

Raport Javor – Test Driven Development (TDD)

Informata të përgjithshme

Emri: Getoar Krasniqi

Java: 3 (e tretë)

Data e dorëzimit: 23/11/2025

Emri i projektit: Quiz Application

1 Përbledhje e javës

Në javën e tretë është implementuar feature 2, në kuadër të së cilës janë përfshirë shtimi i:

- Timer,
- Difficulty levels,
- Enhanced answer review, dhe
- Category for questions.

Në README.md file janë shtuar ndihmesat se si të përdoren të gjitha pjesët e feature 2. Janë përditësuar statuset në fillim të README në përputhje me progresin e bërë në kod.

Gjenerimi i index.html me përqindje të mbulueshmërisë 100%.

Krijimi i “v3.0 Feature 2 Red Phase”: 12 testet e feature 2 të shkruara, të cilat kanë dështuar në mungesë të kodit të implementuar.

Krijimi i “v3.1 Feature 2 Green Phase”: Implementimi i kodit në mënyrë që testet e feature 2 të kalojnë me sukses.

Krijimi i “v3.2 Refactoring”: Refaktorimi i kodit.

Krijimi i branch-ave për secilën javë deri tani duke u bazuar në tag-at e krijuar paraprakisht.

2 Testet e zhvilluara

Nr	Përshkrimi i testit	Statusi (kalon/dështon)	Framework /
1	test_create_multiple_choice_question	kalon	pytest
2	test_check_correct_answer	kalon	pytest
3	test_question_requires_text	kalon	pytest
4	test_create_empty_quiz	kalon	pytest

5	test_add_question_to_quiz	kalon	pytest
6	test_cannot_add_duplicate_questions	kalon	pytest
7	test_get_question_by_index	kalon	pytest
8	test_submit_answer_and_track_score	kalon	pytest
9	test_perfect_score	kalon	pytest
10	test_passing_grade	kalon	pytest
11	test_result_summary	kalon	pytest
12	test_complete_quiz_flow	kalon	pytest
13	test_question_has_category	dështon -> (v3.0) kalon -> (v3.1)	pytest
14	test_quiz_can_get_questions_by_category	dështon -> (v3.0) kalon -> (v3.1)	pytest
15	test_quiz_get_score_by_category	dështon -> (v3.0) kalon -> (v3.1)	pytest
16	test_question_has_difficulty_level	dështon -> (v3.0) kalon -> (v3.1)	pytest
17	test_question_default_difficulty_is_medium	dështon -> (v3.0) kalon -> (v3.1)	pytest
18	test_filter_questions_by_difficulty	dështon -> (v3.0) kalon -> (v3.1)	pytest
19	test_get_incorrect_answers	dështon -> (v3.0) kalon -> (v3.1)	pytest
20	test_get_answer_details	dështon -> (v3.0) kalon -> (v3.1)	pytest
21	test_quiz_has_time_limit	dështon -> (v3.0) kalon -> (v3.1)	pytest
22	test_quiz_starts_timer_on_first_answer	dështon -> (v3.0) kalon -> (v3.1)	pytest
23	test_quiz_tracks_elapsed_time	dështon -> (v3.0) kalon -> (v3.1)	pytest

24	test_quiz_can_check_if_time_expired	dështon -> (v3.0) kalon -> (v3.1)	pytest
----	-------------------------------------	--------------------------------------	--------

3 Implementimi i bërë pas testeve

Implemetimet e bëra pas testeve të feature 2:

Moduli question.py

```
from typing import List, Optional

class Question:
    """Represents a single quiz question with multiple choice options"""

    def __init__(
        self,
        text: str,
        options: List[str],
        correct_answer: str,
        difficulty: str = "medium",
        category: Optional[str] = None,
    ) -> None:
        if not text or text.strip() == "":
            raise ValueError("Question text cannot be empty")

        self.text = text
        self.options = options
        self.correct_answer = correct_answer
        self.difficulty = difficulty
        self.category = category

    def check_answer(self, answer: str) -> bool:
        """Check if the provided answer is correct"""
        return answer == self.correct_answer

    def __eq__(self, other: object) -> bool:
        """Check equality based on question text and options"""
        if not isinstance(other, Question):
            return False
        return (
            self.text == other.text
            and self.options == other.options
            and self.correct_answer == other.correct_answer
        )

```

```

def __hash__(self) -> int:
    """Make Question hashable for use in sets"""
    return hash((self.text, tuple(self.options), self.correct_answer))

