

МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”
КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КУРСОВА РОБОТА

з дисципліни «Об’єктно-орієнтоване програмування»

На тему:

“Магазин Автомобілів”

Студента групи ПЗ-22

спеціальності 6.121

“Інженерія програмного забезпечення”

Солтисюка Д.А.

Керівник: доцент кафедри ПЗ,

к.т.н., доцент Коротєєва Т.О.

Національна шкала _____

Кількість балів___Оцінка ECTS___

Члени комісії

_____	_____
_____	_____
_____	_____

Львів 2022

Зміст

Технічне завдання

Алгоритм роботи програми

Діаграми UML

Код програми

Протокол роботи програми

Опис та обробка виняткових ситуацій

Інструкція користувача

Висновки

Технічне завдання

Розробити програму засобами ООП згідно вказаного варіанту (Варіант №22 (ПЗ-22)). Передбачити віконний режим роботи програми та інтерфейс користувача.

Передбачити ввід даних у двох режимах:

- З клавіатури;
- З файлу;

Передбачити у програмі виняткові ситуації. Продемонструвати викладачу роботу розробленої програми. Сформувати звіт курсової роботи обсягом не менше 20 сторінок.

Завдання варіанту №22:


22. Створити таблицю у візуальному середовищі

Марка авто | Тип (легковий/вантажівка) | Потужність | Розхід палива | Об'єм паливного баку.

- 1) Алгоритмом злиття відсортувати записи за маркою авто
- 2) Знайти всі марки авто, в яких є легкові та вантажні автомобілі.
- 3) Визначити найпотужніший легковий автомобіль та найменш потужну вантажівку
- 4) Вивести всі автомобілі, які без дозаправки можуть проїхати найбільшу відстань, згрупувати їх за типом.
- 5) Відсортувати записи за розходом палива та згрупувати за типом.
- 6) Для кожної марки автомобіля визначити найбільший та найменший об'єм паливного баку

Для класу створити: 1) **Конструктор** за замовчуванням; 2) **Конструктор** з параметрами; 3) **конструктор** копій; 4) перевизначити операції >>, << для зчитування та запису у файл. Для демонстрації роботи програми використати засоби візуального середовища програмування.

№ з/п	Зміст завдання	Дата
1	Здійснити аналітичний огляд літератури за заданою темою та обґрунтувати вибір інструментальних засобів реалізації.	28.10.22
2	Побудова UML діаграм	12.11.22
3	Розробка алгоритмів реалізації	13.11.22
4	Реалізація завдання (кодування)	15.11.22
5	Формування інструкції користувача	17.11.22
6	Оформлення звіту до курсової роботи згідно з вимогами Міжнародних стандартів, дотримуючись такої структури: <ul style="list-style-type: none"> - зміст; - алгоритм розв'язку задачі у покроковому представленні; - діаграми UML класів, прецедентів, послідовності виконання; - код розробленої програми з коментарями; - протокол роботи програми для кожного пункту завдання - інструкція користувача та системні вимоги; - опис виняткових ситуацій; - структура файлу вхідних даних; - висновки; - список використаних джерел. 	18.11.22

Завдання прийнято до виконання:  (Солтисюк Д.А.)

22.08.22 (підпис студента)

Керівник роботи: _____ / Коротєєва Т .О./

1. Загальні положення:

Найменування: Магазин Автомобілів.

Умове позначення: Магазин Автомобілів.

Замовник: кафедра ПЗ

Розробник: Солтисюк Д.А.

Початок робіт: 28.10.2022 р.

Закінчення робіт: 18.11.2022 р.

2. Призначення системи:

Метою створення програми є полегшення збереження та зчитування інформації про машини. Програма вирішує проблему підтримки інформаційних реєстрів, а також дозволяє швидко навігуватись у списку завдяки грамотним сортуванням та фільтруванню. Інформація може зберігатися на носій, щоб використовуватися в майбутньому, або передаватися мережою.

3. Об'єкти даних:

Програма опрацьовує дані про автомобілі, а саме: бренд авто, модель, тип авто, потужність, розхід палива, об'єм паливного баку. Ці дані зберігаються в оперативній пам'яті, а після експорту даних – на фізичному носії.

4. Вимоги до програмного забезпечення:

4.1. На пристрої має бути встановлений браузер та активне підключення до мережі Інтернет. Програма використовує браузерне середовище, остання оновлення якого відбулося не більше ніж 5 років тому.

4.2. *Функціональні вимоги:*

R1. Відображення існуючих автомобілів.

R2. Додавання нової інформації про автомобілі до реєстру.

R3. Імпорт автомобілів з фізичного носія.

R4. Експорт поточного інформаційного реєстру.

R5. Перевірка правильності введеної інформації при введенні у систему.

R6. Розширені можливості фільтрування інформаційного реєстру.

Вимоги до апаратного та програмного забезпечення:

Операційна система: MacOS/Windows xp+/Android/IOS/Linux.

Мінімальний обсяг ОЗП: 0.5Gb.

Мінімально необхідний простір на диску: 32 Мб.

Процесор: 32-розрядний з мінімальною тактовою частотою 1,5 ГГц.

Монітор: мінімальна роздільна здатність 600x400..

Периферійні пристрої: клавіатура та миша.

4.3. *Інші вимоги:*

Вимоги до надійності:

- Повідомлення при введенні неправильних даних у формі.
- Повідомлення, якщо імпортовані дані пошкоджені або мають неправильний формат.
- Повідомлення, якщо експорт даних не можливий, з можливими вказаними причинами.

5. **Стадії розробки:**

Аналіз та специфікація вимог — збір та аналіз вимог замовника до ПЗ та планування якості продукту.

Проектування — визначення структури й поведінки системи та інтерфейсу користувача.

Кодування — розроблення вихідного коду.

Тестування — перевірка програми на наявність помилок та відповідність вимогам, зазначеним у цьому технічному завданні.

Експлуатація — використання програми користувачами.

Супровід — виправлення помилок в роботі програми.

6. Вимоги до технічної документації:

- Діаграми класів UML.
- Технічне завдання.
- Текст програми.
- Інструкція з експлуатації.

Алгоритми роботи програми

Алгоритм OCE - Obtain Capacity Extrema

Знайти найбільш та найменш потужний автомобіль

OCE1 Створення вихідного масиву з двома елементами з вхідного масиву.

OCE2 Ітерація по вхідному масиву, порівняння з елементами з вихідного масиву.

OCE3 Запис у вихідний масив, якщо є локальним екстремумом.

OCE4 Виведення вихідного масиву в статистичний список під таблицею.

Алгоритм QBVB - Query both-vehicle-type brands

Знайти бренди, що мають автомобілі обох типів

QBVB1 Створення пустого вихідного та проміжного масиву.

