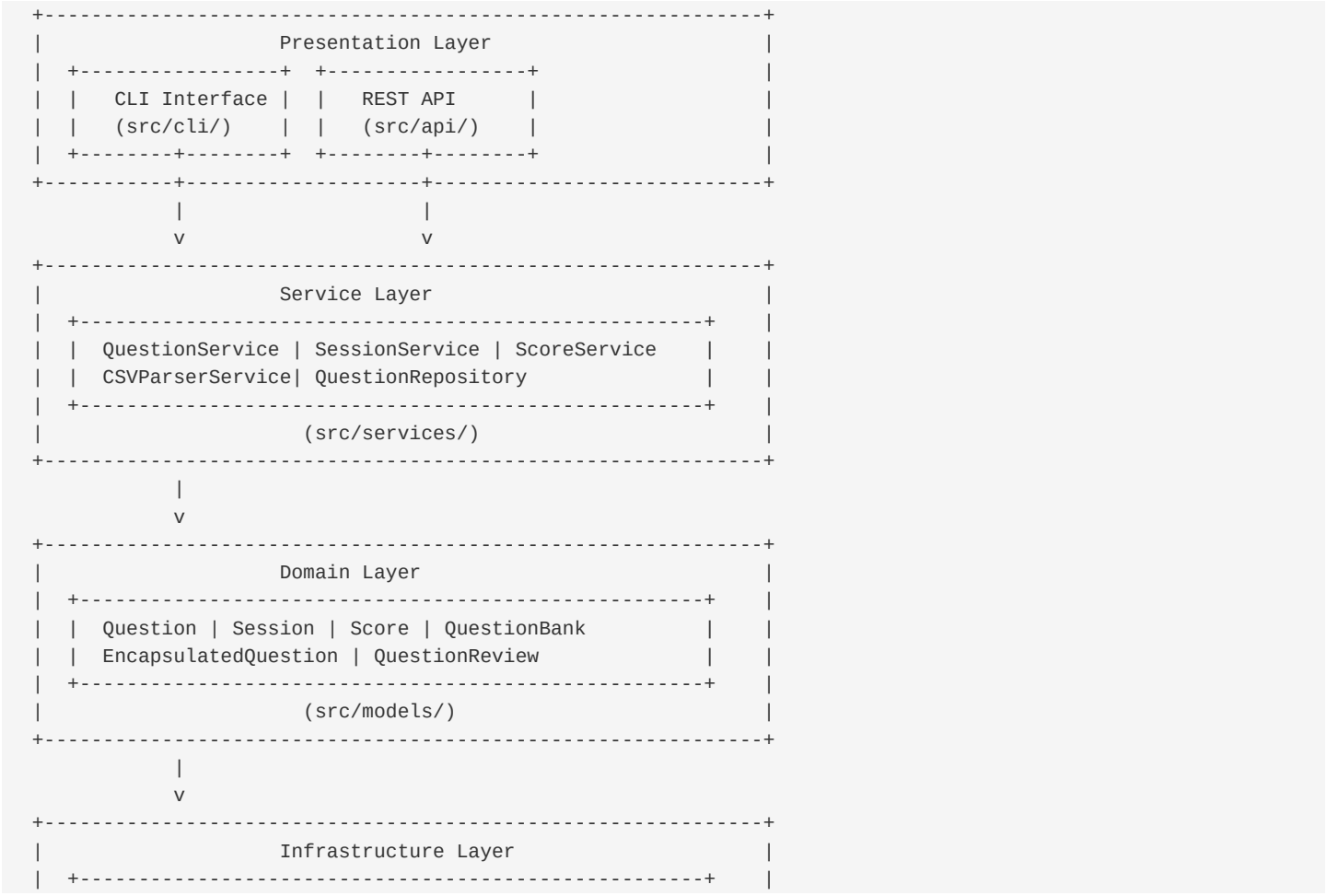
Q&A Practice Application - Architecture Documentation

This document provides a comprehensive overview of the application architecture, project structure, and static code analysis results.

Table of Contents

Architecture Overview

The Q&A Practice Application follows a layered architecture with clear separation of concerns:



```
|  |  Config | Logging | Exceptions | Algorithms         |    |
|  +-------------------------------------------------+    |
|                      (src/utils/)                       |
+---------------------------------------------------------+
            |
            v
+---------------------------------------------------------+
|                    Data Layer                           |
|  +-------------------------------------------------+    |
|  |  CSV Files (question-bank.csv)                  |    |
|  +-------------------------------------------------+    |
|                (src/main/resources/)                    |
+---------------------------------------------------------+
```

