IT314 - Software Engineering

Prof : Saurabh Tiwari

Group 9 : Pull Panda

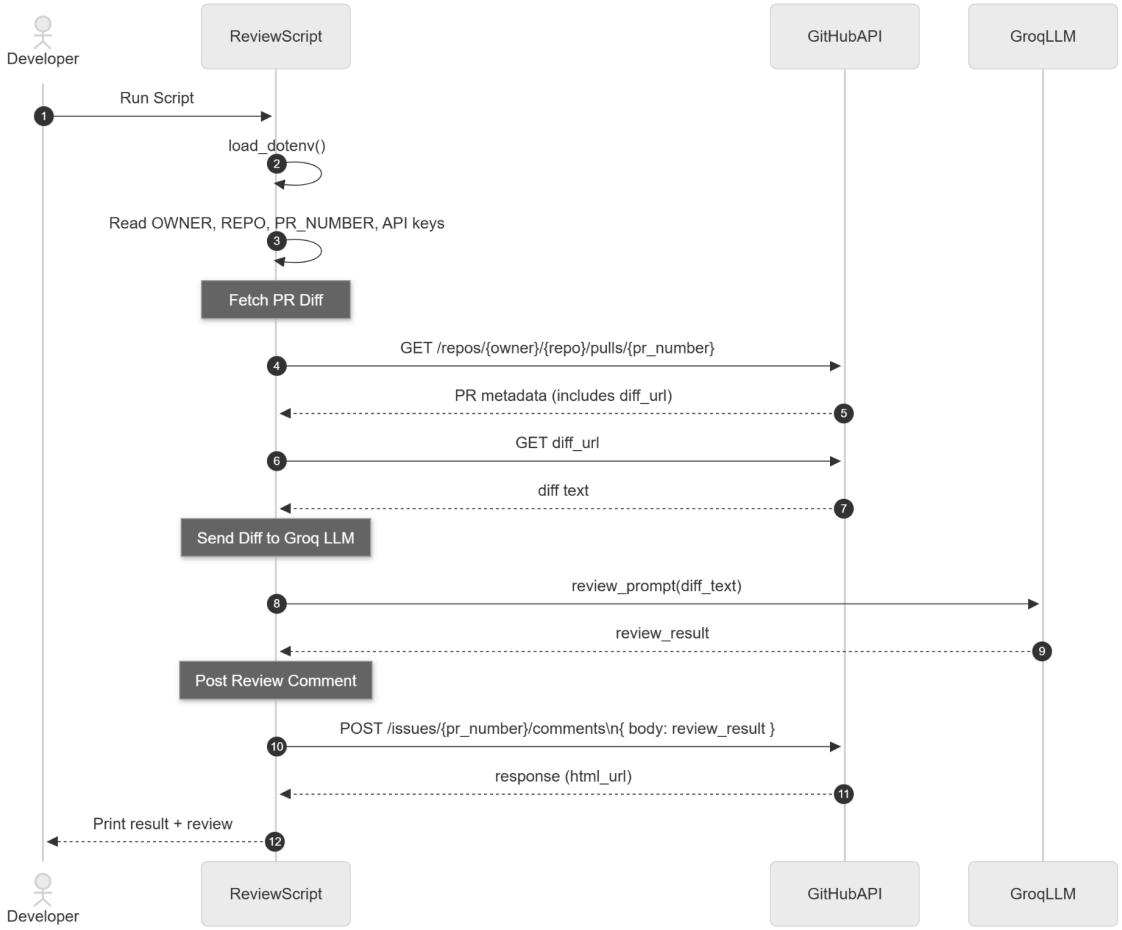
(AI-powered Pull Request Reviewer)

