Київський національний університет імені Тараса Шевченка

Кафедра безпеки та захисту інформації

Факультет інформаційних технологій

ЗВІТ

Лабораторна робота № 1

З дисципліни «Основи алгоритмізації та програмування»

Роботу виконав: Ткачов Михайло Сергійович

Перевірив: Олексій Ткаченко

Київ 2026

Зміст

Лабораторна робота №1

Фронтенд

Мета роботи: Створити базову сторінку застосунку з HTML-формою та відображенням даних у таблиці/списку. Навчитися працювати з DOM (події, читання значень полів, динамічний рендеринг). Реалізувати мінімальну валідацію введення.

Завдання: Реалізувати сервер: Система пропусків у комп'ютерний клас, а також сутність Users, Passes, Reasons, Logs (8 варіант).

Виконання на рівень «задовільно»

Вимоги:

1. Створити сторінку застосунку (заголовок з назвою варіанта).

2. Реалізувати форму додавання запису ключової сутності (4–5 полів, як вказано в Таблиця 1.1).

2.1. у формі є label для кожного поля;

2.2. кнопки мають type="submit"/type="button" де потрібно;

3. Зберігати записи у масиві JavaScript (in-memory);

4. Відображати записи в таблиці/списку після додавання без перезавантаження сторінки.;

4.1. у таблиці є thead/tbody і th;

5. CSS-мінімум:

5.1. базова розкладка через Flexbox (наприклад, «форма зліва, список справа»);

5.2. box-sizing: border-box, нормальні відступи, читабельність;

6. Мінімальна валідація:

6.1. обов'язкові поля не можуть бути порожніми;

6.2. для Url/Email/Date/Number (якщо є) – базова перевірка формату/діапазону;

6.3. показати користувачу повідомлення про помилку (під формою або біля полів);

6.4. кнопка сабміту блокується або сабміт не проходить з явним поясненням.

Виконання на рівень «задовільно»

1)

```
index.html > html > body > div.container > main.layout > section.card > form#createForm
<!doctype html>
<html lang="uk">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title>Система пропусків — Комп'ютерний клас</title>
  <link rel="stylesheet" href="styles.css" />
  <script src="app.js" defer></script>
</head>
<body>
  <div class="container">
    <header class="topbar">
      <div>
        <h1>Система пропусків</h1>
        <p class="hint">Комп'ютерний клас • Варіант 8 (Passes)</p>
      </div>
      <div class="badge">ЛР1</div>
    </header>

    <main class="layout">
      <section class="card">
        <h2 id="formTitle">Новий пропуск</h2>
        <p class="subhint">Заповни форму — запис одразу з'явиться у списку праворуч.</p>

        <form id="createForm" novalidate>
          <div class="field">
            <label for="userNameInput">Студент / відвідувач</label>
            <input id="userNameInput" type="text" autocomplete="name" />
            <p class="error-text" id="userNameError"></p>
          </div>

          <div class="field">
            <label for="reasonSelect">Причина допуску</label>
            <select id="reasonSelect">
              <option value="">Оберіть зі списку</option>
              <option value="Lab">Лабораторна</option>
              <option value="Exam">Іспит / контроль</option>
```

2)

```
              <option value="Consultation">Консультація</option>
              <option value="Other">Інше</option>
            </select>
            <p class="error-text" id="reasonError"></p>
          </div>

          <div class="field">
            <label for="validDateInput">Дата дії пропуску</label>
            <input id="validDateInput" type="date" />
            <p class="error-text" id="validDateError"></p>
          </div>

          <div class="field">
            <label for="commentInput">Коментар</label>
            <textarea id="commentInput" rows="4"></textarea>
            <p class="error-text" id="commentError"></p>
          </div>

          <div class="field">
            <label for="issuerInput">Хто видав</label>
            <input id="issuerInput" type="text" autocomplete="organization" />
            <p class="error-text" id="issuerError"></p>
          </div>

          <div class="buttons">
            <button id="submitBtn" type="submit">Зберегти</button>
            <button id="resetBtn" type="button">Очистити</button>
          </div>

          <p class="form-msg" id="formMsg"></p>
        </form>
      </section>
```

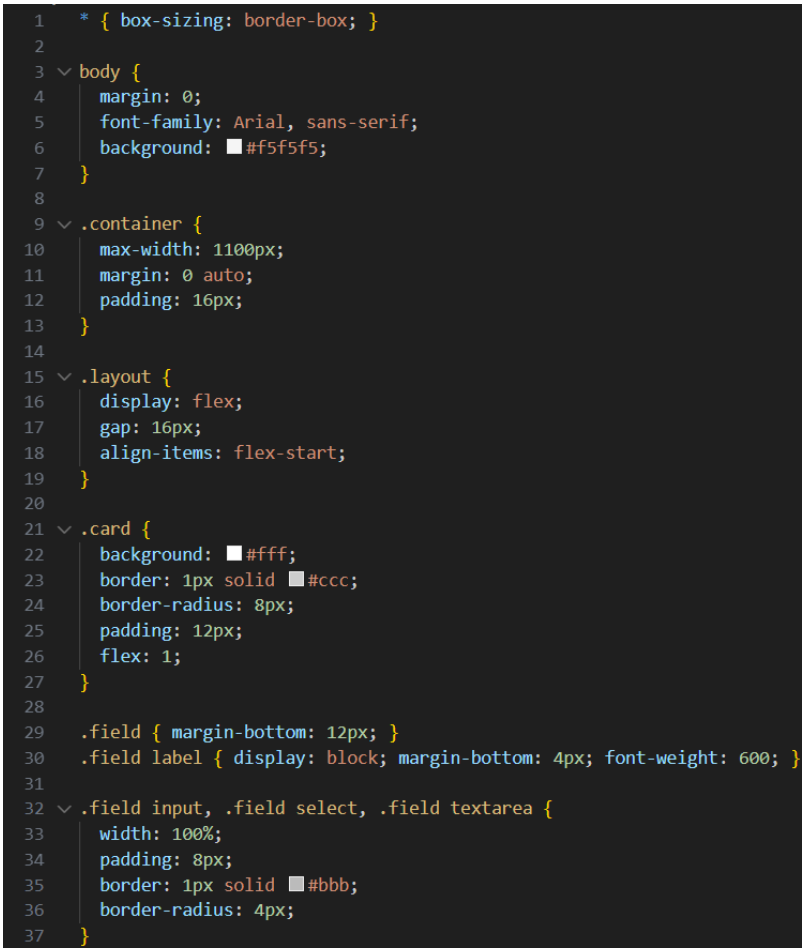
3)

```
<section class="card">
  <h2>Журнал пропусків</h2>
  <p class="subhint">Швидко знайди потрібний запис або відфільтруй.</p>

  <div class="toolbar">
    <input id="searchInput" type="text" placeholder="Пошук за ім'ям..." />
    <select id="filterReason">
      <option value="">Усі причини</option>
      <option value="Lab">Лабораторна</option>
      <option value="Exam">Іспит / контроль</option>
      <option value="Consultation">Консультація</option>
      <option value="Other">Інше</option>
    </select>
    <select id="sortSelect">
      <option value="dateAsc">Дата: спочатку ранні</option>
      <option value="dateDesc">Дата: спочатку пізні</option>
      <option value="userAsc">Ім'я: A→Z</option>
      <option value="userDesc">Ім'я: Z→A</option>
    </select>
  </div>

  <p id="emptyState" class="hint">Поки що порожньо. Додай перший пропуск.</p>

  <table id="itemsTable">
    <thead>
      <tr>
        <th>#</th>
        <th>Студент</th>
        <th>Причина</th>
        <th>Дата</th>
        <th>Хто видав</th>
        <th>Коментар</th>
        <th>Дії</th>
      </tr>
    </thead>
    <tbody id="itemsTableBody"></tbody>
  </table>
</section>
</main>
</div>
</body>
</html>
```

