборця. Наступним кроком для всіх виборців і кандидатів здійснюється перевірка подібності. На основі цієї подібності в

подальшому в кожного виборця розставляється пріоритет для кандидатів. Після цього відповідно до алгоритмів виборчих систем здійснюється загальний підрахунок на основі пріоритетів, що є.

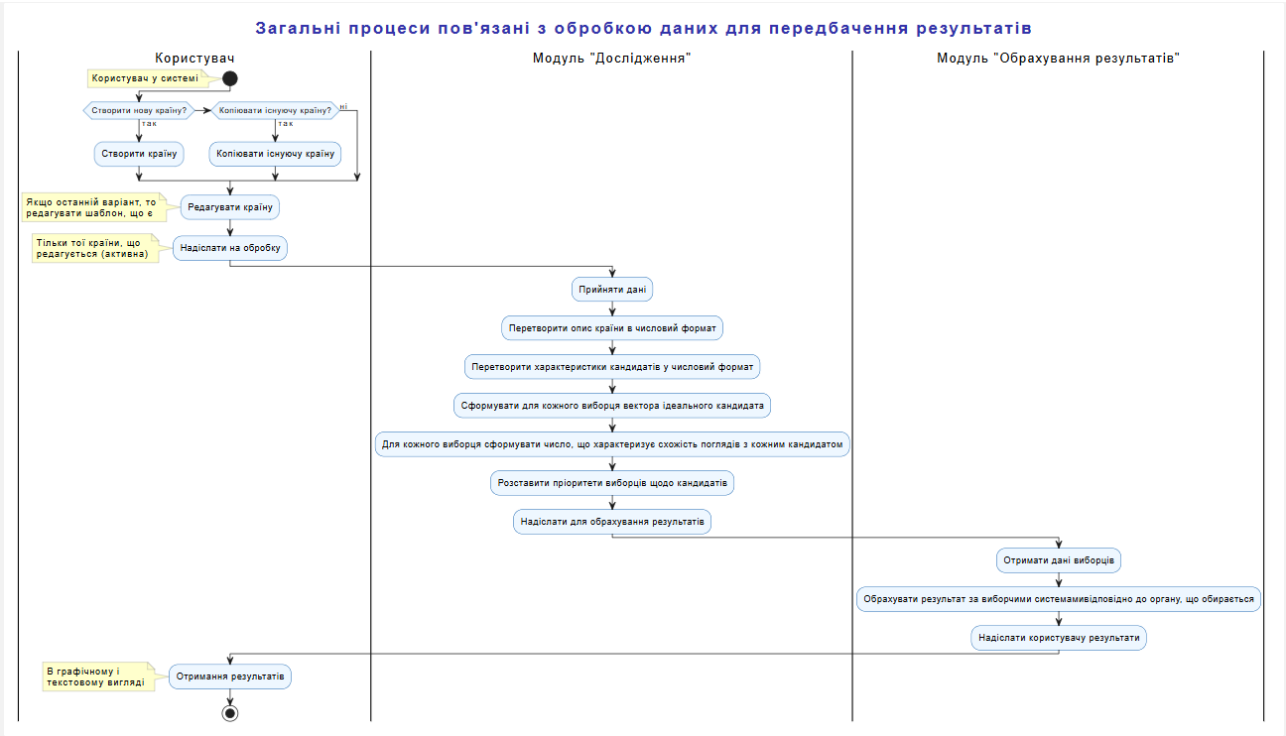


Рис. 2.5. Діаграма активності, що зображує загальний процес

Таблиця 2.3

Характеристики країни

Характеристика	Мінімальне значення	Максимальне значення
war_status	0 = повний мир	10 = активна війна
economic_crisis	0 = стабільна економіка	10 = колапс та гіперінфляція
corruption_level	0 = чистота та відсутність корупції	10 = системна корупція
media_freedom	0 = повна цензура	10 = вільне висловлення
political_stability	0 = хаос	10 = стабільна демократія

Таблиця 2.4

Характеристики кандидатів

Характеристика	Мінімальне значення	Максимальне значення
1	2	3
media_positive	0 = переважно негатив описано	10 = переважно позитив
transparency	0 = переважно приховування інформації	10 = повністю відкритий

1	2	3
program_simplicity	0 = комплексні і ґрунтовні пояснення	10 = легкі заклики
leadership_strength	1 = демократичний лідер	5 = авторитарний лідер
institutional_competence	0 = відсутність управлінського досвіду	10 = глибока експертиза
anti_populism	1 = суцільний популіст	5 = відсутність популізму
social_focus	0 = орієнтація на людину	10 = орієнтація на державу
rule_of_law	0 = виступає проти інституцій, що є	10 = сильна підтримка

В основі роботи лежить дві задачі: перша регресійного характеру для передбачення ймовірності виборця взяти участь у виборах і формування вектора ознак ідеального кандидата; друга рекомендаційна для вимірення схожості між виборцем та кандидатом. Реалізацію першої задачі описано в пунктах 2.3 та 2.4, для другої використовується сформований вектор з 8 ознак ідеального кандидата виборця та за допомогою косинусу подібності (функція взята в Scikit-learn) визначається схожість. Саме косинус подібності показує наскільки схожий напрямок векторів, ігноруючи величину. Далі всі значення сортуються в спадному порядку і найбільшому задається пріоритет 1, а найменшому N, де N – це кількість кандидатів. Саме ці пріоритети використовуються у виборчих системах. Фрагменти коду наведено в додатку В.

Щодо реалізованих виборчих систем для президентських виборів, то система може бути в один тур та кілька. Обов'язковою є перевірка ймовірності участі у виборах більше 60%. Якщо система передбачає кілька турів, то обов'язковою є перевірка чи наступний тур необхідний. Для таких систем, якщо є кандидат, що подолав 50%, цей кандидат вважається переможцем. Відповідно було створено абстрактний клас, що наслідують усі виборчі системи з методом для обрахунків, а багатотурові наслідують ще клас з методами для перевірки необхідності наступного туру і наступним етапом обрахунків. Варто зауважити, що всі системи приймають і повертають виборців, але результати

самих виборів можуть бути представлені у різній формі. Узагальнено інформацію наведено у таблиці 2.5.

Таблиця 2.5

Результати функцій виборчих систем

	Кількість турів	Зміни у об'єктів виборців	Кількість об'єктів, що отримується (без об'єктів виборці)	Результат для представлення
FPTP	Один	Додається словник з ключем назвою системи та вкладеним словником з ключем туром та	1	Повертається словник, що містить кандидатів та їхні відсотки серед тих хто проголосував
TRS	Один або два	ідентифікатором кандидата, якого обрано	1	Словник, що містить до кожного туру словник з кандидатів та їхніх відсотків
RCV	Від одного до N-1, де N-кількість кандидатів			
Колегія виборників	Один		2	Повертається словник, що містить кандидатів та їхні забрані голоси в регіонах загалом і словник окремо по кожному регіону переможця
Метод Кондорсе	Один	-	1	Матриця порівнянням пар 3

Загальну послідовність формування результатів наведено на діаграмі послідовностей на рис. 2.6. На ній зображено основні частини модулів на стороні серверу та функції, що викликаються з відповідними параметрами.

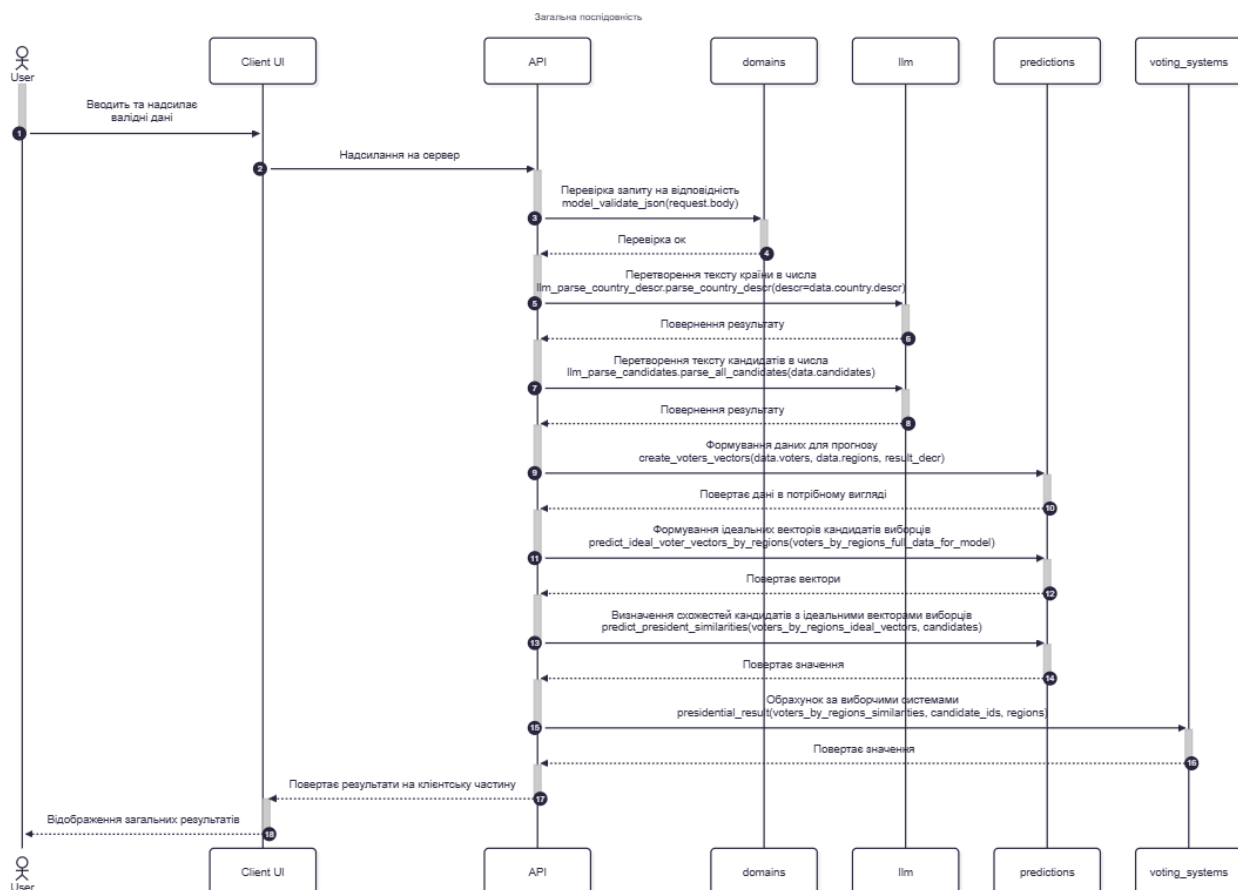


Рис. 2.6. Діаграма послідовностей

Також під час проєктування було виокремлено різні стани елементів (наприклад, виборця на рис. 2.7).

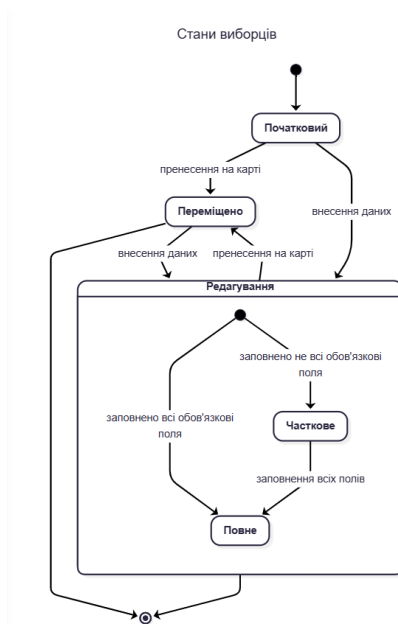


Рис. 2.7. Діаграма станів виборця

2.3 Створення набору даних

На просторах мережі не було знайдено набір даних, що допоміг би у реалізації задачі. Тому було вирішено самостійно згенерувати такі дані. Паралельно з цим було проведено опитування на основі, якого було сформовано ознаки.

Виборець містить 20 ознак (демографічні та внутрішні характеристики), що наведено у таблиці 2.6. Ці ознаки описують самого виборця. В набір додається 5 ознак країни (див. табл. 2.3), що формують вплив на виборця. Як результат необхідно отримати 9 ознак, що наведені у таблиці 2.7.

Таблиця 2.6

Набір ознак виборця для формування ідеального вектора кандидата

№	Ознака	Опис	Тип	Приклади (скорочено опис)
1	2	3	4	5
1	age	Вікова група	Категоріальна	18-30; 31-45; 45-60; 60+
2	gender	Стать		Жіноча; чоловіча
3	nationality	Національність		Титульна нація; національна меншина
4	identity	Ідентичність		Етнічна; громадянська; однакові
5	religion	Релігійний вплив на погляди	Числова	від 0 до 10
6	education	Рівень освіти	Категоріальна	9-11 класів; профтех/коледж; вища освіта; незакінчена вища
7	economic_status	Економічний статус домогосподарства		Низький; нижче середнього; середній; вище середнього; високий; заможний
8	finance_independent	Фінансова незалежність		Повністю залежний; переважно залежний; частково і так і так; переважно самостійний; повністю самостійний
9	interest_economy	Зацікавленість економікою		
10	interest_safety	Зацікавленість безпекою	Числова	від 0 до 10

Клосович І.А.

Савіцький Р.С.

1	2	3	4	5
11	interest_social	Зацікавленість соціальним життям	Числова	від 0 до 10
12	understanding_econ	Рівень розуміння економіки	Категоріальна	Майже не орієнтується у темі; розуміє загальні ідеї; може пояснити основні поняття; впевнено застосовує знання на практиці; може критично оцінювати інформацію
13	understanding_safety	Рівень розуміння безпеки		
14	understanding_social	Рівень розуміння соціального життя		
15	safety_region	Безпека в регіоні	Числова	від 1 до 5
16	political_interest	Рівень зацікавленості політичним життям	Числова	від 0 до 10
17	media_positive_reaction	Реакція на позитив у медіа	Числова	від 0 до 10
18	media_negative_reaction	Реакція на негатив у медіа	Числова	від 0 до 10
19	rating_perception	Сприйняття рейтингів	Числова	від 0 до 10
20	experience_importance	Важливість досвіду	Числова	від 0 до 10

Спершу було згенеровано характеристики лінійно, але це не є найкращим варіантом навіть для симулювання. Тому використовуючи функції сигмоїди, підсилення факторів через множення та інші було згенеровано нелінійний набір даних. Кожного згенерованого виборця було повторено за різних умов (війни, післявоєнного періоду, мирного часу, авторитаризму) та отримано різні

результуючі ознаки (рис. 2.8). Основні елементи генерування нелінійних даних наведено у додатку Г.

Таблиця 2.7

Набір результуючих ознак виборця

№	Ознака	Опис	Тип	Приклади
1	probability_take_part	Ймовірність прийти на вибори	Числова	від 0 до 100
2	candidate_positive_importance	Важливість позитиву в інформації від кандидата		від 0 до 10
3	candidate_negative_fair_importance	Важливість об'єктивності в інформації від кандидата		
4	like_easy_decision	Простота і складність рішень що підтримуються		
5	like_strong_leader_over_law	Схильність до потреби сильних керівників над законом і громадськими дискусіями		від 1 до 5
6	understand_government_institutions	Розуміння функціонування державних інституцій		від 0 до 10
7	every_person_is_expert	Схильність до думки що кожен громадянин експерт в політиці		від 1 до 5
8	person_or_government_importance	Що важливіше при ухваленні у суспільно-політичних питаннях		від 0 до 10
9	believe_government_institutions	Рівень довіри до державних інституцій		

З рис. 2.8. видно, що стандартне відхилення при нелінійному генеруванні більше, що означає сильне розкидання значень, у порівнянні з лінійним.

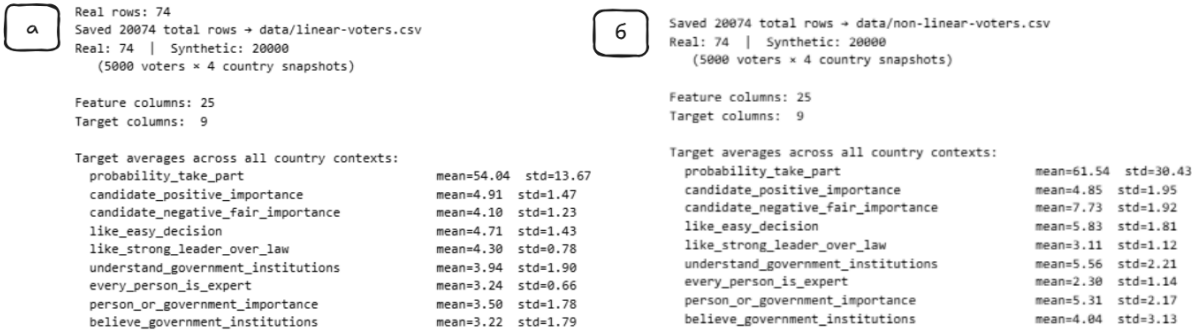


Рис. 2.8. Згенеровані дані: а – лінійно, б – нелінійно

Хоч опитування пройшло 74 людини, 90,5% це люди віком 18-30 років, 54,1% жінки та 45,9% чоловіки і воно цілком може містити похибки, але можна порівняти дані з опитування та згенеровані. На рис. 2.9-а зображено ймовірність прийти на вибори; оскільки зараз умови війни, то було взято окремо зріз виборців у цьому періоді і схожу тенденцію показує рис. 2.9-в. Аналогічно нелінійно згенеровані дані схожі до результатів опитування, що видно на рис. 2.10.

