Team 9 - Runtime Terror
F25

Sprint Review and Retrospective

Project 3 - Repository Architecture Diagramming

Restatement of Purpose

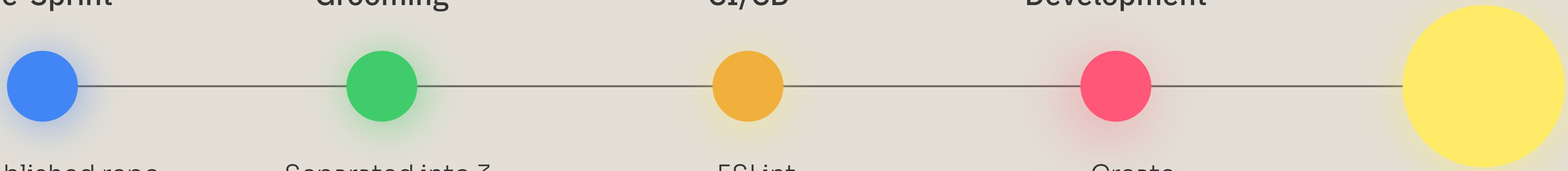
This project aims to turn code into clarity by automating architecture visualization, making complex systems understandable and collaboration effortless.

Goal of this sprint

The goal of our first sprint was to establish a standard set of organizational practices, as well as to design and implement the Stage 1 of our MVP of our web application.

Project roadmap

Sprint 1: 11/5 - 11/14



Pre-Sprint

-Established repo rulesets/structure
-Conducted UX research
-Created PRD
-Created initial user/system flows
-Designed first draft wireframes

Grooming

-Separated into 3 cross-functioning subteams
- Created Sprint Epics and child tickets
-Scheduled sprint planning, stand-ups, and sprint review / retrospective meetings

CI/CD

- ESLint
- JSDocs
- Test suites (preliminary)
-Codacy
- Coverage Badges
- Peer PR reviews

Development

- Create components for homepage UI
- Connect to necessary APIs
- Create JSON infra for local data storage
- Create base for foundational diagramming algorithms and LLM support

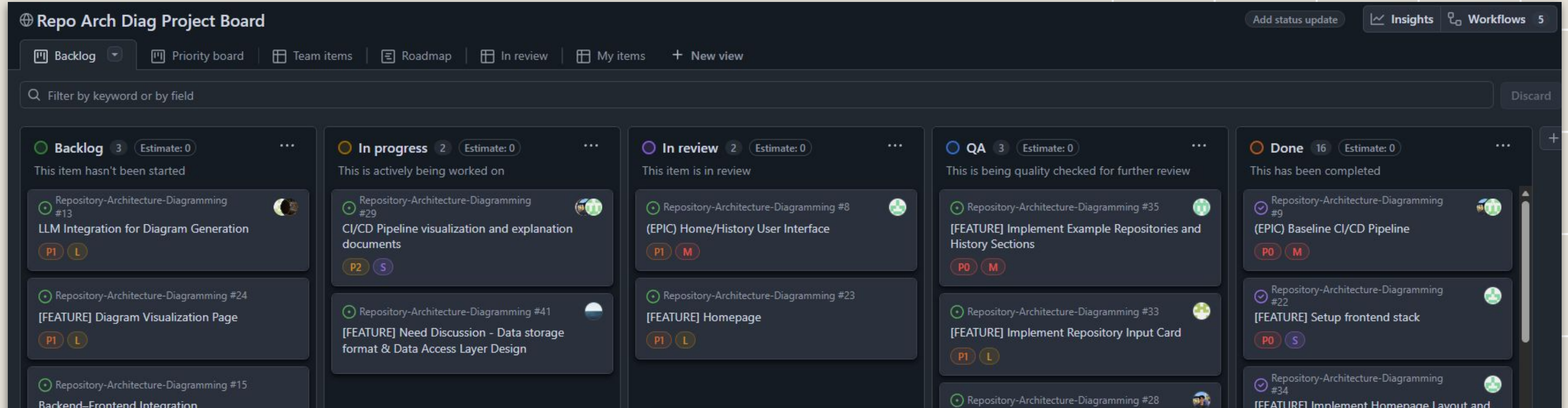
Sprint End

Deliverables:
- Functioning repo systems (CI/CD, file structure, etc.)

- Design and UX documentation

- Stage 1 of MVP (working interface and diagram strategies)

Git



Repo Arch Diag Project Board

Add status update · Insights · Workflows 5

Backlog · Priority board · Team items · Roadmap · In review · My items · + New view

Filter by keyword or by field · Discard

Backlog 3 Estimate: 0
This item hasn't been started

- Repository-Architecture-Diagramming #13
  LLM Integration for Diagram Generation
  P1 L
- Repository-Architecture-Diagramming #24
  [FEATURE] Diagram Visualization Page
  P1 L
- Repository-Architecture-Diagramming #15
  Backend–Frontend Integration
  P1 M

In progress 2 Estimate: 0
This is actively being worked on

- Repository-Architecture-Diagramming #29
  CI/CD Pipeline visualization and explanation documents
  P2 S
- Repository-Architecture-Diagramming #41
  [FEATURE] Need Discussion - Data storage format & Data Access Layer Design

In review 2 Estimate: 0
This item is in review

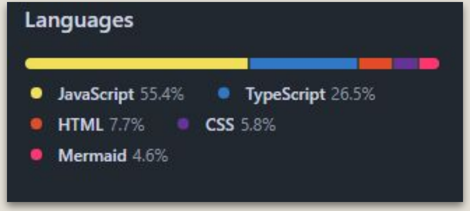
- Repository-Architecture-Diagramming #8
  (EPIC) Home/History User Interface
  P1 M
- Repository-Architecture-Diagramming #23
  [FEATURE] Homepage
  P1 L

QA 3 Estimate: 0
This is being quality checked for further review

- Repository-Architecture-Diagramming #35
  [FEATURE] Implement Example Repositories and History Sections
  P0 M
- Repository-Architecture-Diagramming #33
  [FEATURE] Implement Repository Input Card
  P1 L
- Repository-Architecture-Diagramming #28

