



Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №3

Розробка інтелектуальної інформаційної системи на основі
інтелектуальних Web-агентів

Виконав

студент групи IT-41ф

Новиков Д. М.

Перевірив:

доц. каф. ІСТ
Кравець П.І.

Мета роботи: навчитися створювати інтелектуальні інформаційні системи на основі інтелектуальних Web-агентів, які взаємодіють із певними службами в середовищі Internet і реалізують деякі корисні завдання для користувачів, наприклад, збір необхідної інформації.

Завдання роботи:

1. Ознайомитися з основними теоретичними відомостями за темою роботи.
2. Розробити програмного Web-агента, який взаємодіє із певними службами в Internet та збирає інформацію на основі критеріїв, що задані користувачем.
3. За допомогою протоколу HTTP Web-агент має перевіряти зміни заданих Web сайтів, на яких відображається потрібна інформація (наприклад, новини). При цьому зв'язок з сайтами організовується за допомогою інтерфейсу сокетів.
4. З метою створення підсистеми автоматичного поповнення бази знань інтегрувати створеного програмного агента до інтелектуальної системи підтримки прийняття рішень. При необхідності з метою забезпечення інтеграції розробити додаткове програмне, інформаційне, лінгвістичне, математичне забезпечення.
5. Виконати тестування розробленого програмного забезпечення (інтелектуальна система підтримки прийняття рішень з інтегрованим Web-агентом) шляхом звертання за допомогою Web-агента до заданих користувачем Web-сторінок із динамічно змінюваною інформацією з метою поповнення фактів бази знань.
6. Проаналізувати роботу створеного програмного агента. Виявити його переваги та недоліки.
7. Оформити звіт з роботи.

Хід роботи:

Інтерфейс додатка складається з:

- Верхня панель: поле пошуку з автодоповненням, кнопки Search, Update All, налаштування сортування, Export/Import;
- Список результатів (картки) + секція Favorites (закріплени). Підказка під інпутом (якщо не було отримано жодних результатів).

