

EuroSkills Test Project

Web Development 17

INTRODUCTION TO TEST PROJECT DOCUMENTATION

INTRODUCTION TO TEST PROJECT DOCUMENTATION	2
CONTENTS	2
INTRODUCTION	2
DESCRIPTION OF PROJECT AND TASKS	3
INSTRUCTIONS TO THE COMPETITOR.....	3
Module A – Static Website Design	3
Module B – Dynamic website with server-side rendering.....	5
Module C – Commercial Open API	8
Module D – Interactive Frontend using an API.....	12
Module E – Advanced Web Development.....	15
Module F – Collaborative Challenge.....	19
EQUIPMENT, MACHINERY, INSTALLATIONS AND MATERIALS REQUIRED.....	20
MATERIALS, EQUIPMENT AND TOOLS SUPPLIED BY COMPETITORS IN THEIR TOOLBOX	20
MATERIALS & EQUIPMENT AND TOOLS PROHIBITED IN THE SKILL AREA	20
MARKING SCHEME	20

CONTENTS

This Test Project (proposal) consists of the following documentation/files:

1. This document: ES2023_TP_Web_Development_17.pdf
2. Marking scheme: ES2023_marking_scheme_Web_Development_17.xlsx
3. API Specifications: ES2023_api_specs_Web_Development_17.zip

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**: Commercial Open 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

This would be the instructions that would be given to the competitor if doing this project. The instructions can be increased and improved at the Competition during the preparation time.

Module A – Static Website Design

In this module, you are expected to develop a static website for a client using HTML and CSS only. The website must consist of four different pages which will be explained in detail in the following sections.

Within the media files, the client provided some media, icons, and text. You are free to use them or create custom assets on your own.

As the website will be publicly available, it is important to the client that the website conforms to accessibility standards (WCAG) and implements SEO best practices.

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

- Mobile: 360x640
- Tablet: 768x1024
- Desktop: 1920x1080

No server-side or client-side framework is allowed for module A. CSS preprocessors may be used. All HTML and CSS code must pass the W3C validations, even generated one.

Competitor Information

Module A will be assessed using Google Chrome and Firefox.

The axe browser extension is installed in Google Chrome and allows competitors to validate the website according to the accessibility standards WCAG.

Website Requirements

The goal of the website is to promote a suite of AI driven APIs. Potential customers must be able to inform themselves about the possibilities of those APIs, the pricing, and the team behind this product.

You can add more information and elements to all pages as you see fit. It's also possible to add links that point towards pages that do not exist yet (for example to a login page).

For each page, some example text and more are provided in the media files. However, not all the provided material has to be used.

The following pages must be implemented:

- **Home Page**
A short but catching home page. The idea is to show the product with some simple information and engaging media. It must contain links to the product and pricing pages where interested visitors can find more information.
- **Product Page**
The product page shows the whole AI API suite, by listing all available APIs and their features.
- **Pricing Page**
On the pricing page, the pricing of the APIs must be explained.
- **Team Page**
As the client is also looking to grow their team, a dedicated team page must show the current employees and highlight some core values of the company.

Global Elements

The following elements must be available on all pages:

- **Header:** must at least include a product name in form of text or a logo
- **Navigation:** contains easily accessible links to all pages
- **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

The goal of Module B is to create a server-side rendered website which customers can use to manage their API usage and billing. It is possible to use additional libraries in the frontend for interactivity, but rendering must be performed by the server-side framework, and *not* by a client-side framework calling an API.

The project does not have a database yet, and it is therefore also in the scope of this task to come up with a new database design and import a provided CSV file with partial example data.

As this website will be publicly exposed, it must implement the OWASP guidelines.

Competitor Information

Module B will be assessed using the provided version of Google Chrome. Different security aspects will be tested.

The design of the website is not important in this first iteration. The client will mostly focus on the functionality, but some basic styling is expected to make it readable and usable.

Website Requirements

The website must provide the following functionality.

