CompetiQuest Testing Documentation

Group - 7

Aarushi Goel (Team Lead)	202412002	202412002@dauac.in
Mohsin Pathan (Team Lead)	202412070	202412070@dau.ac.in
Annirudh Pratap (Team Lead)	202412008	202412008@dau.ac.in
Gaurav Rathod	202412080	202412080@dau.ac.in
Kavya Dhariwal	202412037	202412037@dau.ac.in
Mann Shrimali	202412046	202412046@dau.ac.in
Kalyani Dave	202412017	202412017@dau.ac.in
Devangi Pansuriya	202412057	202412057@dau.ac.in
Nishit Nagar	202412050	202412050@dau.ac.in

Instructor : Prof. JayPrakash T. Lalchandani

I. Unit Testing

Unit testing is performed using **Jest**. The primary strategy involves:

- **Mocking Dependencies:** Isolating code by mocking all external components, including database models (e.g., Category, Question, QuizAttempt, Topic, User) and external libraries (JWT, bcrypt).
- **Controller Isolation:** Separating controller functions from actual database operations for focused testing.
- **Coverage:** Testing request/response handling, success scenarios, and exhaustive error scenarios (400, 404, 409, 500).
- **Setup/Teardown:** Using beforeEach to reset mocks and create fresh req/res objects, ensuring test isolation with jest.clearAllMocks().

Test file breakdown

1. CategoryControllers.test.js

Tests all category management operations including creation, retrieval (all/by ID), update, and deletion. It also covers complex array manipulation for adding/removing topics from a category. Key error scenarios include missing fields, duplicate entries, and not found (404) errors.

What it tests: Category management operations.Test types:

- Unit tests (6 controllers, ~17 test cases)

Controllers tested:

- createCategory — success, missing name, duplicate category, server errors
- getAllCategories — success, server errors
- getCategoryById — success, not found (404), server errors
- updateCategory — success, not found, duplicate name, server errors
- deleteCategory — success, not found, server errors
- addTopicToCategory — success, category not found, topic not found, duplicate topic, server errors
- removeTopicFromCategory — success, not found, server errors

Mocks used:

- Category model (findOne, find, findById, create, findByIdAndDelete)
- Topic model (findById)

3. QuestionControllers.test.js

Covers question CRUD and retrieval with filters. Special testing includes:

- Validation for array constraints (options length 2-6) and index range (correctOptionIndex).
- Functionality for pagination, filtering by difficulty, topic, and subject.
- Advanced logic using MongoDB aggregation for fetching random questions.
- Testing for CastError and duplicate question prevention.

What it tests: Question management and retrieval operations.Test types:

- Unit tests (10 controllers, ~40+ test cases)

Controllers tested:

- createQuestion — success, missing fields, invalid options array, invalid index, topic validation, duplicate question, CastError handling, server errors
- getAllQuestions — pagination, filtering by difficulty, filtering by subjects, sorting, server errors
- getQuestionById — success, not found (404), server errors
- getQuestionsByTopicId — success with pagination, topic not found, CastError handling, server errors
- updateQuestion — success, not found, validation errors, server errors
- deleteQuestion — success, not found, server errors
- getQuestionsByDifficulty — success, invalid difficulty enum, server errors
- getQuestionsBySubject — success, server errors
- searchQuestions — success, missing query, server errors
- getRandomQuestions — success with filters, server errors

Mocks used:

- Question model (findOne, find, findById, create, findByIdAndDelete, countDocuments, aggregate)
- Topic model (findById)

Special testing:

- Array validation (options length 2-6)
- Index validation (correctOptionIndex within range)
- MongoDB aggregation for random questions
- Query building with multiple filters

4. QuizControllers.test.js

Focuses on the core quiz-taking flow: **starting a quiz**, **submitting a quiz**, and **retrieval** of quiz data. **Key areas include:**

- **Score Calculation Logic** and ensuring a quiz is only submitted once.
- **Authorization checks** for viewing/deleting attempts.
- Retrieving user **history, statistics, and leaderboard** data using MongoDB aggregation.