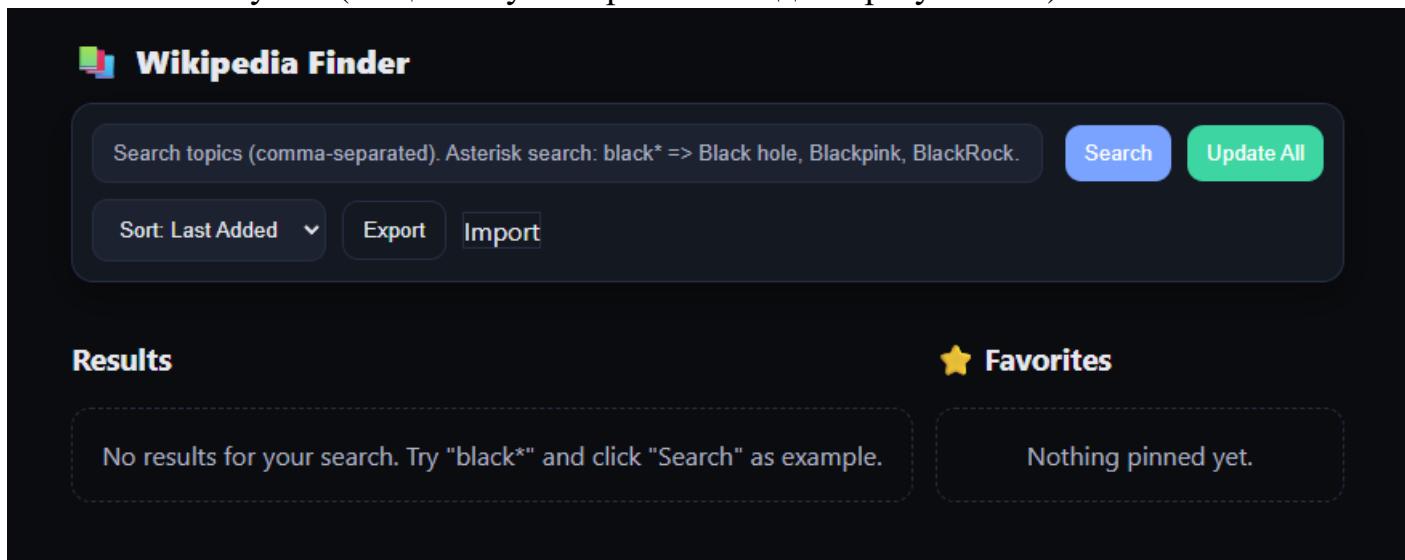


Рис 1 – Інтерфейс програми

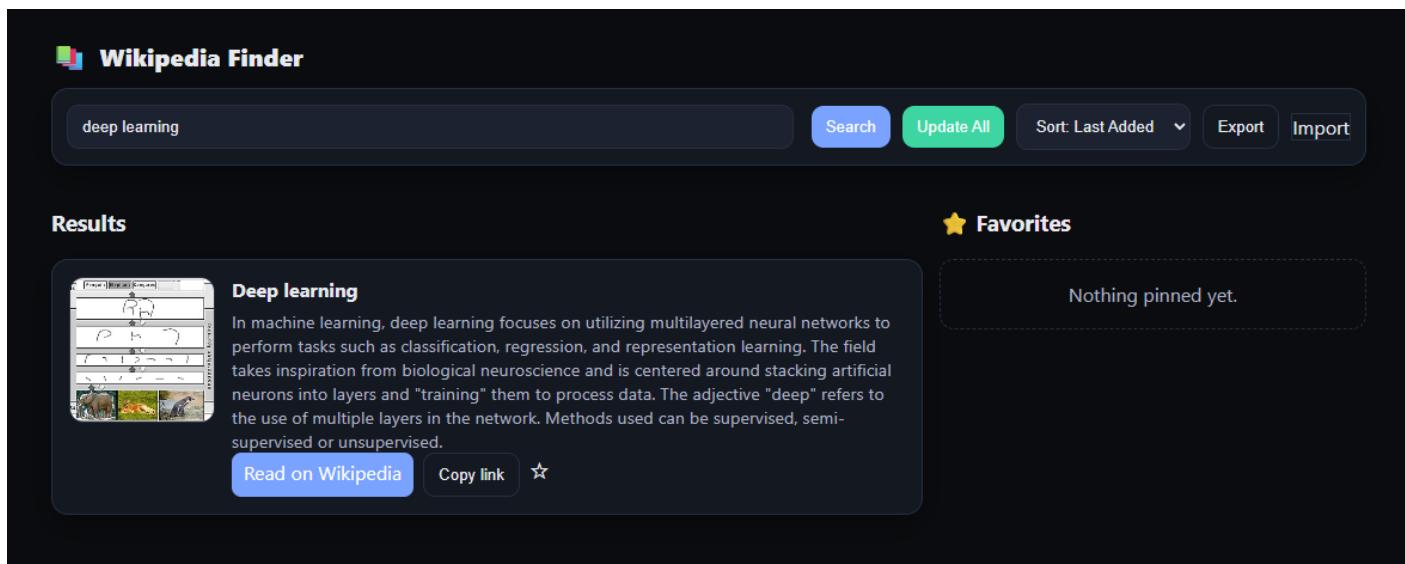


Рис 2 – Результат пошуку

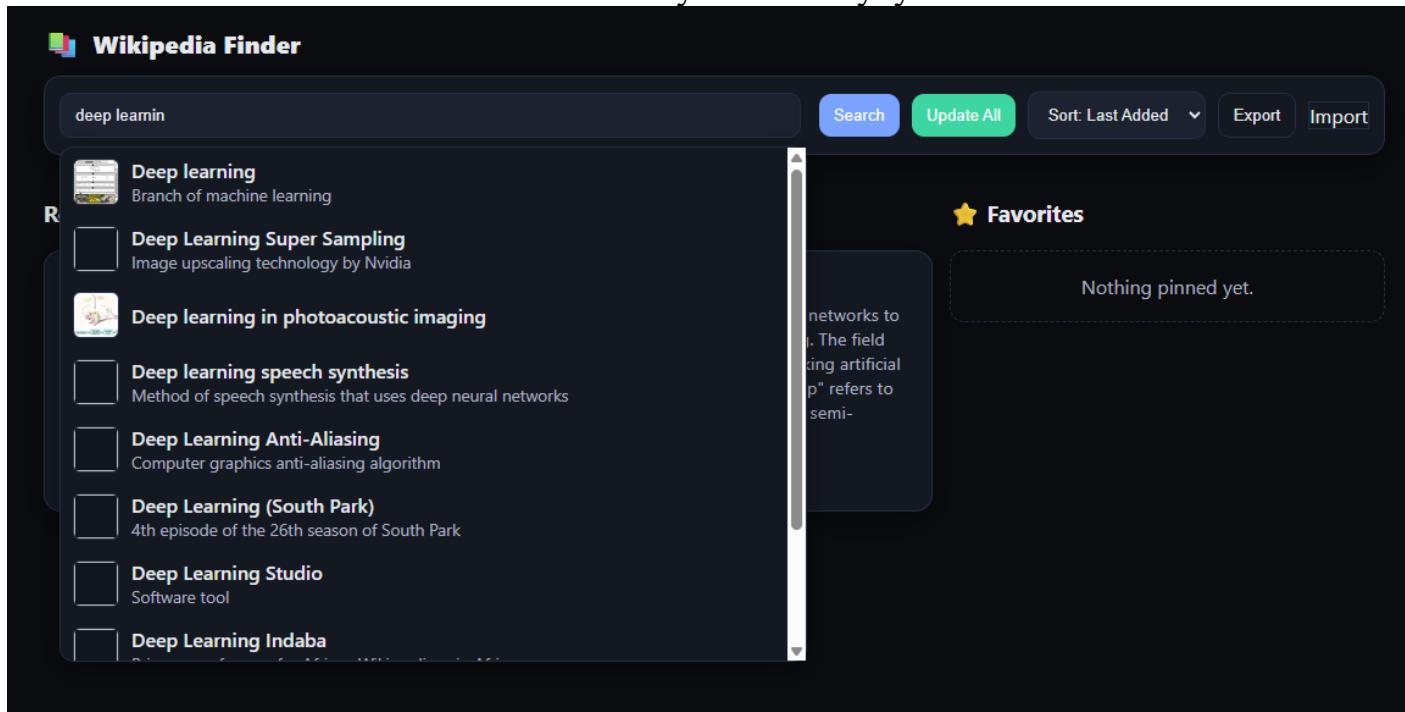


Рис 3 – Поле пошуку з автодоповненням

Розроблена система працює наступним чином:

1. Користувач вводить тему (одну або декілька; підтримуються коми та шаблон із зірочкою *, напр. black*).
2. Інтерфейс виконує запит:
 - a. Автодоповнення (за потреби): для підказок за префіксом;
 - b. Дані картки: Wikipedia REST page/summary/{title} для короткого опису, зображення та посилання.
3. Отримані JSON-дані перетворюються на картки: мініатюра (якщо є), заголовок, стислий опис, та кнопки:
 - a. “Read on Wikipedia” – гіперпосилання на відповідну сторінку на Wikipedia;
 - b. “Copy link” – копіює посилання у буфер обміну;
 - c. “Зірочка” – додає картку до секції Favorites.
4. Для зручності:

- a. Update All повторно завантажує дані для всіх поточних карток (імітація автооновлення);
- b. Favorites дозволяють закріплювати важливі теми;
- c. Export/Import зберігає/відновлює локальну колекцію (JSON);
- d. Сортування: за часом додавання або A-Z;
- e. Астериск-пошук (black*) розгортає запит у топ-відповідники через префікс-пошук.

Приклад роботи:

Розроблена система дозволяє задавати критерії пошуку за публічним API сервісу Wikipedia та отримувати стислі дані стосовно заданої тематики. При цьому вона дозволяє імітувати підтримку автоматичного оновлення даних при їх оновленні на сервісі.

У якості вхідних параметрів приймаються тема пошуку, на виході надається зображення, коротке визначення та посилання на більшу кількість інформації.

Приклад використання:

1. В полі пошуку ввести: Valve, Portal 2 або шаблон black*;
2. Клікнути Search: для black* система автоматично підбере кілька тем (напр., Black hole, Blackpink, BlackRock, Blackwater, etc.);
3. На екрані з'являться картки: зображення (якщо доступне), коротке визначення та посилання на повну статтю;
4. Кнопка Update All оновить усі картки (як приклад імітації “автоматичної підтримки актуальності”).

Wikipedia Finder

Search topics (comma-separated). Asterisk search: black* => Black hole, Blackpink, BlackRock...

Search

Update All

Sort: Last Added ▾

Export

Import

Results



Portal 2

Portal 2 is a 2011 puzzle-platform game developed by Valve for Windows, macOS, Linux, PlayStation 3, and Xbox 360. The digital PC versions are distributed online by Valve's Steam service, while all retail editions are distributed by Electronic Arts. A port for the Nintendo Switch was released as part of the Portal: Companion Collection in June 2022.

[Read on Wikipedia](#)

[Copy link](#)



Valve

A valve is a device or natural object that regulates, directs or controls the flow of a fluid by opening, closing, or partially obstructing various passageways. Valves are technically fittings, but are usually discussed as a separate category. In an open valve, fluid flows in a direction from higher pressure to lower pressure. The word is derived from the Latin *valva*, the moving part of a door, in turn from *volvere*, to turn, roll.

[Read on Wikipedia](#)

[Copy link](#)



★ Favorites

Nothing pinned yet.

Wikipedia Finder

black*

Search

Update All

Sort: Last Added ▾

Export

Import



Black

Darkest color due to absence or absorption of light



Black hole

Astrophysical phenomenon



Blackpink

South Korean girl group



BlackRock

American investment company



Blackwater (company)

American private military contractor



Black Death

1346–1353 pandemic in Eurasia and North Africa



Black Dahlia

American murder victim (1924–1947)



Black Sea

★ Favorites

Nothing pinned yet.

Search topics (comma-separated). Asterisk search: black* => Black hole, Blackpink, BlackRock...

[Search](#)[Update All](#)

Sort: Last Added ▾

[Export](#)[Import](#)

Results

★ Favorites

Nothing pinned yet.



BlackBerry Limited

BlackBerry Limited, formerly Research In Motion (RIM), is a Canadian software company specializing in secure communications and the Internet of Things (IoT). Founded in 1984, it developed the BlackBerry brand of two-way pagers, smartphones, and tablets. The company later transitioned to providing software and services and holds critical software application patents.

[Read on Wikipedia](#)[Copy link](#)

Black Sabbath

Black Sabbath were an English heavy metal band formed in Birmingham in 1968 by guitarist Tony Iommi, drummer Bill Ward, bassist Geezer Butler and vocalist Ozzy Osbourne. After adopting the Black Sabbath name in 1969, they distinguished themselves through occult themes with horror-inspired lyrics and down-tuned guitars. Their first three albums, *Black Sabbath*, *Paranoid*, and *Master of Reality* (1971), were commercially successful, and are cited as pioneering albums in the development of heavy metal. Subsequent albums *Vol. 4* (1972), *Sabbath Bloody Sabbath* (1973), *Sabotage* (1975), *Technical Ecstasy* (1976), and *Never Say Die!* (1978) saw the band explore more experimental and progressive styles.

[Read on Wikipedia](#)[Copy link](#)

Black Sea

The Black Sea is a marginal Mediterranean sea lying between Europe and Asia, east of the Balkans, south of the East European Plain, west of the Caucasus, and north of Anatolia. It is bounded by Bulgaria, Georgia, Romania, Russia, Turkey, and Ukraine. The Black Sea is supplied by major rivers, principally the Danube, Dnieper and Dniester. Consequently, while six countries have a coastline on the sea, its drainage basin includes parts of 24 countries in Europe.

