Національний університет

«КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем

Індивідуальний планувальник часу із використанням технолій штучного інтелекту

Курсова робота

Студента III року навчання

факультету інформатики

Стасюка Іллі Вікторовича

Науковий керівник –

Доцент заступник декана

Афонін А. О.

Київ – 2023

Зміст

Вступ

Тема

Розробка розумного індивідуально-орієнтованого планувальника часу із використанням генетичного алгоритму та великих мовних моделей.

Актуальність

В нашому швидкоплинному сучасному світі, ефективне планування часу стає не просто перевагою, а необхідністю. Тиск часу відчуває кожна людина, від студента до високого керівника, і відповідно виникає потреба в засобах, які б допомагали нам краще розподіляти наш найцінніший ресурс - час. З огляду на це, значний обсяг досліджень було направлено на розробку планувальників часу, інструментів, що мають на меті оптимізувати наше використання часу.

Проте, незважаючи на велику кількість існуючих методик та інструментів для керування часом, багато людей продовжують відчувати виклики у цьому аспекті. Часто відсутність ефективності в плануванні часу пояснюється не недостатнім знанням методик, а складністю їх практичного застосування.

Ця проблема стає особливо важливою в контексті сучасного динамічного інтернет-суспільства, де кожна особа має справлятися з багаточисельними обов'язками і завданнями. Попри це, до цих пір не існує універсального інструменту, орієнтованого на індивідуальні потреби користувача, який би міг вичерпно задовольнити потребу в плануванні завдань та часу на різних масштабах. Також не існує інструменту, який би слугував провідником для людей, які бажають навчитись тайм-менеджменту.

Наша робота спрямована на розробку новітнього, динамічного, персонально-орієнтованого планувальника часу, який використовує генетичний алгоритм та великі мовні моделі для адресації викликів тайм менеджменту.

Ми віримо, що такий підхід може значно поліпшити ефективність планування часу та зробити цей процес більш доступним і зручним для користувачів.

Ми детально розглянемо концепцію такого планувальника, розкриємо технічні деталі його реалізації та оцінимо його потенціал для вирішення проблеми недостатньої ефективності управління часом. Нашою метою є створення інструменту, який не просто допоможе людям краще планувати свій час, але й зробить цей процес більш пізнавальним і цікавим, сприяючи розвитку навичок тайм-менеджменту.

Мета

Мета даного дослідження полягає в розробці прототипу інструменту, що спрощує процес запису та планування численних завдань. Особливий акцент ставиться на застосування технологій штучного інтелекту, зокрема великих мовних моделей та генетичних алгоритмів, які спрощують взаємодію з користувачем. Останнім кроком є тестування практичності розробленого планувальника, а також аналіз перспектив його подальшого удосконалення та розвитку.

Завдання

1. Визначити завдання які постають при плануванні.
2. Визначити функціонал планувальника для вирішення цих завдань.
3. Визначити і формалізувати сутності планувальника.
4. Розробити мінімальну версію планувальника.
5. Провести тестування і аналіз перспектив розробленого планувальника.

Об’єкт дослідження

Об'єктом дослідження є процес планування і використання людиною часу в повсякденному житті з метою виконання задач і досягнення цілей.

Предмет дослідження

Предметом дослідження є розробка індивідуально-орієнтованого планувальника часу, який використовує великі мовні моделі та генетичні алгоритми.

Розділ 1. Модель планувальника

Завдання планування

Планувальник - це комплексна система, яка призначена для вирішення задачі планування. Планування в цьому контексті являє собою три фундаментальні задачі:

- Постановка завдань
- Планування часу
- Аналіз результатів

Для їх вирішення ми побудуємо модель, що відображатиме цілі та наміри користувача, та даватиме йому змогу проводити аналіз своїх успіхів.

Відповідно до завдань вводяться три типи сутностей:

- Діяльність (Activity)
- План (Plan)
- Лог (Log)

На основі них будується обширний функціонал, що вирішує всі поставлені завдання. Детальніше про кожну з них в наступних розділах.

Функціонал

Загалом, планувальник має дуже обширний функціонал, який покликаний в першу чергу задовольнити всі потреби користувача та зробити планування максимально зручним та ефективним.

Список всіх можливостей планувальника із короткими поясненнями наведений нижче.

Постановка завдань

1. Зберігання всіх завдань – від найбільшої цілі до списку покупок на сьогодні. Адже лише врахувавши все можна отримати глобальну картину.

2. Дерево підзавдань. Великі цілі потребують деструктуризації і спрощення. Деталізуйте свої задачі, не ризикуючи згубити ліс за деревами. Розподіляйте задачі по проектам, як корені дерева.
3. Відношення між завданнями. Відношення дозволяють вказати порядок виконання завдань і задати додаткові залежності при плануванні.
4. Дедлайни. Система триматиме вас в курсі всіх можливих дедлайнів та сигналізуватиме про їх наближення та пріоритетність.
5. Пріоритети. В умовах високого навантаження дуже важливо в першу чергу зосередитись на найбільш важливих речах.
6. Статус виконання. Викреслюйте виконані завдання, скасовуйте непотрібні, делегуйте неважливі, або заморожуйте нетермінові - а система потурбується щоби ви завжди бачили лише актуальну інформацію.

Планування часу

1. Керування планами. Легко відображайте свої наміри та тримайти все під контролем і на виду. Зручний таймлайн зведе ваше планування до кількох кліків миші.
2. Рівні планування. Перетворюйте вашу велику ціль в список задач на день.
3. Врахування навантаження. Завжди майте бачення глобальної картини та розуміння наскільки завантажений той чи інший день / тиждень / місяць.
4. Абстраговане планування. Частіше всього ми не знаємо чим саме будемо займатись в майбутньому. Та це не потрібно – виділити час на довгострокову ціль можна відразу, а уточнити задачі вже по ходу виконання.
5. Повторення завдань. Допоможе вам нічого не забути та збереже ваш час на планування.
6. Врахування непередбачуваного. Не можна передбачити все наперед. Але можна пристосуватись до нових умов. Легко редагуйте свої плани, відкладайте на потім, і переключайтесь на інші задачі.
7. Генетична генерація оптимального розкладу на основі ваших обмежень зекономить ваш час на планування і зробить більшу частину марудної роботи.
8. Розумний планувальник. Просто продиктуйте задачу, щоби її не забути. Керуйте своїми планами голосом. Розкажіть про свій день щоби створити підсумок. Штучний інтелект вивчатиме ваші плани і допомагатиме вам ефективніше і швидше планувати. Вам більше не доведеться витрачати безліч зусиль на постійне планування записування усього.

Аналіз результатів

1. Облік затраченого часу. Фундаментальна річ, що дасть вам змогу більше ніколи не задаватись питанням «куди подівся мій час». Аналізуйте

комплексну інформацію через зручний інтерфейс та наглядні діаграми та графіки.
2. Детальна статистика за довільний проміжок часу. Потрібно освіжити в пам’яті чим ви займались останній місяць чи тиждень? Просто перегляньте список завершених чи розпочатих за цей період завдань. Переглядайте та аналізуйте дані у вигляді наглядних графіків.
3. Прогнозування. Найкращий показник того як ви справитесь в майбутньому – це те, як ви справлялись в минулому. Прогнозування допоможе вам врахувати свої помилки і побудувати надійні плани, а також передбачити можливі перешкоди.
4. Відстеження звичок. Плануйте повторювані плани щоби формувати звички. Відстежуйте виконання та пропуски певної звички. Ставте цілі на звичку.
5. Історія діяльності. Не пам’ятаєте коли були у стоматолога? Просто вбийте в пошук і дізнайтеся дату. Або ж впишіть дату, щоби в точності до годин пригадати чим ви займались в потрібний день. Деякі моменти зовсім не хочеться забувати. Записуйте спогади та прикріплюйте фотографії, щоби повернутися до них потім.
6. Глобальна картина життя. Переглядайте на таймлайні роки і десятиліття свого життя та легко пригадуйте важливі події. Школа, робота, подорожі, стосунки – що завгодно, ви маєте право пам’ятати все.

Сутності

Activity

Концепт

В сучасному світі люди зайняті різноманітними справами кожного дня. Від виконання рутинних завдань до складних проектів, від домашніх завдань до корпоративних питань, від приготування їжі до особистого розвитку, від миття посуду до написання курсових робіт. Ця незкінченна множина діяльностей може бути надзвичайно складною для опису та систематизації, оскільки справи можуть бути дуже різні між собою та бути абсолютно унікальними для кожної людини. До прикладу:

- Справи можуть бути як добре розплановані так і різко неочікувані, але однаково потребувати вашої уваги.

- Справи можуть бути одноразовими, або рутинними, повторюючись із певними проміжками.
- Справи можуть бути елементарними, або ж комплексними, які потребують розбиття на менші справи, щоби їх можна було виконувати крок за кроком.
- Також справи можуть відрізнятися тривалістю виконання. Деякі справи можуть бути завершені за декілька годин, тоді як інші можуть займати тижні, місяці або навіть роки.
- В більш складному випадку справи можуть бути взаємопов'язані та залежати одна від одної певним чином. Це робить процес планування та управління своїми справами важким завданням без правильного інструменту.

Проте загалом, вся множина справ які нам доводиться виконувати має певні спільні властивості. Щоби ефективно відобразити наші справи у планувальнику, створимо спеціальну модель, виділивши ці властивості в окрему сутність "Activity".

Концептуалізація справ за допомогою цього поняття дозволяє аналізувати та організовувати справи, визначати їх структуру та взаємозв'язки, оцінювати ресурси, необхідні для їх виконання, а також керувати ними, відстежувати прогрес, порядок виконання та визначати пріоритети.

Визначення

Activity – це самостійна унікальна одиниця, яка описує певну діяльність або множину дій. Activity може бути спрямоване на досягнення конкретної мети – тоді ми також використовуємо термін «Task» (завдання). Або ж Activity може бути просто об’єднанням справ за схожістю, що не обов’язково передбачають завершення, з метою відстеження певних показників. В такому випадку це просто «Activity».

