Technical Documentation & Knowledge Base Details and Example

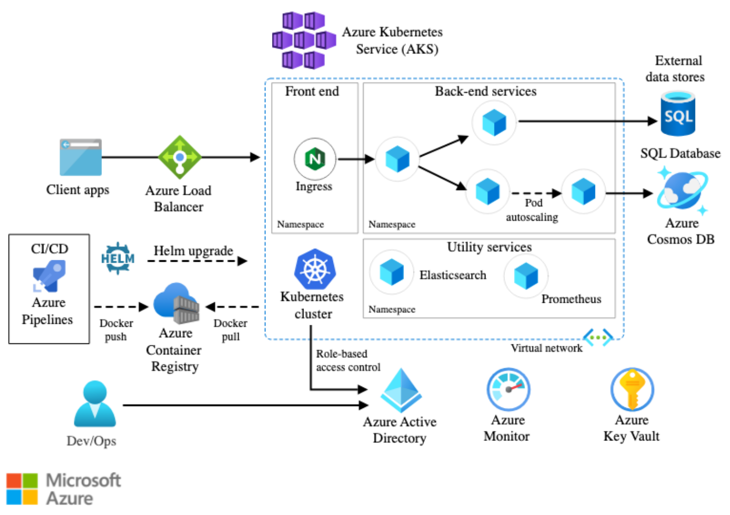
System or Solution Architecture

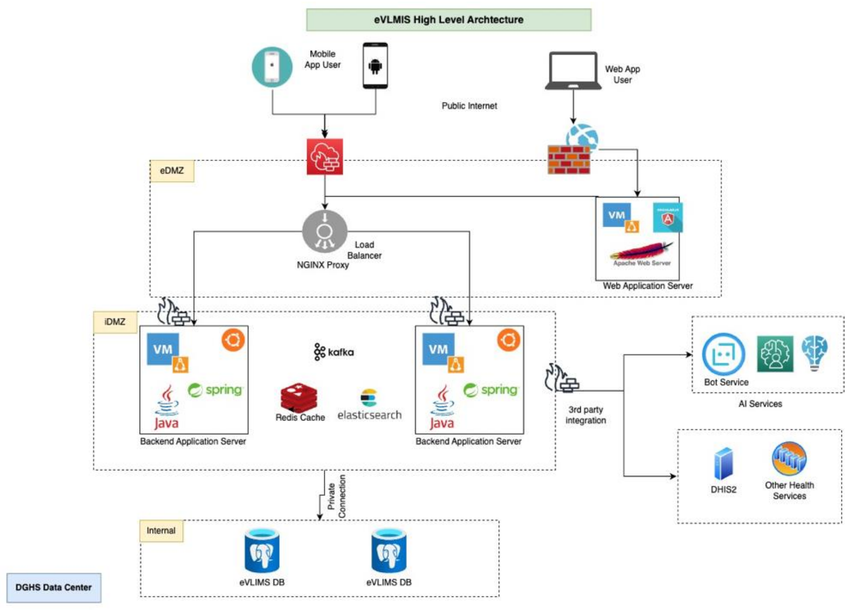
- All system components including third-party: Mandatory
- Networking layer: Mandatory
- Connectivity between the components: Mandatory

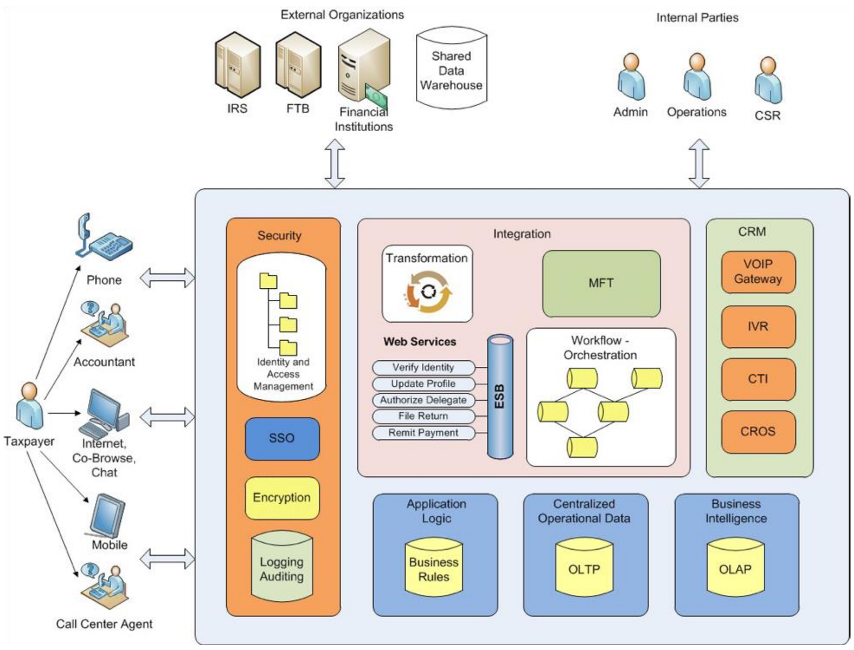
Tips:

- Focus on system components, their interactions, boundaries, security, scalability, and users.
- Use clear and meaningful icons, lines, arrows, images etc.