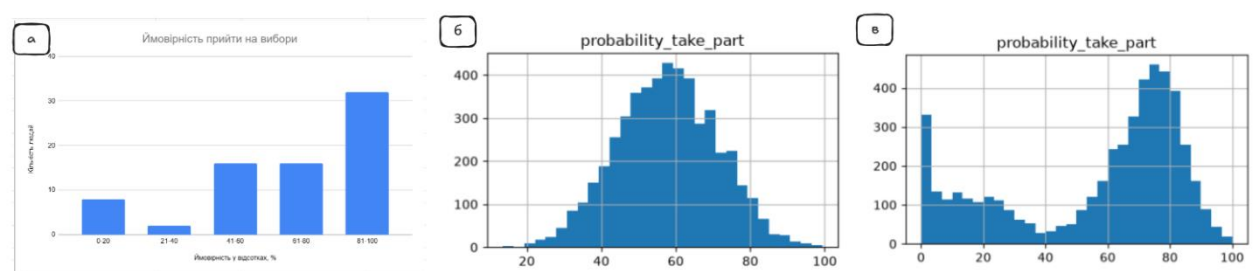


Рис. 2.9. Порівняння результатів ймовірності прийти на вибори: а – дані з опитування, б – лінійне генерування, в – нелінійне генерування

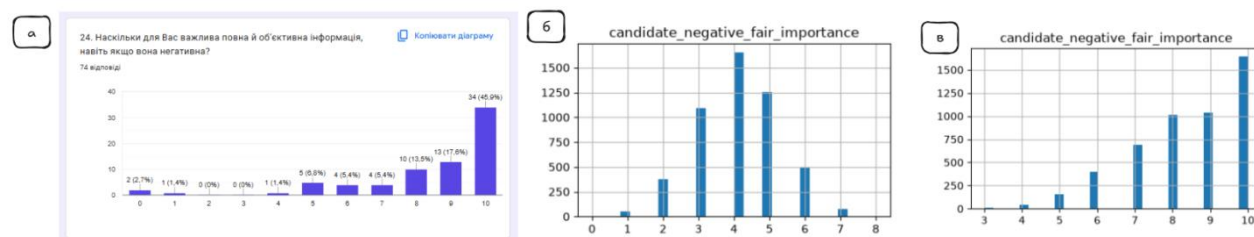


Рис. 2.10. Порівняння результатів важливості негативної інформації: а – дані з опитування, б – лінійне генерування, в – нелінійне генерування

Інші результати можна переглянути у додатку Д. Також можна продивитись матрицю кореляцій вихідних ознак (рис. 2.11). Наприклад, ймовірність участі у виборах має сильну позитивну кореляцію з розумінням державних інституцій та довірою до них; інакше кажучи ці ознаки підсилюють ймовірність. Сприйняття позитивної інформації від кандидатів позитивно корелюється з вподобанням простих рішень, потребою в сильному лідері та ставленням, що кожен може бути експертом. Трохи менша позитивна кореляція між тим, що важливо при ухваленні рішень у суспільно-політичних питаннях та розумінням державних інституцій, але наявна. Інші ознаки здебільшого не впливають одна на одну.

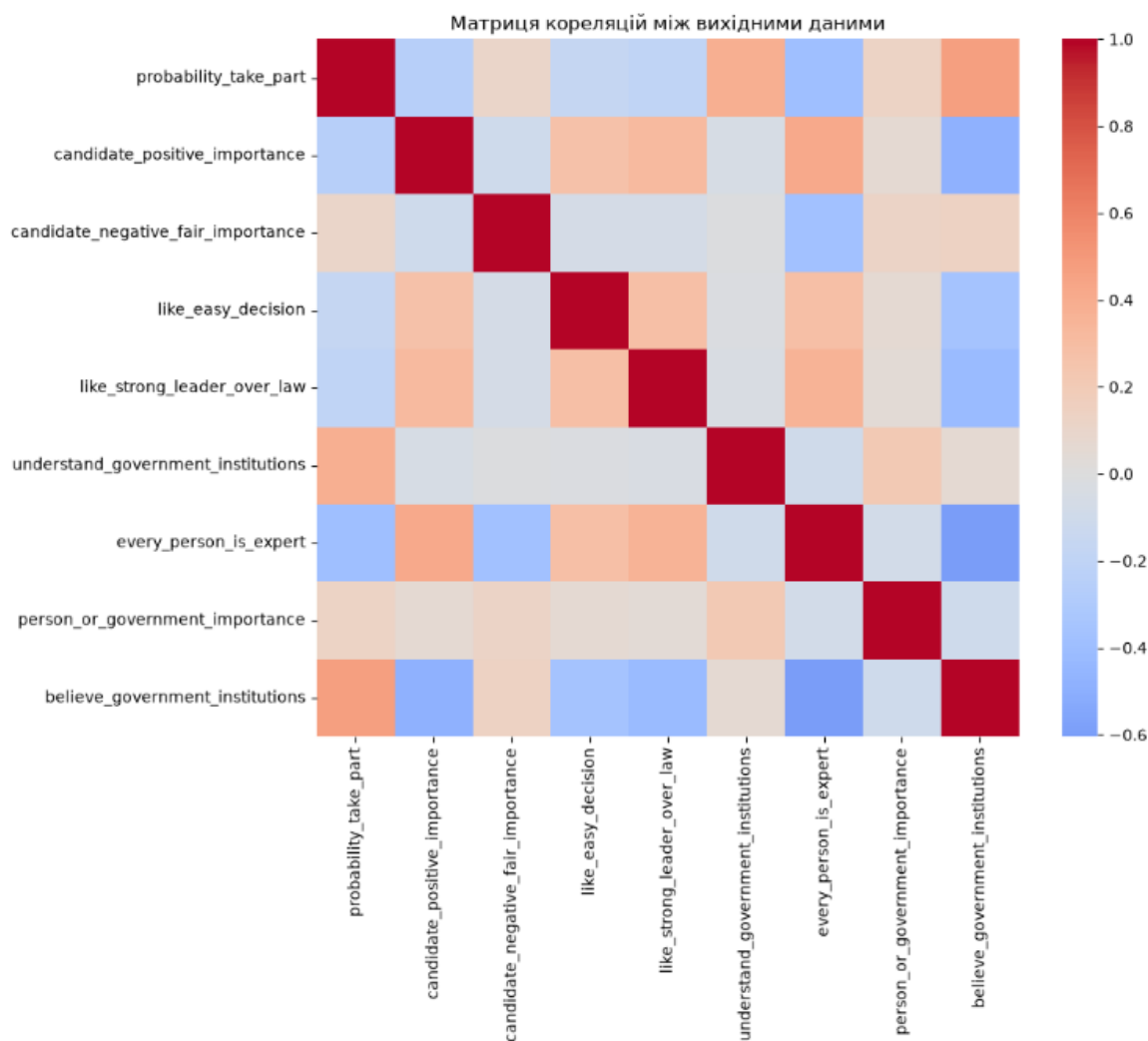


Рис. 2.11. Матриця кореляцій між вихідними даними

2.4 Вибір алгоритмів для регресійної задачі

Для вирішення поставленої задачі використовувались алгоритми Scikit-learn. Задача, як уже зазначалось, регресійна, містить нелінійні дані та ще й необхідно отримати 9 значень на виході.

Спочатку було підготовлено вхідні та вихідні дані. Назви колонок вихідних даних одразу зібрано в послідовності впливу ознак, так як було згенеровано (розуміння інституцій впливає на довіру до них, а це впливає на думку чи кожен є експертом і т. ін). Вхідні колонки відповідно є числовими і категоріальними. Числові значення не змінювали, а от категоріальні необхідно було. Частина таких ознак перетворено за допомогою OrdinalEncoder, даючи цим ознака порядкові номери [35]. Це означає що порядок впливає на

важливість значення (використано для віку, освіти, економічного статусу, рівня фінансової незалежності та розуміння). Інші ознаки, як-от стать, національність та ідентичність, лишилися номінальними і тому використовувся метод OneHotEncoder [36]. Він перетворює категоріальні ознаки на набір бінарних стовпців, де 1 вказує на наявність категорії, а 0 – на її відсутність.

Важливо було також поділити набір даних на групи під певних виборців, щоб уникнути того, що навчання і тестування буде на одному і тому ж виборці, але з різним станом країни. Отже, створено групи за `base_voter_id`, який є звичайним порядковим числом. Це означає, що всі виборці з однаковим цим ідентифікатором потрапляють або до тренування, або до тестування, але всі варіанти виборця з різним станом країни разом.

Наступним кроком розглядались два алгоритми: `RandomForestRegressor` та `HistGradientBoostingRegressor` [37, 38]. Обидва алгоритми добре працюють з нелінійними та ненормалізованими даними. `RandomForestRegressor` підбирає кілька дерев рішень на різних підвибірках набору даних, потім використовує усереднення для підвищення точності прогнозу та зменшення перенавчання. Також він має вбудовану підтримку пропущених значень. `HistGradientBoostingRegressor` є набагато швидшим і підходить для наборів даних з кількістю рядків більше 10 тис. і також гарно працює з пропущеними значеннями. На відміну від RF, HGBT має опцію ранньої зупинки, щоб уникнути додавання нових дерев.

Для отримання кількох вихідних колонок, а не одної, доцільно використати моделі `MultiOutputRegressor` та `RegressorChain` [39, 40]. Їхня відмінність у тому, що перша модель на кожну колонку підбирає регресором значення окремо, коли друга робить прогноз для одної колонки, а наступна колонка вже залежить від наданих значень і вже спрогнозованих попередніх вихідних значень. Саме з цією метою підготовлено порядок вихідних даних.

Отже, було проведено навчання кожної моделі на трьох групах даних і в результаті одержано, що найкраща модель: `HistGradientBoostingRegressor` разом з `RegressorChain` (рис. 2.12).

а-1	20074 rows loaded Навчання моделі: Multi_RandomForestRegressor... MAE std: 0.0013887490789613317 R2 std: 0.0012192770027286569 MAE = 1.593 R ² = 0.503 MSE = 7.706 EVS = 0.503	6-1	Навчання моделі: Multi_HistGradientBoostingRegressor... MAE std: 0.002540290275092717 R2 std: 0.0008314965268494401 MAE = 1.568 R ² = 0.516 MSE = 7.364 EVS = 0.516
а-2	Навчання моделі: Chain_RandomForestRegressor... MAE std: 0.00145121764681595 R2 std: 0.0013580468065485973 MAE = 1.591 R ² = 0.505 MSE = 7.825 EVS = 0.505	6-2	Навчання моделі: Chain_HistGradientBoostingRegressor... MAE std: 0.0027170564321770713 R2 std: 0.0012045161847796693 MAE = 1.565 R ² = 0.517 MSE = 7.447 EVS = 0.517

Рис. 2.12. Результати навчання моделей

Стандартне відхилення під час крос-валідації для усіх моделей є незначним. Це вказує на те, що моделі є стабільними. Середня абсолютна похибка (MAE) найкращий результат показано в HGBT + RC, він є найменшим (див. рис. 2.12-б-2). Коефіцієнт детермінації (R^2) теж найкращий в цієї моделі (іде пояснення 51,7% змін у даних, що є непогано для сфери соціології та політичних вподобань). А от найкращий результат середньої квадратичної помилки (MSE) є в HGBT+ MultiOutputRegressor. Те, що пояснена оцінка дисперсії (EVS) є такою ж як коефіцієнт детермінації, вказує на стабільність моделі.

На основі отриманих оцінок для подальшої роботи використовувались моделі HGBT + RC, що є логічним з огляду на задачу: характеристики залежать одна від одної. Код тренування наведено у додатку Е.

Більшість моделей машинного навчання добре відтворюють середні значення, але не можуть формувати коректний статистичний розподіл. Оскільки модель згладжує індивідуальні відмінності між виборцями та наближає значення до середніх, після навчання моделі було проведено оцінювання розподілу залишків.

Спершу здійснювався діагностичний аналіз якості моделі на тренувальних і реальних даних. В результаті було отримано випадок поганої узагальнювальної здатності моделі. Модель непогано вчиться на синтетичних даних, але погано переноситься на реальний розподіл (різниця між R^2). Також втрачається варіативність: на тренувальних розкид частково відтворюється, а на

реальних відбувається значне стиснення до середнього (різниця в kept).
Результати діагностики наведено у таблиці 2.8 та таблиці 2.9.

Таблиця 2.8

Діагностика на тренувальних даних

Характеристика	R ²	MAE	std_true	std_pred	kept, %
understand_government_institutions	0,53	1,19	2,17	1,56	72
believe_government_institutions	0,82	1,07	3,08	2,76	90
every_person_is_expert	0,77	0,48	1,15	1,00	87
probability_take_part	0,95	5,01	30,36	29,57	97
candidate_positive_importance	0,52	1,04	1,90	1,40	74
candidate_negative_fair_importance	0,33	1,29	1,94	1,06	55
like_easy_decision	0,30	1,21	1,79	0,93	52
like_strong_leader_over_law	0,40	0,72	1,12	0,66	59
person_or_government_importance	0,15	1,62	2,17	0,80	37
MEAN	0,53	1,51	5,08	4,42	69

Таблиця 2.9

Діагностика на реальних даних

Характеристика	R ²	MAE	std_true	std_pred	kept, %
understand_government_institutions	0,21	1,60	2,37	1,41	59
believe_government_institutions	0,18	1,46	2,17	0,34	16
every_person_is_expert	0,19	0,87	1,17	0,27	24
probability_take_part	0,07	22,46	29,57	15,12	51
candidate_positive_importance	0,28	1,78	2,73	1,09	40
candidate_negative_fair_importance	0,14	1,60	2,32	0,45	19
like_easy_decision	0,14	1,78	2,55	0,48	19
like_strong_leader_over_law	0,14	0,85	1,12	0,18	16
person_or_government_importance	0,15	2,18	2,91	0,60	20
MEAN	0,17	3,84	5,21	2,22	29

Далі для кожної прогнозованої характеристики обчислювалось стандартне відхилення залишків між реальними та передбачуваними значеннями. Отримані параметри використано для генерації випадкового шуму з нормального розподілу. Це дало змогу відтворити варіативність електорату та уникнути надмірної однорідності.

Оцінки стандартних відхилень наведено у таблиці 2.10.

Таблиця 2.10

Оцінки стандартних відхилень

Характеристика	Значення
understand_government_institutions	1,48
believe_government_institutions	1,36
every_person_is_expert	0,55
probability_take_part	6,94
candidate_positive_importance	1,31
candidate_negative_fair_importance	1,57
like_easy_decision	1,50
like_strong_leader_over_law	0,88
person_or_government_importance	1,98

Висновки до другого розділу

Другий розділ стосувався проєктування системи. Загалом у розділі було розглянуто різні варіанти реалізації системи, обрано оптимальний варіант, визначено варіанти використання, описано структуру проєкту. Також у розділі продемонстровано проєктування та реалізацію алгоритмів системи, створення набору даних та аналіз алгоритмів для регресійної задачі.

У першому підпункті описано варіанти використання, які деталізовано в історіях користувачів. Варіанти використання продемонстровано на однойменній діаграмі. Ретельно розглянуто зв'язки між основними елементами системи та описано структуру клієнтської і серверної частин.

В другому підпункті охарактеризовано загальний алгоритм роботи, результат наведено на діаграмі активності. Було визначено, що в основі роботи

лежить дві задачі: перша є регресійно для передбачення ідеального вектору кандидата, друга рекомендаційна. Розглянуто різні виборчі системи та різницю результатів у кожній з них. На діаграмі послідовностей наведено взаємодію модулів у часі.

Було створено штучний набір даних для задачі. Виборець містить 20 ознак, що описують його, ще 5 є контекстом країни, в якій він є. Характеристики згенеровано двома шляхами: лінійно та нелінійно. Результати двох генерувань було порівняно з вибіркою з проведеного опитування.

Після формування даних був крок обрання необхідного алгоритму. Лінійні алгоритми було одразу відкинуто, для розгляду взято RandomForestRegressor та HistGradientBoostingRegressor, які взаємодіяли разом з MultiOutputRegressor та RegressorChain. За результатами найкращою виявилась комбінація HGBT + RC, яку було використано впродовж розробки.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА РОБОТА З СИСТЕМОЮ

3.1 Налаштування системи

Початок роботи з будь-яким проектом починається з налаштування Git та віддаленого репозиторію. Для ефективного ведення роботи було розглянуто різні моделі розгалуження Git – Git Flow. Вибір методології залежить від розміру команди та проекту. Незважаючи на те, що в команді одна людина, робити все в одній гілці є поганою організацією.

Перша розглянута методологія це Main-Only Strategy [41]. Розробка ведеться у двох гілках: main та develop. Її переваги найкраще проявляються у малих командах і проектах з коротким життєвим циклом. Але є ризики, що прямі зміни можуть призвести до нестабільності.

Кращим варіантом виступає Feature Branching. Окрім гілок main та develop, від гілки розробки створюються гілки для функцій. Такий підхід дозволяє паралельно виконувати різні завдання та тестувати кожен функцію окремо. Усі зміни з гілок функцій спершу вливаються у гілку розробки, а далі (за умови успішного проходження всіх тестів) у головну. Ізолюваність гілок мінімізує ризик впливу однієї недопрацьованої функції на інші напрями розробки та забезпечує стабільність основної кодової бази. Команда розробників може одночасно працювати над декількома функціональностями без конфліктів коду до моменту їх інтеграції у спільну гілку.

До попередньої методології подібний GitHub Flow з відсутністю гілки для розробки та вливанням усіх гілок функцій одразу в головну. Для роботи обрано стратегію Feature Branching через ізолюваність та легкість відкату у разі наявності помилок у функції. Записи змін зроблено у стилі Conventional Commit [42]. Така структура комітів забезпечує чітку класифікацію змін за типом (feat, fix, refactor, docs та інші) з прискоренням навігації історією проекту. Автоматизовані інструменти можуть аналізувати такі коміти для генерації списків змін (changelog) та визначення версій продукту за принципом семантичного версіонування. Поєднання Feature Branching зі стандартизованими повідомленнями комітів формує надійну основу для

довгострокової підтримки проєкту та залучення нових учасників до команди розробників. Результат наведено на рис. 3.1.