Activity має рекурсивну структуру, тобто одна сутність може включати в себе дочірні сутності. Це зумовлює формування ієрархічної структури, де одне Activity може мати під собою ціле дерево нащадків, та нащадків своїх нащадків. Часто таке розбиття є просто необхідним для досягнення мети. Воно сприяє формуванню чіткої

перспективи обсягу роботи, необхідної для виконання. А також конкретизує завдання, тобто зменшує його рівень абстракції і невизначеності, роблячи його доступним до виконання. При цьому зберігаються різні рівні абстракції, що відповідають різним рівням ієрархії - від найнижчого до найвищого відповідно. Це дає можливість одночасно відстежувати як кожну деталь завдання, так і аналізувати його із висоти пташиного польоту. Адже перше необхідне власне для виконання завдання, тоді як останнє – для відстеження свого прогресу та коригування і досягнення цілей в цілому.

Activity також визначає всі інші можливі відношення. Ці залежності можуть вплинути на час виконання та дедлайни. Наприклад, виконання одного завдання може вплинути на можливість виконання іншого, яке передбачає використання тих самих ресурсів. Або ж, певне завдання може бути залежним від іншого, яке має бути завершене перед тим, як розпочати нове.

Загалом, Activity інкапсулює в собі всі можливі властивості, відношення та показники, що характеризують діяльність, її зв’язки та стан.

Властивості – це значимі для нас дані, які визначають кожну унікальну сутність Activity згідно нашої моделі.

Основні властивості включають в себе:

- Назва – унікальна коротка назва, що ідентифікує та описує діяльність.

Та властивості які мають лише Task:

- Статус – значення, яке відображає стан виконання діяльності. В загальному ці стани можна згрупувати як «Не розпочато», «Розпочато», та «Закрито».
- Дедлайн – дата, що позначає кінцевий термін, до якого повинне бути завершене завдання.
- Оцінка часу – сукупний орієнтовний час який потрібен для закриття цього завдання. Визначається користувачем.
- Пріоритет – важливість завдання, яку визначає користувач. Більш важливим завданням повинно виділятися більше ресурсів, у випадку їх дефіциту.

Відношення – це певні зв’язки із іншими сутностями. Відношення допомагають формувати структуру діяльностей і враховувати певні показники або обмеження зумовлені залежнастями між діяльностями.

Структурні відношення:

- Дочірні Activity – це більш конкретні Activity, що складають частину даної сутності. Відображає множину завдань, які потрібно виконати для досягнення мети, закріпленої за даним завданням. Їх може бути необмежена кількість і вони можуть різнитись за обсягом і властивостями загалом. Показники дочірніх Activity також враховуються в батьківському Activity.
- Батьківські Activity – це більш загальні Activity, в які входить дана сутність. По суті означають більш загальну мету, якої стосується виконання даного завдання. Батьківських Activity також може бути більше одного, за винятком того що лише одне з них може бути сутністю Task. Деякі показники батьківських Activity також можуть враховуватись в дочірніх Activity, до прикладу пріоритет.

Інші відношення:

- Блокуючі завдання – це відношення, яке позначає завдання які повинні бути завершені перед початком даного завдання. Враховується при автоматичній генерації розкладу, а також при визначенні терміновості.
- Плани – визначає плани, що стосуються конкретної діяльності. Це фундаментальна залежність, яка дає змогу керувати часом. Детальніше про це в секції «План».

Показники – це певні значення, що обраховуються за доступною інформацією у процесі існування та взаємодії із Activity. Вони призначені надавати користувачу корисну і зрозумілу інформацію щодо стану дільності.

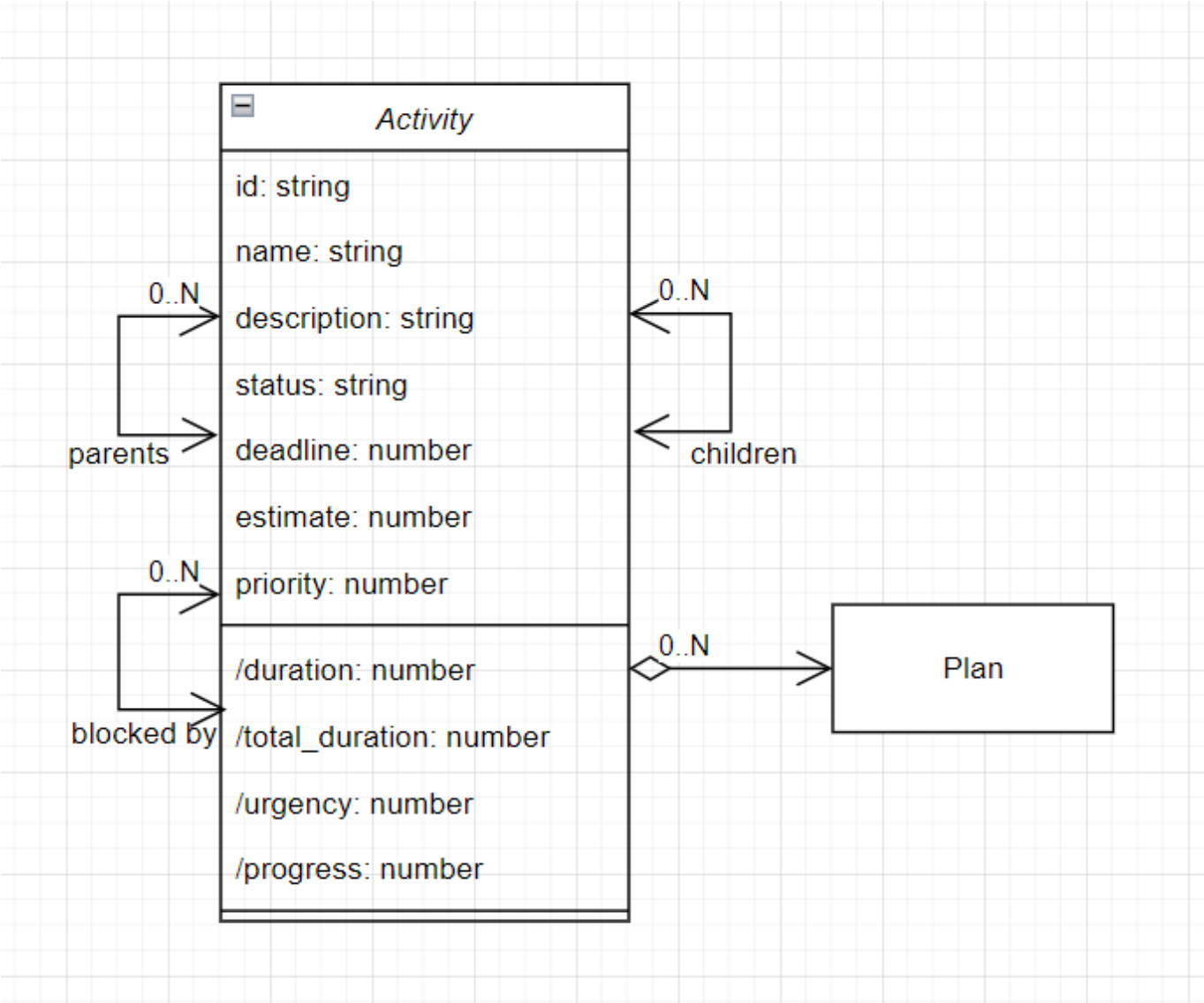
Основні показники:

- Тривалість – показує скільки часу загалом було витрачено на дану діяльність. Цей показник також включає в себе час затрачений на дочірні діяльності, виключаючи можливі повторення у нащадках.
- Об’єм – показує який орієнтовний обсяг в часових одиницях роботи потрібно виконати, щоби завершити дане завдання. Обраховується на основі оцінок часу дочірніх Діяльностей та статистики про фактично затрачений час. Це дає змогу врахувати і скорегувати суб’єктивність оцінки часу користувачем.
- Терміновість – подібно до пріоритету, але обраховується на основі об’єму та дедлайну. Показує наскільки нагальним є це завдання. Чим більший об’єм роботи потрібно виконати та чим менше часу на це є – тим терміновіше завдання. Це дає змогу користувачу не просто орієнтуватися на свої пріоритети, які часто бувають визначені неоднозначно, а оцінювати ситуацію на основі актульного стану.
- Прогрес – це показник, який показує відсоток виконаної роботи від загального обсягу. Цей показник може бути корисним для відстеження прогресу виконання завдання та оцінки результативності. Користувач може встановити мінімальний рівень прогресу, який вважатиме прийнятним для завершення діяльності.

В цілому, показники допомагають користувачеві більш точно оцінювати та керувати своєю діяльністю, враховуючи різні фактори, які можуть вплинути на процес виконання завдання та його результат.

Формалізація

Формалізуємо ці властивості та можливі відношення між сутностями.



Властивості:

- id – унікальний ідентифікатор, що використовується як для розрізнення різних Activity, так і для встановлення реляцій між ними.
- name – Назва. Може містити скорочення та абревіатури.
- description – розгорнутий опис, де за потреби розшифровується назва, та який містить ключові слова, по яким система планувальника могтиме здійснювати пошук і розпізнавати належність до певного контексту.
- status – Статус. Певне значення із скінченного переліку значень, які може редагувати користувач:
    - Open
        - To do - потрібно зробити.

        - Suspended - потрібно зробити в загальному, але не актуальне зараз.
        - Blocked - завдання очікує завершення якогось іншого завдання.
    - In progress
        - In progress - завдання було почато, але ще не закінчене.
    - Closed
        - Done - завдання повністю виконане, мети досягнуто.
        - Cancelled - завдання скасоване через відсутність потреби в ньому.
        - Abandoned - завдання покинуте через брак інтересу чи мотивації і навряд буде виконуватись.
- deadline – Дедлайн – дата та, можливо, час.
- estimate – Оцінка часу в годинах.

Показники:

- duration – Тривалість в годинах.
- total_duration – Об’єм в годинах.
- urgency – Терміновість, невід’ємне число. Чим більше значення, тим більша терміновість.
- progress – Прогрес у відсотках.

Відношення:

- parents – батьківські Activity.
- children – дочірні Activity.
- blocked_by - блокуючі Activity.

Plan

Концепт

Людині притаманно прагнути контролюати своє буття і хоча б приблизно знати своє майбутнє. Тому ми постійно будуємо плани. Плани вивчитись в іневерситеті чи почати займатись спортом, плани зустрітись із друзями чи сходити в кіно, плани

викинути сміття чи приготувати їсти. Ми визначаємо завдання які нам потрібно зробити, але окрім цього ми постійно маємо приблизне уявлення чи сподівання коли і як довго ми будемо над ними працювати. Іншими словами ми постійно займаємось плануванням та розподіленням свого часу.

