EuroSkills Test Project

Web Development 17

CONTENTS

This Test Project consists of the following documentation/files:

1. This document: ES2025_TP_Web_Development_17.pdf
2. Marking scheme: ES2025_Marking_Scheme_Web_Development_17.xlsx

INTRODUCTION

The proposed test project aims to provide a comprehensive challenge in web development by covering six distinct modules. Each module focuses on different aspects of web design and development, ranging from static website design to advanced techniques and collaboration. This project aims to challenge competitors with a well-rounded skill set that encompasses both front-end and back-end development, as well as integration with external services through APIs.

DESCRIPTION OF PROJECT AND TASKS

The Test Project will consist of 6 modules with the following topics:

- Module **A**: Static Website Design (HTML/CSS)
- Module **B**: Dynamic website with server-side rendering
- Module **C**: REST API
- Module **D**: Interactive Frontend using an API
- Module **E**: Advanced Web Development
- Module **F**: Collaborative Challenge

These modules will be further described in this document.

INSTRUCTIONS TO THE COMPETITOR

Module A – Static Website Design

In this module, you must develop a static website for a client using only HTML and CSS.

No server-side or client-side framework is allowed for module A.

CSS preprocessors may be used.

All HTML and CSS code, generated or hand-written, must pass the W3C HTML (https://validator.w3.org/) and CSS (https://jigsaw.w3.org/css-validator/) validators.

Competitor Information

Module A will be assessed using Google Chrome and Firefox.

The axe browser extension installed in Google Chrome is going to be used to assess that the website is "accessibility supported" according to WCAG.

Website Requirements

The client wants to build a new offshore wind farm.

They have now assigned you the task of implementing the promotion website to promote their new project and find possible investors.

They have provided you with content for the website in the media files directory.

Not all the assets have to be used. You can also create additional assets and text as you see fit.

It's also possible to add links to pages that do not yet exist (for example, a login page).

The website must be responsive and support at least the following viewports:

- Mobile portrait: 320×480 px
- Tablet portrait: 768×1024 px
- Desktop: 1280×800 px

As the website will be publicly available, it is essential to the client that the website conforms to accessibility guidelines (WCAG, at least level AA) and implements SEO best practices.

The following pages must be implemented.

Home Page

Introduce the project on the home page with the following essentials:

- Embedded visualization video of the offshore wind farm (provided)
    - Autoplay when page is opened without user interaction
    - Video is looped
    - Controls are hidden
    - Takes the full browser width while always maintaining the original aspect ratio
- Brief project overview
- "Key Facts" highlight strip
- Map (provided) showing the proposed offshore wind farm location
- Partner / sponsor logos

The home page must be the index page (index.html).

Investors Page

Both companies and private individuals should be able to become investors in the project.

It is possible to invest with multiple options:

- Fund one or more wind turbines
    - This allows the investor to provide a short text of at most 25 characters or a company logo that will be engraved on the turbine
- Become a presenting sponsor
    - This allows the investor to provide a logo that will be present on the website and all official communication
- Support the project with a variable amount of money

Those different options have to be displayed on the page.

To be able to show interest in becoming an investor, the page should contain a form to gather contact data (name, email, address, phone number) and it should allow to specify which investment option they choose.

Based on the chosen investment options, the specific fields of that option are presented.

Suitable form field types and validation must be chosen.

Handling of the form submission does not have to be implemented at the moment.

Visitor Tours Page

Once the wind farm has been built, visitor tours will be provided to explain the technology and to see it in close-up action.

The first few planned tours are already listed on this page with their dates and times, and users should be able to book them.

To book a tour, it's enough to display a "Book now" button next to a tour date. Further functionality does not have to be implemented at the moment.

This page should also contain a map showing the offshore wind farm's planned location and the visitor tour's meeting point.

Global Elements

The following elements must be visible on all pages:

- **Navigation**
    - must contain links to all three pages
    - design must clearly highlight the current page
    - links must have adequate hover effects
- **Footer**
    - includes at least a copyright notice

All elements can be enriched with more information where it makes sense.

Module B – Dynamic website with server-side rendering

Goal

Build a **server-side rendered (SSR)** web app.

You can optionally use JavaScript to enhance the user experience, but the initial content of each must be rendered on the server. Test to check: Fetch the page in a CLI tool (like `curl` or `wget`) and verify that the HTML contains the content.

Tech Rules

- Must work in Google Chrome.
- Use any CSS framework or your own styles, but keep branding consistent.

Website & Branding Integration

- Optional merge with Module A: You may embed your static pages from Module A into this app, or keep them separate.
- Consistent look: Colours, fonts, and logo placements must match across all pages (static & dynamic).
- Easy navigation: Put clear links in the main nav bar (or landing page buttons) so experts can reach Investors, Tours, Admin pages in ≤2 clicks.
- Media: Re-use images/media from Module A or create your own files.
- Admin section: Has a clear and polished design, it is very clear to the user how to approve / reject requests.

