Product Design Specification

Team/Project Info

- Project Title: GitInsight
- Team Name: GitInsight
- Team Members: Ankith Indrakumar, Akhil Patil Bagili, Shanmukha Sri Harsha Anvilla

  Reason for Team Name: The name "GitInsight" was chosen to reflect our team's mission of providing deep insights and understanding of Git repositories, highlighting our focus on making complex data accessible and insightful for users with varying levels of technical expertise.

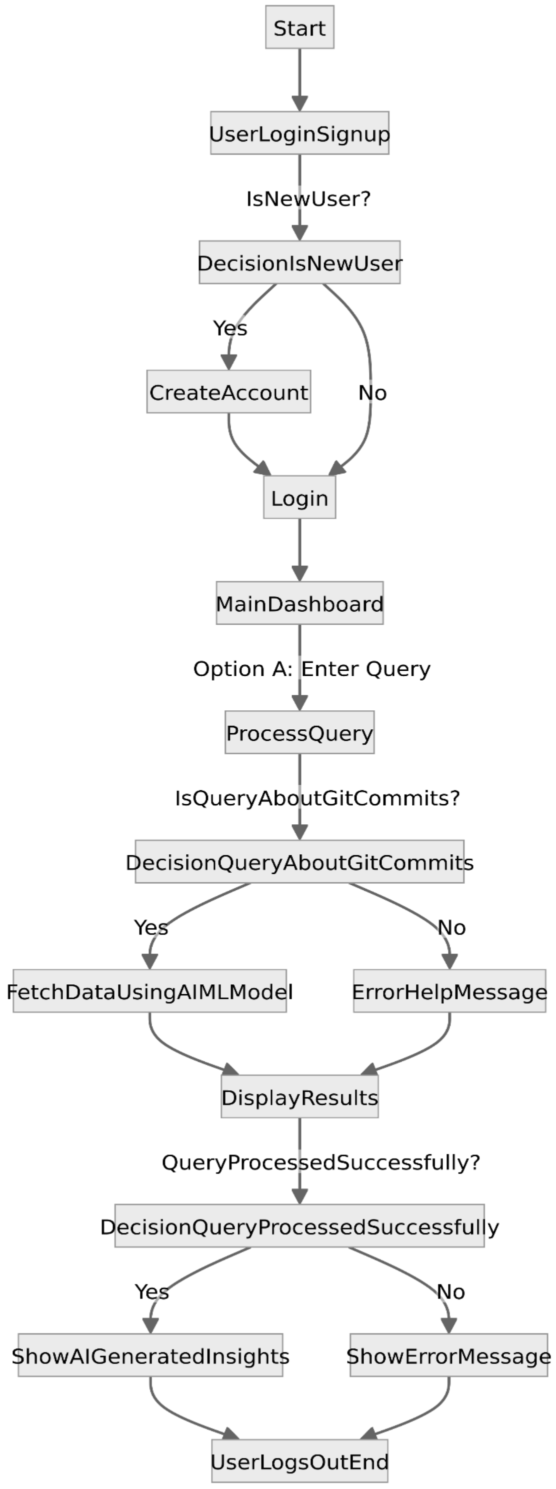
  Technologies Used:
  - Backend: Flask, Astra DB, llama-index, Cassio
  - Frontend: React, Tailwind CSS
  - AI/ML Modeling: GPT-4, RAG (Retrieval-Augmented Generation) model
  - Deployment: Render

  Specific Tasks per Person:
  - Ankith Indrakumar: Project management, backend development, AI/ML modeling, integration, database management, security, deployment.
  - Akhil Patil Bagili: Project management, frontend development, UI/UX design, database management, security, deployment, testing, documentation.
  - Shanmukha Sri Harsha Anvilla: Frontend development, UI/UX design, AI/ML modeling, integration, deployment, testing, documentation.

