AI Agent for Task Automation: Fix the Code

Nirjhar & Vardhan

August 20, 2024

Outline

Problem Statement

Given a repository containing bugs, our goal is to:

- Identify and locate the buggy file(s) based on the bug description

Problem Statement

Given a repository containing bugs, our goal is to:

- Identify and locate the buggy file(s) based on the bug description
- Suggest and implement corrected version(s) of the buggy file(s)

Example

```
Bug -
Title - Resolve Loader Exception for valid pdf files.
Description - When I uploaded a valid file, loader process throws
exception with the message that can't read an empty file.
Actual Behavior: The file should be processed as it is not an empty
file and the chatbot should appear without throwing any exception.
```

What are LLMs?

- Advanced AI models trained on vast text data
- Understand and generate human-like language
- Perform tasks such as:
  - Text generation
  - Translation
  - Summarization
  - Code generation
- Examples:
  - OpenAI's GPT series
  - Google's BERT
  - Meta's LLaMA

What are Agents?

- Software entities performing specific tasks

What are Agents?

- Software entities performing specific tasks
- Equipped with an LLM

What are Agents?

- Software entities performing specific tasks
- Equipped with an LLM
- Have a defined role and goal

What are Agents?

- Software entities performing specific tasks
- Equipped with an LLM
- Have a defined role and goal
- Examples:

What are Agents?

- Software entities performing specific tasks
- Equipped with an LLM
- Have a defined role and goal
- Examples:
  - Article Generating System: Planner agent, Writer agent, Editor agent

What are Agents?

- Software entities performing specific tasks
- Equipped with an LLM
- Have a defined role and goal
- Examples:
  - Article Generating System: Planner agent, Writer agent, Editor agent
  - Automatic Mailing System: Web Scraping agent, Email Drafting agent, Email Reviewing agent, Automatic Email Sending agent

Approach

Our approach mirrors the process a developer would follow:

1. Analyze dependencies and relationships between files to understand how they interact.

Approach

Our approach mirrors the process a developer would follow:

1. Analyze dependencies and relationships between files to understand how they interact.
2. Identify the buggy file(s) related to the issue, based on the provided description.

Approach

Our approach mirrors the process a developer would follow:

1. Analyze dependencies and relationships between files to understand how they interact.
2. Identify the buggy file(s) related to the issue, based on the provided description.
3. Use AI agents to fix the identified buggy files, considering any related dependencies.

Implementation

- **Model:** llama3-8b-8192 from Groq

Implementation

- **Model:** llama3-8b-8192 from Groq
- **Agent System:** Built using the 'crewai' library, enabling agentic way for code analysis, fixing and reviewing.

Implementation

- **Model:** llama3-8b-8192 from Groq
- **Agent System:** Built using the 'crewai' library, enabling agentic way for code analysis, fixing and reviewing.
- **User Interface:** Streamlit-based, offering a simple and intuitive way for users to interact with the system.

Implementation

- **Model:** llama3-8b-8192 from Groq
- **Agent System:** Built using the 'crewai' library, enabling agentic way for code analysis, fixing and reviewing.
- **User Interface:** Streamlit-based, offering a simple and intuitive way for users to interact with the system.
- **Key Functions:**

