ZUMI BACKEND ARCHITECTURE

1. Repository: zumi-backend
   Purpose:
   This repository will hold the source code for the REST API services. It will manage business logic,
   data handling, and integrations with external systems.
   Key Features:

- Handles requests from zumi-client.
- Interfaces with databases, third-party services, and GCP backend services.
- Implements authentication, authorization, and other server-side logic.
- Built with scalability and high performance in mind.

Directory Structure:
/ # Root directory
|-- src/ # Application source code
| |-- app/ # Request handling logic
| | |-- modules/ # All necessary classes grouped by specific target
| | | |-- user/ # Contains (controller, use-case(business logic), dto)
| | | | |-- controller/ # Request handler
| | | | |-- use-case/ # Business logic
| | | | |-- dto/ # Business model (wrapper above appropriate database entity)
| |-- core/ # Reusable shared constructions such as constants, functions etc.
| | |-- data/ # Reusable constants, classes so on
| | |-- utils/ # Different utility functions
| |-- libs/ # Reusable shared constructions such as constants, functions etc.
| |-- gateways/ # All necessary types.

|-- tests/ # Unit and integration tests
|-- docs/ # API documentation (Swagger, Postman collections)
|-- scripts/ # Deployment and maintenance scripts
|-- .env.example # Example environment variables
|-- Dockerfile # Docker configuration for containerization
|-- Makefile # Automation scripts for development tasks
|--
README.md # Overview of the repository
Versioning & Workflow:

- Follows Gitflow for branch management (e.g., main, develop, feature/*).
- CI/CD pipeline integrates linting, unit tests, and automated deployments to staging and
  production.