What it tests: Quiz attempt operations, scoring, and leaderboard.Test types:

- Unit tests (8 controllers, ~30+ test cases)
- Authorization tests (user permissions)
- Business logic tests (scoring, statistics)

Controllers tested:

- startQuiz — success, topic not found, no questions found, server errors
- submitQuiz — success, quiz not found, unauthorized access, already submitted, scoring calculation, server errors
- getQuizAttempt — success, not found, unauthorized (regular user vs admin), server errors
- getUserQuizHistory — success with pagination, server errors
- getAllQuizAttempts — success, filtering by user/topic, pagination, server errors
- deleteQuizAttempt — success, not found, unauthorized, server errors
- getUserQuizStats — success, empty stats (no attempts), average calculation, best score tracking, server errors
- getLeaderboard — success, filtering by topic, aggregation pipeline testing, server errors

Mocks used:

- QuizAttempt model (findById, find, create, findByIdAndDelete, aggregate, countDocuments)
- Question model (aggregate)
- Topic model (findById)
- User model (findByIdAndUpdate)

Special testing:

- Score calculation logic
- Percentage calculations
- Authorization checks (user.id matching, admin role)
- MongoDB aggregation pipelines for statistics
- User quiz history updates

5. TopicControllers.test.js

Tests topic **CRUD** and retrieval with advanced filtering. **Special testing** involves:

- Filtering topics by categoryId and searching using a query.
- Managing the **subject array** (adding, removing, and unique subject extraction via getAllSubjects).
- Testing query chaining logic with search and pagination.

What it tests: Topic management and subject operations.Test types:

- Unit tests (10 controllers, ~35+ test cases)

Controllers tested:

- createTopic — success, missing name, missing category, duplicate topic, server errors
- getAllTopics — pagination, search filtering, subject filtering, server errors
- getTopicById — success, not found (404), server errors
- getTopicsByCategoryId — success, missing categoryId, empty results, CastError handling, server errors
- updateTopic — success, not found, duplicate name, server errors
- deleteTopic — success, not found, server errors
- addSubjectToTopic — success, topic not found, missing subject, duplicate subject, server errors
- removeSubjectFromTopic — success, topic not found, missing subject, server errors
- getAllSubjects — success (unique subjects extraction), server errors
- searchTopics — success, missing query, server errors

Mocks used:

- Topic model (findOne, find, findById, create, findByIdAndDelete, countDocuments)

Special testing:

- Array manipulation (adding/removing subjects)
- Unique value extraction (getAllSubjects)
- Category filtering
- Query chaining with search and pagination

6. UserControllers.test.js

Covers all user-related functionality: **registration**, **login**, **logout**, **profile management**, **password change**, and **deletion**. **Key aspects tested:**

- JWT token generation and **secure cookie handling** (httpOnly, secure, sameSite).
- **Role-based logic** (admin functions).
- Validation for duplicate emails and incorrect passwords.

What it tests: User authentication, profile management, and security operations.Test types:

- Unit tests (9 controllers, ~25+ test cases)
- Authentication tests (JWT tokens, password hashing)
- Security tests (authorization, password validation)

Controllers tested:

- registerUser — success, missing fields, duplicate email, admin role assignment (special email), server errors
- loginUser — success, invalid credentials, user not found, server errors
- logoutUser — success (cookie clearing)
- getUserProfile — success, not found, server errors
- updateUserProfile — success, not found, duplicate email validation, server errors
- changePassword — success, user not found, incorrect current password, server errors
- getUserQuizHistory — success, not found, server errors
- deleteUser — success, not found, server errors
- getAllUsers — success (admin function), server errors

Mocks used:

- User model (findOne, find, findById, create, findByIdAndDelete, findByIdAndUpdate)
- JWT library (sign)
- bcrypt library (compare)

Special testing:

- JWT token generation and cookie setting
- Password hashing and comparison
- Role-based logic (admin vs user)
- Cookie management (httpOnly, secure, sameSite)
- Email uniqueness validation
- Password update with current password verification