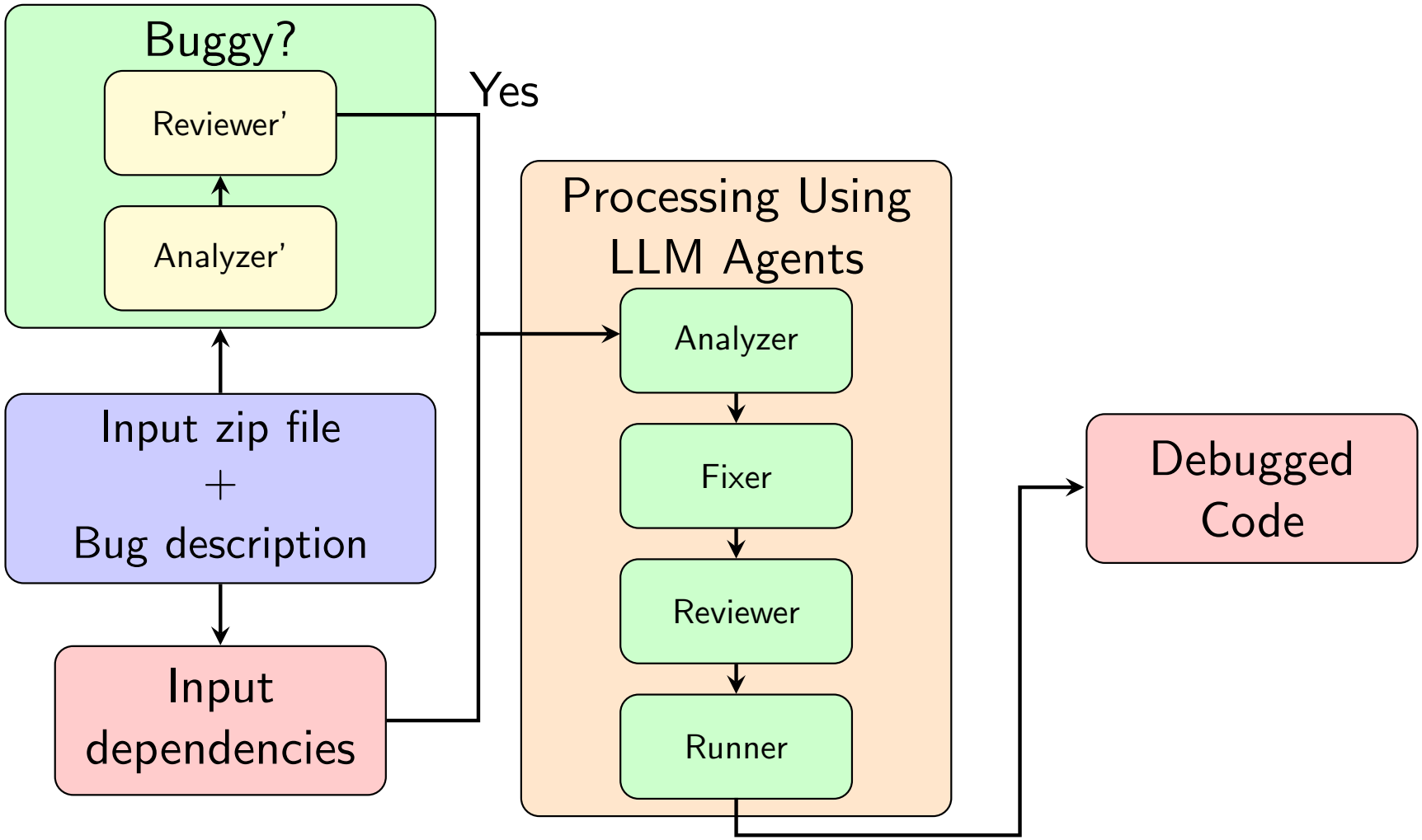
Implementation

- **Model:** llama3-8b-8192 from Groq
- **Agent System:** Built using the 'crewai' library, enabling agentic way for code analysis, fixing and reviewing.
- **User Interface:** Streamlit-based, offering a simple and intuitive way for users to interact with the system.
- **Key Functions:**
  - Analyzes code to locate buggy files using AI-based code inspection.