QBVB2 Ітерація по вхідному масиву.

QBVB3 Створення запису про наявність того чи іншого типу автомобілю в проміжному масиві.

QBVB4 Ітерація по проміжному масиву.

QBVB5 Додавання в вихідний масив брендів які мають обидва типи автомобілів присутні.

QBVB6 Виведення вихідного масиву в статистичний список під таблицею.

Алгоритм GBFCE - Get Brand Fuel Consumption Extrema

Знайти мінімальний та максимальний розхід палива для кожного бренду

GBFCE1 Створення пустого вихідного масиву.

GBFCE2 Ітерація по вхідному масиву.

GBFCE3 Додавання бренду до вихідного масиву, якщо його немає.

GBFCE4 Порівняння ітератора з записаним мінімумом та максимумом.

GBFCE5 Перезаписати мінімум чи максимум якщо елемент менший чи більший відповідно.

QBVB6 Виведення вихідного масиву в статистичний список під таблицею.

Алгоритм SBLP - Sort by Longest Path

Сортувати автомобілі за найдовшим шляхом, який вони можуть пройти

SBLP1 Ітерація по вхідному масиву починаючи з другого елементу.

SBLP2 Обчислювання можливого найдовшого шляху, який може пройти автомобіль.

SBLP3 Порівняння обчисленого значення з минулим та перестановка елементів.

SBLP4 Виведення вихідного масиву в статистичний список під таблицею.

Алгоритм SBFC - Sort by Fuel Consumption

Сортувати автомобілі по розходу палива

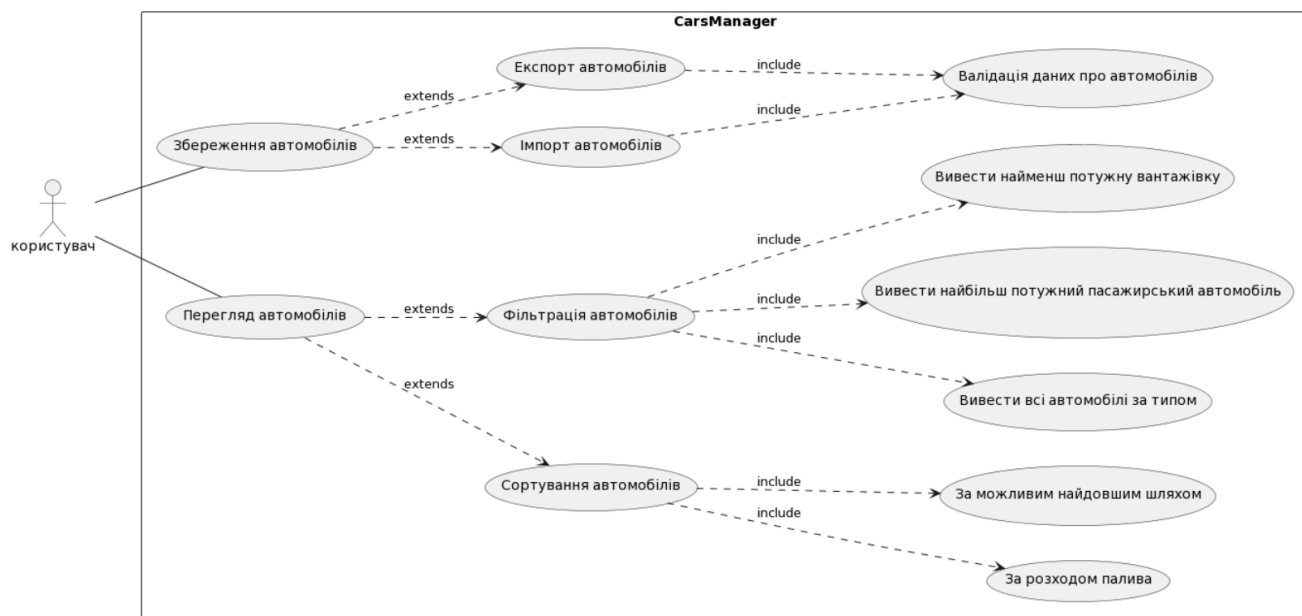
SBFC1 Ітерація по вхідному масиву починаючи з другого елементу.

SBFC2 Порівняння значення розходу палива з минулим та перестановка елементів.

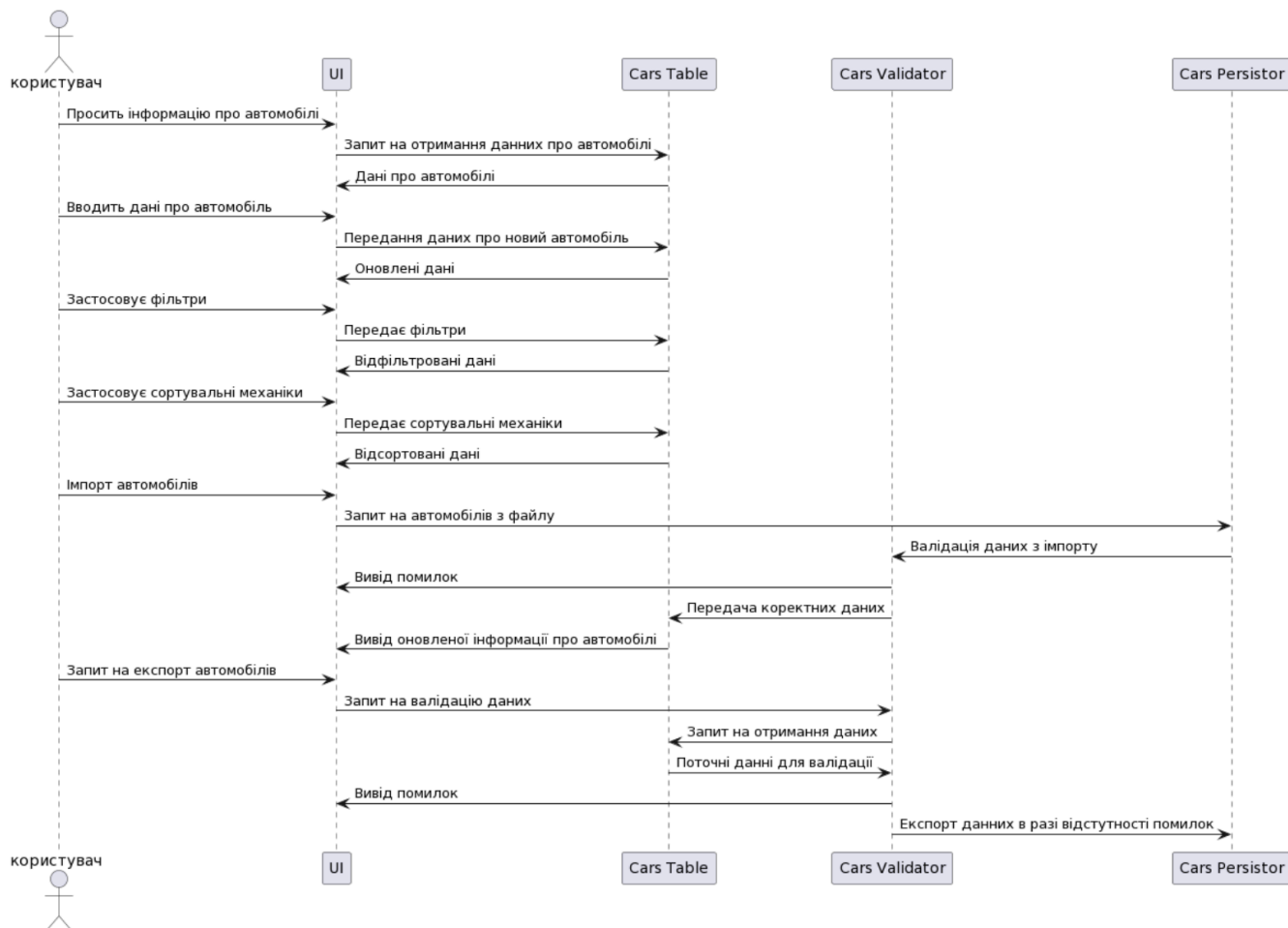
SBFC3 Виведення вихідного масиву в статистичний список під таблицею.