[Read on Wikipedia](#)[Copy link](#)

Black Dahlia

Elizabeth Short, posthumously known as the Black Dahlia, was an American woman found murdered in the Leimert Park neighborhood of Los Angeles, California, on January 15, 1947. Her case became highly publicized owing to the gruesome nature of the crime, which included the mutilation and bisection of her corpse.

[Read on Wikipedia](#)[Copy link](#)

Black Death

The Black Death was a bubonic plague pandemic that occurred in Europe from 1346 to 1353. It was one of the most fatal pandemics in human history; as many as 50 million people perished, perhaps 50% of Europe's 14th century population. The disease is caused by the bacterium *Yersinia pestis* and spread by fleas and through the air. One of the most significant events in European history, the Black Death had far-reaching population, economic, and cultural impacts. It was the beginning of the second plague pandemic. The plague created religious, social and economic upheavals, with profound effects on the course of European history.

Search topics (comma-separated). Asterisk search: black* => Black hole, Blackpink, BlackRock...

[Search](#)[Update All](#)

Sort: Last Added ▾

[Export](#)[Import](#)

Results

★ Favorites

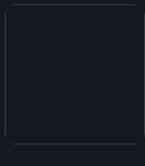
Nothing pinned yet.

Loading...Fetching summary for **Black hole...****Blackpink**

Blackpink is a South Korean girl group formed by YG Entertainment. The group is composed of four members: Jisoo, Jennie, Rosé, and Lisa. Regarded by various publications as the "biggest girl group in the world", they are recognized as a leading force in the Korean Wave and an ambassador of the "girl crush" concept in K-pop, which explores themes of self-confidence and female empowerment.

[Read on Wikipedia](#)[Copy link](#)**BlackRock**

BlackRock, Inc. is an American multinational investment company. Founded in 1988, initially as an enterprise risk management and fixed income institutional asset manager, BlackRock is the world's largest asset manager, with \$13.5 trillion in assets under management as of 2025. Headquartered in New York City, BlackRock has 70 offices in 30 countries and clients in 100 countries.

[Read on Wikipedia](#)[Copy link](#)**Blackwater (company)**

Constellis, formerly Blackwater, is an American private military contractor founded on December 26, 1997, by former Navy SEAL officer Erik Prince. It was renamed Xe Services in 2009, and was again renamed to Academi in 2011, after it was acquired by a group of private investors. In 2014, Academi merged with Triple Canopy to form Constellis Holdings.

[Read on Wikipedia](#)[Copy link](#)

Висновки: за результатом виконання лабораторної роботи створено інтелектуальну інформаційну системи на основі Web-агенту, яка взаємодіє з публічними API Wikipedia. Вона приймає критерії пошуку від користувача, звертається до відкритих кінцевих точок, обробляє відповіді та відображає стислі довідкові картки.

Вихідний код застосунку можна знайти за наступним посиланням на [GitHub](#).

ДОДАТОК А

Вихідний код алгоритму

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Wikipedia Finder</title>
    <style>
      /* ---- Dark UI ---- */
      :root {
        --bg: #0b0c0f;
        --fg: #e7e9ee;
        --muted: #aeb3c2;
        --card: #141821;
        --border: #263043;
        --accent: #7aa2ff;
        --accent-2: #3dd6a2;
        --danger: #ff6b6b;
        --chip: #1d2230;
        --chip-border: #2b354a;
        --shadow: 0 8px 24px rgba(0, 0, 0, .35);
      }

      html,
      body {
        height: 100%
      }

      body {
        margin: 0;
        font-family: ui-sans-serif, system-ui, -apple-system, Segoe UI, Roboto, Arial;
        background: var(--bg);
        color: var(--fg);
        display: flex;
        flex-direction: column;
      }

      .container {
        max-width: 1100px;
        margin: 28px auto;
        padding: 0 16px
      }

      .title {
        font-weight: 800;
        letter-spacing: .2px;
        display: flex;
        align-items: center;
        gap: 10px;
        margin-bottom: 12px
      }

      .badge {
        font-size: .8rem;
        padding: 2px 8px;
        border: 1px solid var(--border);
        border-radius: 999px;
        background: var(--chip)
      }

      .toolbar {
        display: flex;
        align-items: center;
        gap: 10px;
        flex-wrap: wrap;
        background: var(--card);
        border: 1px solid var(--border);
        border-radius: 14px;
        padding: 12px;
        box-shadow: var(--shadow)
      }

      /* Prevent overlap: input column grows, others keep natural width */
    </style>
  </head>
  <body>
```

```
.ac-wrap {
  position: relative;
  flex: 1 1 520px;
  min-width: 240px
}

.toolbar> :not(.ac-wrap) {
  flex: 0 0 auto;
}

input[type="text"],
select,
button {
  padding: 10px 12px;
  border-radius: 10px;
  border: 1px solid var(--border);
  background: var(--chip);
  color: var(--fg);
  outline: none;
}

input::placeholder {
  color: var(--muted)
}

button {
  cursor: pointer;
  border: 1px solid var(--border)
}

.btn {
  background: var(--accent);
  border: none;
  color: #fff
}

.btn.secondary {
  background: var(--accent-2)
}

.btn.ghost {
  background: transparent;
  border: 1px solid var(--border);
  color: var(--fg)
}

.grid {
  display: grid;
  grid-template-columns: repeat(12, 1fr);
  gap: 14px;
  margin-top: 16px
}