Global Requirements

- Authentication:
    - To log in, users enter their email. The system sends a login link to their email. Clicking the link logs them in. Users are identified by their email. There are no passwords. The same is true for the admin user.
- Security:
    - Implement a CSRF token on every form or action (POST/PUT/DELETE), usually with a hidden input field.
    - Escape/encode user input in views to avoid cross-site scripting (XSS) attacks.
    - Use parameterised queries or rely on an ORM to prevent SQL injection.
- Organization:
    - Your code must be split into clear modules and ensure consistent separation of concerns (database, render / model, logic).
    - Forms with missing or invalid data must show an error message. The error message must be clear, understandable and user-friendly.

Email

At various points, the system sends emails to users. You must not actually send emails, but mock their sending.

You can define the email body in plain text or HTML format.

Store the sent emails in a database table and display them on a dedicated page `/mock-emails`. The table must include the recipient, subject, body, and sent timestamp. Sent emails are sorted by their timestamp.

The body of the email must be rendered with new lines preserved and links clickable. If you decided to use plain text emails, you must convert them to HTML for display on the `/mock-emails` page.

Additional requirements:

- The page must be accessible without authentication.
- Styling is not required, but the table must be readable.
- The page does not have to be linked anywhere. When in doubt you can specify the link in the README.md file of your Git repository.

Pages and Features

Each page must be accessible via a unique URL. The URLs must be descriptive and follow a consistent pattern.

Presenting Sponsor Page

The presenting sponsor page displays the logo of the approved presenting sponsors.

The page shows at least 3 sponsors. However, if there are fewer than 3 approved sponsors, the page replaces those with a placeholder image. The page is not limited to 3 sponsors, but must show at least 3 (real or placeholder).

Investors Page

The investors page allows website visitors to invest and show existing investments.

Total support amount: Displays the total amount of support in DKK by all investors across all investment types.

Turbine spots visual: Shows available, pending, and approved turbine spots. A turbine has **10 spots** and they are numbered from 1 to 10. Each spot can be funded by a single investor. Any website visitors can see the spots and their availability.

Investing: A link or button opens a form to submit an investment request:

- Requires name, email, address, phone number
- Choose type of investment which shows different fields:
  - Fund turbine: Requires a turbine spot selection, a logo upload (PNG, max 1 MB) or a ≤25 char text.
  - Presenting sponsor: Requires a logo upload (PNG, max 1 MB).
  - Support amount: Requires a money field for the amount. (Currency is DKK)

After submitting a valid form, the user receives an email confirming their requests and that it is now pending.

- Subject: "Investment Request Confirmation #{investmentReference}"
  - `{investmentReference}` is a unique reference generated by the system. The format is `INV-XXXXXXXX`, where `XXXXXXXX` is composed of 8 digits, incrementing by one for each investment request.
- Body:

```
Hi {name}!

Thank you for your investment request.
We have received your request and it is now pending approval.

Investemnt reference: {investmentReference}
Investment Type: {investmentType}
Turbine Spot: {turbineSpot} (if applicable)
Logo/Text: {logoOrText} (if applicable)
Support Amount: {supportAmount} (if applicable)

We will shortly review your request and provide you with details on the
next steps (such as payment details).

Stay breezy and sustainable!
```

Additionally, if they chose to fund a turbine, the turbine spot is blocked for other users until the admin approves or rejects the request. The turbine spot shows as "pending".

Once an investment is approved by an admin:

- Fund turbines: The turbine spot shows the uploaded logo or text (and no longer shows "pending").
- Presenting sponsor: The logo shows on the presenting sponsor page.
- Support amount: The amount is added to the total support amount.

... and the user receives an email confirming the approval:

- Subject: "Investment Request Approved #{investmentReference}"
- Body:

```
Hi {name}!

Your investment request has been approved!

Investment reference: {investmentReference}

Stay breezy and sustainable!
```

If the admin rejects the investment request, the user receives an email with the rejection.

You must use the same subject and body as for the approval, but replace the word "Approved" with "Rejected".

Visitor Tours Page

The tours page allows users to book seats on tours.

- Shows all tours with date, time, and available seats left.
- Authenticated users can book seats on a tour. If the user books more seats than available, the system prevents over-booking and shows an error message, and allows the user to select fewer seats.

The booking form requires tour selection, number of seats, and personal details: name, address, and phone number.

The booking is linked to the authenticated user. A single user can book multiple tours, including multiple times on the same tour.

Once the booking form is submitted and successfully validated, the user is redirected to the Tours page, where they can see their bookings at the top of the page. The booking status is "confirmed".

Also after a successful booking, the system sends an email to the user with the booking details:

- Subject: "Wind Farm Tour Booking Confirmation"
- Body:

```
Hi {name}!

You have successfully booked {numberOfSeats} seats on our Wind Farm tour.

Date: {tourDate} {tourTime}
Booking Details: {linkToToursPage}

We look forward to seeing you.

Stay breezy and sustainable!
```

Once the booking is confirmed, the user can cancel it. They can do this by clicking a "Cancel" button next to the booking on the Tours page. The booking status changes to "cancelled". The user can cancel a booking at any time.

