

CS 588 Final Project - Phase 3

Group 17

Kamble, Soham (skamble2@hawk.iit.edu) A20517098

Chawla, Bhavya (bchawla@hawk.iit.edu) A20516957

The use of generative AI to generate estimates for the project plan, design document, source code, test plan and test cases, and user API documentation

Table of Content

Table of Content	2
1. Problem statement, features, and requirements	4
Problem Statement:	4
Features:	4
1. Data Source:	4
2. Data Preparation and Analysis	5
3. Open AI Integration	5
4. Prompt Engineering	5
Prompt for Estimated Effort:	5
Prompt for Object Points:	6
5. Dynamic SQL Query Generation and Execution:	7
6. Error Handling	7
7. Estimate Prediction	7
8. Scalability	7
9. Accuracy	7
Requirements:	8
Functional Requirements	8
Task-Specific Estimation:	8
Customization:	8
Real-time Estimation:	8
Data Integration:	8
Non-Functional Requirements	9
Accuracy of Estimates:	9
Cost Control:	9
Response Time:	9
Scalability	9
2. Architecture and design diagrams	9
Dataset Attributes:	9
Schema for TestingData Table:	10
Context Diagram:	11
Feature Tree:	12
Flowchart:	12
3. Measurements and metrics that are used in your benchmark for the comparative analysis of the experimental results	13
The formula for estimated effort and object points based on dataset attributes	13
Estimated effort:	13
Actual effort:	13
Object Points:	13
R ² Score	13

Scatter Plots	14
Monitoring with Prometheus	15
4. List of tools used to conduct/develop the experiments	16
Development Environment	16
Visual Studio Code:	16
Database Management	16
PgAdmin:	16
PostgreSQL:	16
Machine Learning and Natural Language Processing	17
OpenAI GPT-3 API:	17
Data Processing and Analysis	17
Pandas:	17
SQL Query Generation and Execution	17
SQLAlchemy:	17
Visualization	17
Matplotlib:	17
Metric Tracking and Monitoring	17
Prometheus:	17
5. Data sources that you have used for your final project.	18
6. Executive summary of your final research report	19
Key Features and Processes	20
Data Source:	20
OpenAI Integration:	20
Prompt Engineering:	20
Estimation:	20
Accuracy and Evaluation:	20
Monitoring and Metrics:	20

1. Problem statement, features, and requirements

Problem Statement:

The purpose of this project is to use generative AI capabilities (GPT-3 from OpenAI to improve the automation of project management tasks. It includes fully integrating a PostgreSQL database and functioning as a repository for project data that we got from the SEERA Software Cost Estimation Dataset.

Our code interacts with the PostgreSQL database, effectively fetching and generating project estimates. Additionally, OpenAI's intelligent virtual assistant is an essential part, engaged not only for data retrieval but also for content generation, responding to prompts for project plans, design documents, and other related artifacts.

In addition to its generative role, the virtual assistant is important in calculating estimations for effort duration, using the formula established within the SEERA dataset as a standard. This formula is essential to our approach, giving a systematic means to calculate estimated durations and store them within the PostgreSQL database.

We determine the accuracy and performance of our estimations through a comparison between the estimated durations and the actual durations derived from the SEERA dataset. This analysis serves as an essential evaluation method, confirming the reliability and applicability of our generative AI model.

Our project seeks to go beyond the standard models of project management, providing a fusion of generative AI and accurate performance validation.

Features:

1. Data Source:

SEERA Software Cost Estimation Dataset

We are using the SEERA Software Cost Estimation Dataset as the primary data source for estimation by the generative AI model to ensure the AI model is trained and has knowledge of a diverse and comprehensive dataset, enhancing its ability to generate accurate estimates.

The dataset has a collection of 120 software development project data from 46 organizations, and the dataset contains 76 attributes that can be considered for the estimations.

