Chapter 5: Testing

5.1 Introduction

This document outlines the testing strategy, scope, and detailed test cases for the CodeReview AI project, covering both backend (Django) and frontend (React) applications. The goal is to ensure reliability, correctness, and maintainability through comprehensive automated testing.

5.2 Features to be tested/not to be tested

5.2.1 Features to be tested

Backend

- Models: Creation, constraints, and string representations for all major models.
- Serializers: Serialization, deserialization, and validation for all major serializers.
- Services: Core service functions, including GitHub API integration (with mocks).
- Tasks: Celery/async task logic and cost calculation.
- Permissions: All custom permission classes.
- Webhooks: Main event dispatch logic.
- Views & ViewSets: Key endpoints, admin endpoints, and their permissions.
- Async Handlers: Async event handler logic.

Frontend

- Authentication Flow: Navigation, login, OAuth callback, dashboard redirection.
- Dashboard: User statistics, recent activity.
- Repository Management: Repositories/orgs list, overview, PRs, commits, review page.
- Component Functionality: Navbar, modal, layout, feedback, thread, report components.

5.2.2 Features not to be tested

- Third-party libraries (Django, DRF, Celery, Cypress, etc.).
- UI/Frontend (for backend plan) and backend (for frontend plan) except for integration points.
- Database engine internals.
- External APIs (only integration points are mocked and tested).
- Performance/Load (not covered in this functional test plan).
- Mobile responsiveness (manual testing performed for frontend).

5.3 Pass/Fail Criteria

- **Pass:**
  - All automated tests must pass (status `ok` or `PASSED`).
  - No unhandled exceptions or errors during test execution.
  - All critical features and endpoints must be covered by at least one test.
- **Fail:**
  - Any test failure (status `FAIL` or `ERROR`) is considered a fail.
  - Unhandled exceptions, 500 errors, or database constraint violations during tests.

5.4 Approach/Strategy

- **Automated Testing:**
  - Backend: Django's `TestCase`, DRF's `APIClient`, mocks for external APIs, async test support.
  - Frontend: Cypress for E2E and component tests, API mocking/stubbing.
- **Test Organization:**

- Tests are organized by logical area: models, serializers, services, tasks, permissions, webhooks, views, viewsets, async (backend); flows and components (frontend).
- Each test is isolated and uses its own setup.

- **Continuous Integration:**
  - Tests are intended to be run on every commit/pull request.
- **Edge Cases:**
  - Tests include invalid input, permission denied, and error handling scenarios.

5.5 Test Cases with Specifications

Backend Test Cases

Test Case ID	Area	Test Name / File	Description	Input/Setup	Expected Result
TC-001	Model	test_models.py / test_user_str	User string representation	Create User	Username as str
TC-002	Model	test_models.py / test_repo_str	Repository string representation	Create Repository	Repo name as str
TC-003	Model	test_models.py / test_pr_str	PullRequest string representation	Create PR	PR title in str
TC-004	Model	test_models.py / test_commit_str	Commit string representation	Create Commit	Commit hash as str
TC-005	Model	test_models.py / test_review_str	Review string representation	Create Review	PR/Commit in str
TC-006	Model	test_models.py / test_thread_str	Thread string representation	Create Thread	Thread info as str
TC-007	Model	test_models.py / test_comment_str	Comment string representation	Create Comment	User/thread in str
TC-008	Model	test_models.py / test_llm_usage_str	LLMUsage string representation	Create LLMUsage	User/review in str
TC-009	Model	test_models.py / test_feedback_str	ReviewFeedback string representation	Create Feedback	User/review in str
TC-010	Model	test_models.py / test_webhook_event_str	WebhookEventLog string representation	Create WebhookEventLog	Event info as str
TC-011	Model	test_models.py / test_review_constraint	Review constraint (PR or commit required)	Create invalid Review	ValidationError
TC-012	Serializer	test_serializers.py / test_user_serializer	UserSerializer serialization	Serialize User	Correct JSON output
TC-013	Serializer	test_serializers.py / test_repository_serializer	RepositorySerializer serialization	Serialize Repository	Correct JSON output
TC-014	Serializer	test_serializers.py / test_pr_serializer	PRSerializer serialization	Serialize PR	Correct JSON output
TC-015	Serializer	test_serializers.py / test_commit_serializer	CommitSerializer serialization	Serialize Commit	Correct JSON output
TC-016	Serializer	test_serializers.py / test_review_serializer	ReviewSerializer serialization	Serialize Review	Correct JSON output