The seats booked by the user are released back to the available seats for that tour.

Admin `/admin/*` Pages

All admin pages are protected by authentication. Only users with the **admin** role can access them.

All admin pages should be accessible via the `/admin/*` URL pattern.

The route `/admin` shows the admin dashboard which contains links to the following pages:

- Investment Admin Page
- Tours Admin Page

Investment Admin Page

The investment admin page allows the admin to manage all investments made by users.

It shows all investments made by users, including the type, status, and personal details (name, email, address, phone number).

The admin can approve or reject the pending investment requests with a button.

Tours Admin Page

The tours admin page allows the admin to manage tours.

- Shows all tours with date, time, capacity and available seats.
- Allows the admin to create new tours.
- Tours without any bookings can be deleted.
- Each tour has a link to view bookings for that tour. Each booking shows the personal details and the number of seats booked.
- The admin can cancel individual bookings:
  - releases the booked seats back to the available seats.
  - if all seats are cancelled, the tour can be deleted.
  - sends an email to the user with the cancellation details:
    - Subject: "Wind Farm Tour Booking Cancellation"
    - Body:

```
Hi {name}!


Your booking for the Wind Farm tour on {tourDate} at {tourTime} has
been cancelled.

We apologize for any inconvenience this may cause.

If you have any questions, please contact us.


Stay breezy and sustainable!
```

Database

You are free to design the database schema as you see fit.

You must provide a SQL dump file with both the **structure** and **initial data** to seed the database.

It must be committed to the Git repository in the root directory as `seed.sql`.

The SQL dump must include the following data:

- 2 users:
  - Email: `admin@localhost` / Role: `admin`
  - Email: `user@localhost` / Role: `user`
- 10 turbines (unique names, e.g. `Turbine 1`, `Turbine 2`, etc.)
- 2 investments: Fund turbine: Turbine, text
- 2 investments: Fund turbine: Turbine, logo

- 2 investments: Presenting sponsor: Logo
- 5 tours: unique date + time, capacity 10
- 2 bookings: on same tour, 2 seats each

The SQL dump will be assessed for correctness and completeness.

You may include additional seed data as long as the required rows above remain intact.

Example user flow

These user flows demonstrate the expected functionality of the application. You can use it to test your implementation.

It does not cover all edge cases, but it should give you a good idea of how the application should work.

Tour booking flow

1. Alex opens the website, goes to the tours page, and sees all tours.
2. Alex signs up with their email address. Receives an email (found in `/mock-emails`), clicks the link to log in.
3. Alex selects a tour with available seats and books **2 seats**.
4. Alex receives a confirmation email with booking details.
5. Alex goes to the tours page and sees their booking at the top with status "confirmed".
6. Alex decides to cancel the booking.
7. Alex clicks the "Cancel" button next to the booking.
8. Alex receives an email confirming the cancellation.
9. Alex goes to the tours page and sees their booking status changed to "cancelled".
10. Alex can book the same tour again or a different one. The seats are available again.

Investment flow

1. Alex opens the website, goes to the investors page, and sees available turbine spots.
2. Alex signs up with their email address. Receives an email (found in `/mock-emails`), clicks the link to log in.
3. Opens form to invest, select "Fund turbine", enters personal details and adds text or uploads logo.
4. An admin approves the investment request.
5. Alex sees the turbine spot with their logo or text on the investors page.

More flows exist, but this should give you a good idea of the expected functionality.

Module C – REST API

In this module, you must develop a backend REST API that provides live monitoring and control capabilities for an offshore wind farm. You are tasked with building an API for use by a frontend application (Module D).

The backend must consume turbine data from an external API, process it, cache it, and provide a clean and frontend-ready version. Additionally, it must support alerts, turbine control, and assigning roles.

Competitor Information

- The backend will be tested using HTTP clients and a test suite that will interact with your API.
    - Note: The test suite will be provided in the competitor handout.
- A mock server that simulates the external API is provided.
    - Note: The competitor handout will include the mock server URL and details.
- The API must be secure and reject unauthorized requests.
- The backend must provide the API specified in the provided OpenAPI specification.
- The use of a relational database is required; an in-memory or file-based store is not acceptable.
- The backend must implement all domain logic as described.

Scenario

You are building the backend for a wind farm off the coast of Denmark. A third-party system provides raw turbine data via an external API. Your backend acts as a middle layer: it fetches, validates, caches, interprets, and serves this data to a frontend. You also expose control, alert, and role-assigning functionality through your own API.

Requirements

1. Authentication and Access Control

Roles:

The API must support three roles with different access levels:

- anonymous: no access to protected endpoints. No authentication required.
- operator: read + control + acknowledge alerts. Requires authentication.
- admin: full access, including assigning roles. Requires authentication.