4)

Рисунки 1, 2, 3, 4 — код у HTML для реалізування сервера системи пропусків у комп'ютерний клас, а також сутність Users, Passes, Reasons, Logs

5)



```
* { box-sizing: border-box; }

body {
  margin: 0;
  font-family: Arial, sans-serif;
  background: #f5f5f5;
}

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

.layout {
  display: flex;
  gap: 16px;
  align-items: flex-start;
}

.card {
  background: #fff;
  border: 1px solid #ccc;
  border-radius: 8px;
  padding: 12px;
  flex: 1;
}

.field { margin-bottom: 12px; }
.field label { display: block; margin-bottom: 4px; font-weight: 600; }

.field input, .field select, .field textarea {
  width: 100%;
  padding: 8px;
  border: 1px solid #bbb;
  border-radius: 4px;
}
```

6)

```

.buttons { display: flex; gap: 8px; }

button {
  padding: 8px 12px;
  border: 1px solid #888;
  border-radius: 4px;
  cursor: pointer;
}

.hint { color: #444; }
.error-text { color: #c00; font-size: 12px; min-height: 14px; margin: 4px 0 0; }
.invalid { border-color: #c00; }

table { width: 100%; border-collapse: collapse; }
th, td { border: 1px solid #ccc; padding: 8px; vertical-align: top; }
th { background: #eee; }

@media (max-width: 900px) {
  .layout { flex-direction: column; }
}

```

Рисунки 5 та 6 — код у CSS для реалізування сервера системи пропусків у комп’ютерний клас, а також сутність Users, Passes, Reasons, Logs

7)

```
JS app.js > form.addEventListener("submit") callback
1   "use strict";
2
3
4   let passes = [];
5
6   const form = document.getElementById("createForm");
7
8   const userNameInput = document.getElementById("userNameInput");
9   const reasonSelect = document.getElementById("reasonSelect");
10  const validDateInput = document.getElementById("validDateInput");
11  const commentInput = document.getElementById("commentInput");
12  const issuerInput = document.getElementById("issuerInput");
13
14  const resetBtn = document.getElementById("resetBtn");
15  const formMsg = document.getElementById("formMsg");
16
17  const tbody = document.getElementById("itemsTableBody");
18  const emptyState = document.getElementById("emptyState");
19
20  // init
21  render();
22
23  form.addEventListener("submit", (e) => {
24    e.preventDefault();
25
26    const dto = readForm();
27    if (!validate(dto)) {
28      setMsg("Виправ помилки у формі.");
29      return;
30    }
31
32    passes.push({
33      userName: dto.userName.trim(),
34      reason: dto.reason,
35      validDate: dto.validDate,
36      comment: dto.comment.trim(),
37      issuer: dto.issuer.trim(),
```

8)

```
38    });
39
40    form.reset();
41    clearErrors();
42    setMsg("✅ Додано.");
43    render();
44  });
45
46  resetBtn.addEventListener("click", () => {
47    form.reset();
48    clearErrors();
49    setMsg("");
50  });
51
52  function readForm() {
53    return {
54      userName: userNameInput.value,
55      reason: reasonSelect.value,
56      validDate: validDateInput.value,
57      comment: commentInput.value,
58      issuer: issuerInput.value,
59    };
60  }
61
62  function validate(dto) {
63    clearErrors();
64    let ok = true;
65
66    if (dto.userName.trim() === "") {
67      showError("userNameInput", "userNameError", "UserName обов'язковий.");
68      ok = false;
69    }
70    if (dto.reason === "") {
71      showError("reasonSelect", "reasonError", "Оберіть причину.");
```

9)

```
    showError("reasonSelect", "reasonError", "Оберіть причину.");
    ok = false;
  }
  if (dto.validDate === "") {
    showError("validDateInput", "validDateError", "Вкажіть дату.");
    ok = false;
  }
  if (dto.comment.trim().length < 5) {
    showError("commentInput", "commentError", "Коментар: мінімум 5 символів.");
    ok = false;
  }
  if (dto.issuer.trim() === "") {
    showError("issuerInput", "issuerError", "Issuer обов'язковий.");
    ok = false;
  }

  return ok;
}

function render() {
  emptyState.style.display = passes.length === 0 ? "block" : "none";

  tbody.innerHTML = passes.map((x, i) => `
    <tr>
      <td>${i + 1}</td>
      <td>${escapeHtml(x.userName)}</td>
      <td>${escapeHtml(x.reason)}</td>
      <td>${escapeHtml(x.validDate)}</td>
      <td>${escapeHtml(x.issuer)}</td>
      <td>${escapeHtml(x.comment)}</td>
    </tr>
  `).join("");
}

function setMsg(text) {
```

10)

```
    formMsg.textContent = text;
  }

  function showError(inputId, errorId, msg) {
    document.getElementById(inputId).classList.add("invalid");
    document.getElementById(errorId).textContent = msg;
  }

  function clearErrors() {
    clearOne("userNameInput", "userNameError");
    clearOne("reasonSelect", "reasonError");
    clearOne("validDateInput", "validDateError");
    clearOne("commentInput", "commentError");
    clearOne("issuerInput", "issuerError");
  }

  function clearOne(inputId, errorId) {
    document.getElementById(inputId).classList.remove("invalid");
    document.getElementById(errorId).textContent = "";
  }

  function escapeHtml(text) {
    return String(text)
      .replaceAll("&", "&amp;")
      .replaceAll("<", "&lt;")
      .replaceAll(">", "&gt;")
      .replaceAll('"', "&quot;")
      .replaceAll("'", "&#039;");
  }

```

Рисунки 7, 8, 9, 10 — код у JS для реалізування сервера системи пропусків у комп'ютерний клас, а також сутність Users, Passes, Reasons, Logs

Виконання завдання на рівень «Добре»

Вимоги:

1. Реалізовано видалення елементів зі списку + коректний перерендер;

2. Реалізовано редагування (через форму/модальне/inline) з оновленням state;

3. Дані мають стабільний ідентифікатор (не індекс у масиві як «id»);

4. Код розбитий мінімум на логічні блоки: state, render, handlers (або 2 файли), без «все в одному скрипті»;

5. Додано базовий UX: очищення/фокус форми після успіху, блокування повторного сабміту або захист від дублювання.