.col-8 {
  grid-column: span 8
}

.col-4 {
  grid-column: span 4
}

@media(max-width:960px) {

  .col-8,
  .col-4 {
    grid-column: span 12
  }
}

.card {
  background: var(--card);
  border: 1px solid var(--border);
  border-radius: 14px;
  padding: 14px;
  box-shadow: var(--shadow);
}
```

```
display: flex;
gap: 14px;
align-items: flex-start;
}

.thumb {
width: 120px;
height: 120px;
border-radius: 12px;
object-fit: cover;
border: 1px solid var(--border);
background: #00000022
}

.meta {
flex: 1;
min-width: 0
}

.fact-title {
font-weight: 700;
margin: 0 0 6px 0
}

.muted {
color: var(--muted);
font-size: .9rem
}

.link {
color: #fff;
text-decoration: none;
background: var(--accent);
padding: 6px 10px;
border-radius: 8px
}

.actions {
display: flex;
gap: 8px;
flex-wrap: wrap
}

.star {
cursor: pointer;
user-select: none;
font-size: 20px
}

.empty {
padding: 18px;
border: 1px dashed var(--border);
border-radius: 12px;
text-align: center;
color: var(--muted)
}

/* ---- Autocomplete ---- */
.ac-panel {
position: absolute;
z-index: 20;
top: 44px;
left: 0;
right: 0;
background: var(--card);
border: 1px solid var(--border);
border-radius: 12px;
box-shadow: var(--shadow);
max-height: 420px;
overflow: auto;
display: none;
}

.ac-item {
display: flex;
```

```

    gap: 10px;
    align-items: center;
    padding: 8px 10px;
    cursor: pointer
}

.ac-item:hover,
.ac-item.active {
  background: #fffffff12
}

.ac-thumb {
  width: 36px;
  height: 36px;
  border-radius: 6px;
  object-fit: cover;
  border: 1px solid var(--border);
  background: #00000022
}

.ac-title {
  font-weight: 600
}

.ac-desc {
  font-size: .85rem;
  color: var(--muted)
}

.kbd {
  font-family: ui-monospace, SFMono-Regular, Menlo, Consolas, "Liberation Mono", monospace;
  background: #fffffff12;
  border: 1px solid var(--border);
  padding: 2px 6px;
  border-radius: 6px;
  color: var(--muted)
}

```

</style>

</head>

<body>

```

<div class="container">
  <div class="title">
    <span style="font-size:22px">🔍</span>
    <div>
      <div style="font-size:20px">Wikipedia Finder</div>
    </div>
  </div>
  <div class="toolbar">
    <div class="ac-wrap">
      <input id="search" type="text" placeholder="Search topics (comma-separated). Asterisk search: black* => Black hole, Blackpink, BlackRock..." style="width:95%" />
      <div id="ac" class="ac-panel"></div>
    </div>
    <button id="search-button" class="btn">Search</button>
    <button id="update-all-button" class="btn secondary">Update All</button>
    <select id="sorter">
      <option value="recent">Sort: Last Added</option>
      <option value="az">Sort: A → Z</option>
    </select>
    <button id="export" class="btn ghost">Export</button>
    <label class="btn ghost" style="cursor:pointer">Import <input id="import" type="file" accept="application/json" style="display:none" />
    </label>
  </div>
  <div class="grid">
    <div class="col-8">
      <h3>Results</h3>
      <div id="facts-list"></div>
    </div>
    <div class="col-4">
      <h3>★ Favorites</h3>
      <div id="favorites"></div>
    </div>
  </div>
</div>

```

```
<script>
  const $ = s => document.querySelector(s);
  const factsList = $('#facts-list');
  const favoritesBox = $('#favorites');
  const searchInput = $('#search');
  const searchBtn = $('#search-button');
  const updateAllBtn = $('#update-all-button');
  const sorter = $('#sorter');
  const exportBtn = $('#export');
  const importInp = $('#import');
  const acPanel = $('#ac');
  const LS_KEY = 'facts:facts',
    LS_FAV = 'facts:favorites';
  const load = (k, def) => {
    try {
      return JSON.parse(localStorage.getItem(k)) ?? def;
    } catch {
      return def;
    }
  };
  const save = (k, v) => localStorage.setItem(k, JSON.stringify(v));
  let items = load(LS_KEY, []); // [{title, extract, thumbnail, originalImage, content_urls, ts}]
  let favorites = load(LS_FAV, []); // [title]
  // ----- Wikipedia summary -----
  async function fetchFact(title) {
    const url = `https://en.wikipedia.org/api/rest_v1/page/summary/${encodeURIComponent(title)}`;
    const res = await fetch(url, {
      headers: {
        'Accept': 'application/json'
      }
    });
    if (!res.ok) throw new Error('Failed to fetch: ' + title);
    return await res.json();
  }
  // ----- Autocomplete (prefixsearch) -----
  async function fetchSuggestions(prefix, limit = 10) {
    const endpoint = new URL('https://en.wikipedia.org/w/api.php');
    endpoint.search = new URLSearchParams({
      action: 'query',
      format: 'json',
      origin: '*',
      generator: 'prefixsearch',
      gpssearch: prefix,
      gpslimit: String(limit),
      prop: 'pageimages|description',
      piprop: 'thumbnail',
      pithumbsize: '48'
    }).toString();
    const res = await fetch(endpoint, {
      headers: {
        'Accept': 'application/json'
      }
    });
    if (!res.ok) return [];
    const data = await res.json();
    if (!data.query || !data.query.pages) return [];
    return Object.values(data.query.pages).sort((a, b) => a.index - b.index).map(p => ({
      title: p.title,
      desc: p.description || '',
      thumb: p.thumbnail?.source || ''
    }));
  }
  // ----- Asterisk expansion -----
  async function expandQueryToTopics(raw, limitEach = 10) {
    const tokens = raw.split(',').map(s => s.trim()).filter(Boolean);
    const topics = [];
    for (const t of tokens) {
      if (t.endsWith('*')) {
        const prefix = t.slice(0, -1);
        if (!prefix) continue;
        const sug = await fetchSuggestions(prefix, limitEach);
        for (const s of sug) topics.push(s.title);
      } else {
        topics.push(t);
      }
    }
  }

```

