SWE Report

N-tier Application Architecture and Development

E-commerce app

Name: Amr Khaled Gaber

ID: 21100834

Alamein International University

Advanced Computer Technology (ACT)

Contents

1. Executive Summary:

This report outlines the development of a N-tier Application, implemented to provide modularity and separation of concerns across different layers of the system: the API layer, Business Logic Layer (BLL), and Data Access Layer (DAL). The architecture is designed to ensure scalability, maintainability, and separation of concerns, with each tier performing specific responsibilities. The application allows smooth communication between the user interface and the database through well-defined layers.

The following report elaborates on the project architecture, technologies used, implementation details, challenges encountered, and solutions provided. Future recommendations for improvement are also highlighted.

2. Introduction:

2.1 Project Overview:

The N-tier application is a software system designed with a layered architecture, dividing the application into three distinct layers:

1. **API Layer:** Responsible for handling incoming client requests and sending appropriate responses.
2. **Business Logic Layer (BLL):** Implements the core business rules and logic that the application must follow.
3. **Data Access Layer (DAL):** Manages database operations and handles data persistence.

2.2 Objective:

The objective of this project is to create a modular, scalable, and maintainable software system using the N-tier architecture, which separates the concerns of presentation, business logic, and data access to ensure ease of maintenance, scalability, and security.

2.3 Scope:

This report covers the system's architecture, design, implementation details, testing strategy, and evaluation of performance, along with challenges and future enhancements.

3. System Requirements:

3.1 Functional Requirements:

- The system should allow users to interact with the API for **CRUD** (Create, Read, Update, Delete) operations.
- The business logic should enforce rules on data processing before sending/receiving from the database.
- The system should persist data using the Data Access Layer and maintain data integrity**.**

3.2 Non-Functional Requirements:

- **Scalability:** The system should be able to scale to handle more requests and larger datasets.
- **Security:** Sensitive data should be encrypted and protected from unauthorized access.
- **Performance:** The system must ensure low response times and high availability.

- **Maintainability:** The code must follow best practices to ensure maintainability and ease of updates.

4. System Architecture:

4.1 Architecture Overview:

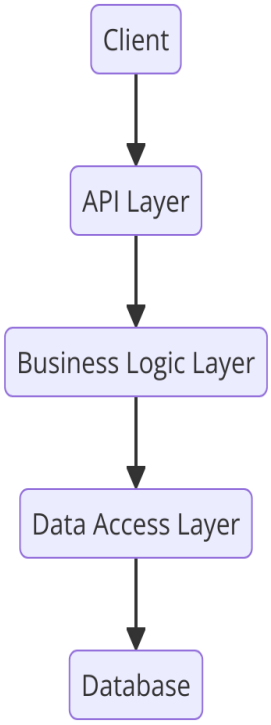
The system follows a typical **N-tier architecture**, dividing the application into the following tiers:

1. **API Layer**: Exposes HTTP endpoints and handles request/response routing.
2. **Business Logic Layer (BLL)**: Processes incoming requests, applies business rules, and orchestrates the logic.
3. **Data Access Layer (DAL)**: Manages communication with the database, handling data retrieval, updates, and persistence.

4.2 Technology Stack:

- **API Layer**: ASP.NET Core (C#), RESTful APIs
- **Business Logic Layer (BLL)**: C#, .NET Framework
- **Data Access Layer (DAL)**: Entity Framework (C#), SQL Server
- **Database**: Microsoft SQL Server

4.3 Architectural Diagram:



4.4 Layer Interaction:

- The **API layer** handles requests from the client and routes them to the business logic.
- The **BLL** processes the request, applies rules, and interacts with the DAL.

- The **DAL** interacts with the database to retrieve or store data, and the result is sent back through the BLL to the API for a response to the client.

5. Design and Implementation:

5.1 API Layer:

- **Responsibility**: The API layer exposes endpoints for client interaction.
- **Technologies** Used: ASP.NET Core, HTTP, JSON
- **Endpoints**: Endpoints for CRUD operations (e.g., GET /items, POST /items, PUT /items/{id}, DELETE /items/{id}).

5.2 Business Logic Layer (BLL):

- **Responsibility**: This layer contains the core business logic that processes data from the API before interacting with the database.
- **Technologies** Used: C# with .NET Core
- **Key Logic**: Implemented services that handle validation, business rules, and orchestration for CRUD operations.

5.3 Data Access Layer (DAL):

- **Responsibility**: Handles communication with the database, including querying, updating, and managing database transactions.
- **Technologies** Used: Entity Framework, SQL Server.
- **Data** Model: The data model is mapped using Entity Framework, which enables easy interaction between the application and SQL Server.

6. Challenges and Solutions:

Several challenges were encountered during the development of the N-tier application:

- Database Connectivity: Ensuring smooth communication between the DAL and the database, especially handling connection pooling. This was solved by configuring Entity Framework connection settings for better resource management.
- Scalability Concerns: The system needed to be designed with future scalability in mind. By using the repository pattern and separating concerns, the system can easily scale horizontally as demand increases.

- Maintaining Business Rules Consistency: The BLL had to ensure that all business rules were consistently applied. This was addressed by creating a centralized service to validate requests before any database transaction.