```

Moduli quiz.py

```

from typing import Dict, List, Optional
from src.question import Question
from src.result import QuizResult
import time


class Quiz:
    """Represents a quiz with multiple questions"""

    def __init__(self, title: str, time_limit_seconds: Optional[int] = None) -> None:
        self.title = title
        self.questions: List[Question] = []
        self.answers: Dict[int, str] = {} # Maps question index to submitted answer
        self.time_limit_seconds = time_limit_seconds
        self.start_time: Optional[float] = None

    def add_question(self, question: Question) -> None:
        """Add a question to the quiz, avoiding duplicates"""
        if question not in self.questions:
            self.questions.append(question)

    def get_question(self, index: int) -> Question:
        """Get a question by its index"""
        return self.questions[index]

    def submit_answer(self, question_index: int, answer: str) -> None:
        """Submit an answer for a specific question"""
        # Start timer on first answer submission
        if self.start_time is None:
            self.start_time = time.time()

        self.answers[question_index] = answer

    def get_elapsed_time(self) -> float:
        """Get the elapsed time since the quiz started"""
        if self.start_time is None:
            return 0
        return time.time() - self.start_time

```

```
def is_time_expired(self) -> bool:
    """Check if the time limit has been exceeded"""
    if self.time_limit_seconds is None:
        return False
    return self.get_elapsed_time() > self.time_limit_seconds

def get_result(self) -> QuizResult:
    """Calculate and return the quiz result"""
    score = 0
    total = len(self.questions)

    for index, answer in self.answers.items():
        if index < len(self.questions):
            question = self.questions[index]
            if question.check_answer(answer):
                score += 1

    return QuizResult(score, total)

def get_questions_by_difficulty(self, difficulty: str) -> List[Question]:
    """Get all questions with a specific difficulty level"""
    return [q for q in self.questions if q.difficulty == difficulty]

def get_questions_by_category(self, category: Optional[str]) -> List[Question]:
    """Get all questions in a specific category"""
    return [q for q in self.questions if q.category == category]

def get_incorrect_answers(self) -> List[int]:
    """Get a list of indices for incorrectly answered questions"""
    incorrect = []
    for index, answer in self.answers.items():
        if index < len(self.questions):
            question = self.questions[index]
            if not question.check_answer(answer):
                incorrect.append(index)
    return incorrect

def get_answer_details(self, question_index: int) -> Dict:
    """Get detailed information about a specific answer"""
    question = self.questions[question_index]
    submitted_answer = self.answers.get(question_index)

    return {
        "question": question,
        "submitted_answer": submitted_answer,
```

```

        "correct_answer": question.correct_answer,
        "is_correct": question.check_answer(submitted_answer) if submitted_answer
else False,
    }

def get_score_by_category(self, category: Optional[str]) -> Dict[str, float]:
    """Get the score for questions in a specific category"""
    category_questions = []
    score = 0

    for index, question in enumerate(self.questions):
        if question.category == category:
            category_questions.append(index)
            if index in self.answers and
question.check_answer(self.answers[index]):
                score += 1

    total = len(category_questions)
    percentage = (score / total * 100) if total > 0 else 0.0

    return {"score": score, "total": total, "percentage": percentage}

```

Moduli result.py

```

class QuizResult:
    """Represents the result of a completed quiz"""

    def __init__(self, score: int, total: int) -> None:
        self.score = score
        self.total = total
        self.percentage = (score / total * 100) if total > 0 else 0.0

    def is_perfect(self) -> bool:
        """Check if the score is perfect (100%)"""
        return self.score == self.total

    def is_passing(self, threshold: int = 60) -> bool:
        """Check if the score meets the passing threshold"""
        return self.percentage >= threshold

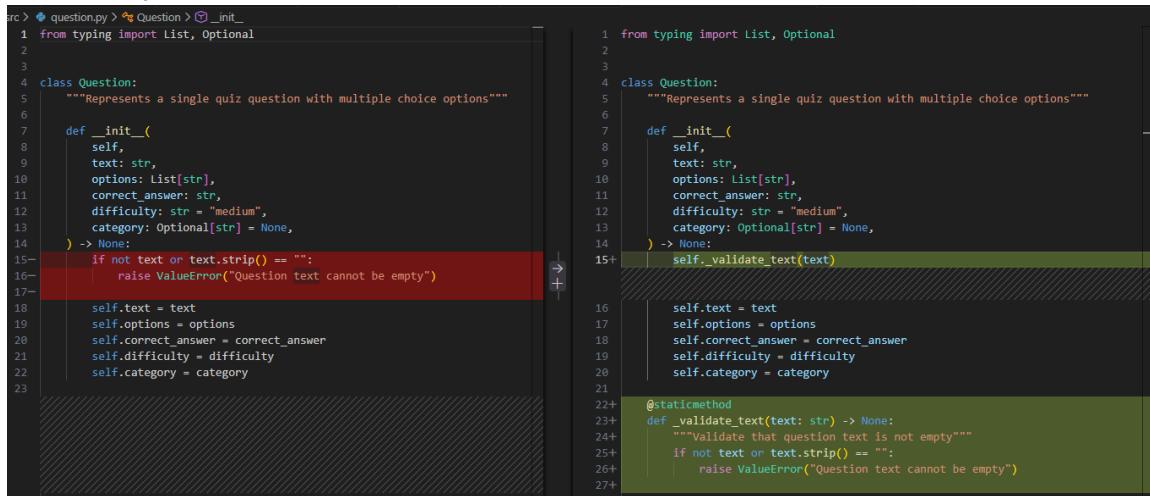
    def get_summary(self) -> str:
        """Get a formatted summary of the results"""
        return f"Score: {self.score}/{self.total} ({self.percentage:.1f}%)"