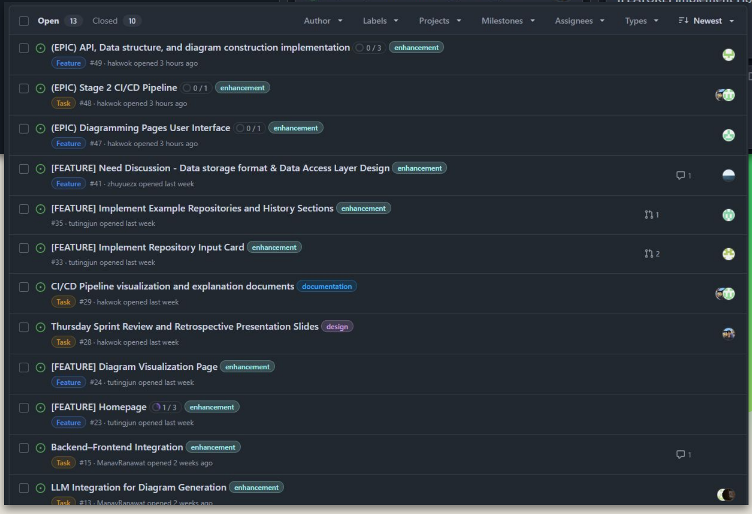
Done 16 Estimate: 0
This has been completed

- Repository-Architecture-Diagramming #9
  (EPIC) Baseline CI/CD Pipeline
  P0 M
- Repository-Architecture-Diagramming #22
  [FEATURE] Setup frontend stack
  P0 S
- Repository-Architecture-Diagramming #34
  [FEATURE] Implement Homepage Layout and
- Diagramming



Languages

- JavaScript 55.4%
- TypeScript 26.5%
- HTML 7.7%
- CSS 5.8%
- Mermaid 4.6%

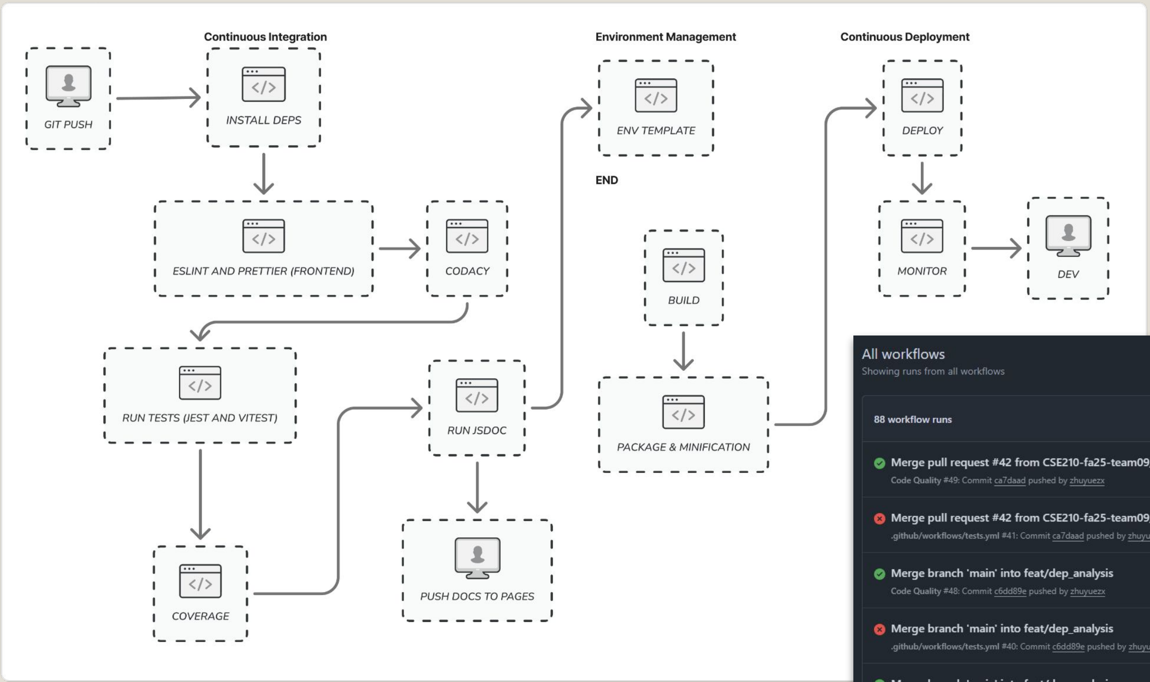


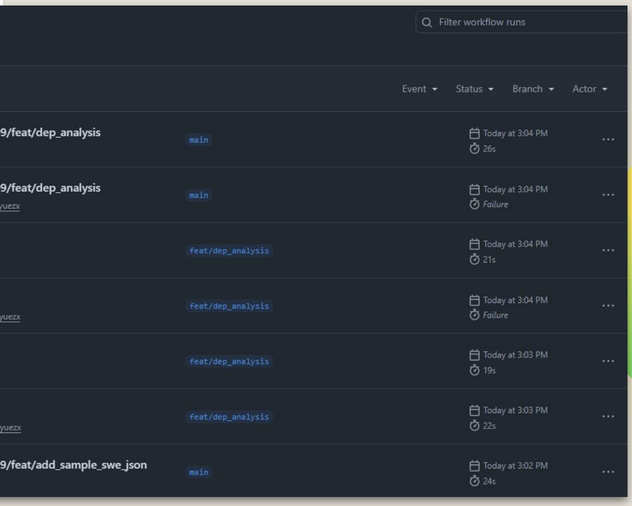
□ Open 13 Closed 10 · Author · Labels · Projects · Milestones · Assignees · Types · Newest

- □ (EPIC) API, Data structure, and diagram construction implementation 0 / 3 enhancement
  Feature #49 · hakwok opened 3 hours ago
- □ (EPIC) Stage 2 CI/CD Pipeline 0 / 1 enhancement
  Task #48 · hakwok opened 3 hours ago
- □ (EPIC) Diagramming Pages User Interface 0 / 1 enhancement
  Feature #47 · hakwok opened 3 hours ago
- □ [FEATURE] Need Discussion - Data storage format & Data Access Layer Design enhancement
  Feature #41 · zhuyuezx opened last week — 1
- □ [FEATURE] Implement Example Repositories and History Sections enhancement
  #35 · tutingjun opened last week — 1
- □ [FEATURE] Implement Repository Input Card enhancement
  #33 · tutingjun opened last week — 2
- □ CI/CD Pipeline visualization and explanation documents documentation
  Task #29 · hakwok opened last week
- □ Thursday Sprint Review and Retrospective Presentation Slides design
  Task #28 · hakwok opened last week
- □ [FEATURE] Diagram Visualization Page enhancement
  Feature #24 · tutingjun opened last week
- □ [FEATURE] Homepage 1 / 3 enhancement
  Feature #23 · tutingjun opened last week
- □ Backend–Frontend Integration enhancement
  Task #15 · ManavRanawat opened 2 weeks ago — 1
- □ LLM Integration for Diagram Generation enhancement
  Task #13 · ManavRanawat opened 2 weeks ago