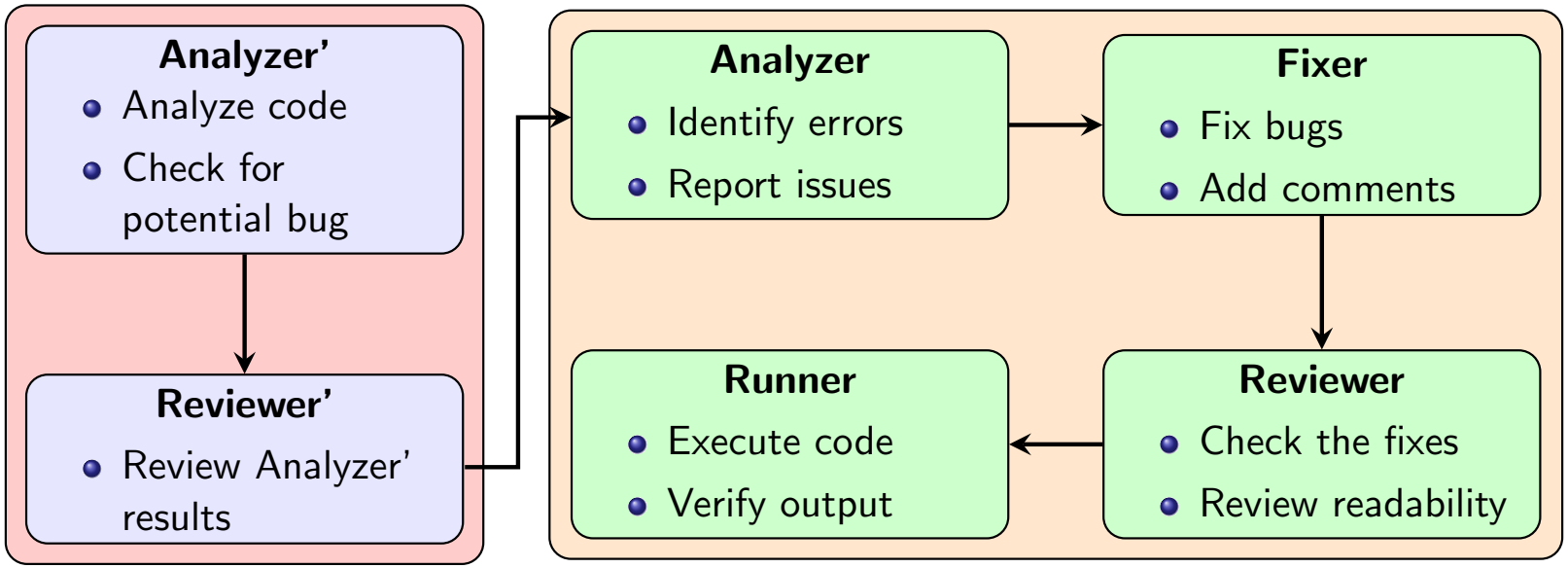
Implementation

- **Model:** llama3-8b-8192 from Groq
- **Agent System:** Built using the 'crewai' library, enabling agentic way for code analysis, fixing and reviewing.
- **User Interface:** Streamlit-based, offering a simple and intuitive way for users to interact with the system.
- **Key Functions:**
  - Analyzes code to locate buggy files using AI-based code inspection.
  - Generates fixes for the identified issues, ensuring the solution is effective and doesn't introduce new problems.

Implementation flowchart



Workflow of Agents



Implementation Details

- **Dependencies:** Manually stored in a JSON file, capturing file relationships for accurate bug identification.

Implementation Details

- **Dependencies:** Manually stored in a JSON file, capturing file relationships for accurate bug identification.
- **Bug Detection:**

Implementation Details

- **Dependencies:** Manually stored in a JSON file, capturing file relationships for accurate bug identification.
- **Bug Detection:**
  - Files requiring fixes are marked as 'Yes'

Implementation Details

- **Dependencies:** Manually stored in a JSON file, capturing file relationships for accurate bug identification.
- **Bug Detection:**
  - Files requiring fixes are marked as 'Yes'
  - Files not needing fixes are marked as 'No'

Implementation Details

- **Dependencies:** Manually stored in a JSON file, capturing file relationships for accurate bug identification.
- **Bug Detection:**
  - Files requiring fixes are marked as 'Yes'
  - Files not needing fixes are marked as 'No'
- **Fix Process:** Once identified, the system generates corrected code and provides an explanation for the changes.

User Interface

- **Inputs Required:**

User Interface

- **Inputs Required:**
  - Bug description from the user

User Interface

- **Inputs Required:**
  - Bug description from the user
  - Repository in ZIP format, containing the codebase

User Interface

- **Inputs Required:**
  - Bug description from the user
  - Repository in ZIP format, containing the codebase
  - Dependencies in JSON format, detailing file relationships

User Interface

- **Inputs Required:**
  - Bug description from the user
  - Repository in ZIP format, containing the codebase
  - Dependencies in JSON format, detailing file relationships
- **Outputs:**

User Interface

- **Inputs Required:**
  - Bug description from the user
  - Repository in ZIP format, containing the codebase
  - Dependencies in JSON format, detailing file relationships
- **Outputs:**
  - List of probable buggy files based on the analysis

User Interface

- **Inputs Required:**
  - Bug description from the user
  - Repository in ZIP format, containing the codebase
  - Dependencies in JSON format, detailing file relationships
- **Outputs:**
  - List of probable buggy files based on the analysis
  - Fixed code for those files, accompanied by explanations to help users understand the changes made

References

1. OpenAI's GPT: https://chat.openai.com/
2. CrewAI: https://www.crewai.com/
3. Prompt Engineering: https://www.promptingguide.ai/

Thank you!