Static Testing Tools and White-box test design

Static Testing

Rubocop

Linting and style enforcement. Rules are defined in `./server/.rubocop.yml`.

```
# analyze project
bundle exec rubocop

# Run analyzer and make rubocop automatically fix linting issues
bundle exec rubocop -a
```

Successful rubocop:

```
bundle exec rubocop
Inspecting 114 files
..............................................................................
..................................

114 files inspected, no offenses detected
.rubocop.yml: Style/SpaceInsideArrayLiteralBrackets has the wrong namespace -
replace it with Layout/SpaceInsideArrayLiteralBrackets
```

Linting errors highlighted by rubocop

```
bundle exec rubocop
Inspecting 114 files
.....C.........................................................................
...................................

Offenses:

app/controllers/api/v1/auth_controller.rb:10:19: C: [Correctable]
Layout/SpaceInsideReferenceBrackets: Do not use space inside reference
brackets.
    code = params[ :code ]
                   ^
app/controllers/api/v1/auth_controller.rb:10:25: C:
Layout/SpaceInsideReferenceBrackets: Do not use space inside reference
brackets.
    code = params[ :code ]
                         ^
app/controllers/api/v1/auth_controller.rb:18:19: C: [Correctable]
Style/StringLiterals: Prefer single-quoted strings when you don't need string
interpolation or special symbols.
      redirect_to "https://munchora.pro/home"
                  ^^^^^^^^^^^^^^^^^^^^^^^^^^^


114 files inspected, 3 offenses detected, 2 offenses autocorrectable
.rubocop.yml: Style/SpaceInsideArrayLiteralBrackets has the wrong namespace -
replace it with Layout/SpaceInsideArrayLiteralBrackets
```

SonarQube

SonarQube (https://www.sonarsource.com/) a *static code analysis tool* that automatically inspects code for bugs, vulnerabilities, code smells, and test coverage without running the program.

SonarQube can be run through the use of Docker (https://medium.com/@index23/start-sonarqube-server-and-run-analyses-locally-with-docker-4550eb7112a3)

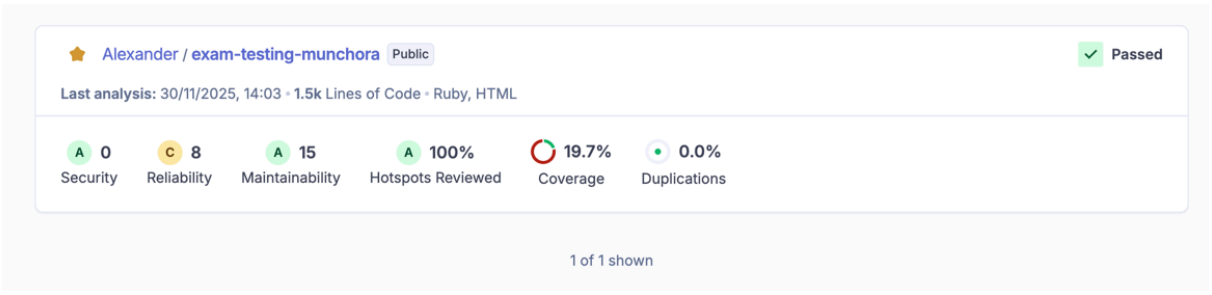
```
cd ./server
docker-compose -f docker-compose-sonar-qube.yml up
```

Go to **SonarQube dashboard** on `http://localhost:9000` - default credentials are login: `admin` password: `admin`

Use following command to scan project with *SonnarScanner*:

```
docker run \
    --rm \
    -v "$(pwd):/usr/src" \
    --network="host" \
    -e SONAR_HOST_URL="http://localhost:9000" \
    -e SONAR_SCANNER_OPTS="-Dsonar.projectKey=server -Dsonar.sources=./ -
Dsonar.test=test -Dsonar.javascript.lcov.reportPaths=test/coverage/lcov.info" \
    -e SONAR_TOKEN="${SONAR_TOKEN}" \
    sonarsource/sonar-scanner-cli
```

Sonarqube do also provide a cloud based solution which can be included in CI pipeline:


Alexander / exam-testing-munchora Public

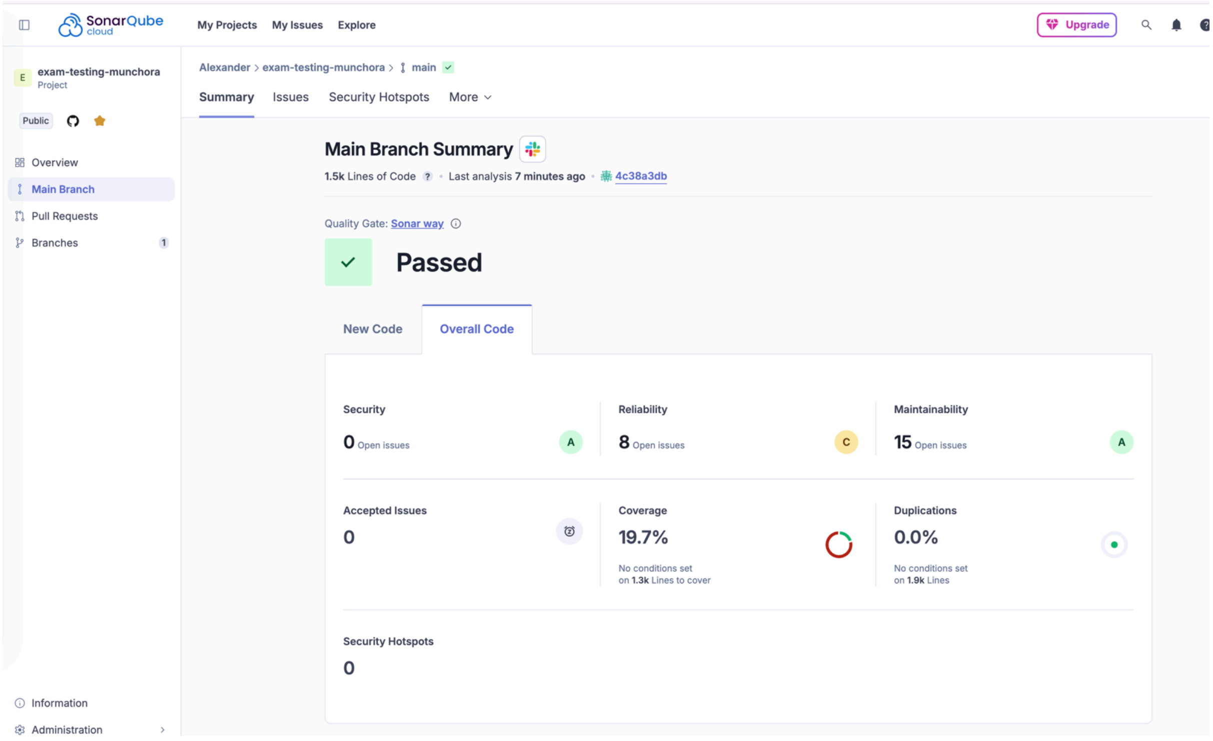
Passed

Last analysis: 30/11/2025, 14:03 • 1.5k Lines of Code • Ruby, HTML

A 0 Security
C 8 Reliability
A 15 Maintainability
A 100% Hotspots Reviewed
19.7% Coverage
0.0% Duplications

1 of 1 shown

The total test coverage is displayed, which is a bit low since the controllers will be tests by Postman/Newman CI and not provided for SonarQube. Amount of Security Risks, Duplications, Reliability and Maintainability issues are also displayed.



Main Branch Summary

1.5k Lines of Code • Last analysis 7 minutes ago • 4c38a3db

Quality Gate: Sonar way

Passed

New Code | Overall Code

Security 0 Open issues A	Reliability 8 Open issues C	Maintainability 15 Open issues A
Accepted Issues 0	Coverage 19.7% No conditions set on 1.3k Lines to cover	Duplications 0.0% No conditions set on 1.9k Lines
Security Hotspots 0		

Shows the same stats as previous image just a bit more explanatory

White Box Design Techniques

Focuses on the code and the structural elements.