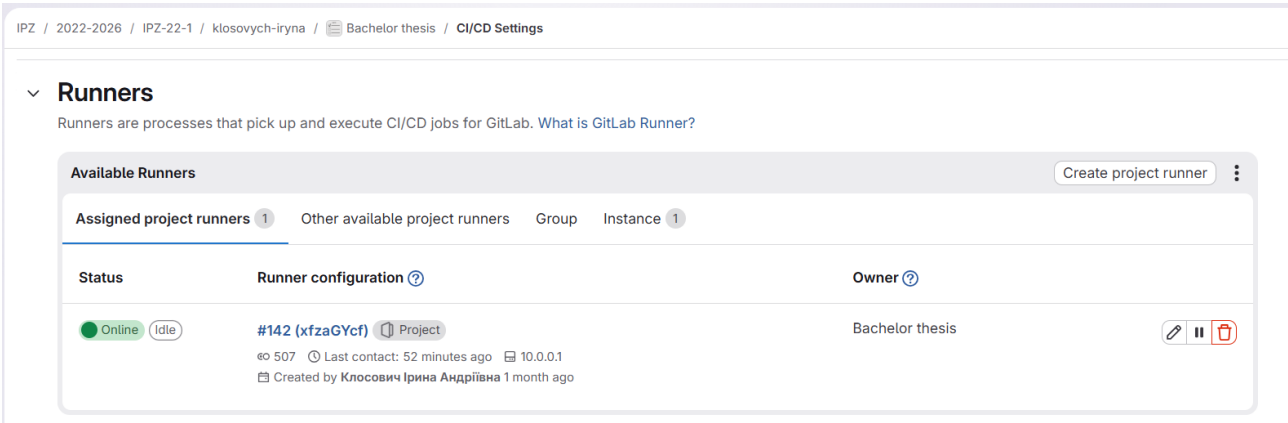


Рис. 3.1. Зареєстрований ранер

Також основну гілку було захищено, щоби зміни могли вносити лише супроводжувачі (Maintainers). Показано на рис. 3.2.

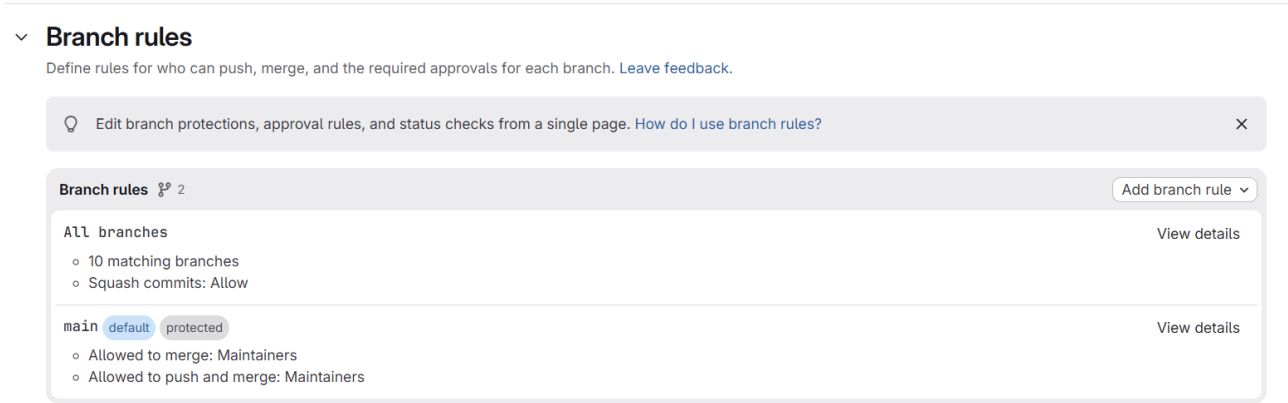


Рис. 3.2. Правила для гілок

Для розгортання на хмарних сервісах необхідний був GitHub репозиторій, оскільки ці сервіси не підтримували зв'язок з саморозгорнутим GitLab. Наведено результат на рис. 3.3. Зміни до дзеркального репозиторію вносяться автоматично після надсилання кожної зміни з будь-якої гілки.

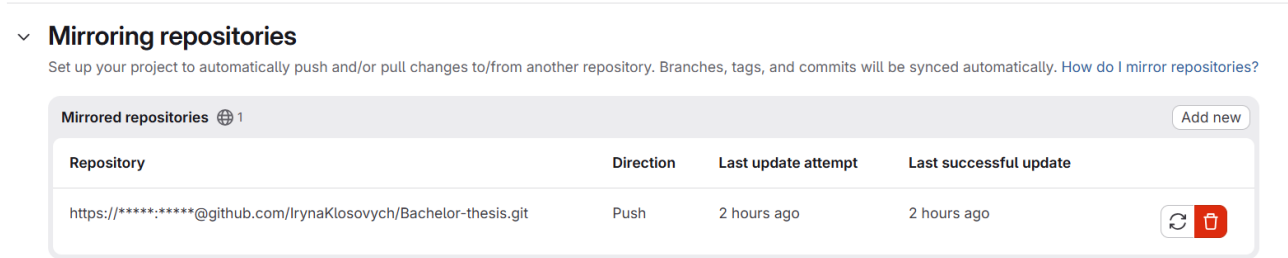


Рис. 3.3. Дзеркало GitHub

Оскільки розробка поділена на серверну та клієнтську частини, до кожної було додано Dockerfile. Серверну частину разом з моделлю було виставлено на хмарному сервісі Render, а клієнтську – на Vercel [43, 44]. Діаграма розгортання представлена на рис. 3.4.

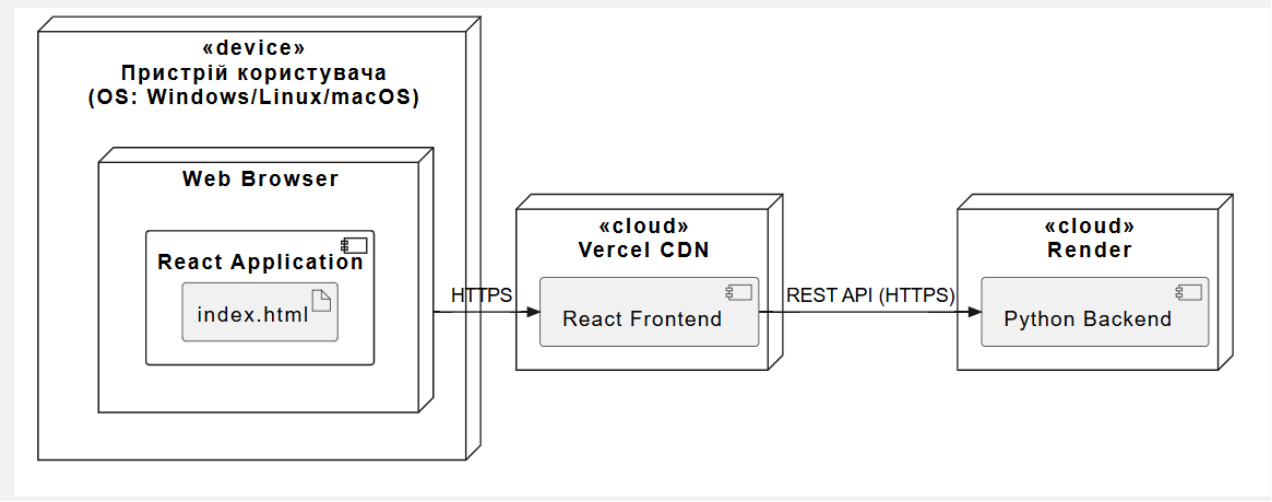


Рис. 3.4. Діаграма розгортання

З діаграми бачимо, що з пристрою користувача за допомогою HTTPS запиту є з'єднання з Vercel, у свою чергу який зв'язується через REST API з Render. Результати розгортання наведено на рис. 3.5-3.6 для Vercel та Render відповідно.

У Dockerfile для обох частин спочатку обирається мінімальний за обсягом Linux дистрибутив і створюється робоча директорія. Далі здійснюється копіювання усіх залежностей та їх встановлення. Останнім рядком є прописана команда запуску.

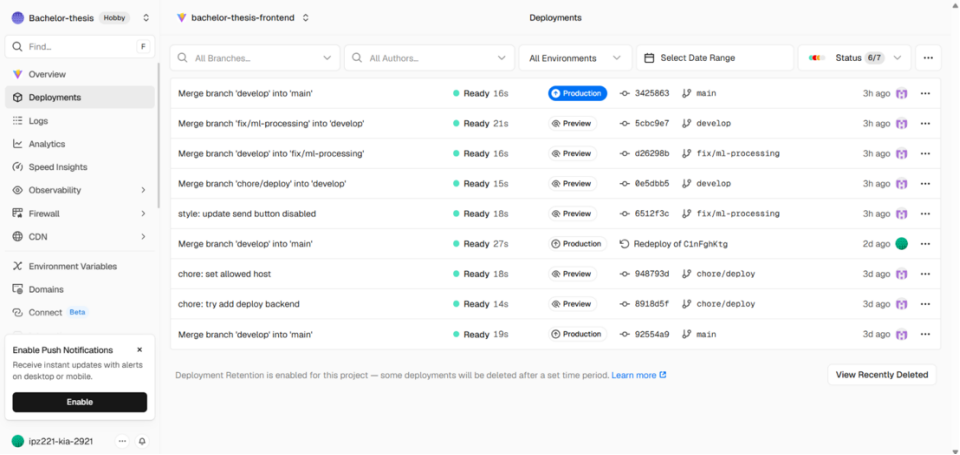


Рис. 3.5. Результат розгортання на Vercel

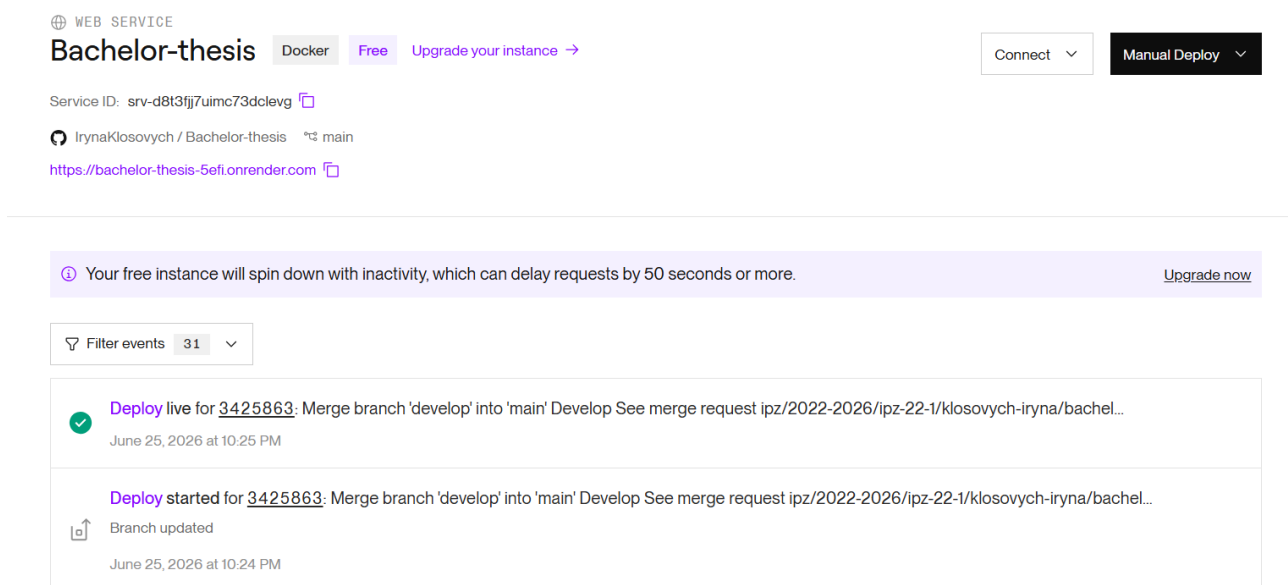


Рис. 3.6. Результат розгортання на Render

Щодо конфігураційного файлу `.gitlab-ci.yml`, то в ньому автоматизовано три стадії: збірку, тестування та Docker-збірку для двох частин проєкту. На етапі збірки встановлюються всі залежності та відбувається перевірка коректності конфігурацій за допомогою відповідних команд для бекенду і фронтенду. Це дозволяє виявляти помилки ще до запуску застосунку. На етапі тестування запускаються тести, що перевіряються бізнес-логіку програми.

Останній етап пов'язаний з контейнеризацією застосунку та використовує концепцію Docker-in-Docker (DinD). Суть полягає в тому, що запускається окремий Docker daemon (головний фоновий процес Docker) всередині контейнера і слухається на внутрішньому порту 2376 (TLS). В змінні внесено стандартний шлях, де зберігаються TLS сертифікати, `/certs`. Docker CLI читає сертифікати з тої папки і підключається по TLS. Якщо прибрати цей шлях, то з'єднання буде без шифрування, що є небезпечним для випуску.

Налаштування передачі файлів між стадіями та кешування для прискорення pipeline не реалізовувалось.

3.2 Інтерфейс програми

Розробка під навчально-дослідницьку тематику вимагає багато доступних і зрозумілих елементів для роботи. Тому для користувачів з маленьким

розміром екрану виводиться повідомлення, що краще змінити пристрій (рис. 3.7).

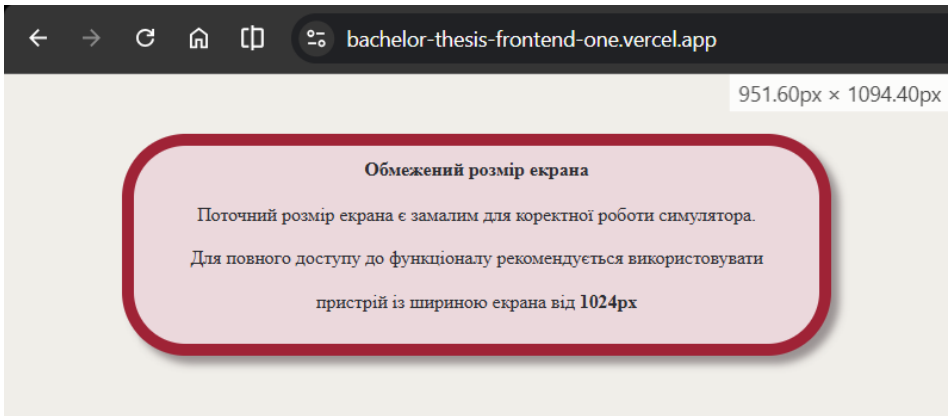


Рис. 3.7. Результат виведення повідомлення про недостатній розмір екрану

Якщо було здійснено вхід із задовільним розміром екран, то користувач може побачити повідомлення про розробку та кнопку для створення країни (рис. 3.8). Лічильник для країн буде порожній. Як тільки буде натиснуто кнопку для створення країни (рис. 3.9-1), то з’явиться поле для роботи. Під номером 2 на рис. 3.9 показано місце для зміни назви, цифрою 3 позначено контейнер для країни і виборців візуально. Кнопка під номером 4 додає виборця до карти і в контейнер під цифрою 5, де будуть вводитись характеристики виборців, відсортованих по регіонах. Цифрою 6 позначено перемикач між країнами, що показує послідовний номер країни із загальної кількості. Панель під номером 7 містить кнопки для копіювання та видалення країни.

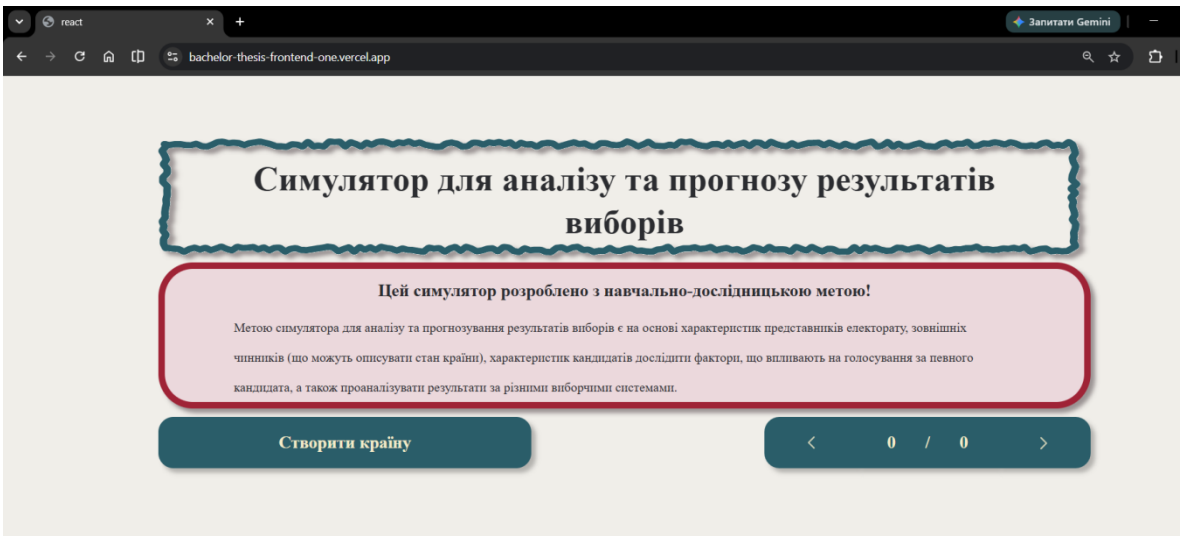


Рис. 3.8. Початок роботи із системою

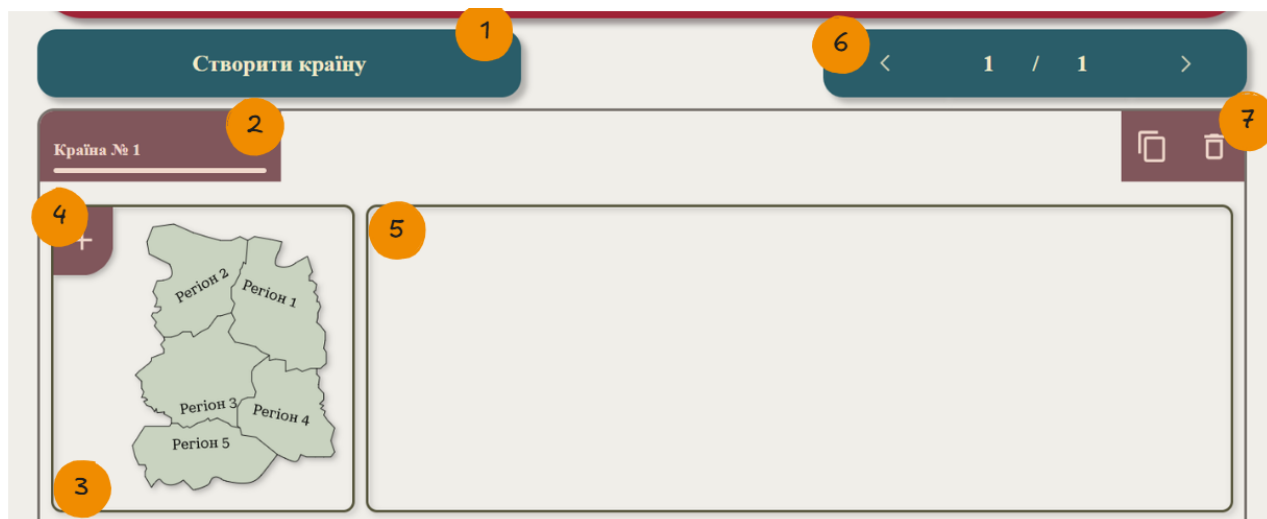


Рис. 3.9. Елементи для роботи загалом та з країною

На рис. 3.10 зображено загальну та регіональні статеві-вікові піраміди. При змінах характеристик виборчої групи (статі, віку та кількості людей) вони будуть оновлюватись. Внизу графіків позначено кількість голосів для президентських виборів та кількість мандатів для парламентських. Цей тип графіків реалізовано з метою візуального відображення структури населення країни для користувача для подальшого аналізу, а також уникнення переваги одної категорії населення в регіоні чи країні, що робить баланс більш реалістичним.

