Infosys Springboard

HONQP Gen

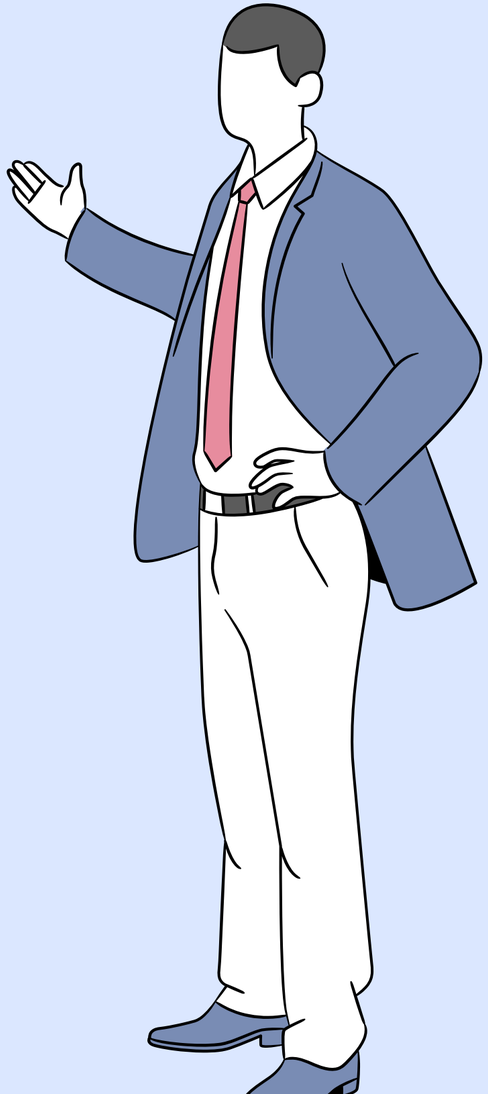
Team Mentor:
Priyanka Mathapati
Sarayu Keshavan

Team Members:
Praadnya H
Prannay Hemachandran
Shivangi Singh
Swati Swagatika
Ajit Yadav
Avni Soni

Infosys Springboard

About the project

HONQPGen is a cutting-edge initiative focusing on automated extraction of comprehensive method insights and intricate class relationships in Java code, optimizing code documentation through sophisticated parsing and analysis techniques.



Infosys Springboard

HONQPGen

Today's Agenda

1. Objective
2. Background Research
3. Technology Used
4. Block Diagram
5. Takeaways/Learning Outcomes
6. Output
7. Obstacles

Objective

Method Description Generation

Develop an automatic Java method description generator using Python.It will parse Java source code, extract relevant information about classes, data members, and methods, and with the help of python generate human-readable descriptions.

Class Diagram Creation

Develop an intelligent tool capable of creating detailed class diagrams for Java code, illustrating relationships between classes.

Background Research

Methodological Evolution: Navigating Challenges from Pre-trained Models to Custom Neural Networks for Java Method Description Generation.

DATASET CREATION	Curated a dataset of 50 Java code samples.
TEXT PROCESSING	Utilized NLTK for tokenization, lemmatization, and stemming to extract relevant information from code snippets.
PRE-TRAINED MODELS EXPLORATION	Explored fine-tuning of pre-trained models (Google T5 Fan Base, BERT from Hugging Face) but encountered challenges due to the unique nature of the problem.
CUSTOM NEURAL NETWORK	Implemented a second approach by building a custom neural network from scratch but RNNs faced accuracy issues attributed to the small dataset size.

Technology Used

a. Parsing and Analysis:

Utilized libraries:

1. **javalang** for parsing Java code and extracting syntax tree information.
2. **antlr4** for parsing Java code and walking the Abstract Syntax Tree (AST) to gather relevant details

We employed these libraries to analyze the structure and behavior of Java code samples which was created manually, extracting essential information for documentation.