Statement Coverage: Measures whether each line of code has been executed by the test suite at least once. **decision coverage**: Measures whether **each decision (true/false outcome) of every conditional statement** has been exercised at least once.

The code coverage is collected by *rspec* and *simplecov*, so whenever test command `bundle exec rspec` is executed a coverage report is generated in JSON and HTML located at `./server/coverage/index.html` showing line-by-line coverage and totals.

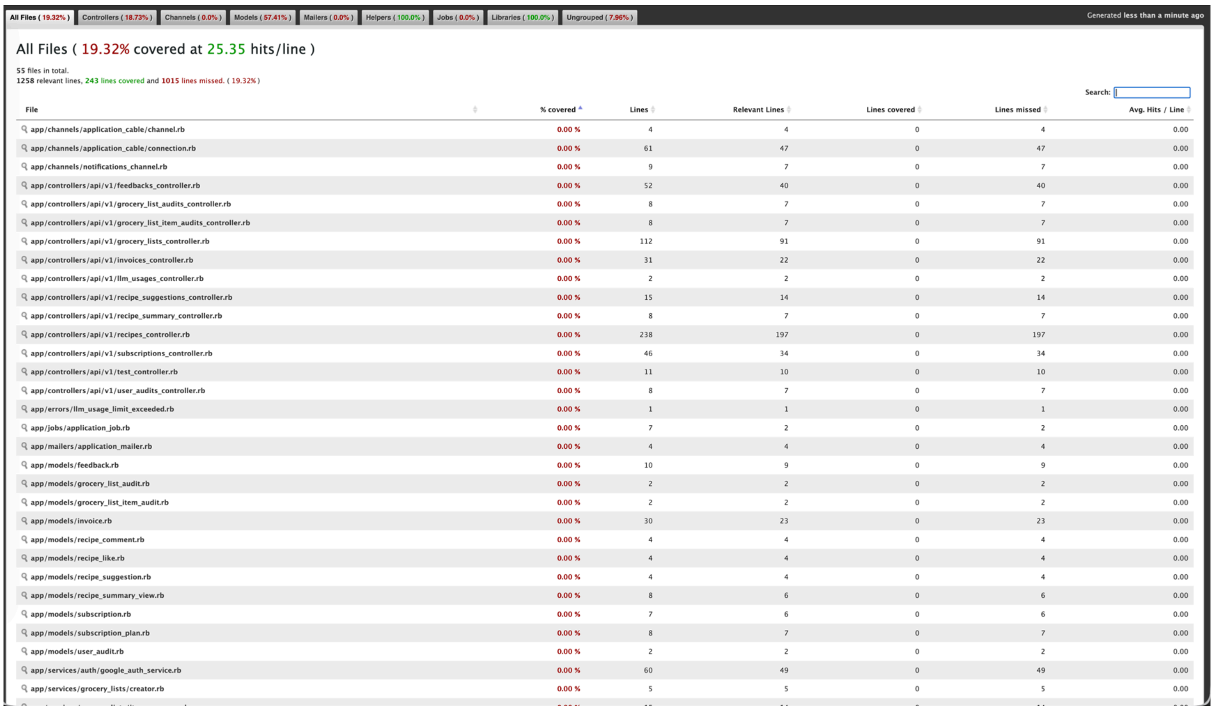
Default *simplecov* measures coverage by line coverage which can be an issue with ternary operator/one line conditionals `number.odd? ? "odd" : "even"`

it can be set to use branch coverage instead:

```
SimpleCov.start do
  enable_coverage :branch
end
```

The code coverage report is an excellent tool that helps you ensure tests are efficiently distributed throughout the codebase. It does this by visually identifying uncovered lines and branches of logic.

Code coverage reports helps designing tests that validate not only the common "happy path", but also critical edge cases and "negative paths" (like error handling, exceptions, and the else block of an if statement) where alternative parts of the system are executed.