11)

```
<!doctype html>
<html lang="uk">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title>Система пропусків — Комп’ютерний клас</title>
  <link rel="stylesheet" href="styles.css" />
  <script src="app.js" defer></script>
</head>

<body>
  <div class="container">
    <header class="topbar">
      <div>
        <h1>Система пропусків</h1>
        <p class="hint">Комп’ютерний клас</p>
      </div>
      <div class="badge">ЛР1</div>
    </header>

    <main class="layout">
      <section class="card">
        <h2 id="formTitle">Новий пропуск</h2>
        <p class="subhint">Заповніть форму для створення пропуску.</p>

        <form id="createForm" novalidate>
          <div class="field">
            <label for="userNameInput">Студент / відвідувач</label>
            <input id="userNameInput" type="text" autocomplete="name" />
            <p class="error-text" id="userNameError"></p>
          </div>

          <div class="field">
            <label for="reasonSelect">Причина допуску</label>
            <select id="reasonSelect">
              <option value="">Оберіть зі списку</option>
              <option value="Lab">Лабораторна</option>
```

12)

```
            <option value="Exam">Іспит / контроль</option>
            <option value="Consultation">Консультація</option>
            <option value="Other">Інше</option>
          </select>
          <p class="error-text" id="reasonError"></p>
        </div>

        <div class="field">
          <label for="validDateInput">Дата дії пропуску</label>
          <input id="validDateInput" type="date" />
          <p class="error-text" id="validDateError"></p>
        </div>

        <div class="field">
          <label for="commentInput">Коментар</label>
          <textarea id="commentInput" rows="4"></textarea>
          <p class="error-text" id="commentError"></p>
        </div>

        <div class="field">
          <label for="issuerInput">Хто видав</label>
          <input id="issuerInput" type="text" autocomplete="organization" />
          <p class="error-text" id="issuerError"></p>
        </div>

        <div class="buttons">
          <button id="submitBtn" type="submit">Зберегти</button>
          <button id="resetBtn" type="button">Очистити</button>
          <button id="cancelEditBtn" type="button" class="ghost" style="display:none;">Скасувати редагування</button>
        </div>
```

13)

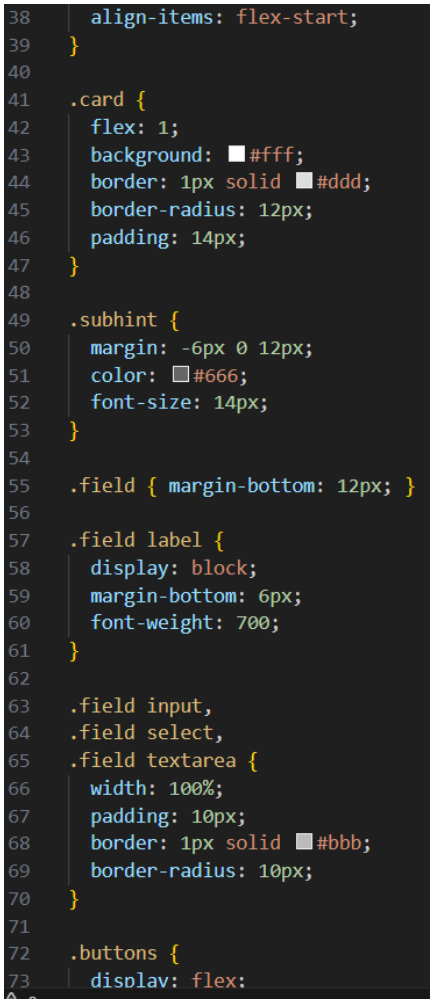
```
            <p class="form-msg" id="formMsg"></p>
          </form>
        </section>

        <section class="card">
          <h2>Журнал пропусків</h2>
          <p class="subhint">Редагуйте або видаляйте записи зі списку.</p>

          <p id="emptyState" class="hint">Поки що порожньо. Додайте перший пропуск.</p>

          <div class="table-wrap">
            <table id="itemsTable">
              <thead>
                <tr>
                  <th>#</th>
                  <th>Студент</th>
                  <th>Причина</th>
                  <th>Дата</th>
                  <th>Хто видав</th>
                  <th>Коментар</th>
                  <th>Дії</th>
                </tr>
              </thead>
              <tbody id="itemsTableBody"></tbody>
            </table>
          </div>
        </section>
      </main>
    </div>
  </body>
</html>
```

Рисунки 11, 12, 13 — код в HTML варіанту 8 на р. «Добре»

14)

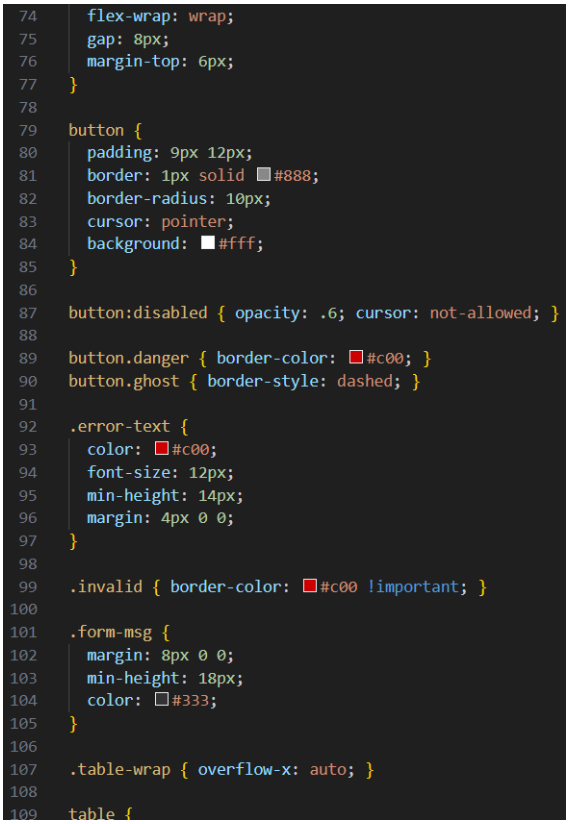
```css
* { box-sizing: border-box; }

body {
  margin: 0;
  font-family: Arial, sans-serif;
  background: #f5f5f5;
  color: #111;
}

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

.topbar {
  display: flex;
  justify-content: space-between;
  align-items: center;
  gap: 12px;
  margin-bottom: 14px;
}

h1 { margin: 0; font-size: 32px; }
.hint { margin: 6px 0 0; color: #555; }

.badge {
  padding: 8px 12px;
  border: 1px solid #ccc;
  border-radius: 999px;
  background: #fff;
  font-weight: 700;
}

.layout {
  display: flex;
  gap: 16px;
```

15)



```css
  align-items: flex-start;
}

.card {
  flex: 1;
  background: #fff;
  border: 1px solid #ddd;
  border-radius: 12px;
  padding: 14px;
}

.subhint {
  margin: -6px 0 12px;
  color: #666;
  font-size: 14px;
}

.field { margin-bottom: 12px; }

.field label {
  display: block;
  margin-bottom: 6px;
  font-weight: 700;
}

.field input,
.field select,
.field textarea {
  width: 100%;
  padding: 10px;
  border: 1px solid #bbb;
  border-radius: 10px;
}

.buttons {
  display: flex;
```