Design Document

Sequence diagrams

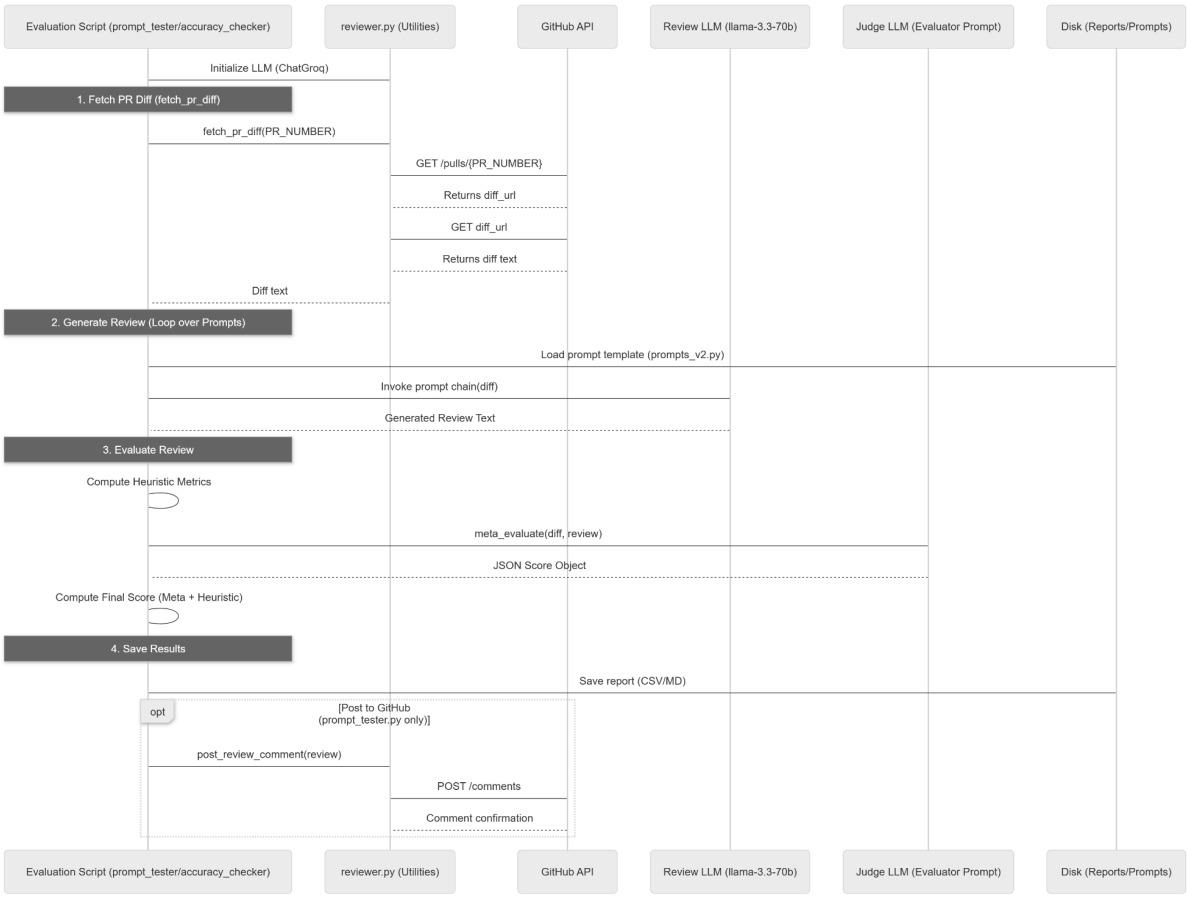
Sprint 1

(version_1)