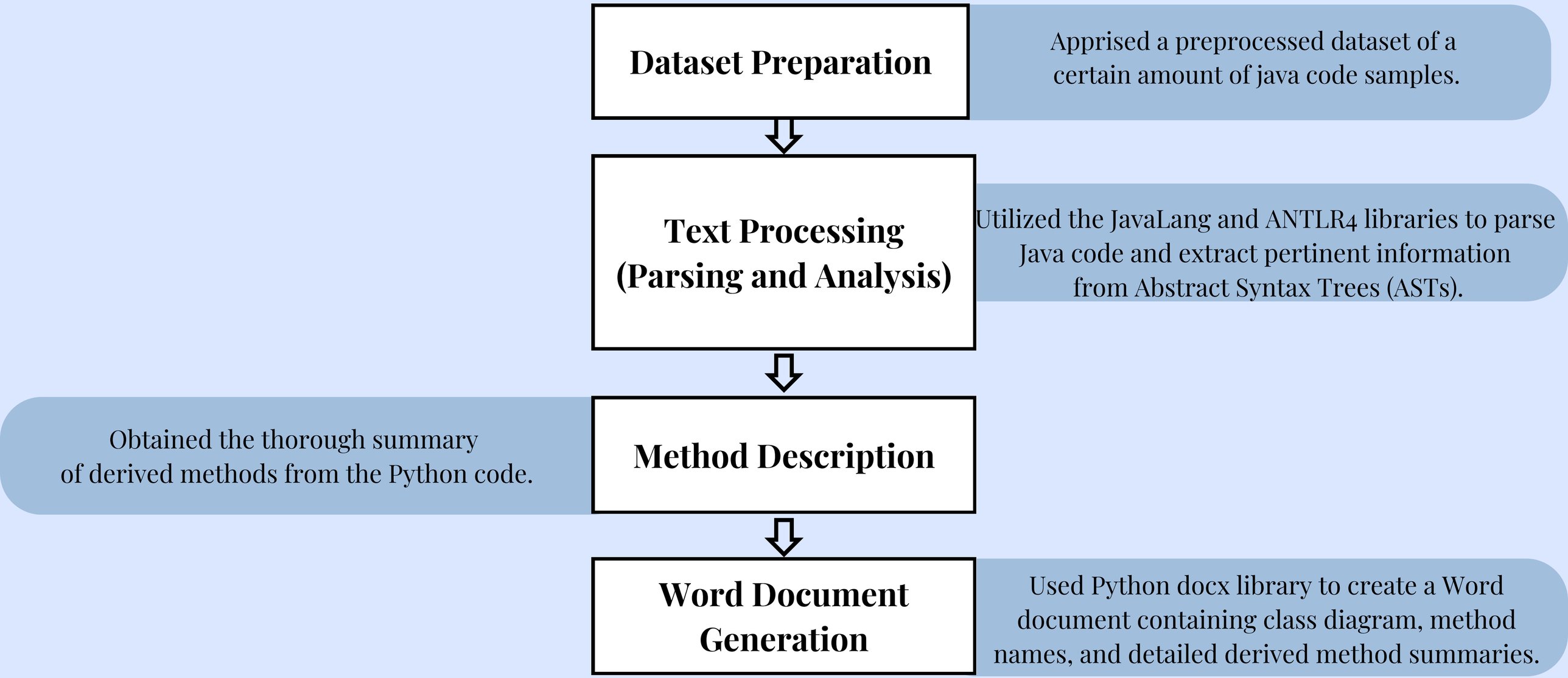
b. Documentation Generation:

Utilized Python libraries:

1. Utilized the docx library in Python for generating structured Word documents.
2. Applied document styling and formatting to present information in a clear and organized manner.

Block Diagram

HONQPGen



Takeaways/Learning Outcomes

Gained proficiency in utilizing Python libraries (javalang and antlr4) for parsing Java code and extracting information from the AST.