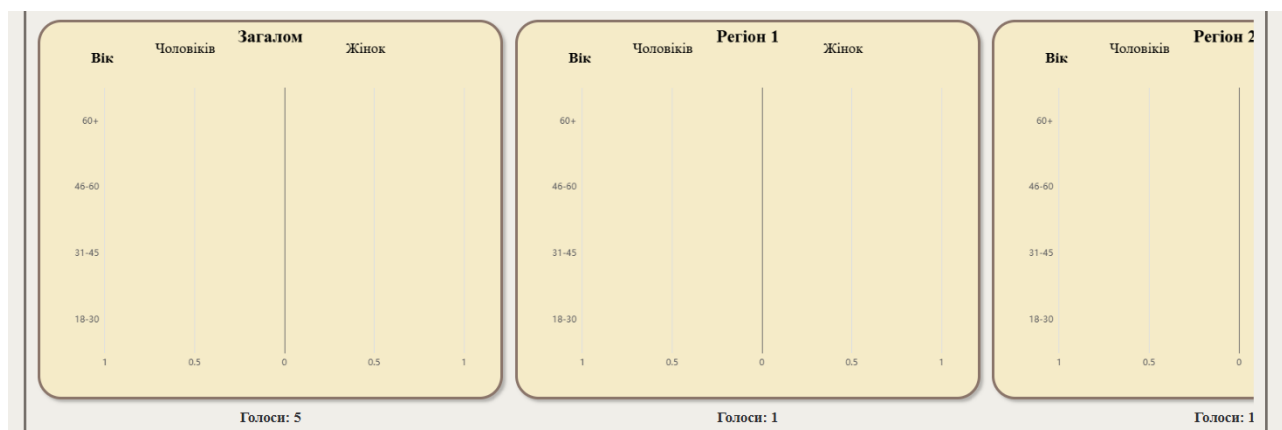


Рис. 3.10. Статеві-вікові піраміди

На рис. 3.11 зображено такі інші елементи: 1 – перемикач між типом виборів за органом, що обирається; 2 – поле для заповнення опису країни; 3 – кнопка для додавання кандидата (або партії); 4 – контейнер для доданих виборців; 5 – кнопка для обрахування результатів; 6 – контейнер для результатів.

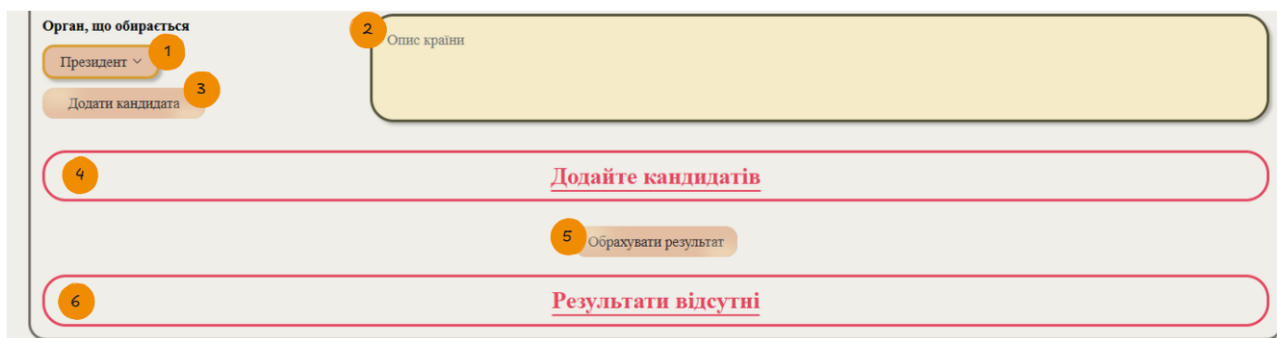


Рис. 3.11. Інші елементи країни

На рис. 3.12 зображено під цифрою 1 зміну назви країни. На цьому ж рисунку при додаванні виборчої групи на карту додано і в таблиці (цифра 2), а при переміщенні групи виділяються зовнішньою лінією.



Рис. 3.12. Зміни, що зроблено в країні

Якщо натиснути на регіон, його назва виділиться червоним кольором, як на рис. 3.13-1 та з'явиться панель для вибору рівня небезпеки (цифра 2 на рис. 3.13). При кліку на виборчу групу можна видалити її у панелі під номером 3.

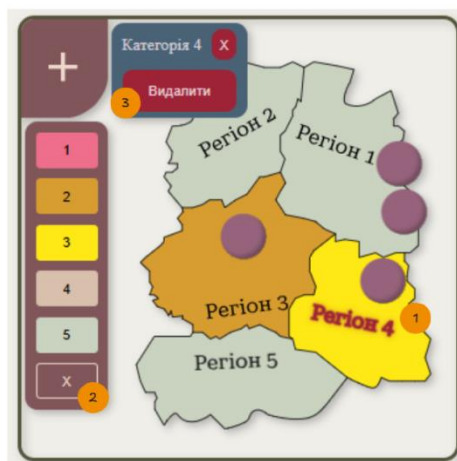


Рис. 3.13. Карта з елементами зміни небезпеки та виборцями

Також групи, які більші 1000 людей є більшим за розміром (рис. 3.14). При заповненні інформації про груп на половину колір змінюється на оранжевий (див. рис. 3.14), а якщо повністю, то зелений (рис. 3.15).

Region 1

Кількість людей	Рівень освіти	
100	Вища освіта ▾	Сер
1001	Оберіть рівень освіти ▾	Обе

Рис. 3.14. Зміна кількості людей і кольору групи при заповненні

Region 1

дія	Реакція на негатив у медіа	Рівень сприйняття рейтингів	Важливість досвіду
	6 ▾	4 ▾	4 ▾
в медіа ▾	Оберіть рівень впливу негативу в медіа ▾	Оберіть рівень сприйняття рейтингів ▾	Оберіть рівень важливості досвіду ▾

Рис. 3.15. Повне заповнення виборчої групи

Приклад оновлених статеві-вікових пірамід і голосів наведено на рис. 3.16.

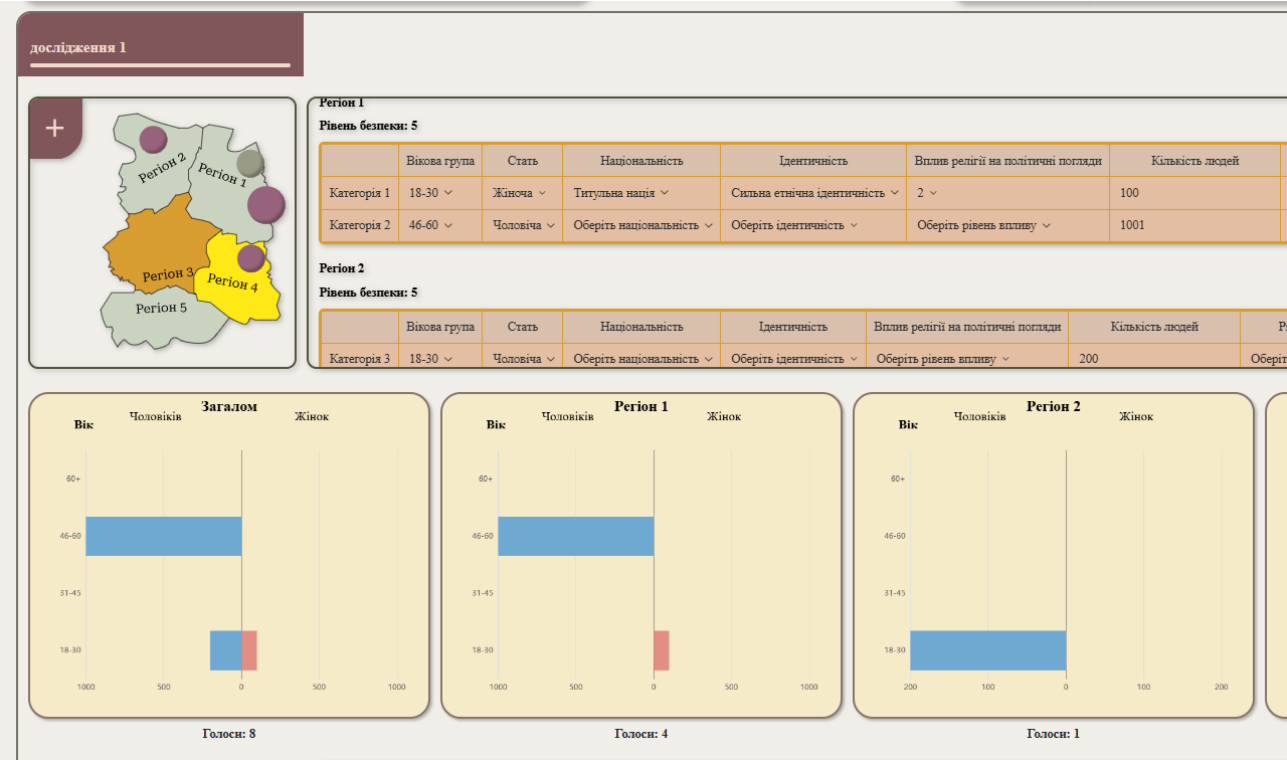


Рис. 3.16. Зміна статеві-вікових пірамід

Форма для заповнення президента (рис. 3.17) та партії (рис. 3.18) містять майже одні й ті самі поля, але в партії можна додати кандидата до конкретного регіону, де він буде висуватись. Якщо кандидата додано до одного регіону і необхідно змінити, то є панелька, яка показує скільки ще можна додати кандидатів балотуватись у регіоні. Кожна форма має свій колір.

Рис. 3.17. Форма кандидата в президенти

Рис. 3.18. Форма для заповнення інформації про партію

Скопійована країна містить усі елементи, що й та, яку захотіли скопіювати, крім результатів. При додаванні і копіюванні активною країною стає нова, а лічильник змінює кількість. Приклад на рис. 3.19.

Рис. 3.19. Скопійована країна

Якщо користувач захоче надіслати незаповнені дані або заповнені не зовсім правильно, в правому нижньому кутку будуть висвічені помилки як на рис. 3.20-3.21.

Рис. 3.20. Помилки при президентському режимі

Рис. 3.21. Помилки при парламентському режимі

Приклад заповненої країни міститься на рис. 3.22. Коли все гаразд при кліку на кнопку дані надсилаються на сервер, а кнопка блокується як на рис. 3.23.

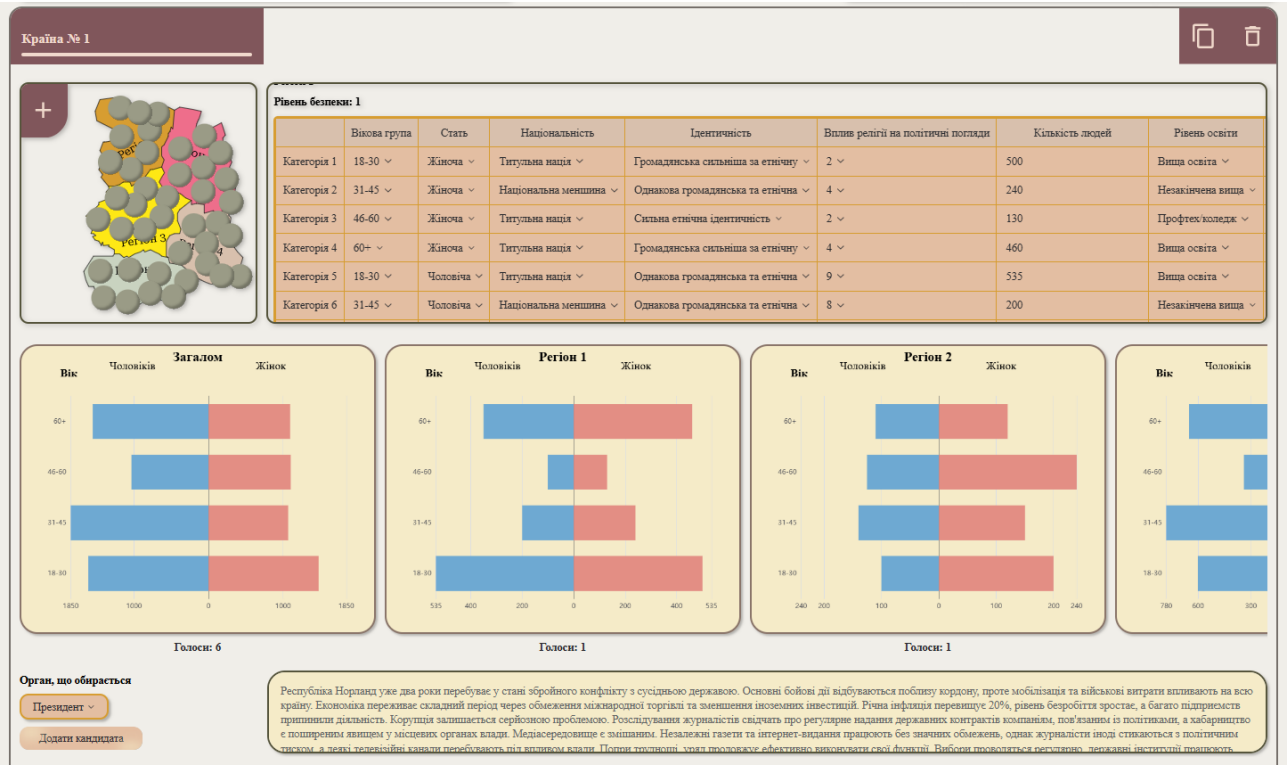


Рис. 3.22. Приклад заповненої країни



Рис. 3.23. Обрахування результатів

Отримані результати можна переглянути у таблиці, де вказано ймовірність прийти на вибори та пріоритети виборців за кандидатів (рис. 3.24). Нижче є панель для перемикання між відповідними виборчими системами, збоку і знизу від цієї панелі будуть зображені результати виборчої системи у найзручнішому вигляді (рис. 3.25–3.30). Якщо наявна карта, то при наведенні на

виборчу групу буде нотатка (див. рис. 3.25). Карта зручна при аналізі перетікання голосів.

Регіон 1						
	Ймовірність прийти на вибори у %	Пріоритети виборців за кандидатів				
		Кандидат А	Кандидат Б	Кандидат В	Кандидат Г	Кандидат Д
Категорія 1	69.30	2	4	3	5	1
Категорія 2	74.19	3	2	4	5	1
Категорія 3	100.00	3	2	4	5	1
Категорія 4	75.26	1	4	3	5	2
Категорія 5	100.00	2	4	1	5	3

Рис. 3.24. Таблиця результатів

В системах FPTP (див. рис. 3.25), TRS (див. рис. 3.26) та альтернативного голосування (див. рис. 3.28-3.29) результати відображаються у відсотках, коли в системі колегії виборників (див. рис. 3.27) беруться голоси регіонів.