Test Case ID	Area	Test Name / File	Description	Input/Setup	Expected Result
TC-017	Serializer	test_serializers.py / test_thread_serializer	ThreadSerializer serialization	Serialize Thread	Correct JSON output
TC-018	Serializer	test_serializers.py / test_comment_serializer	CommentSerializer serialization	Serialize Comment	Correct JSON output
TC-019	Serializer	test_serializers.py / test_llm_usage_serializer	LLMUsageSerializer serialization	Serialize LLMUsage	Correct JSON output
TC-020	Serializer	test_serializers.py / test_feedback_serializer	ReviewFeedbackSerializer serialization	Serialize Feedback	Correct JSON output
TC-021	Serializer	test_serializers.py / test_webhook_event_serializer	WebhookEventLogSerializer serialization	Serialize WebhookEventLog	Correct JSON output
TC-022	Service	test_services.py / test_generate_and_validate_oauth_state	OAuth state generation/validation	Simulate request/session	State generated/valid
TC-023	Service	test_services.py / test_get_github_oauth_redirect_url	GitHub OAuth URL generation	Provide state	URL with params
TC-024	Service	test_services.py / test_exchange_code_for_github_token	GitHub token exchange (mocked)	Mock code, call service	Token returned
TC-025	Service	test_services.py / test_get_github_user_info	GitHub user info fetch (mocked)	Mock token, call service	User info returned
TC-026	Task	test_tasks.py / test_calculate_cost_default	Cost calculation (default model)	Token usage dict	Correct cost value
TC-027	Task	test_tasks.py / test_calculate_cost_gpt4	Cost calculation (gpt-4 model)	Token usage dict	Correct cost value
TC-028	Task	test_tasks.py / test_calculate_cost_llama	Cost calculation (llama model)	Token usage dict	Correct cost value
TC-029	Permission	test_permissions.py / test_is_repository_owner	IsRepositoryOwner positive/negative	Owner/non-owner request	True/False
TC-030	Permission	test_permissions.py / test_can_access_repository_owner	CanAccessRepository (owner)	Owner request	True
TC-031	Permission	test_permissions.py / test_can_access_repository_not_owner	CanAccessRepository (not owner)	Non-owner request	False
TC-032	Webhook	test_webhooks.py / test_handle_event_dispatch	Webhook event dispatch	Simulate event	Handler called
TC-033	View	test_views.py / test_current_user_view	/api/user/me/ returns user info	Authenticated GET	200/400/404, user data
TC-034	View	test_views.py / test_user_repositories_view	/api/user/repositories/ returns repos	Authenticated GET	200/400/404
TC-035	ViewSet	test_viewsets.py / test_repository_retrieve	Retrieve repository endpoint	GET /api/repositories/{id}	200/403/404
TC-036	ViewSet	test_viewsets.py / test_repository_delete	Delete repository endpoint	DELETE /api/repositories/{id}	204/403/404
TC-037	ViewSet	test_viewsets.py / test_admin_stats_view	Admin stats endpoint	GET /api/admin/stats/	200/403/404

Test Case ID	Area	Test Name / File	Description	Input/Setup	Expected Result
TC-038	ViewSet	test_viewsets.py / test_admin_user_list_view	Admin user list endpoint	GET /api/admin/users/	200/403/404
TC-039	ViewSet	test_viewsets.py / test_admin_user_update_view	Admin user update endpoint	PUT /api/admin/users/{id}/	200/403/404
TC-040	Async	test_async.py / test_handle_event_dispatch	Async handler event dispatch	Simulate async event	Awaited handler

Frontend Test Cases

A. Authentication Flow

Test Case ID	Description	Steps	Expected Result
TC-Auth-01	Landing page navigation	Visit `/`, click "Get Started Free"	Navigates to `/login`
TC-Auth-02	Login page UI	Visit `/login`	"Sign in to CodeReview AI" and "Continue with GitHub" are visible
TC-Auth-03	OAuth callback	Visit `/auth/callback?token=fake-token`	Redirects to `/dashboard` and dashboard loads

B. Dashboard

Test Case ID	Description	Steps	Expected Result
TC-Dash-01	Dashboard stats	Visit `/dashboard`	Stats for reviews, repositories, PRs, and commits are visible

C. Repository Management

Test Case ID	Description	Steps	Expected Result
TC-Repo-01	View repositories	Visit `/repositories`	List of repositories and organizations is displayed
TC-Repo-02	Repository overview	Visit `/repo/1/overview`	Overview and repo details are visible
TC-Repo-03	Pull requests list	Visit `/repo/1/pulls`	List of pull requests is displayed
TC-Repo-04	Review page	Visit `/review/1`	Review details are visible

D. Component Tests

Test Case ID	Component	Description	Steps	Expected Result
TC-Comp-01	Navbar	Unauthenticated	Render Navbar with no user	"Login" and "Home" links are visible
TC-Comp-02	Navbar	Authenticated	Render Navbar with user	"Logout" and user info are visible
TC-Comp-03	Modal	Confirm dialog	Render Modal with type "confirm"	"Cancel" and "Delete" buttons work
TC-Comp-04	Layout	Authenticated	Render Layout with user	Sidebar and user menu are visible

Test Case ID	Component	Description	Steps	Expected Result
TC-Comp-05	ReviewFeedback	Feedback submission	Submit feedback	Feedback and AI response are shown
TC-Comp-06	ThreadList	Thread reply	Expand thread and reply	Message is sent and displayed
TC-Comp-07	ReviewReport	Summary and issues	Render with review data	Summary and issues are displayed

Note:

- All backend test cases are automated using Django's test runner.
- All frontend test cases are automated using Cypress and can be run via `npx cypress run` (E2E) or `npx cypress run-ct` (component).
- Some tests (e.g., real OAuth, backend integration) are simulated or mocked due to environment constraints.