MENEDŻERSKA AKADEMIA NAUK STOSOWANYCH W WARSZAWIE

Application of web systems in information systems

Meeting 5 – Lab

16.05.2025 9:45 – 11:30

Meeting agenda

- Research, Low-Fidelity Prototyping, and First Testing
- Create Low-Fidelity Wireframes
- Peer Review and First Usability Test
- Iterative Improvement of Wireframes
- Project Setup, Component Architecture, and Initial Implementation

Meeting 5 - Objective

Extend your working system by adding a reporting dashboard that visualizes key data metrics and allows end-users to export current data. You will pull real data from your backend, render charts in the browser, and implement a simple “Download” action.

Part I - Identify Key Metrics

Pick **two meaningful metrics** from your data model — e.g.:

- Monthly count of bookings/records
- Average processing time per entry
- Distribution of users by role/type

- Chart types overview: https://www.chartjs.org/docs/latest/charts/
- When to use each chart: https://www.data-to-viz.com/

Deliverable: List of two metrics and chart types chosen.

Part II - Add a “Dashboard” Route and Layout

- Define the Route - In your main router file (e.g. src/App.jsx or src/router/index.js), import your new Dashboard component: *import Dashboard from './pages/Dashboard.jsx';*
- Add a route entry:

```
<Routes>
        {/* existing routes */}
        <Route path="/dashboard" element={<Dashboard />} />
</Routes>
```

- Scaffold the Page Component - Create src/pages/Dashboard.jsx with a basic function (placeholder for functions).
- Wire It Into Your Layout - If you have a shared layout (e.g. src/components/Layout.jsx), ensure it includes a link to “Dashboard” in your navbar/sidebar.
- Verify - Start your dev server (npm run dev) and navigate to http://localhost:3000/dashboard. You should see placeholders and the button.

Deliverable: Empty /dashboard page in your app.

Part III - Fetch Data from Backend

- **Decide on Endpoints,** For example:
  - GET /api/bookings/stats/monthly-count
  - GET /api/users/stats/by-role
- Create a Data Service Module:
- Hook or State in Dashboar - In Dashboard.jsx, import and call these functions inside a useEffect:
- Error Handling & Loading State
  - Add a loading spinner or “Loading…” text until the promise resolves.
  - Surround API calls with try/catch and show an error message if it fails.

Deliverable: JS functions or hooks that return { labels: [], data: [] } for each metric.

```jsx
import { useState, useEffect } from 'react';
import { fetchMonthlyCounts, fetchUsersByRole } from '../services/api';

export default function Dashboard() {
  const [chartData1, setChartData1] = useState({ labels: [], values: [] });
  const [chartData2, setChartData2] = useState({ labels: [], values: [] });

  useEffect(() => {
    fetchMonthlyCounts().then(setChartData1);
    fetchUsersByRole().then(setChartData2);
  }, []);

  // …render charts using chartData1/chartData2…
}
```

```jsx
import axios from 'axios';
const api = axios.create({ baseURL: import.meta.env.VITE_API_URL });

export async function fetchMonthlyCounts() {
  const { data } = await api.get('/bookings/stats/monthly-count');
  // data example: [{ month: '2025-01', count: 12 }, …]
  return {
    labels: data.map(d => d.month),
    values: data.map(d => d.count),
  };
}

export async function fetchUsersByRole() {
  const { data } = await api.get('/users/stats/by-role');
  // data example: [{ role: 'admin', count: 5 }, …]
  return {
    labels: data.map(d => d.role),
    values: data.map(d => d.count),
  };
}
```

Part IV - Render Charts with Chart.js

- Install Dependencies
  npm install chart.js react-chartjs-2
- Import and Register Chart.js Components - *In src/charts/setup.js* (according to the official library documentation)
- In your entry point (main.jsx), import this setup once: *import './charts/setup';*
- Create a Reusable Chart Component - *In src/components/BarChart.jsx* (according to the official library documentation)
- Use Charts in Dashboard (in Dashboard.jsx)

- Chart.js React wrapper: https://react-chartjs-2.js.org/
- Basic example: https://www.chartjs.org/docs/latest/getting-started/

Deliverable: Fully populated charts on /dashboard

Part V - Implement Data Export

- Choose one export format:
  - **CSV**: Use PapaParse or plain JS to stringify your data arrays, then trigger a downloa
  - **PDF**: Use jsPDF to render text/tables and trigger download.

Wire the export function to your “Export” button so that clicking it downloads the current dataset.

Deliverable: Working “Download CSV” or “Download PDF” button (screenshot + code snippet).

Part VI - Cleanup & Documentation

Ensure your code is modular (chart logic in its own component/hook).
• Write a brief section in your README or a 1-page PDF covering:

- Which metrics you chose and why
- Chart library used and configuration highlights
- Export approach (CSV vs PDF) and any limitations

Deliverable: Updated README or PDF report.

Grading criteria

- Dashboard Route & Layout - */dashboard* route added and wired into navigation; layout matches spec (titles, containers, export button); responsive spacing and styling via Tailwind (or CSS).
- Data Fetching & State Management - Data service module created; fetch/axios calls return correct { labels, values }; loading and error states handled gracefully in UI.
- Chart Rendering - Two charts rendered with Chart.js (or equivalent), using reusable components; data correctly bound; charts display titles, axis labels, and respond to container size.
- Export Functionality - “Download” button exports current dataset in chosen format (CSV or PDF); file contents match on-screen data; download works across browsers.