Key Architectural Principles

- Separation of Concerns: Each layer has distinct responsibilities

---

Project Structure

```
q_and_a_practise1/
+-- src/                          # Source code
|   +-- api/                      # REST API (FastAPI)
|   |   +-- main.py              # API entry point
|   |   +-- routes/              # API route handlers
|   |       +-- difficulties.py  # Difficulty endpoints
|   |       +-- questions.py     # Question endpoints
|   |       +-- scores.py        # Score endpoints
|   |       +-- sessions.py      # Session endpoints
|   |       +-- topics.py        # Topic endpoints
|   |
|   +-- cli/                      # Command Line Interface
|   |   +-- main.py              # CLI entry point
|   |   +-- commands.py          # CLI command handlers
|   |
|   +-- models/                   # Domain models
|   |   +-- base_question.py     # Base question class
|   |   +-- question.py          # Question model
|   |   +-- encapsulated_question.py  # Encapsulated question
|   |   +-- question_bank.py     # Question bank container
|   |   +-- question_review.py   # Question review model
|   |   +-- session.py           # User session model
|   |   +-- score.py             # Score model
|   |
|   +-- services/                 # Business logic
|   |   +-- interfaces.py        # Service interfaces
|   |   +-- question_service.py  # Question operations
|   |   +-- question_repository.py  # Data access
|   |   +-- session_service.py   # Session management
|   |   +-- score_service.py     # Score calculation
|   |   +-- csv_parser.py        # CSV file operations
|   |   +-- di_setup.py          # Dependency injection
|   |
|   +-- utils/                    # Utilities
|   |   +-- algorithms.py        # Sorting/searching
|   |   +-- config.py            # Configuration
|   |   +-- container.py         # DI container
|   |   +-- exceptions.py        # Custom exceptions
```

```
|   |   +-- logging_config.py    # Logging setup
|   |
|   +-- main/resources/           # Data files
|   |   +-- question-bank.csv    # Question data
|   |
|   +-- web/                      # Web utilities
|       +-- main.py             # Web entry point
|
+-- tests/                        # Test suite
|   +-- unit/                    # Unit tests
|   |   +-- test_api/           # API tests
|   |   +-- test_cli/           # CLI tests
|   |   +-- test_models/        # Model tests
|   |   +-- test_services/      # Service tests
|   |   +-- test_utils/         # Utility tests
|   +-- integration/             # Integration tests
|
+-- specs/                        # Specifications
|   +-- 001-qa-app/
|       +-- spec.md             # Feature specification
|       +-- plan.md             # Implementation plan
|       +-- tasks.md            # Task tracking
|
+-- docs/                         # Documentation
|   +-- user_guide.md           # User guide
|   +-- architecture.md         # This file
|
+-- pyproject.toml              # Python project config
+-- uv.lock                     # Dependency lock file
+-- README.md                   # Project readme
```

Module Descriptions

`src/api/` - REST API Layer

FastAPI-based REST API providing HTTP endpoints for the Q&A application.

File	Purpose
`main.py`	Application entry point, mi...
`routes/topics.py`	GET /api/topics - List avai...
`routes/difficulties.py`	GET /api/difficulties - Lis...
`routes/sessions.py`	Session CRUD operations
`routes/questions.py`	Question retrieval endpoints
`routes/scores.py`	Score calculation and retri...

`src/cli/` - Command Line Interface

Terminal-based interface for interactive Q&A practice.

File	Purpose
`main.py`	CLI entry point, argument p...
`commands.py`	Command implementations (li...

`src/models/` - Domain Models

Core business entities with validation and behavior.

File	Purpose
`base_question.py`	Abstract base class for que...
`question.py`	Question entity with valida...
`encapsulated_question.py`	Question with encapsulated ...
`question_bank.py`	Collection of questions wit...
`question_review.py`	Review data for answered qu...
`session.py`	User session with state man...
`score.py`	Score calculation and stati...

`src/services/` - Business Logic Layer

Service classes implementing core business operations.

File	Purpose
`interfaces.py`	Abstract service interfaces
`question_service.py`	Question CRUD, search, filt...
`question_repository.py`	Data access abstraction
`session_service.py`	Session lifecycle management
`score_service.py`	Score calculation, recommen...
`csv_parser.py`	CSV file parsing and valida...
`di_setup.py`	Dependency injection config...

`src/utils/` - Infrastructure Utilities

Cross-cutting concerns and helper functions.

File	Purpose
`algorithms.py`	Sorting (bubble, quick, mer...
`config.py`	Application configuration m...
`container.py`	Dependency injection container
`exceptions.py`	Custom exception hierarchy
`logging_config.py`	Structured logging configur...