```

    }
    // de-dup
    const seen = new Set();
    const out = [];
    for (const t of topics) {
        const k = t.toLowerCase();
        if (!seen.has(k)) {
            seen.add(k);
            out.push(t);
        }
    }
    return out;
}
// ----- Rendering -----
function createCard(data) {
    const el = document.createElement('div');
    el.className = 'card';
    const imgSrc = data.thumbnail?.source || data.originalimage?.source || '';
    const img = document.createElement('img');
    img.className = 'thumb';
    if (imgSrc) {
        img.loading = 'lazy';
        img.src = imgSrc;
        img.alt = data.title;
    } else img.style.opacity = .2;
    const meta = document.createElement('div');
    meta.className = 'meta';
    const title = document.createElement('h4');
    title.className = 'fact-title';
    title.textContent = data.title || 'Untitled';
    const desc = document.createElement('div');
    desc.className = 'muted';
    desc.textContent = data.extract || 'No summary available.';
    const actions = document.createElement('div');
    actions.className = 'actions';
    const a = document.createElement('a');
    a.className = 'link';
    a.href = data.content_urls?.desktop?.page || '#';
    a.target = '_blank';
    a.textContent = 'Read on Wikipedia';
    const star = document.createElement('span');
    star.className = 'star';
    const fav = favorites.includes(data.title);
    star.textContent = fav ? '★' : '☆';
    star.title = fav ? 'Remove from favorites' : 'Add to favorites';
    star.onclick = () => {
        toggleFavorite(data.title);
        star.textContent = favorites.includes(data.title) ? '★' : '☆';
        renderFavorites();
    };
    const copy = document.createElement('button');
    copy.className = 'btn ghost';
    copy.textContent = 'Copy link';
    copy.onclick = async () => {
        if (a.href && a.href !== '#') await navigator.clipboard.writeText(a.href);
        copy.textContent = 'Copied!';
        setTimeout(() => copy.textContent = 'Copy link', 900);
    };
    actions.append(a, copy, star);
    meta.append(title, desc, actions);
    el.append(img, meta);
    return el;
}

function render() {
    let list = [...items];
    if (sorter.value === 'az') list.sort((a, b) => a.title.localeCompare(b.title));
    else list.sort((a, b) => b.ts - a.ts);
    factsList.innerHTML = '';
    if (!list.length) {
        const empty = document.createElement('div');
        empty.className = 'empty';
        empty.innerHTML = 'No results for your search. Try "black*" and click "Search" as example.';
        factsList.appendChild(empty);
    } else {

```

```

        for (const it of list) factsList.appendChild(createCard(it));
    }
    renderFavorites();
}

function renderFavorites() {
    favoritesBox.innerHTML = '';
    const favs = items.filter(i => favorites.includes(i.title));
    if (!favs.length) {
        const empty = document.createElement('div');
        empty.className = 'empty';
        empty.textContent = 'Nothing pinned yet.';
        favoritesBox.appendChild(empty);
        return;
    }
    for (const it of favs) favoritesBox.appendChild(createCard(it));
}
function toggleFavorite(title) {
    const i = favorites.indexOf(title);
    if (i === -1) favorites.push(title);
    else favorites.splice(i, 1);
    save(LS_FAV, favorites);
}
// ----- Actions -----
async function addTopic(title) {
    // quick inline skeleton
    const sk = document.createElement('div');
    sk.className = 'card';
    sk.innerHTML =
<div class="thumb" style="opacity:.2"></div>
<div class="meta">
    <div class="fact-title">Loading...</div>
    <div class="muted">Fetching summary for
        <b>${title}</b>...
    </div>
</div>';
    factsList.prepend(sk);
    try {
        const data = await fetchFact(title);
        const obj = {
            title: data.title,
            extract: data.extract,
            thumbnail: data.thumbnail,
            originalimage: data.originalimage,
            content_urls: data.content_urls,
            ts: Date.now()
        };
        const idx = items.findIndex(x => x.title === obj.title);
        if (idx >= 0) items[idx] = obj;
        else items.unshift(obj);
        save(LS_KEY, items);
    } catch (e) {
        const err = document.createElement('div');
        err.className = 'card';
        err.style.borderColor = 'var(--danger)';
        err.innerHTML =
<div class="meta">
    <div class="fact-title">Failed: ${title}</div>
    <div class="muted">Could not load summary.</div>
</div>';
        factsList.prepend(err);
    } finally {
        sk.remove();
        render();
    }
}
async function performSearch() {
    const raw = searchInput.value.trim();
    if (!raw) {
        alert('Please enter a topic');
        return;
    }
    const topics = await expandQueryToTopics(raw, 10); // expand asterisk tokens
    for (const t of topics) await addTopic(t);
    searchInput.value = '';
}

```