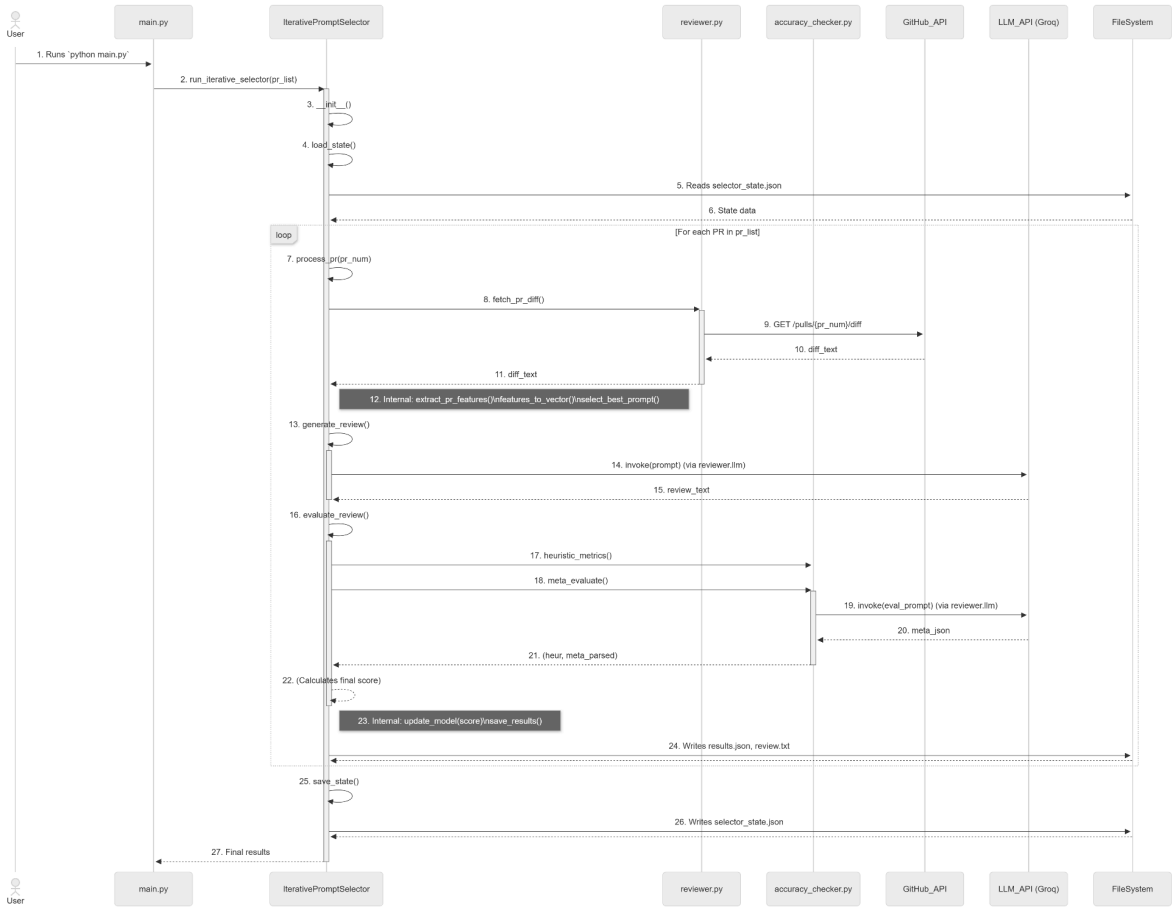


Sprint 2

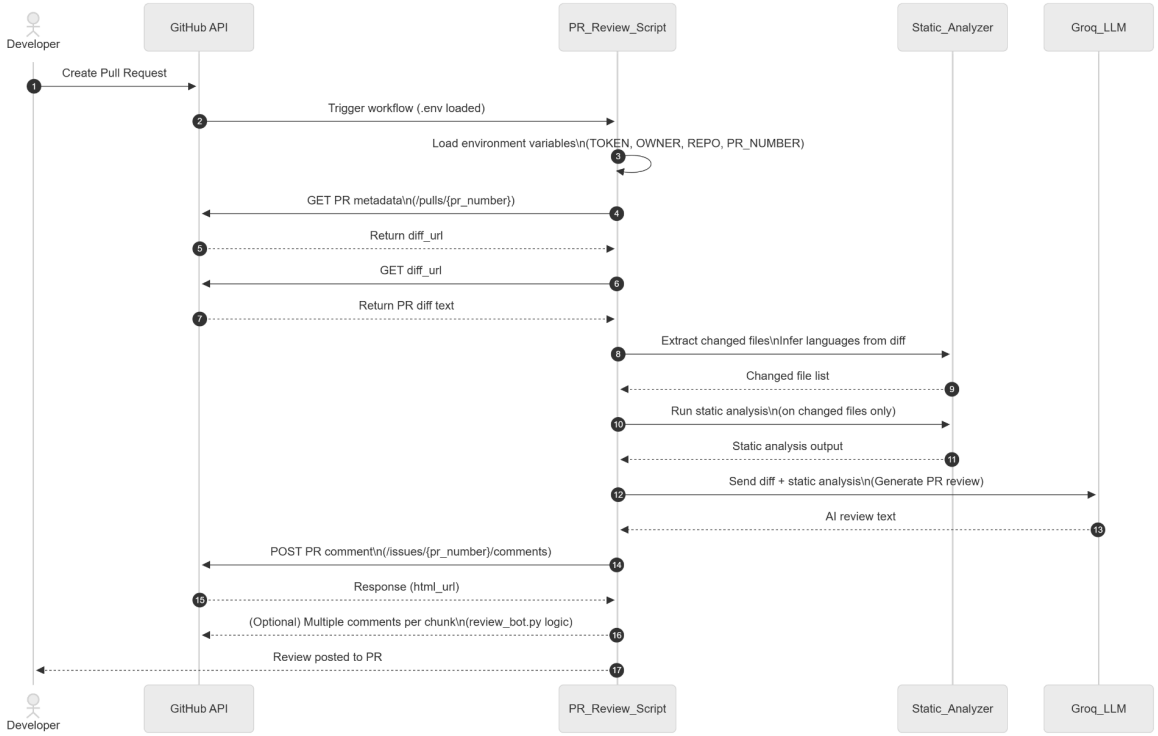
(version_1.1)



(version_1.2)