```

Refaktorimi i bërë

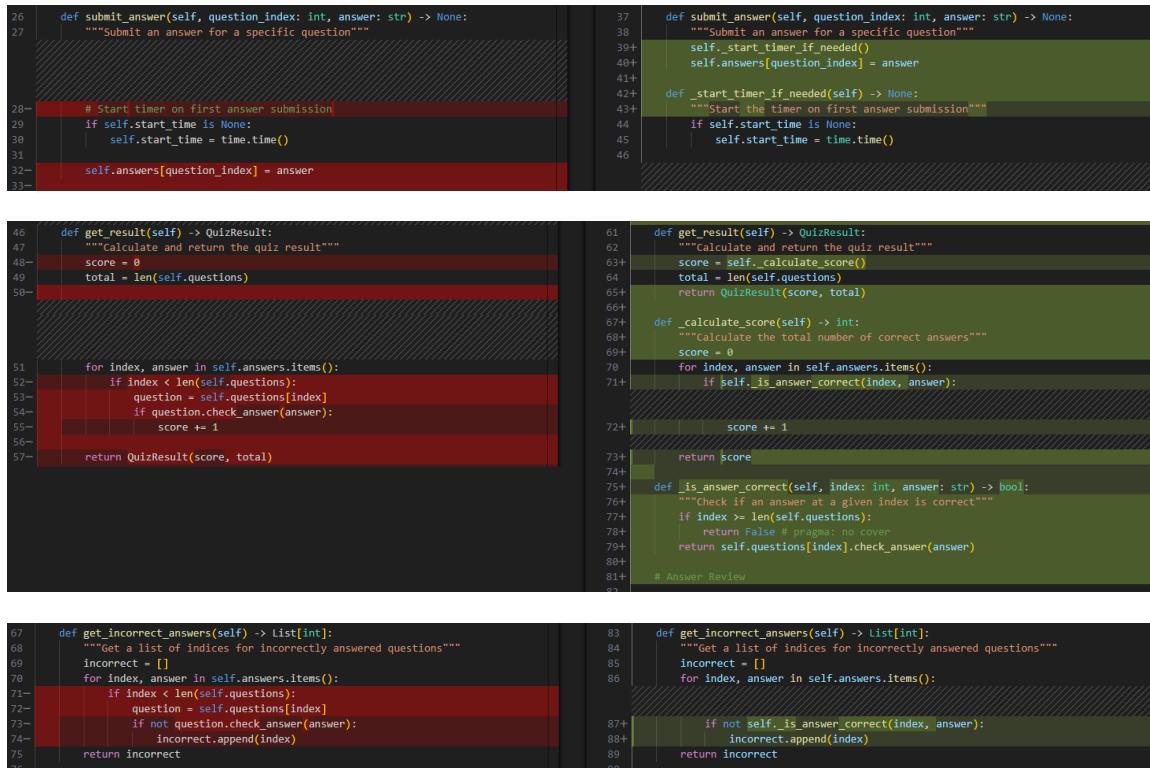
Refaktorimet e bëra në secilin modul (ana e majtë është kodi i vjetër, ana e djathtë është refaktorimi i bërë):

question.py



```
src > question.py > Question > __init__  
1 from typing import List, Optional  
2  
3  
4 class Question:  
5     """Represents a single quiz question with multiple choice options"""  
6  
7     def __init__(  
8         self,  
9         text: str,  
10        options: List[str],  
11        correct_answer: str,  
12        difficulty: str = "medium",  
13        category: Optional[str] = None,  
14    ) -> None:  
15        if not text or text.strip() == "":  
16            raise ValueError("Question text cannot be empty")  
17  
18        self.text = text  
19        self.options = options  
20        self.correct_answer = correct_answer  
21        self.difficulty = difficulty  
22        self.category = category  
23  
1 from typing import List, Optional  
2  
3  
4 class Question:  
5     """Represents a single quiz question with multiple choice options"""  
6  
7     def __init__(  
8         self,  
9         text: str,  
10        options: List[str],  
11        correct_answer: str,  
12        difficulty: str = "medium",  
13        category: Optional[str] = None,  
14    ) -> None:  
15+         self._validate_text(text)  
16  
17        self.text = text  
18        self.options = options  
19        self.correct_answer = correct_answer  
20        self.difficulty = difficulty  
21        self.category = category  
22  
22+     @staticmethod  
23+     def _validate_text(text: str) -> None:  
24+         """Validate that question text is not empty"""  
25+         if not text or text.strip() == "":  
26+             raise ValueError("Question text cannot be empty")  
27
```