```

    hideAC();
}
async function performSearchSingle(title) {
  if (!title) return;
  await addTopic(title);
  hideAC();
}
async function updateAll() {
  if (!items.length) return;
  const titles = items.map(x => x.title);
  for (const t of titles) await addTopic(t);
}
function doExport() {
  const payload = {
    items,
    favorites,
    exportedAt: new Date().toISOString()
  };
  const blob = new Blob([JSON.stringify(payload, null, 2)], {
    type: 'application/json'
  });
  const a = document.createElement('a');
  a.href = URL.createObjectURL(blob);
  a.download = 'facts-board.json';
  a.click();
}
function doImport(file) {
  const r = new FileReader();
  r.onload = () => {
    try {
      const j = JSON.parse(r.result);
      if (Array.isArray(j.items)) items = j.items;
      if (Array.isArray(j.favorites)) favorites = j.favorites;
      save(LS_KEY, items);
      save(LS_FAV, favorites);
      render();
    } catch {
      alert('Invalid JSON');
    }
  };
  r.readAsText(file);
}
// ----- Autocomplete UI -----
let acIndex = -1,
  acTimer;

function hideAC() {
  acPanel.style.display = 'none';
  acPanel.innerHTML = '';
  acIndex = -1;
}
function showAC() {
  if (acPanel.children.length) acPanel.style.display = 'block';
}
function setActive(idx) {
  const children = [...acPanel.children];
  children.forEach(el => el.classList.remove('active'));
  if (idx >= 0 && idx < children.length) {
    children[idx].classList.add('active');
    acIndex = idx;
  }
}
function renderAC(list) {
  acPanel.innerHTML = '';
  if (!list.length) {
    hideAC();
    return;
  }
  for (let i = 0; i < list.length; i++) {
    const it = list[i];
    const item = document.createElement('div');
    item.className = 'ac-item';
    item.dataset.title = it.title;
    item.innerHTML =

```