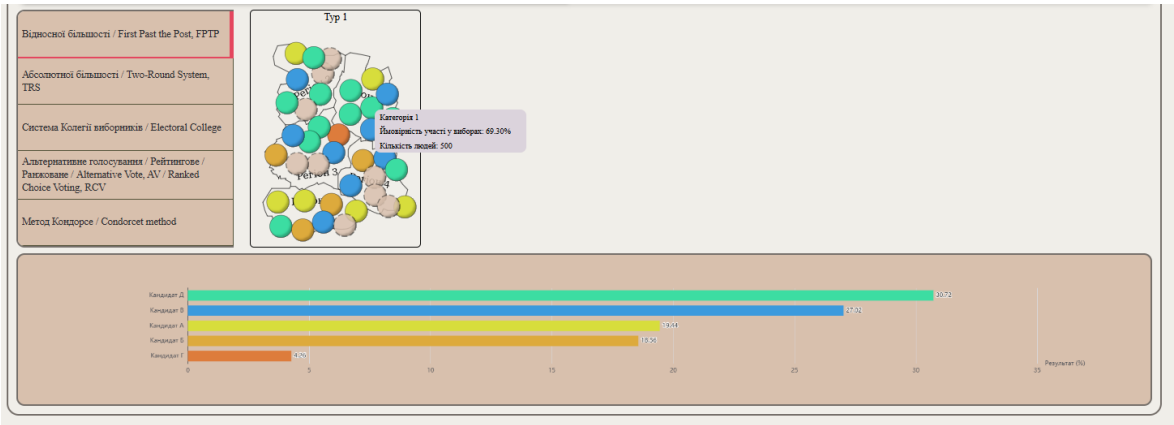


Рис. 3.25. FPTP система

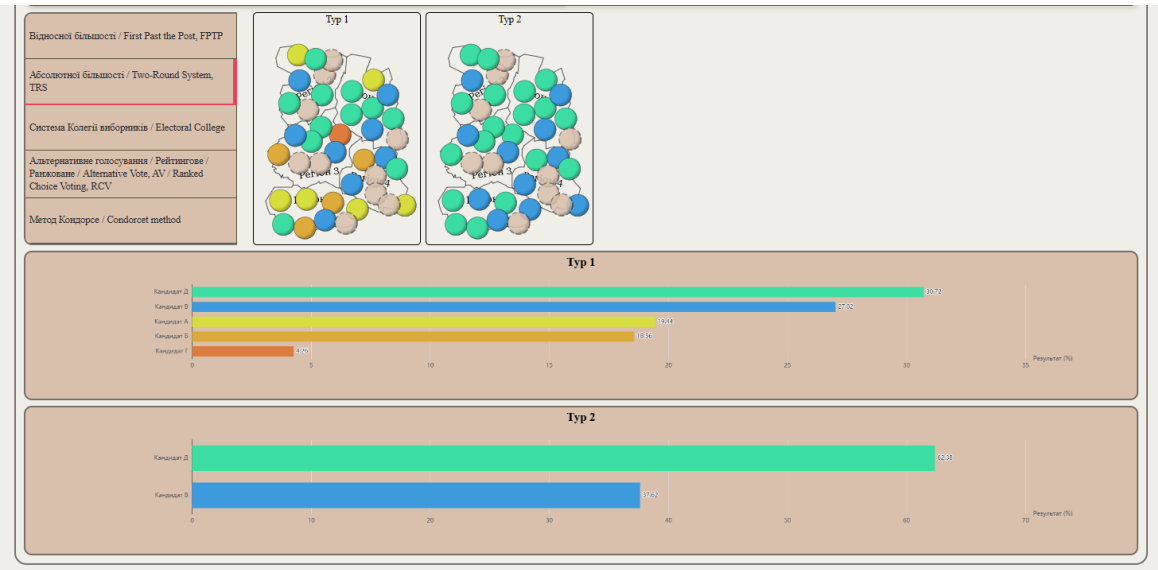


Рис. 3.26. TRS система

Для зручності в системі колегії виборників забрані регіони зафарбовуються у колір переможця.

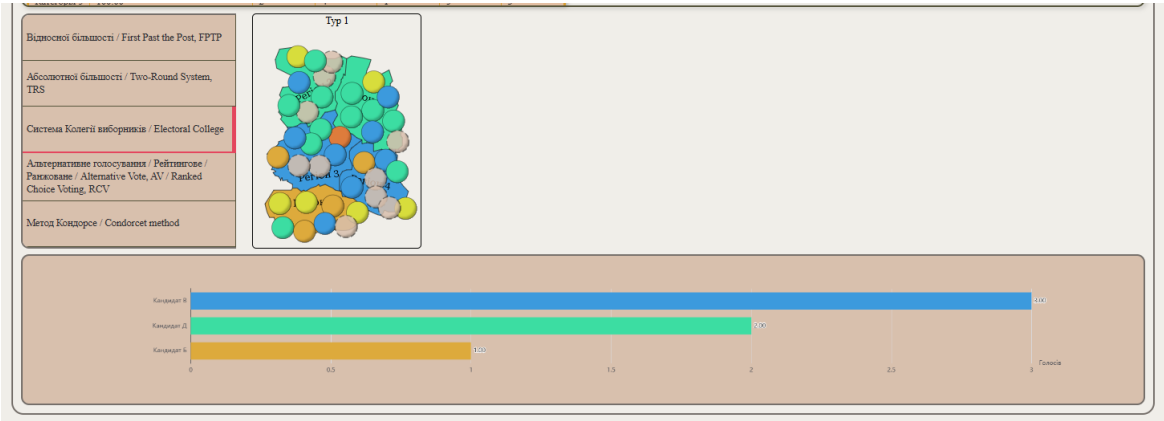


Рис. 3.27. Колегія виборців

Незалежно від виборчої системи на всіх картах виборців, що не голосували зафарбовано у нейтральний колір та позначено пунктиром.

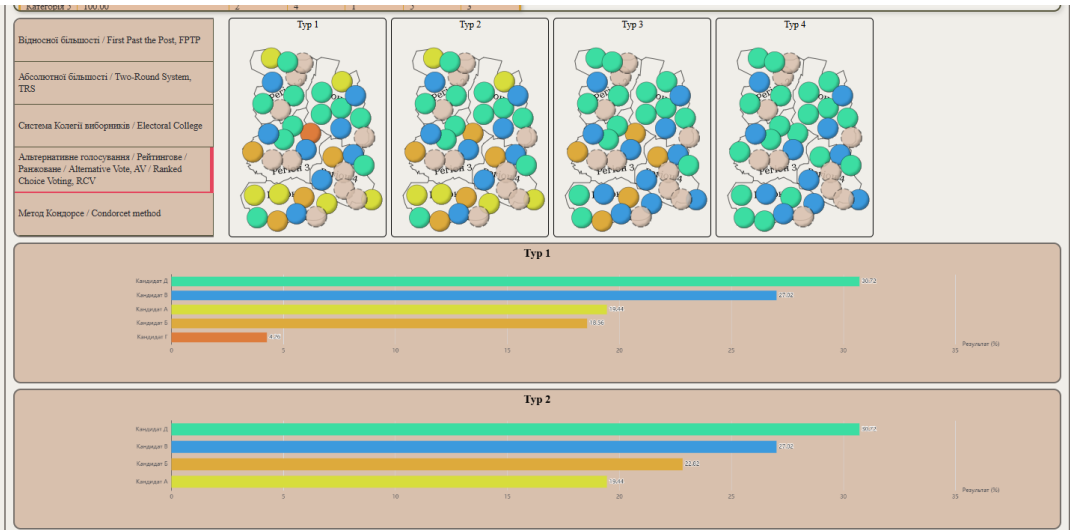


Рис. 3.28. Альтернативне голосування, частина перша

На прикладі альтернативного голосування з картами найлегше простежується як відбувається перетікання голосів.

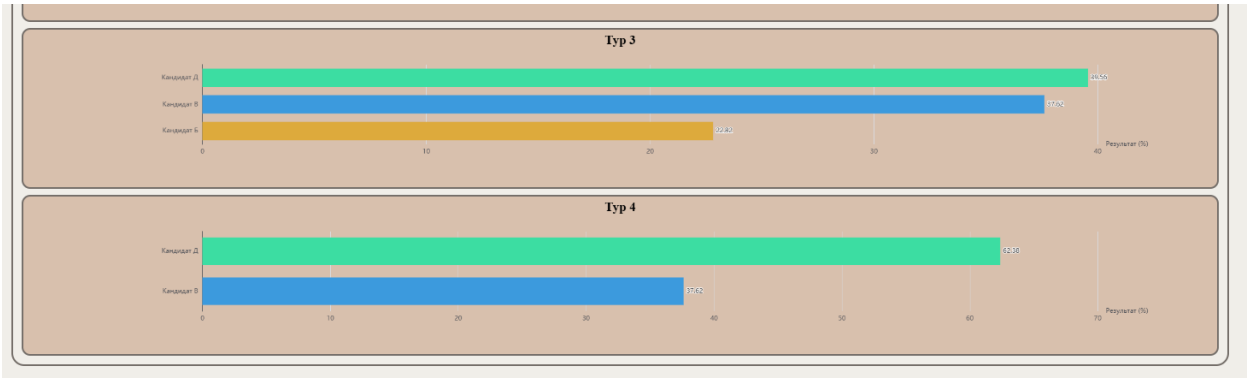


Рис. 3.29. Альтернативне голосування, частина друга

Відносної більшості / First Past the Post, FPTP					
Абсолютної більшості / Two-Round System, TRS					
Система Колегії виборників / Electoral College					
Альтернативне голосування / Рейтингове / Ранжоване / Alternative Vote, AV / Ranked Choice Voting, RCV					
Метод Кондорсе / Condorcet method					

	Кандидат А	Кандидат Б	Кандидат В	Кандидат Г	Кандидат Д
Кандидат А	Той самий кандидат	Перемога 5175 > 2800	Перемога 5820 > 2155	Перемога 7635 > 340	Програш 3705 < 4270
Кандидат Б	Програш 2800 < 5175	Той самий кандидат	Програш 3170 < 4805	Перемога 7635 > 340	Програш 1820 < 6155
Кандидат В	Програш 2155 < 5820	Перемога 4805 > 3170	Той самий кандидат	Перемога 7635 > 340	Програш 3000 < 4975
Кандидат Г	Програш 340 < 7635	Програш 340 < 7635	Програш 340 < 7635	Той самий кандидат	Програш 340 < 7635
Кандидат Д	Перемога 4270 > 3705	Перемога 6155 > 1820	Перемога 4975 > 3000	Перемога 7635 > 340	Той самий кандидат

Рис. 3.30. Система Кондорсе

Якщо аналізувати наданий приклад результатів, то в усіх системах переміг кандидат Д, крім колегії виборників, де важливо було забрати голоси регіонів з найбільшою їхньою кількістю.

3.3 Тестування програми

Тестування є невід’ємною частиною розробки. Починається воно ще на етапі написання вимог. Це відповідає принципу раннього тестування: процес пошуку дефектів має починатись якомога раніше, а саме на етапі аналізу вимог та проєктування. До прикладу, було передбачено, що в кількість людей не можна написати від’ємні числа, літери та інші символи, що не є числами. Також було передбачено, що якщо користувач залишає поле назви країни порожнім, то зберігається стара назва, а якщо поле кількості людей, то воно автоматично дорівнює нулю і вважається незаповненим.

Важливими були перевірки логічні щодо того як користувач заповнює дані: регіони не можуть лишатись порожніми; в кожному регіоні повинен зберігатись баланс між статями та різними віковими групами; перед надсиланням усі поля повинні бути заповнені; сума електорального рейтингу кандидатів повинна бути 100%; не можна дозволити від партії висувати кількість людей, що перевищує кількість мандатів у регіоні.

Наступним етапом після планування та передбачення помилок є їхня перевірка завдяки тестам. В межах роботи було написано 10 юніт-тестів для

серверу для перевірки виборчих систем (рис. 3.31) і ще 9 для перевірки клієнтської частини (рис. 3.32). Інтеграційні тести та E2E тести не проводились.

```

• (.venv) PS [redacted]\Bachelor thesis\backend> python manage.py test
Found 10 test(s).
System check identified no issues (0 silenced).
.....
-----
Ran 10 tests in 0.011s

OK

```

Рис. 3.31. Тести із серверної частини

В тестах виборчих систем задаються виборчі групи з кількістю людей та пріоритетами, а потім залежно від системи перевіряється результат першого туру, варіанти наявності та відсутності наступних турів.

В тестах для фронтенду за допомогою агіа-атрибутів перевірявся розмір екрану. Також важливими були тести на перевірку коректності кількості населення в регіонах за умови якщо є додані групи і якщо нема. Якщо кількість населення рівна нулю, то мінімальна кількість голосів/мандатів повинна бути один. Також важливе місце займає перевірка пропорційності розподілу голосів між регіонами та встановлення мінімального значення одиниці, а максимального – чотири. В цьому прикладі було використано техніки тест-дизайну покриття гілок (Branch Coverage) та покриття операторів (Statement Coverage).

```

• (.venv) PS [redacted]\Bachelor thesis\frontend> npm run test
Debugger listening on ws://127.0.0.1:57571/24f5e6e9-91ca-43a6-aca3-e4b72257f86b
For help, see: https://nodejs.org/en/docs/inspector
Debugger attached.

> frontend@0.0.0 test
> vitest

Debugger listening on ws://127.0.0.1:62603/313a2659-c968-4f39-a710-05009af31473
For help, see: https://nodejs.org/en/docs/inspector
Debugger attached.

DEV v4.1.5 [redacted]/Bachelor thesis/frontend

Debugger listening on ws://127.0.0.1:62607/e35d205b-e637-45de-9963-07628a3889b4
For help, see: https://nodejs.org/en/docs/inspector
Debugger listening on ws://127.0.0.1:62608/0f7cc561-00e4-47c8-9fe6-5a23f098c08e
For help, see: https://nodejs.org/en/docs/inspector
Debugger attached.
Debugger attached.
✓ src/__tests__/RegionSeats.test.ts (5 tests) 7ms
✓ src/__tests__/ScreenSize.test.tsx (2 tests) 128ms

Test Files  2 passed (2)
Tests      7 passed (7)
Start at   05:12:24
Duration  10.79s (transform 1.74s, setup 1.06s, import 5.95s, tests 136ms, environment 7.58s)

```

Рис. 3.32. Тести клієнтської частини

В подальшому важливо періодично змінювати тести відповідно до принципу парадоксу пестицидів, коли при повторі одних і тих самих тестів система пристосовується до них, а самі тести втрачають свою ефективність.

Під час мануального тестування було виявлено баг, пов'язаний з видаленням кандидатів та результатами. Результат описано у таблиці 3.1.

Таблиця 3.1

Баг-репорт

ID/ім'я	Видалення кандидатів, що впливає на відображення результатів	
Опис/резюме (Summary)	Після видалення кандидата у таблиці результатів зникає кандидат, на графіках показується його ідентифікатор замість імені, зникає колір групи на карті	
Вплив (Impact)	Серйозність (Severity)	Висока (Major)
	Пріоритет (Priority)	Високий (High)
Детальний опис багу	Середовище	Chrome Версія 149.0.7827.197 (Офіційне складання) (64-розрядна версія)
	Вихідна інформація	Карта з результатами
	Візуальний доказ	Рис. 3.33
	Дії для відтворення	1. Обрахувати результати 2. Видалити кандидата 3. Таблиця та інші елементи результатів будуть змінені
	Очікувані результати (Expected results)	Результати лишаються як були, включаючи видаленого кандидата
	Фактичні результати (Actual results)	Після видалення вплив на відображення результатів
Додаткова інформація (Additions)	Відсутня	

Також було виявлено, що повзунки для вимірювання медіа та рейтингів у браузері Mozilla Firefox не є стилізованим відповідно до системи. Рівень серйозності та пріоритет незначний.

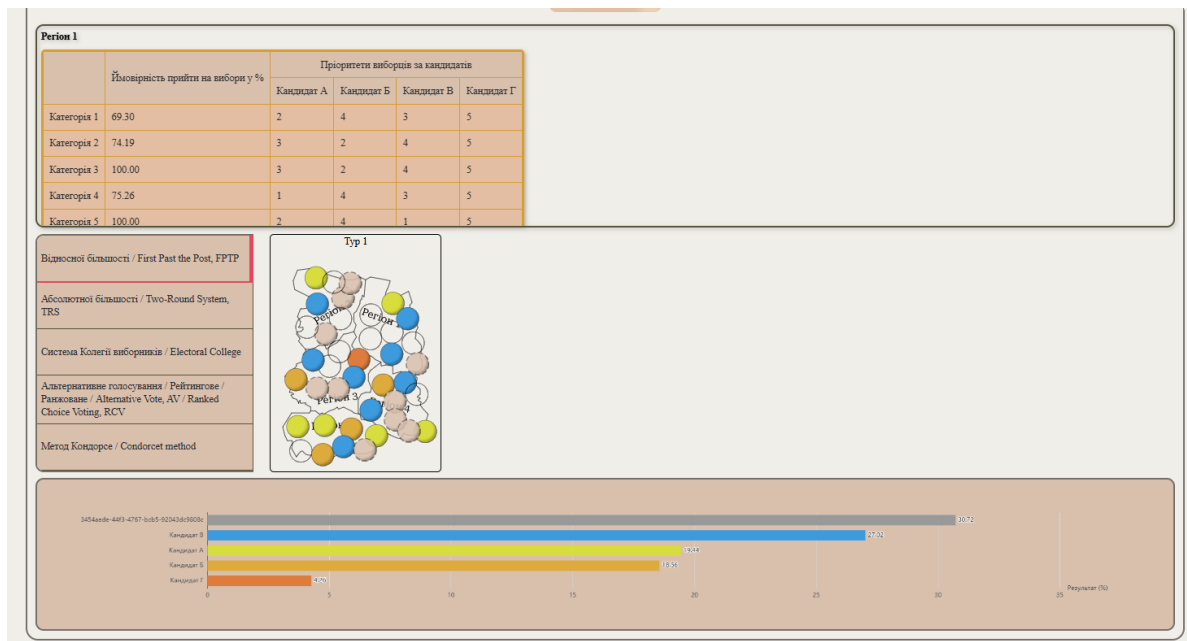


Рис. 3.33. Баг пов'язаний з видаленням кандидатів

Саме на цих прикладах найкраще видно перший принцип тестування: тести показують наявність дефектів, але не гарантують їхню відсутність. Навіть при ретельних перевірках лишається ймовірність, що будуть приховані баги.

Варто також зазначити, що під час мануального тестування використовувались описані у другому розділі варіанти використання та діаграми станів.

Нефункціональне тестування в межах роботи не проводилось. Успішність збірки, тестування та контейнеризації наведено на рис. 3.34.