Діаграми UML

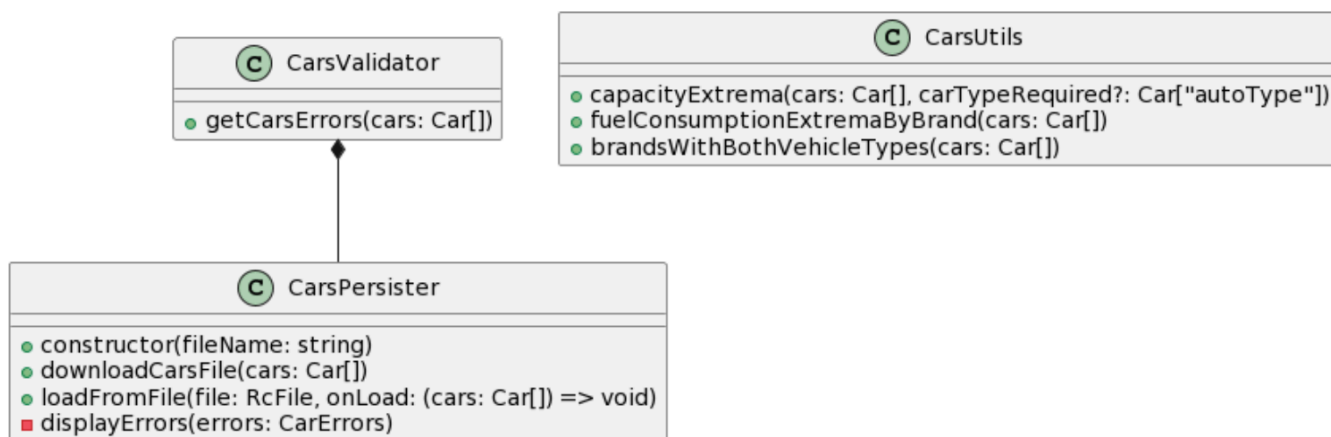
Діаграма прецедентів



Діаграма послідовностей



Діаграма класів



Код програми

Ініціалізатори:

- _app.tsx:

```
import "antd/dist/antd.css"
import type { AppProps } from "next/app"
import "../styles/Home.css"
import "../styles/global.css"

function MyApp({ Component, pageProps }: AppProps) {
  return <Component {...pageProps} />
}

export default MyApp
```

- index.tsx:

```
import { PageHeader } from "antd"
import type { NextPage } from "next"
import Head from "next/head"
import { useEffect } from "react"
import { CarsTable } from "../components/CarsTable"

const Home: NextPage = () => {
  useEffect(() => {
    // prompt before closing the tab
    window.onbeforeunload = () => ""
  }, [])

  return (
    <div className="site-page-header-ghost-wrapper">
      <Head>
        <title>Cars Manager</title>
        <meta name="description" content="Your personal car info manager" />
        <link rel="icon" href="/favicon.ico" />
      </Head>

      <PageHeader ghost={false} title="Cars Manager">
        <main>
          <CarsTable />
        </main>
      </PageHeader>
    </div>
  )
}

export default Home
```

Моделі інформації:

- schema.ts:

```
import Ajv, { JTDDDataType } from "ajv/dist/jtd"
import { capitalize } from "."

const carSchema = {
  elements: {
    properties: {
      key: { type: "int32" },
      brand: { type: "string" },
      model: { type: "string" },
      autoType: { enum: ["Passenger car", "Lorry"] },
      capacity: { type: "uint32" },
      fuelConsumption: { type: "uint32" },
      fuelTankVolume: { type: "uint32" },
    },
    optionalProperties: {},
  },
} as const

export type Car = JTDDDataType<typeof carSchema>[number]

export type CarsValidatorReturnType = Array<{
  message: string
  meta: string
}>

export const customCarValidators: Partial<
  Record<keyof Car, (s: Car) => string[]>
> = {} as const

const ajv = new Ajv({ allErrors: true })
export const carsSerializer = ajv.compileSerializer<Car[]>(carSchema)
const _compiledValidator = ajv.compile<Car[]>(carSchema)

export const carsValidator = (cars: Car[]): CarsValidatorReturnType => {
  _compiledValidator(cars)

  const errors: CarsValidatorReturnType =
    _compiledValidator.errors?.map((e) => ({
      message: e.message
      ? `${capitalize(e.keyword)} error: ${capitalize(e.message)}`
      : "Error",
      meta: `at ${e.instancePath}`,
    })) || []

  cars.forEach((car) =>
    Object.entries(customCarValidators).map(
      ([propertyName, propertyValidator]) => {
        errors.push(
          ...propertyValidator(car).map((e) => ({
            message: e,
```

```

        meta: `at ${propertyName} of ${JSON.stringify(car)}`,
      )))
    )
  }
)
)

return errors
}

```

Класи-утиліти:

- utils.ts:

```

import { RcFile } from "antd/lib/upload"
import {
  Car,
  carsSerializer,
  carsValidator,
  CarsValidatorReturnType,
} from "./schemas"
import { message, notification } from "antd"

export const capitalize = (s: any) => {
  if (typeof s !== "string") return ""
  return s.charAt(0).toUpperCase() + s.slice(1)
}

export const MIN = 0
export const MAX = 1

type FuelConsumptionExtrema = {
  [brand: string]: [number, number]
}

export class CarsUtils {
  static capacityExtrema(
    cars: Car[],
    carTypeRequired?: Car["autoType"]
  ): [Car, Car] {
    if (carTypeRequired) {
      cars = cars.filter((car) => car.autoType === carTypeRequired)
    }

    const extrema = cars.slice(1, -1).reduce(
      (acc: [Car, Car], curr) => {
        acc[0] =
          acc[0] === undefined || curr.capacity < acc[0].capacity
            ? curr
            : acc[0]
        acc[1] =
          acc[1] === undefined || curr.capacity > acc[1].capacity
            ? curr

```

```

        : acc[1]

        return acc
    },
    [cars[0], cars[cars.length - 1]]
)

return extrema
}

static fuelConsumptionExtremaByBrand(cars: Car[]): FuelConsumptionExtrema {
    const brandsFuelConsumption: FuelConsumptionExtrema = {}

    for (let car of cars) {
        if (!brandsFuelConsumption[car.brand]) {
            brandsFuelConsumption[car.brand] = [
                Number.POSITIVE_INFINITY,
                Number.NEGATIVE_INFINITY,
            ]
        }

        const [min, max] = brandsFuelConsumption[car.brand]
        brandsFuelConsumption[car.brand][MIN] =
            car.fuelConsumption < min ? car.fuelConsumption : min
        brandsFuelConsumption[car.brand][MAX] =
            car.fuelConsumption > max ? car.fuelConsumption : max
    }

    return brandsFuelConsumption
}

static brandsWithBothVehicleTypes(cars: Car[]): string[] {
    const PASSANGER_CARS = 0
    const LORRIES = 1

    const proxy: { [x in Car["autoType"]]: number } = {
        Lorry: LORRIES,
        "Passenger car": PASSANGER_CARS,
    }

    const intermediateBrandsArr: { [brand: string]: [boolean, boolean] } = {}

    for (let car of cars) {
        if (!intermediateBrandsArr[car.brand]) {
            intermediateBrandsArr[car.brand] = [false, false]
        }

        intermediateBrandsArr[car.brand][proxy[car.autoType]] = true
    }

    return Object.entries(intermediateBrandsArr)
        .map(([brand, autoTypes]) => {
            if (autoTypes[PASSANGER_CARS] && autoTypes[LORRIES]) {
                return brand
            }
        })
}

```

```

        }
    })
    .filter(Boolean) as string[]
}
}

class CarsPersister {
    public fileName: string

    constructor(fileName?: string) {
        this.fileName = fileName || "cars"
    }

    downloadCarsFile(cars: Car[]) {
        if (!cars.length) return

        const raw = carsSerializer(cars)
        const encodedObj = URL.createObjectURL(
            new Blob([raw], {
                type: "application/json",
            })
        )

        // create "a" HTML element with href to file
        const link = document.createElement("a")
        link.href = encodedObj
        link.download = `${this.fileName}.json`
        document.body.appendChild(link)
        link.click()

        // clean up "a" element & remove ObjectURL
        document.body.removeChild(link)
        URL.revokeObjectURL(encodedObj)
    }

    loadFromFile(file: RcFile, onLoad: (cars: Car[]) => void) {
        const reader = new FileReader()
        reader.readAsText(file)
        reader.onload = () => {
            const raw = reader.result?.toString() || "[]"
            let cars: Car[] = []
            let errors: CarsValidatorReturnTypes = []

            try {
                cars = JSON.parse(raw)
            } catch (e) {
                if (e instanceof Error) {
                    errors.push({
                        message: `File is corrupted (wrong json)`,
                        meta: e.message,
                    })
                }
            }
            errors.push(...carsValidator(cars))
        }
    }
}

```

```

    if (errors?.length) {
      errors.forEach((e) =>
        notification.error({
          message: `${e.message}`,
          description: `Please fix the detected error: ${e.meta}`,
          placement: "bottomLeft",
          duration: 30,
        })
      )
    }

    if (cars.length) {
      onLoad(cars)
      message.success(`Loaded cars from ${file.name}`)
    } else {
      message.info(`File ${file.name} was empty`)
    }
  }
}

export const carsPersistor = new CarsPersister()

export const generateExtremaFuelConsumptionDesc = (
  extrema: FuelConsumptionExtrema
) => {
  let compiled = ""

  for (const [brand, [min, max]] of Object.entries(extrema)) {
    compiled += `${brand} (min: ${min}, max: ${max}), `
  }

  return compiled.slice(0, -2)
}

```

Графічного інтерфейсу таблиці:

- table.tsx:

```

export type CarsTableEditProps = {
  edit?: {
    component?: (save: () => void, ref: Ref<any>) => React.ReactNode
    serialize?: (value: any) => any
    deserialize?: (value: any) => any
    rules?: Rule[]
  }
  dataIndex: keyof Car
}