quiz.py



```
26     def submit_answer(self, question_index: int, answer: str) -> None:  
27         """Submit an answer for a specific question"""  
28  
28-         # Start timer on first answer submission  
29         if self.start_time is None:  
30             self.start_time = time.time()  
31  
32         self.answers[question_index] = answer  
33  
37     def submit_answer(self, question_index: int, answer: str) -> None:  
38         """Submit an answer for a specific question"""  
39+         self._start_timer_if_needed()  
40+         self.answers[question_index] = answer  
41+  
42+     def _start_timer_if_needed(self) -> None:  
43+         """Start the timer on first answer submission"""  
44         if self.start_time is None:  
45             self.start_time = time.time()  
46  
46  
51     def get_result(self) -> QuizResult:  
52         """Calculate and return the quiz result"""  
53         score = 0  
54         total = len(self.questions)  
55  
56         for index, answer in self.answers.items():  
57             if index < len(self.questions):  
58                 question = self.questions[index]  
59                 if question.check_answer(answer):  
60                     score += 1  
61  
61     def get_result(self) -> QuizResult:  
62         """Calculate and return the quiz result"""  
63         score = self._calculate_score()  
64         total = len(self.questions)  
65         return QuizResult(score, total)  
66  
67+     def _calculate_score(self) -> int:  
68+         """Calculate the total number of correct answers"""  
69+         score = 0  
70         for index, answer in self.answers.items():  
71             if self.is_answer_correct(index, answer):  
72                 score += 1  
73  
73+         return score  
74  
75+     def is_answer_correct(self, index: int, answer: str) -> bool:  
76+         """Check if an answer at a given index is correct"""  
77+         if index >= len(self.questions):  
78+             return False # pragma: no cover  
79+         return self.questions[index].check_answer(answer)  
80+  
81+     # Answer Review  
82  
82  
87+     def get_incorrect_answers(self) -> List[int]:  
88+         """Get a list of indices for incorrectly answered questions"""  
89+         incorrect = []  
90+         for index, answer in self.answers.items():  
91+             if not self.is_answer_correct(index, answer):  
92+                 incorrect.append(index)  
93  
93
```

```
def get_score_by_category(self, category: Optional[str] -> Dict[str, float]:
    """Get the score for questions in a specific category"""
    category_questions = []
    score = 0

    for index, question in enumerate(self.questions):
        if question.category == category:
            category_questions.append(index)
            if index in self.answers and question.check_answer(self.answers[index]):
                score += 1

    total = len(category_questions)
    percentage = (score / total * 100) if total > 0 else 0.0

    return {"score": score, "total": total, "percentage": percentage}

def _get_category_question_indices(self, category: str) -> List[int]:
    """Get indices of all questions in a category"""
    return [
        index for index, question in enumerate(self.questions)
        if question.category == category
    ]

def _calculate_category_score(self, indices: List[int]) -> int:
    """Calculate score for a list of question indices"""
    score = 0
    for index in indices:
        if index in self.answers and self._is_answer_correct(index, self.answers[index]):
            score += 1
    return score
```

result.py

```
4 def __init__(self, score: int, total: int) -> None:
5     self.score = score
6     self.total = total
7     self.percentage = (score / total * 100) if total > 0 else 0.0
```

→

```
4 def __init__(self, score: int, total: int) -> None:
5     self.score = score
6     self.total = total
7     self.percentage = self._calculate_percentage(score, total)

8+ @staticmethod
9+ def _calculate_percentage(score: int, total: int) -> float:
10+     """Calculate percentage score"""
11+     return (score / total * 100) if total > 0 else 0.0
```

5 Probleme të hasura

Gjatë shkrimit të testeve dhe implementimit të kodit për feature 2 nuk kam hasur në ndonjë problem.

6 Plani për javën e ardhshme

Gjatë javës tjetër planifikohet të bëhet lidhja e aplikacionit me databazën dhe vurja në funksion e API-ve.