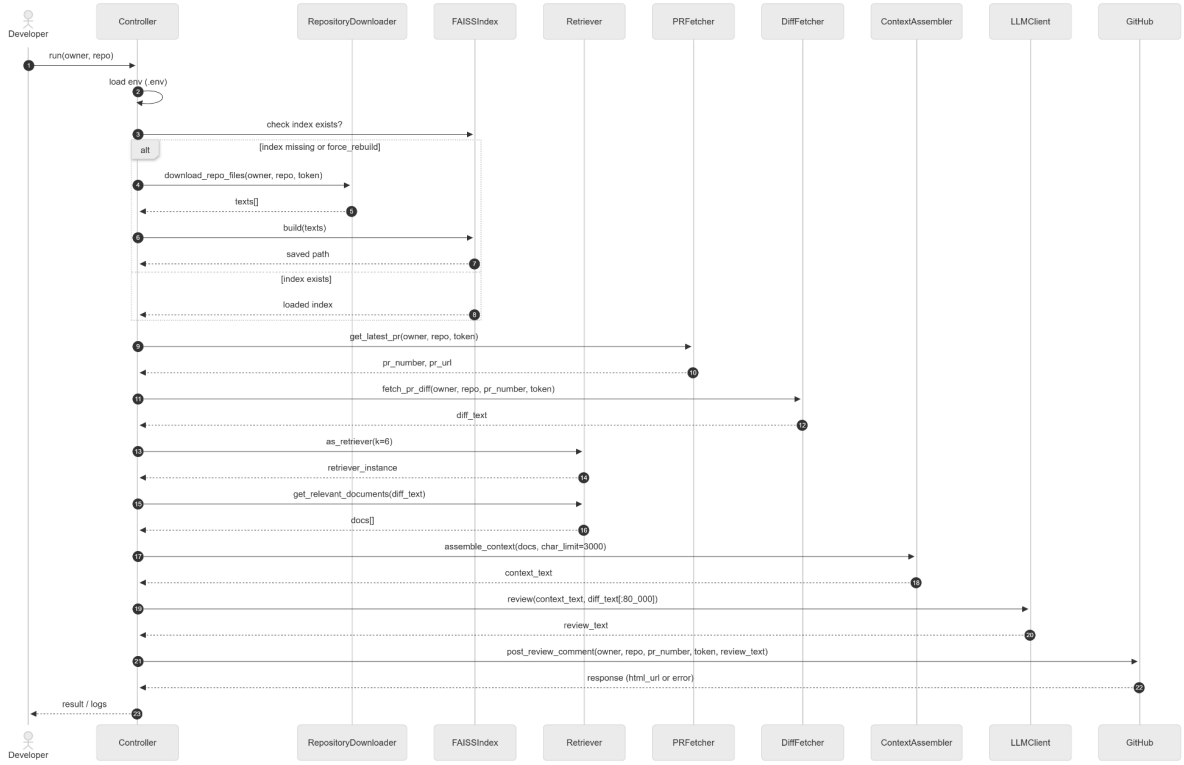

User
main.py
IterativePromptSelector
reviewer.py
accuracy_checker.py
GitHub_API
LLM_API (Groq)
FileSystem
1. Runs `python main.py`
2. run_iterative_selector(pr_list)
3. __init__()
4. load_state()
5. Reads selector_state.json
6. State data
loop
[For each PR in pr_list]
7. process_pr(pr_num)
8. fetch_pr_diff()
9. GET /pulls/{pr_num}/diff
10. diff_text
11. diff_text
12. Internal: extract_pr_features()/vfeatures_to_vector()/vselect_best_prompt()
13. generate_review()
14. invoke(prompt) (via reviewer.llm)
15. review_text
16. evaluate_review()
17. heuristic_metrics()
18. meta_evaluate()
19. invoke(eval_prompt) (via reviewer.llm)
20. meta_json
21. (heur, meta_parsed)
22. (Calculates final score)
23. Internal: update_model(score)/vsave_results()
24. Writes results.json, review.txt
25. save_state()
26. Writes selector_state.json
27. Final results

(version_2)


Developer
GitHub API
PR_Review_Script
Static_Analyzer
Groq_LLM
Create Pull Request
Trigger workflow (.env loaded)
Load environment variables\n(TOKEN, OWNER, REPO, PR_NUMBER)
GET PR metadata\n(/pulls/{pr_number})
Return diff_url
GET diff_url
Return PR diff text
Extract changed files\nInfer languages from diff
Changed file list
Run static analysis\n(on changed files only)
Static analysis output
Send diff + static analysis\n(Generate PR review)
AI review text
POST PR comment\n(/issues/{pr_number}/comments)
Response (html_url)
(Optional) Multiple comments per chunk\n(review_bot.py logic)
Review posted to PR
Developer
GitHub API
PR_Review_Script
Static_Analyzer
Groq_LLM