Поняття «план» тут є доволі розмитим. Адже, окрім відношення до різних діяльностей, в залежності від контексту ми можемо мати на увазі як плани на день, так і плани на цілий рік. При цьому чим ширший проміжок часу, тим менш детерміновані наші плани, тим менше ми маємо усвідомлення про те, коли саме будемо їх виконувати.

Скоріше за все ви маєте досить конкретні плани на тиждень вперед. В найпростішому випадку у вас є уявлення про свій завтрашній день. І коли вас запитають чи вільні ви увечері, ви швидко проаналізуєте що у вас було заплановано на цей час, та чи можливо ці плани перенести – і тоді матимете відповідь.

З меншою вірогідністю ви знаєте чим будете займатись наступних кілька місяців. Можливо ви плануєте завершити стару справу чи розпочати новий проект, або ж почати займатись йогою чи знайти собі нове житло. Ці плани охоплюють куди ширший проміжок часу і разом з тим несуть в собі менше впевненості. Адже якщо вас запитають, чи будете ви вільні увечері рівно через місяць, то скоріш за все ви відповісте «не знаю», хоча ви можете бути впевнені на 100% що таки почнете займатись йогою до того часу.

Відобразити це широке та нечітке поняття може бути доволі складно, проте воно є фундаментальним в системі планування часу. Для цього введемо в нашій моделі введемо сутність Plan («План»).

Це дасть змогу описувати і вимірювати свій час та організувати свої дії. Це допоможе нам не просто зберігати контроль над своїм часом, а й ефективно ним керувати, аналізувати та якнайкраще адаптуватись під нові обставини.

Визначення

Plan – це одиниця планування часу. План описує що, коли та як довго буде виконуватись або виконувалось. План обов’язково належить до певного Activity та займає певний часовий відрізок. Також план може мати передбачену тривалість. В сукупності плани утворюють кінцевий розклад, який задовольняє обмеженням і вимогам користувача і орієнтований на оптимальне використання часу.

Сутність План інкапсує в собі наступні властивості:

- Часовий проміжок – визначає інтервал часу в межах якого має виконуватись діяльність, якої стосується план. Може належати до різних рівнів планування. Рівні визначають максимальну та мінімальну протяжність часового відрізку, який займає план. До прикладу можна виділити рівень годин, днів, тижнів, місяців, кварталів та років.
- Оцінка часу – визначає кількість часу, яка повинна бути витрачена на діяльність плану за вказаний відрізок часу плану.
- Нотатка – опціональна інформація, що стосується конкретно цього плану, а не діяльності загалом.

Показники:

- Тривалість – складне значення, яке визначається на основі оцінок часу планів нижчих рівнів, які стосуються діяльності даного плану. Цей показник допомагає враховувати загальну оцінку часу при конкретизації планів, а також розуміти де були допущені прорахунки.

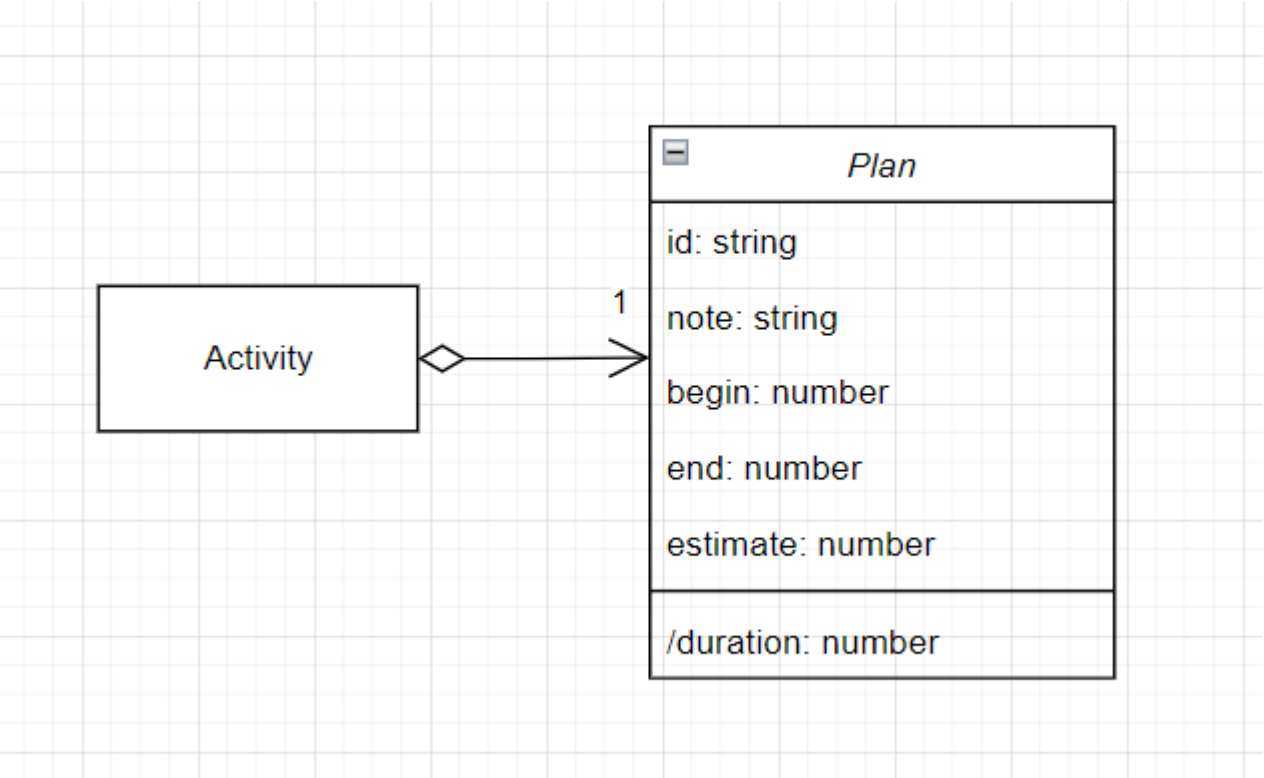
Та відношення:

- Activity - кожен план обов’язково стосується певної діяльності.

Такий підхід дає змогу універсально визначати плани різних рівнів та, відповідно, різного ступеня конкретності. Це дає змогу будувати плани на широкі проміжки часу, скажімо на місяці, не знаючи в які дні конкретно ми будемо їх виконувати. Проте це й не потрібно, адже для того щоби завершити завдання і правильно визначити своє навантаження, достатньо лише оцінити необхідну тривалість цього плану, тобто сукупний час, який треба витратити за цілий місяць.

Таким чином ми можемо абстрагуватися від неважливих деталей і будувати та змінювати свої плани більш широко, на рівні місяців чи навіть років. А із часом поступово уточнювати їх аж до конкретних годин. При цьому можна легко врахувати необхідні обмеження та рівномірно розподілити велику кількість часу, необхідну на виконання завдання, по великій кількості днів.

Формалізація



Властивості:

- id – унікальний ідентифікатор плану. Також служить для встановлення реляцій.
- note – Нотатка.
- begin – початок Часового проміжку.
- end – кінець Часового проміжку.
- estimate – Оцінка часу.

Показники

- duration – Тривалість.

Відношення:

- Activity – діяльність до якої належить даних план.

Log

Концепт

Людині притаманно переоцінювати себе, особливо коли задача полягає в плануванні часу. Цей феномен оптимістичного упередження щодо необхідної кількості часу для завершення задачі відомий як «planning fallacy» [4]. Вірогідно вам самим добре відомо як це, або гірше того, ви досі продовжуєте допускатися цієї помилки. Результатом такої недооцінки доволі очевидні. Проте щоби не потрапляти щоразу в таку пастку, потрібно мати певні вміння. Вміння, які набуваються із досвідом при плануванні та, найголовніше, аналізі часу. Саме аналіз часу, виявлення своїх прорахунків та вияснення їхніх причин дозволяє нам навчитись правильно керувати своїм часом.

Проте це доволі важко, або й зовсім неможливо зробити у випадку тривалих проміжків часу. Якщо ви не пам’ятаєте скільки часу планували витратити, та не знаєте скільки часу витратили, то ні про який аналіз не може й іти мови. Тому набуття цього визначального вміння потребує наявності правильного підходу та інструменту. У контексті нашого планувальника часу це завдання адресується сутністю «Log» (лог, запис).

Визначення

Log – це, подібно до плану, одиниця запису часу. Він має таку ж саму структуру, але інше призначення – логи слугують інструментом обліку витраченого часу. Тобто логи створюються постфактум і позначають кількість фактично витраченого часу за певний часовий проміжок.

Концепт збереження логів відіграє дуже важливу роль в системі планувальника. Логи роблять можливим повноцінний аналіз часу та відстеження прогресу, тому що вони зберігають інформацію про затрачений час та про те, скільки часу планувалось

витратити насправді. І хоч по одинці ця інформація не є особливо корисною, в загальній сукупності вона дає змогу знайти прорахунки та зрозуміти причини переоцінки часу, а відповідно й те, як їх можна усунути.

При аналізі часу дуже важливо бути певним в точності інформації про витрачений час, інакше це може призвести до неправильних припущень. Для того щоби забезпечити цю точність, від користувача очікується вчасне створення логів, які відображатимуть справжній стан справ.

Подібно до планів, логи можуть належати до рівнів логування, але зазвичай в цьому немає великої потреби, оскільки логування найпростіше робити по ходу виконання завдань на рівні годин та днів.

Сутність Log має таку ж структуру і такі ж властивості як сутність Plan.

Розділ 2. Процес планування

Процес планування є ітеративним і складається із трьох фундаментальних етапів. Він являє собою поступове перетворення ваших цілей в тверді наміри, що формалізовані відповідними сутностями планувальника. Кожен етап вирішує конкретну проблему планування та залежить від результатів попереднього етапу.

Постановка завдань

Визначення цілей і завдань

Планування починається із визначення усіх можливих завдань, від великих та важливих, до малих та незначних. Нам необхідно враховувати все, щоби отримати глобальну картину і могти мінімізувати прорахунки та правильно розподілити час на великих проміжках часу. Це гарантує що нічого не загубиться і не буде невидимих затрат часу. Більше того, це дає змогу проводити комплексний аналіз затраченого часу.