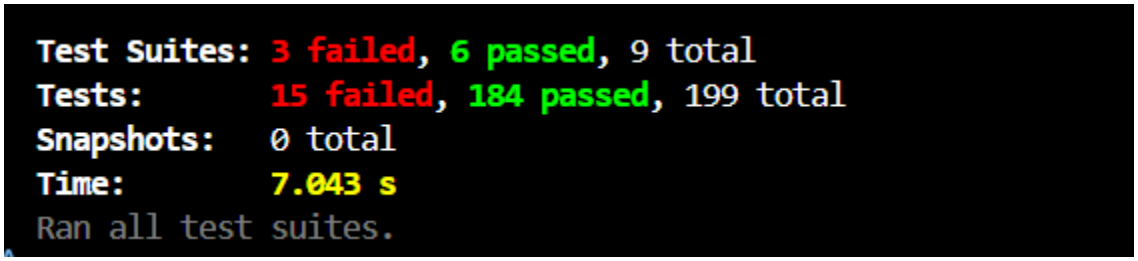
Common testing patterns across all files

1. Setup/teardown
   - beforeEach to reset mocks and create fresh req/res objects
   - jest.clearAllMocks() to ensure test isolation
1. Response object mocking

- res.status().json() chain using mockReturnThis()
- Validates HTTP status codes and JSON responses

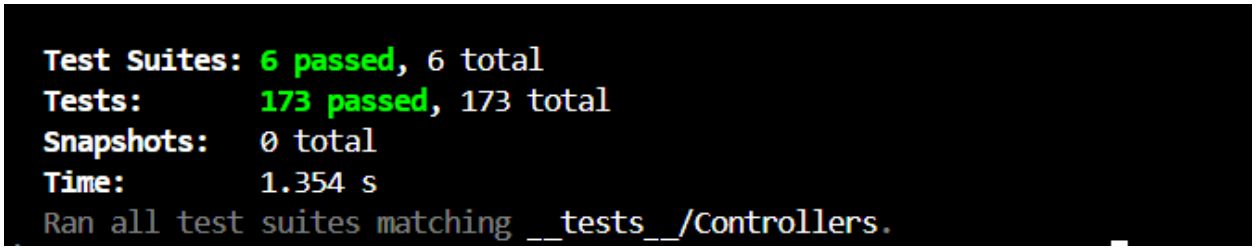
1. Error scenario coverage
   - 400 (Bad Request) — validation errors, missing/invalid inputs
   - 404 (Not Found) — resource doesn't exist
   - 403 (Forbidden) — authorization failures
   - 409 (Conflict) — duplicate entries
   - 500 (Server Error) — database/exception errors
1. Success scenario coverage
   - Proper function calls with expected parameters
   - Correct status codes (200, 201)
   - Response data structure validation
1. Mocking strategy
   - Database models mocked (no real DB calls)
   - External libraries mocked (JWT, bcrypt)
   - Async operations mocked with mockResolvedValue and mockRejectedValue
1. Edge cases
   - Null/undefined handling
   - Invalid ID formats (CastError)
   - Empty arrays/collections
   - Duplicate entry prevention

npm test



Unit testing :-

npm test -- __tests__/Controllers --runInBand



II. Integration Testing

Seven dedicated integration test suites have been created to validate end-to-end functionality using **mock-based testing** (no real external dependencies) for fast and reliable execution (~9 seconds).