7 Mbulesa e testimit (nëse aplikohet)

- % e mbulesës aktuale: 100%
 - Vegla e përdorur: pytest-cov

8 Komente / reflektime të grupit

Për këtë javë ka qenë në planifikim të bëhet edhe lidhja e aplikacionit me databazën dhe vurja në funksion e API-ve. Për shkak të ndarjes së punës në mënyrë më të barabartë ndërmjet javëve është vendosur që ky implementim të bëhet javën tjetër (Java 4).

Ngjitjet (opsionale)

- Screenshots nga testet që kalojnë / dështojnë

v3.0 Feature 2 Red Phase (12 testet e krijuara dështojnë):

```
PS C:\Users\DELL\Documents\GitHub\quiz-app> pytest
===== test session starts =====
platform win32 -- Python 3.13.7, pytest-9.0.1, pluggy-1.6.0 -- C:\Users\DELL\AppData\Local\Programs\Python\Python313\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\DELL\Documents\GitHub\quiz-app
configfile: pyproject.toml
testpaths: tests
plugins: anyio-4.11.0, cov-7.0.0
collected 24 items

tests/test_integration.py::TestQuizIntegration::test_complete_quiz_flow PASSED
[ 4%]
tests/test_question.py::TestQuestion::test_create_multiple_choice_question PASSED
[ 8%]
tests/test_question.py::TestQuestion::test_check_correct_answer PASSED
[ 12%]
tests/test_question.py::TestQuestion::test_question_requires_text PASSED
[ 16%]
tests/test_quiz.py::TestQuiz::test_create_empty_quiz PASSED
[ 20%]
tests/test_quiz.py::TestQuiz::test_add_question_to_quiz PASSED
[ 25%]
tests/test_quiz.py::TestQuiz::test_cannot_add_duplicate_questions PASSED
[ 29%]
tests/test_quiz.py::TestQuiz::test_get_question_by_index PASSED
[ 33%]
tests/test_quiz.py::TestQuiz::test_submit_answer_and_track_score PASSED
[ 37%]
tests/test_quiz_categories.py::TestQuizCategories::test_question_has_category FAILED
[ 41%]
tests/test_quiz_categories.py::TestQuizCategories::test_quiz_can_get_questions_by_category FAILED
[ 45%]
tests/test_quiz_categories.py::TestQuizCategories::test_quiz_get_score_by_category FAILED
[ 50%]
tests/test_quiz_difficulty.py::TestQuestionDifficulty::test_question_has_difficulty_level FAILED
[ 54%]
tests/test_quiz_difficulty.py::TestQuestionDifficulty::test_question_default_difficulty_is_medium FAILED
[ 58%]
tests/test_quiz_difficulty.py::TestQuestionDifficulty::test_filter_questions_by_difficulty FAILED
[ 62%]
tests/test_quiz_review.py::TestQuizReview::test_get_incorrect_answers FAILED
[ 66%]
tests/test_quiz_review.py::TestQuizReview::test_get_answer_details FAILED
[ 70%]
tests/test_quiz_timer.py::TestQuizTimer::test_quiz_has_time_limit FAILED
[ 75%]
tests/test_quiz_timer.py::TestQuizTimer::test_quiz_starts_timer_on_first_answer FAILED
[ 79%]
tests/test_quiz_timer.py::TestQuizTimer::test_quiz_tracks_elapsed_time FAILED
[ 83%]
tests/test_quiz_timer.py::TestQuizTimer::test_quiz_can_check_if_time_expired PASSED
[ 87%]
tests/test_result.py::TestQuizResult::test_perfect_score PASSED
[ 91%]
tests/test_result.py::TestQuizResult::test_passing_grade PASSED
[ 95%]
tests/test_result.py::TestQuizResult::test_result_summary PASSED
[ 100%]
:Users\DELL\AppData\Local\Programs\Python\Python313\Lib\site-packages\coverage\inorout.py:521: CoverageWarning: Module quiz was never imported. (module-not-imported); see https://coverage.readthedocs.io/en/7.11.3/messages.html#warning-module-not-imported
self.warn(f"Module {pkg} was never imported.", slug="module-not-imported")
C:\Users\DELL\AppData\Local\Programs\Python\Python313\Lib\site-packages\coverage\control.py:950: CoverageWarning: No data was collected. (no-data-collected); see https://coverage.readthedocs.io/en/7.11.3/messages.html#warning-no-data-collected
self._warn("No data was collected.", slug="no-data-collected")
```