В першу чергу визначаються ваші найважливіші цілі, як от розробка свого стартапу. Відповідно до них відбувається постановка проектів, які являють собою завдання верхнього рівня ієрархії, та інших великих завдань. Великі завдань потребують подальшої конкретизації, але цей етап далеко не завжди є очевидним. Проте це не проблема, тому що у даній системі достатньо лише тієї інформації якою ви володієте прямо зараз, щоби змогти виділити необхідний на виконання цього завдання час.

До списку діяльностей також вносяться будь-які малі завдання, як от покупка хлібу в магазині. Незважаючи на їх можливу незначущість, їх проте може бути дуже багато і вони теж повинні бути враховані.

Кожне завдання чи діяльність записується як відповідна сутність Activity.

Структуризація завдань

В результаті ми отримаємо розмаїту множину завдань, які можуть включати як цілі проекти, так і кількахвилинні задачі. Зрозуміло, що така велика різниця в масштабах робить неможливим їх порівняння, а відповідно й раціональний розподіл часу. Тому великі завдання потребують конкретизації та розбиття на менші підзавдання. Малі ж завдання в свою чергу може бути зручно згрупувати під більш загальними діяльностями, що в подальшому дасть змогу легше враховувати їх на більш загальному рівні.

На цьому етапі формується структура діяльностей, які можна зобразити як дерево, або ж як орієнтований граф. Таке представлення допоможе візуально оцінити об’єм необхідної роботи, а також легко переглядати деталі комплексних задач, не втрачаючи загальності.

Визначення обмежень та вимог

Після того як була визначена основна множина діяльностей, залишається додати базову, проте необхідну для подальшого планування інформацію, таку як оцінки часу та дедлайни.

Дедлайни потрібні для пріоритезації завдань на основі терміновості та формування оптимальних планів, що забезпечить вчасне виконання поставлених завдань.

Оцінка часу потрібна для правильних розрахунків часу при довгостроковому плануванні. Це дає змогу «бронювати» час заздалегідь та правильно рахувати сукупне навантаження, що в свою чергу позбавляє необхідності відразу враховувати всі підзавдання.

Пріоритети слугують індикатором важливості завдання. Таким чином планувальник буде знати, що ці речі мають бути виконані першими. І при плануванні у випадку високого навантаження та нестачі часу для виконання всіх завдань, першими будуть виконуватись саме ті, що мають високі пріоритети.

Статус виконання потрібен в першу чергу для того, щоби зосереджувати свою увагу лише на актуальних завданнях. Адже після завершення завдання, користувач більше

не має в ньому інтересу в контексті планування. Проте воно все ще повинно бути доступним для перегляду за необхідності.

Раціональне визначення властивостей діяльностей може бути складною задачею, але цей етап є ключовим у плануванні часу. Адже наступний етап планування початково базується саме на цій інформації і тому оптимальність кінцевого загального плану залежить від реалістичності цих властивостей. Очевидно, що завжди буде присутня похибка, проте вона її можливо мінімізувати завдяки етапу аналізу часу та ітеративності процесу планування.

Планування часу

Етап планування часу являє собою свідомий розподіл часу на визначені завдання, з врахуванням усіх заданих обмежень. Метою цього етапу є отримання оптимального розкладу, тобто сукупності планів, яка дозволить продуктивніше використовувати час і дійти до поставлених цілей.

У випадку коли кількість завдань дуже велика, а ресурси обмежені, планування допоможе розподілити час найоптимальнішим чином і заздалегідь визначити критичні ділянки. Більше не потрібно буде постійно тримати в голові свій розклад, що стає значно складнішим, коли у вас кожного дня різні завдання.

Крім того планування робить можливим паралельно виконання багатьох завдань (не плутати із мультитаскінгом). Іншими словами, воно дає змогу набагато легше зосередитись на багатьох завданнях в різний час дня чи тижня, при цьому не втрачаючи курсу до своїх цілей і дотримуючись пріоритетів.

Наостанок, у випадку коли завдання є комплексними і довготривалими, планування стає просто необхідним для виконання завдань в строк.

Як вже було згадано вище, плани можна розподілити по різним рівням за проміжком часу який вони охоплюють. Такий підхід дає змогу планувати в різній перспективі, при цьому працюючи із планами які подібні між собою за охопленими проміжками часу. Це значно спрощує їх сприйняття, відображення, а також обрахунок

показників, таких як загальне навантаження. Крім того це пришвидшує сам процес планування, оскільки користувачу стає елементарно простіше створювати ці плани в системі – достатньо просто один раз клікнути в потрібному місці, замість постійно марудно вводити дві дати.

Не менш важливим є те, що рівні планів дарують можливість абстрагованого планування. Тобто ви абстрагуєтесь від конкретних дат та значень, та починаєте своє планування із вказання сукупної кількості часу, яку потрібно виділити на завдання за певний проміжок часу. Чим вищий рівень, тим більш тривалому проміжку часу і менш конкретним планам він відповідає.

Алгоритм

Коли всі завдання визначено, процес планування починається із найвищого рівня, до прикладу місяця. Спочатку плануються кореневі завдання та діяльності. Для кожної з них на місяць виділяється кількість часу, яка базується на їхній оцінці часу. На цьому етапі враховуються пріоритети, дедлайни та сукупне навантаження. За потреби робиться переоцінка часу чи дедлайнів та перерозподіл ресурсів. Тобто вже на початковому етапі планування часу можна виявити прорахунки.

Коли верхній рівень місяців розплановано оптимальним чином, ми переходимо до наступного кроку і спускаємось на рівень нижче, в даному випадку тижнів. Далі для кожного завдання планування відбувається в контексті більш широких планів, тобто планів вищого рівня. Тобто для окремого завдання ми повинні розподілити час по чотирьох тижневих планах таким чином, щоби сумарна кількість часу дорівнювала тому, який був запланований нами на місяць на попередньому етапі. Тут ми також враховуємо сумарне навантаження та інші вимоги. Такий підхід має великі переваги, оскільки він дає змогу дуже просто задовольнити дедлайни та пріоритети, просто дотримуючись вимог які ми задали на рівні вище. Це дає нам можливість деталізувати свої плани, не втрачаючи глобальної картини. В результаті цього кроку ми отримаємо оптимальну множину планів для даного рівня і одночасно створимо основу для наступного.

Цей процес відбувається поступово, аж допоки ми не спустимось до базового рівня, скажімо рівня годин. Таким чином із наших великих цілей та планів ми отримуємо конкретний ту-ду список на день, із яким уже можливо працювати. При цьому можна бути певним, що при дотриманні цих планів ми вкладемось в строк і отримаємо найбільш оптимальний результат в глобальній перспективі. Це було б неможливо зробити, якби не поділяли плани на рівні.

В процесі самого спуску є два способи зменшення абстракції:

- Уточнення роботи, тобто Діяльностей.
- Уточнення часу, тобто Планів.

Уточнення роботи

Уточнення роботи означає що ви спускаєтесь по дереву діяльностей вниз і будуєте плани для нащадків діяльності вищого плану. При цьому проміжок часу планів не обов’язково повинен звужуватись. До прикладу у вас є місячний план на навчання в університеті. Воно передбачає в собі вивчення кількох предметів паралельно. Для кожного з предметів ви також виділяєте час на місяць в контексті обмежень плану на навчання в університеті. Таким чином відбувається конкретизація плану через уточнення завдань. В результаті на найнижчому рівні планів у вас будуть лише конкретні кінцеві завдання.

Уточнення часу

Уточнення часу означає звуження проміжку охопленого часу для планів відповідно до рівнів. У випадку місячного плану на навчання в університеті, це означатиме що ви створюєте тижневі плани на навчання в університеті, які в сукупності задовольнятимуть вимогам вищого рівня. Ключовим моментом тут є те, що сама перспектива планування теж зменшується. Тобто ви можете запланувати лише перший тиждень, замість всіх чотирьох, а інші визначити уже по ходу. Відповідно й для нижчих рівнів. Це дає можливість поступово уточнювати плани. Це є дуже важливою потребою, оскільки чим нижчий рівень планів, тим конкретнішими вони повинні бути. Відповідно щільність планів, їх кількість та кількість зусиль

потрібних для планування збільшується. При збільшенні кількості планів збільшується і ймовірність того що щось піде не так. Тому тим менш впевнені ми можемо бути в тому, що буде далі. Саме тому поступове уточнення планів із часом є фундаментальною потребою.

Аналіз результатів

Аналіз результатів – це напівавтоматичний процес, який на основі інформації про витрачений час на кожне завдання допомагає знайти причини прорахунків, врахувати їх і перерозподілити свій час, та, найголовніше, навчитись правильно оцінювати час, що максимізує ефективність, а й відтак користь від планування часу.

Процес аналізу є напівавтоматичним тому, що по більшій мірі це програмна обробка даних, які порівнюються та підраховуються. На основі цього в кінцевому рахунку будуються наглядні графіки, які користувач використовуватиме в цілях аналізу та отримання висновків. В складнішому випадку можлива реалізація інтелектуальної системи на основі методів штучного інтелекту, яка видаватиме рекомендації щодо змін.

Сам аналіз являє собою обрахунок і порівняння сумарних кількостей фактично витраченого та запланованого часу. Це дає змогу отримати багато корисної інформації, як от:

- які плани були виконані, а які не були завершені
- прогрес виконання діяльностей
- момент, коли робота пішла не по плану
- чи були дотримані пріоритети
- чи були дотримані дедлайни
- як сукупне навантаження співвідноситься із дедлайнами. Підвищення навантаження перед самим дедлайном свідчить про тенденцію відкладати все на останній момент і є явним сигналом про необхідне переосмислення свого планування.

Розділ 3. Розробка планувальника

Стек

Планувальник – це складна динамічна система, інтерфейс якої повинен компактно та зрозуміло відображати велику кількість інформації для користувача, а також забезпечити просто та інтуїтивне додавання, видаляння та редагування цієї інформації.

Крім того, система планувальника повинна бути легкодоступною з різних місць та девайсів, при цьому вся інформація повинна швидко синхронізуватись на кожному з пристроїв. Адже потреба переглядати та редагувати свої плани виникає часто, і користувач не може бути обмежений лише одним пристроєм.

Для цієї мети найкраще підходить використання динамічного веб застосунку, який підключатиметься до центрального сервера. Це дасть можливість швидкої синхронізації та легкого бекапу даних, а також легкої доступності планувальника, оскільки браузер є на кожному пристрої і це не потребує встановлення окремих застосунків.