CI/CD Pipeline

CI/CD Pipeline





All workflows

Showing runs from all workflows

88 workflow runs

Merge pull request #42 from CSE210-fa25-team09/feat/dep_analysis
Code Quality #49: Commit ca7daad pushed by zhuyuezx
main
Today at 3:04 PM
26s

Merge pull request #42 from CSE210-fa25-team09/feat/dep_analysis
.github/workflows/tests.yml #41: Commit ca7daad pushed by zhuyuezx
main
Today at 3:04 PM
Failure

Merge branch 'main' into feat/dep_analysis
Code Quality #48: Commit c6dd89e pushed by zhuyuezx
feat/dep_analysis
Today at 3:04 PM
21s

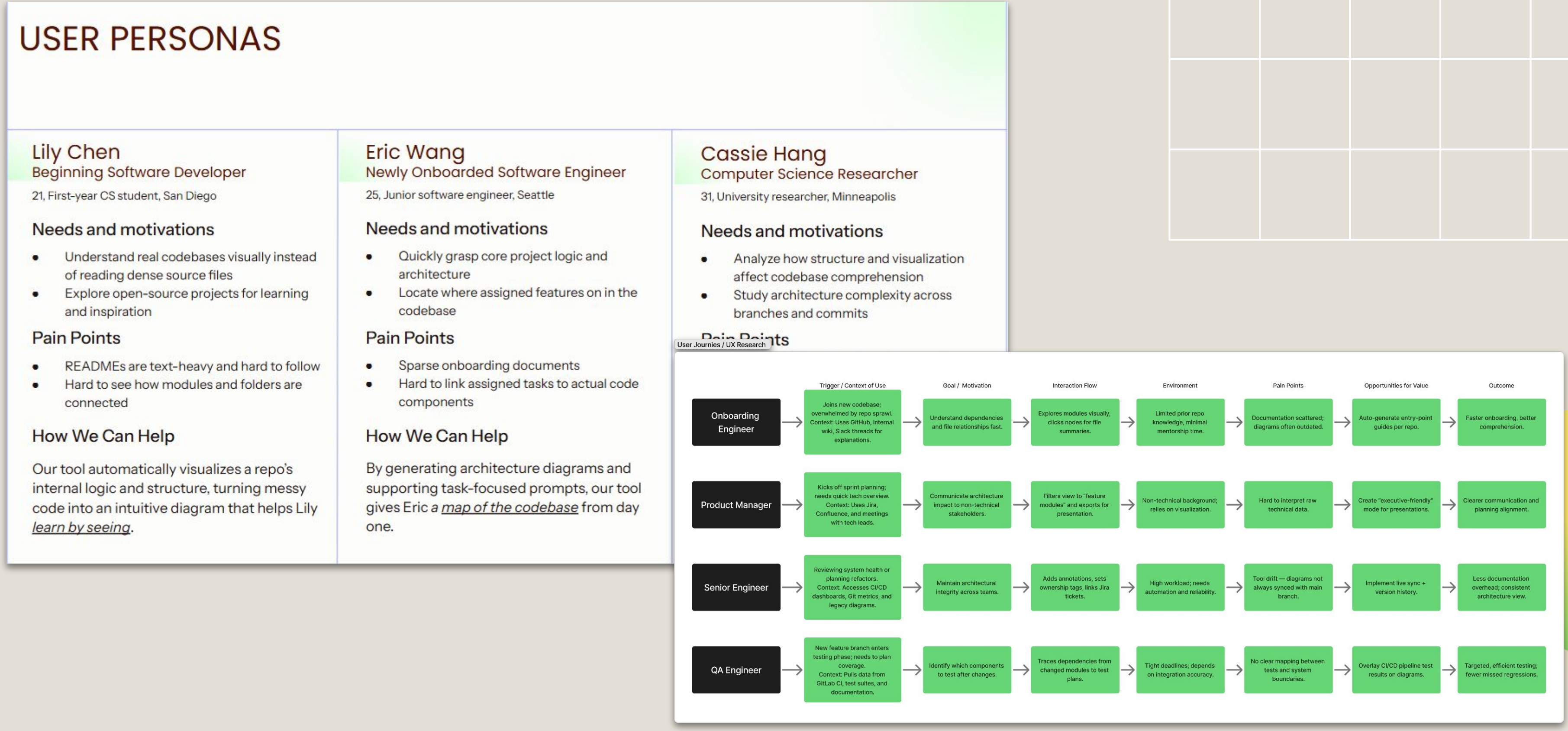
Merge branch 'main' into feat/dep_analysis
.github/workflows/tests.yml #40: Commit c6dd89e pushed by zhuyuezx
feat/dep_analysis
Today at 3:04 PM
Failure

Merge branch 'main' into feat/dep_analysis
Code Quality #47: Commit 3d5be03 pushed by zhuyuezx
feat/dep_analysis
Today at 3:03 PM
19s

Merge branch 'main' into feat/dep_analysis
.github/workflows/tests.yml #39: Commit 3d5be03 pushed by zhuyuezx
feat/dep_analysis
Today at 3:03 PM
22s

Merge pull request #38 from CSE210-fa25-team09/feat/add_sample_swe_json
Code Quality #46: Commit 762280f pushed by hakwok
main
Today at 3:02 PM
24s

UX Research



USER PERSONAS

Lily Chen
Beginning Software Developer

21, First-year CS student, San Diego

Needs and motivations

- Understand real codebases visually instead of reading dense source files
- Explore open-source projects for learning and inspiration

Pain Points

- READMEs are text-heavy and hard to follow
- Hard to see how modules and folders are connected

How We Can Help

Our tool automatically visualizes a repo's internal logic and structure, turning messy code into an intuitive diagram that helps Lily *learn by seeing*.

Eric Wang
Newly Onboarded Software Engineer

25, Junior software engineer, Seattle

Needs and motivations

- Quickly grasp core project logic and architecture
- Locate where assigned features on in the codebase

Pain Points

- Sparse onboarding documents
- Hard to link assigned tasks to actual code components

How We Can Help

By generating architecture diagrams and supporting task-focused prompts, our tool gives Eric a *map of the codebase* from day one.