v3.1 Feature 2 Green Phase (të gjitha kalojnë):

```
PS C:\Users\DELL\Documents\GitHub\quiz-app> pytest
===== test session starts =====
platform win32 -- Python 3.13.7, pytest-9.0.1, pluggy-1.6.0 -- C:\Users\DELL\AppData\Local\Programs\Python\Python313\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\DELL\Documents\GitHub\quiz-app
configfile: pyproject.toml
testpaths: tests
plugins: anyio-4.11.0, cov-7.0.0
collected 24 items

tests/test_integration.py::TestQuizIntegration::test_complete_quiz_flow PASSED
[ 4%]
tests/test_question.py::TestQuestion::test_create_multiple_choice_question PASSED
[ 8%]
tests/test_question.py::TestQuestion::test_check_correct_answer PASSED
[ 12%]
tests/test_question.py::TestQuestion::test_question_requires_text PASSED
[ 16%]
tests/test_quiz.py::TestQuiz::test_create_empty_quiz PASSED
[ 20%]
tests/test_quiz.py::TestQuiz::test_add_question_to_quiz PASSED
[ 25%]
tests/test_quiz.py::TestQuiz::test_cannot_add_duplicate_questions PASSED
[ 29%]
tests/test_quiz.py::TestQuiz::test_get_question_by_index PASSED
[ 33%]
tests/test_quiz.py::TestQuiz::test_submit_answer_and_track_score PASSED
[ 37%]
tests/test_quiz_categories.py::TestQuizCategories::test_question_has_category PASSED
[ 41%]
tests/test_quiz_categories.py::TestQuizCategories::test_quiz_can_get_questions_by_category PASSED
[ 45%]
tests/test_quiz_categories.py::TestQuizCategories::test_quiz_get_score_by_category PASSED
[ 50%]
tests/test_quiz_difficulty.py::TestQuestionDifficulty::test_question_has_difficulty_level PASSED
[ 54%]
tests/test_quiz_difficulty.py::TestQuestionDifficulty::test_question_default_difficulty_is_medium PASSED
[ 58%]
tests/test_quiz_difficulty.py::TestQuestionDifficulty::test_filter_questions_by_difficulty PASSED
[ 62%]
tests/test_quiz_review.py::TestQuizReview::test_get_incorrect_answers PASSED
[ 66%]
tests/test_quiz_review.py::TestQuizReview::test_get_answer_details PASSED
[ 70%]
tests/test_quiz_timer.py::TestQuizTimer::test_quiz_has_time_limit PASSED
[ 75%]
tests/test_quiz_timer.py::TestQuizTimer::test_quiz_starts_timer_on_first_answer PASSED
[ 79%]
tests/test_quiz_timer.py::TestQuizTimer::test_quiz_tracks_elapsed_time PASSED
[ 83%]
tests/test_quiz_timer.py::TestQuizTimer::test_quiz_can_check_if_time_expired PASSED
[ 87%]
tests/test_result.py::TestQuizResult::test_perfect_score PASSED
[ 91%]
tests/test_result.py::TestQuizResult::test_passing_grade PASSED
[ 95%]
tests/test_result.py::TestQuizResult::test_result_summary PASSED
[ 100%]
:Users\DELL\AppData\Local\Programs\Python\Python313\Lib\site-packages\coverage\inorout.py:521: CoverageWarning: Module quiz was never imported. (module-not-imported); see https://coverage.readthedocs.io/en/7.11.3/messages.html#warning-module-not-imported
self.warn(f"Module {pkg} was never imported.", slug="module-not-imported")
C:\Users\DELL\AppData\Local\Programs\Python\Python313\Lib\site-packages\coverage\control.py:950: CoverageWarning: No data was collected. (no-data-collected); see https://coverage.readthedocs.io/en/7.11.3/messages.html#warning-no-data-collected
self._warn("No data was collected.", slug="no-data-collected")
```

- Link i GitHub-it <https://github.com/Geti23/quiz-app>

- Raporti i CI/CD ose code coverage (në faqen tjetër):

v3.0 Feature 2 Red Phase

← Tests

✖ Feature 2 tests (red phase of feature 2) #4

Summary

Triggered via push 3 hours ago	Status	Total duration	Artifacts
Geti23 pushed -o_22a8085_main	Failure	25s	-

Jobs

- ⌚ test (ubuntu-latest, 3.9)
- ⌚ test (ubuntu-latest, 3.10)
- ⌚ test (ubuntu-latest, 3.11)
- ⌚ test (ubuntu-latest, 3.12)
- ⌚ test (windows-latest, 3.9)
- ⌚ test (windows-latest, 3.10)
- ⌚ test (windows-latest, 3.11)
- ⌚ test (windows-latest, 3.12)
- ⌚ test (macos-latest, 3.9)
- ⌚ test (macos-latest, 3.10)
- ✖ test (macos-latest, 3.11)
- ⌚ test (macos-latest, 3.12)
- ✖ lint