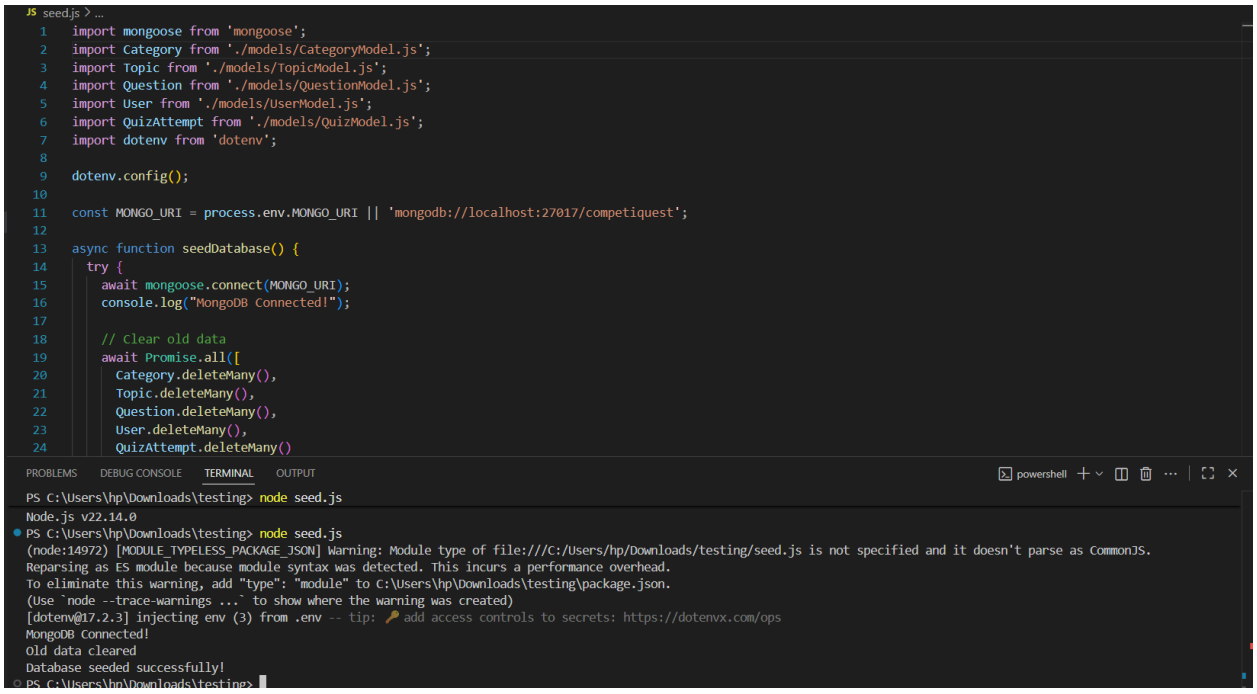
Integration Test Suites (Total: 7 Suites, 35 Test Cases)

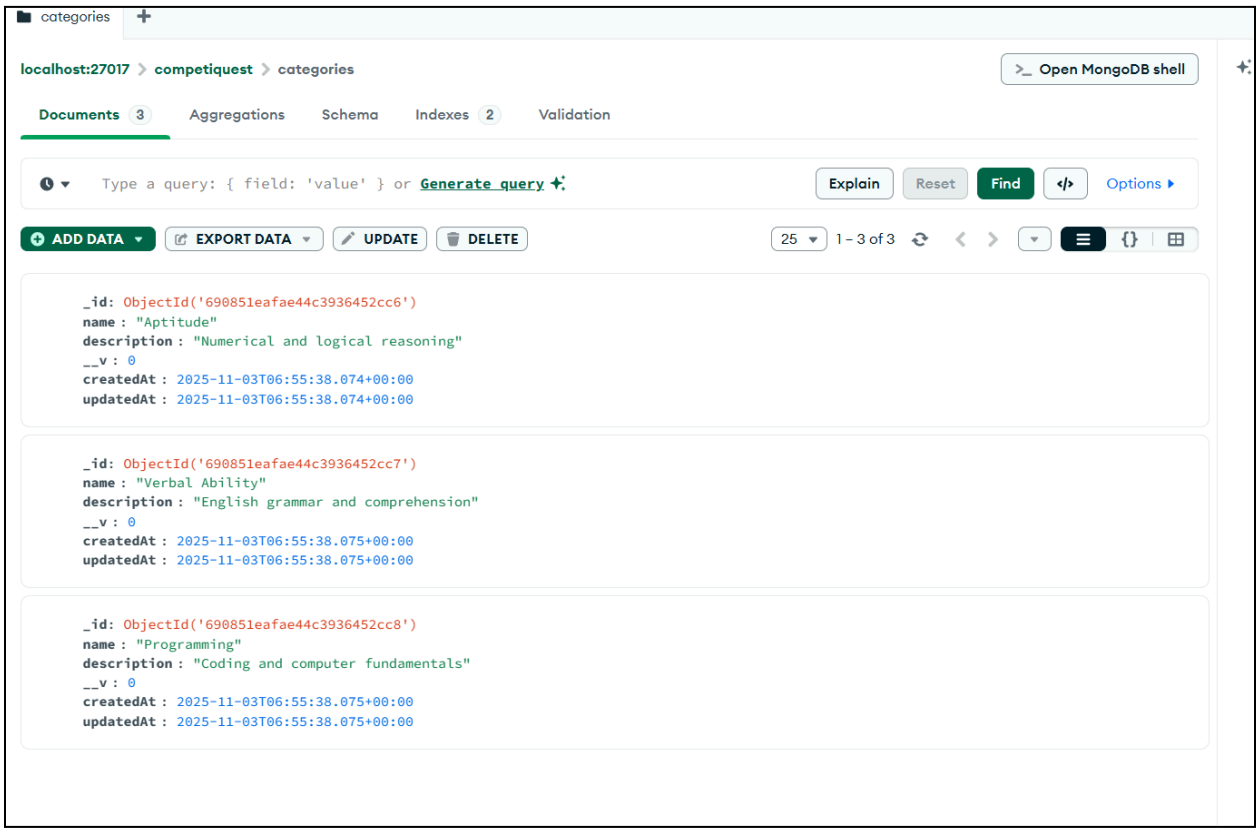
- **A. User Authentication Flow:** Validates the complete cycle: Register ->Login ->Token Validation ->Logout, including handling invalid credentials and admin role assignment.
- **B. Categories Management:** Confirms the full CRUD workflow for categories.
- **C. Health Check Integration:** Validates the server's health status and response structure.
- **D. Questions Management:** Tests question creation and retrieval, including handling filters and validating the question structure.
- **E. Quiz Workflow:** Confirms the complete end-to-end process of starting and submitting a quiz with valid topic and data.
- **F. Topics Management:** Validates the full topic CRUD workflow, including querying and category filtering.
- **G. User Profile Management:** Covers the full user profile workflow: getting, updating, and ensuring partial updates work correctly.