Design Patterns

Patterns Implemented

Pattern	Location	Purpose
Repository	`question_repository.py`	Abstract data access
Service Layer	`src/services/`	Encapsulate business logic

Dependency Injection	`di_setup.py`, `container.py`	Loose coupling
Factory	`CSVParserService`	Create question objects
Strategy	`algorithms.py`	Interchangeable sorting alg...
Template Method	`base_question.py`	Define algorithm skeleton
Observer	Session state changes	Event notification

SOLID Principles

- Single Responsibility: Each class has one reason to change

Static Code Analysis

Summary Statistics

Metric	Value
Total Python Files (src/)	35
**Total Python Files (tests...	29
Total Lines of Code (src/)	12,813
**Total Lines of Code (test...	11,074
Total Lines of Code	23,887
Code-to-Test Ratio	1.16:1

Lines of Code by Module

Module	Files	Lines of Code	% of Total
`src/services/`	7	5,112	39.9%
`src/models/`	7	3,365	26.3%
`src/utils/`	5	1,767	13.8%
`src/api/`	6	1,012	7.9%
`src/cli/`	2	778	6.1%
Total	**27**	**12,813**	**100%**

Functions and Classes by Module

Module	Functions	Classes	Total
`src/models/`	244	19	263
`src/services/`	134	13	147
`src/utils/`	75	16	91
`src/api/`	30	11	41
`src/cli/`	18	1	19
Total	**501**	**60**	**561**

Cyclomatic Complexity by Module

Cyclomatic complexity measures the number of independent paths through code. Lower is better.

Module	Complexity	Rating
`src/api/`	38	[LOW] Low
`src/cli/`	75	[LOW] Low
`src/utils/`	119	[MED] Medium
`src/models/`	257	[MED] Medium
`src/services/`	578	[HIGH] High
Total	**1,067**	-

Complexity Analysis

- Low (< 100): Simple, easy to test and maintain

Average Complexity per Function

Module	Avg Complexity/Function
`src/api/`	1.27
`src/cli/`	4.17
`src/utils/`	1.59
`src/models/`	1.05
`src/services/`	4.31
Overall	**2.13**

Test Coverage Summary

Overall Coverage

Metric	Value
Total Tests	555
Tests Passing	555 (100%)
Overall Coverage	62%
Target Coverage	90%

Coverage by Module

Module	Stmts	Miss	Coverage	Status
`src/services/interfaces.py`	14	0	**100%**	[OK] Excellent
`src/services/question_repo`	69	0	**100%**	[OK] Excellent
`src/models/encapsulated_q`	257	12	**95%**	[OK] Excellent
`src/utils/exceptions.py`	41	3	**93%**	[OK] Excellent

`src/models/session.py`	133	16	**88%**	[OK] Good
`src/utils/logging_config.py`	87	13	**85%**	[OK] Good
`src/api/routes/topics.py`	17	3	**82%**	[OK] Good
`src/api/routes/difficultie...`	17	3	**82%**	[OK] Good
`src/api/main.py`	87	17	**80%**	[OK] Good
`src/api/routes/sessions.py`	138	30	**78%**	[OK] Good
`src/services/di_setup.py`	79	19	**76%**	[OK] Good
`src/models/question_review...`	80	19	**76%**	[OK] Good
`src/cli/main.py`	76	22	**71%**	[MED] Medium
`src/utils/algorithms.py`	324	98	**70%**	[MED] Medium
`src/utils/config.py`	97	32	**67%**	[MED] Medium
`src/models/base_question.py`	147	52	**65%**	[MED] Medium
`src/models/question_bank.py`	164	58	**65%**	[MED] Medium
`src/models/question.py`	225	83	**63%**	[MED] Medium
`src/utils/container.py`	29	12	**59%**	[MED] Medium
`src/cli/commands.py`	262	115	**56%**	[MED] Medium
`src/models/score.py`	154	67	**56%**	[MED] Medium
`src/services/session_servi...`	262	124	**53%**	[MED] Medium
`src/api/routes/questions.py`	51	25	**51%**	[MED] Medium
`src/api/routes/scores.py`	41	20	**51%**	[MED] Medium
`src/services/question_serv...`	642	313	**51%**	[MED] Medium
`src/services/score_service...`	257	149	**42%**	[CRIT] Low
`src/services/csv_parser.py`	464	285	**39%**	[CRIT] Low
TOTAL	**4214**	**1590**	**62%**	[MED] Medium

Coverage by Layer

Layer	Avg Coverage
API Layer (`src/api/`)	70%
CLI Layer (`src/cli/`)	64%
Models (`src/models/`)	73%
Services (`src/services/`)	66%
Utils (`src/utils/`)	72%

Test Distribution

Test Category	Count
Unit Tests	481
Contract Tests	30
Integration Tests	44
Quality Tests	80+
Total	**555+**

Recommendations for Improving Coverage

1. High Priority (< 50% coverage):

- csv_parser.py - Add tests for file operations and validation

---

Last Updated: December 8, 2025