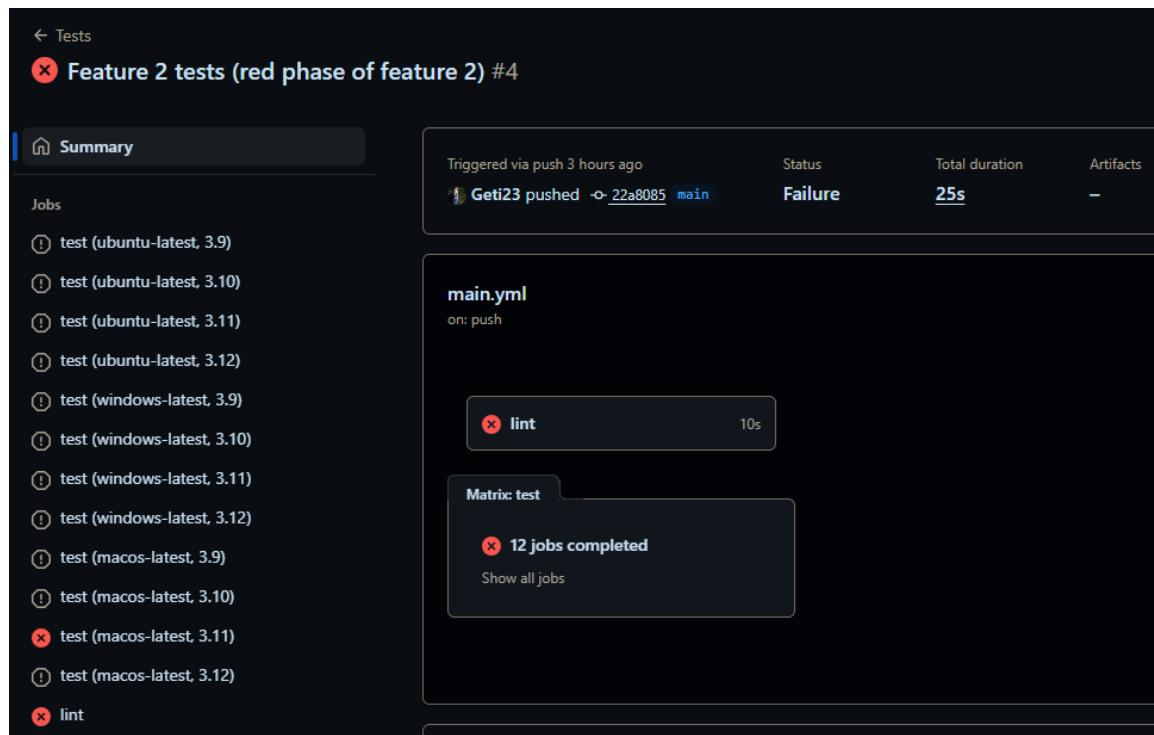
main.yml
on: push

✖ lint 10s

Matrix: test

✖ 12 jobs completed

Show all jobs



v3.1 Feature 2 Green Phase

← Tests

✓ Feature 2 implementation (green phase of feature 2) #8

Summary

Triggered via push 1 hour ago	Status	Total duration	Artifacts
Geti23 pushed -o_3e7844a_main	Success	53s	-

Jobs

- ⌚ test (ubuntu-latest, 3.9)
- ⌚ test (ubuntu-latest, 3.10)
- ⌚ test (ubuntu-latest, 3.11)
- ⌚ test (ubuntu-latest, 3.12)
- ⌚ test (windows-latest, 3.9)
- ⌚ test (windows-latest, 3.10)
- ⌚ test (windows-latest, 3.11)
- ⌚ test (windows-latest, 3.12)
- ⌚ test (macos-latest, 3.9)
- ⌚ test (macos-latest, 3.10)
- ⌚ test (macos-latest, 3.11)
- ⌚ test (macos-latest, 3.12)
- ⌚ lint

