Team Members

Sipho Sehlapelo

Owethu Dlamini

Kelsey Hamann

Sibusiso Mngomezulu

David Musa-Aisien

Team Roles

David

Currently serving as the **Documentation Specialist**, **Testing Engineer**, and partially as an **Architect** and **DevOps Engineer**.

- **Documentation Specialist**
  Responsible for accurately organizing team requirements, project specifications, documenting project progress, and maintaining standards for Git commits.
- **Testing Engineer**
  Oversees all testing activities, including unit testing, integration testing, end-to-end (E2E) testing, and acceptance testing. Also acts as a **DevOps** engineer responsible for setting up and maintaining the CI/CD pipeline on the Git repository.
- **Architect (Partial)**
  Contributes to the structuring of the codebase to ensure modularity and loose coupling.
- **Designer**
  Responsible for the overall user experience and visual design of the application. Works with the project manager on all components related to the frontend.

Sipho

Currently serving as the **Project Manager**, **UI Engineer**, and **Designer**.

- **Project Manager**
  Leads the project. Responsible for organizing the team, allocating tasks, and ensuring time management. Assists any members who are struggling, communicates with the Project Owner, and may assist with documentation.
- **UI Engineer**
  Responsible for the overall user experience and visual presentation of the system. Works closely with the backend team. Documents the code and communicates changes with the designer and backend.

- **Designer**
  Collaborates with the architect and frontend team. Documents and signs off on the design and any changes throughout the project.

Kelsey

Currently serving on the backend as the **Integration Engineer**, and partially as a **Data Engineer**, **Testing Engineer**, and **Architect**.

- **Integration Engineer**
  Primarily responsible for API development. Ensures proper data transfer and formats between components.
- **Data Engineer**
  Sets up and configures the database using open-source technologies.
- **Testing Engineer**
  Collaborates with the co-testing engineer to define the acceptable formats for data requests and test values.
- **Architect (Partial)**
  Contributes to modular and loosely coupled code structuring.

Sibusiso

Currently serving on the backend team as a Data Engineer, with major contributions to Service Engineering**.**

- **Data Engineer**
  Handles the scraping and management of information from repositories and databases.
- **Services Engineer**
  Develops the core functionality of the backend based on information received through the handling of GIT repositories. Responsible for processing this information and feeding it back to and working with integration engineers.

Group Practices

- Rather than assigning all design and technical decisions to individual roles, major decisions were made collectively during in-person team meetings.
- Given our small team size and the need for iterative development, we adopted an Agile development process**.**

- To implement this effectively, we chose the SCRUM methodology, which emphasizes well-defined roles, short development cycles, and frequent feedback loops. This ensures alignment with project goals and fosters continuous improvement.

SCRUM Plan

Roles

- **Product Manager:** Sipho
- **Scrum Master:** David
- **Development Team:** Sipho, Kelsey, Sibusiso, Owethu, David

Process

1. **Backlog Prioritization**
    - Gather requirements and user stories.
    - Prioritize by business value and dependencies.
2. **Sprint Planning**
    - Define the sprint goal.
    - Select and estimate backlog items.
    - Break each item into actionable tasks.
3. **Daily Scrum**
    - 15-minute stand-ups:
        1. What did I do yesterday?
        2. What will I do today?
        3. What challenges do I have and do I require assistance?
        4. (Length, timing, and frequency can be adjusted as needed.)
4. **Sprint Execution**
    - Team delivers on their tasks.
    - Scrum Master removes blockers.
5. **Sprint Review & Retrospective**
    - **Review:** Demo completed work to stakeholders.
    - **Retrospective:** Identify “What went well,” “What didn’t,” and “Action items.”

Project Plan

Sprint 1 (4 weeks)

Goal: Deliver a working skeleton of front-end and back-end, plus three core use cases.
Deliverables:

- **UI/UX:** Profile view & avatar update
- **Team Management:** Search, create, and join teams
- **Data Collection:** Pull data from GitHub via Octokit
- **Documentation:** High-level functional & non-functional requirements
- **CI/CD Plan:** Pipeline design documented in Git

Success Criteria:

- All three use cases execute without errors
- CI/CD pipeline is architected (even if not fully implemented)
- Weekly client demos and feedback sessions held

Possible Difficulties:

- **Mocking external services:** Unit and integration tests will require stubbing GitHub APIs and CI/CD endpoints.
- **Early CI/CD adoption:** Spinning up a robust pipeline this early may divert effort from core features.

Sprint 2 (4 weeks)

Goal: Complete at least eight total use cases (the original three plus five new ones), finalize CI/CD, and refine requirements.

Additional Use Cases:

- The system should be able to display basic repository info: usernames of contributors, number of contributors, languages etc.
- Implement permission based control.
- Implement a basic repository dashboard: A working dashboard showing DORA data
- Implement an improved Change Failure Rate Calculation.
- System should be able to display a trend of Change Failure Rate.
- AI assisted explanation for a metric using rule based logic.

Deliverables:

- Implementation of the five additional use cases

- CI/CD pipeline fully configured and running (build, test and deploy)
- Revised and detailed functional/non-functional requirements document
- Technology decision log shared with the client

Success Criteria:

- All eight use cases (the original three plus the five newly prioritized) execute end-to-end without errors.
- Each use case has an associated acceptance test that passes in the CI/CD pipeline.
- The client has reviewed and signed off on the key technology decisions.

Possible Difficulties:

- **Time frame:** There is restrictive time frame for the amount of required progress. Careful allocation of resources and prioritization of use cases will be observed.

Branching Strategy

An adjusted/custom Git flow strategy will be used:

GIT FLOW BRANCH	PURPOSE
main	Stable and production ready code, will merge from release or the hotfix branch
develop	This branch will be used mostly for integration purposes and will often have the newest changes to the system that may not yet be ready
feature	This branch will be used specially for our system to store each decoupled module and section of our code. We will have a domain for each module such as a UI domain, API domain, JS-server domain, Testing domain etc. The following convention will be used: feature/UI domain/<related code and directories>
release	Likely will be used towards the end of the system development as the final QA tested code
bugfixes/hotfixes	Used for fixes when unexpected issues arises in the code. Will be merged into dev and or main branches. Bugfix will be used for non-critical bugs while hotfix will be used for urgent bugs.