**INTERNET PROGRAMMING**Laboratory work 2-1  
Variant – 7

Prepared by:

student of *Ba-121-22-4-SE*,

*Anna Kuts`*

Accepted by:

*Makliuk O.O.*

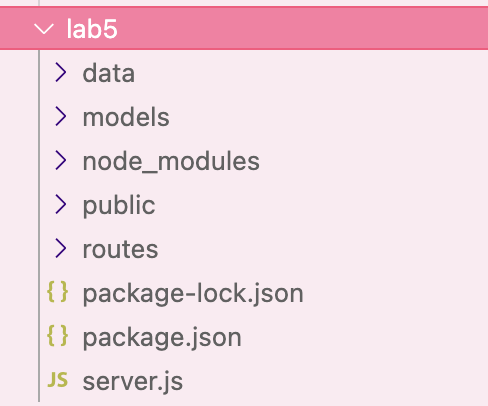
**PART 1: REST API   
Task**

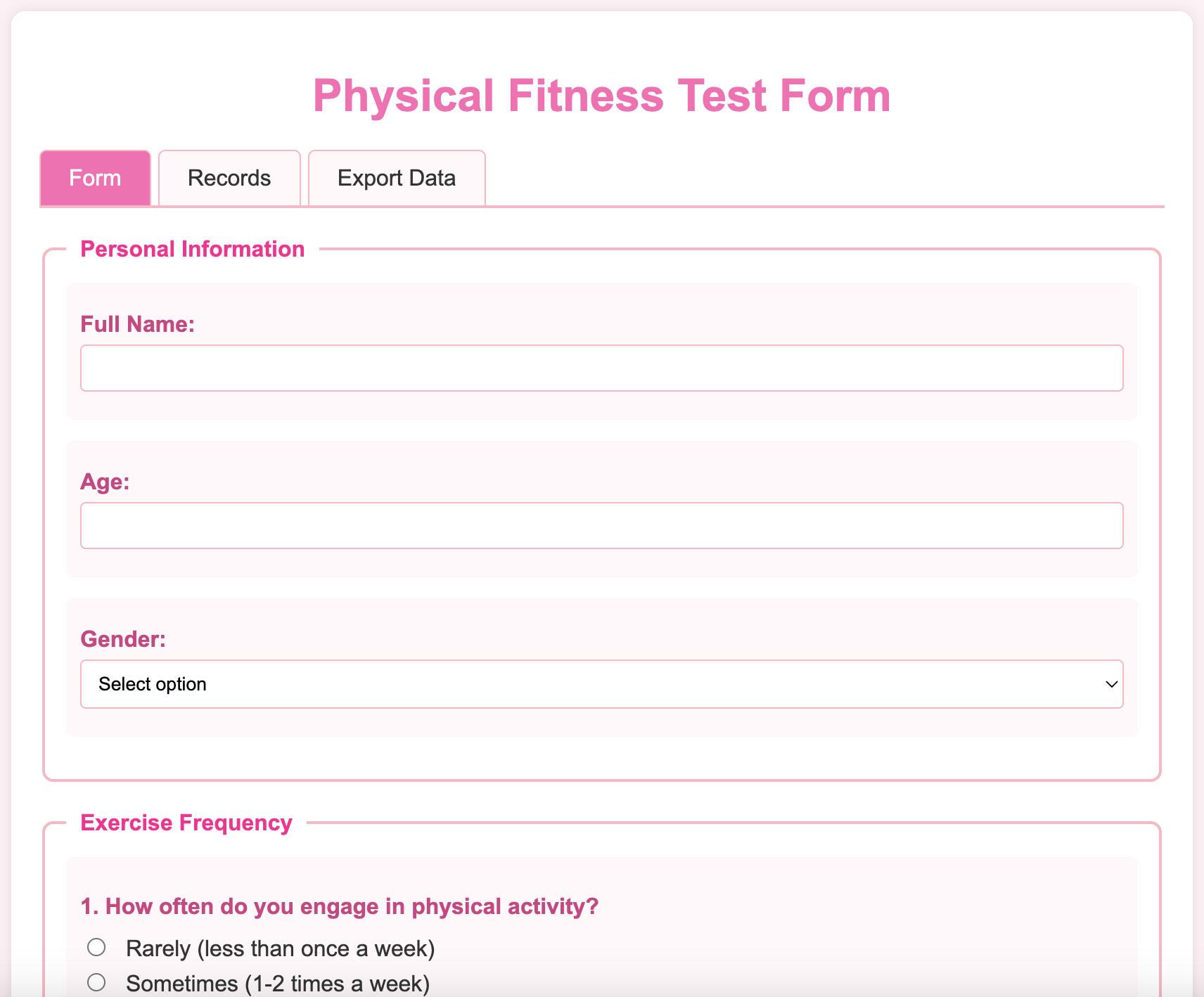
1. Creating a client-side HTML+JavaScript to create form data (see Lab 1-4 option)

2. Creating a server-side (Node/js+express.js,) that executes client requests:

* Creating data from record forms
* Updating records
* Writing data to files (.xml, .json).
* Reading records.
* Deleting records.

**Solution**



  
<https://github.com/Jasokaa/IP-Olenin>

**Conclusion**

In this work, I learnt the basics and use of Rest API using the example of creating a web application using express.js tools.

1. What is REST API and what are its main principles?

* REST API (Representational State Transfer Application Programming Interface) is an architectural style for designing networked applications. It uses standard HTTP methods to interact with resources represented as URLs.
* Main principles of REST:
  + Statelessness: Each request from client to server must contain all information needed to understand and process the request. The server does not store client context between requests.
  + Client-Server Architecture: Separation of client and server concerns.
  + Uniform Interface: Resources are identified by URLs, and standard HTTP methods (GET, POST, PUT, DELETE) are used.
  + Resource-Based: Everything is considered a resource, accessed via URIs.
  + Cacheable: Responses must define themselves as cacheable or not to improve performance.
  + Layered System: Architecture can have multiple layers (proxies, gateways) to improve scalability.

2. What HTTP methods are used in REST API and what are they for?

* GET: Retrieve data from the server (read-only).
* POST: Create a new resource on the server.
* PUT: Update an existing resource or create it if it does not exist (idempotent).
* DELETE: Remove a resource from the server.

3. How to distinguish RESTful API from other types of APIs?

* RESTful APIs strictly follow REST architectural principles, such as statelessness, resource-based URLs, and use of standard HTTP methods.
* They use standard HTTP status codes and methods uniformly.
* Other APIs might use different protocols (like SOAP), rely on custom methods, or maintain session states, which REST avoids.

4. How to ensure data security in REST API, specifically authentication and authorization?

* Authentication: Verify the identity of the user/client, often using methods like:
  + API keys
  + OAuth tokens
  + JWT (JSON Web Tokens)
  + Basic Authentication over HTTPS
* Authorization: Control access rights to resources based on authenticated user’s permissions. Often implemented through scopes or roles.
* Use HTTPS to encrypt data in transit.
* Validate all inputs to prevent injection attacks.
* Implement rate limiting and logging.

5. What are best practices for designing REST API for ease of use and maintainability?

* Use clear and consistent naming conventions for endpoints (e.g., nouns for resources).
* Use proper HTTP status codes to indicate the result of operations.
* Support versioning in URLs or headers to avoid breaking changes.
* Provide meaningful error messages and documentation.
* Keep URLs simple and intuitive.
* Use pagination, filtering, and sorting for large datasets.
* Support HATEOAS (Hypermedia as the Engine of Application State) to help clients discover actions dynamically.
* Ensure statelessness in requests.
* Use JSON as the standard data format unless otherwise needed.