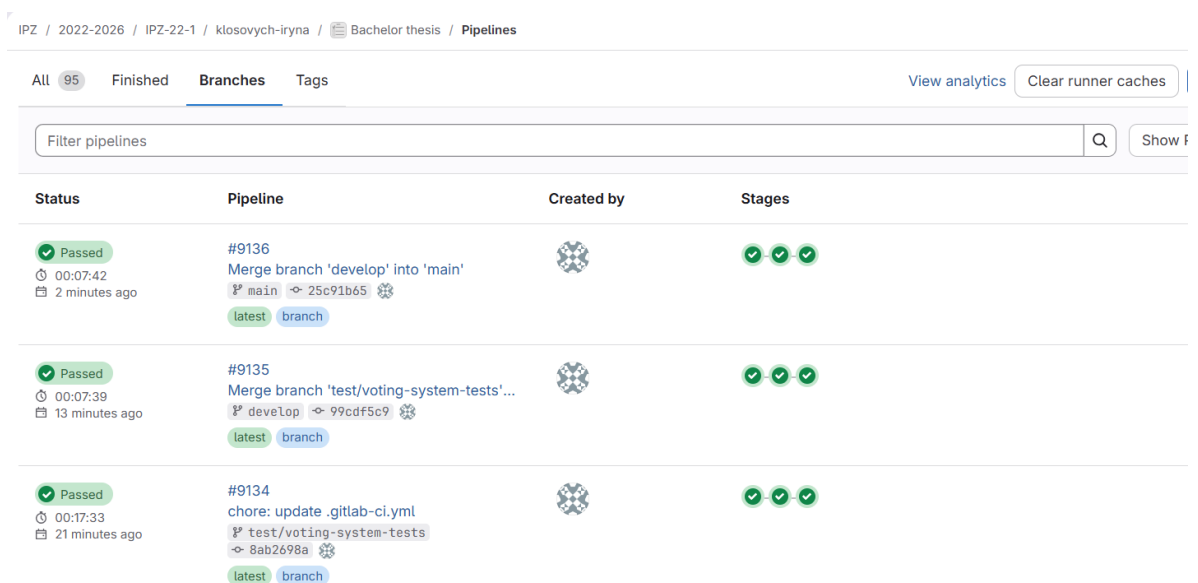


Рис. 3.34. Успішне проходження усіх стадій

Висновки до третього розділу

У третьому розділі було продемонстровано налаштування системи, описано як взаємодіяти з інтерфейсом та тестування розробки.

В першому підпункті розглянуто різні методології Git ведення проєкту, детально розказано про розгортання за стосунку на сторонніх сервісах, наведено відповідну діаграму. Для серверної частини обрано Render, а для клієнтської – Vercel.

У другому підпункті досліджено взаємодію із розробленим продуктом, описано етапи від створення країни і заповнення до отримання результатів. Особливу увагу звернено на взаємодію з картою та результати обчислень.

Третя частина орієнтовна на тестування продукту. Відповідно було описано передбачені помилки заздалегідь під час планування. Вказано що було розроблено 19 автоматичних юніт-тестів, які успішно пройдено. Під час тестування виявлено баг, який описано в баг-репорті.

ВИСНОВКИ

В рамках випускної кваліфікаційної роботи бакалавра було досліджено тематику політико-соціальних процесів, а саме виборчого та розроблено симулятор для аналізу і прогнозу результатів виборів. Головною метою було створення симулятора для прогнозування результатів виборів на основі демографічних та внутрішньоособистісних рис представників електорату, з врахуванням зовнішніх чинників, як-от стан країни, а також виборчої системи.

На початку було сформовано завдання, що полягали в аналізі аналогів системи та дослідження тематики; визначення архітектури системи; проєктування та розробка алгоритмів системи; реалізація функціоналу; тестування системи.

В першому розділі проаналізовано п'ять аналогів розроблюваної системи, з яких виокремлено найкращі рішення для реалізації та доповнено своїми ідеями та баченням. У цьому ж розділі прийнято рішення про розробку системи з гібридною архітектурою, проаналізовано переваги відповідно до життєвого циклу проєкту. Після цього було перейдено до обрання відповідних технологій, що допомогли реалізувати поставлені задачі ефективно. Для серверної частини обрано мову Python і фреймворк Django, для клієнтської частини – мову TypeScript та бібліотеку React.

У другому розділі розглянуто різні шляхи реалізації самої ідеї, обрано оптимальний до наявних ресурсів та задач. Наступним кроком було знайдено одного актора та визначено варіанти використання системи, далі описано структуру взаємодії основних елементів. Також охарактеризовано основний алгоритм системи та послідовність взаємодії її модулів. Визначено, що перша частина завдання є регресійною задачею, а друга – рекомендаційною. Особливу увагу приділено формуванню набору даних з 20 ознак виборців та 5 ознак країни, щоб отримати ймовірність приходу на вибори та 8 ознак ідеального кандидата. Важливо, щоб ознаки були наближені до реальних, тому здійснено спробу сформувати нелінійний набір, який використовувався для тренування моделі. Для моделі проаналізовано чотири різні варіанти, найкращим виявився

алгоритм HistGradientBoostingRegressor у поєднанні з RegressorChain за більшістю з метрик.

У третьому розділі в першому підпункті описано вибір методології ведення проєкту з використанням Git, продемонстровано діаграму розгортання та інші елементи конфігурацій. Особливу увагу приділено демонстрації інтерфейсу розробленої програми та її роботи. Останнім кроком охарактеризовано як відбувалось тестування та знайдені недоліки.

У підсумку всі задачі, поставлені у кваліфікаційній роботі, було виконано.

		Клосович І.А.			ІПЗ.КР.Б –121 – 26 – ПЗ	
		Савіцький Р.С.				
						77

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Алієв Г.Є. Політологія: Навч. посібник для студентів вищих навчальних закладів. – Полтава: АСМІ, 2007. – 280 с.
2. Вонсович С.Г. Політичний аналіз і прогноз: навчально-методичний посібник [Електронний ресурс]. Кам'янець-Подільський: Кам'янець-Подільський національний університет імені Івана Огієнка, 2023. 228 с. URL: <http://elar.kpnu.edu.ua:8081/xmlui/handle/123456789/7048> (дата звернення: 05.01.2026)
3. Клосович І.А., Савіцький Р.С., Толстой І.А. Моделювання та порівняльний аналіз виборчих систем у симуляторі прогнозу результатів виборів. Тези XIX Міжнародної науково-практичної конференції «Перспективи розвитку сучасної науки та освіти», 29-30 червня 2026 року. Львів: «Львівський науковий форум», 2026. [Подано до друку]
4. Клосович І.А., Савіцький Р.С., Толстой І.А. Застосування регресійного машинного навчання для прогнозування електоральної поведінки. Тези XIX Міжнародної науково-практичної конференції «Перспективи розвитку сучасної науки та освіти», 29-30 червня 2026 року. Львів: «Львівський науковий форум», 2026. [Подано до друку]
5. Гра GerryMander [Електронний ресурс]. URL: <http://gametheorytest.com/gerry/game/> (дата звернення: 10.01.2026)
6. Визначення терміну джеррімендерингу [Електронний ресурс]. URL: <https://dictionary.cambridge.org/uk/dictionary/english/gerrymander> (дата звернення: 10.01.2026)
7. Гра Election Lab [Електронний ресурс]. URL: <https://election-lab-frontend.vercel.app/> (дата звернення: 10.01.2026)
8. Стаття How ranked-choice voting could change the way democracy works новинного сайту The Washington Post [Електронний ресурс]. URL: <https://www.washingtonpost.com/politics/interactive/2021/ranked-choice-voting-guide/> (дата звернення: 10.01.2026)

9. Сайт із симуляціями про виборчі системи smartvotesim.com [Електронний ресурс]. URL: <https://smartvotesim.com/> (дата звернення: 10.01.2026)
10. Сайт із симуляцією perthirtysix.com [Електронний ресурс]. URL: <https://perthirtysix.com/tool/voting-systems-simulator> (дата звернення: 10.01.2026)
11. Robert C. Martin. Clean Architecture: A Craftsman's Guide to Software Structure and Design. – USA: Prentice Hall, 2017.
12. Architectural Styles in Software Engineering [Електронний ресурс]. URL: <https://shapingsoftware.com/architectural-styles/> (дата звернення: 15.01.2026)
13. Enterprise software architecture patterns: The complete guide [Електронний ресурс]. URL: <https://vfunction.com/blog/enterprise-software-architecture-patterns/> (дата звернення: 15.01.2026)
14. Software Development Life Cycle (SDLC) [Електронний ресурс]. URL: <https://www.geeksforgeeks.org/software-engineering/software-development-life-cycle-sdlc/> (дата звернення: 15.01.2026)
15. Top 10 Software Architecture Styles You Should Know [Електронний ресурс]. URL: <https://bool.dev/blog/detail/top10-software-architecture-styles> (дата звернення: 16.01.2026)
16. Повний огляд REST: нюанси, поради, приклади [Електронний ресурс]. URL: <https://dou.ua/forums/topic/50364/> (дата звернення: 16.01.2026)
17. Figma [Електронний ресурс]. URL: <https://www.figma.com/> (дата звернення 18.01.2026)
18. Visual Studio Code [Електронний ресурс]. URL: <https://code.visualstudio.com/docs/getstarted/overview> (дата звернення 20.01.2026)
19. PlantUML [Електронний ресурс]. URL: <https://plantuml.com/> (дата звернення: 20.01.2026)
20. Mermaid [Електронний ресурс]. URL: <https://mermaid.js.org/intro/> (дата звернення: 20.01.2026)

21. Git [Електронний ресурс]. URL: <https://git-scm.com/> (дата звернення: 21.01.2026)
22. GitLab [Електронний ресурс]. URL: <https://docs.gitlab.com/> (дата звернення: 21.01.2026)
23. GitHub [Електронний ресурс]. URL: <https://docs.github.com/en> (дата звернення: 26.06.2026)
24. Vite [Електронний ресурс]. URL: <https://vite.dev/guide/> (дата звернення: 20.01.2026)
25. Порівнюємо React, Angular і Vue – найпопулярніші бібліотеки й фреймворки у 2022 році [Електронний ресурс]. URL: <https://dou.ua/forums/topic/39933/> (дата звернення: 02.02.2026)
26. Як правильно вибрати стек для свого проєкту? [Електронний ресурс]. URL: <https://wezom.com.ua/ua/blog/tehnologicheskij-stek-proekta> (дата звернення: 02.02.2026)
27. Ідеальний стейт-менеджер для React у 2024 році [Електронний ресурс]. URL: <https://dou.ua/forums/topic/48235/> (дата звернення: 10.02.2026)
28. FastAPI [Електронний ресурс]. URL: <https://fastapi.tiangolo.com/> (дата звернення: 11.02.2026)
29. Django [Електронний ресурс]. URL: <https://docs.djangoproject.com/en/5.2/> (дата звернення: 11.02.2026)
30. NumPy [Електронний ресурс]. URL: <https://numpy.org/> (дата звернення: 15.02.2026)
31. Scikit-learn [Електронний ресурс]. URL: <http://scikit-learn.org/stable/> (дата звернення: 15.02.2026)
32. Ruff [Електронний ресурс]. URL: <https://docs.astral.sh/ruff/> (дата звернення: 16.02.2026)
33. ESLint [Електронний ресурс]. URL: <https://www.npmjs.com/package/eslint> (дата звернення: 16.02.2026)
34. Docker Desktop [Електронний ресурс]. URL: <https://www.docker.com/products/docker-desktop/> (дата звернення: 05.05.2026)

35. OrdinalEncoder [Електронний ресурс]. URL: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OrdinalEncoder.html> (дата звернення: 12.05.2026)
36. OneHotEncoder [Електронний ресурс]. URL: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html> (дата звернення: 12.05.2026)
37. RandomForestRegressor [Електронний ресурс]. URL: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html> (дата звернення: 12.05.2026)
38. HistGradientBoostingRegressor [Електронний ресурс]. URL: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.HistGradientBoostingRegressor.html> (дата звернення: 12.05.2026)
39. MultiOutputRegressor [Електронний ресурс]. URL: <https://scikit-learn.org/stable/modules/generated/sklearn.multioutput.MultiOutputRegressor.html> (дата звернення: 13.05.2026)
40. RegressorChain [Електронний ресурс]. URL: <https://scikit-learn.org/stable/modules/generated/sklearn.multioutput.RegressorChain.html> (дата звернення: 13.05.2026)
41. 6 Types of Git Branching Strategy for DevOps [Електронний ресурс]. URL: <https://dev.to/juniourrau/6-types-of-git-branching-strategy-g54> (дата звернення: 10.02.2026)
42. How to Write Better Git Commit Messages – A Step-By-Step Guide [Електронний ресурс]. URL: <https://www.freecodecamp.org/news/how-to-write-better-git-commit-messages/> (дата звернення: 10.02.2026)
43. Render [Електронний ресурс]. URL: <https://render.com/> (дата звернення: 24.06.2026)
44. Vercel [Електронний ресурс]. URL: <https://vercel.com/> (дата звернення: 24.06.2026)

ДОДАТКИ

		Клосович І.А.			ІПЗ.КР.Б –121 – 26 – ПЗ	
		Савіцький Р.С.				
						82

Глосарій

Виборець – представник електорату; той, хто бере участь у виборах, голосуючи за певного кандидата. У системі задаються різні характеристики виборця.

Вибори – це загальноприйнятий демократичний механізм, в процесі якого відбувається формування представницьких органів державної влади та місцевого самоврядування. В системі передбачено президентські та парламентські вибори.

Виборча система – сукупність установлених законом правил, принципів і критеріїв, за допомогою яких визначаються результати виборів представницьких органів. Розрізняють мажоритарну, пропорційну та змішані види.

Кандидат – людина, що висувається на виборну посаду. В системі це або одна людина для президентських виборів, або партія для парламентських зі своїми представниками.

Країна – територія, що становить єдність із погляду історії, населення тощо. Характеризується формою правління, внутрішнім та зовнішнім станом тощо. В системі візуально представлена картою, до якої прикріплено виборців, та іншою інформацією, яку заповнює користувач у відповідних блоках біля карти.

Медіа-вплив – інформаційний вплив медіа на виборців. У системі характеризує стан країни та кандидатів.

Обробка виборчих даних – алгоритмічне опрацювання введених характеристик для обрахунку результатів виборів.

Результат виборів – підсумок голосування, який у системі представлено у графічній та за потреби текстовій формі.

Характеристики виборця – параметри, що описують виборця та мають вплив на його вибір. Включає демографічні (вік, стать, фінансове

становище тощо) та внутрішньоособистісні (рівень розуміння тем, зацікавленості тощо) дані.

Пріоритезація історій користувачів

№	Історія користувача	Must/Should/ Could/Won't have
1	2	3
1	Я, як користувач, хочу бачити повідомлення-попередження при вході з пристрою з маленьким екраном, щоб розуміти, що працювати буде незручно, і мати змогу перейти на інший пристрій для комфортної роботи з великою кількістю даних.	Must
2	Я, як користувач, хочу створювати порожній шаблон країни, щоб вносити власні дані.	Must
3	Я, як користувач, хочу копіювати країну, щоб не вводити однакові дані кілька разів.	Must
4	Я, як користувач, хочу видаляти країну, щоб непотрібні шаблони не заважали подальшій роботі.	Must
5	Я, як користувач, хочу задавати назву країні, щоб розрізняти створені варіанти.	Should
6	Я, як користувач, хочу бачити карту країни з виборчими групами, щоб візуально оцінювати рівень небезпеки територій і розподіл груп.	Must
7	Я, як користувач, хочу додавати виборчі групи на карту, щоб підвищити точність аналізу.	Must
8	Я, як користувач, хочу змінювати рівень небезпеки територій, щоб краще моделювати ситуацію в країні.	Must
9	Я, як користувач, хочу видаляти виборчі групи, щоб прибрати зайві елементи.	Must
10	Я, як користувач, хочу бачити поруч із картою таблицю регіонів країни з усіма виборчими групами, щоб мати змогу вводити необхідні дані для аналізу.	Must
11	Я, як користувач, хочу переміщувати виборчі групи між регіонами, щоб моделювати різні сценарії.	Must

1	2	3
12	Я, як користувач, хочу бачити зміни в таблиці при зміні карти, щоб забезпечити актуальність і узгодженість даних.	Must
13	Я, як користувач, хочу бачити статево-вікові піраміди по країні та регіонах, щоб аналізувати співвідношення груп і реалістичність даних.	Must
14	Я, як користувач, хочу обирати тип виборів залежно від органу, що обирається, щоб аналізувати відповідні виборчі системи.	Must
15	Я, як користувач, хочу додавати опис країни та поточної ситуації, щоб підвищити точність прогнозування.	Must
16	Я, як користувач, хочу додавати кандидатів до країни, щоб моделювати процес виборів.	Must
17	Я, як користувач, хочу змінювати характеристики кандидатів, щоб забезпечити різноманітність і кращу наочність виборчого процесу.	Must
18	Я, як користувач, хочу видаляти кандидатів, щоб вони не впливали на результати симуляції.	Must
19	Я, як користувач, хочу після налаштування мати кнопку для надсилання даних на обрахунок.	Must
20	Я, як користувач, після отримання результатів хочу бачити панель вибору виборчої системи, щоб переглядати результати для кожної з них.	Must
21	Я, як користувач, у випадку президентських виборів хочу бачити покрокове (турове) відображення результатів, щоб краще розуміти процес.	Must
22	Я, як користувач, у випадку парламентських виборів хочу бачити розподіл місць у парламенті у відсотках, щоб результати були наочними.	Must
23	Я, як користувач, хочу бачити різні кольори виборчих груп при налаштуванні, щоб бачити рівень заповнення інформації	Should
24	Я, як користувач, хочу бачити кількість країн, що є, щоб орієнтуватись в кількості створених макетів	Should
25	Я, як користувач, хочу переключатись між різними країнами, щоб робота була ефективнішою	Should

```

def _predict_candidate_voter_similarity(
    voter: CalculationVotingGroup, candidate: CandidateScores
):
    voter_vector = normalize_axes(voter.get_ideal_vector())
    cand_vector = normalize_axes(candidate.as_vector())
    v1 = voter_vector.reshape(1, -1)
    v2 = cand_vector.reshape(1, -1)

    score = cosine_similarity(v1, v2)[0][0]
    return score

def _predict_candidates_for_voter(voter, candidates):
    voter_result = {}
    for candidateId, candidate in candidates.items():
        score = _predict_candidate_voter_similarity(voter, candidate)
        voter_result[candidateId] = {"score": score}
    voter_result = add_priorities(voter_result)
    return voter_result

def predict_president_similarities(
    voters_by_regions: dict[UUID, list[CalculationVotingGroup]],
    candidates: CandidateScoresMap,
) -> dict[UUID, list]:
    for regionId, voters in voters_by_regions.items():
        for voter in voters:
            voter_result = _predict_candidates_for_voter(voter, candidates)
            voter.set_president_candidates_rank(voter_result)
    return voters_by_regions

def add_priorities(scored: dict[UUID, dict]) -> dict[UUID, CandidatesRank]:
    items = [(cand_id, data["score"]) for cand_id, data in scored.items()]

    items = sorted(items, key=lambda x: x[1], reverse=True)

    return {
        cand_id: CandidatesRank.model_validate({
            "score": score,
            "priority": rank + 1
        })
        for rank, (cand_id, score) in enumerate(items)
    }