Cassie Hang
Computer Science Researcher

31, University researcher, Minneapolis

Needs and motivations

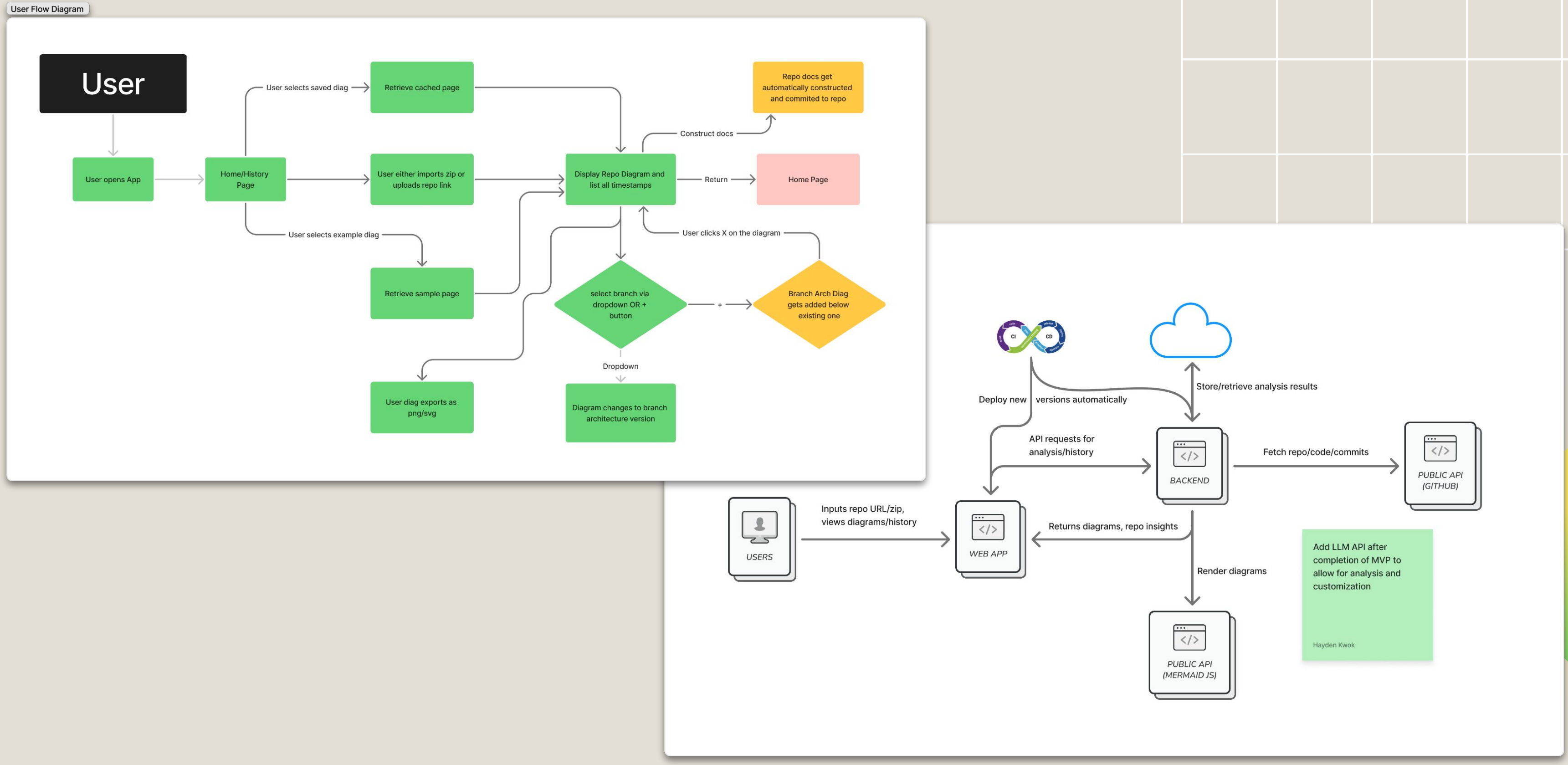
- Analyze how structure and visualization affect codebase comprehension
- Study architecture complexity across branches and commits

Pain Points

User Journies / UX Research

	Trigger / Context of Use	Goal / Motivation	Interaction Flow	Environment	Pain Points	Opportunities for Value	Outcome
Onboarding Engineer	Joins new codebase; overwhelmed by repo sprawl. Context: Uses GitHub, internal wiki, Slack threads for explanations.	Understand dependencies and file relationships fast.	Explores modules visually, clicks nodes for file summaries.	Limited prior repo knowledge, minimal mentorship time.	Documentation scattered; diagrams often outdated.	Auto-generate entry-point guides per repo.	Faster onboarding, better comprehension.
Product Manager	Kicks off sprint planning; needs quick tech overview. Context: Uses Jira, Confluence, and meetings with tech leads.	Communicate architecture impact to non-technical stakeholders.	Filters view to "feature modules" and exports for presentation.	Non-technical background; relies on visualization.	Hard to interpret raw technical data.	Create "executive-friendly" mode for presentations.	Clearer communication and planning alignment.
Senior Engineer	Reviewing system health or planning refactors. Context: Accesses CI/CD dashboards, Git metrics, and legacy diagrams.	Maintain architectural integrity across teams.	Adds annotations, sets ownership tags, links Jira tickets.	High workload; needs automation and reliability.	Tool drift — diagrams not always synced with main branch.	Implement live sync + version history.	Less documentation overhead; consistent architecture view.
QA Engineer	New feature branch enters testing phase; needs to plan coverage. Context: Pulls data from GitLab CI, test suites, and documentation.	Identify which components to test after changes.	Traces dependencies from changed modules to test plans.	Tight deadlines; depends on integration accuracy.	No clear mapping between tests and system boundaries.	Overlay CI/CD pipeline test results on diagrams.	Targeted, efficient testing; fewer missed regressions.

Flowcharts



User Flow Diagram

User

User opens App

Home/History Page

User selects saved diag

Retrieve cached page

User either imports zip or uploads repo link

User selects example diag

Retrieve sample page

Display Repo Diagram and list all timestamps

Construct docs

Repo docs get automatically constructed and commited to repo

Return

Home Page

User clicks X on the diagram

select branch via dropdown OR + button

+

Branch Arch Diag gets added below existing one

Dropdown

Diagram changes to branch architecture version

User diag exports as png/svg

CI

CD

Deploy new versions automatically

Store/retrieve analysis results

API requests for analysis/history

BACKEND

Fetch repo/code/commits

PUBLIC API (GITHUB)