UI/UX

Для написання складного динамічного інтерфейсу та користувацької частини планувальника, було прийнято рішення використовувати фреймворк React на мові TypeScript.

React - це відкрита JavaScript бібліотека, що використовується для побудови інтерфейсу користувача (UI), зокрема односторінкових динамічних сайтів. Тому використання React для побудови планувальника є логічним вибором. React забезпечує ефективне відображення даних і оновлення їх на стороні клієнта. Його віртуальний DOM (Document Object Model) дозволяє швидко оновлювати сторінку, не потребуючи повного перезавантаження сторінки. Крім того, React дозволяє створювати компоненти, які можна перевикористовувати в інших частинах додатку, що забезпечує зменшення кількості коду і полегшує його розуміння. У цьому

контексті використання React дозволить забезпечити ефективну і масштабовану розробку, а також покращить продуктивність та швидкість відображення веб-сайту для користувачів.

Server

Для написання серверної частини додатку прийнято рішення використовувати платформу Node.js в парі із фреймворком Express.
Node.js є популярною платформою для розробки серверних додатків на JavaScript, що забезпечує високу продуктивність і масштабованість. Використання Node.js для створення сервера динамічного веб-додатку на основі React є логічним вибором, оскільки воно дозволяє використовувати JavaScript як на стороні клієнта, так і на стороні сервера. Це дозволяє використовувати одну мову програмування на всіх рівнях додатку, що забезпечує простоту і зрозумілість розробки. Крім того, Node.js забезпечує швидкий доступ до даних і може обробляти багато запитів одночасно, що забезпечує високу продуктивність серверної частини веб-додатку.

Express – в свою чергу, це популярний фреймворк для створення серверних додатків на Node.js. Використання Express для створення серверу динамічного веб-додатку є логічним вибором, оскільки він забезпечує простоту розробки і високу ефективність для створення швидких і надійних серверів. Express забезпечує широкий набір функціональних можливостей для створення серверу, включаючи маршрутизацію, створення різних видів запитів і відповідей, обробку запитів із завантаженням файлів, підтримку шаблонів та багато іншого. Крім того, Express дозволяє інтегрувати різні модулі і бібліотеки, що дозволяє розширювати можливості серверу, збільшувати його ефективність та забезпечувати його масштабованість.

Data storage

Як уже було сказано, система планувальника потребує швидкого доступу до даних, а також легкої синхронізації. З цих міркувань логічним вибором стало використання serverless бази даних Firestore.

Firestore є гнучкою базою даних, що розроблена Google для збереження і синхронізації даних між додатками, які працюють на різних платформах. Використання Firestore для зберігання даних для веб-додатку є цілком природнім вибором, оскільки вона забезпечує швидкий доступ до даних і має розширену функціональність для роботи з даними. Firestore підтримує такі функції як реальний час, запити у реальному часі та синхронізацію даних між пристроями, що дозволяє забезпечити потокову передачу даних для веб-додатку. Крім того, Firestore забезпечує високий рівень безпеки для даних, що зберігаються в базі даних, забезпечуючи шифрування даних та автентифікацію користувачів. Наостанок, Firestore можна використовути «out of the box», тобто вона не потребує складних додаткових налаштувань і побудови цілої інфраструктури.

Враховуючи всі переваги, які надає Firestore, використання цієї бази даних є логічним вибором для забезпечення легкого, ефективного і безпечного збереження даних для веб-додатку.

Генетична генерація розкладу

При великій кількості завдань кількість вимог які потрібно задовольнити для отримання оптимального розкладу зростає експоненційно. А при великій кількості різних обмежень знайти оптимальну комбінацію планів стає дуже часозатратно.

Для вирішення цієї проблеми в планувальнику реалізовано автоматичну генерацію оптимального розкладу на основі заданих обмежень за допомогою генетичного алгоритму.

Генетичний алгоритм - це еволюційний підхід до пошуку оптимальних рішень. Він працює шляхом застосування принципів природного відбору та еволюції до генерації рішень.

Генетичний алгоритм має значні переваги при застосування для задач планування порівняно із традиційними методами:

1. Генетичні алгоритми можуть знайти оптимальний розклад для великої кількості завдань. З традиційними методами планування часу це може бути дуже складно, оскільки вони швидко стають непрактичними, коли кількість завдань зростає.
2. Генетичні алгоритми є ефективними для знаходження оптимального рішення в умовах невизначеності. Традиційні методи планування часу можуть бути обмежені, якщо є невизначені фактори, які можуть впливати на результат.
3. Генетичні алгоритми легко застосувати задач де наявно багато обмежень. До того ж, їх легко пристосувати до нових видів обмежень. Традиційні методи в такому випадку стають надто складними та жорсткими, їх дуже важко підлаштувати під нові види обмежень. Або ж задача стає настільки складною, що алгоритм пошуку рішення взагалі не відомий.
4. Іншою перевагою генетичних алгоритмів є їхній високий рівень паралелізації. Оскільки кожен ітераційний крок генетичного алгоритму не залежить від результатів попереднього кроку, то можна виконувати багато кроків генетичного алгоритму паралельно.

Ген

Для реалізації генетичного алгоритму нам потрібно закодувати інформацію яку ми будемо оптимізувати у вигляді гену.

Ген – це самостійна цілісна одиниця з якою працює генетичний алгоритм. Ген являє собою набір даних, які будуть оптимізуватись.

Кодування планів та завдань у гени відбувається наступним чином.

Спочатку визначимо яка інформація буде сталою впродовж процесу генерації. В нашому випадку це:

- Кількість діяльностей, для яких генеруються плани
- Мінімальний початок часу для кожного плану, який дорівнюватиме не раніше ніж теперішньому моменту.
- Максимально можлива тривалість для кожного плану

Спираючись на це, наш ген являтиме собою послідовність частин, кожна із яких відповідатиме одній діяльності, для якої генеруємо плани. Такий підхід дозволяє легко дотримуватись обмеження на кількість завдань, а крім того, дуже ефективно реалізувати кросинговер на рівні цих частин. Такий кросинговер гарантуватиме те що кожна із частин копіюється цілісно, не порушуючи своєї локальної оптимальності.

В свою чергу, кожна із цих частин скрадатиметься із довільної кількості підчастин, які відповідатимуть за окремі плани. Кожна підчастина містить два числа, які кодують часовий слот плану та тривалість плану в відносних часових одиницях відповідно. Саме ці числа будуть ціллю процесу мутацій.

Часовий слот – це індекс певної «комірки» часу відносно мінімального часу при генерації. «Комірка» - це певні проміжки часу згідно із рівнями планування. До прикладу це можуть бути окремі години, дні, або тижні.

Відносні часові одиниці – це справжня тривалість, нормована за певним множником відносно рівня планування. До прикладу при генерації планів на день нормально визначати їх тривалість в годинах. Але при генерації планів на місяць одна година може бути надто малою тривалістю. Тоді за допомогою множника ми нормуємо одиниці тривалості до, скажімо, 4 годин. Таким чином одна відносна одиниця відповідатиме 4 годинам часу. Це дозволяє узагальнити алгоритм для різних рівнів планування.

Така конфігурація гену є дуже адаптивною і забезпечує достатню швидкість алгоритму.

Генерація гену

В загальному випадку ген генерується випадково, таким чином, щоби задовольняти базовим обмеженням (максимальна тривалість, мінімальний час і так далі).

Проте в нашому випадку часто виникатимуть ситуації коли ми хочемо покращити існуючу послідовність планів за допомогою генетичного алгоритму. Для цього також реалізується створення гену на основі існуючої послідовності планів, який

слугуватиме відправною точною для генетичного алгоритму. Цей процес є зворотнім до декодування гену.

Алгоритм

Основний алгоритм генетичного планування складається з наступних кроків:

Підготовка:

- Початкова популяція - створення випадкових послідовностей планів, закодованих у вигляді генів, які задовольняють основним обмеженням.
- Оцінка пристосованості - розрахунок пристосованості кожного гену відповідно до обмежень.

Ітерація:

- Відбір - вибір кращих генів для наступного покоління.
- Кросинговер - обмін частинами між двома батьківськими генами для створення нащадків.
- Мутація - випадкове змінення деяких частин генів для додавання різноманітності в популяції.
- Оцінка пристосованості - розрахунок пристосованості кожного нового гену.

На кожній ітерації генетичного алгоритму відбувається процес оптимізації, в результаті якого генеруються нові послідовності планів, які мають кращу пристосованість, тобто вони задовольняють більшій кількості обмежень. З кожним поколінням пристосованість популяції збільшується, оскільки кращі рішення відбираються і продовжують еволюціонувати.

Фітнес

Фітнес-функція є ключовою складовою генетичного алгоритму. Вона оцінює якість кожного потенційного рішення на основі заданих обмежень. Обмеження при цьому можуть бути довільні, головним моментом для обрахунку фітнес функції є правильна числова оцінка штрафу за їх порушення. Таким чином, в нашому випадку

фітнес функція – це функція втрат. Тобто наша задача мінімізувати її. Чим ближче значення до 0 – тим краще план задовольняє обмеженням.

Основні обмеження включають в себе:

- Сукупна тривалість планів повинна дорівнювати оцінці часу самої діяльності.
- Плани повинні знаходитись перед дедлайном.
- Сукупне навантаження на один часовий слот не повинно перевищувати максимальне значення.
- Тривалість плану не повинна перевищувати максимальне значення тривалості
- Плани заблокованих діяльностей повинні бути після всіх планів діяльностей, що їх блокують.

Додаткові обмеження можуть включати в себе:

- Певні слоти, в які може попадати певний план
- Певні інтервали між планами
- Заборона на розміщення певних двох планів поруч

Селекція

Селекція в генетичному алгоритмі полягає у виборі найкращих індивідів з популяції, які будуть схрещені для створення наступного покоління. В нашому випадку селекція вибирає гени із найменшим значенням фітнес функції.

Існують різні методи селекції, найбільш поширеними з яких є:

- Турнірна селекція: з популяції випадковим чином вибираються кілька індивідів, і з них обирається найкращий за значенням фітнесу.
- Рулеткова селекція: індивіди з популяції вибираються випадковим чином з ймовірностю, яка пропорційна їхньому значенню фітнесу. Тобто, чим кращий індивід, тим більші шанси, що він буде обраний для наступного покоління.
- Елітарна селекція: найкращі індивіди з популяції переносяться до наступного покоління без змін. Це дозволяє зберегти найкращі гени та значення фітнесу в популяції та підвищити швидкість збіжності генетичного алгоритму.