2. Data Preparation and Analysis

We have been involved in preprocessing and analyzing the raw dataset to identify patterns, outliers, and relevant features. This enhanced the quality of the input data, making it suitable for training and improving the generative AI model's accuracy.

3. Open AI Integration

We have integrated OpenAI's GPT-3 into the project to leverage its capabilities for generating human-like text based on prompts, and we utilize advanced language models to automate the generation of estimates and results.

4. Prompt Engineering

We put a lot of effort into crafting effective and contextually relevant prompts to extract the desired information from the generative AI model. Also, we are using the *text-davinci-003* model and have adjusted the temperature to optimize the interaction with the AI model to generate accurate and meaningful project-related content.

Prompt for Estimated Effort:

```
# Prepare the prompt for the first part of Estimated Effort (Duration * Team members)
effort_part1_prompt = f"Calculate the first part of Estimated Effort using the formula:
Estimated duration ({row['estimatedduration']} days) * Dedicated Team members
({row['dedicatedteammembers']})."

```

```
# Make an API call for the first part of Estimated Effort
try:
    with response_time.time():
        response_part1 = openai.Completion.create(
            engine="text-davinci-003",
            prompt=effort_part1_prompt,
            temperature=0.7,
            max_tokens=50,
            n=1,
            stop=None
        )
        api_calls.inc() # Increment the API call counter
except Exception as e:
    errors.inc() # Increment the error counter
    print(f"Error: Unable to make an API call. Detailed error: {e}")

```

```

# Extract the first part of Estimated Effort
effort_part1 = extract_numeric_value(response_part1.choices[0].text) if
response_part1.choices else None

# Calculate the second part of Estimated Effort (Remaining Team members * 0.5)
remaining_team_members = row['teamsize'] - row['dedicatedteammembers']
effort_part2_prompt = f"Calculate the second part of Estimated Effort using the formula:
Remaining Team members ({remaining_team_members}) * 0.5."

# Make an API call for the second part of Estimated Effort
try:
    with response_time.time():
        response_part2 = openai.Completion.create(
            engine="text-davinci-003",
            prompt=effort_part2_prompt,
            temperature=0.7,
            max_tokens=50,
            n=1,
            stop=None
        )
    api_calls.inc() # Increment the API call counter

except Exception as e:
    errors.inc() # Increment the error counter
    print(f"Error: Unable to make an API call. Detailed error: {e}")

```

Prompt for Object Points:

```

object_points_prompt = f"Calculate Object Points as follows:\n"
object_points_prompt += f"1. Sum the Number of screens ({row['numberofscreens']}).\n"
object_points_prompt += f"2. Sum the Number of reports ({row['numberofreports']}).\n"
object_points_prompt += f"The final Object Points are the sum of these values.\n"

```

```

try:
    with response_time.time():
        response_object_points = openai.Completion.create(
            engine="text-davinci-003",
            prompt=object_points_prompt,
            temperature=0.7,
            max_tokens=50,
            n=1,
            stop=None
        )
    api_calls.inc()

```

```
except Exception as e:  
    errors.inc() # Increment the error counter  
    print(f"Error: Unable to make an API call. Detailed error: {e}")
```

5. Dynamic SQL Query Generation and Execution:

Generates and executes SQL queries based on prompts, showcasing the integration of natural language processing with database management.

6. Error Handling

We have implemented measures to identify and handle errors or inaccuracies in the generative AI model's outputs. To attain one of our goals and requirements, to ensure the reliability of the generated estimates.

7. Estimate Prediction

We utilized the model to predict estimates for the effort of each project based on the utilized attributes, and we also estimated the object points for all the projects. These estimates provide automated and data-driven estimates for different aspects of the project life cycle and further scale it.

8. Scalability

We are assured that the system can further accommodate the increasing complexities of estimations and data volumes and can be implemented to include more estimations in the future.