16)



```
  flex-wrap: wrap;
  gap: 8px;
  margin-top: 6px;
}

button {
  padding: 9px 12px;
  border: 1px solid #888;
  border-radius: 10px;
  cursor: pointer;
  background: #fff;
}

button:disabled { opacity: .6; cursor: not-allowed; }

button.danger { border-color: #c00; }
button.ghost { border-style: dashed; }

.error-text {
  color: #c00;
  font-size: 12px;
  min-height: 14px;
  margin: 4px 0 0;
}

.invalid { border-color: #c00 !important; }

.form-msg {
  margin: 8px 0 0;
  min-height: 18px;
  color: #333;
}

.table-wrap { overflow-x: auto; }

table {
```

17)

```
    width: 100%;
    border-collapse: collapse;
  }

  th, td {
    border: 1px solid #ddd;
    padding: 8px;
    vertical-align: top;
  }

  th { background: #eee; }

  @media (max-width: 900px) {
    .layout { flex-direction: column; }
  }
```

Рисунки 14, 15, 16, і 17 — код в HTML варіанту 8 на р. «Добре»

18)

```
const state = {
  items: [],
  editId: null,
  isSubmitting: false
};

function createId() {
  return crypto.randomUUID
    ? crypto.randomUUID()
    : "id_" + Date.now() + "_" + Math.random().toString(16).slice(2);
}


const form = document.getElementById("createForm");
const tbody = document.getElementById("itemsTableBody");
const emptyState = document.getElementById("emptyState");

const userNameInput = document.getElementById("userNameInput");
const reasonSelect = document.getElementById("reasonSelect");
const validDateInput = document.getElementById("validDateInput");
const commentInput = document.getElementById("commentInput");
const issuerInput = document.getElementById("issuerInput");

const submitBtn = document.getElementById("submitBtn");
const resetBtn = document.getElementById("resetBtn");

const formTitle = document.getElementById("formTitle");
const formMsg = document.getElementById("formMsg");


function render() {
  emptyState.style.display = state.items.length === 0 ? "block" : "none";

  tbody.innerHTML = state.items.map((item, index) => `
    <tr>
      <td>${index + 1}</td>
      <td>${item.userName}</td>
```

19)

```
      <td>${item.reason}</td>
      <td>${item.validDate}</td>
      <td>${item.issuer}</td>
      <td>${item.comment}</td>
      <td>
        <button data-id="${item.id}" class="edit-btn">Редагувати</button>
        <button data-id="${item.id}" class="delete-btn">Видалити</button>
      </td>
    </tr>
  `).join("");

  submitBtn.disabled = state.isSubmitting;
  formTitle.textContent = state.editId ? "Редагувати пропуск" : "Новий пропуск";
}


form.addEventListener("submit", function (e) {
  e.preventDefault();
  if (state.isSubmitting) return;

  const userName = userNameInput.value.trim();
  const reason = reasonSelect.value;
  const validDate = validDateInput.value;
  const comment = commentInput.value.trim();
  const issuer = issuerInput.value.trim();

  if (!userName || !reason || !validDate || !issuer) {
    formMsg.textContent = "Заповніть всі обов'язкові поля.";
    return;
  }

  state.isSubmitting = true;
  render();

  if (state.editId === null) {
```

20)

```
    state.items.push({
      id: createId(),
      userName,
      reason,
      validDate,
      comment,
      issuer
    });
    formMsg.textContent = "Запис додано.";
  } else {

    const item = state.items.find(x => x.id === state.editId);
    if (item) {
      item.userName = userName;
      item.reason = reason;
      item.validDate = validDate;
      item.comment = comment;
      item.issuer = issuer;
      formMsg.textContent = "Зміни збережено.";
    }
    state.editId = null;
  }

  form.reset();
  userNameInput.focus();

  state.isSubmitting = false;
  render();
});

resetBtn.addEventListener("click", function () {
  form.reset();
  state.editId = null;
  formMsg.textContent = "";
  render();
});
```

21)

```
tbody.addEventListener("click", function (e) {
  const id = e.target.dataset.id;
  if (!id) return;

  if (e.target.classList.contains("delete-btn")) {
    state.items = state.items.filter(item => item.id !== id);
    formMsg.textContent = "Запис видалено.";
    render();
  }

  if (e.target.classList.contains("edit-btn")) {
    const item = state.items.find(item => item.id === id);
    if (!item) return;

    userNameInput.value = item.userName;
    reasonSelect.value = item.reason;
    validDateInput.value = item.validDate;
    commentInput.value = item.comment;
    issuerInput.value = item.issuer;

    state.editId = id;
    userNameInput.focus();
    render();
  }
});
```

Рисунки 18, 19 ,20 і 21 — код в JS варіанту 8 на р. «Добре»

Виконання на рівень «відмінно»

Вимоги:

1. Делегування подій для елементів списку/таблиці (без навішування обробників на кожну кнопку при кожному рендері);

2. Фільтр або пошук по списку (хоч один варіант) + коректна робота з state;

3. Сортування (мінімум за одним полем) без поломки редагування/видалення;

4. Персистентність: localStorage (load/save) або інший дозволений спосіб збереження стану.

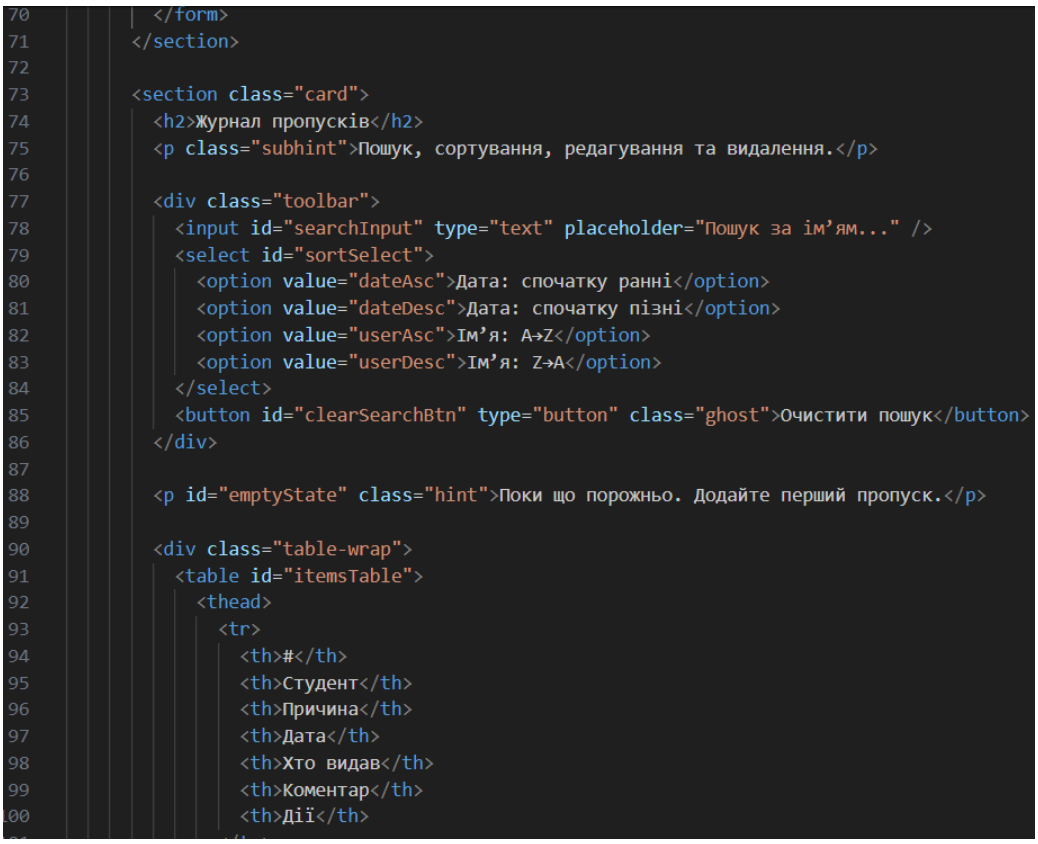
22)

```
<!doctype html>
<html lang="uk">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title>Система пропусків — Комп’ютерний клас</title>
  <link rel="stylesheet" href="styles.css" />
  <script src="app.js" defer></script>
</head>

<body>
  <div class="container">
    <header class="topbar">
      <div>
        <h1>Система пропусків</h1>
        <p class="hint">Комп’ютерний клас</p>
      </div>
      <div class="badge">ЛР1</div>
    </header>

    <main class="layout">
      <section class="card">
        <h2 id="formTitle">Новий пропуск</h2>
        <p class="subhint">Заповніть форму для створення пропуску.</p>

        <form id="createForm" novalidate>
          <div class="field">
            <label for="userNameInput">Студент / відвідувач</label>
            <input id="userNameInput" type="text" autocomplete="name" />
            <p class="error-text" id="userNameError"></p>
          </div>

          <div class="field">
            <label for="reasonSelect">Причина допуску</label>
            <select id="reasonSelect">
              <option value="">Оберіть зі списку</option>
              <option value="Lab">Лабораторна</option>
```