Sample (3 given):







Coding structure and guidelines

Tips: Add comprehensive documentation to the README.md file in the code repository. Clearly outline the code structure and guidelines, ensuring that any developer can easily start working on the project with minimal effort, approximately 10% of the team member's hourly effort. Consider creating a "Docs" folder in the root directory for more detailed or complex information in separate files.
Sample : test-repo-for-docs/README.md at main · tazbir-bs/test-repo-for-docs (github.com)

API Design Guideline

Tips: Create an API design guideline to serve as a guidebook for new developers and as a foundation for ongoing maintenance. Document this in the README.md file for easy reference.
Sample: test-repo-for-docs/README.md at main · tazbir-bs/test-repo-for-docs (github.com)

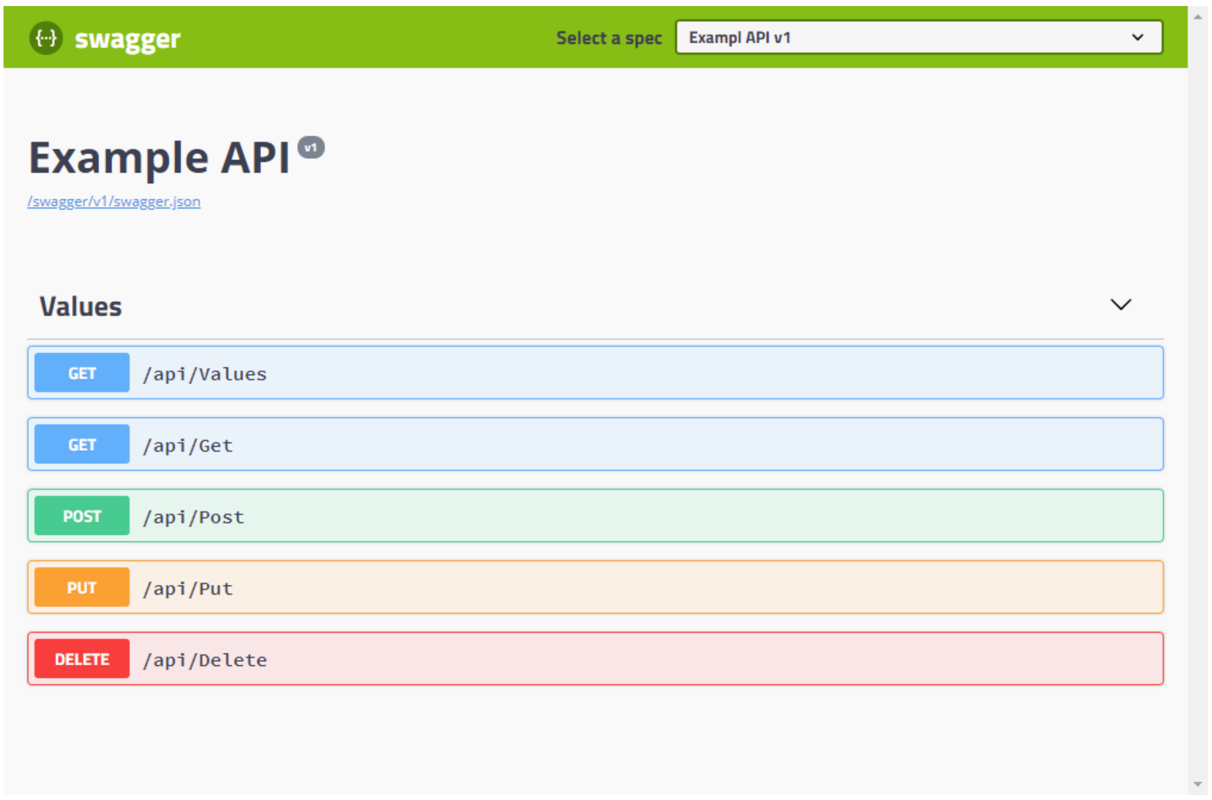
Logging guideline

Tips: use clear messages for easy issue identification, include timestamps and error codes, and regularly update log levels to maintain a balance between valuable information and avoiding unnecessary noise. This ensures efficient debugging and enhances overall code maintainability. Maintain the guidelines for logging into the README.md file.
Sample: test-repo-for-docs/README.md at main · tazbir-bs/test-repo-for-docs (github.com)

API Documentation

Tips: It can automatically generate API endpoints using Swagger at minimum. If you are maintaining API documentation differently, please document the process in this section with proper links and descriptions.