В нашому випадку найкращим методом є турнірна селекція, оскільки вона дозволяє зберігати найкращих представників, але й разом з тим не викидати усіх інших, що дозволяє краще зберігати диверсифікацію між поколіннями. Це дозволяє забезпечити рівновагу між розмаїттям популяції та збереженням найкращих рішень. З одного боку, важливо збільшувати розмаїття популяції, щоб уникнути застрягання в локальних максимумах (мінімумах). З іншого боку, важливо зберігати найкращі рішення, щоб забезпечити швидкий збіг до глобального максимуму (мінімуму).

Кросинговер

Кросинговер застосовується для того щоби поєднати різні гени, тим самим збільшуючи розмаїття популяції. При цьому правильне застосування кросинговеру дозволяє отримати нові гени, що задовольняють обмеженням.

В нашому випадку, як згадувалося вище, структура гену була спеціально створена таким чином, щоби забезпечити легкий кросинговер. Тож ми використовуватимемо один із найпростіших його варіантів – одноточковий кросинговер.

Мутація

Мутація застосовується для того щоби створити більш різноманітну популяцію рішень. В процесі мутації дані гену змінюються випадковим чином із певною імовірністю. В нашому випадку імовірність задається як відсоток від частини гену, яка зазнає мутацій. Для того щоби порахувати імовірність для кожної окремої частини, використовується формула:

Імовірність мутації для гену / довжину гену

Такий підхід дає змогу отримувати гени, що знаходяться набагато далі від початкового, тим самим це дозволяє «вистрибувати» із локальних мінімумів і знаходити кращі рішення.

Розумне планування

Великою проблемою в плануванні є потреба в постійному записі нових та актуалізації існуючих завдань та планів. Створення потужного інструменту із

зручним інтерфейсом частково вирішує цю проблему. Проте потреба постійно вручну вводити нову інформацію про завдання і плани та їх відповідні властивості може бути доволі марудною роботою. При цьому чим потужніший стає інструмент, тим більшої кількості контекстуальної інформації він розуміє та потребує. Це демотивує та відштовхує користувача.

З цієї причини було вирішено створити «розумний» планувальник, який здатен розуміти користувача на рівні звичайної мови і навіть голосу, після чого аналізувати цю інформацію та робити відповідні дії.

Оскільки дана задача передбачає обширну роботу із неструктурованою текстовою інформацієї від користувача, то для її вирішення оптимальним вибором є використання великих мовних моделей, таких як gpt-3.5-turbo від OpenAI для парсингу, обробки та класифікації тексту.

OpenAI gpt-3.5-turbo є однією із найпотужніших великих мовних моделей на даний момент, здатною виконувати розуміння природної мови на високому рівні, включаючи здатність розуміти контекст та виконувати аналіз сентименту.

Парсинг тексту це складний процес, який може включати в себе кілька комплексних етапів. Проте саме OpenAI API включає в себе лише базові функції, тож велику кількість логіки потрібно надбудовувати самостійно. Але на допомогу тут приходить популярна бібілотека LangChain.

LangChain дозволяє виконувати різноманітні завдання, пов'язані з обробкою тексту, такі як токенізація, частиномовний аналіз, лематизація, тематичний аналіз та інше. Крім того вона значно спрощує взаємодію із різними API та допомагає поєднувати різні додаткові інструменти, такі як пошук тексту у гугл, обчислення формул, пошук тексту за схожістю у власних векторних сховищах та інші. Це значно зменшує час та зусилля, необхідні для розробки власних алгоритмів обробки тексту, а також забезпечує значно потужніші можливості та кращі набагато результати. Крім того, бібліотека має зручний та простий API, що позбавляє необхідності вивчення API кожного окремого інструменту та його нюансів.

В контексті планувальника нашим завданням є перетворення тексту користувача на список завдань та планів, із коректним відображенням у моделі всіх згаданих в тексті вимог. Вирішення цього завдання відбувається в три етапи:

- Отримання та попередня обробка тексту
- Парсинг і структуризація тексту
- Класифікація отриманих планів

Отримання та попередня обробка тексту

В нашому випадку користувач буде диктувати або вводити розповідь про свої завдання і плани.

У випадку продиктованої розповіді, нам потрібно перетворити її у текст для подальшої обробки. В нашому випадку для цього використовується ще одна велика мовна модель для обробки голосу від OpenAI – whisper. Вона чудово справляється із обробкою української мови і має похибку менше 8%, що є дуже високим показником.

Сама передобробка тексту може включати в себе очищення тексту від небажаних символів, нормалізацію тексту (наприклад, переведення тексту в нижній регістр), а також розбиття тексту на речення або фрази.

Парсинг і структуризація тексту

Метою парсингу в нашому випадку є виділення окремих завдань та планів, які мають бути виконані, відповідно до контексту або інтенцій, виявлених у тексті.

Для вирішення цієї задачі у випадку використання великих мовних моделей дуже важливим є генерація правильного текстового запиту, оскільки саме від того, як коректно сформульовано запит, залежить якість та точність отриманої відповіді.

Крім того, дуже важливо надати релевантні приклади, як має бути отримана і оформлена відповідь. Правильно підібрані приклади різних випадків заміняють об’ємні пояснення і в загальному дають найкращі результати. Це зменшує розмір

запиту, а відповідно пришвидшує генерацію відповіді та зменшує її грошову вартість.

В загальному, запит який використовується для парсингу тексту за допомогою великої мовної моделі, викладає ось так (приклади не включено):

```
You will receive a text from user. Convert it to a list of tasks.

The output should be formatted as JSON in the following schema:
[
    {
        title: string // Short and descriptive title. Split into separate tasks if conjunction is present.
        date: string | null // Calculate dates relative to today's date in format dd-mm-yyyy. Put null if date is not mentioned.
        status: boolean // true if task is in the past tense, false otherwise.
    }
]

Do not output anything except for the extracted information. Do not add any clarifying information. Do not add any fields that are not in the schema.
All output must be in valid JSON format and follow the schema specified above. Wrap the JSON in <json> tags.

EXAMPLE:
{example}
```

Класифікація отриманих планів

Після того як текст був оброблений і ми витягли із нього плани користувача в потрібному вигляді, нам потрібно встановити, яких існуючих діяльностей вони стосуються. Або ж, якщо таких немає, створити нові діяльності для відповідних планів.

Для того щоби знайти цю відповідність, ми знову використовуємо велику мовну модель та з її допомогою проводимо семантичний пошук, співставляючи підписи планів та назви існуючих діяльностей. Це дозволяє добитись найкращих результатів та нівелює проблеми із неточностями, використанням синонімів чи

перефразуваннями. Це є дуже важливим аспектом, оскільки при встановленні відповідностей за розповіддю користувача таких проблем буде дуже багато.

Проте тут виникає проблема – діяльностей може бути дуже велика кількість і ми не можемо співставляти плани із кожною з них. Адже, по перше, це буде нерозумно, оскільки більшість із них будуть навіть близько нерелевантні. По друге це неефективно і дорого, та значно збільшує навантаження на модель і мережу, через кількість тексту який потрібно передавати та обробляти. А по третє, це просто технічно неможливо, оскільки мовна модель gpt-3.5-turbo здатна розуміти обмежений контекст максимальним обсягом в 3000 токенів, що дорівнює приблизно 1000 слів.

Тому для вирішення цієї проблеми в хід вступають векторні бази даних (vectorstore) та вбудовування векторних функцій (embeddings).

Вбудовування генеруються моделями штучного інтелекту (такими як великі мовні моделі) і мають велику кількість атрибутів або функцій, що ускладнює керування їх представленням. У контексті штучного інтелекту та машинного навчання ці функції представляють різні виміри даних, які є важливими для розуміння шаблонів, взаємозв’язків і базових структур [11].

Для обробки даних такого типу потрібні спеціальні бази даних. Векторні бази даних задовольняють цю вимогу, пропонуючи оптимізоване зберігання та можливості запитів для вбудовування. Вони навмисно розроблені для роботи з цим типом даних і пропонують продуктивність, масштабованість і гнучкість, необхідні для максимального використання даних [11].

В нашому випадку ми обмежилися використанням простого векторного сховища HNSWLib, яке має базовий функціонал, зберігається локально і не потребує додаткових налаштувань.

Отже, для того щоби ефективно класифікувати плани, ми робимо наступні кроки:

- За допомогою пошуку по схожості у векторній базі даних шукаємо релевантні діяльності для кожного плану

- На основі найкращих результатів формуємо відповідний запит для великої мовної моделі
- Отриману відповідь обробляємо як JSON та витягаємо і співставляємо потрібну інформацію.

Пошук по схожості

Для кожного плану за допомогою пошуку по схожості знаходимо релевантні діяльності по їхньому опису. Для того щоби зробити це максимально якісно, інформація яка вбудована у векторному сховищі має бути заздалегідь правильно підготовлена. Будь які абревіатури чи скорочення в описах діяльностей мають бути розшифровані, а текстові записи чисел переведені в цифри. Крім того використання синонімів в ключових словах покращує якість пошуку. Це дуже важливо, адже від результатів пошуку за схожістю тексту залежить результат подальшої класифікації.

Формування запиту

Наступним етапом ми формуємо запит до мовної моделі, базуючись на самому плані та певній кількості найкращих знайдених відповідностей на попередньому етапі. В запиті ми також просимо в мовної моделі надавати показник впевненості у відповіді, та пояснення в цілях дебагу відповідей. Запит який використовується в планувальник виглядає наступним чином:

```
You will be given a text and a list of records from user. Find the best corresponding match of text on the list of records.

The output should be formatted as JSON in the following schema:
{
index: number // An index of the matched record line starting from 0. Each record will be on the separate line.
confidence: number // A number 0-5 of how close you think the match is. Put 0 if it seems like there is no match.
explanation: string // Explanation why you think this is the best match.
}

Do not output anything except for the extracted information. Do not add any clarifying information. Do not add any fields that are not in the schema.
```

```
All output must be in JSON format and follow the schema specified above. Wrap the JSON in <json> tags.
```

Обробка відповіді

Отримавши відповідь у форматі JSON, ми співставляємо план із діяльністю за вказаним індексом. На цьому етапі класифікацію можна вважати завершеною. У випадку якщо показник впевненості надто низький, ми можемо просигналізувати про це користувачу для необхідності подальших уточнень.