23)

```
        <option value="Exam">Іспит / контроль</option>
        <option value="Consultation">Консультація</option>
        <option value="Other">Інше</option>
      </select>
      <p class="error-text" id="reasonError"></p>
    </div>

    <div class="field">
      <label for="validDateInput">Дата дії пропуску</label>
      <input id="validDateInput" type="date" />
      <p class="error-text" id="validDateError"></p>
    </div>

    <div class="field">
      <label for="commentInput">Коментар</label>
      <textarea id="commentInput" rows="4"></textarea>
      <p class="error-text" id="commentError"></p>
    </div>

    <div class="field">
      <label for="issuerInput">Хто видав</label>
      <input id="issuerInput" type="text" autocomplete="organization" />
      <p class="error-text" id="issuerError"></p>
    </div>

    <div class="buttons">
      <button id="submitBtn" type="submit">Зберегти</button>
      <button id="resetBtn" type="button">Очистити</button>
      <button id="cancelEditBtn" type="button" class="ghost" style="display:none;">Скасувати редагування</button>
    </div>

    <p class="form-msg" id="formMsg"></p>
```

24)



```
    </form>
  </section>

  <section class="card">
    <h2>Журнал пропусків</h2>
    <p class="subhint">Пошук, сортування, редагування та видалення.</p>

    <div class="toolbar">
      <input id="searchInput" type="text" placeholder="Пошук за ім'ям..." />
      <select id="sortSelect">
        <option value="dateAsc">Дата: спочатку ранні</option>
        <option value="dateDesc">Дата: спочатку пізні</option>
        <option value="userAsc">Ім'я: A→Z</option>
        <option value="userDesc">Ім'я: Z→A</option>
      </select>
      <button id="clearSearchBtn" type="button" class="ghost">Очистити пошук</button>
    </div>

    <p id="emptyState" class="hint">Поки що порожньо. Додайте перший пропуск.</p>

    <div class="table-wrap">
      <table id="itemsTable">
        <thead>
          <tr>
            <th>#</th>
            <th>Студент</th>
            <th>Причина</th>
            <th>Дата</th>
            <th>Хто видав</th>
            <th>Коментар</th>
            <th>Дії</th>
```

Рисунки 22, 23 і 24 — код в HTML варіанту 8 на р. «Відмінно»

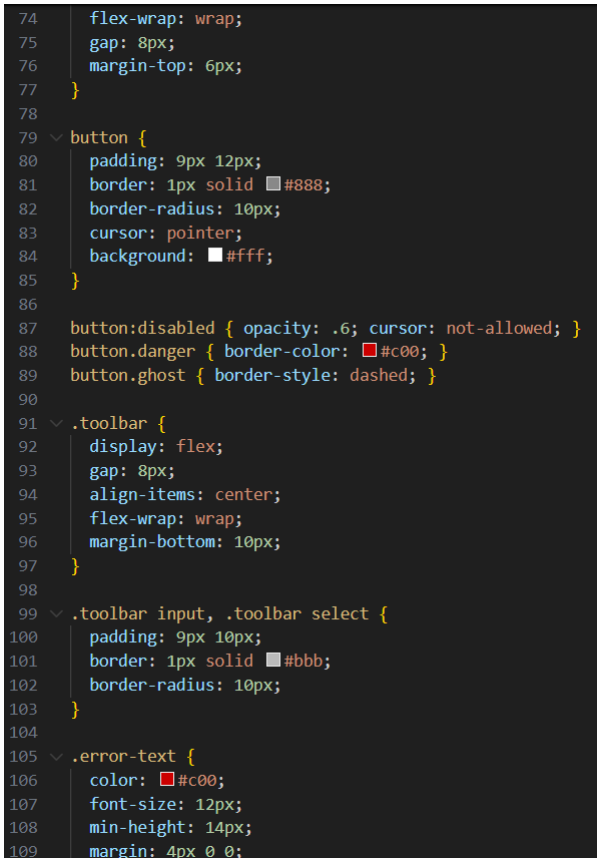
25)

```css
* { box-sizing: border-box; }

body {
  margin: 0;
  font-family: Arial, sans-serif;
  background: #f5f5f5;
  color: #111;
}

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

.topbar {
  display: flex;
  justify-content: space-between;
  align-items: center;
  gap: 12px;
  margin-bottom: 14px;
}

h1 { margin: 0; font-size: 32px; }
.hint { margin: 6px 0 0; color: #555; }

.badge {
  padding: 8px 12px;
  border: 1px solid #ccc;
  border-radius: 999px;
  background: #fff;
  font-weight: 700;
}

.layout {
  display: flex;
  gap: 16px;
```

26)