```

```

from pathlib import Path

import generation_data as gn
import numpy as np

Path("data").mkdir(exist_ok=True)
rng = np.random.default_rng(42)

# — Voter generator

```

```

def generate_voter_base(index: int) -> dict:
    age = rng.choice(gn.AGE_GROUPS, p=[0.20, 0.30, 0.28, 0.22])
    sex = rng.choice(gn.SEX, p=[0.54, 0.46])

    edu = rng.choice(gn.EDUCATIONS, p=[0.20, 0.35, 0.15, 0.30])
    econ_st = rng.choice(gn.ECON_STATUS, p=[0.22, 0.18, 0.16, 0.30, 0.10,
0.04])
    fin_ind = rng.choice(gn.FIN_INDEPENDENT, p=[0.15, 0.20, 0.08, 0.19,
0.38])

    age_idx = gn.AGE_GROUPS.index(age)
    edu_idx = gn.EDUCATIONS.index(edu)

    und_econ = rng.choice(gn.UNDERSTANDING, p=[0.10, 0.25, 0.30, 0.25,
0.10])
    und_safety = rng.choice(gn.UNDERSTANDING, p=[0.08, 0.22, 0.32, 0.28,
0.10])
    und_social = rng.choice(gn.UNDERSTANDING, p=[0.10, 0.25, 0.30, 0.25,
0.10])
    und_map = {und: index for index, und in enumerate(gn.UNDERSTANDING)}

    return {
        "voter_id": index,
        "age": age,
        "sex": sex,
        "nationality": rng.choice(gn.NATIONALITIES, p=[0.80, 0.20]),
        "identity": rng.choice(gn.IDENTITIES, p=[0.35, 0.45, 0.20]),
        "religion": int(np.clip(rng.normal(4, 2.5), 0, 10)),
        "education": edu,
        "economic_status": econ_st,
        "finance_independent": fin_ind,
        "interest_econ": int(np.clip(rng.normal(6 - age_idx * 0.4, 2), 0,
10)),
        "interest_safety": int(np.clip(rng.normal(5 + age_idx * 0.7, 2), 0,
10)),
        "interest_social": int(
            np.clip(rng.normal(5 + (1 if sex == "Жіноча" else 0), 2), 0, 10)
        ),
        "understanding_econ": und_econ,
        "understanding_safety": und_safety,
        "understanding_social": und_social,
        "und_econ_num": und_map[und_econ],
        "und_safety_num": und_map[und_safety],
        "und_social_num": und_map[und_social],
        "safety_region": int(np.clip(rng.normal(3.0, 1.2), 1, 5)),
    }

```

```

        "political_interest": int(np.clip(rng.normal(5 + edu_idx * 0.4, 2.5),
0, 10)),
        "media_positive_reaction": int(
            np.clip(rng.normal(5 - edu_idx * 0.4, 2), 0, 10)
        ),
        "media_negative_reaction": int(
            np.clip(rng.normal(6 - edu_idx * 0.3, 2), 0, 10)
        ),
        "rating_perception": int(np.clip(rng.normal(5, 2), 0, 10)),
        "experience_importance": int(np.clip(rng.normal(6, 2), 0, 10)),
    }

```

— Categories

```

AGE_GROUPS = ["18-30", "31-45", "45-60", "60+"]
SEX = ["Чоловіча", "Жіноча"]
NATIONALITIES = ["Титульна нація", "Національна меншина"]
IDENTITIES = [
    "Однакова громадянська та етнічна",
    "Громадянська сильніша за етнічну",
    "Сильна етнічна ідентичність",
]

EDUCATIONS = [
    "9-11 класи", # 0.20
    "Профтех/коледж", # 0.35
    "Незакінчена вища", # 0.15
    "Вища освіта", # 0.30
]

ECON_STATUS = [
    "Низький", # 0.22
    "Нижче середнього", # 0.18
    "Середній", # 0.16
    "Вище середнього", # 0.30
    "Високий", # 0.10
    "Заможний", # 0.04
]

FIN_INDEPENDENT = [
    "Повністю залежний", # 0.15
    "Переважно залежний", # 0.20
    "Частково і так, і так", # 0.08
    "Переважно самостійний", # 0.19
    "Повністю самостійний", # 0.38
]

UNDERSTANDING = [
    "Не орієнтується у темі та не може пояснити базові поняття",
    "Розуміє загальні ідеї, але складно пояснювати або застосовувати їх",
    "Може пояснити основні поняття іншій людині та розуміє типові приклади",
    "Впевнено застосовує знання на практиці та може аналізувати ситуації",
    "Може критично оцінювати інформацію, знаходити помилки та аргументувати позицію",
]

```

— Country snapshots used during training

```

COUNTRY_SNAPSHOTS = [

```

```

# wartime
{
    "war_status": 8,
    "economic_crisis": 6,
    "corruption_level": 7,
    "media_freedom": 4,
    "political_stability": 3,
},
# post-war recovery
{
    "war_status": 2,
    "economic_crisis": 7,
    "corruption_level": 6,
    "media_freedom": 5,
    "political_stability": 4,
},
# peacetime stable
{
    "war_status": 0,
    "economic_crisis": 2,
    "corruption_level": 4,
    "media_freedom": 7,
    "political_stability": 7,
},
# crisis authoritarian
{
    "war_status": 5,
    "economic_crisis": 8,
    "corruption_level": 9,
    "media_freedom": 2,
    "political_stability": 2,
},
]

COUNTRY_FIELDS = [
    "war_status",
    "economic_crisis",
    "corruption_level",
    "media_freedom",
    "political_stability",
]

TARGET_COLS = [
    "probability_take_part",
    "candidate_positive_importance",
    "candidate_negative_fair_importance",
    "like_easy_decision",
    "like_strong_leader_over_law",
    "understand_government_institutions",
    "every_person_is_expert",
    "person_or_government_importance",
    "believe_government_institutions",
]

"""
GENERATE TRAINING DATA (with country state)
=====
KEY CHANGE: country state fields are appended to every voter row.

```

This means the model learns HOW country context shifts voter preferences.

WHY BAKE COUNTRY INTO VOTER ROWS (not pass separately)?

The model maps: voter_profile + country_state → ideal_vector
Different country states produce different ideal vectors for the same voter.

Example:

voter A in peace (war_status=1) → ideal_security = 4.2
voter A in war (war_status=9) → ideal_security = 8.7
same voter, different country context, very different priorities.

COUNTRY FIELDS (all 0-10):

war_status	0=peace	10=active war
economic_crisis	0=stable	10=severe crisis
corruption_level	0=clean	10=highly corrupt
media_freedom	0=censored	10=free press
political_stability	0=chaos	10=stable

TRAINING STRATEGY:

Generating voters under MULTIPLE country states so the model learns the relationship, not just one scenario.
Each voter is cloned across several country snapshots.

OUTPUT: data/non-linear-voters.csv

"""

```
from pathlib import Path
```

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from generate_voter_base import generate_voter_base
from generation_data import (
    COUNTRY_FIELDS,
    COUNTRY_SNAPSHOTS,
    ECON_STATUS,
    EDUCATIONS,
    TARGET_COLS,
)
```

```
import constants
```

```
Path("data").mkdir(exist_ok=True)
rng = np.random.default_rng(42)
```

```
def sigmoid(x, midpoint=5, steepness=1.0):
    """
    It is ideal for applications where probability estimation is needed
    The non-linear nature
    Saturation effects
    """
    return 1 / (1 + np.exp(-steepness * (x - midpoint)))

def interaction(a, b, scale=10):
    """
```

```

    Multiplicative interaction: two factors amplify each other.
    """
    return (a * b) / scale

def diminishing(x, scale=3.0):
    """
    Diminishing returns via square root.
    First units of x matter a lot, later units matter less.
    """
    return scale * np.sqrt(max(0, x))

#
=====
# NON-LINEAR TARGET DERIVATION
#
=====
def derive_targets(voter: dict, country: dict) -> dict:
    edu_idx = (
        EDUCATIONS.index(voter["education"]) if voter["education"] in
EDUCATIONS else 1
    )
    eco_idx = (
        ECON_STATUS.index(voter["economic_status"])
        if voter["economic_status"] in ECON_STATUS
        else 2
    )
    und_avg = (
        voter["und_econ_num"] + voter["und_safety_num"] +
voter["und_social_num"]
    ) / 3

    war = country["war_status"]
    econ = country["economic_crisis"]
    corr = country["corruption_level"]
    media = country["media_freedom"]
    stab = country["political_stability"]

    # — 1. understand_government_institutions (0-10)
    edu_effect = diminishing(edu_idx, scale=2.0)
    und_effect = diminishing(und_avg, scale=1.5)
    gov_und = np.clip(
        edu_effect
        + und_effect
        + voter["political_interest"] * 0.35
        + rng.normal(0, 1.5),
        0,
        10,
    )
    understand_government_institutions = int(gov_und)

    # — 2. believe_government_institutions (0-10)
    trust_collapse = (1 - sigmoid(corr, midpoint=5.5, steepness=1.5)) * 6
    trust = np.clip(
        trust_collapse

```

```

    + stab * 0.35
    + media * 0.12
    + gov_und * 0.10
    + rng.normal(0, 1.5),
    0,
    10,
)
believe_government_institutions = int(trust)

# — 3. every_person_is_expert (1-5)


---


    institutional_collapse = interaction(corr, 10 - stab, scale=10)
    populism_spike = sigmoid(institutional_collapse, midpoint=5,
steepness=2.5) * 2.5
    pop = np.clip(
        1.0
        + populism_spike
        + diminishing(10 - gov_und, scale=0.2)
        + rng.normal(0, 0.5),
        1,
        5,
    )
    every_person_is_expert = int(round(pop))

# — 4. probability_take_part (0-100)


---


    interest_effect = (
        sigmoid(voter["political_interest"], midpoint=3.5, steepness=2.0) *
72
    )
    urgency = interaction(war, econ, scale=4) * 1.5
    disillusion_raw = interaction(corr, 10 - stab, scale=10)
    disillusion = sigmoid(disillusion_raw, midpoint=4, steepness=1.5) * 35
    trust_bonus = trust * 3.2
    edu_eco_effect = diminishing(edu_idx, scale=3) + diminishing(eco_idx,
scale=2)

    prob = np.clip(
        interest_effect
        + urgency
        + trust_bonus
        + edu_eco_effect
        - disillusion
        + rng.normal(0, 6),
        0,
        100,
    )
    probability_take_part = round(float(prob), 1)

# — 5. candidate_positive_importance (0-10)


---


    censorship_effect = sigmoid(10 - media, midpoint=5, steepness=1.5) * 1.5
    pos_imp = np.clip(
        voter["media_positive_reaction"] * 0.45
        + diminishing(10 - und_avg, scale=0.7)
        + censorship_effect
        + interaction(war, 10 - media, scale=10) * 0.3
        + rng.normal(0, 1.3),
        0,
        10,

```

```

)
candidate_positive_importance = int(pos_imp)

# — 6. candidate_negative_fair_importance (0-10)


---


need_truth = sigmoid(corr, midpoint=5, steepness=1.2) * 0.8
access_truth = sigmoid(media, midpoint=4, steepness=1.0) * 0.8
truth_effect = need_truth * access_truth * 5

hopelessness = interaction(corr, 10 - media, scale=10)
hope_penalty = sigmoid(hopelessness, midpoint=7, steepness=2.0) * 2.5

neg_imp = np.clip(
    6.0
    + voter["media_negative_reaction"] * 0.20
    + diminishing(und_avg, scale=0.5)
    + truth_effect
    - hope_penalty
    + rng.normal(0, 1.8),
    0,
    10,
)
candidate_negative_fair_importance = int(neg_imp)

# — 7. like_easy_decision (0-10)


---


simplicity_pull = sigmoid(10 - und_avg, midpoint=5, steepness=1.5) * 3.5
instability_effect = interaction(war, 10 - stab, scale=10) * 0.4
edu_effect_easy = diminishing(3 - edu_idx, scale=0.6) if edu_idx < 3
else 0
media_neg_effect = voter["media_negative_reaction"] * 0.15

easy = np.clip(
    1.5
    + simplicity_pull
    + instability_effect
    + edu_effect_easy
    - media_neg_effect
    + gov_und * 0.08
    + rng.normal(0, 1.5),
    0,
    10,
)
like_easy_decision = int(easy)

# — 8. like_strong_leader_over_law (1-5)


---


war_instability = interaction(war, 10 - stab, scale=10)
auth_spike = sigmoid(war_instability, midpoint=4, steepness=1.2) * 1.8

understanding_pull = (1 - und_avg / 4) * 1.2

auth = np.clip(
    1.5 # base
    + auth_spike
    + understanding_pull
    + corr * 0.06
    + rng.normal(0, 0.9),
    1,

```

```

        5,
    )
    like_strong_leader_over_law = int(round(auth))

# — 9. person_or_government_importance (0-10)


---


    state_focus = sigmoid(war + econ, midpoint=8, steepness=1.0) * 3.0
    anti_state = sigmoid(corr, midpoint=7, steepness=1.2) * 3.0

    pers_gov = np.clip(
        3.5
        + state_focus
        - anti_state
        + voter["political_interest"] * 0.12
        + gov_und * 0.15
        + rng.normal(0, 2.0),
        0,
        10,
    )
    person_or_government_importance = int(pers_gov)

    return {
        "probability_take_part": probability_take_part,
        "candidate_positive_importance": candidate_positive_importance,
        "candidate_negative_fair_importance":
candidate_negative_fair_importance,
        "like_easy_decision": like_easy_decision,
        "like_strong_leader_over_law": like_strong_leader_over_law,
        "understand_government_institutions":
understand_government_institutions,
        "every_person_is_expert": every_person_is_expert,
        "person_or_government_importance": person_or_government_importance,
        "believe_government_institutions": believe_government_institutions,
    }

# — Generate: each voter × each country snapshot


---


N_BASE = (
    constants.N_BASE_GENERATE_VOTERS # base voters; total rows = N_BASE ×
len(COUNTRY_SNAPSHOTS)
)
rows = []

for i in range(N_BASE):
    base = generate_voter_base(i)
    for snap_idx, country in enumerate(COUNTRY_SNAPSHOTS):
        row = dict(base)
        row["snapshot_id"] = snap_idx
        row["base_voter_id"] = i
        row["voter_id"] = i * len(COUNTRY_SNAPSHOTS) + snap_idx

        for field in COUNTRY_FIELDS:
            row[field] = country[field]

        row.update(derive_targets(row, country))

    del row["und_econ_num"]
    del row["und_safety_num"]

```

```

        del row["und_social_num"]

    rows.append(row)

synth_df = pd.DataFrame(rows)
synth_df["source"] = "synth"
n_synth = len(synth_df)

real_df = pd.read_csv(constants.REAL_DATASET_NAME)
real_df["probability_take_part"] = pd.to_numeric(
    real_df["probability_take_part"], errors="coerce"
).clip(0, 100)
real_df["source"] = "real"
real_df["snapshot_id"] = len(COUNTRY_SNAPSHOTS)
n_real = len(real_df)

start_id = len(synth_df)
real_df["voter_id"] = range(start_id, start_id + len(real_df))
real_df["base_voter_id"] = range(N_BASE, N_BASE + len(real_df))

all_cols = list(synth_df.columns.union(real_df.columns))
combined = pd.concat([synth_df, real_df], ignore_index=True)
total = len(combined)

out_path = constants.NON_LINEAR_VOTERS_OUTPUT
combined.to_csv(out_path, index=False)

print(f"Saved {total} total rows → {out_path}")
print(f"Real: {n_real} | Synthetic: {n_synth}")

print(f"    ({N_BASE} voters × {len(COUNTRY_SNAPSHOTS)} country snapshots)")
print(
    f"\nFeature columns: {len([c for c in combined.columns if c not in TARGET_COLS + ['voter_id'] + ['source'] + ['base_voter_id'] + ['snapshot_id']])}"
)
print(f"Target columns: {len(TARGET_COLS)}")
print("\nTarget averages across all country contexts:")
for col in TARGET_COLS:
    print(
        f"    {col:<45s} mean={combined[col].mean():.2f} std={combined[col].std():.2f}"
    )

print("Розподіл усіх вихідних колонок")
target_cols = TARGET_COLS

combined[target_cols].hist(figsize=(18, 12), bins=30)
plt.suptitle("Розподіл усіх вихідних колонок", fontsize=16)
plt.tight_layout()
plt.show()

print("Матриця кореляцій між вихідними даними")
plt.figure(figsize=(10, 8))

```

```
corr = combined[target_cols].corr()

sns.heatmap(corr, annot=False, cmap="coolwarm", center=0)
plt.title("Матриця кореляцій між вихідними даними")
plt.show()