USERS

Inputs repo URL/zip, views diagrams/history

WEB APP

Returns diagrams, repo insights

Render diagrams

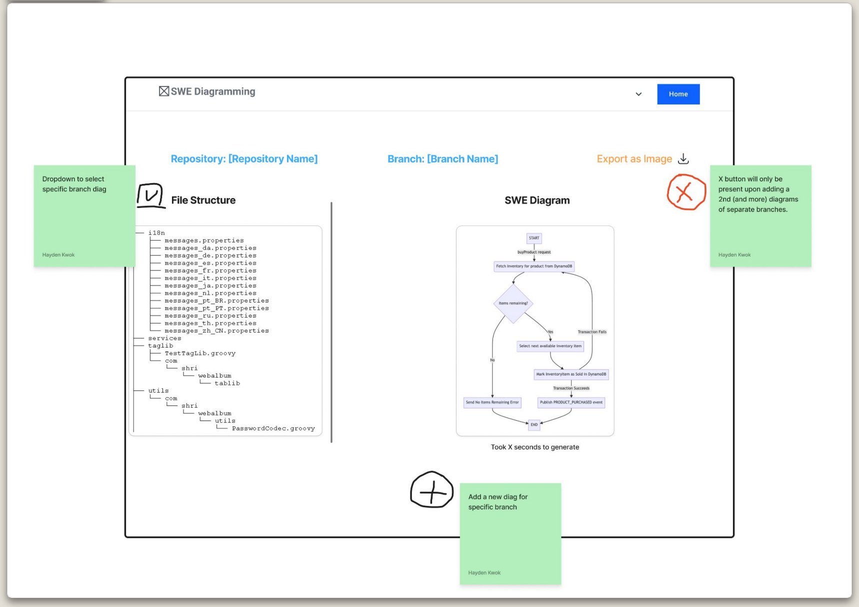
PUBLIC API (MERMAID JS)

Add LLM API after completion of MVP to allow for analysis and customization

Hayden Kwok

Initial Wireframes

Wireframe Diagram Page



☒SWE Diagramming

Home

Repository: [Repository Name]

Branch: [Branch Name]

Export as Image

Dropdown to select specific branch diag

Hayden Kwok

File Structure

```
── i18n
   ├── messages.properties
   ├── messages_da.properties
   ├── messages_de.properties
   ├── messages_es.properties
   ├── messages_fr.properties
   ├── messages_it.properties
   ├── messages_ja.properties
   ├── messages_nl.properties
   ├── messages_pt_BR.properties
   ├── messages_pt_PT.properties
   ├── messages_ru.properties
   ├── messages_th.properties
   └── messages_zh_CN.properties
── services
── taglib
   ├── TestTagLib.groovy
   └── com
       └── shri
           └── webalbum
               └── tablib
── utils
   └── com
       └── shri
           └── webalbum
               └── utils
                   └── PasswordCodec.groovy
```

SWE Diagram

START
buyProduct request
Fetch inventory for product from DynamoDB
Items remaining?
Yes
No
Transaction Fails
Select next available inventory item
Mark InventoryItem as Sold in DynamoDB
Transaction Succeeds
Send No Items Remaining Error
Publish PRODUCT_PURCHASED event
END

Took X seconds to generate

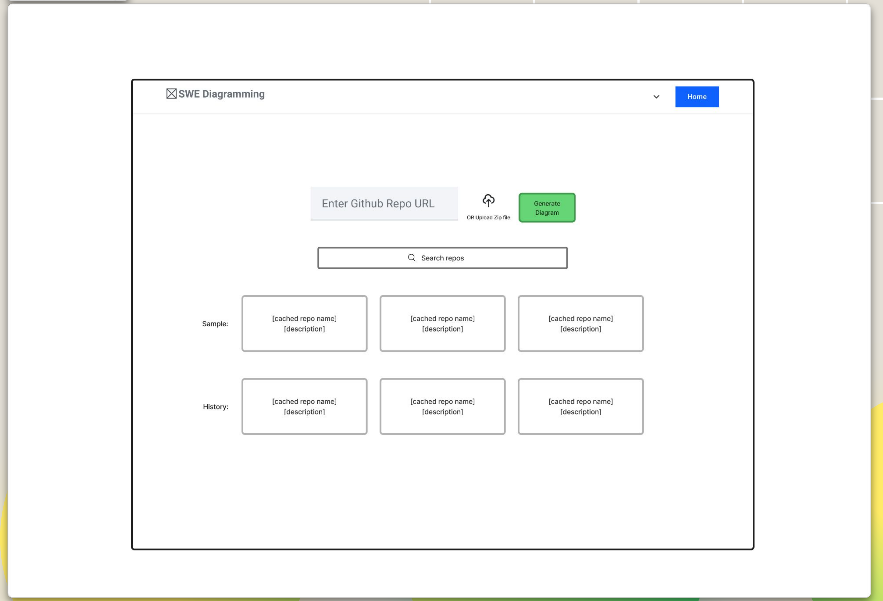
X button will only be present upon adding a 2nd (and more) diagrams of separate branches.