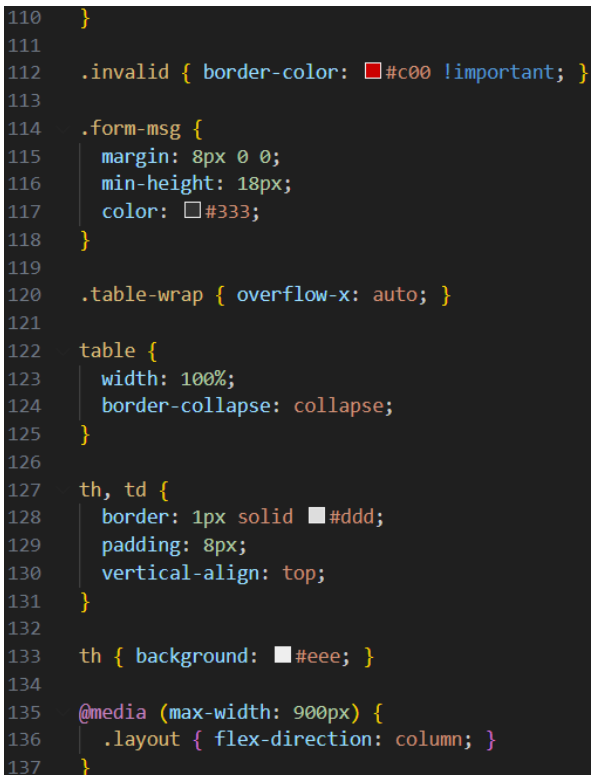
```css
  align-items: flex-start;
}

.card {
  flex: 1;
  background: #fff;
  border: 1px solid #ddd;
  border-radius: 12px;
  padding: 14px;
}

.subhint {
  margin: -6px 0 12px;
  color: #666;
  font-size: 14px;
}

.field { margin-bottom: 12px; }

.field label {
  display: block;
  margin-bottom: 6px;
  font-weight: 700;
}

.field input,
.field select,
.field textarea {
  width: 100%;
  padding: 10px;
  border: 1px solid #bbb;
  border-radius: 10px;
}

.buttons {
  display: flex;
```

27)



```
    flex-wrap: wrap;
    gap: 8px;
    margin-top: 6px;
  }

  button {
    padding: 9px 12px;
    border: 1px solid #888;
    border-radius: 10px;
    cursor: pointer;
    background: #fff;
  }

  button:disabled { opacity: .6; cursor: not-allowed; }
  button.danger { border-color: #c00; }
  button.ghost { border-style: dashed; }

  .toolbar {
    display: flex;
    gap: 8px;
    align-items: center;
    flex-wrap: wrap;
    margin-bottom: 10px;
  }

  .toolbar input, .toolbar select {
    padding: 9px 10px;
    border: 1px solid #bbb;
    border-radius: 10px;
  }

  .error-text {
    color: #c00;
    font-size: 12px;
    min-height: 14px;
    margin: 4px 0 0;
```

28)



```
  }

  .invalid { border-color: #c00 !important; }

  .form-msg {
    margin: 8px 0 0;
    min-height: 18px;
    color: #333;
  }

  .table-wrap { overflow-x: auto; }

  table {
    width: 100%;
    border-collapse: collapse;
  }

  th, td {
    border: 1px solid #ddd;
    padding: 8px;
    vertical-align: top;
  }

  th { background: #eee; }

  @media (max-width: 900px) {
    .layout { flex-direction: column; }
  }
```

Рисунки 25, 26, 27 і 28 — код в CSS варіанту 8 на р. «Відмінно»

29)

```
    };
    localStorage.setItem(STORAGE_KEY, JSON.stringify(payload));
  }


const dom = {
    form: document.getElementById("createForm"),

    formTitle: document.getElementById("formTitle"),
    formMsg: document.getElementById("formMsg"),

    userNameInput: document.getElementById("userNameInput"),
    reasonSelect: document.getElementById("reasonSelect"),
    validDateInput: document.getElementById("validDateInput"),
    commentInput: document.getElementById("commentInput"),
    issuerInput: document.getElementById("issuerInput"),

    userNameError: document.getElementById("userNameError"),
    reasonError: document.getElementById("reasonError"),
    validDateError: document.getElementById("validDateError"),
    commentError: document.getElementById("commentError"),
    issuerError: document.getElementById("issuerError"),

    submitBtn: document.getElementById("submitBtn"),
    resetBtn: document.getElementById("resetBtn"),
    cancelEditBtn: document.getElementById("cancelEditBtn"),

    searchInput: document.getElementById("searchInput"),
    sortSelect: document.getElementById("sortSelect"),
    clearSearchBtn: document.getElementById("clearSearchBtn"),

    emptyState: document.getElementById("emptyState"),
    tbody: document.getElementById("itemsTableBody"),
  };
```

30)

```
function esc(v) {
  return String(v)
    .replaceAll("&", "&amp;")
    .replaceAll("<", "&lt;")
    .replaceAll(">", "&gt;")
    .replaceAll('"', "&quot;")
    .replaceAll("'", "&#039;");
}

function getViewItems() {
  const q = state.searchQuery.trim().toLowerCase();
  let list = state.items;

  if (q) {
    list = list.filter(x => (x.userName || "").toLowerCase().includes(q));
  }

  const sorted = [...list];

  sorted.sort((a, b) => {
    const mode = state.sortMode;

    if (mode === "userAsc" || mode === "userDesc") {
      const av = (a.userName || "").toLowerCase();
      const bv = (b.userName || "").toLowerCase();
      const cmp = av.localeCompare(bv, "uk");
      return mode === "userAsc" ? cmp : -cmp;
    }

    const ad = a.validDate || "";
    const bd = b.validDate || "";
    const cmp = ad.localeCompare(bd);
    return mode === "dateAsc" ? cmp : -cmp;
  });

  return sorted;
```

31)

```
}

function render() {
  const isEdit = state.editId !== null;
  dom.formTitle.textContent = isEdit ? "Редагувати пропуск" : "Новий пропуск";
  dom.submitBtn.textContent = isEdit ? "Зберегти зміни" : "Зберегти";
  dom.cancelEditBtn.style.display = isEdit ? "inline-block" : "none";
  dom.submitBtn.disabled = state.isSubmitting;

  dom.searchInput.value = state.searchQuery;
  dom.sortSelect.value = state.sortMode;

  const view = getViewItems();

  dom.emptyState.style.display = view.length === 0 ? "block" : "none";

  dom.tbody.innerHTML = view.map((x, i) => `
    <tr data-row-id="${esc(x.id)}">
      <td>${i + 1}</td>
      <td>${esc(x.userName)}</td>
      <td>${esc(x.reason)}</td>
      <td>${esc(x.validDate)}</td>
      <td>${esc(x.issuer)}</td>
      <td>${esc(x.comment)}</td>
      <td>
        <button type="button" class="edit-btn" data-id="${esc(x.id)}">Редагувати</button>
        <button type="button" class="delete-btn danger" data-id="${esc(x.id)}">Видалити</button>
      </td>
    </tr>
  `).join("");
}


function setFormMessage(msg) {
  dom.formMsg.textContent = msg;
}

```

32)

```
function showError(control, errorEl, msg) {
  control.classList.add("invalid");
  errorEl.textContent = msg;
}

function clearErrors() {
  [
    dom.userNameInput,
    dom.reasonSelect,
    dom.validDateInput,
    dom.commentInput,
    dom.issuerInput,
  ].forEach(el => el.classList.remove("invalid"));

  [
    dom.userNameError,
    dom.reasonError,
    dom.validDateError,
    dom.commentError,
    dom.issuerError,
  ].forEach(el => (el.textContent = ""));
}

function readForm() {
  return {
    userName: dom.userNameInput.value,
    reason: dom.reasonSelect.value,
    validDate: dom.validDateInput.value,
    comment: dom.commentInput.value,
    issuer: dom.issuerInput.value,
  };
}

function validate(dto) {
  let ok = true;

  if (dto.userName.trim() === "") {
```