Login

All other pages are protected and not accessible to non-authenticated users. Login must be possible by providing a username and a password. As the first version of the website will only allow users to sign up by invitation only, it is not necessary to be able to register accounts.

However, please create the following accounts:

- Username: demo1
Password: skills2023d1
- Username: demo2
Password: skills2023d2

The password must be stored in a secure way (hashed) in case someone gets access to the database.

Workspaces

Users can create as many workspaces as they like. Workspaces act as a way to separate the API usage. All of the following functionality (API tokens, billing quotas, bills) are scoped to a workspace.

After login, the user is redirected to their list of workspaces. On that page, they can create or update workspaces. Users can only access and modify their own namespaces.

For each workspace, they have the possibility to manage the API tokens, billing quotas, and bills. This additional functionality can also be provided on separate pages through links and does not have to be on the same page.

A workspace has the following attributes:

- A required **title** (max 100 characters, unique within the account)
- An optional **description** of any length

API Tokens

For each workspace, it is possible to create one or more API tokens. All available tokens of a workspace are listed with their name and the creation date.

The actual token is only revealed once when it is created. It is not possible to view the token again after creation.

Each token can be revoked. If it is revoked, it cannot be used anymore. It is also not possible to activate a revoked token again. In the token list, it is clearly visible when a token is revoked and the revocation date is shown.

A token has the following attributes:

- A required **name** (max 100 characters)
- A randomly generated **token** of at least 40 characters
- An automatically set **creation date**
- A **revocation date** which is set once it is revoked

Billing Quotas

It is possible to set and remove a user-defined billing quota per workspace. Each call to the provided API costs a certain amount of money. If a billing quota is set, it defines the maximum amount that can be spent on API calls within the assigned workspace per calendar month. If the quota is exceeded, usage of the API is not possible anymore for all API tokens of this workspace.

Quotas are displayed in the following way:

- If no quota is set, the costs of the current calendar month is shown, but it must also be clear that there is no maximum.
- If a quota is set, the costs and the maximum of the current calendar month are shown. Also, the number of remaining days in the current billing cycle is shown so users know when it will reset.

A billing quota has the following attributes:

- A **limit** in dollars for each calendar month

Bills

For each passed calendar month, a bill is generated and visible to the users.


Each API token can access services that are exposed through an API. The user pays for using those services based on the time it takes to compute the result. Each service can cost a different amount of money per second.

The bill contains the following data per API token and per accessed service which is exposed through an API:

- Usage of the service for a specific API token in seconds
- Cost of the service per second (static value per service)
- Usage cost of the specific token and service
- If a token did not access a service, that service must not be listed
- If a token was not used at all, that token must not be listed

A total row will also show the total cost over all API tokens and services. The total costs are rounded to two decimal points for displaying purpose only.

The client has provided the following example mockup to show their idea of a bill. However, it is possible to change it or come up with a completely different layout.

<div>Select Month </div>			
Token	Time	Per sec.	Total
production token			
Service #1	1039 s	0.0015 \$	1.56 \$
Service #2	501 s	0.005 \$	2.50 \$
development token			
Service #1	162 s	0.0015 \$	0.24 \$
<div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> </div>			
Total			4.31 \$

Example Data

To already have some example data to generate bills and check if calculations are correct, the client provided a CSV file with some billing related example data.

Import the data of this file into your own database schema. The imported data must be normalized.

Module C – Commercial Open API

Module C will focus on the implementation of a REST API.

Introduction

You are asked to create a REST API that will be used commercially. The API must be built with features which allow it to be commercialized and made available publicly and openly. The functionality created by you in this module, builds on top of the functionality created in module B. However, you are given a working solution of Module B. This solution includes a database dump including the data that is stored in it. You must use this schema and data and are not allowed to use your own Module B solution.

The functionality of the API will be to provide external access to a number of Artificial Intelligence (AI) services, which will be run as separate services. Those AI services expose a REST API themselves, which however should not be publicly accessible, as they do not have any level of security and reliability as the open API that you create. For development purposes you will have access to the AI services, but in a production environment, these services would be sealed away, so that only your server application could access the AI service endpoints. The code of the AI services is not available to you.