Hayden Kwok

Add a new diag for specific branch

Hayden Kwok

Wireframe Home/History Page



☒SWE Diagramming

Home

Enter Github Repo URL

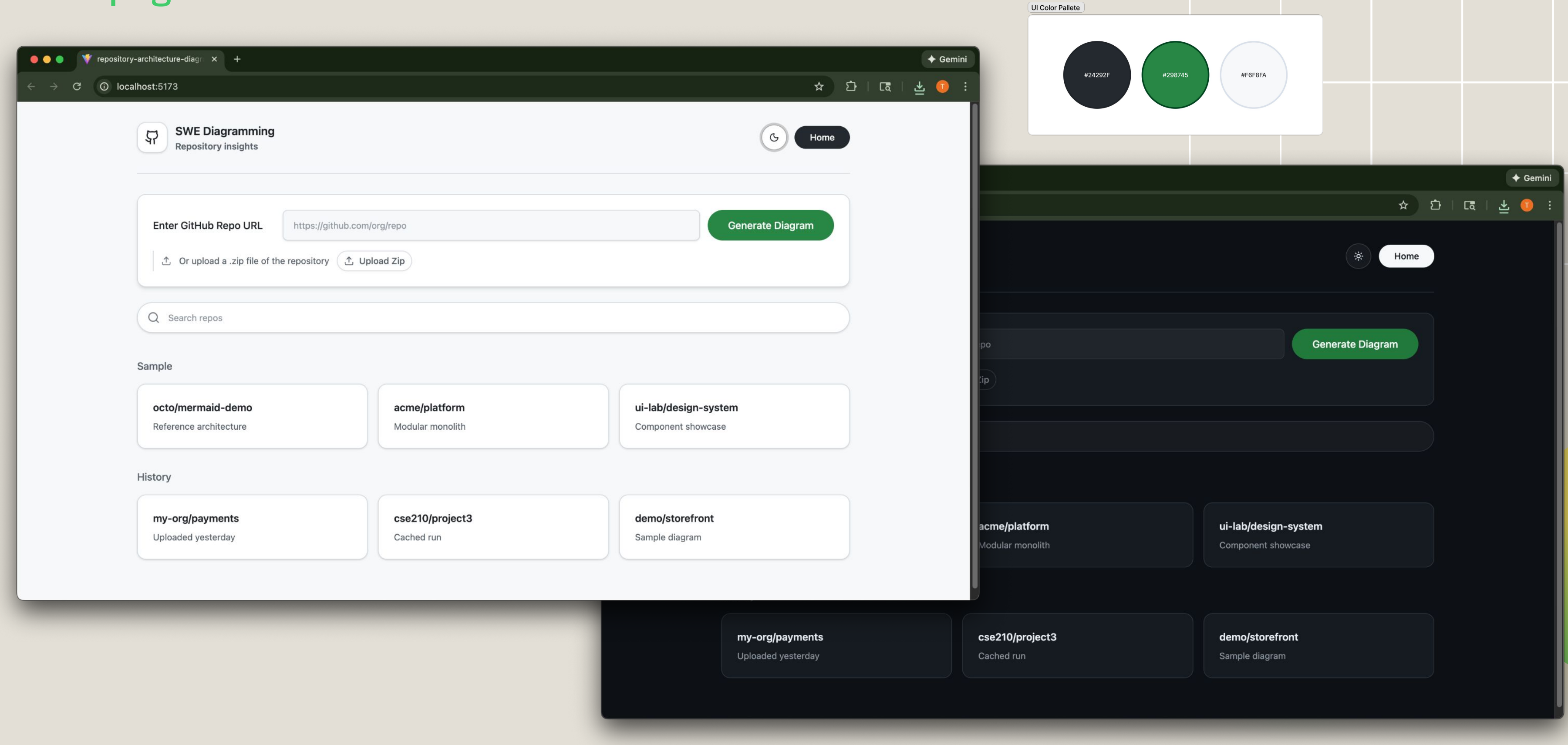
OR Upload Zip file

Generate Diagram

Search repos

Sample:	[cached repo name] [description]	[cached repo name] [description]	[cached repo name] [description]
History:	[cached repo name] [description]	[cached repo name] [description]	[cached repo name] [description]

Homepage UI



SWE Diagramming
Repository insights

Home

Enter GitHub Repo URL

https://github.com/org/repo

Generate Diagram

Or upload a .zip file of the repository

Upload Zip

Search repos

Sample

octo/mermaid-demo Reference architecture	acme/platform Modular monolith	ui-lab/design-system Component showcase

History

my-org/payments Uploaded yesterday	cse210/project3 Cached run	demo/storefront Sample diagram