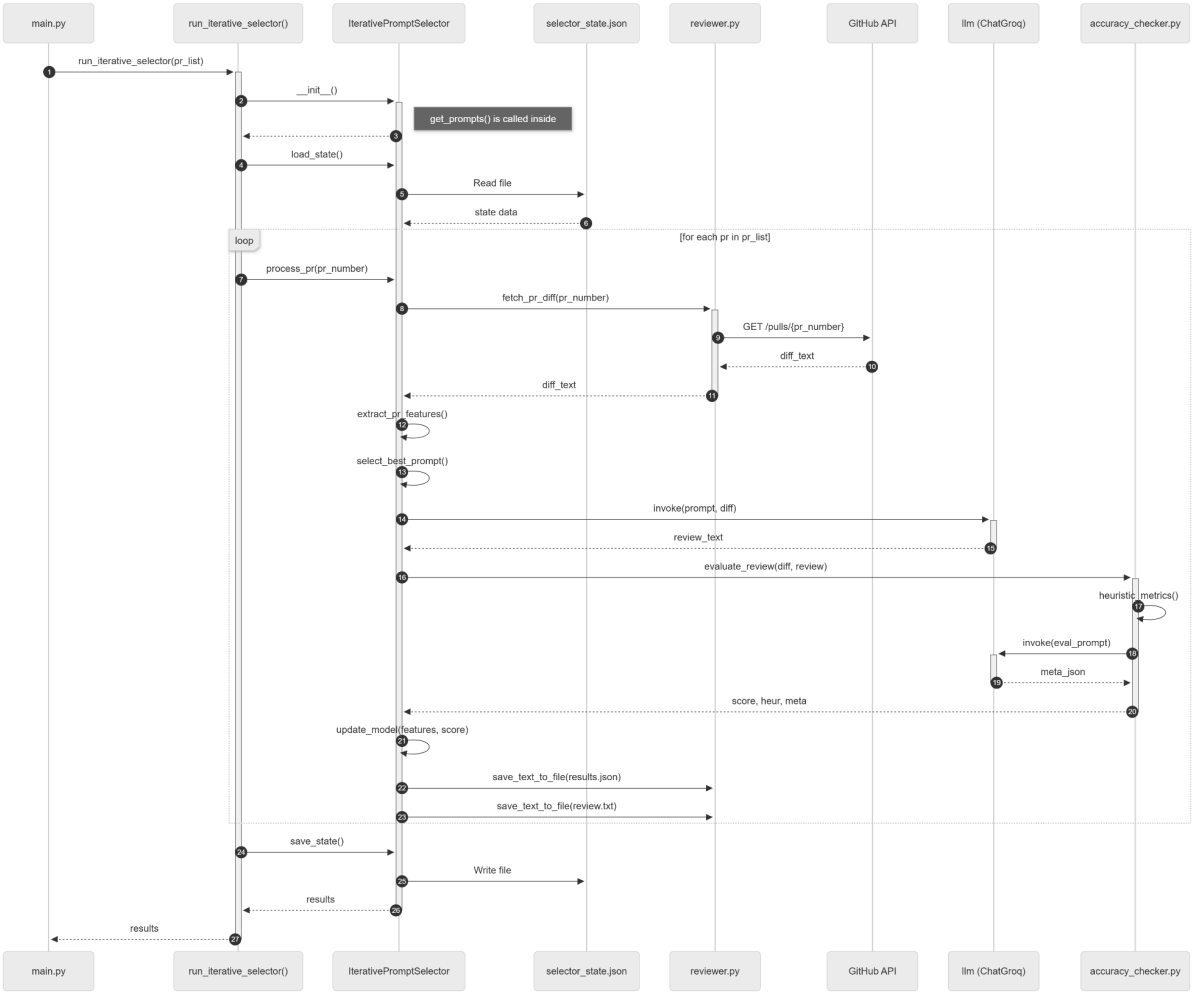
Sprint 3

(version_1_traditional _rag)