33)

```
    showError(dom.userNameInput, dom.userNameError, "Вкажіть ім'я.");
    ok = false;
  }
  if (dto.reason === "") {
    showError(dom.reasonSelect, dom.reasonError, "Оберіть причину.");
    ok = false;
  }
  if (dto.validDate === "") {
    showError(dom.validDateInput, dom.validDateError, "Оберіть дату.");
    ok = false;
  }
  if (dto.comment.trim().length < 3) {
    showError(dom.commentInput, dom.commentError, "Коментар має бути хоча б 3 символи.");
    ok = false;
  }
  if (dto.issuer.trim() === "") {
    showError(dom.issuerInput, dom.issuerError, "Вкажіть, хто видав.");
    ok = false;
  }

  return ok;
}


function addItem(dto) {
  state.items.push({
    id: createId(),
    userName: dto.userName.trim(),
    reason: dto.reason,
    validDate: dto.validDate,
    comment: dto.comment.trim(),
    issuer: dto.issuer.trim(),
  });
}

function updateItem(id, dto) {
```

34)

```
function updateItem(id, dto) {
  const idx = state.items.findIndex(x => x.id === id);
  if (idx === -1) return false;

  state.items[idx] = {
    ...state.items[idx],
    userName: dto.userName.trim(),
    reason: dto.reason,
    validDate: dto.validDate,
    comment: dto.comment.trim(),
    issuer: dto.issuer.trim(),
  };
  return true;
}

function removeItem(id) {
  const before = state.items.length;
  state.items = state.items.filter(x => x.id !== id);
  if (state.editId === id) state.editId = null;
  return state.items.length !== before;
}

function startEdit(id) {
  const item = state.items.find(x => x.id === id);
  if (!item) return;

  state.editId = id;

  dom.userNameInput.value = item.userName;
  dom.reasonSelect.value = item.reason;
  dom.validDateInput.value = item.validDate;
  dom.commentInput.value = item.comment;
  dom.issuerInput.value = item.issuer;

  clearErrors();
  setFormMessage("🖍️ Режим редагування.");
```

35)

```
    setFormMessage("Режим редагування.");
    dom.userNameInput.focus();
    render();
  }

  function cancelEdit() {
    state.editId = null;
    dom.form.reset();
    clearErrors();
    setFormMessage("Редагування скасовано.");
    dom.userNameInput.focus();
    render();
  }


  function onSubmit(e) {
    e.preventDefault();
    if (state.isSubmitting) return;

    clearErrors();
    setFormMessage("");

    const dto = readForm();
    if (!validate(dto)) {
      setFormMessage("Виправте помилки у формі.");
      return;
    }

    state.isSubmitting = true;
    render();

    try {
      if (state.editId === null) {
        addItem(dto);
        setFormMessage("Запис додано.");
      } else {
```

36)

```
        const ok = updateItem(state.editId, dto);
        setFormMessage(ok ? "Зміни збережено." : "Запис не знайдено.");
        state.editId = null;
      }

      saveState();

      dom.form.reset();
      dom.userNameInput.focus();
      clearErrors();
    } finally {
      state.isSubmitting = false;
      render();
    }
  }

  function onReset() {
    dom.form.reset();
    clearErrors();
    setFormMessage("");
    dom.userNameInput.focus();
  }

  function onTableClick(e) {
    const target = e.target;
    if (!(target instanceof HTMLElement)) return;

    const id = target.dataset.id;
    if (!id) return;

    if (target.classList.contains("delete-btn")) {
      const removed = removeItem(id);
      if (removed) {
        saveState();
        setFormMessage("Запис видалено.");
        render();
```

37)

```
        render();
      }
      return;
    }

    if (target.classList.contains("edit-btn")) {
      startEdit(id);
      return;
    }
  }

  function onSearchInput() {
    state.searchQuery = dom.searchInput.value;
    saveState();
    render();
  }

  function onClearSearch() {
    state.searchQuery = "";
    dom.searchInput.value = "";
    saveState();
    render();
    dom.searchInput.focus();
  }

  function onSortChange() {
    state.sortMode = dom.sortSelect.value;
    saveState();
    render();
  }



  (function init() {
    loadState();
```

38)

```

    dom.form.addEventListener("submit", onSubmit);
    dom.resetBtn.addEventListener("click", onReset);
    dom.cancelEditBtn.addEventListener("click", cancelEdit);

    dom.tbody.addEventListener("click", onTableClick);

    dom.searchInput.addEventListener("input", onSearchInput);
    dom.clearSearchBtn.addEventListener("click", onClearSearch);
    dom.sortSelect.addEventListener("change", onSortChange);

    render();
    dom.userNameInput.focus();
  })();

```

Рисунки 29 – 30 — код в JS варіанту 8 на р. «Відмінно»

Опис структури інтерфейсу HTML (задовільно)

1. Загальна структура сторінки

Веб-сторінка реалізована у вигляді односторінкового інтерфейсу системи обліку пропусків. Структура сторінки складається з шапки та основної частини, яка містить форму введення даних і журнал пропусків.

2. Шапка сторінки (topbar)

У шапці сторінки відображається назва системи, додаткова інформація про лабораторну роботу та номер варіанту. Також присутня візуальна позначка лабораторної роботи, що допомагає ідентифікувати проєкт.

3. Основний макет сторінки (layout)

Основна частина сторінки поділена на дві логічні панелі, розташовані поруч: форму створення пропуску та журнал пропусків. Така структура дозволяє одночасно вводити дані та переглядати результати.

4. Форма створення пропуску (createForm)

Форма призначена для введення та редагування інформації про пропуск. Вона містить поля для введення імені студента, причини допуску, дати дії, коментаря та інформації про особу, що видала пропуск. Перевірка коректності введених даних виконується програмно.

5. Валідація та повідомлення

Для кожного поля форми передбачено окреме місце для відображення повідомлень про помилки. У нижній частині форми відображаються службові повідомлення про результат виконання дій (додавання або збереження запису).

6. Панель керування журналом (toolbar)

Над таблицею журналу розміщена панель керування, яка дозволяє виконувати пошук записів за іменем, фільтрацію за причиною допуску та сортування за датою або ім’ям.

7. Журнал пропусків

Журнал пропусків представлений у вигляді таблиці, де кожен рядок відповідає одному пропуску. Таблиця містить основну інформацію про пропуск та дії для редагування або видалення записів.

8. Динамічне оновлення даних

Вміст журналу оновлюється динамічно відповідно до поточного стану даних без перезавантаження сторінки. У разі відсутності записів відображається повідомлення про порожній журнал.

Опис стилів та макету інтерфейсу CSS (задовільно)

1. Загальні стилі сторінки

Загальні стилі визначають базовий вигляд веб-додатку: шрифт, фоновий колір сторінки та базові відступи. Це створює нейтральний, читабельний інтерфейс без зайвих візуальних об'єктів.

2. Контейнер сторінки (container)

Контейнер використовується для центрування контенту та обмеження максимальної ширини сторінки. Це запобігає надмірному розтягуванню інтерфейсу на великих екранах і робить сторінку зручною для сприйняття.

