MiniGit System - Project Report

📋 1. Data Structures Used

- **Linked Lists**: Track the sequence of commits.

- **Trees**: Represent directory structures in each commit.

- **Hash Maps/Dictionaries**: Fast lookup of commit metadata, file versions, etc.

- **Strings**: Store user inputs like commit messages and file paths.

🏗️ 2. Design Decisions

- Designed with a **modular architecture**:

  - `main.py` — Entry point & CLI handler

  - `repository.py` — Initializes and manages the repository

  - `staging.py` — Handles staging area logic

  - `commit.py` — Creates and links commit objects

  - `filetree.py` — Stores snapshots of directory structure

- Chose to use **plain text** for metadata to keep the system transparent and beginner-friendly.

- CLI mimics basic Git commands to make usage intuitive.

🐞 3. Limitations

- No support for merging or branching.

- No user authentication or remote push/pull functionality.

- Binary files and large files are not efficiently handled.

- Command parsing is basic and lacks comprehensive error handling.

🔮 4. Future Improvements

- Add **branching** and **merge conflict resolution**.

- Implement a **remote push/pull** feature.

- Introduce **compression** or **delta encoding** to optimize storage.

- Better error handling and improved CLI feedback.

⚡ 5. Architecture Diagram

```mermaid
flowchart TD
    A[User CLI] --> B[main.py]
    B --> C[repository.py]
    B --> D[staging.py]
    B --> E[commit.py]
    B --> F[filetree.py]
    E --> G[Commit Object]
    F --> H[File Tree Snapshot]
```