Login: `POST /auth/login`

- Input: `{ "username": string, "password": string }`
- Output: `{ "token": string, "role": string }`
- Validate token for subsequent requests
    - Validate token on each request to be in the `Authorization` header as `Bearer <token>`
    - The token string must sufficiently sophisticate to prevent brute-force attacks (i.e. don't just use the username as token)

Test Users:

You must provide test users for authentication.

See the **Database** section below for details.

Role Access:

- Public (no auth required) endpoints:
  - `POST /auth/login` — authenticate user and return token
  - `GET /turbines` — read all turbines (id, name, location, status)
  - `GET /turbines/:id/status` — read turbine status
  - `GET /turbines/:id/actions` — read triggered actions for a turbine
- Protected endpoints (require auth & role):
  - `GET /alerts` — list active alerts
  - `POST /alerts/:id/ack` — acknowledge an alert
  - `POST /turbines/:id/control` — control turbine pitch and yaw
  - `POST /turbines/:id/start` — start turbine
  - `POST /turbines/:id/shutdown` — shutdown turbine
  - `POST /turbines/:id/maintenance` — enter maintenance mode
  - `GET /turbines/:id/logs` — get turbine logs
  - `POST /auth/assign-role` (admin only)

2. External API Integration

Turbine status is not stored locally. You must fetch live data from an external API when the frontend requests it.

You do not need to implement a polling mechanism; the frontend will request data as needed.

External API Info:

Two URLs are provided during the competition:

- Mock API base URL (e.g. http://mock-api:8000) – use this in code for all programmatic calls: `GET /turbines`, `GET /turbines/:id/logs`, `POST /turbines/:id/control`, etc.
- Control-panel URL (e.g. http://mock-control:8080) - a browser-only UI for manually simulating scenarios (empty data, partial data, errors). Do not call it from your backend.

Mock API Details:

- Authentication: Send `Authorization: Bearer <token>` in headers (token will be provided during the competition).
- Endpoint: `GET /turbines`
- Returns:

```
{
    "timestamp": "2025-06-21T10:12:00Z",
    "data": [
        {
            "id": 1,
            "name": "Turbine A1",
            "location": {
                "lat": 56.4501,
                "lng": 8.3465
            },
            "rpm": 47,
            "powerMw": 1.9,
            "yaw": 270,
            "pitch": 22,
            "temperature": 35.2,
            "status": "started"
        },
        ...
    ]
}
```

An OpenAPI specification will be provided during the competition.

It would not make sense to work with the real external API during the competition, you will be provided with a mock server that simulates the external API. This mock server also includes a control panel where you can simulate scenarios, such as empty data, partial data, missing properties or errors.

Each turbine has the following properties.

Property	Type	Description
id	Integer	Unique turbine identifier
name	string	Turbine name
location	object	`{ lat: float, lng: float }`
rpm	integer	Rotations per minute (dynamic)
powerMw	float	Power output in megawatts (dynamic)
yaw	integer	Yaw angle in degrees (0–360)
pitch	integer	Blade pitch angle in degrees (-90–90)
temperature	float	Temperature in °C (dynamic)
status	string	One of: `"started"`, `"maintenance"`, `"shutdown"`

Special Cases:

The external API may return:

- A timestamp indicating when the data was last updated.
- Empty data: `"data": []`
- Partial turbines: `"data": [{...}, {...}]` (some turbines may be missing)
- Partial properties: Some turbines may not have all properties filled in (e.g., `temperature` may be `null`).
- Errors (e.g. network timeouts or 500s)

You must handle these cases gracefully:

- The response you serve to the frontend must always include a `freshness` field for each turbine and also for each turbine property, indicating whether the data is `live`, `cached`, or `missing`.
- The response must also include a `lastUpdated` timestamp for each turbine and each property, indicating when it was last updated.

3. Alerts

You must implement the specified alert rules. The rules are evaluated on fetch of the turbine data from the external API (no polling required).

Alerts must be stored in a database and can be acknowledged by the operator. Your logic must deduplicate alerts and ensure that each alert is only triggered once per turbine. A new alert can only be triggered if the previous alert is resolved (the rule was evaluated to `false`).

For now, only one alert needs to be implemented:

- High RPM Alert: If `rpm` exceeds 60, trigger an alert.

Alert status:

- Firing state: `firing` (active alert) or `resolved` (alert rule is no longer true)
- Acknowledged state: `acknowledged` (alert has been acknowledged by the operator) or `unacknowledged` (alert has not been acknowledged)

The alerts can be retrieved via endpoint `GET /alerts` and acknowledged via `POST /alerts/:id/ack`.

4. Turbine Control

Control actions:

- Set pitch and yaw angles via `POST /turbines/:id/control`
    - Input: `{ "pitch": integer, "yaw": integer }`
    - Valid ranges: `pitch: -90 to 90`, `yaw: 0 to 360`
    - Response: `{ "status": "success" }` or error message
- State transitions with the turbine: `POST /turbines/:id/:action` (`action` can be `start`, `shutdown`, or `maintenance`)
    - Response: `{ "status": "success" }` or error message

- The transition must be valid based on the current turbine state (see state transitions below).