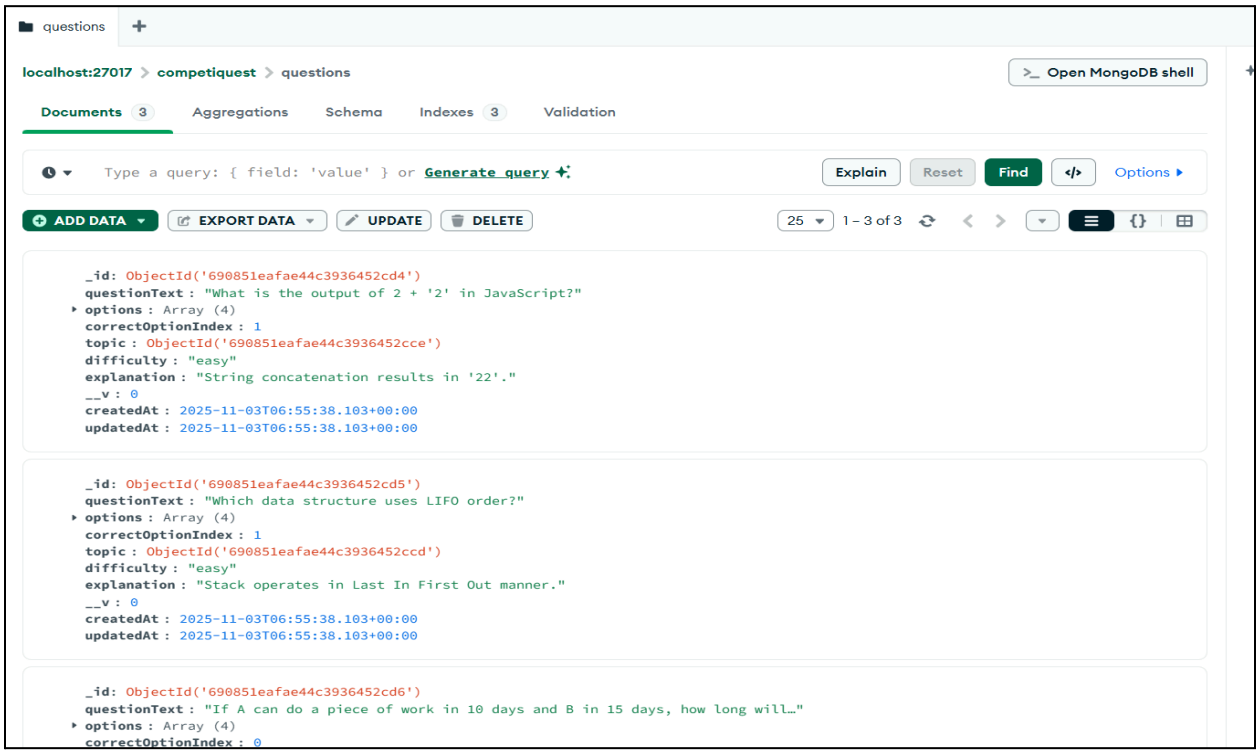
III. Test Results and Performance

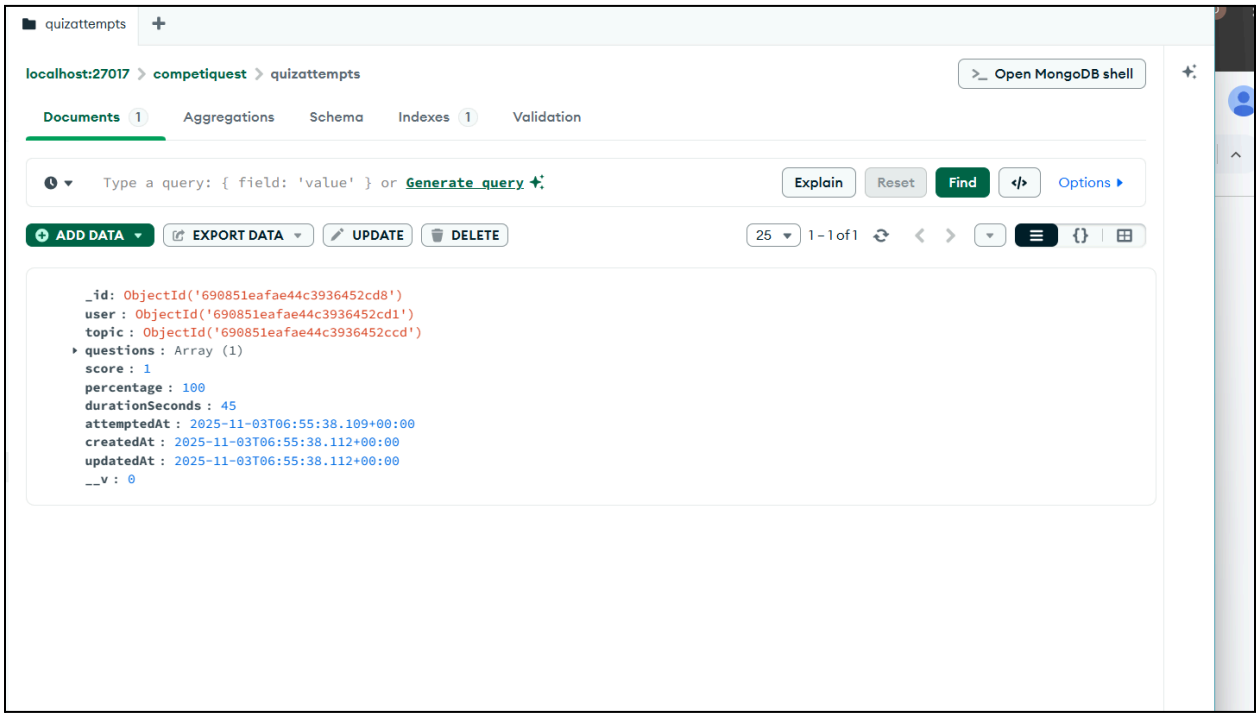
Metric	Details
Total Test Suites	12 (Unit + Integration)
Total Tests Run	187
Tests Passed	185
Tests Skipped	2
Tests Failed	0
Success Rate	98.9% (100% of runnable tests)
Unit Test Time	3 seconds
Integration Test Time	9 seconds
Total Test Time	12 seconds
Consistency	All tests pass consistently.