```


Протокол роботи програми

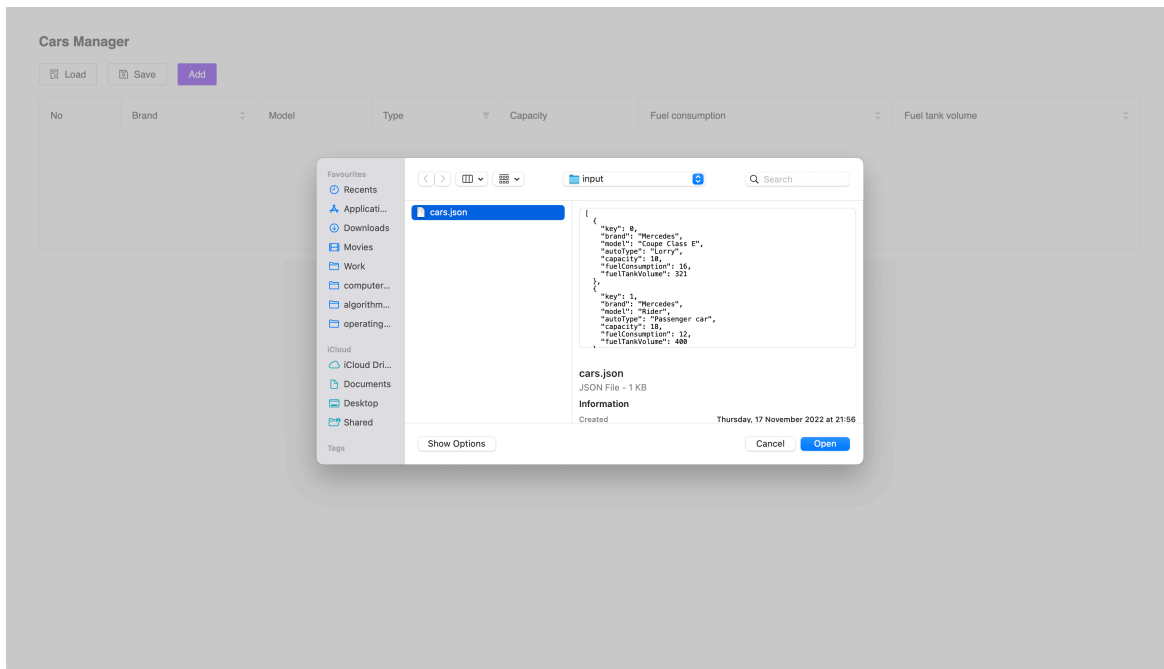


Рис. 1 – інтерфейс програми, користувач вибирає опцію імпорту

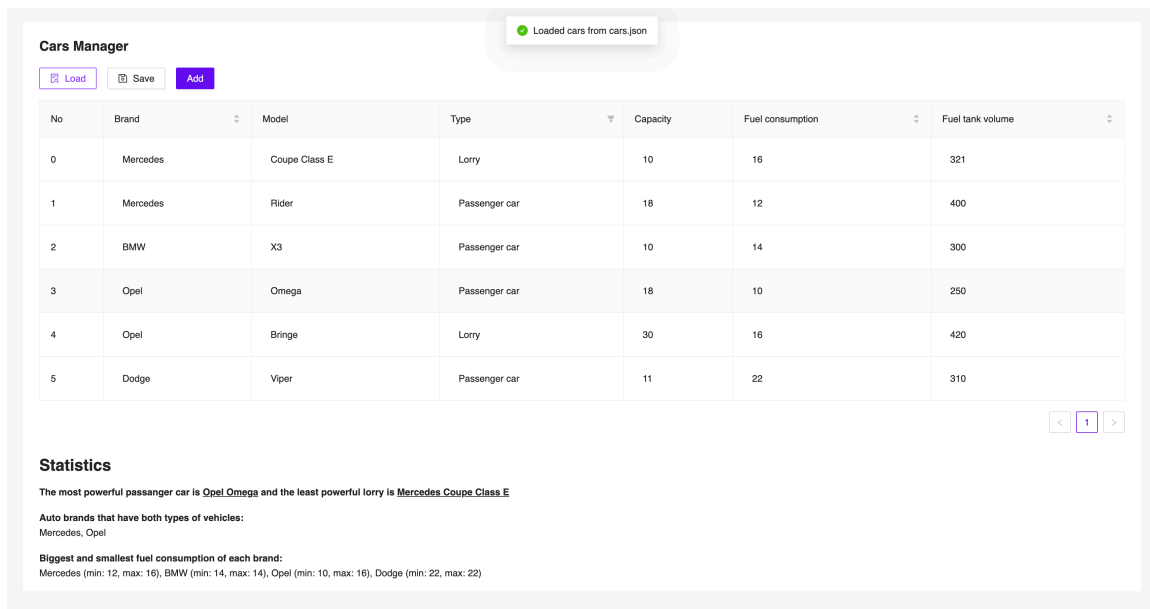


Рис. 2 – успішний імпорт даних користувачем. Сформована таблиця автомобілів.

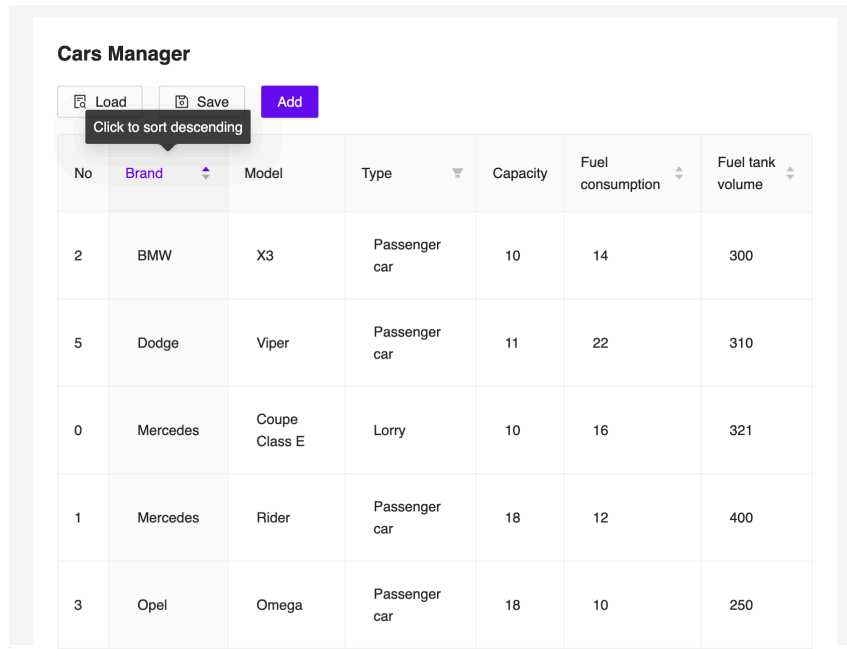


Рис. 3 – користувач сортує записи по назві бренду в алфавітному порядку

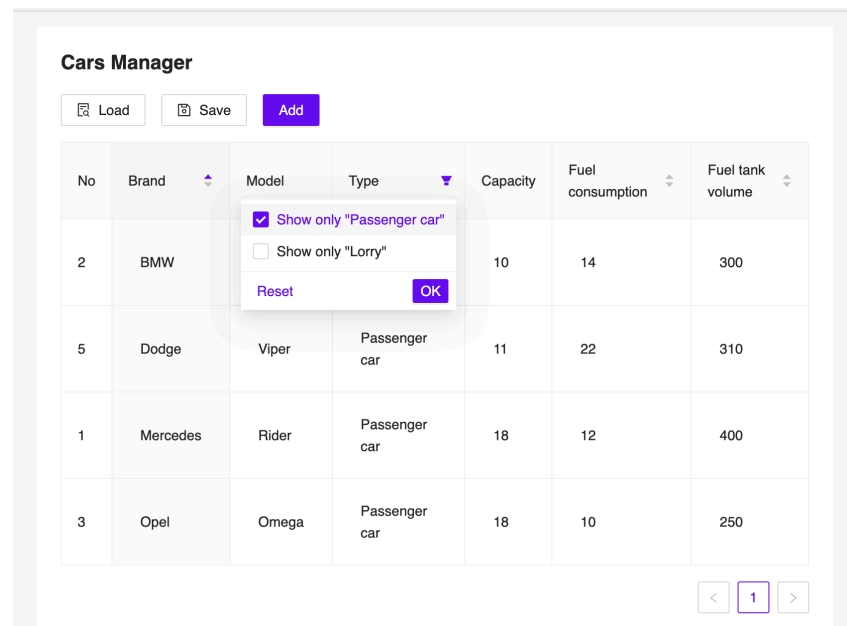








Рис. 4 – користувач застосовує фільтр “Показати лише пасажирські автомобілі

Cars Manager

 Load

 Save

Add

No	Brand 	Model	Type 	Capacity	Fuel consumption 	Fuel tank volume 
3	Opel	Omega	Passenger car	18	10	250
1	Mercedes	Rider	Passenger car	18	12	400
2	BMW	X3	Passenger car	10	14	300
5	Dodge	Viper	Passenger car	11	22	310