Sample:



Example API v1
/swagger/v1/swagger.json

Values

```
GET     /api/Values
GET     /api/Get
POST    /api/Post
PUT     /api/Put
DELETE  /api/Delete
```

Version Control and Release Guideline (GIT)

Tips: If you have a solid version control strategy, document it in the README.md file and keep it updated. If your current strategy needs improvement, follow the example below, ensuring you incorporate at least 80% of the provided process. This helps maintain an organized and efficient version control system for your project.
Sample: test-repo-for-docs/README.md at main · tazbir-bs/test-repo-for-docs (github.com)

Code Review Process and Checklist

Step 1: Add Code Review Checklist to Project Repo
BS Code Review Checklist

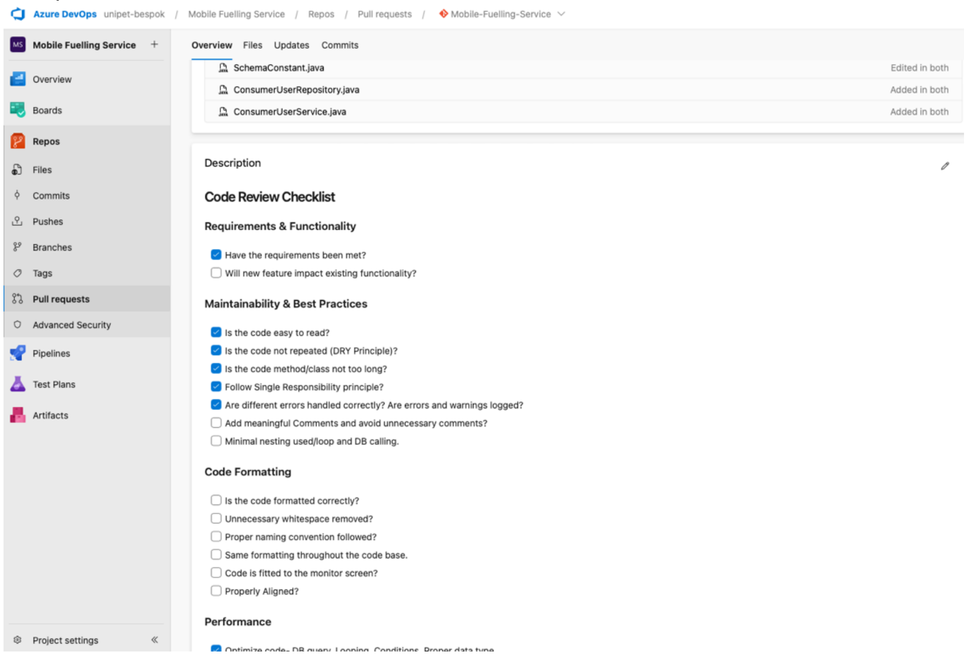
A Text format document can be developed as a checklist and saved as PullRequest_Guideline.md in the root folder of the project.

Note: If the git repo provider does not support the pull request template ignore the pull request template settings. Follow the checklist manually. Upload the review report at BS erp under project menu.

As different platform uses different techniques to enforce templating the PR checklist, we encourage you to follow the guidelines of the platform to automatically load the template in the PR description when created. Depending on the source control (GitHub, Azure DevOps), need to adapt the pull request template configuration.

- Github pull request template
- Azure DevOps Git pull request template

Sample:



SchemaConstant.java	Edited in both
ConsumerUserRepository.java	Added in both
ConsumerUserService.java	Added in both

Description

Code Review Checklist

Requirements & Functionality

- [✓] Have the requirements been met?
- [ ] Will new feature impact existing functionality?

Maintainability & Best Practices

- [✓] Is the code easy to read?
- [✓] Is the code not repeated (DRY Principle)?
- [✓] Is the code method/class not too long?
- [✓] Follow Single Responsibility principle?
- [✓] Are different errors handled correctly? Are errors and warnings logged?
- [ ] Add meaningful Comments and avoid unnecessary comments?
- [ ] Minimal nesting used/loop and DB calling.