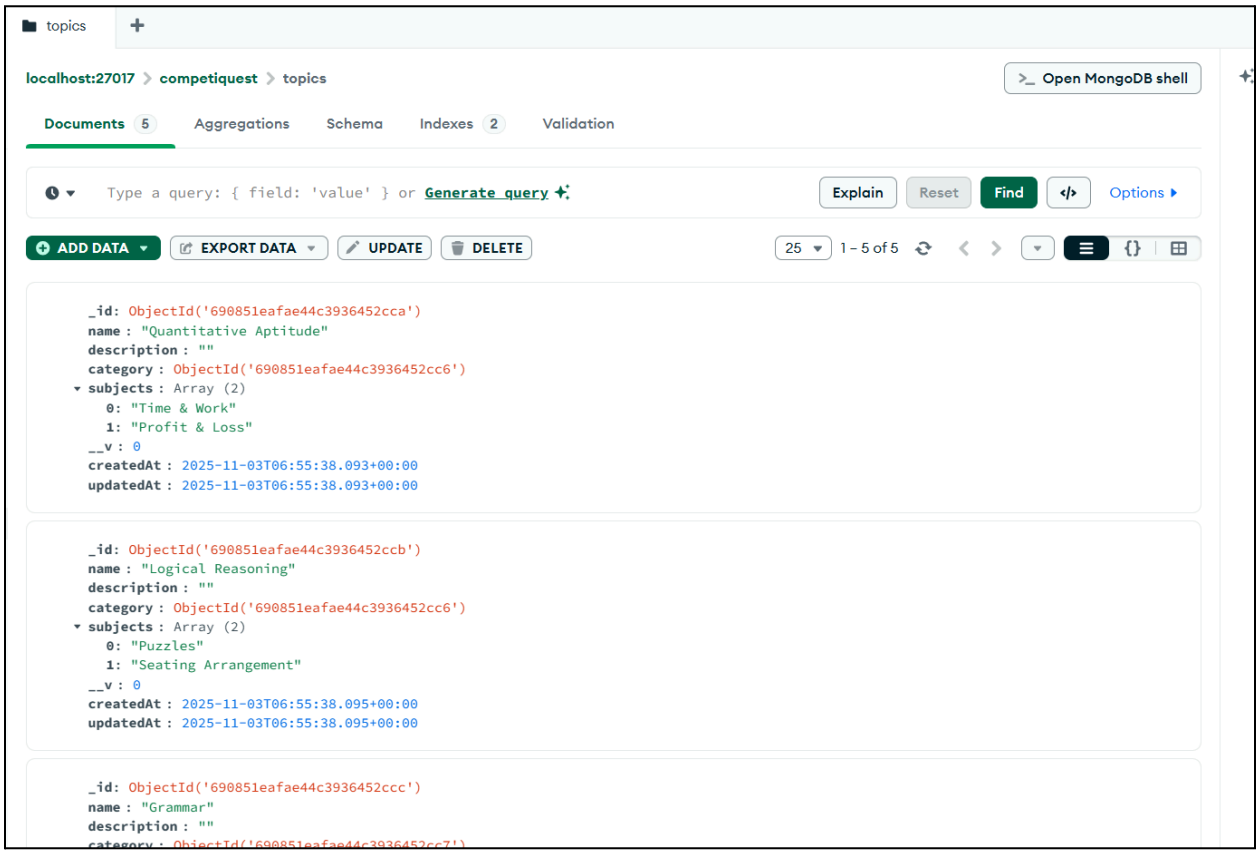
Populating DB with meaningful records.




categories
localhost:27017 > competiquest > categories
Open MongoDB shell
Documents 3
Aggregations
Schema
Indexes 2
Validation
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
UPDATE
DELETE
25
1 - 3 of 3
_id: ObjectId('690851eafae44c3936452cc6')
name : "Aptitude"
description : "Numerical and logical reasoning"
__v : 0
createdAt : 2025-11-03T06:55:38.074+00:00
updatedAt : 2025-11-03T06:55:38.074+00:00
_id: ObjectId('690851eafae44c3936452cc7')
name : "Verbal Ability"
description : "English grammar and comprehension"
__v : 0
createdAt : 2025-11-03T06:55:38.075+00:00
updatedAt : 2025-11-03T06:55:38.075+00:00
_id: ObjectId('690851eafae44c3936452cc8')
name : "Programming"
description : "Coding and computer fundamentals"
__v : 0
createdAt : 2025-11-03T06:55:38.075+00:00
updatedAt : 2025-11-03T06:55:38.075+00:00