< 1 >

Рис. 5 – користувач застосовує фільтр та сортує записи по розходу палива одночасно

Statistics

The most powerful passanger car is Opel Omega and the least powerful lorry is Mercedes Coupe Class E

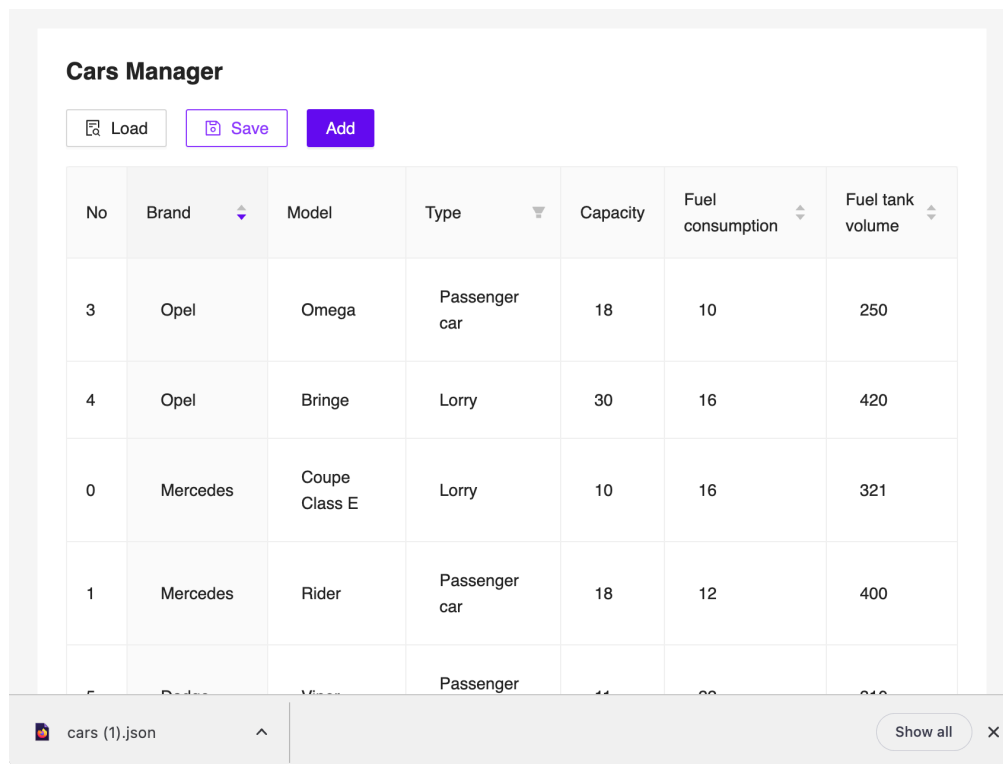
Auto brands that have both types of vehicles:

Mercedes, Opel

Biggest and smallest fuel consumption of each brand:

Mercedes (min: 12, max: 16), BMW (min: 14, max: 14), Opel (min: 10, max: 16), Dodge (min: 22, max: 22)

Рис. 6 – Статистичні дані по наведеній інформації з таблиці



The screenshot shows a web application titled "Cars Manager". It features a table with 7 columns: No, Brand, Model, Type, Capacity, Fuel consumption, and Fuel tank volume. The table contains 5 rows of data. Below the table, there is a file explorer showing a file named "cars (1).json".

No	Brand	Model	Type	Capacity	Fuel consumption	Fuel tank volume
3	Opel	Omega	Passenger car	18	10	250
4	Opel	Bringe	Lorry	30	16	420
0	Mercedes	Coupe Class E	Lorry	10	16	321
1	Mercedes	Rider	Passenger car	18	12	400
5	Dodge	Viper	Passenger	14	22	240

cars (1).json Show all X

Рис. 7 – Експорт даних – успішно

```

1 [
2   {
3     "key": 0,
4     "brand": "Mercedes",
5     "model": "Coupe Class E",
6     "autoType": "Lorry",
7     "capacity": 10,
8     "fuelConsumption": 16,
9     "fuelTankVolume": 321
10  },
11  {
12    "key": 1,
13    "brand": "Mercedes",
14    "model": "Rider",
15    "autoType": "Passenger car",
16    "capacity": 18,
17    "fuelConsumption": 12,
18    "fuelTankVolume": 400
19  },
20  {
21    "key": 2,
22    "brand": "BMW",
23    "model": "X3",
24    "autoType": "Passenger car",
25    "capacity": 10,
26    "fuelConsumption": 14,
27    "fuelTankVolume": 300
28  },
29  {
30    "key": 3,
31    "brand": "Opel",
32    "model": "Omega",
33    "autoType": "Passenger car",
34    "capacity": 18,
35    "fuelConsumption": 10,
36    "fuelTankVolume": 250
37  },
38  {
39    "key": 4,
40    "brand": "Opel",
41    "model": "Bringe",
42    "autoType": "Lorry",
43    "capacity": 30,
44    "fuelConsumption": 16,
45    "fuelTankVolume": 420
46  },
47  {
48    "key": 5,
49    "brand": "Dodge",
50    "model": "Viper",
51    "autoType": "Passenger car",
52    "capacity": 11,
53    "fuelConsumption": 22,
54    "fuelTankVolume": 310

```

Рис.8 – Вигляд експортованих даних / даних для імпорту.