snap0 = combined[combined["snapshot_id"] == 0]
snap0[TARGET_COLS].hist(figsize=(18, 10), bins=30)
plt.suptitle("Snapshot 0 distributions")
plt.show()
```

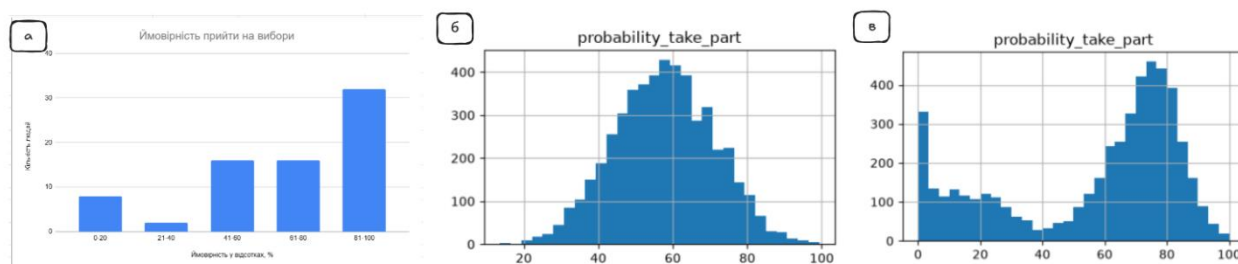


Рис. Порівняння результатів ймовірності прийти на вибори: а – дані з опитування, б – лінійне генерування, в – нелінійне генерування
0% – відсутня явка, 100% – точно буде

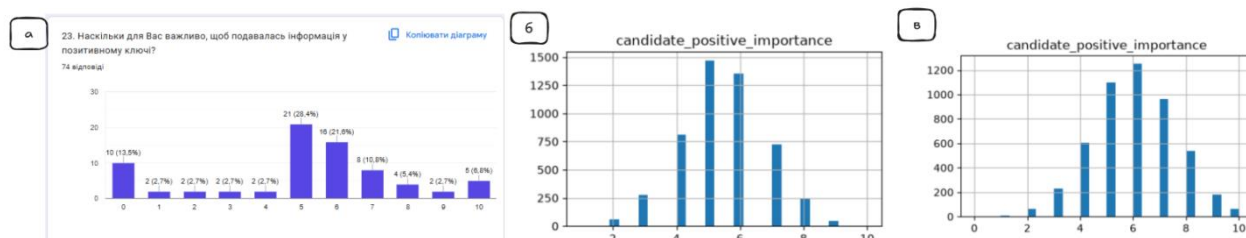


Рис. Порівняння результатів важливості позитивної інформації: а – дані з опитування, б – лінійне генерування, в – нелінійне генерування
0 – не важливо, 10 – дуже важливо

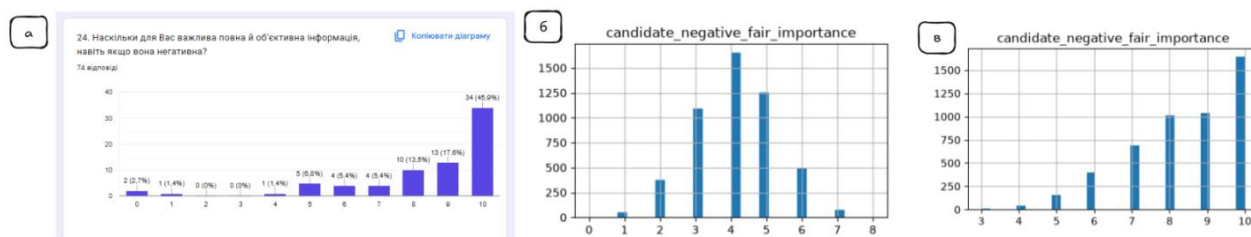


Рис. Порівняння результатів важливості чесної негативної інформації: а – дані з опитування, б – лінійне генерування, в – нелінійне генерування
0 – не важливо, 10 – дуже важливо

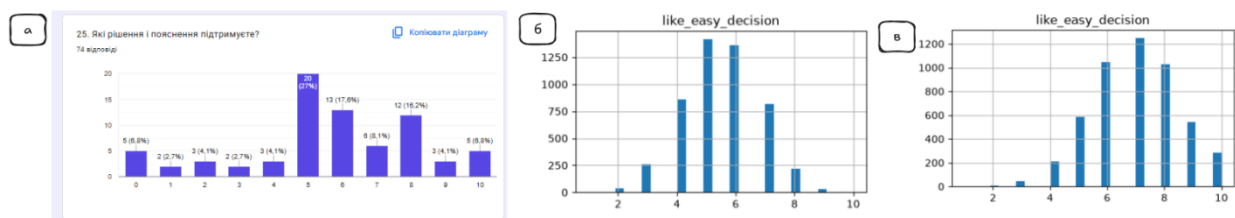


Рис. Порівняння результатів схильності до протих рішень: а – дані з опитування, б – лінійне генерування, в – нелінійне генерування
0 – прості рішення, 10 – складні рішення та пояснення

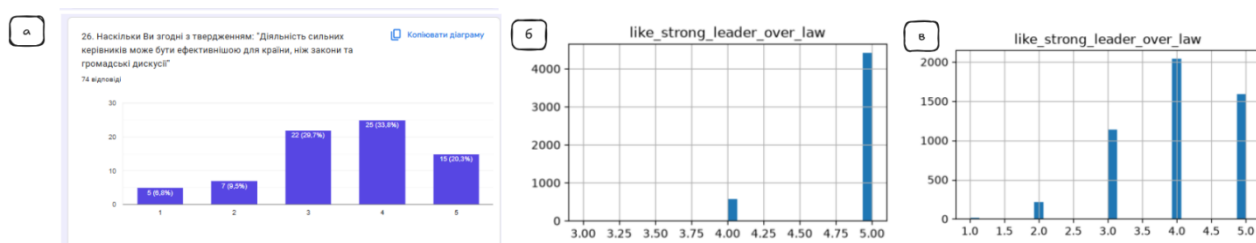


Рис. Порівняння результатів схильності до авторитаризму: а – дані з опитування, б – лінійне генерування, в – нелінійне генерування
1 – демократичні, 5 – авторитарні

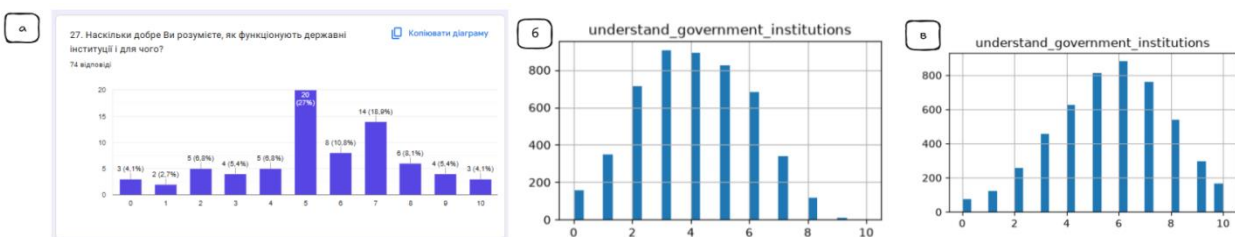


Рис. Порівняння результатів розуміння функціонування державних інституцій: а – дані з опитування, б – лінійне генерування, в – нелінійне генерування
0 – не розуміють, 10 – добре розбираються

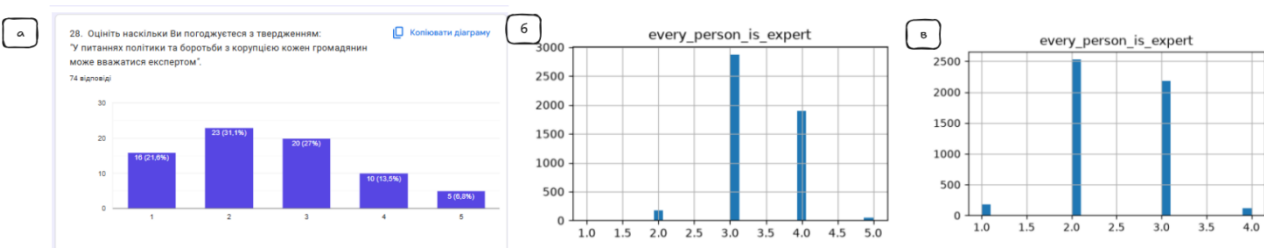


Рис. Порівняння результатів думок, що кожен може бути експертом у політиці: а – дані з опитування, б – лінійне генерування, в – нелінійне генерування
1 – не погодження з тезою, 5 – погодження з тезою

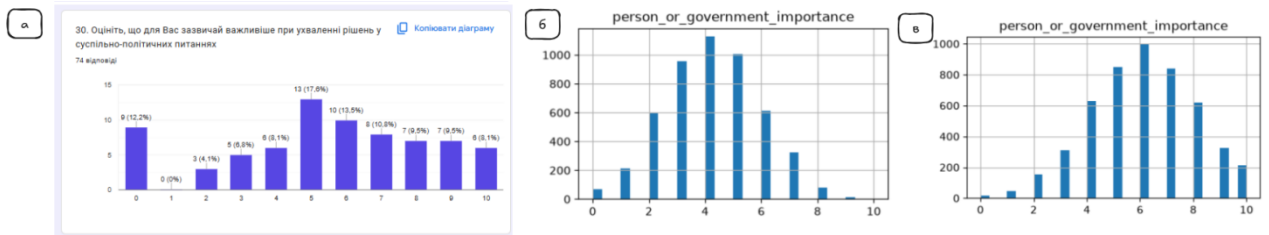


Рис. Порівняння результатів що важливо при ухваленні суспільно-політичних рішень: а – дані з опитування, б – лінійне генерування, в – нелінійне генерування

0 – особисті інтереси, 10 – державні інтереси

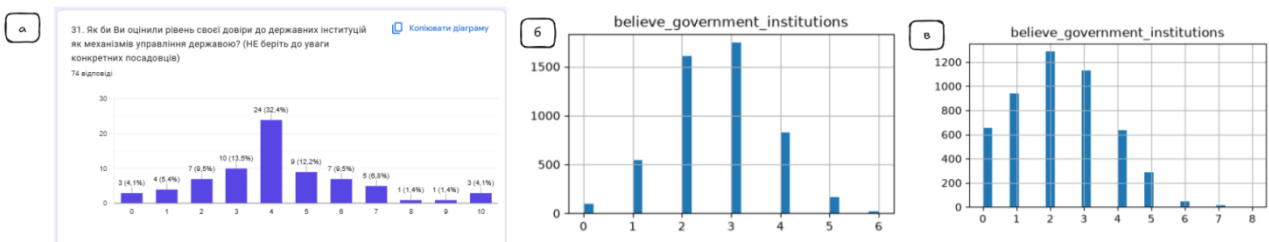


Рис. Порівняння результатів довіри до державних інституцій: а – дані з опитування, б – лінійне генерування, в – нелінійне генерування

0 – не довіра, 10 – сильна довіра

```

"""
TRAIN MODELS
=====
Trains regression models on voter profile + country state → ideal vector.

INPUT  (25 columns): voter demographics + attitudes + 5 country fields
OUTPUT (9 columns):  voter's preference profile

The model learns:
    "given this voter AND this country context, predict their ideal platform"
"""

import matplotlib
import numpy as np
import pandas as pd

matplotlib.use("Agg")
# import matplotlib.pyplot as plt
from pathlib import Path

import joblib
from sklearn.base import clone
from sklearn.compose import ColumnTransformer
from sklearn.ensemble import HistGradientBoostingRegressor,
RandomForestRegressor
from sklearn.metrics import (
    explained_variance_score,
    mean_absolute_error,
    mean_squared_error,
    r2_score,
)
from sklearn.model_selection import GroupShuffleSplit
from sklearn.multioutput import MultiOutputRegressor, RegressorChain
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder, OrdinalEncoder

BASE_DIR = Path().resolve()
Path("models").mkdir(exist_ok=True)
Path("results").mkdir(exist_ok=True)
DATA_PATH = BASE_DIR / "data" / "data" / "non-linear-voters.csv"

# — Load

```

```

df = pd.read_csv(DATA_PATH)
print(f"\n{len(df)} rows loaded")
# df.info()
y_cols = [
    "understand_government_institutions",
    "believe_government_institutions",
    "every_person_is_expert",
    "probability_take_part",
    "candidate_positive_importance",
    "candidate_negative_fair_importance",
    "like_easy_decision",
    "like_strong_leader_over_law",
    "person_or_government_importance",
]
X = df.drop(columns=y_cols)

```

```

X = X.drop(columns=["voter_id", "snapshot_id", "base_voter_id", "source"])

y = df[y_cols]
num_cols = X.select_dtypes(include=["int64", "float64"]).columns
cat_cols = X.select_dtypes(include=["object", "string"]).columns
groups = df["base_voter_id"]

# print(num_cols)
# print(cat_cols)
ORDINAL_COLS = [
    "age",
    "education",
    "economic_status",
    "finance_independent",
    "understanding_econ",
    "understanding_safety",
    "understanding_social",
]
NOMINAL_COLS = ["sex", "nationality", "identity"]
AGE_GROUPS = ["18-30", "31-45", "45-60", "60+"]
EDUCATIONS_LIST = ["9-11 класи", "Профтех/коледж", "Незакінчена вища", "Вища освіта"]
ECON_STATUS_LIST = [
    "Низький",
    "Нижче середнього",
    "Середній",
    "Вище середнього",
    "Високий",
    "Заможний",
]
FIN_INDEPENDENT_LIST = [
    "Повністю залежний",
    "Переважно залежний",
    "Частково і так, і так",
    "Переважно самостійний",
    "Повністю самостійний",
]
UNDERSTANDING = [
    "Не орієнтується у темі та не може пояснити базові поняття",
    "Розуміє загальні ідеї, але складно пояснювати або застосовувати їх",
    "Може пояснити основні поняття іншій людині та розуміє типові приклади",
    "Впевнено застосовує знання на практиці та може аналізувати ситуації",
    "Може критично оцінювати інформацію, знаходити помилки та аргументувати позицію",
]
categories = [
    AGE_GROUPS,
    EDUCATIONS_LIST,
    ECON_STATUS_LIST,
    FIN_INDEPENDENT_LIST,
    UNDERSTANDING,
    UNDERSTANDING,
    UNDERSTANDING,
]

ordinal_transformer = Pipeline(
    [
        (
            "encode",
            OrdinalEncoder(

```

```

        categories=categories,
        handle_unknown="use_encoded_value",
        unknown_value=-1,
    ),
),
]
)

nominal_transformer = Pipeline(
    [
        ("encode", OneHotEncoder(handle_unknown="ignore",
sparse_output=False)),
    ]
)

# print(ordinal_transformer.named_steps)
# sample = pd.DataFrame({
#     "age": ["60+"],
#     "education": ["Вища освіта"],
#     "economic_status": ["Високий"],
#     "finance_independent": ["Повністю самостійний"],
#     "understanding_econ": ["Може пояснити основні поняття іншій людині та
розуміє типові приклади"],
#     "understanding_safety": ["Не орієнтується у темі та не може пояснити
базові поняття"],
#     "understanding_social": ["Може критично оцінювати інформацію, знаходити
помилки та аргументувати позицію"]
# })

# print(ordinal_transformer.fit(X[ORDINAL_COLS]).transform(sample))

# nominal_transformer.fit(X[NOMINAL_COLS])
# print(
#     nominal_transformer.named_steps["encode"].categories_
# )

preprocessor = ColumnTransformer(
    transformers=[
        ("num", "passthrough", num_cols),
        ("ordinal_transformer", ordinal_transformer, ORDINAL_COLS),
        ("nominal_transformer", nominal_transformer, NOMINAL_COLS),
    ]
)

# names = preprocessor.get_feature_names_out()

# for i, name in enumerate(names):
#     print(i, name)

gss = GroupShuffleSplit(n_splits=3, test_size=0.2, random_state=42)

def make_model(model, use_chain=False):
    wrapper = RegressorChain(model) if use_chain else
MultiOutputRegressor(model)
    return Pipeline(
        [
            ("preprocess", preprocessor),
            ("model", wrapper),
        ]
    )

```

```

    ]
)

MODELS = {
    "Multi_RandomForestRegressor": make_model(
        RandomForestRegressor(n_estimators=200, random_state=42, n_jobs=-1),
        use_chain=False,
    ),
    "Chain_RandomForestRegressor": make_model(
        RandomForestRegressor(n_estimators=200, random_state=42, n_jobs=-1),
        use_chain=True,
    ),
    "Multi_HistGradientBoostingRegressor": make_model(
        HistGradientBoostingRegressor(
            random_state=42, max_iter=100, learning_rate=0.05, max_depth=5
        ),
        use_chain=False,
    ),
    "Chain_HistGradientBoostingRegressor": make_model(
        HistGradientBoostingRegressor(
            random_state=42, max_iter=100, learning_rate=0.05, max_depth=5
        ),
        use_chain=True,
    ),
}

results = []

for name, model in MODELS.items():
    print(f"\nНавчання моделі: {name}...")

    fold_scores = []

    for train_idx, test_idx in gss.split(X, y, groups):
        X_train, X_test = X.iloc[train_idx], X.iloc[test_idx]
        y_train, y_test = y.iloc[train_idx], y.iloc[test_idx]

        pipe = clone(model)
        pipe.fit(X_train, y_train)
        Y_pred = pipe.predict(X_test)

        fold_scores.append(
            {
                "mae": mean_absolute_error(y_test, Y_pred),
                "r2": r2_score(y_test, Y_pred),
                "mse": mean_squared_error(y_test, Y_pred),
                "evs": explained_variance_score(y_test, Y_pred),
            }
        )

    test_mae = np.mean([f["mae"] for f in fold_scores])
    test_r2 = np.mean([f["r2"] for f in fold_scores])
    mse = np.mean([f["mse"] for f in fold_scores])
    evs = np.mean([f["evs"] for f in fold_scores])
    print("MAE std:", np.std([f["mae"] for f in fold_scores]))
    print("R2 std:", np.std([f["r2"] for f in fold_scores]))

    results.append(

```

```

        {
            "model": name,
            "test_mae": round(test_mae, 3),
            "test_r2": round(test_r2, 3),
            "mse": round(mse, 3),
            "evs": round(evs, 3),
        }
    )

    print(f"    MAE = {test_mae:.3f}")
    print(f"    R^2  = {test_r2:.3f}")
    print(f"    MSE = {mse:.3f}")
    print(f"    EVS  = {evs:.3f}")

res_df = pd.DataFrame(results)
print("\n=== Фінальні результати ===")
print(res_df.to_string(index=False))

print("\nЗбереження моделі")

best_model_name = min(results, key=lambda r: r["test_mae"])["model"]

best_pipe = None

for name, model in MODELS.items():
    print(f"Фінальне тренування: {name}")

    model.fit(X, y)

    joblib.dump(model, f"models/updated_{name}.pkl")

    if name == best_model_name:
        best_pipe = model
        joblib.dump(best_pipe, "models/update_best_model.pkl")

with open("models/updated_best_model.txt", "w") as f:
    f.write(best_model_name)

res_df.to_csv("results/updated_model_comparison.csv", index=False)

print(f"Збережено найкращу модель: {best_model_name}")

```