A documentation of the AI services APIs in the form of an OpenAPI specification and a generated documentation are provided. You are supposed to build code which wraps these APIs and exposes them through a single API. The AI services expose different ways of accessing their features and how to pass and receive data. Details on how to access the AI services are also provided in a separate document.

There is a specification for the open API that you are tasked to create. This specification is in the form of an OpenAPI specification and a generated documentation based on it. The specification describes the endpoints that you are supposed to implement. You can find the specification of the API in the file `ai-api/api.yml` and the generated documentation in the file `ai-api/api-docs.html`.

The API shall be implemented using one of the provided frameworks.

Assessment

Module C will be assessed using tools which directly access the API created by you. The API will be tested for its functionality and its adherence to the specification. The API will also be tested for its security and reliability.

Any modifications in the provided backend of previous modules, including any changes to the part of the database that was given, will not be taken into account.

Instructions

The API that you are supposed to create builds on top of the functionality created in module B. You may need to extend the schema of the database to store additional data. Please provide a database SQL dump of the schema and data that you are using. Please do not change the given tables and data, but also include it in your final database dump.

Authentication

The API endpoints must require an API token to be passed in the `X-API-TOKEN` header. The token can be created by the user using the solution of module B. The token shall be used to identify the user and workspace. If the token is not valid, the API must return an `401` error.

Quotas

The API endpoints must enforce the user-defined quotas. The quotas are defined by the user in the solution of module B. The quotas define the maximum that the user wants to spend per workspace in total. It is not known at the start of the API call if the quota will be exceeded. The API must

therefore allow the user to make the call if the quota is not exceeded, but must also enforce the quota if it is exceeded. The last API call may lead to exceeding the quota, but that is accepted. The API must return an 403 error if the quota is already exceeded.

Billing

The API endpoints must track the AI service usage and update the user's billing data accordingly. If an operation fails, the billing data must not be updated. The billing data generated by your API service can be viewed using the solution of module B. The price of a request is computed by multiplying the configured price of that AI service with the number of started milliseconds of the request. The price of a request is rounded up to the next 100th of a cent. If the service uses an asynchronous API, the timer starts to count when the job was started until the job is finished.

Error Handling

There several types of errors that can occur. The API endpoints must handle these errors and return the appropriate HTTP status code. The API endpoints must also return a JSON object with an error message.

Errors:

- 400 Bad Request: The request is malformed. This error shall be returned if the request is malformed or if the request is missing required fields.
- 401 Unauthorized: The user is not authorized to make the request. This error shall be returned if the token is invalid.
- 403 Forbidden: The user is not allowed to make the request. This error shall be returned if the quota is exceeded.
- 404 Not Found: The requested resource was not found. This error shall be returned if the requested conversation, job or image was not found.
- 503 Service Unavailable: The service is not available. This error shall be returned if the AI service is not available or if the AI service returns an unexpected error.
-

AI Services

The AI services are

- provided as separate services
- are not part of the API that you are supposed to create.
- are not publicly accessible.
- are only accessed from the server application.
- are documented in the form of an OpenAPI specification and a generated documentation.

AI Service 1: ChatterBlast - Chat Bot

This service can be used to chat with.

Usage and Differences to the unified API:

The ChatterBlast API requires that you create the conversation first through an endpoint, before you can send a prompt. So, on the unified API, it's one call, behind the scenes you must do 2 calls. The conversation ID is generated by the code you write, is sent to the ChatterBlast API, provided to the user and stored in the database to prohibit concurrent calls.

After the creation of the conversation, the second endpoint can then be used to fetch a partial conversation response in plain text.

The service you build needs to fetch the `GET` endpoint and provide the partial answer to the user until it is final. The frontend will be polling for the answer. After the response is complete, the user can submit another prompt to continue the conversation. It must not be possible to submit a prompt while the service is still processing the previous conversation input. This means you need to store the state of the conversation on your database.