Home

po

Generate Diagram

ip

acme/platform
Modular monolith

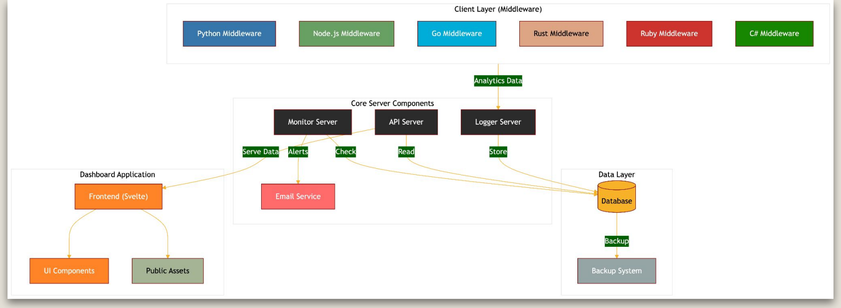
ui-lab/design-system
Component showcase

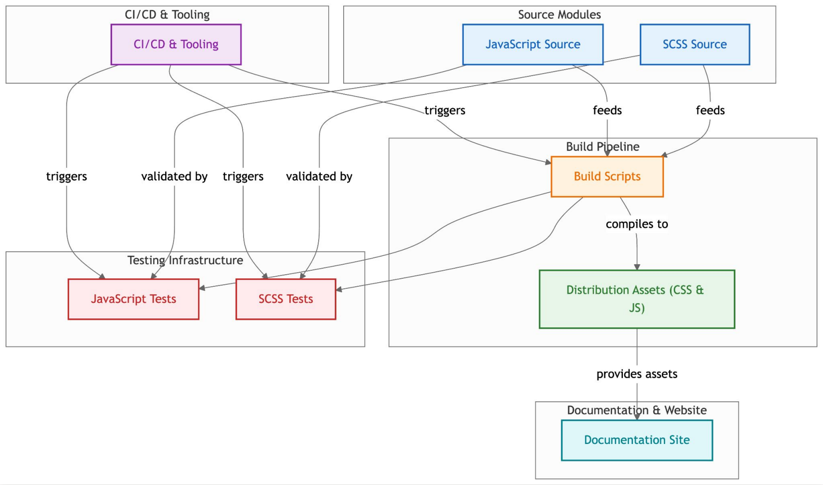
my-org/payments
Uploaded yesterday

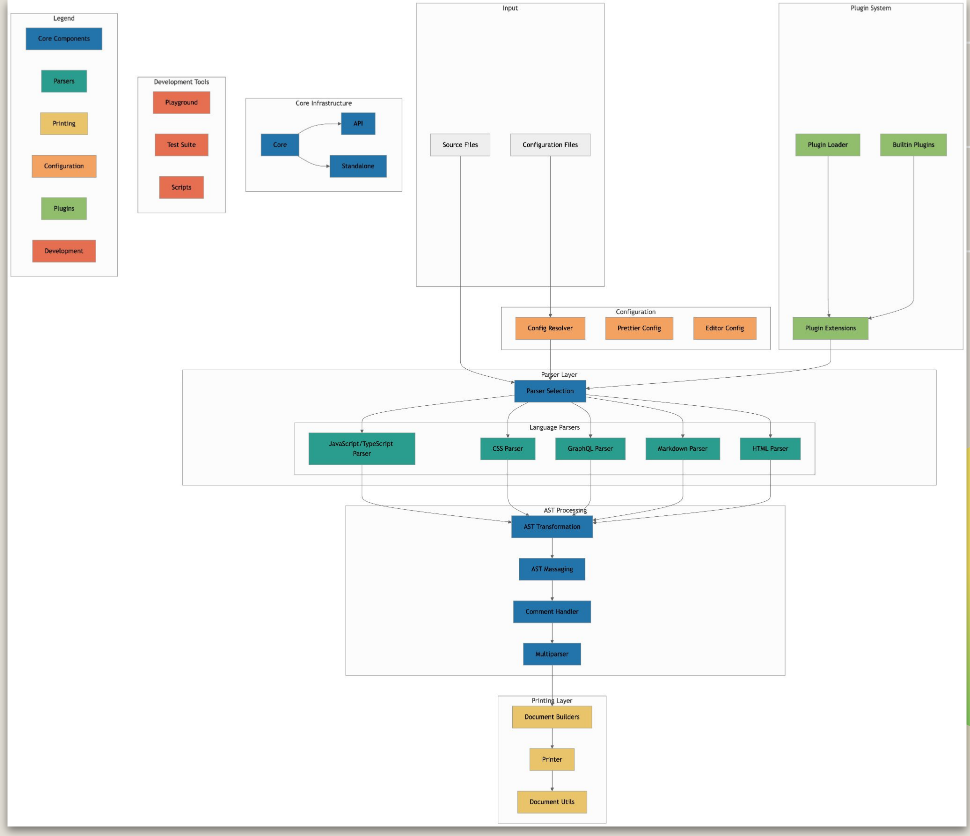
cse210/project3
Cached run

demo/storefront
Sample diagram

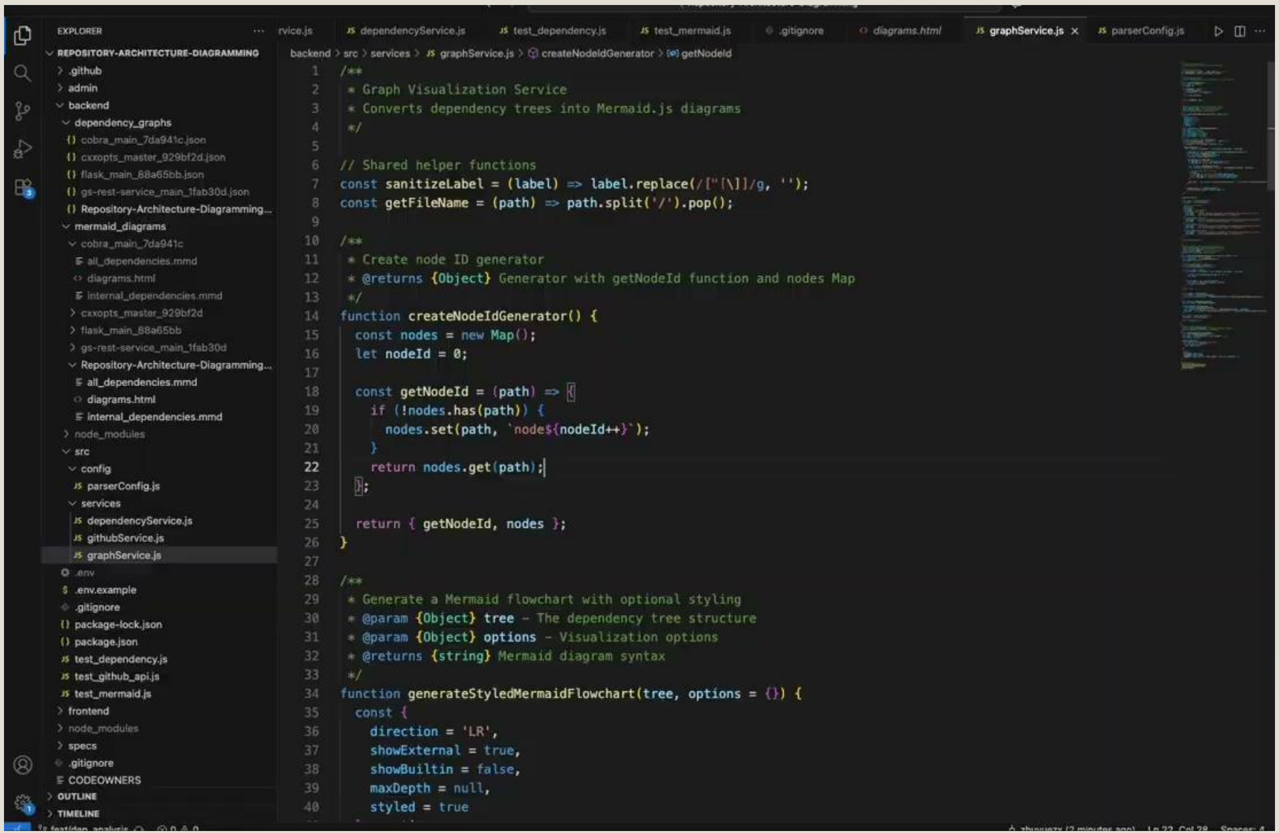
Sample Diagram Structures



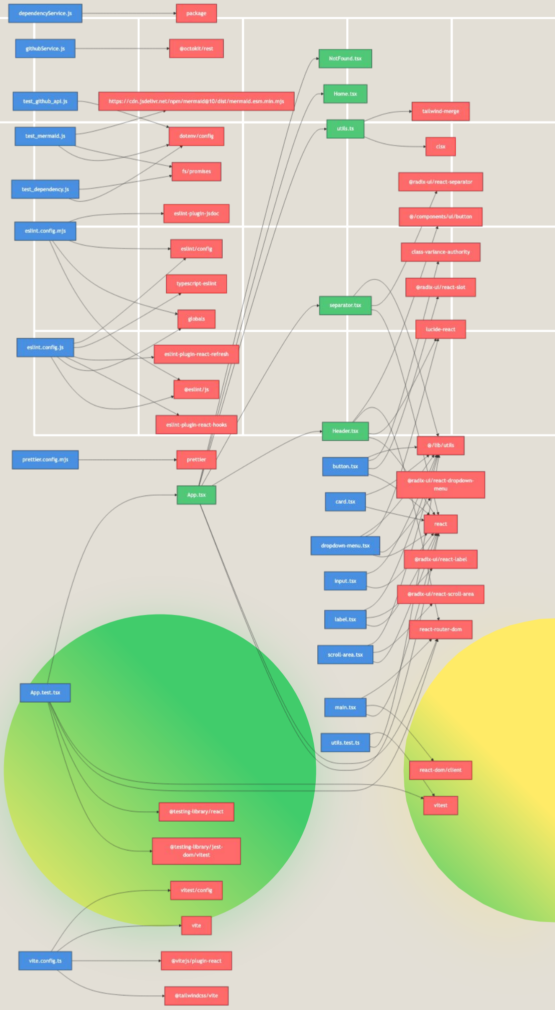




Backend Workspace



```javascript
/**
 * Graph Visualization Service
 * Converts dependency trees into Mermaid.js diagrams
 */

// Shared helper functions
const sanitizeLabel = (label) => label.replace(/["[\]]/g, '');
const getFileName = (path) => path.split('/').pop();

/**
 * Create node ID generator
 * @returns {Object} Generator with getNodeId function and nodes Map
 */
function createNodeIdGenerator() {
  const nodes = new Map();
  let nodeId = 0;

  const getNodeId = (path) => {
    if (!nodes.has(path)) {
      nodes.set(path, `node${nodeId++}`);
    }
    return nodes.get(path);
  };

  return { getNodeId, nodes };
}

/**
 * Generate a Mermaid flowchart with optional styling
 * @param {Object} tree - The dependency tree structure
 * @param {Object} options - Visualization options
 * @returns {string} Mermaid diagram syntax
 */
function generateStyledMermaidFlowchart(tree, options = {}) {
  const {
    direction = 'LR',
    showExternal = true,
    showBuiltin = false,
    maxDepth = null,
    styled = true
```