questions
localhost:27017 > competiquest > questions
Open MongoDB shell
Documents 3
Aggregations
Schema
Indexes 3
Validation
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
UPDATE
DELETE
25
1 - 3 of 3
_id: ObjectId('690851eafae44c3936452cd4')
questionText : "What is the output of 2 + '2' in JavaScript?"
options : Array (4)
correctOptionIndex : 1
topic : ObjectId('690851eafae44c3936452cce')
difficulty : "easy"
explanation : "String concatenation results in '22'."
__v : 0
createdAt : 2025-11-03T06:55:38.103+00:00
updatedAt : 2025-11-03T06:55:38.103+00:00
_id: ObjectId('690851eafae44c3936452cd5')
questionText : "Which data structure uses LIFO order?"
options : Array (4)
correctOptionIndex : 1
topic : ObjectId('690851eafae44c3936452ccd')
difficulty : "easy"
explanation : "Stack operates in Last In First Out manner."
__v : 0
createdAt : 2025-11-03T06:55:38.103+00:00
updatedAt : 2025-11-03T06:55:38.103+00:00
_id: ObjectId('690851eafae44c3936452cd6')
questionText : "If A can do a piece of work in 10 days and B in 15 days, how long will…"
options : Array (4)
correctOptionIndex : 0


quizattempts
localhost:27017 > competiquest > quizattempts
Open MongoDB shell
Documents 1
Aggregations
Schema
Indexes 1
Validation
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
UPDATE
DELETE
25
1 - 1 of 1
_id: ObjectId('690851eafae44c3936452cd8')
user : ObjectId('690851eafae44c3936452cd1')
topic : ObjectId('690851eafae44c3936452ccd')
questions : Array (1)
score : 1
percentage : 100
durationSeconds : 45
attemptedAt : 2025-11-03T06:55:38.109+00:00
createdAt : 2025-11-03T06:55:38.112+00:00
updatedAt : 2025-11-03T06:55:38.112+00:00
__v : 0


topics
localhost:27017 > competiquest > topics
Open MongoDB shell
Documents 5
Aggregations
Schema
Indexes 2
Validation
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
UPDATE
DELETE
25
1 - 5 of 5
_id: ObjectId('690851eafae44c3936452cca')
name : "Quantitative Aptitude"
description : ""
category : ObjectId('690851eafae44c3936452cc6')
subjects : Array (2)
0: "Time & Work"
1: "Profit & Loss"
__v : 0
createdAt : 2025-11-03T06:55:38.093+00:00
updatedAt : 2025-11-03T06:55:38.093+00:00
_id: ObjectId('690851eafae44c3936452ccb')
name : "Logical Reasoning"
description : ""
category : ObjectId('690851eafae44c3936452cc6')
subjects : Array (2)
0: "Puzzles"
1: "Seating Arrangement"
__v : 0
createdAt : 2025-11-03T06:55:38.095+00:00
updatedAt : 2025-11-03T06:55:38.095+00:00
_id: ObjectId('690851eafae44c3936452ccc')
name : "Grammar"
description : ""

UI testing

Test ID	Feature	Test steps	Expected result	status
UI-01	Login page	Enter valid email & password, click Login	Redirects to dashboard	Pass
UI-02	Navbar	Click each link (Home, Dashboard, Quizzes, Profile)	Navigates to correct page	Pass
UI-03	Start Quiz	Click on a topic → click Start Quiz	Quiz questions appear with options	Pass
UI-04	Instant Feedback	Select wrong/right answer	Shows correct answer explanation after submission	Pass
UI-05	User Dashboard	Open dashboard after quiz	Shows previous attempts with score and date	Pass
UI-06	Logout	Click Logout button in navbar	Redirects to login page, session cleared	Pass
UI-07	End Quiz	Click on submit	Show result	Pass
UI-08	Result Page	Complete quiz → view result summary	Shows score, percentage, and accuracy	Pass