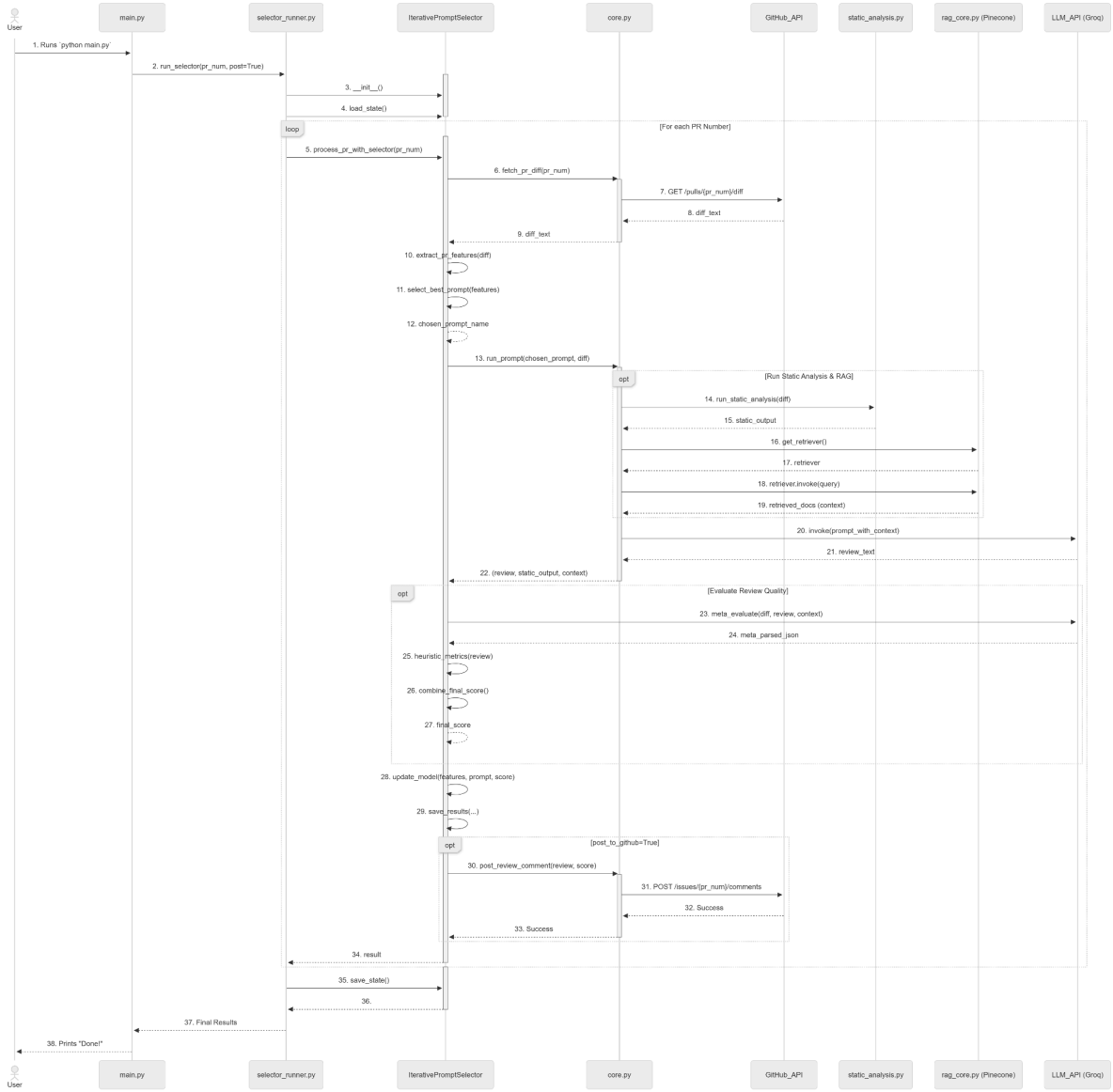


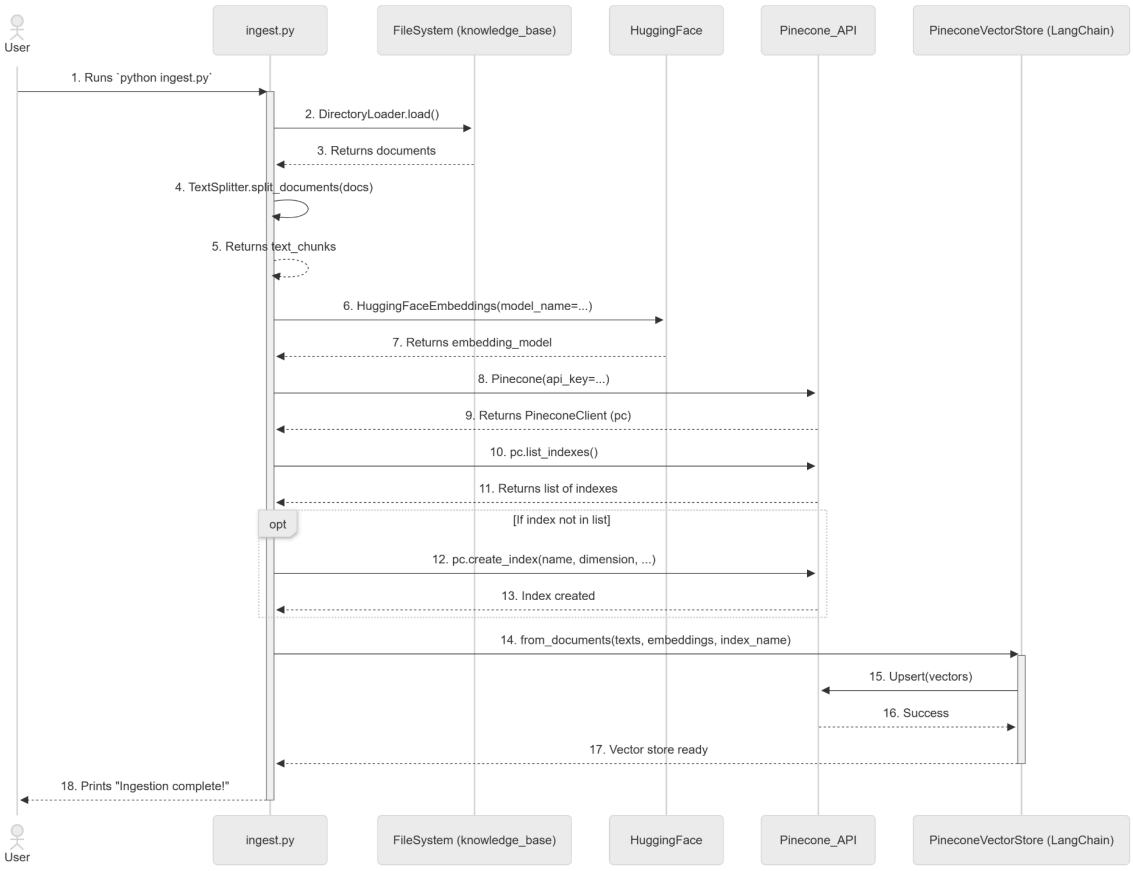
Sprint 4

(version_1.3)

