Starting a project with Git from scratch while setting up a React and MongoDB stack is a smart move for version control. Here’s a basic approach to structure your Git workflow and project setup:

**Step 1: Initialize Git Repository**

1. **Create a new folder** for your project.
2. Inside this folder, **initialize a Git repository**:

git init

1. Create a .gitignore file to exclude unnecessary files

touch .gitignore

Add typical entries for Node.js projects (e.g., node\_modules/, .env).

# Dependency directories

node\_modules/

# Optional npm cache directory

.npm

# Debug log files

npm-debug.log\*

yarn-debug.log\*

yarn-error.log\*

# Environment variables

.env

.env.local

.env.\*.local

# Logs

logs

\*.log

logs/\*

# OS-specific files

.DS\_Store

# IDE-specific files

.vscode/

.idea/

#text files

.txt

**Step 2: Initial Commit**

1. **Stage and commit** the .gitignore file:

git add .gitignore git commit -m "Initialize repository with .gitignore"

**Step 3: Set Up the Backend (MongoDB, Express)**

1. **Create a server directory** for the backend.
2. **Initialize a new Node.js project** in the backend folder:

cd server npm init -y

1. **Install required packages**

npm install express mongoose dotenv

1. **Create an entry file** (index.js or server.js), set up Express, and configure MongoDB with Mongoose.

**Step 4: Set Up the Frontend (React)**

1. **Navigate to the project root** and create a React app:

create React app

>>cd ../../

>>c13-public-washrooms

>>npm create vite@latest

>>yes

>>write a name of the project ‘client’

>>Select framework: React

>>Select a variant: Javascript

Done. Now run

cd [Project Name]

npm install

npm run dev

1. **Add any additional dependencies, like Axios for HTTP requests.**

### Step 5: Git Workflow

1. After setting up initial code, **stage and commit** everything:

git add .

git commit -m "Setup backend and frontend structure"

1. **Create branches** for each new feature or component:

git checkout -b feature/initial-setup

1. Push the project to a remote repository, like GitHub, once the initial setup is complete

git remote add origin <your-repo-url> git push -u origin main

**Step 6: Continuous Commits**

* After significant changes or additions, commit with meaningful messages.
* Push frequently to keep your repository updated and minimize conflicts.

$ git push -u origin main To https://github.com/IshaPatel-1992/WashroomDatasetCOG.git ! [rejected] main -> main (fetch first) error: failed to push some refs to 'https://github.com/IshaPatel-1992/WashroomDatasetCOG.git' hint: Updates were rejected because the remote contains work that you do not hint: have locally. This is usually caused by another repository pushing to hint: the same ref. If you want to integrate the remote changes, use hint: 'git pull' before pushing again. hint: See the 'Note about fast-forwards' in 'git push --help' for details.

The --allow-unrelated-histories option is necessary because this is a new local repository that might have an unrelated history from the remote.

git pull origin main --allow-unrelated-histories

 **Resolve Any Merge Conflicts (if they occur)**:

* If there are conflicts, Git will show them in the output. Open the conflicted files, make the necessary changes, and save the files.

 **Commit the Merge** (if there were conflicts):

git add .

git commit -m "Merge remote changes"

**Push the Merged Code to the Remote Repository**:

git push -u origin main

Successfully resolved issue with git setup