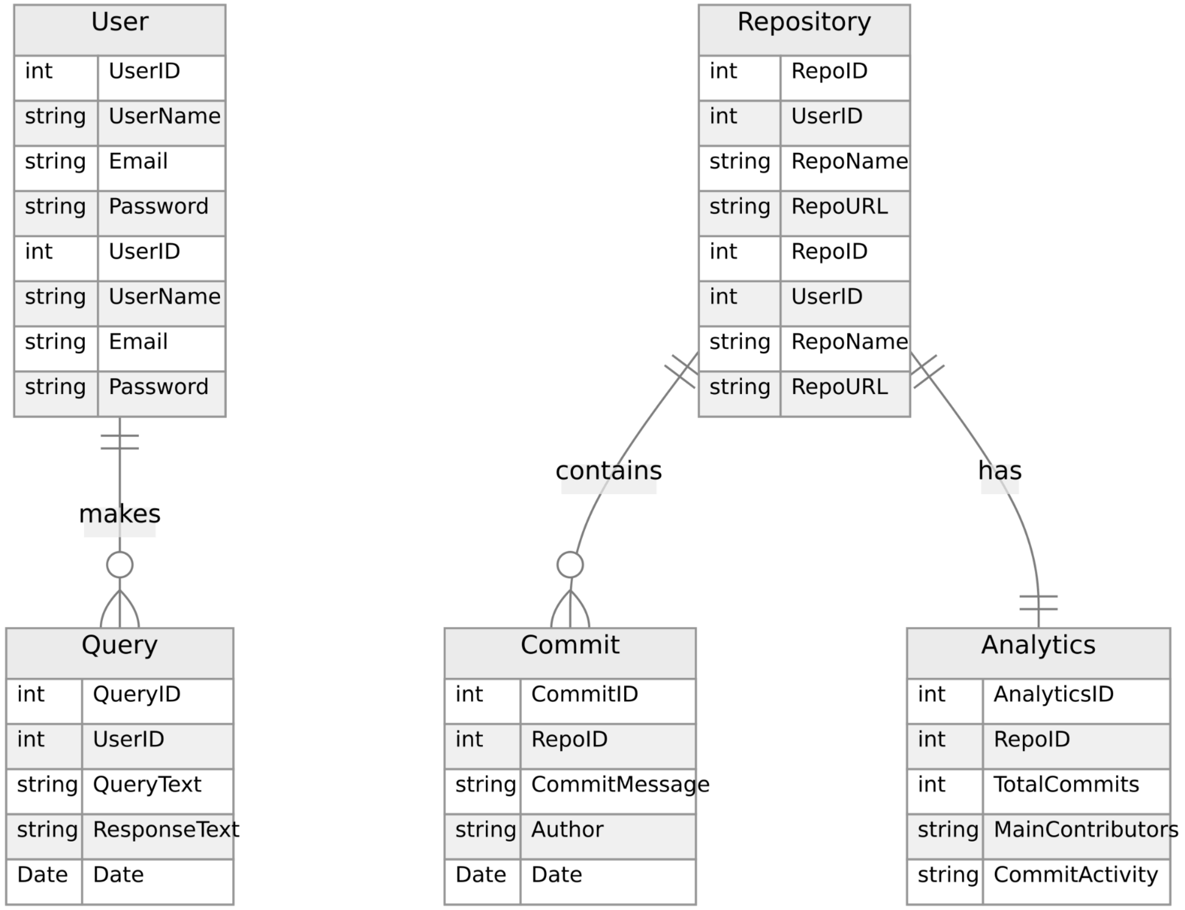
Project Description

- **End Product:** GitInsight is a web application designed to offer developers, project managers, and non-technical users insightful analytics and a deeper understanding of Git commits without requiring command line expertise. It features a user-friendly chat interface that allows users to ask open-ended questions about Git commits, with responses generated by an AI model. The application provides detailed analytics beyond simple commit counts, aiming to enhance productivity and project management efficiency.

Flowchart:


Start
UserLoginSignup
IsNewUser?
DecisionIsNewUser
Yes
No
CreateAccount
Login
MainDashboard
Option A: Enter Query
ProcessQuery
IsQueryAboutGitCommits?
DecisionQueryAboutGitCommits
Yes
No
FetchDataUsingAIMLModel
ErrorHelpMessage
DisplayResults
QueryProcessedSuccessfully?
DecisionQueryProcessedSuccessfully
Yes
No
ShowAIGeneratedInsights
ShowErrorMessage
UserLogsOutEnd

ER Diagram:



User	
int	UserID
string	UserName
string	Email
string	Password
int	UserID
string	UserName
string	Email
string	Password

Repository	
int	RepoID
int	UserID
string	RepoName
string	RepoURL
int	RepoID
int	UserID
string	RepoName
string	RepoURL

Query	
int	QueryID
int	UserID
string	QueryText
string	ResponseText
Date	Date

Commit	
int	CommitID
int	RepoID
string	CommitMessage
string	Author
Date	Date

Analytics	
int	AnalyticsID
int	RepoID
int	TotalCommits
string	MainContributors
string	CommitActivity

- **Algorithms/ML/AI Schemes:** Utilizes Vector DBs for efficient data indexing and retrieval, and RAG (Retrieval-Augmented Generation) model powered by OpenAI's GPT models for generating responses and understanding user queries.
- **Market Space and Selling Points:**
  - Market Space: Targets developers, project managers, and non-technical users needing insights into Git commits without the complexity of the command line.
  - Selling Points: Unique ability to answer open-ended questions about Git commits, user-friendly chat interface, and detailed analytics beyond simple commit counts.

Functional Specifications

- **Product Features:**
  a. Chat Interface: Enables users to interact with the AI model by asking open-ended questions about Git commits.
  b. Detailed Analytics Dashboard: Provides visual analytics on commit activity, including trends, contributor statistics, and more.
  c. User Management: Allows for user account creation, management, and customization of preferences.
  d. Integration with Git Repositories: Users can connect their Git repositories for real-time insights and analytics.
  e. Responsive Design: Ensures a seamless user experience across various devices and screen sizes.
- **Multiplayer Features:** While GitInsight primarily focuses on individual use, its collaborative features enable multiple users to work on analyzing and managing a single Git repository together, fostering a team-oriented approach to project management and data analysis.

Deployment

- Prepare Flask application by ensuring all dependencies are listed in a requirements.txt file and it is configured for production.
- Create an account on Render and navigate to the dashboard to create a new web service.
- Connect the GitHub repository to Render, selecting the repository where your Flask application is stored.
- Configure the build settings according to the Flask application requirements. Render will detect our requirements.txt file and install the necessary dependencies.
- Set up any environment variables or secret files that our application requires through the Render dashboard.
- Choose the desired plan and region for deployment, then click 'Deploy'. Render will automatically deploy our application and provide a public URL for access.
- Monitor the application's performance through Render's dashboard and make adjustments or scale our service as needed.

Milestone Features:

- **M1 (Week 1-2):** Project setup, initial design, backend development kickoff
- **M2 (Week 3-4):** AI model development, frontend development starts

- **M3 (Week 5-6):** Database and vector search integration, AI and frontend integration
- **M4 (Week 7-8):** Security measures and API integration, initial testing phase
- **M5 (Week 9-10):** Feedback implementation, final adjustments, deployment, and documentation completion