All Files (19.32%) | Controllers (18.73%) | Channels (0.0%) | Models (57.41%) | Mailers (0.0%) | Helpers (100.0%) | Jobs (0.0%) | Libraries (100.0%) | Ungrouped (7.96%)

Generated less than a minute ago

All Files (19.32% covered at 25.35 hits/line)

55 files in total.
1258 relevant lines, 243 lines covered and 1015 lines missed. (19.32%)

Search:

File	% covered	Lines	Relevant Lines	Lines covered	Lines missed	Avg. Hits / Line
app/channels/application_cable/channel.rb	0.00 %	4	4	0	4	0.00
app/channels/application_cable/connection.rb	0.00 %	61	47	0	47	0.00
app/channels/notifications_channel.rb	0.00 %	9	7	0	7	0.00
app/controllers/api/v1/feedbacks_controller.rb	0.00 %	52	40	0	40	0.00
app/controllers/api/v1/grocery_list_audits_controller.rb	0.00 %	8	7	0	7	0.00
app/controllers/api/v1/grocery_list_item_audits_controller.rb	0.00 %	8	7	0	7	0.00
app/controllers/api/v1/grocery_lists_controller.rb	0.00 %	112	91	0	91	0.00
app/controllers/api/v1/invoices_controller.rb	0.00 %	31	22	0	22	0.00
app/controllers/api/v1/llm_usages_controller.rb	0.00 %	2	2	0	2	0.00
app/controllers/api/v1/recipe_suggestions_controller.rb	0.00 %	15	14	0	14	0.00
app/controllers/api/v1/recipe_summary_controller.rb	0.00 %	8	7	0	7	0.00
app/controllers/api/v1/recipes_controller.rb	0.00 %	238	197	0	197	0.00
app/controllers/api/v1/subscriptions_controller.rb	0.00 %	46	34	0	34	0.00
app/controllers/api/v1/test_controller.rb	0.00 %	11	10	0	10	0.00
app/controllers/api/v1/user_audits_controller.rb	0.00 %	8	7	0	7	0.00
app/errors/llm_usage_limit_exceeded.rb	0.00 %	1	1	0	1	0.00
app/jobs/application_job.rb	0.00 %	7	2	0	2	0.00
app/mailers/application_mailer.rb	0.00 %	4	4	0	4	0.00
app/models/feedback.rb	0.00 %	10	9	0	9	0.00
app/models/grocery_list_audit.rb	0.00 %	2	2	0	2	0.00
app/models/grocery_list_item_audit.rb	0.00 %	2	2	0	2	0.00
app/models/invoice.rb	0.00 %	30	23	0	23	0.00
app/models/recipe_comment.rb	0.00 %	4	4	0	4	0.00
app/models/recipe_like.rb	0.00 %	4	4	0	4	0.00
app/models/recipe_suggestion.rb	0.00 %	4	4	0	4	0.00
app/models/recipe_summary_view.rb	0.00 %	8	6	0	6	0.00
app/models/subscription.rb	0.00 %	7	6	0	6	0.00
app/models/subscription_plan.rb	0.00 %	8	7	0	7	0.00
app/models/user_audit.rb	0.00 %	2	2	0	2	0.00
app/services/auth/google_auth_service.rb	0.00 %	60	49	0	49	0.00
app/services/grocery_lists/creator.rb	0.00 %	5	5	0	5	0.00

User creation | User #initialize

Test cases 100% statement coverage

```
#1. first_name="John", last_name="Doe", email="john@doe.com", provider=nil,
uid=nil, password="secret123"
    # TRUE path provider.blank? & TRUE :password is_a?(String)

#2. first_name="John", last_name="Doe", email="john@doe.com",
provider="google", uid="abc123", password=nil
    # TRUE branch of both provider.present? and uid.present?
```

Test cases 100% decision coverage

```
#1. first_name="John", last_name="Doe", email="john@doe.com", provider=nil,
uid=nil, password="secret123"
    # TRUE path for provider.blank? and TRUE for :password is_a?(String)

#2. first_name="John", last_name="Doe", email="john@doe.com", provider=nil,
uid=nil, password=1234
    # FALSE branch of :pasword is_a?(String)

#3. first_name="John", last_name="Doe", email="john@doe.com",
provider="google", uid=nil, password=nil
    # FALSE branch of uid.present? when provider.present?

#3. first_name="John", last_name="Doe", email="john@doe.com",
provider="google", uid="abc123", password=nil
    # TRUE branch of both provider.present? and uid.present?
```