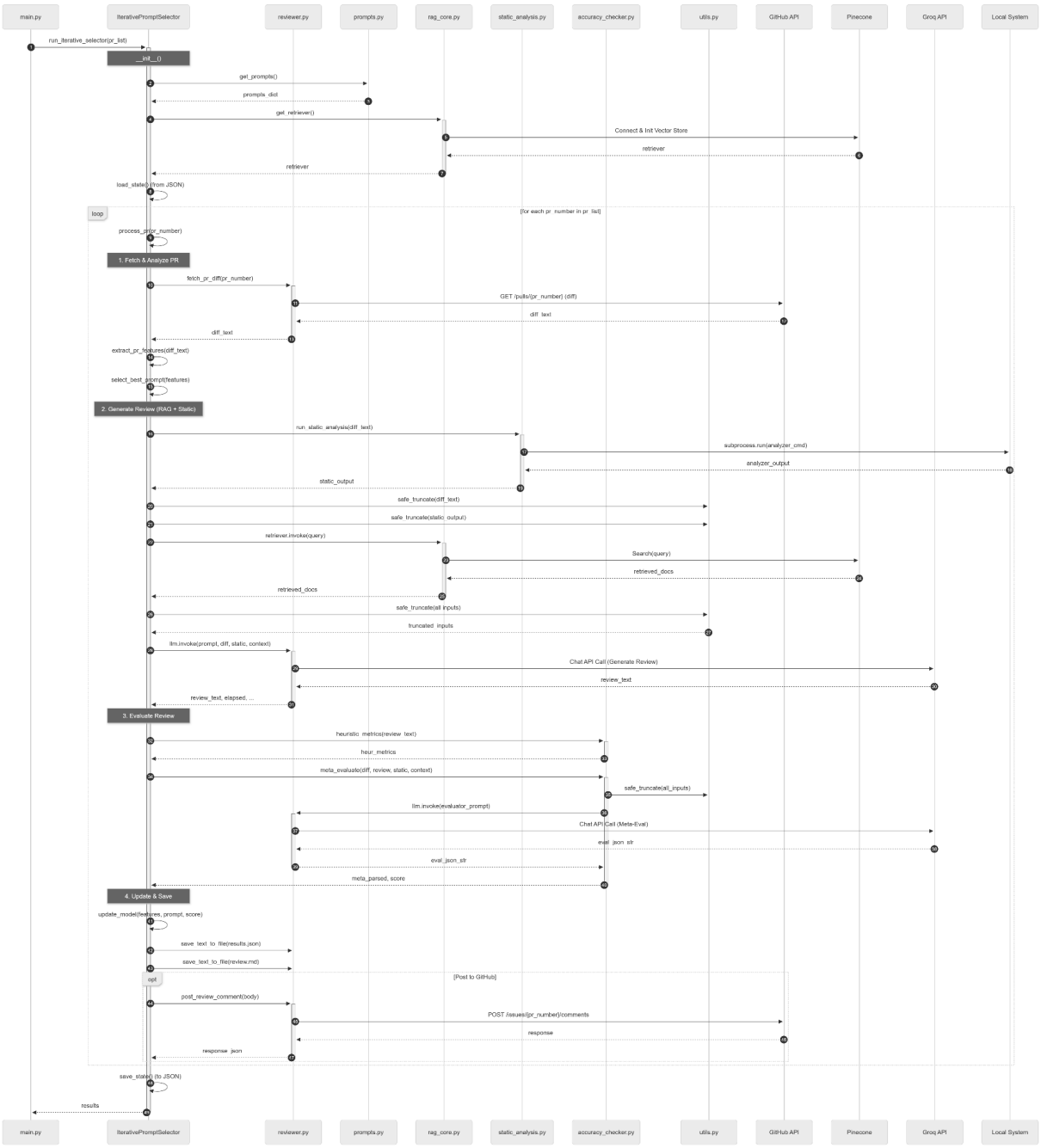


(Version_1.2_rag)


User
main.py
selector_runner.py
IterativePromptSelector
core.py
GitHub_API
static_analysis.py
rag_core.py (Pinecone)
LLM_API (Groq)
1. Runs `python main.py`
2. run_selector(pr_num, post=True)
3. __init__()
4. load_state()
loop
[For each PR Number]
5. process_pr_with_selector(pr_num)
6. fetch_pr_diff(pr_num)
7. GET /pulls/{pr_num}/diff
8. diff_text
9. diff_text
10. extract_pr_features(diff)
11. select_best_prompt(features)
12. chosen_prompt_name
13. run_prompt(chosen_prompt, diff)
opt
[Run Static Analysis & RAG]
14. run_static_analysis(diff)
15. static_output
16. get_retriever()
17. retriever
18. retriever.invoke(query)
19. retrieved_docs (context)
20. invoke(prompt_with_context)
21. review_text
22. (review, static_output, context)
opt
[Evaluate Review Quality]
23. meta_evaluate(diff, review, context)
24. meta_parsed_json
25. heuristic_metrics(review)
26. combine_final_score()
27. final_score
28. update_model(features, prompt, score)
29. save_results(...)
opt
[post_to_github=True]
30. post_review_comment(review, score)
31. POST /issues/{pr_num}/comments
32. Success
33. Success
34. result
35. save_state()
36.
37. Final Results
38. Prints "Done!"