Демонстрація формату даних при роботі з програмою.

key - Номер автомобілю - ціле 32 бітне число

brand - Назва бренду/виробника

model - Назва моделі автомобіля

autoType - Тип автомобілю одна із стрічок (Пасажи́рське авто, Вантажівка)

capacity - Потужність автомобілю - ціле 32 бітне число

fuelConsumption - Розхід палива - ціле 32 бітне число

fuelTankVolume - Об'єм паливного баку - ціле 32 бітне число

Опис та обробка виняткових ситуацій

1. Неправильний тип автомобілю при імпорті даних

The screenshot shows the 'Cars Manager' application interface. At the top, there is a green notification bubble that says 'Loaded cars from cars.json'. Below this, there are three buttons: 'Load' (purple), 'Save' (grey), and 'Add' (purple). The main part of the interface is a table with the following columns: 'No', 'Brand', 'Model', 'Type', 'Capacity', 'Fuel consumption', and 'Fuel tank volume'. The table contains four rows of data. The first row has '0' in the 'No' column, 'Mercedes' in 'Brand', 'Coupe Class E' in 'Model', 'Bebra' in 'Type', '10' in 'Capacity', '16' in 'Fuel consumption', and '321' in 'Fuel tank volume'. The second row has '1' in 'No', 'Mercedes' in 'Brand', 'Rider' in 'Model', 'Passenger car' in 'Type', '18' in 'Capacity', '12' in 'Fuel consumption', and '400' in 'Fuel tank volume'. The third row has an empty 'No' column, 'Mercedes' in 'Brand', 'Rider' in 'Model', 'Passenger car' in 'Type', '10' in 'Capacity', '14' in 'Fuel consumption', and '300' in 'Fuel tank volume'. The fourth row has '3' in 'No', 'Opel' in 'Brand', 'Omega' in 'Model', 'Passenger car' in 'Type', '18' in 'Capacity', '10' in 'Fuel consumption', and '250' in 'Fuel tank volume'. A red error message box is overlaid on the table, stating: 'Enum error: Must be equal to one of the allowed values. Please fix the detected error: at /0/autoType'. The error message is accompanied by a red 'X' icon and a close button.

No	Brand	Model	Type	Capacity	Fuel consumption	Fuel tank volume
0	Mercedes	Coupe Class E	Bebra	10	16	321
1	Mercedes	Rider	Passenger car	18	12	400
	Mercedes	Rider	Passenger car	10	14	300
3	Opel	Omega	Passenger car	18	10	250

Рис.9 – Виняткова ситуація.

2. Невірний тип у полі “Capacity” при імпорті даних

The screenshot shows the 'Cars Manager' interface. At the top, there is a green notification bubble that says 'Loaded cars from cars.json'. Below this are three buttons: 'Load' (purple), 'Save' (grey), and 'Add' (purple). The main part of the interface is a table with the following columns: 'No', 'Brand', 'Model', 'Type', 'Capacity', 'Fuel consumption', and 'Fuel tank volume'. The table contains four rows of data. An error message is displayed over the table, indicating a 'Type error: Must be uint32' at the 'Capacity' field of the first row, where the value is 'Hello'.

No	Brand	Model	Type	Capacity	Fuel consumption	Fuel tank volume
0	Mercedes	Coupe Class E	Lorry	10	16	321
1	Mercedes	Rider	Passenger car	Hello	12	400
2			Passenger	10	14	300
3	Opel	Omega	Passenger car	18	10	250

Error Message: Type error: Must be uint32
Please fix the detected error: at /1/capacity

Рис.10 – Вийняткова ситуація.

3. Невалідний формат даних json при імпорті

The screenshot shows the 'Cars Manager' interface. At the top, there is a grey notification bubble that says 'No data'. Below this are three buttons: 'Load' (purple), 'Save' (grey), and 'Add' (purple). The main part of the interface is a table that is currently empty. An error message is displayed over the table, indicating a 'File is corrupted (wrong json)' at position 360, where a comma or closing bracket is expected after an array element.

Error Message: File is corrupted (wrong json)
Please fix the detected error: Expected ',' or ']' after array element in JSON at position 360

Рис.11 – Вийняткова ситуація.

4. Число замість стрічки для імені при імпорті

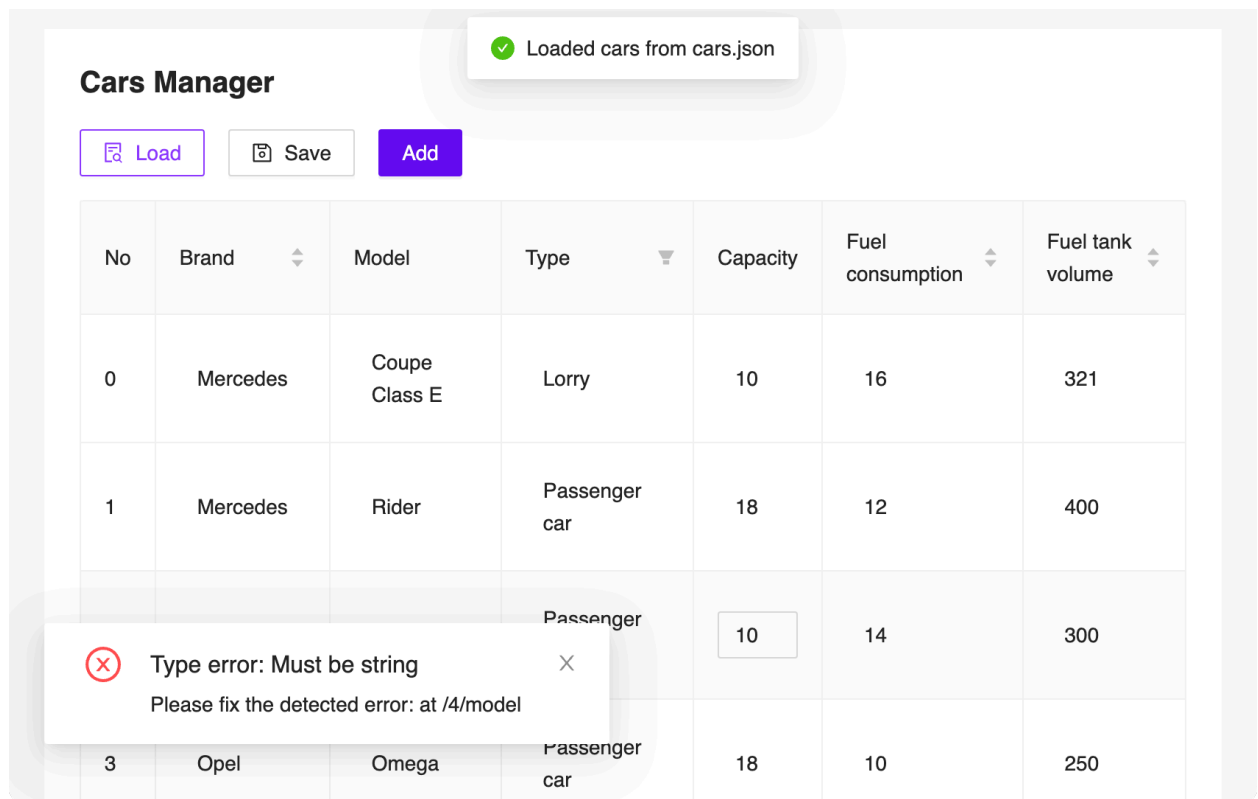


Рис.12 – Вийняткова ситуація.

Інструкція для користувача

Призначення

Метою створення програми є відображення даних про автомобілі. Це має полегшити облік автомобілів для власників бізнесу.