Billing:

An indicator `<EOF>Took {duration in millis}ms` is printed at the end once the answer is final. Example: `<EOF>Took 4254ms`. The duration signifies how long the service took to generate the response. This is the time that must be used to determine how long it took and consequently update the billing. That indicator must not be part of the response forwarded to the frontend.

Specification:

You can find the specification of the provided API in the file `provided-ai-services/chatterblast.yml` and the generated documentation in the file `provided-ai-services/chatterblast.html`.

AI Service 2: DreamWeaver - Image Generation

This service is used to dream up (generate) images based on a text prompt.

Usage and Differences to the unified API:

The DreamWeaver API accepts a JSON object with a text prompt field. The endpoint is asynchronous and returns a JSON object with a job ID. The job ID can be used to query the status, progress and an image URL that contains a preliminary and low resolution image of what has been generated so far. Once the job is finished, the result contains a resource ID and a high resolution image URL that can be used to retrieve the generated image. The user has further actions to choose from: Upscale 2x, zoom in 2x or zoom out 2x. These actions require the generated image resource ID and also return a job ID.

The image URLs will point to an image that is stored on the AI service server. Your server application needs to download the image and store it locally and generate a new URL for the frontend to access the image. This also needs to work for the preliminary images.

Billing:

You must store when the job started and when it ends to update the billing (since it's an async API call).

Specification:

You can find the specification of the provided API in the file `provided-ai-services/dreamweaver.yml` and the generated documentation in the file `provided-ai-services/dreamweaver.html`.

AI Service 3: MindReader - Image Recognition

This service is used to recognize objects in images.

Usage and Differences to the unified API:

The MindReader API accepts a `multipart/form-data` field `image` and returns a JSON object with the recognized objects and their probabilities. The endpoint is synchronous and returns the result within a few seconds.

The API requires a mapping of fields: The unified API you build must return `x`, `y` and `width`, `height` while the provided API returns `top`, `left`, `bottom`, and `right` for the bounding box.

Billing:

The duration of the request must be used to determine how long it took and consequently update the billing.

Specification:

You can find the specification of the provided API in the file `provided-ai-services/mindreader.yml` and the generated documentation in the file `provided-ai-services/mindreader.html`.

Module D – Interactive Frontend using an API

You are asked to create a frontend for the REST API from module C. Since the functionality created by you in this module builds on top of the functionality created in module B and C, you will be given a working solution of Module C. You must use the provided solution and are not allowed to build on top of your own Module C solution.

The users of the frontend will be able to discover the available AI services that the API provides. Each service is then represented on a separate page which exhibits several interactive elements, each custom to the service. There will be complex data inputs and outputs, some of them are asynchronous. The goal of the frontend is to hide this complexity from the user. The frontend must also handle errors and display them to the user in a comprehensible way.

Initially the input elements are disabled. When a user wants to start using the service, they are prompted to enter an API token, which can be generated with the solution from module B. The API token must then be sent to the server with every request. The API token shall also be stored in the current browser instance, so that the user does not have to enter it again, even if they reload the page or navigate to another AI service.

You must implement the frontend using a framework. It is possible to use additional libraries. The application must be a Single Page Application (SPA). The routing must be handled by the framework. Page reloads must present the same content to the user as previously visible, except unsaved user driven inputs or temporary outputs.

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 previous modules, including any changes to the database, will not be taken into account.

Error Handling

The API can sometimes return errors or the billing quota could be used up. The frontend must handle these errors and display them to the user in a comprehensible way. The following errors must be handled:

- `400 Bad Request` – The request was malformed. The user must be notified that they have entered invalid data.
- `401 Unauthorized` – The API token is invalid. The user must be prompted to enter a new API token.
- `403 Forbidden` – The billing quota has been used up. The user must be notified that they have to wait until the next month to use the service again or increase their quota.
- `503 Service Unavailable` – The service is temporarily unavailable. The user must be notified that the service is currently unavailable and to try again later.