MVP

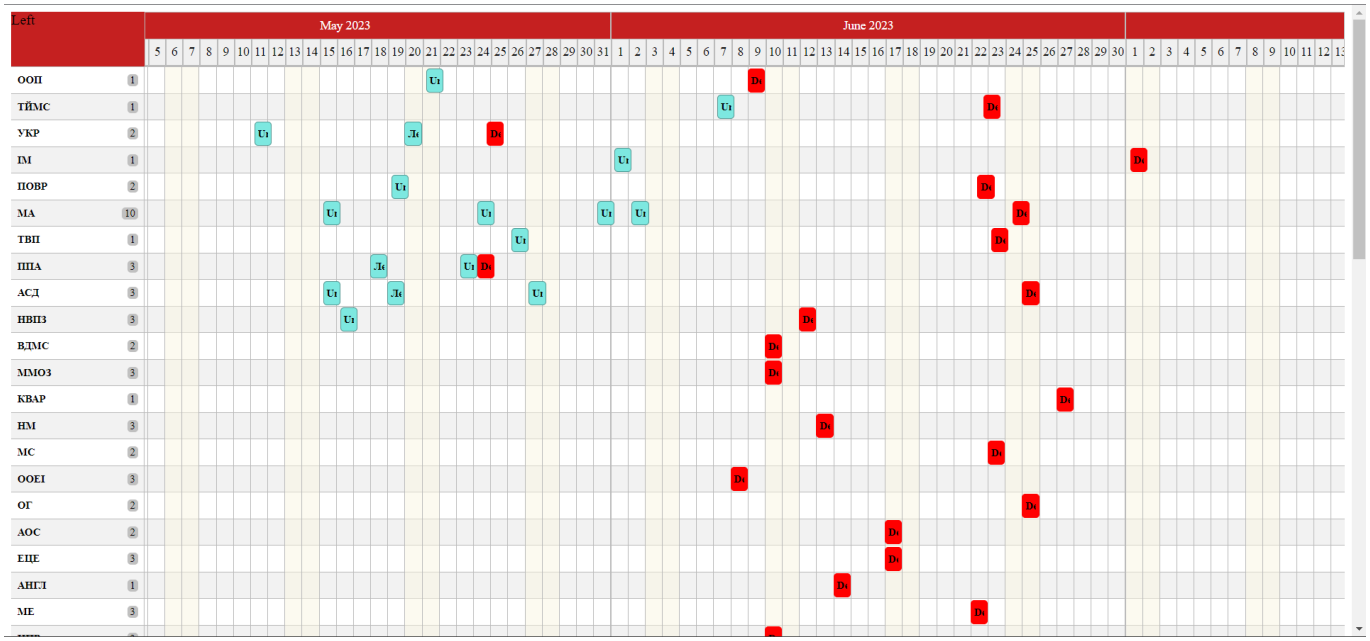
Реалізація планувальника мала за собою на меті перевірити роботоздатність моделі та практичність такого інструменту. Для цього було створено MVP, який включає в себе мінімальний функціонал.

Реалізований функціонал

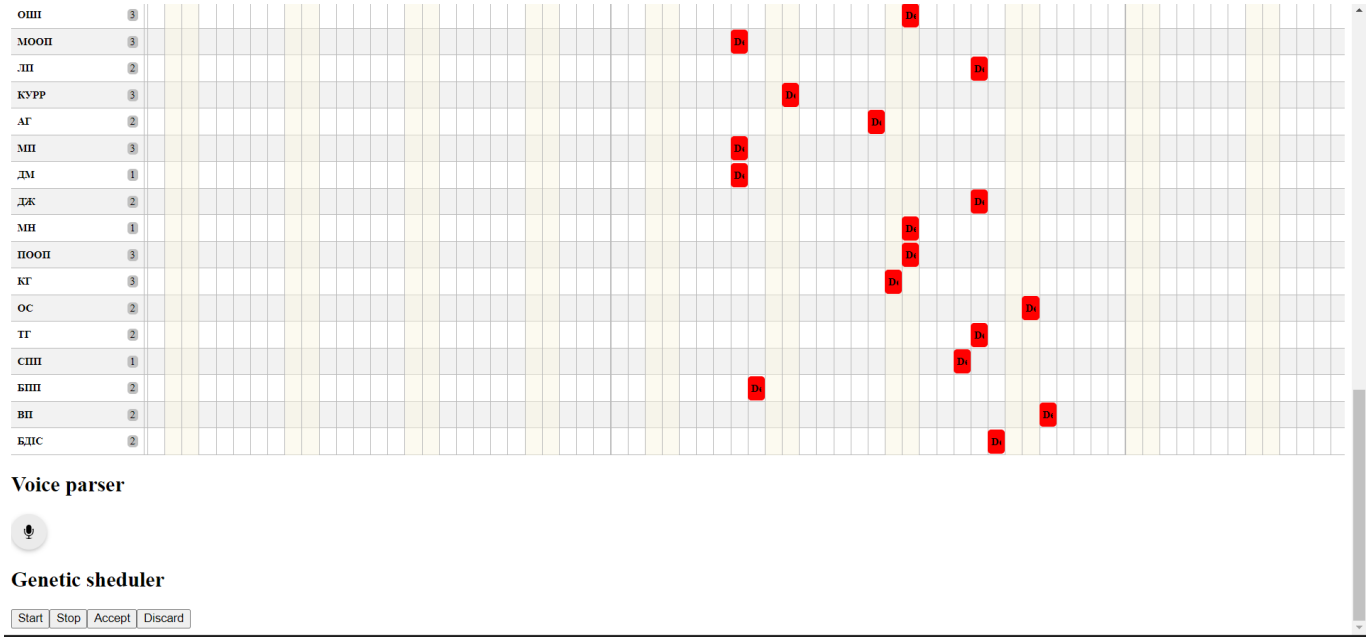
- Завдання, без ієрархічної структури та відношень.
- Плани, без поділу на рівні планування
- Візуальне подання у вигляді списку завдань і таймлайну планів
- Додавання, редагування та видалення планів і завдань
- Збереження інформації у базі даних
- Перетворення голосу і тексту в список планів
- Класифікація планів по існуючим завданням
- Генетична генерація оптимального розкладу