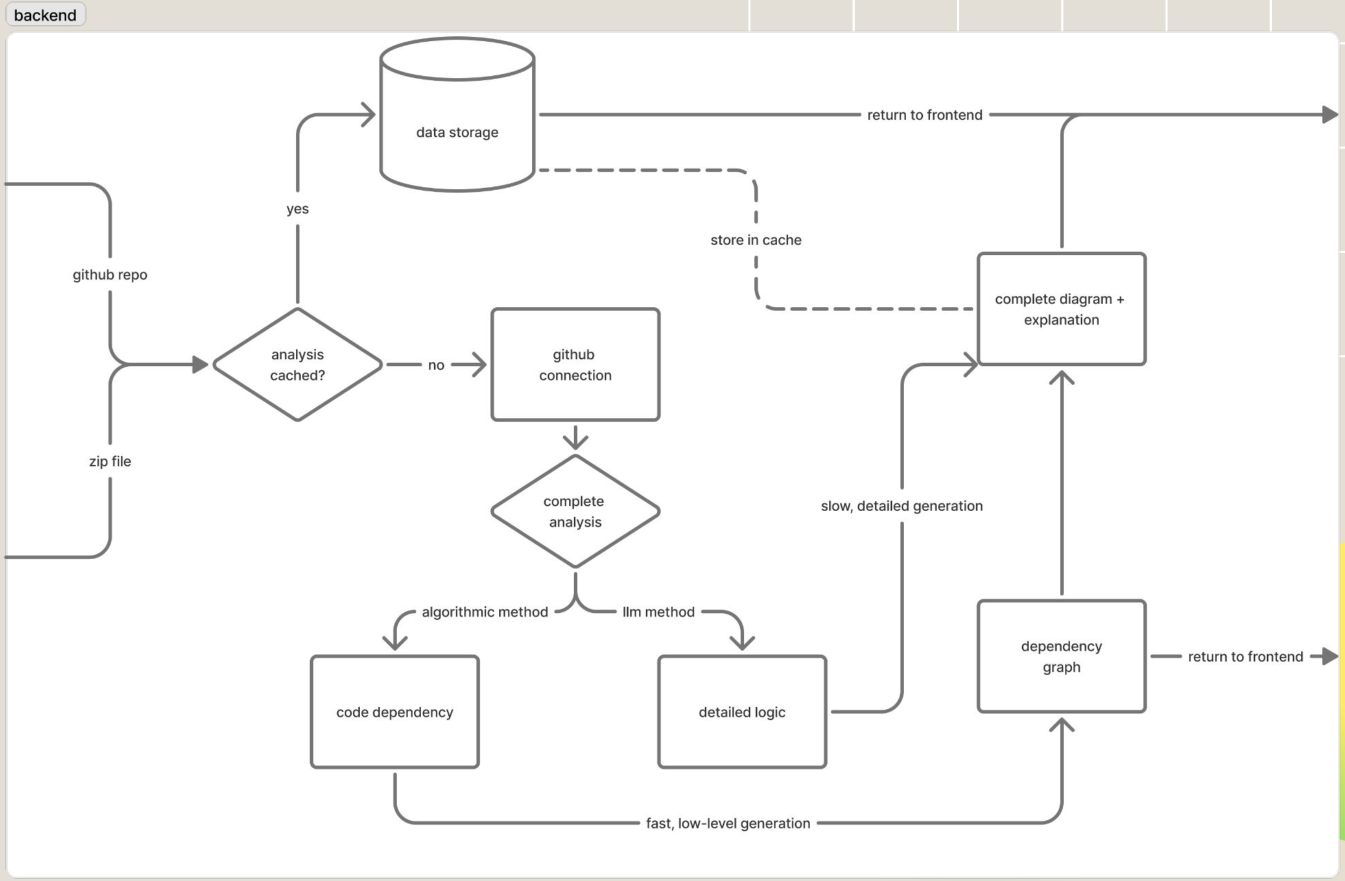
Backend Architecture

Core idea:

- Algorithmic method for fast response
- LLM for comprehensive analysis
- Currently exploring way to integrate both

Current Progress:

- Algorithmic approach ready
- Tuning LLM approach
- Setting up backend server



Future iterations and sprint goals

Sprint #2 : 11/17 - 11/21

Frontend: Complete diagram page UI, connect to diagramming strategy to UI output.

Backend: Finalize development on strategy integration

Deployment: Complete pipeline

Sprint #3 : 11/24 - 11/28

Extend beyond MVP requirements and iterate on the web app from being just visualization tool to interactive tool (modification, integrations, etc.)

What went well

1

Team structure

The team is split up into frontend, backend, and deployment with an inherent cross-functioning responsibility expectation. Everyone has shown to work nicely in their subgroups and beyond as a result of strong forming/norming.

2

Smooth and efficient decision making for ADRs and Design

As a result of subgroup leadership, smaller subteams allow for quicker decisions with fewer required approvals and quicker discussion.

3

Iterative and Adaptable practices

Although decisions are made swift and easy as a result of the team structure, the concept of following two-way technical doors and starting from a low level of complexity allows the software to iterate and adapt with less sacrifice

What did not go so well

1 **Sticking to timeline and procrastination**

Many tickets were left idle until later in the sprint, which influences project management decisions such as the need to extend sprints or reducing scope of the project complexity.

2 **Workshare / Dependency**

Although the subteam structure has worked out nicely for decision making and management, the heightened responsibilities of leads have allowed others to have a reduced role, disregarding the goal of event participation.

3 **Attendance in meetings**

It has been hard to find meeting times that accommodate all members, which leaves out important voices during discussions about ADRs and design.

Addressing the challenges in future sprints

How we'll address or avoid this challenge

Timeline

Assign more concrete due dates in addition to priority markers to tickets within the sprint timeline to streamline urgency.

Workshare

Potentially set up a more measures of contribution:

- Story points
- Ticket completions
- Feedback loops

Attendance

Streamline delegation of information by automatically dispersing documentation discussed to those that could not attend.

Any Feedback?
Please Share!

Thank you