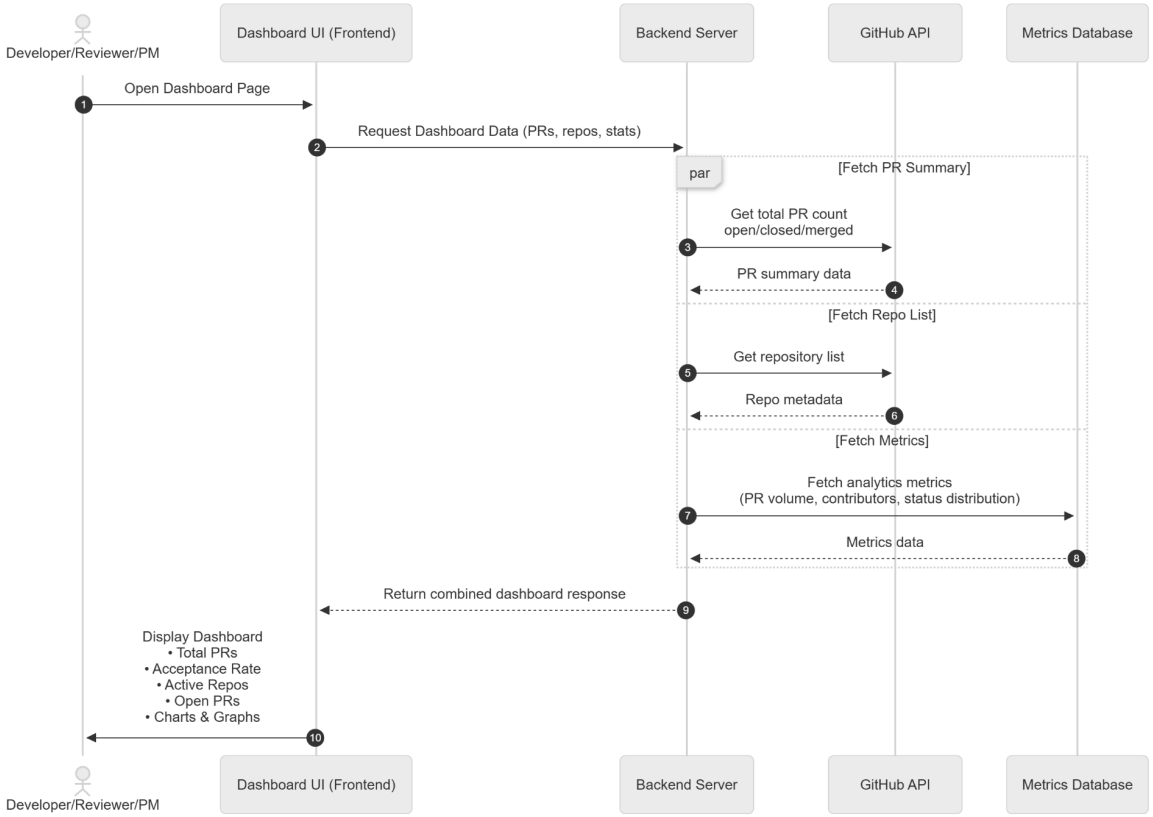


(version_1.3_rag)


main.py
IterativePromptSelector
reviewer.py
prompts.py
rag_core.py
static_analysis.py
accuracy_checker.py
utils.py
GitHub API
Pinecone
Groq API
Local System
run_iterative_selector(pr_list)
__init__()
get_prompts()
prompts_dict
get_retriever()
Connect & Init Vector Store
retriever
retriever
load_state() (from JSON)
loop
[for each pr_number in pr_list]
process_pr(pr_number)
1. Fetch & Analyze PR
fetch_pr_diff(pr_number)
GET /pulls/{pr_number} (diff)
diff_text
diff_text
extract_pr_features(diff_text)
select_best_prompt(features)
2. Generate Review (RAG + Static)
run_static_analysis(diff_text)
subprocess.run(analyzer_cmd)
analyzer_output
static_output
safe_truncate(diff_text)
safe_truncate(static_output)
retriever.invoke(query)
Search(query)
retrieved_docs
retrieved_docs
safe_truncate(all_inputs)
truncated_inputs
llm.invoke(prompt, diff, static, context)
Chat API Call (Generate Review)
review_text
review_text, elapsed, ...
3. Evaluate Review
heuristic_metrics(review_text)
heur_metrics
meta_evaluate(diff, review, static, context)
safe_truncate(all_inputs)
llm.invoke(evaluator_prompt)
Chat API Call (Meta-Eval)
eval_json_str
eval_json_str
meta_parsed, score
4. Update & Save
update_model(features, prompt, score)
save_text_to_file(results.json)
save_text_to_file(review.md)
opt
[Post to GitHub]
post_review_comment(body)
POST /issues/{pr_number}/comments
response
response_json
save_state() (to JSON)
results

Sprint 5

(Dashboard)


Developer/Reviewer/PM
Dashboard UI (Frontend)
Backend Server
GitHub API
Metrics Database
Open Dashboard Page
1
Request Dashboard Data (PRs, repos, stats)
2
par
[Fetch PR Summary]
Get total PR count
open/closed/merged
3
PR summary data
4
[Fetch Repo List]
Get repository list
5
Repo metadata
6
[Fetch Metrics]
Fetch analytics metrics
(PR volume, contributors, status distribution)
7
Metrics data
8
Return combined dashboard response
9
Display Dashboard
• Total PRs
• Acceptance Rate
• Active Repos
• Open PRs
• Charts & Graphs
10
Developer/Reviewer/PM
Dashboard UI (Frontend)
Backend Server
GitHub API
Metrics Database