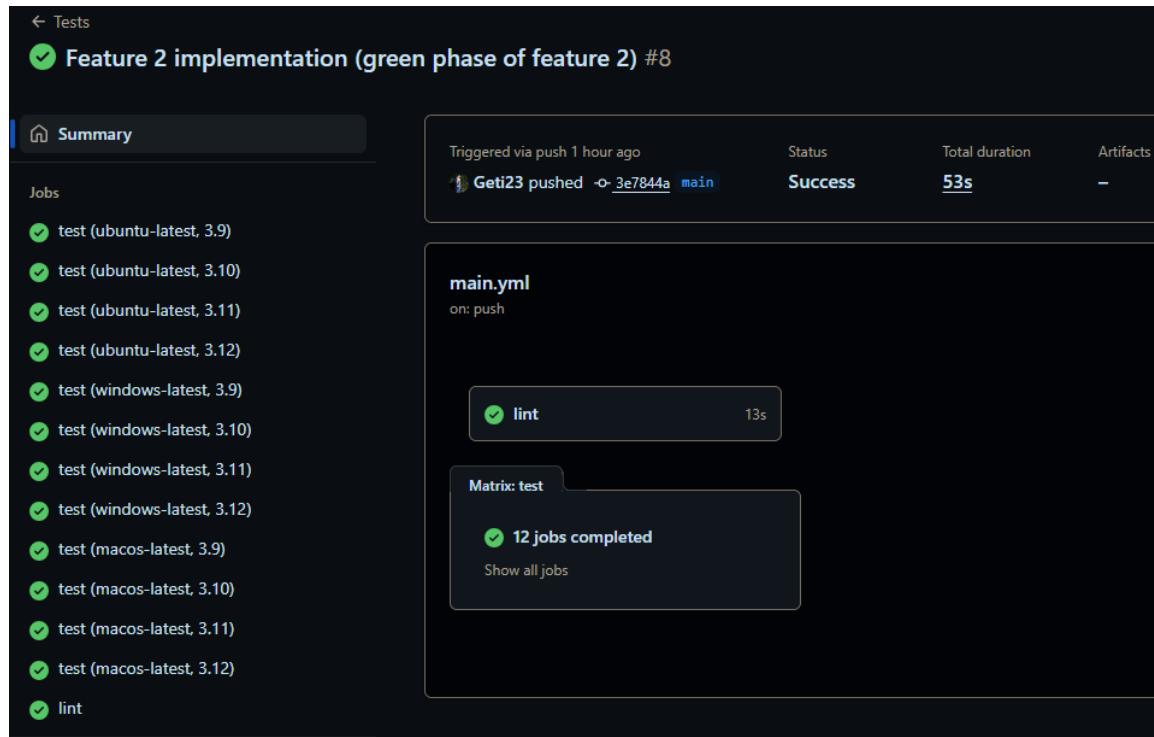
main.yml
on: push

✓ lint 13s

Matrix: test

✓ 12 jobs completed

Show all jobs



Coverage report: 100%

Files Functions Classes

coverage.py v7.11.3, created at 2025-11-23 23:22 +0100

File ▲	function	statements	missing	excluded	coverage
src__init__.py	(no function)	4	0	0	100%
src\question.py	Question.__init__	6	0	0	100%
src\question.py	Question._validate_text	2	0	0	100%
src\question.py	Question.check_answer	1	0	0	100%
src\question.py	Question.__eq__	2	0	1	100%
src\question.py	Question.__hash__	0	0	1	100%
src\question.py	Question.__repr__	0	0	2	100%
src\question.py	(no function)	8	0	1	100%
src\quiz.py	Quiz.__init__	5	0	0	100%
src\quiz.py	Quiz.add_question	2	0	0	100%
src\quiz.py	Quiz.get_question	1	0	0	100%
src\quiz.py	Quiz.submit_answer	2	0	0	100%
src\quiz.py	Quiz.start_timer_if_needed	2	0	0	100%
src\quiz.py	Quiz.get_elapsed_time	2	0	1	100%
src\quiz.py	Quiz.is_time_expired	2	0	1	100%
src\quiz.py	Quiz.get_result	3	0	0	100%
src\quiz.py	Quiz.calculate_score	5	0	0	100%
src\quiz.py	Quiz.is_answer_correct	2	0	1	100%
src\quiz.py	Quiz.get_incorrect_answers	5	0	0	100%
src\quiz.py	Quiz.get_answer_details	3	0	0	100%
src\quiz.py	Quiz.get_questions_by_difficulty	1	0	0	100%
src\quiz.py	Quiz.get_questions_by_category	1	0	0	100%
src\quiz.py	Quiz.get_score_by_category	5	0	0	100%
src\quiz.py	Quiz.get_category_question_indices	1	0	0	100%
src\quiz.py	Quiz.calculate_category_score	5	0	0	100%
src\quiz.py	Quiz.__repr__	0	0	2	100%
src\quiz.py	(no function)	22	0	1	100%
src\result.py	QuizResult.__init__	3	0	0	100%
src\result.py	QuizResult.calculate_percentage	1	0	0	100%
src\result.py	QuizResult.is_perfect	1	0	0	100%
src\result.py	QuizResult.is_passing	1	0	0	100%
src\result.py	QuizResult.get_summary	1	0	0	100%
src\result.py	QuizResult.__repr__	0	0	2	100%
src\result.py	(no function)	7	0	1	100%
Total		106	0	14	100%

coverage.py v7.11.3, created at 2025-11-23 23:22 +0100