3. Шапка сторінки (topbar)

Шапка сторінки оформлена як горизонтальний блок, у якому елементи вирівняні по ширині. Це дозволяє чітко розділити основну назву системи та службову інформацію.

4. Основний макет сторінки (layout)

Основний макет реалізований у вигляді двоколонкової структури. Ліва колонка призначена для форми введення даних, права — для журналу пропусків. Такий підхід забезпечує зручну роботу з даними без необхідності перемикання між екранами.

5. Панелі інтерфейсу (card)

Панелі інтерфейсу використовуються для візуального відокремлення логічних частин сторінки. Кожна панель має власний фон, рамку та заокруглені кути, що робить структуру сторінки наочною і впорядкованою.

6. Поля введення та підписи (field)

Поля введення згруповані таким чином, щоб кожне поле мало чіткий підпис, елемент введення та місце для відображення помилки. Це забезпечує логічну послідовність введення даних і покращує зручність користування формою.

7. Кнопки керування

Кнопки керування оформлені в єдиному стилі та згруповані разом. Візуальне розрізнення кнопок дозволяє інтуїтивно зрозуміти їх призначення, а неактивні стани додатково сигналізують про неможливість виконання дії.

8. Панель керування журналом (toolbar)

Панель керування журналом розташована над таблицею та містить елементи для пошуку, фільтрації та сортування записів. Стилізація забезпечує компактне розміщення елементів без перевантаження інтерфейсу.

9. Таблиця журналу пропусків

Таблиця оформлена у класичному табличному вигляді з чітким розділенням заголовків і рядків. Таке оформлення полегшує перегляд великої кількості записів та швидке порівняння даних.

10. Повідомлення та помилки

Для повідомлень про помилки та службових повідомлень використовується окреме візуальне оформлення. Помилки підсвічуються, що допомагає користувачу швидко знайти та виправити некоректно введені дані.

11. Адаптивність інтерфейсу

Інтерфейс адаптований до різних розмірів екранів. На вузьких екранах двоколонковий макет автоматично перетворюється на одноколонковий, що забезпечує коректне відображення сторінки на мобільних пристроях.

Опис логіки роботи програми JavaScript (задовільно)

1. Загальна роль JavaScript у проєкті

JavaScript відповідає за логіку роботи веб-додатку. Він забезпечує обробку дій користувача, перевірку введених даних, керування списком пропусків та динамічне оновлення інтерфейсу без перезавантаження сторінки.

2. Зберігання та структура даних

Інформація про пропуски зберігається у внутрішній структурі даних у вигляді списку записів.
Кожен запис містить дані про студента, причину допуску, дату дії, коментар та особу, що видала пропуск.
Для ідентифікації записів використовується унікальний ідентифікатор.

3. Додавання нового пропуску

При натисканні кнопки збереження JavaScript зчитує дані з форми, перевіряє їх коректність та формує новий запис.
Після успішної перевірки запис додається до списку, а форма очищується для подальшого введення даних.

4. Валідація введених даних

Перед додаванням або збереженням запису виконується перевірка правильності введених даних.
У разі помилки відповідні поля підсвічуються, а під ними відображаються повідомлення з поясненням, що саме потрібно виправити.

5. Відображення журналу пропусків

Відображення журналу реалізовано динамічно.
Після кожної зміни даних (додавання, редагування, видалення) список пропусків оновлюється, і таблиця відображає актуальний стан даних без перезавантаження сторінки.

6. Редагування та видалення записів

Для кожного запису в журналі передбачені дії редагування та видалення.
Редагування переводить форму у відповідний режим із підстановкою поточних даних запису.

Видалення прибирає запис зі списку та оновлює журнал.

7. Пошук, фільтрація та сортування

Для зручності користувача реалізовано механізми:

- пошуку записів за ім’ям студента,
- фільтрації за причиною допуску,
- сортування записів за датою або за ім’ям.

Ці операції змінюють лише відображення даних, не порушуючи їх структуру.

8. Делегування подій

Обробка дій користувача (редагування, видалення) реалізована за допомогою делегування подій.
Це дозволяє коректно обробляти натискання по елементах, які створюються динамічно під час оновлення таблиці.

9. Збереження даних у локальному сховищі

Для збереження даних між перезавантаженнями сторінки використовується локальне сховище браузера.
При завантаженні сторінки дані автоматично відновлюються, що забезпечує безперервність роботи з системою.

10. Системні повідомлення

JavaScript керує відображенням службових повідомлень для користувача, таких як успішне додавання, збереження або повідомлення про помилки. Це робить взаємодію з інтерфейсом більш зрозумілою.

Рівень «Добре»

У порівнянні з базовою реалізацією, функціональність системи була розширена.

1. Реалізовано видалення записів: Для кожного пропуску в журналі передбачено кнопку видалення. Після видалення елементу список коректно перерендерюється без перезавантаження сторінки.
2. Реалізовано редагування записів через форму: При натисканні кнопки редагування дані вибраного запису автоматично підставляються у форму. Після збереження зміни оновлюють відповідний запис у внутрішньому стані системи.

3. Використання стабільного ідентифікатора: Кожен запис має унікальний ідентифікатор, який не залежить від позиції в масиві. Це дозволяє коректно виконувати операції редагування та видалення незалежно від порядку відображення записів.
4. Логічна структура коду: Програма організована у вигляді логічних блоків:

   зберігання стану (state),

   відображення інтерфейсу (render),

   обробники подій (handlers).

   Такий підхід підвищує читабельність і підтримуваність коду.
5. Покращений UX: Після успішного додавання або редагування форма очищується, встановлюється фокус на перше поле, а також реалізовано захист від повторного сабміту.

Рівень «Відмінно»

Подальше розширення функціональності дозволило реалізувати більш складну логіку роботи з даними.

1. Делегування подій: Обробка натискань на кнопки редагування та видалення реалізована через делегування подій. Це означає, що використовується один обробник для всієї таблиці, що забезпечує коректну роботу з динамічно сформованими елементами.
2. Пошук по списку: Реалізовано можливість пошуку записів за ім’ям студента. Пошук здійснюється шляхом фільтрації даних на основі поточного стану програми без зміни внутрішньої структури масиву.

   Дані зберігаються у внутрішньому стані програми та додатково синхронізуються з локальним сховищем браузера, що забезпечує збереження інформації після перезавантаження сторінки.
3. Сортування записів: Реалізовано сортування записів за датою або за іменем. Сортування не впливає на внутрішній стан даних і не порушує механізм редагування та видалення, оскільки всі операції виконуються за унікальним ідентифікатором.

4. Персистентність даних (localStorage): Дані зберігаються у локальному сховищі браузера. Після перезавантаження сторінки список пропусків автоматично відновлюється, що забезпечує збереження стану системи.

Висновок: під час виконання лабораторної роботи було реалізовано односторінковий веб-додаток для обліку пропусків у комп'ютерний клас. Система забезпечує створення, редагування та видалення записів, а також підтримує пошук, сортування та збереження даних у локальному сховищі браузера. Було застосовано принцип зберігання стану (state), делегування подій та динамічне оновлення інтерфейсу без перезавантаження сторінки. Реалізація дозволяє працювати з даними зручно та стабільно, а структура коду забезпечує його читабельність і подальшу розширюваність.