Allowed Transitions:

1. Started → Shutdown: Shutdown, e.g., for low wind or grid issues
2. Shutdown → Started: Restart after shutdown
3. Shutdown → Maintenance: Maintenance can only be performed during downtime
4. Maintenance → Shutdown: Maintenance completed, turbine shuts down, making it ready for restart

The user actions must be saved in the database and can be retrieved via the unauthenticated endpoint `GET /turbines/:id/actions`.

- Each action includes a timestamp and the user who performed the action.
- The action type is one of `control`, `start`, `shutdown`, `maintenance`.
- A `control` action also includes the pitch and yaw values.

5. Turbine Logs

You must implement a logging system for each turbine. The logs can be retrieved from the external API via the endpoint `GET /turbines/:id/logs` which returns a `plaintext` response with the following format:

```
2025-06-21T10:12:00Z [Info] Turbine started using config /etc/turbine-a1.json
2025-06-21T10:15:00Z [Warning] Turbine start delayed due to low wind conditions
2025-06-21T10:20:00Z [Error] Turbine lost satellite fallback connection. Details:
multi line error message belonging to the sensor failure
second line of the error message still belongs to the sensor failure
2025-06-21T10:25:00Z [Info] Maintenance mode activated
```

The logs must be parsed to include:

- Turbine ID: not embedded in each log line; use the `:id` path parameter from the request
- Timestamp: parsed from the log line
- Log level: extracted from the log line and normalized to lowercase (`info`, `warning`, `error`)
- Message with new lines preserved: the rest of the log line after the timestamp and level until a new line and another log line with timestamp, level, and message starts

You will have to develop a parser that converts the plaintext logs and stores them in a structured format in the database.

The external API will return the last 1000 log entries for a turbine, and you must ensure you do not store duplicate logs based on the timestamp and message. Logs are only fetched when the frontend requests them, so you do not need to implement a polling mechanism. The external log endpoint can

be unavailable or return an error, in which case you must handle this gracefully and return the cached logs if available.

The endpoint to retrieve the logs is `GET /turbines/:id/logs`. The frontend will use this endpoint to display the logs in a user-friendly format. The sort order of the logs must be from oldest to newest, and only the newest 1000 logs must be returned at a time. No pagination is required.

The frontend also requires a search functionality to filter logs by level and message substring.

These shall be implemented in the backend as query parameters:

- `levels`: Filter logs by log level (e.g., `info`, `warning`, `error`). A comma separated list of levels can be provided.
- `message`: Filter logs by a substring in the message

6. Role Management

You must implement a role management system that allows the admin to assign roles to users. The admin can assign roles to users via the endpoint `POST /auth/assign-role`. The request must include the username and the role to assign.

A user with admin role cannot remove their own admin role.

7. Database

You are free to design the database schema as you see fit.

You must provide a SQL dump file with both the **structure** and **initial data** to seed the database.

It must be committed to the Git repository in the root directory as `seed.sql`.

The SQL dump must include the following data:

- user without role: `username: user`, `password: user12345`
- operator: `username: bob`, `password: bob12345`
- admin: `username: alice`, `password: alice12345`

The `seed.sql` file must not contain the plaintext passwords, but rather the hashed passwords.

You may include additional seed data as long as the required rows above remain intact and tests still pass.

8. API Specification for Frontend

An OpenAPI specification will be provided during the competition. It will include all endpoints, request and response schemas, examples, and authentication details. You must ensure your API adheres to this specification.

Error responses, such as authentication errors or validation errors, will be detailed in the OpenAPI specification.

Module D – Interactive Frontend using an API

You are asked to create a frontend for the REST API from module C.

Since the functionality you will create in this module builds on top of the functionality of module C, you will be given a working solution of module C.

You must build on the provided solution of module C instead of your own to ensure full feature parity.

Competitor Information

- The frontend can be implemented using a framework and other available libraries.
- The application must be a Single Page Application (SPA).
- Page reloads must present the same content to the user as previously visible, except unsaved user-driven inputs.

Assessment

Module D will be assessed using the provided version of Google Chrome. The assessment will include functional tests as well as user experience.

Any modifications in the provided backend of the previous module, including any changes to the database, will not be taken into account.

Requirements

1. Authentication and Access Control

The website is publicly accessible for read-only usage, so no login is needed to access the website in general.

However, some actions and pages require an authenticated user.

Performing those actions must not be possible and links to those pages must not be visible for unauthenticated users.

Accessing a page that requires an authenticated user (such as the turbine logs page) must redirect unauthenticated users to the login page and, after a successful login, return them to the initially requested page.

Accessing a page that requires a specific role and the logged-in user does not have the necessary role must show an error page.

To log in, a button must be present on all pages to open a login form.

The login form authenticates the user with the provided credentials against the API and stores the returned token.

Refreshing the page must keep the login state.

If a user is logged in, they have a logout button on every page, which will clear the stored token and return the user to the read-only version.