Інтерфейс

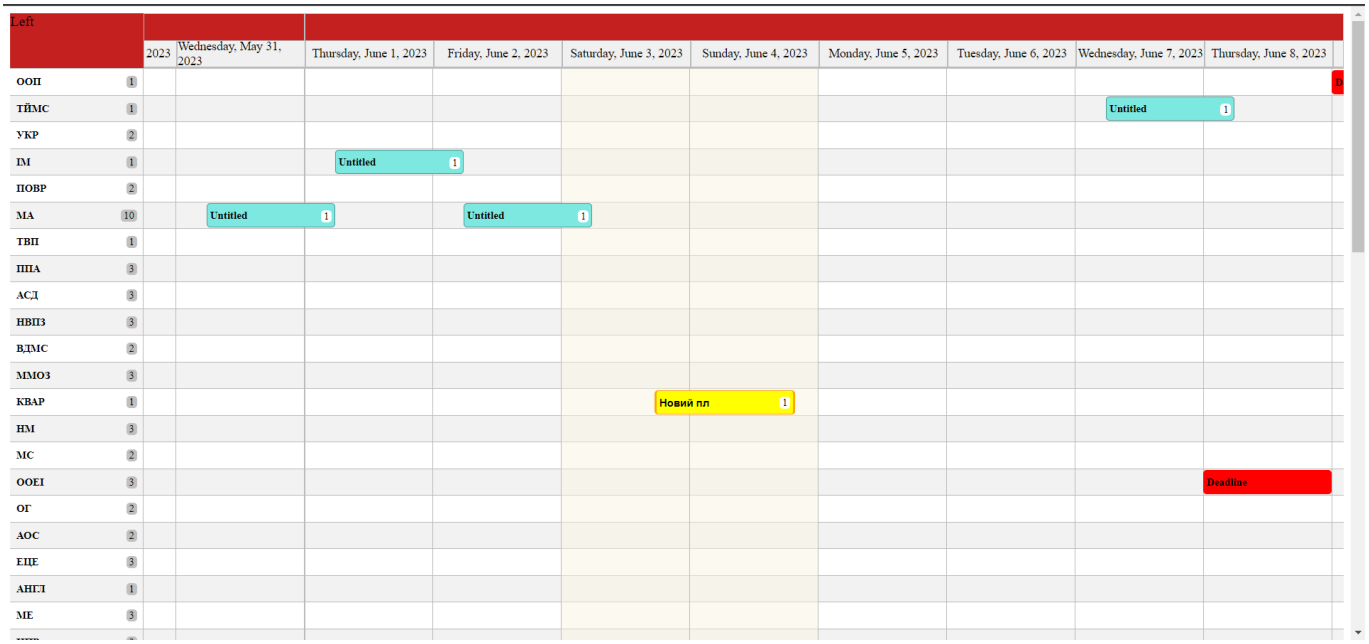
Загальний вигляд



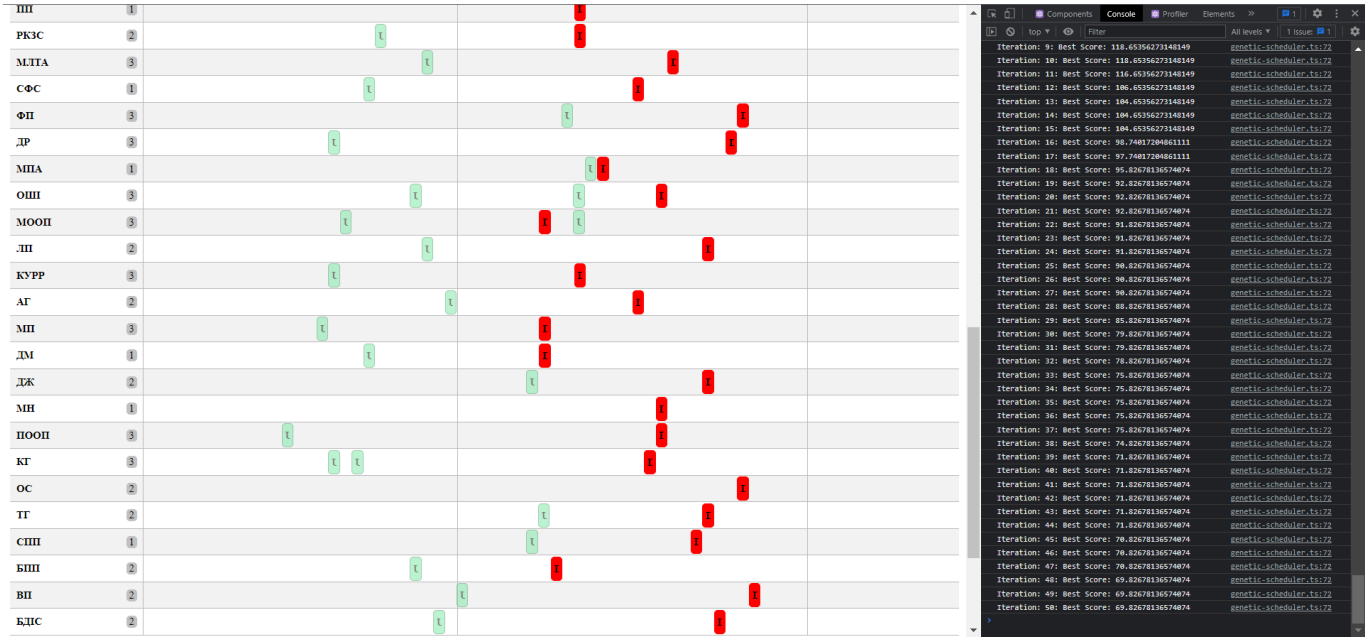
Інструменти



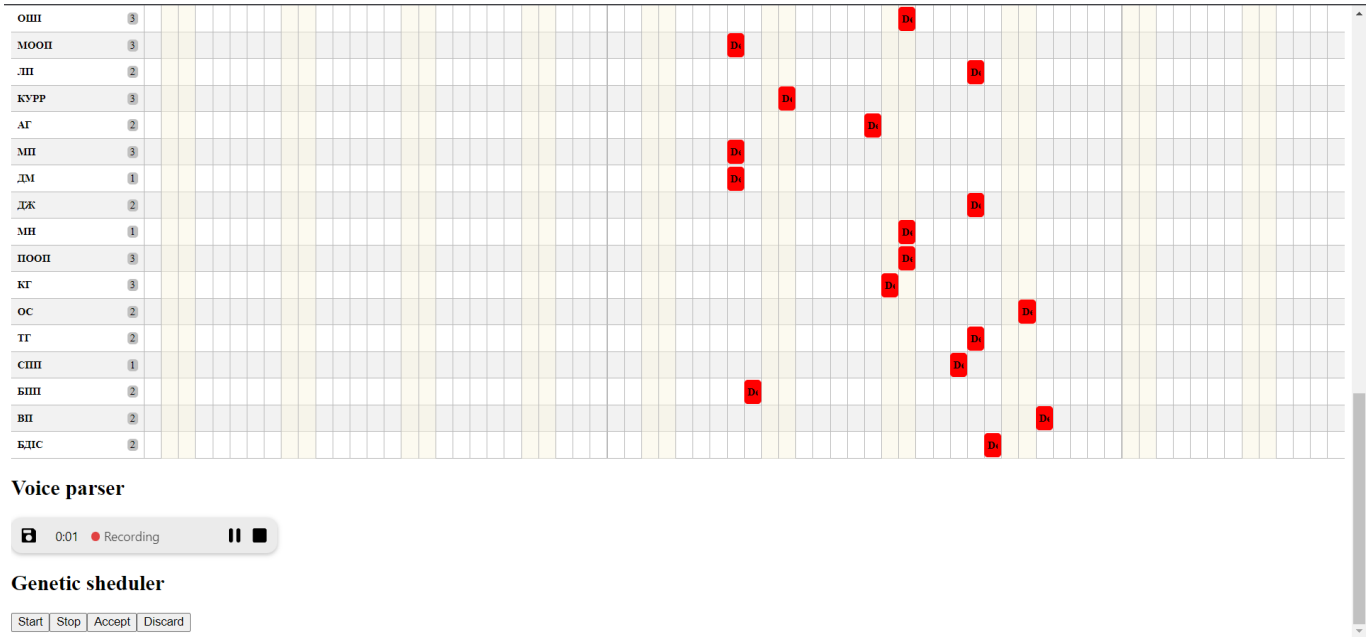
Редагування планів



Генерація планів



Диктування планів



Оцінка та аналіз результатів

Завдання і плани

Тестування показало, що дана модель досить вправно справляється із різними потребами планування і здатна якісно передати інтенції користувача. Проте важко зробити повноцінні висновки, оскільки в теперішній реалізації планувальник є малопрактичним. Причиною є те, що йому бракує багатьох описаних вище можливостей, зокрема структуризація діяльностей та рівні планування. Тож, в цьому контексті, теперішня реалізація потребує подальшої доробки.

Генетичний алгоритм

Використання генетичного алгоритму показало потенціал. Він здатен легко справитись із такими обмеженнями на оцінку часу, дедлайн та максимальне навантаження, та за кілька секунд згенерувати оптимальний розклад, або ж найкращий можливий варіант, якщо оптимальної комбінації не існує. Це свідчить про те, що введення нових видів обмежень не складе для нього проблеми задовольнити їх.

Проте одним із помічених недоліків є те, що фінальний розклад виглядає надто хаотично і може дуже сильно вар’юватись. Для вирішення цієї проблеми потрібно

ввести нове обмеження, яке ставитиме плани якомога раніше до дедлайну та робитиме їх більш скупченими або систематизованими для однієї діяльності, або ж плануватиме діяльності більш впорядковано.

Крім цього, корисним також буде реалізація системи динамічних параметрів генетичного алгоритму, яка збільшуватиме його температуру при наближенні до глобального мінімуму. Це дозволить не застрягати в локальних мінімумах і швидше приходити до оптимального результату, коли кількість обмежень, які залишається задовольнити, стає дуже малою.

Великі мовні моделі

Використання великої мовної моделі gpt-3.5-turbo для парсингу розповіді користувача показало себе вкрай добре. Реалізована система здатна правильно розпізнати і класифікувати згадані користувачем завдання та плани. І це справді значно зменшує кількість потрібної від користувача роботи - йому достатньо просто розповісти про це і плани самі з’являться на екрані, після чого залишається лиш внести корективи за необхідності.

Один із помічених недоліків являє собою достатньо довгу затримку при обробці тексту, яка коливається між 5 та 20 секундами. Наполовину ця проблема пов’язана із поганою реалізацією операцій оновлення та видалення на стороні сервера, через брак х підтримки в бібліотеці HNSWLib. Проте це можна легко вирішити, використавши більш потужне рішення, до прикладу Pinecone. Інша частина проблеми полягає в довгому очікуванні відповіді від великої мовної моделі. Проте це теж можна нівелювати, якщо використати стрімінг та відображати результати поступово, по мірі надходження відповідей.

Висновки

1. Визначення завдань при плануванні: Після дослідження різних аспектів процесу планування, ми виявили низку ключових завдань, із яких складається ефективний процес планування. Ці завдання можна генералізувати як: постановку задач, що в ключає в себе їх визначення, розбиття та пріоритезацію; планування часу, що являє собою ітеративний розподіл ресурсів; та аналіз результатів, який дає змогу виправити помилки і досягти найвищої продуктивності.

2. Визначення функціоналу планувальника: Ми змогли визначити необхідний функціонал планувальника, що відповідає визначеним завданням планування. Він включає в себе, але не обмежується: дерево підзавдань, розподіл часу по завданнях за допомогою планів, рівні планування, зручний перегляд планів у вигляді таймлайну, генетична генерація оптимального розкладу, розумне перетворення тексту у список задач, та відслідковування свого прогресу.

3. Визначення і формалізація сутностей планувальника: Ми змогли успішно розділити інформацію, необхідну для вирішення визначених задач планувальника, на три фундаментальні сутності планувальника: діяльності, плани та логи. Це дало нам зрозуміти основну структуру та відношення, які повинен використовувати планувальник.

4. Розробка мінімальної версії планувальника: Ми розробили прототип планувальника, який включає в себе найбільш базовішу частину раніше визначеного функціоналу. Цей прототип демонструє роботоздатність даної системи та те, наскільки вона відповідає очікуванням. Також ми зрозуміли першочергові аспекти та недоліки, що потребують допрацювання.

5. Тестування і аналіз перспектив розробленого планувальника: Прототип планувальника було протестовано у персональному використанні. За результатами тестування, планувальник виявився досить зручним для сприйняття і відображення інформації, проте малопрактичним в теперішній реалізації. Тим не менш, ми дійшли висновку, що розроблена модель завдань і планів є роботоздатною і зрозумілою для

користувача. Тому подальша розробка та удосконалення системи має великі перспективи. Крім того, розроблений прототип допоміг нам визначити першочергові напрямки для подальшого удосконалення планувальника.

Список використаних джерел

1. A case study: using genetic algorithm for job scheduling problem / B. Taˇgtekin et al. *arXiv.org e-Print archive*. URL: https://arxiv.org/pdf/2106.04854.pdf.
2. Alyami A. Impact of time-management on the student's academic performance: a cross-sectional study. *SCIRP Open Access*. URL: https://www.scirp.org/journal/paperinformation.aspx?paperid=107573.
3. Claessens B. J. C. A review of time management literature. *ResearchGate*. URL: https://www.researchgate.net/publication/228664480_A_Review_of_Time_Management_Literature.
4. Contributors to Wikimedia projects. Planning fallacy. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/Planning_fallacy.
5. Firebase documentation. *Firebase*. URL: https://firebase.google.com/docs.
6. GIAN - MHRD, IIT Kharagpur. Scheduling genetic algorithm, 2020. *YouTube*. URL: https://www.youtube.com/watch?v=oHMuJzCZjek.
7. GitHub - namespace-ee/react-calendar-timeline: a modern and responsive react timeline component. *GitHub*. URL: https://github.com/namespace-ee/react-calendar-timeline.
8. GitHub - nmslib/hnswlib: Header-only C++/python library for fast approximate nearest neighbors. *GitHub*. URL: https://github.com/nmslib/hnswlib.
9. OpenAI API. *OpenAI API*. URL: https://platform.openai.com/docs.
10. Welcome to LangChain. *Langchain*. URL: https://js.langchain.com/docs/.
11. What is a vector database?. *Pinecone*. URL: https://www.pinecone.io/learn/vector-database/.