Code Formatting

- [ ] Is the code formatted correctly?
- [ ] Unnecessary whitespace removed?
- [ ] Proper naming convention followed?
- [ ] Same formatting throughout the code base.
- [ ] Code is fitted to the monitor screen?
- [ ] Properly Aligned?

Performance

- [✓] Optimize code- DB query, Looping, Conditions, Proper data type

Step 2: Configure Pull Request Review Setting at Git

Configure git pull request for review settings. Developers write code and create a pull request. Another developer or tech lead must review and approve the pull request. Review Policy will depend on the git branch and project types and stages. The internal team need to decide on the review policy.

- Github pull request review management
- Azure DevOps pull request review management

Data Architecture and Details

Tips: Write database and storage details, database script backup plan and migration steps.
Sample:

Primary DB	SQL
DB Details	MSSQL v19
File Storage	AWS S3
Other Storage	Cosmos DB, Azure Key Vault

Note: if data is not present: keep the row blank

DB Update Scripts/Tools:

- Execute DDL (Data Definition Language) and DML (Data Manipulation Language) scripts using your preferred database management tool (e.g., SQL Server Management Studio, MySQL Workbench).
- Ensure proper permissions, backup data before executing, and review scripts for potential impacts on the database structure or data.

Backup Scripts/Tools:

- Use database-specific backup tools (e.g., mysqldump, pg_dump) or built-in features of your database management system to regularly create backups.
- Schedule backups to run at appropriate intervals, store them securely, and periodically test restoration processes to ensure data recovery readiness.

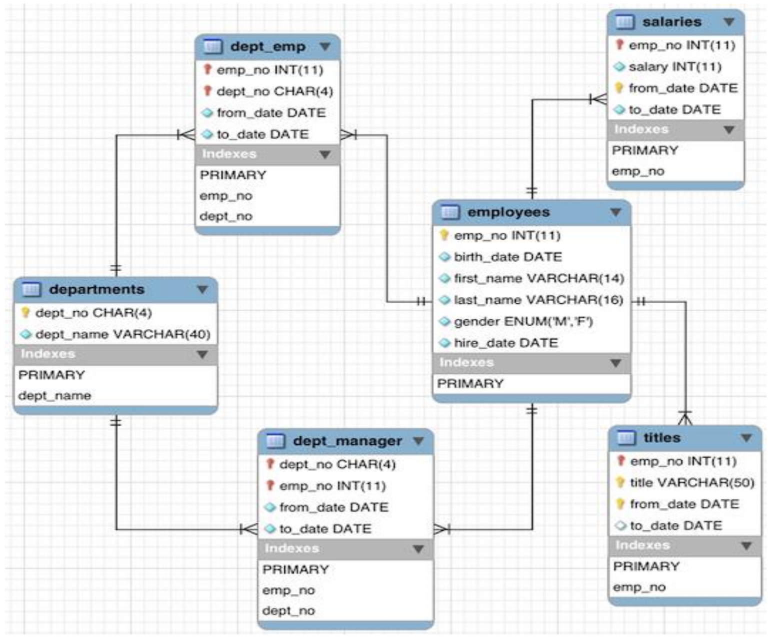
DB Migration:

- Use preferred ORM library to generate migrations.
- Do not directly update the database without creating migrations.

Entity Relationship Diagrams (ERD)

- Generate when required using a DB tool
- Use any professional diagram drawing tool if you just started the project.

Sample:



DevOps details (CI-CD process and scripts)

Tips: To have a satisfied DevOps culture, there are some prerequisites to fulfil.

- Maintain a well-structured git-flow with the standard branching system mentioned in the VCS section.
- Define a deployment pipeline with multiple stages (e.g., Dev, QA, Staging, Production).
- Expected to have minimal documentation on how to
  - Prepare the environment to
    - build the application
    - Run the application
  - Package the application

- Manage secrets and sensitive information securely (e.g., environment variables or secret management tools).

Sample: https://github.com/tazbir-bs/test-repo-for-docs/blob/main/README.md#devops-practices

Testing Strategy Document

The Test Strategy Document is a living document that is created in the project's Quality Definition Phase after the Requirements have been specified. This describes the scope, approach, resources, and schedules for the testing activities of the project. This includes defining what will be tested, who will perform testing, how testing will be managed and associated risks and contingencies.

Diagram Design

Tips:

- Use industry-standard design language like UML where applicable
- Use clear and meaningful icons, lines, arrows, images etc
- Class diagram, Sequence Diagram, Use Case Diagram: if required by any part.