Pages

The following pages must be implemented:

Home

The home page must display a list of all available AI services. Each service must be represented by a link to the corresponding page. The list must be sorted alphabetically by the name of the service.

Service ChatterBlast

The ChatterBlast service is a chatbot. The user can enter a message and the chatbot will respond with a message.

The page must contain the following elements:

- A text input field for the user to enter a message.
- A button to clear the text input.
- A button to send the message to the chatbot for the current conversation or create a new conversation if it's the first one.
- An area to display the response from the chatbot.
- A button to start a new conversation with the chatbot.

The responses from the chatbot must be shown as they become available, even if they are only partially complete. The response text must be animated in a typewriter style. The animation must contain a blinking cursor and each character must be rendered individually with a random delay between 2ms and 20ms. You must poll the backend to get the current response. The polling interval must be 1 second.

While the response is incomplete, the button to send a new message must be disabled.

Service DreamWeaver

The DreamWeaver service is an image generator. The user can enter a text and the service will generate a new image.

The page must contain the following elements:

- A text input field for the user to enter a text.
- A button to clear the text input.
- A button to generate a new image.
- An area to display the generated image.
- A button to save the image to the local file system.
- A button to upscale the image.
- A button to zoom in and out of the image.

A loading indicator with progress in percentage must be displayed while the image is being generated, and the preliminary images must be animated with a fade-in effect until the final image is available. You must poll the backend to check if the job progress and to retrieve the preliminary image. The polling interval must be 2 seconds.

While the image is being generated, the button to generate a new image must be disabled.

Service MindReader

The MindReader service is a mind reader. The user can upload an image and the service will recognize the objects in the image.

The page must contain the following elements:

- A file input field for the user to upload an image.
- A button to upload the image to the service.
- Once objects are recognized:
 - A message that indicates how many objects have been recognized.
 - The recognized objects are shown with a transparent rectangle with a red border on top of the image alongside a label in the upper left corner of the rectangle.

While the image is being uploaded and the objects recognized, the button to upload a new image must be disabled and a loading indicator must be shown.

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 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).

Task 2: Creating a Progressive Web App (PWA)

Your task is to create a progressive web app for an AI news site. The backend is already provided to you with an OpenAPI specification (`task2/api-spec.yaml` and `task2/api-spec.html`) explaining the available endpoints. The frontend must be created from scratch. However, functionality is more important than look and feel. A simple icon for the app is provided to you already.

The app has the following requirements:

- It can be installed on the user's device.
- There is one view, showing all recent news articles in a list. The API returns only the 10 most recent articles and only those 10 articles have to be shown.
- It must work offline, meaning the last successfully loaded news articles are shown if the user does not have an Internet connection.
- If the user is online, articles are always loaded from the API and not returned from cache.
- Notifications about new articles can be received as explained below and open the app to the list view when clicked.

Because PWAs do not work with file URLs, a simple HTTP server is provided to you. It can be started by running `npm start` inside the `task2/src` folder and serves the content of the whole directory on `localhost:8080`.

Notifications

As the competition takes place in an offline environment, it is not possible to implement real push notifications. Therefore, they must be implemented a bit differently and for this to work, the app needs to be always open.

- The app polls the provided endpoint every 10 seconds in the background
- If there is a new article, a notification must be shown, but only if the app is not visible at the moment (app window open in the background)
- The notification contains the title of the new article and an application icon

Task 3: Creating a Web Component

For this task, you are expected to create three web components and embed them in a simple website. You are not allowed to use any framework or library for this task.

<limited-textarea>

This component renders a textarea that has a character limit. However, this limit is not strictly enforced, meaning the user can still enter more characters but the field is not considered valid anymore. It is displayed to the user how many characters they have left. The color of the remaining character number changes to orange if there are 10% or less of the characters left, and to red once the limit is exceeded.