```

        
        <div style="min-width:0">
            <div class="ac-title">${it.title}</div>
            <div class="ac-desc">${it.desc||''}</div>
        </div>;
    // CLICK → run search immediately for this item
    item.onclick = () => performSearchSingle(it.title);
    item.onmouseenter = () => setActive(i);
    acPanel.appendChild(item);
}
acIndex = -1;
showAC();
}
searchInput.addEventListener('input', () => {
    clearTimeout(acTimer);
    const raw = searchInput.value.trim();
    const last = raw.split(',').pop().trim();
    const starPrefix = last.endsWith('*') ? last.slice(0, -1) : last;
    if (!starPrefix) {
        hideAC();
        return;
    }
    acTimer = setTimeout(async () => {
        const list = await fetchSuggestions(starPrefix, 10);
        renderAC(list);
    }, 160);
});
// keyboard on dropdown
searchInput.addEventListener('keydown', (e) => {
    if (acPanel.style.display !== 'block') return;
    const items = [...acPanel.children];
    if (e.key === 'ArrowDown') {
        e.preventDefault();
        setActive(Math.min(acIndex + 1, items.length - 1));
    } else if (e.key === 'ArrowUp') {
        e.preventDefault();
        setActive(Math.max(acIndex - 1, 0));
    } else if (e.key === 'Enter') {
        if (acIndex >= 0) {
            e.preventDefault();
            const t = items[acIndex].dataset.title;
            performSearchSingle(t);
        }
    } else if (e.key === 'Escape') {
        hideAC();
    }
});
document.addEventListener('click', (e) => {
    if (!e.target.closest('.ac-wrap')) hideAC();
});
// ----- Wire up -----
searchBtn.addEventListener('click', performSearch);
updateAllBtn.addEventListener('click', updateAll);
sorter.addEventListener('change', render);
exportBtn.addEventListener('click', doExport);
importInp.addEventListener('change', e => {
    const f = e.target.files?[0];
    if (f) doImport(f);
    e.target.value = '';
});
window.addEventListener('keydown', e => {
    if (e.key === '/' && document.activeElement !== searchInput) {
        e.preventDefault();
        searchInput.focus();
        searchInput.select();
    }
});
// First paint
render();
</script>
</body>
</html>

```

ДОДАТОК Б

Контрольні питання

1. Що являє собою програмний агент?

Програмний агент – це програмна сутність, що самостійно сприймає середовище, приймає рішення та виконує дії для досягнення цілей користувача чи системи. Часто працює у складі мультиагентних систем, має власний стан та поведінку.

2. Поясніть та проаналізуйте властивості агентів: автономність, адаптивність, комунікативність, здатність до співробітництва, персоніфікованість, мобільність.

- Автономність – здатність працювати без постійного втручання людини; сам керує життєвим циклом і планом дій;
- Адаптивність – здатність змінювати поведінку на основі даних/зворотного зв’язку (правила, ML, евристики);
- Комунікативність – обмін повідомленнями з людьми/сервісами/іншими агентами (API, протоколи);
- Співробітництво – координує дії з іншими агентами (розподіл задач, узгодження, конкуренція/кооперація);
- Персоніфікованість – налаштовується під конкретного користувача (історія, профіль);
- Мобільність – здатність мігрувати між вузлами/процесами або принаймні віддалено діяти в різних середовищах.

3. З якою метою застосовуються агенти, використання яких залежить від завдань?

Мета застосування (за завданнями):

- Пошук/збір даних: моніторинг сайтів, парсинг, фільтрація;
- Рекомендації/персоналізація: підбір контенту/товарів;
- Торгівля/аукціони: автоматичні ставки, закупівлі;
- Автоматизація рутин: періодичні перевірки, сповіщення;
- Інтеграція: обмін даними між сервісами.

4. Що таке розважальні агенти?

Програмні персонажі/боти для ігор, віртуальних світів, чатів та медіа, що взаємодіють із користувачем (NPC, віртуальні ведучі, компаньйони).

5. Проаналізуйте комп’ютерні віруси як програмні агенти. Це шкідливі агенти з ворожими цілями (порушення цілісності, конфіденційності, доступності). Автономні (працюють без людини), мобільні (поширяються по мережі/носіях), самовідтворювані, мають елементи комунікації (може використовувати певні канали для взаємодії з іншими зараженими системами (наприклад, відправляючи зловмисні повідомлення або виконуючи запити до серверів команд)). Зазвичай не адаптивні у класичному сенсі, але можливі обfuscaciя для уникнення детекції.

6. Які методи використовуються для надання інтелектуальності програмним агентам?

Методи для надання інтелектуальності агентам включають машинне навчання, нейронні мережі, системи на основі знань, дерева правил, нечітку логіку, еволюційні алгоритми.

7. Які протоколи використовує Web-агент?

- HTTP/HTTPS – основний для взаємодії з веб-ресурсами та клієнтами;
- WebSocket – двосторонні сесії в реальному часі (за потреби);
- SMTP/IMAP/POP3 – для email-сповіщень (за потреби);
- FTP/SFTP/HTTP – завантаження файлів (за потреби).

8. Виконайте аналіз архітектури Web-агенту?

- Клієнтська частина (Agent Core): відповідає за ініціалізацію та виконання завдань, отриманих від користувача;
- HTTP-сервер: взаємодіє з користувачем через веб-браузер, обробляючи запити;
- HTTP-клієнт: для з'єднання з іншими серверами та отримання інформації;
- Парсер: витягає потрібні факти з HTML/JSON (регекси/DOM/селектори);
- Сховище/кеш: хеші контенту, мітки часу для виявлення змін;
- Web-інтерфейс: UI + API для пошуку/перегляду/оновлення;
- Конфігурація: джерела, періодичність, ключові слова, фільтри.

9. Якими властивостями володіє Web-агент?

- Автономність – здатність працювати у фоновому режимі без постійного контролю користувача;
- Комунікативність – здатність взаємодіяти з веб-серверами через стандартні протоколи (HTTP);
- Адаптивність – може налаштовувати параметри пошуку відповідно до вимог користувача;
- Персоніфікованість – можливість додавати результати у обране/зберігати історію/налаштування (якщо реалізовано);
- Мобільність – можливість переміщатися по Інтернету для збору необхідної інформації.

10. Проаналізуйте протокол, який використовується для передачі користувачу результатів пошуку за допомогою Web-агента. Агент одночасно виступає HTTP-клієнтом (до зовнішніх ресурсів) і HTTP-сервером (до браузера), забезпечуючи інтерактивний перегляд і оновлення результатів пошуку.

- HTTP-клієнт, що входить до складу Web-агента, надсилає запити на зовнішні веб-сервери для отримання потрібних відомостей;
- Повернені дані агент опрацьовує та віддає результат по HTTP у вигляді HTML-сторінок або JSON;
- Далі власний HTTP-сервер агента публікує ці результати для браузера користувача, забезпечуючи інтерактивну взаємодію.

11. Наведіть і поясніть схему потоків усередині Web-агенту?

- UI/API: приймає запити користувача (фільтри, перегляд);
- Збір даних: web-агент отримує критерії та формує запит до зовнішнього ресурсу (наприклад, веб-сервера, API або бази даних);
- Парсинг: обробляє HTML/JSON у структуровані сутності (новини, матчі);
- Обробка отриманої інформації: фільтрація/сортування (за потреби);
- Передача результатів користувачу (опціонально): надсилає сповіщення або сигналізує UI про виконання завдання;

- Взаємодія з результатами: користувач взаємодіє з отриманим результатом (вибрати потрібну інформації/змінити критерії пошуку тощо).

12. Проаналізуйте основні функції обраного інструментарію розробки Web-агенту: внутрішня структура, параметри, основні змінні, їх призначення та використання.

- HTTP-клієнт;
- Парсинг відповіді: title, extract, thumbnail/originalimage, content_urls.desktop.page; для підказок – title, description, міні-thumbnail;
- Пошук: коми – багато тем. * в кінці (напр. black*) – розгортання через prefixsearch (Топ-N);
- Фільтри/сортування (клієнт): за title; сортування Last Added або A-Z;
- Збереження (кеш): localStorage
 - facts:facts – картки;
 - facts:favorites – обрані результати.
- UI: поле з автодоповненням (клік по підказці = миттєвий пошук), кнопки Search/Update/Export/Import, список карток + секція Favorites.
- Параметри: DOM-ідентифікатори елементів, ключі localStorage.
- Призначення: клієнтський пошук, зручна навігація та збереження результатів пошуку.