9. Accuracy

We have focused on evaluating and improving the accuracy of the model when generating estimates. Taking into account efficient prompt engineering and formulas, we aimed to validate the model's effectiveness and enhance its practical utility. We have evaluated the accuracy of our model with the actual estimates with R^2 and scatter plots.

Requirements:

Functional Requirements

Task-Specific Estimation:

Our model enables us to provide accurate estimates for specific project management tasks like environments, users, developers, projects, and products. It is able to compute, based on related attributes, the required estimations.

Customization:

We can further customize and fine-tune the generative AI model for more specific data, like countries, tasks, and documents.

Real-time Estimation:

We ensure the AI system provides estimates in real-time, facilitating quick decision-making.

Data Integration:

We integrated seamlessly with the GPT API and Postgres database to enhance the diversity of data and CRUD operations

Non-Functional Requirements

Accuracy of Estimates:

Our system provides estimates with a high degree of accuracy. With our efforts to minimize errors, we received an accuracy of 0.941 and 0.970, which underscores the model's precision and reliability for object points and estimation effort, respectively.

Cost Control:

We have cost control requirements as a future scope to ensure that the estimated project plans align with budget constraints to facilitate effective cost control. The attributes are yet to be determined for cost control estimation.

Response Time:

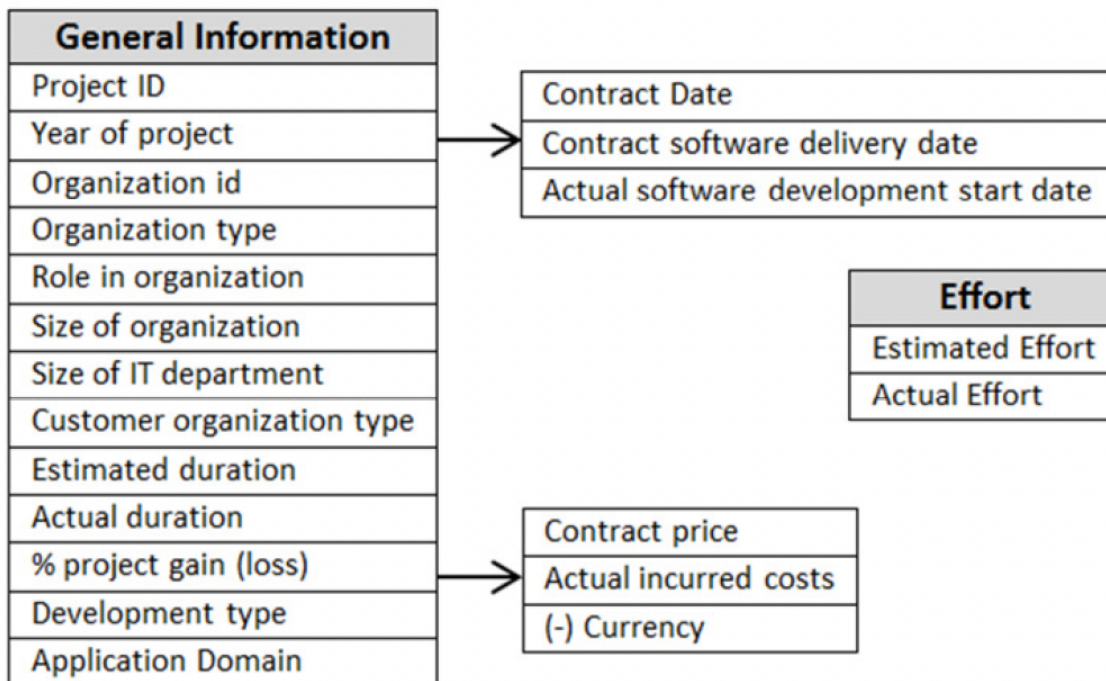
The system responds to queries and prompts within seconds, ensuring real-time usability.