Компоненти ПЗ

Програму розроблено мовою TypeScript в термінальному редакторі тексту NeoVim. Програма розрахована на користувачів ОС на яку можна встановити сучасний браузер.

Вимоги до програмного та апаратного забезпечення:

Операційна система: MacOS/Windows xp+/Android/IOS/Linux.

Мінімальний обсяг ОЗП: 0.5Gb.

Мінімально необхідний простір на диску: 32 Мб.

Процесор: 32-розрядний з мінімальною тактовою частотою 1,5 ГГц.

Монітор: мінімальна роздільна здатність 600x400

Периферійні пристрої: клавіатура та миша

Інсталяція ПЗ

Перед використанням веб додатку, потрібно встановити сучасний веб браузер

Робота з програмою

Робота з програмою не вимагає досконалих навичок роботи з програмним забезпеченням та орієнтована на пересічного користувача. Нижче наведені підказки для роботи з даним ПЗ.

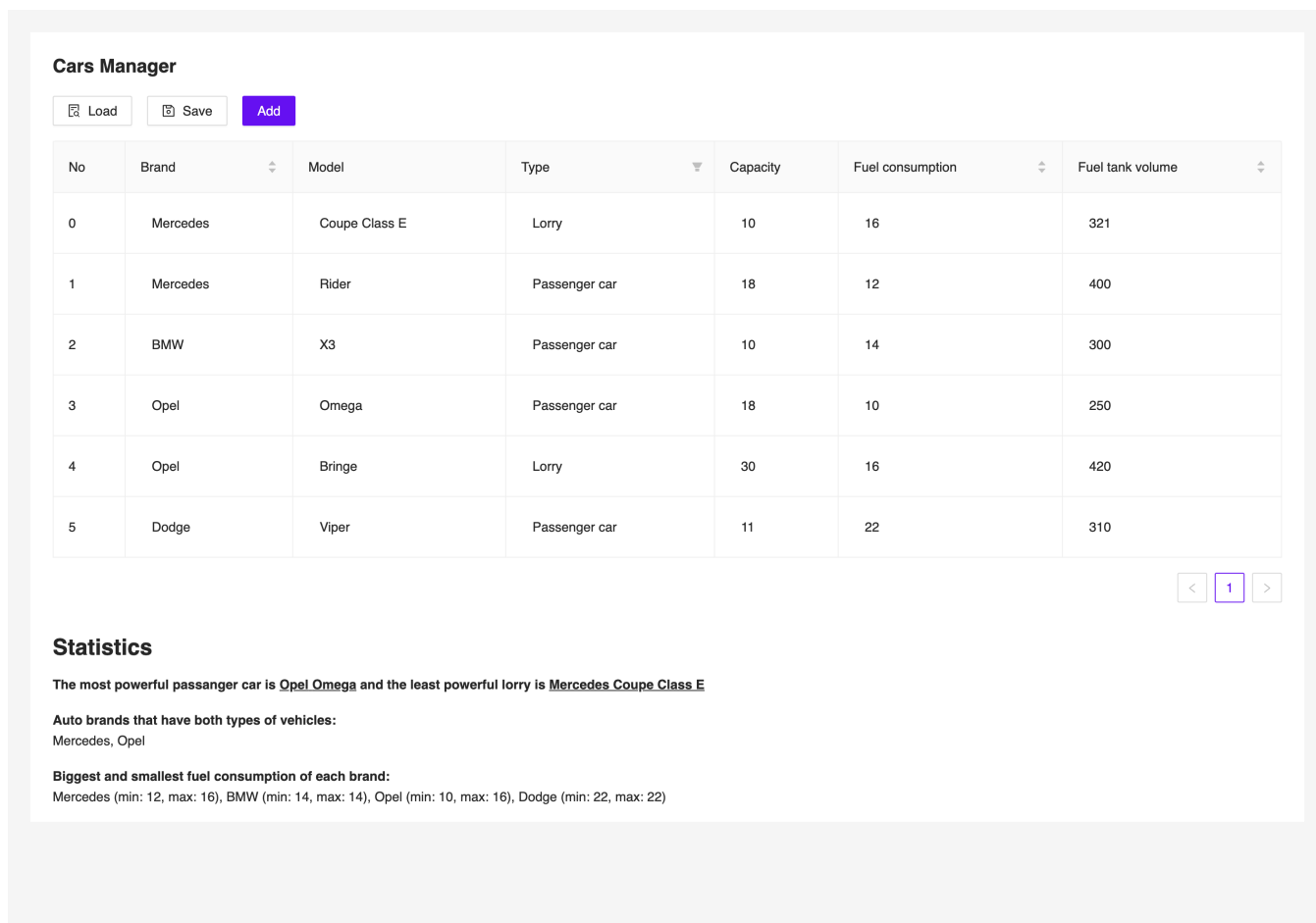


Рис. 22 – Графічний інтерфейс програми

Для створення нового автомобіля, натисніть кнопку “**Add**” та заповніть новий рядок який з’явився в таблиці або імпортуйте файл (кнопка “**Load**”).

Для фільтрування або сортування, скористайтесь іконками на назві стовпчиків таблиці.

Ви також можете експортувати дані з таблиці, натиснувши кнопку “**Export**”

Висновки

Під час виконання цієї курсової роботи я засвоїв навички розробки програмного забезпечення мовою програмування TypeScript. Я закріпив знання створення графічних інтерфейсів за допомогою веб-бібліотеки React та мета-фреймворку Next.js.

Я також закріпив свої знання в об'єктно-орієнтованому/функціональному програмуванні та шаблонах проектування.

Я оновив знання створення UML-діаграм, як-от: діаграма прецедентів, діаграма послідовностей, діаграма класів.

Під час виконання курсової я використав декілька структур даних, як-от список та мапа.

Для розширення проєкта буде доцільно додати інтеграцію із реляційною або ж нереляційною базою даних.

Список літератури

- Об'єктно-орієнтоване програмування: методичні вказівки до виконання курсової роботи для студентів напрямку 6.121 «Інженерія програмного забезпечення» / Укл. Коротєєва Т.О., Дяконюк Л.М.– Львів: Національний університет “Львівська політехніка” кафедра програмного забезпечення, 2020. – 27с.
- Патерни проєктування. Рефакторінг.Гуру. (n.d.). Цитовано: Жовтень 24, 2022, <https://refactoring.guru/design-patterns>
- Офіційна онлайн документація веб бібліотеки React - <https://reactjs.org/docs/react-api.html>
- Офіційна онлайн документація веб фреймворку Next.js - <https://nextjs.org/docs/getting-started>
- Офіційна онлайн документація бібліотеки веб компонентів Ant Design - <https://ant.design/docs/react/introduce>