PROJECT ABSTRACT

Git Commit History Visualizer

A web app rendering Git commit history as an interactive branch graph with commit diffs and contributor analytics. Delivered built-to-order with full source, sample dataset, ingestion script, report, PPT and viva kit.

---

01. Title

Git Commit History Visualizer

02. Introduction / Background

`git log` is a wall of text: team members can see that commits exist but cannot see how branches relate, where merges happened, or who wrote what. A visualiser that parses the commit graph and renders branches as colour-coded lanes — commits as nodes, merges as curves — makes history explorable: click any node for the full commit with diff stats, and open the contributor panel for commits, line churn, merge activity and hotspot files. Pure SVG rendering keeps it dependency-free and precisely controllable.

03. Problem Statement

Textual git history hides branch structure and authorship patterns from students working in teams. The project addresses this with a web app that lays out the commit graph as an interactive diagram, shows per-commit diffs on click, aggregates contributor analytics, and ingests any real repository via a documented `git log` export format.

04. Objectives

- Render commits, branches and merges as an interactive SVG branch graph.
- Show commit detail on click: message, author, SHA, parent, file stats and diff hunks.
- Aggregate contributor analytics: commits, lines added/removed, share of repo.
- Report repository pulse: totals, branches, merged/open PRs, hotspot files, merge timeline.
- Ship a realistic sample team dataset (5 contributors, 4 branches, 3 merged PRs).
- Deliver the complete student kit: source, ingestion script, report, PPT and viva Q&A.;

05. Proposed Methodology

Commit data (SHA, parents, author, timestamp, message, file stats) loads from the bundled JSON dataset or the ingestion script's output. A layout algorithm assigns each commit a branch lane and an x-position by topological order and draws merge curves between lanes. Hand-rolled SVG renders nodes, lanes, curves and labels with click handlers for the detail view. An aggregation pass computes per-author churn, repository totals and hotspot files for the analytics panels.

06. System Architecture / Working

The dataset loads into the graph module, which topologically orders commits, assigns lanes per branch and emits SVG: lane lines, commit nodes, bezier merge curves and labels. Clicking a node opens the detail panel with metadata, per-file add/del stats and highlighted diff hunks. The analytics module aggregates authors and repository totals. The ingestion script converts `git log --pretty` output into the documented JSON schema for real repositories.

07. Hardware / Software Requirements

- Any modern browser (design target)
- HTML5, CSS3, JavaScript ES6, SVG
- Python 3 (git-log ingestion script)
- Git (for ingesting real repositories)

08. Expected Outcomes

- An interactive branch graph for a realistic team repository.
- Click-through commit detail with diff hunks.
- Contributor and repository analytics panels.
- A complete report, PPT and viva Q&A; on graph layout and Git internals.

09. Applications

- Exploring team-project history visually.
- Teaching graph algorithms and Git internals.
- Demonstrating dependency-free SVG data visualisation.

10. Future Scope

- Live repository watching with automatic re-render.
- Blame heat-maps overlaid on the file tree.
- 10,000+ commit datasets via pre-aggregated layout.
- Conflict-prediction hints from branch divergence.