The following users are provided to test authentication:

- user without role: `username: user`, `password: user12345`

- operator: `username: bob`, `password: bob12345`

- admin: `username: alice`, `password: alice12345`

2. Turbine Map Page

The turbine map page is the index page (default page when opening the web app) and is publicly accessible.

The central part of this page is a map showing the location of all turbines.

They are rendered in a rectangle where the turbine with the lowest latitude and lowest longitude is located in the bottom left corner, and the turbine with the highest latitude and highest longitude is located in the top right corner.

All turbines are fully visible and rendered in the correct relative location.

The map does not have to be an actual map and can, for example, just be a blue rectangle (indicating the sea) or a grid.

Each turbine has to be represented with an icon.

The icon color represents the status of the turbine, where `started` turbines are green, `maintenance` are orange, and `shutdown` are gray.

3. Turbine Detail Pane

Clicking on a turbine icon on the map shows the turbine detail pane, which is opened to the right of the page while still showing the turbine map in the left part of the page.

The turbine detail features the following information, if available:

- Name of the turbine
- Status (`started`, `maintenance`, `shutdown`)
- Latitude & longitude
- Yaw & Pitch
- Stats (Rotations per minute, power output in megawatts, temperature in °C)
- Freshness status of the turbine (`live`, `cached`, `missing`) and last update of the turbine

For status, yaw, pitch, RPM, power output, and temperature, the freshness status and last update of that field must be shown next to it.

A warning icon must be visible if a field has a missing status.

While having the detail pane open, the turbine data is refreshed in a 5-second interval, and the view is automatically updated.

For consistent display of information, the icon color on the map has to be updated as well.

The detail pane also features a simple power output graph, where the x-axis is the time and the y-axis is the power output (no labels required).

With each data refresh (every 5 seconds), a new data point gets added to the graph.

The graph data does not have to be persisted, meaning closing the detail pane and opening it again will start with an empty graph again.

Reloading the page while having the detail pane of a turbine open must also show the detail pane again.

4. 3D Turbine Model

The client found a script on the internet that renders a 3D turbine with a given yaw, pitch, and RPM.

This script must be integrated into the turbine detail pane to visualize the turbine stats.

The actual yaw, pitch, and RPM values of the turbine have to be passed to the script after the initial load of turbine data and with each subsequent refresh of data.

The README.md provided in the media files contains instructions on how to use the script.

The script must be loaded using best practices of the used framework (e.g. only loaded when used, not loaded multiple times).

5. Turbine Control

If the user is authenticated and has the role `operator` or `admin`, they can control the turbine within the turbine detail pane.

This includes:

- Modifying yaw (0 – 360) & pitch (-90 – 90) according to the allowed values of the API
- Transition the turbine to a different state according to the allowed transitions of the API

The frontend must already validate those actions (same requirements as for the API) and not send a request to the API in case of invalid values.

6. Alerts Page

The alerts page is only accessible for authenticated users with the role `operator` or `admin`.

List all active alerts with the following information:

- Timestamp in the format `dd.MM.yyyy HH:mm:ss`
- Alert type
- Turbine name
- Firing state
- Acknowledged state

There must also be a button for each alert to mark it as acknowledged, if it has not been acknowledged yet.

After acknowledging an alert, the status of that alert has to be updated in the list.

7. Actions Page

The actions page of a turbine is publicly accessible and can be opened from within the turbine detail pane.

It shows a list of all performed actions with the following information:

- Action type

- Pitch & yaw values in case the action type is `control`
- Timestamp in the format `dd.MM.yyyy HH:mm:ss`
- User who performed the action

8. Logs Page

The logs page is only accessible for authenticated users with the role `operator` or `admin` and can be opened from within the turbine detail pane.

It shows a list of all returned turbine logs with the following information:

- Timestamp in the format `dd.MM.yyyy HH:mm:ss`

- Log level

- Message with new lines preserved

Users must also be able to search (within the message) and filter by one or more log levels.

The search and filter functionality from the backend must be used.

9. Design

The website must implement a simple but clean design.

