Perfect! Let's create a **step-by-step professional Git workflow** for your Stage 1–3, assuming you're starting with the sampled 5k dataset. I'll give you **concrete actions** you can follow.

Step 0: Initial Setup

1. Go to GitHub and create a **new private repository**, e.g., `search-engine-project`.

2. On your PC, navigate to your project folder:

```
cd "C:\Users\Muhammad Haris\OneDrive\Desktop\Data_structure_Project\search engine project"
```

3. Initialize Git (if not already done):

```
git init
git remote add origin <your_repo_url>
```

4. Create a `.gitignore` to exclude binaries and output:

```
output/
*.exe
*.bin
*.gz
```

Commit `.gitignore`:

```
git add .gitignore
git commit -m "Add .gitignore to exclude output and binaries"
git push -u origin main
```

Step 1: Stage 1 – Lexicon

Goal: Commit each small step of Lexicon creation.

1. **Add Lexicon class skeleton**:

```
git add include/stage1_lexicon.h src/stage1_lexicon.cpp
git commit -m "MH: Add Lexicon class skeleton with build() method"
git push
```

2. **Implement token parsing and build() logic**:

```
git commit -am "MH: Implement Lexicon::build() to read dataset and store term frequencies"
git push
```

3. **Implement save_segmented() to JSON/segment output**:

```
git commit -am "MH: Implement Lexicon::save_segmented() to save lexicon segments"
git push
```

4. **Test Stage 1 on sample 5k data and fix minor issues**:

```
git commit -am "MH: Test Stage 1 on sample dataset and fix bugs"
git push
```

✅ At this point, **Stage 1 is fully committed**.

Step 2: Stage 2 – Forward Index

Goal: Track the forward index construction.

1. **Add ForwardIndex class skeleton**:

```
git add include/stage2_forward_index.h src/stage2_forward_index.cpp
git commit -m "MH: Add ForwardIndex class skeleton with build() method"
git push
```

2. **Implement build() logic (map docs → term_ids + positions)**:

```
git commit -am "MH: Implement ForwardIndex::build() to convert documents to term_id lists"
git push
```

3. **Implement save_segmented() to save forward index segments**:

```
git commit -am "MH: Implement ForwardIndex::save_segmented()"
git push
```

4. **Test forward index on sample 5k data**:

```
git commit -am "MH: Test Stage 2 ForwardIndex on sample dataset"
git push
```

✅ Stage 2 is now committed.

Step 3: Stage 3 – Inverted Index

Goal: Build inverted index and update lexicon posting_ptr.

1. **Add InvertedIndex class skeleton**:

```
git add include/stage3_inverted_index.h src/stage3_inverted_index.cpp
git commit -m "MH: Add InvertedIndex class skeleton with build() method"
git push
```

2. **Implement build() logic (term → postings)**:

```
git commit -am "MH: Implement InvertedIndex::build() to create term posting lists"
git push
```

3. **Implement save_segmented() and save_segmented_with_lexicon()**:

git commit -am "MH: Implement save_segmented() and save_segmented_with_lexicon() to save inverted index and update lexicon posting_ptr"
git push

4. **Test full pipeline (Stage 1–3) on 5k sample**:

git commit -am "MH: Test full pipeline (Stage 1–3) on sample 5k dataset; verify lexicon posting_ptr"
git push

5. **Update README with instructions**:

git add README.md
git commit -m "MH: Add README with dataset info, usage, and how to run pipeline"
git push

✅ Stage 3 is now fully committed.

Step 4: Moving to Full Dataset Later

- Create a new branch for the 70k dataset:

git checkout -b full_dataset

- Update `main.cpp` path to the full dataset.
- Run the pipeline on the full dataset.
- Commit the results with messages like:

git commit -am "MH: Run full pipeline on 70k dataset; verify outputs"
git push -u origin full_dataset

This separates **sample testing history** from **full dataset execution**, which is **very professional**.

Your Current Status

If you follow this workflow starting **from the sample 5k data**, you can easily reach **20+ meaningful commits**, with:

- Stage 1 (Lexicon) → 4 commits
- Stage 2 (ForwardIndex) → 4 commits
- Stage 3 (InvertedIndex) → 5 commits