Code Parsing and Analysis

Developed skills in integrating external libraries seamlessly into the Python workflow for code analysis purposes.

Integration of Libraries

Acquired knowledge and experience in using the docx library to create professional and structured documentation.

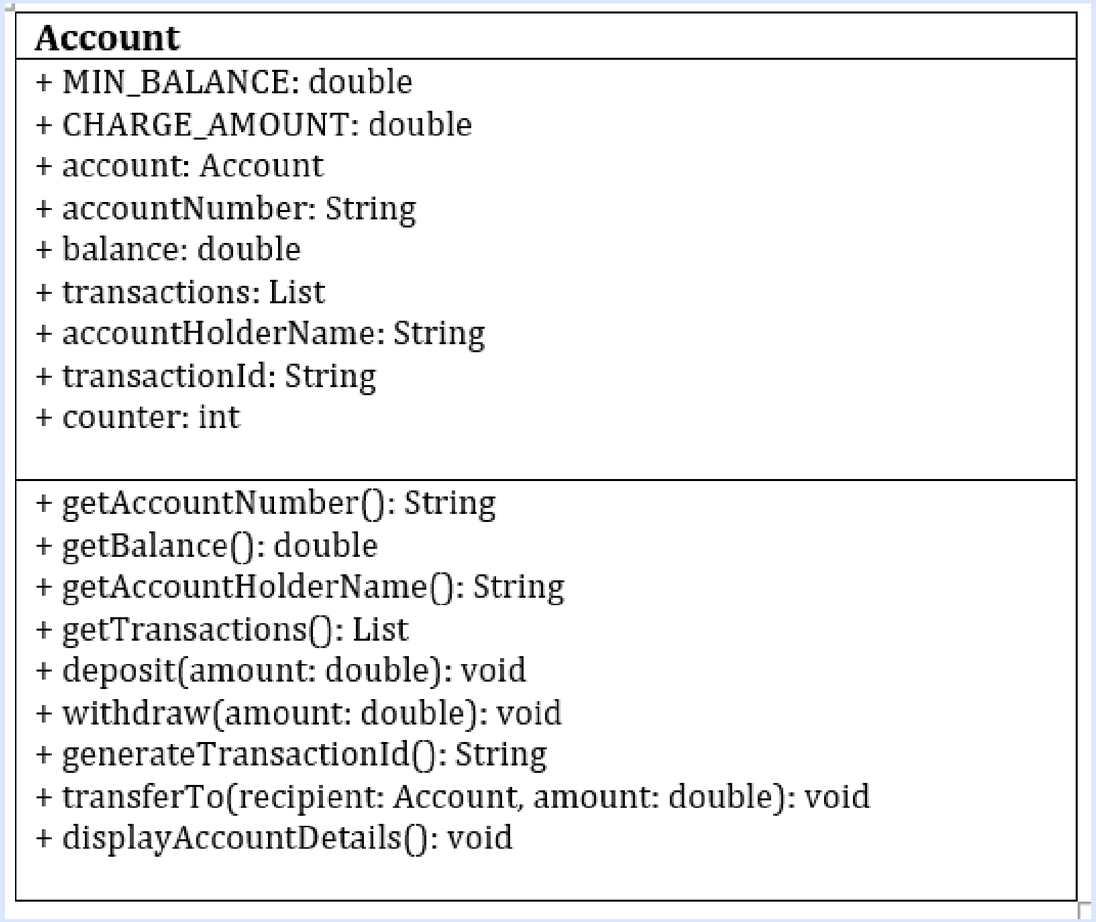
Document Generation

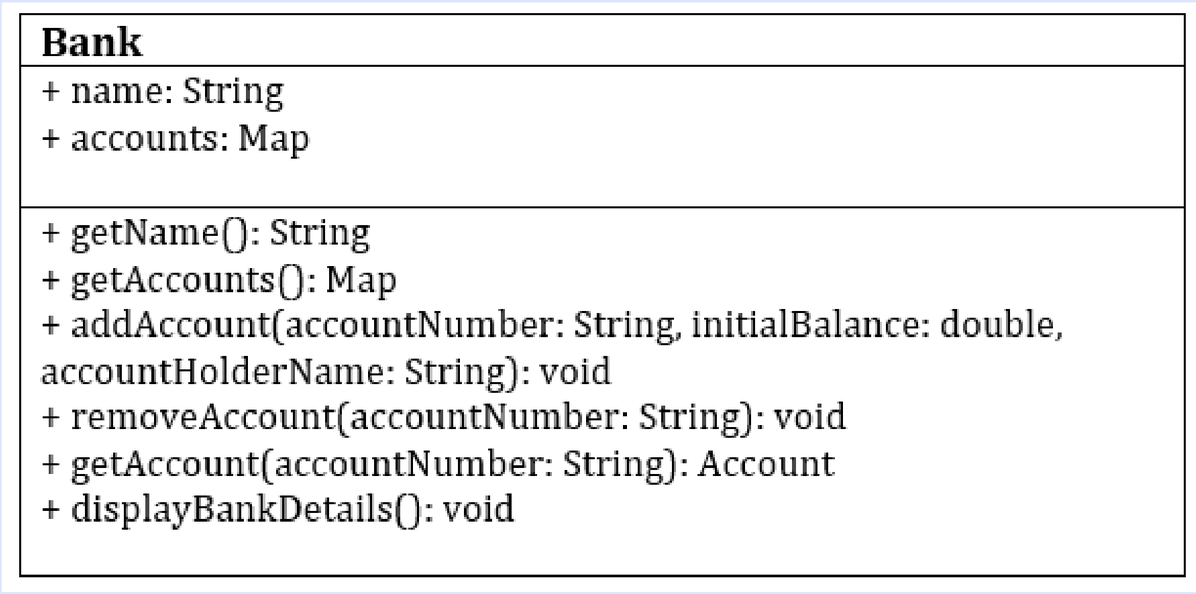
Developed problem-solving skills in addressing challenges related to code analysis, ensuring accuracy in method descriptions.

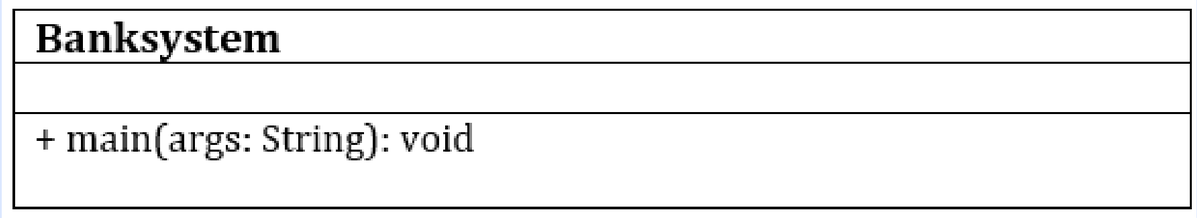
Problem-Solving

Output

Class Diagram Generation







Method Description

Method Name: generateTransactionId

- This method auto generates and sets the transactionId.
- The transactionId would be prefixed by the last 3 characters of the accountHolderName in lowercase followed by auto-generated value starting from 1001.
- The auto-generated value would be incremented by one for the next transactionId. Use static variable counter appropriately to implement the auto generation logic.

Obstacles

a. Incomplete Method Descriptions:

Challenge: The current implementation generates descriptions for specific methods, leaving room for enhancement.

Solution: Refining the method description logic to cover a broader range of method patterns, ensuring comprehensive documentation.

b. Class Relationship Identification:

Challenge: The code cannot identify relationships between classes in the generated documentation.

Solution: Investigating and implementing techniques to detect and represent class relationships, including associations, aggregations etc for a more holistic view of the code structure.

THANK YOU