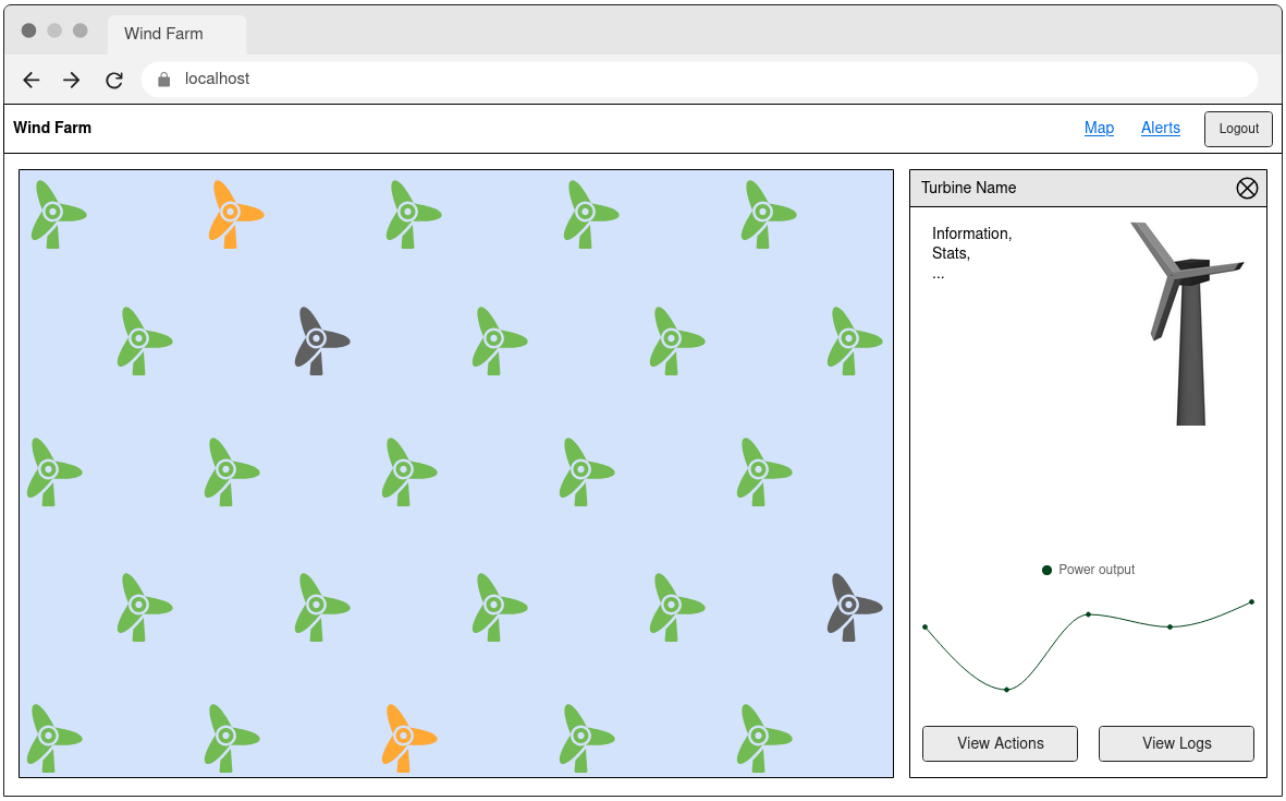
The focus is on managing turbines; therefore, key information must be presented clearly without overwhelming the user with too much text and numbers.

The website must work for a desktop size of 1280×800 px.

No text or icons are cut off, and that viewport has no horizontal scrolling.

The client has provided a mockup of the main page to show their vision.

It only serves as an example, and the layout can be changed in any way as long as it still fulfills the requirements.



Module E – Advanced Web Development

In this module, you are expected to solve three tasks.

Within the media files, you will find three starter kits for each task. You are expected to use these starter kits as a base for your solution. You are not allowed to use any frameworks or libraries for this module except a testing framework for task 1.

Task 1: Writing automated tests

You are given a JavaScript project that has no automated tests. You must write automated unit tests for the project.

A complete test set is expected which covers 100% of the provided code lines and conditionals. A JavaScript testing framework must be used.

These are the assessment criteria for this task:

- The tests are grouped into files logically.
- The tests are written in a way that they are easy to understand.
- The tests pass when running against the original code.
- The tests cover 100% of the provided code lines and conditionals.
- The tests do not pass any logically mutated version of that code (Mutation Testing).
- Each test file is submitted in a separate Pull Request with a clear title and description.
- Commit messages are clear and descriptive.

Task 2: OWASP Top 10 Security Vulnerabilities

You are given a JavaScript project that has code security vulnerabilities. You must find and document these vulnerabilities and suggest improvements. You will document them in an issue tracker. The security vulnerabilities must be documented in a way that they can be understood by a developer with basic knowledge of security issues. You must also provide a summary of the security vulnerabilities you found in a central issue on the issue tracker that links to the individual issues.

These are the assessment criteria for this task:

- The vulnerabilities are documented in a way that they can be understood by a developer with basic knowledge of security vulnerabilities.
- The issues include:
    - a description of the issue,
    - the place(s) in the code where the issue is located,
    - why it is a security issue,
    - to which OWASP Top 10 category it belongs,
    - and concretely how to fix it minimally invasive.
- The issues are documented in an issue tracker.
- The issues are grouped logically.
- As many real vulnerabilities as possible are documented.

The OWASP Top 10 documentation is going to be provided in the media files.

Details on the issue tracker will be provided in the competitor handout.

Task 3: CSS Dark Mode Toggle

Your client calls you about a new feature request for a simple static website they already have.