Attributes:

- **maxchars**: Maximum number of characters (default: 500)

Events:

- **change**: Triggered immediately each time the content of the textarea changed. Event attributes:
 - **value**: string Value of the textarea (if limit is exceeded, only contains the allowed number of characters)
 - **valid**: boolean If the content is valid (limit not exceeded)

Other requirements:

- The component must not inherit any other styles that might be globally set on the page (except for fonts)
- The attributes of the component can change at any time and the component is updated automatically in this case
- The textarea must take the full available width and is not resizable
- The example website adds an event listener for the change event that logs the event data to the browser's console
- The x characters left text is located below the textarea and has the color #666
- When the limit is exceeded, a negative value is shown for the x characters left label indicating how many characters the user has to delete
- The number of characters has the color #000 by default, #f0620d if below 10%, or #ea1010 when the limit is exceeded

Example:

```
<limited-textarea maxchars="250"></limited-textarea>
```

This is a test. This is a test. This is a test. This is a test. This is a test.
This is a test. This is a test. This is a test. This is a test. This is a test.
This is a test. This is a test. This is a test. This is a test. This

22 characters left

<confirmation-modal>

Create a simple modal that displays any content that is provided, as well as a button with the provided label which closes the modal. It opens automatically when the modal element is created.

Attributes:

- `label`: Button label which closes the modal (default: Ok)
- `children`: Any valid HTML element which will be rendered inside the modal.

Events:

- `confirm`: Triggered when the modal is closed. There are no event attributes.

Other requirements:

- The attributes of the component can change at any time and the component is updated automatically in this case.
- While the modal is open, the rest of the website has an overlay of color `rgba(0, 0, 0, 0.3)`
- The button is aligned to the right and has a margin of at least 10px to the rest of the content.
- The example website adds an event listener for the confirm event that logs when the modal is closed.

Example:

```
<confirmation-modal label="Accept">
  <h2>This is my modal</h2>
  And this is some content.<br/>
  I can be <b style="color: green">any</b> HTML.
</confirmation-modal>
```



<count-down>

The countdown component counts down to the given time and date and displays a message once that is reached.

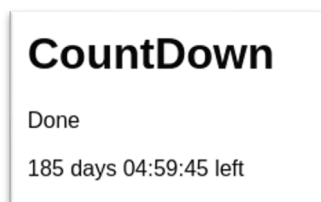
Attributes:

- `date`: Target date as an ISO 8601 string
- `message`: Message which will be displayed once the date is reached

Other requirements:

- The attributes of the component can change at any time and the component is updated automatically in this case
- Once the target is reached, the countdown will be replaced with the provided message.
- The remaining time is displayed in the following format: 10 days 17:03:10 left (hours, minutes, and seconds have always 2 digits)
- The remaining time gets updated automatically every second
- The example website has two countdowns on the page:
 - The first one counts down to 15 seconds in the future (after the page is loaded) and shows the message `Done` when completed
 - The second one counts down to 01.01.2024 00:00:00 and shows the message `Happy new year!` when completed

Example:



Module F – Collaborative Challenge

After having created most components for the client, you are now asked to form a group with other competitors to create your own creative AI service that can be integrated into the system built in previous modules. You can be as creative as you want, but the function should emulate an AI. The functionality itself can be implemented in any way, and it can also be faked. In fact, given the limited time, your only option likely is to fake it. Imagine writing a simple switch-case or return random values. The focus is on the creativity and the presentation of your idea.

However, the service must be implemented and working, so the function could actually be integrated into the system. You don't have to integrate it into the frontend nor backend you built in previous modules, but you must be able to demo it in some way or form.

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

Your presentation must be divided into three parts:

1. A Hook to start the presentation and catch the audience's attention.
2. A Middle Part where you explain your work and might also demonstrate something.
3. 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 service 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 service which can be integrated into the system, 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: TBD](#)

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 Microsoft OS or Ubuntu Linux 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

Every Test Project must be accompanied by an associated marking scheme matching the assessment criteria as given in the Technical Description (Marking Summary). For each of these criteria a detailed list of aspects to be assessed must be defined.

- See file: ES2023_marking_scheme_Web_Development_17.xlsx