API Documentation Diploma Project Requirements

Scope and Goals

The goal of this document is to provide students with a clear overview of the expectations and evaluation criteria for diploma projects that include automated testing.

The document defines:

- Minimum requirements – mandatory criteria to obtain a passing grade (grade = 5).
- Maximum requirements – advanced criteria to achieve the highest grade (grade = 10).
- Major errors and shortcomings – critical issues resulting in penalties.

Minimum Requirements (for a passing grade)

1. **Result should be structured and well-organized API documentation set**, covering the essential components of an API reference (endpoints, parameters, request/response formats, authentication, errors). The documentation must be clear, technically accurate, and easy to navigate.
2. **The student must provide a complete API specification**, created using industry standards — such as **OpenAPI/Swagger**, **GraphQL schema**, or **gRPC proto definitions**. The specification must be validated using appropriate tools (Swagger Editor, Spectral, Redocly, etc.) and provided as a version-controlled file (YAML/JSON).
3. **Each API endpoint must be documented with meaningful examples**, including at minimum:
    - sample requests
    - sample responses
    - HTTP status codes
    - error messages

   - data model descriptions
     Examples must reflect real-world usage and cover both typical and edge-case scenarios.

4. **The project must include a minimum of one "Getting Started" guide and at least one step-by-step integration tutorial**, showing how a developer would authenticate and interact with the API in a practical workflow.
5. **If the API supports multiple operations or resources, the documentation must include a high-level architecture overview**, such as:
   - system context diagram
   - request flow diagram
   - explanation of modules/resources
     This overview must help readers understand how the API fits into its broader system.
6. **The documentation must be published or prepared in a developer-accessible format**, such as:
   - an online documentation site (GitBook, Docusaurus, Redoc, Swagger UI)
   - a well-structured Markdown documentation repository
     The deliverable must be clean, structured, and operational.
7. **The written report must describe the documentation strategy**, including tools used, standards followed, naming conventions, versioning approach, and formatting rules.
   The student must also evaluate any known gaps or limitations in the documentation.
8. **All source files must follow a consistent formatting and naming convention**, including clear folder structure (e.g., /reference, /guides, /examples, /schemas), ensuring traceability and maintainability.

Maximum Requirements (for the highest grade)

1. **The project must deliver a highly comprehensive API documentation system**, including not only reference docs but also conceptual guides, best practices, tutorials, onboarding flows, release notes, and versioning documentation. The

documentation must demonstrate mastery of information architecture and developer experience (DX) design.

2. **The API documentation must thoroughly describe advanced topics**, such as:
   - complex request flows
   - pagination strategies
   - rate limits
   - authentication/authorization deep dive
   - error taxonomy
   - idempotency
   - versioning strategy
     The student must show deep understanding of API design principles and standards.
3. **The project must include detailed diagrams and service interaction descriptions**, for example:
   - sequence diagrams
   - component diagrams
   - dependency maps
   - request–response lifecycles
     For multi-service or microservice APIs, the documentation should include cross-service contract explanations.
4. **API quality must be enhanced using advanced documentation tools or methodologies**, such as:
   - API linting or schema validation (Spectral, Redocly CLI)
   - contract testing definitions (e.g., Pact contracts)
   - auto-generated examples
   - API mock servers
   - developer sandbox environments

Major Errors and Shortcomings Affecting Evaluation

Category	**Example of Issue**	**Penalty**
Completeness	Key endpoints undocumented; missing request/response examples; incomplete data models	**–2 points**
Accuracy	Incorrect endpoint descriptions, parameters, or status codes; mismatched examples vs. actual API behavior	**–1 point**
Structure & Navigation	Poor documentation organization; unclear hierarchy; missing TOC; hard-to-find sections	**–1 point**
Specification Quality	Invalid or inconsistent OpenAPI/Swagger schema; schema fails validation tools	**–1 point**
Developer Experience	Missing authentication guide, onboarding flow, or integration tutorial	**–1 point**
Examples	Missing or trivial example requests/responses; examples not runnable or inconsistent	**–2 points**
Versioning & Updates	No versioning strategy or missing changelog; inconsistent endpoint versions	**–1 point**
Security Documentation	Missing authentication/authorization details, unclear token handling, or missing security warnings	**–2 points**
Error Handling	Error codes undocumented or incomplete; no explanation of error structure or recovery steps	**–1 point**