SP-202 Agentic Software Debugger & Documenter

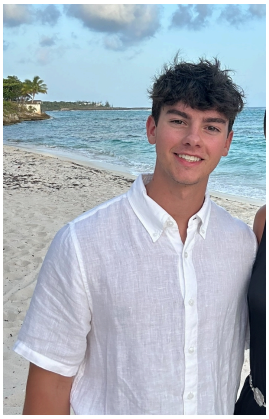
Project Plan

CS 4850 — Sec 02 — Spring 2026

January 23, 2026

Sharon Perry

Jade Le
Team Lead
Documentation & Testing



Preston Dietz
Development & Testing

Omodemi Olaoluwa
Development & Testing

Team Members:

Name	Role	Cell Phone / Alt Email
Jade Le (Team Lead)	Documentation & Test	818.270.8979 jle12@students.kennesaw.edu
Preston Dietz	Development & Test	678-997-7713 pdietz1@students.kennesaw.edu
Olaoluwa Omodemi	Development & Test	470-266-9850 oomodem2@students.kennesaw.edu
Sharon Perry	Project Owner / Advisor	770.329.3895 Sperry46@kennesaw.edu

Project Overview

This project implements a multi-agent code analysis system using Python and the Google Agent Development Kit. The system employs SequentialAgent and LoopAgent patterns to automate code quality workflows. When a developer submits a code snippet, three specialized agents work in sequence, the first agent analyzes the code to identify potential bugs, the second agent automatically generates fixes for detected issues, and the third agent produces documentation. A LoopAgent pattern ensures code quality by validating fixes through simulated linter checks. If validation fails, the code is automatically routed back to the fixing agent for refinement. This iterative approach demonstrates how intelligent agents can collaborate to enhance code quality and developer productivity.

Project Website

(temporary link) https://github.com/Agentic-debugger

Deliverables

Requirements

1. Project requirements summary (scope, inputs/outputs, etc.)
2. Agent architecture design (description and design of workflow)

Design & Implementation

3. Analyst agent implementation
4. Fixer LoopAgent implementation
5. Static analysis (linter) integration
6. Documentation agent implementation

Testing

7. Test code samples
8. Linter validation reports
9. Safety and loop control mechanisms

Documentation (Final Report Package)

10. GitHub repo organization
11. Project website
12. Final project report
13. Video presentation

Group Meeting Schedule Date/Time

Mondays @ 1:00pm

Collaboration and Communication Plan

Teams (weekly status meetings) and Discord

Project Schedule and Task Planning

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	**Project Name:**	Agentic SWD																		
2	**Report Date:**	1/25/2026																		
3						*Milestone #1*				*Milestone #2*				*Milestone #3*				C-Day		
4	**Phase**	**Tasks**	**Complete%**	**Current Status Memo**	**Assigned To**	01/25	02/01	02/08	02/15	02/22	03/01	03/08	03/15	03/22	03/29	04/05	04/12	04/19	04/26	05/03
5	Requirements	define scope & constraints	0%		TBD	3	3	3												
6		define functional requirements	0%		TBD	4	4	4												
7		define non-func requirements	0%		TBD	3	3	3												
8		define system I/O & agent roles	0%		TBD	4	4	4												
9		write and review SRS	0%		TBD	3	5	5												
10	Project design	design system architecture	0%		TBD	2	2	2												
11		design SequentialAgent flow	0%		TBD	3	3	3												
12		design LoopAgent logic	0%		TBD	3	3	3												
13		design static analysis integration	0%		TBD	3	3	3												
14		write and review SDD	0%		TBD	3	5	5												
15	Development	implement Analyst agent	0%		TBD				3	3	3	3	3							
16		implement Fixer LoopAgent	0%		TBD				3	3	3	3	3							
17		implement static analysis integrat	0%		TBD					3	3	3	3	3	3					
18		implement Documentation agent	0%		TBD					3	3	3	3	3	3					
19		write development doc	0%		TBD					3	3	3								
20		prep test code samples	0%		TBD						2	2	2	2	2					
21		write STP & STR	0%		TBD								2	2	2	2	2			
22		start testing/validating system	0%		TBD						4	4	4	4	4	4	4			
23		validate loop safety & error handl	0%		TBD						3	3	3	3	3	3				
24	Final report	validate system behavior	0%		TBD											2	2	2	2	2
25		write final project report	0%		TBD											3	3	3	3	3
26		finalize website & repo	0%		TBD											3	3	3	3	3
27		prep video & presentation	0%		TBD											4	4	4	4	4
28		Poster preparation	0%		TBD													3	3	3
29		Final report submission to D2L	0%		TBD														5	3
30																				
31				**Total work hours**	319	31	35	35	6	15	24	24	23	17	17	21	18	15	20	18
32																				
33		* formally define how you will develop this project including source code manager																		
34																				
35	*Legend*																			
36	Planned																			
37	Delayed																			
38	Number	Work: man hours																		

Version Control Plan

This project will be making use of Git as the version control system due to its industry-standard adoption, flexibility and tooling support. The repository is hosted within a GitHub Organization, which supports collaboration, access control, and industry-style project management.

Repository Structure

- Agents/ - individual agent implementations
  - Bug Detector agent
  - Fixer Agent
  - Documentation Agent
  - LoopAgent Logic
- Linting/ - Simulated Linter rules and validations checks
- Orchestration/ - sequential execution and control flow
- Tests/ - unit and integration tests
- docs/ - system and agent documentation

This structure enables independent development and testing of every agent.

Branching

This project will be following Git branching mode:

- Main
    - Stable tested code
    - Represents submission ready and demonstration versions
- Feature branches
    - Used for developing or refining individual agents or behaviours
- Experimental branches
    - Used to test alternative agent logic or loopagent retry strategies
    - Safely isolated from stable code

Commits and Collaboration

- Commits are small, frequent and descriptive
- Commit messages clearly document agent behaviour changes and system logic
- Pull requests are used to review changes before merging into main

Version and Milestones

Project versions and major milestones are marked using Git Tags, making it easy to reference, reproduce and evaluate specific system states

By using Git within a dedicated GitHub Organization, this project adopts industry-standard version control practices while remaining appropriate for an academic setting. The approach supports collaboration, iterative agent refinement, and long-term maintainability, all of which are critical for a SequentialAgent and LoopAgent-based system.