They would like you to implement a dark mode to the existing website with the following requirements:

- The preferred color scheme from the operating system should be applied without JavaScript being needed
- If JavaScript is enabled, a new toggle must be added to the header which switches between light and dark theme, allowing the user to overwrite the system default theme
- If the theme is changed with the manual toggle in the header, the chosen theme must be preserved over page reloads, when the tab is closed and the page opened again, and across browser restarts
- The background color of the page in dark mode should be `#121212`
- All text on the page should be well readable in the dark mode with a color contrast ratio of at least 7:1 to meet WCAG Level AAA
- If an image has a dark theme variant (provided, next to the light theme variant), it should be used when the dark theme is active, even without JavaScript enabled
- As the colors might have to be tuned later, all colors should be referenced as CSS variables, which are defined in a central place, even colors that are set in the already provided styles
- No CSS preprocessors or frameworks, nor JavaScript frameworks can be used

To track this feature request and to be able to reference it in the future, you must record it in an issue tracker.

Create a new issue summarizing the client's request and list all requirements.

Keep the status of the issue up-to-date as you progress with the implementation, and create a link between the issue and the pull request(s) containing the feature in both directions.

Ultimately, all pull requests must be merged, and the ticket must be closed.

Module F – Collaborative Challenge

"What if wind…" powers a game? paints art? solves a social problem? teaches kids? personalises music?

Goal: Create a small web experience that celebrates wind power in Denmark.

You are then asked to present your group's result to the other competitors and the experts.

Your presentation must be divided into three parts:

- A *Hook* to start the presentation and catch the audience's attention.
- A *Middle Part* where you explain your work and might also demonstrate something.
- A *Close* where you round out the presentation.

The middle part must contain a demo. Additionally, you could include any of these elements or others:

- Ambition: Details of the goal you set yourself.
- Approach: Illustrate how you worked to achieve your goal.
- Impact: What impact would your idea have on the world?
- Journey: Refer to your journey during this module. Show how you worked, what you learned on the way, where you succeeded or failed, or what discoveries you made.

Assessment

You will be assessed by your presence and participation in the module, that there is a worked out idea, and that the presentation is done in a professional manner. Attendance is mandatory for all competitors to all presentations.

EQUIPMENT, MACHINERY, INSTALLATIONS AND MATERIALS REQUIRED

It is expected that all Test Projects can be done by competitors based on the equipment and materials specified in the Infrastructure Lists*.

URL to Infrastructure List: https://il.worldskills.org/#/events/612/lists/1525/public

MATERIALS, EQUIPMENT AND TOOLS SUPPLIED BY COMPETITORS IN THEIR TOOLBOX

Competitors may bring the following items:

- Mouse with mousepad
- A maximum of one USB keyboard in the Competitors desired language.
  Note: If the keyboard brought by the Competitor does not work then a standard keyboard will be provided by the Competition Organizer
- Language file for the operating system to make the keyboard work correctly
- Headset and extension cable

Any device brought in by the Competitor may not have any internal memory storage. Assigned Experts and Workshop Manager have the right to disallow certain equipment brought by Competitors. Backup equipment is allowed in case of failure but should always be kept inside the Competitors locker.

MATERIALS & EQUIPMENT AND TOOLS PROHIBITED IN THE SKILL AREA

The Skill area is the area outside the experts' room within the Workshop Area.

- Extra software
- Mobile phones
- Tablet devices
- Smart watches
- Photography/Video devices
- USB Drives
- Any device brought into the workshop may not have any internal memory storage devices

The Chief Expert, Deputy Chief Expert and Workshop Manager have the right to disallow equipment brought by Competitors.

MARKING SCHEME

ID	Description	Day 1	Day 2	Day 3
A	Static Website Design	16.50		
A1	Common Elements	1.75		
A2	Home Page	3.25		
A3	Investors Page	3.75		
A4	Visitors tours page	1.75		
A5	Standards	4.00		
A6	Organization	2.00		
B	Dynamic website with server-side rendering	23.00		
B1	Organization, Design, Email Templates	5.25		
B2	Security & Authentication	5.00		
B3	Architecture & Infrastructure	4.50		
B4	Functionality — Public Site	5.75		
B5	Functionality — Admin Interface	2.50		
C	REST API		16.75	
C1	Organization		1.75	
C2	Authentication & Access Control		3.25	
C3	Turbine Data & Freshness		4.00	
C4	Control & Action Log		3.00	
C5	Alerts Lifecycle		2.75	
C6	Log Retrieval & Search		2.00	
D	REST API		22.00	
D1	Authentication		3.25	
D2	Turbine Map Page		2.75	
D3	Turbine Detail Pane		4.35	
D4	3D Turbine Model		1.50	
D5	Turbine Control		2.50	
D6	Alerts Page		1.75	
D7	Actions Page		1.00	
D8	Logs Page		2.00	
D9	General Requirements		2.90	
E	Advanced Web Development			16.75
E1	Task 1 - Writing automated tests			5.00
E2	Task 2 - OWASP Top 10 Security Vulnerabilities			5.50
E3	Task 3 - CSS Dark Mode Toggle			6.25
F	Fun: Collaborative Challenge			5.00
F1	Productive teamwork			1.00
F2	Presentation			2.00
F3	Implementation			2.00