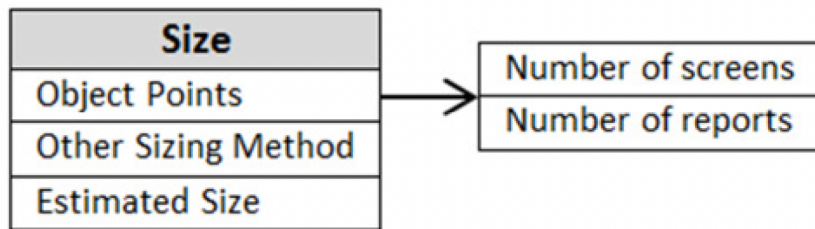
Scalability

The system should efficiently handle increasing attributes and larger datasets.

2. Architecture and design diagrams

Dataset Attributes:

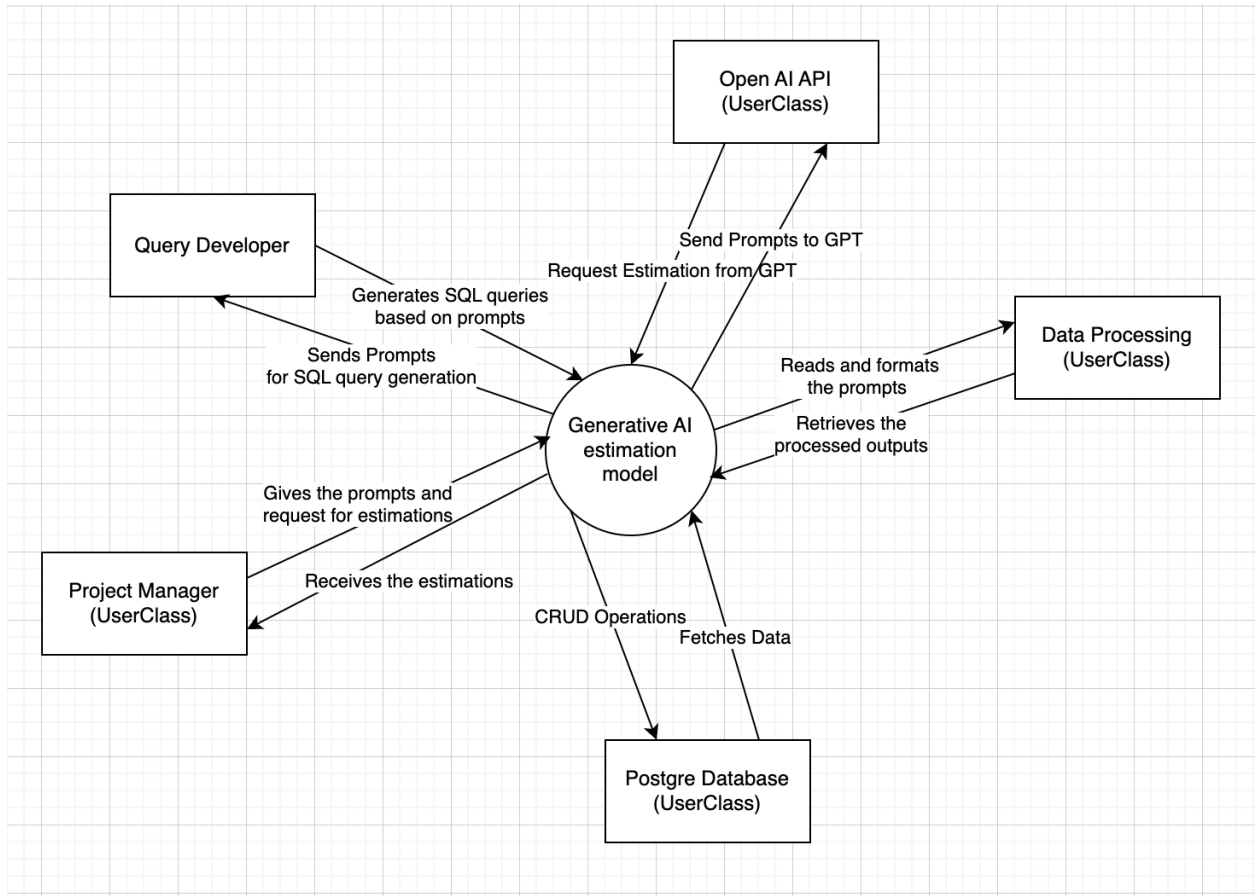




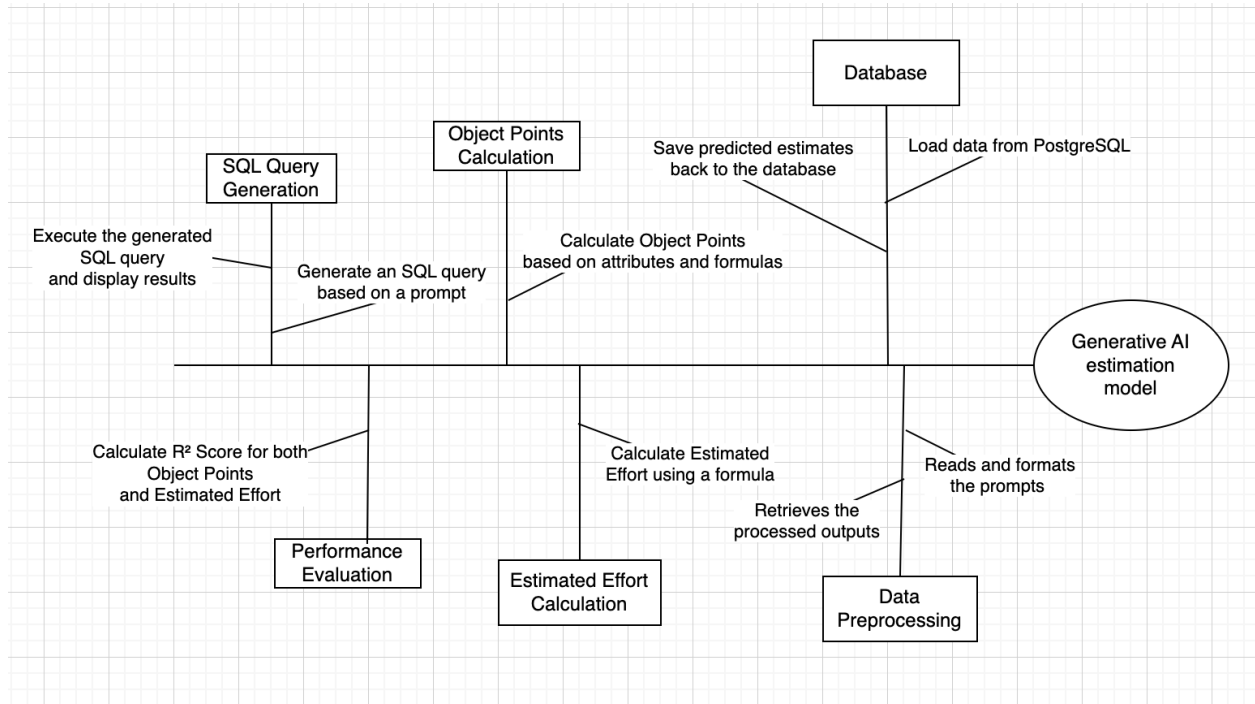
Schema for TestingData Table:

```
CREATE TABLE Public."TestingData" (
    projID INT,
    yearofproject INT,
    organizationid INT,
    sizeoforganization INT,
    numberofscreens INT,
    numberofreports INT,
    estimatedduration FLOAT,
    teamsize INT,
    dedicatedteammembers INT,
    dailyworkinghours INT,
    monthlyworkinghours INT,
    actualobjectpoints INT,
    actualestimatedeffort INT
);
```

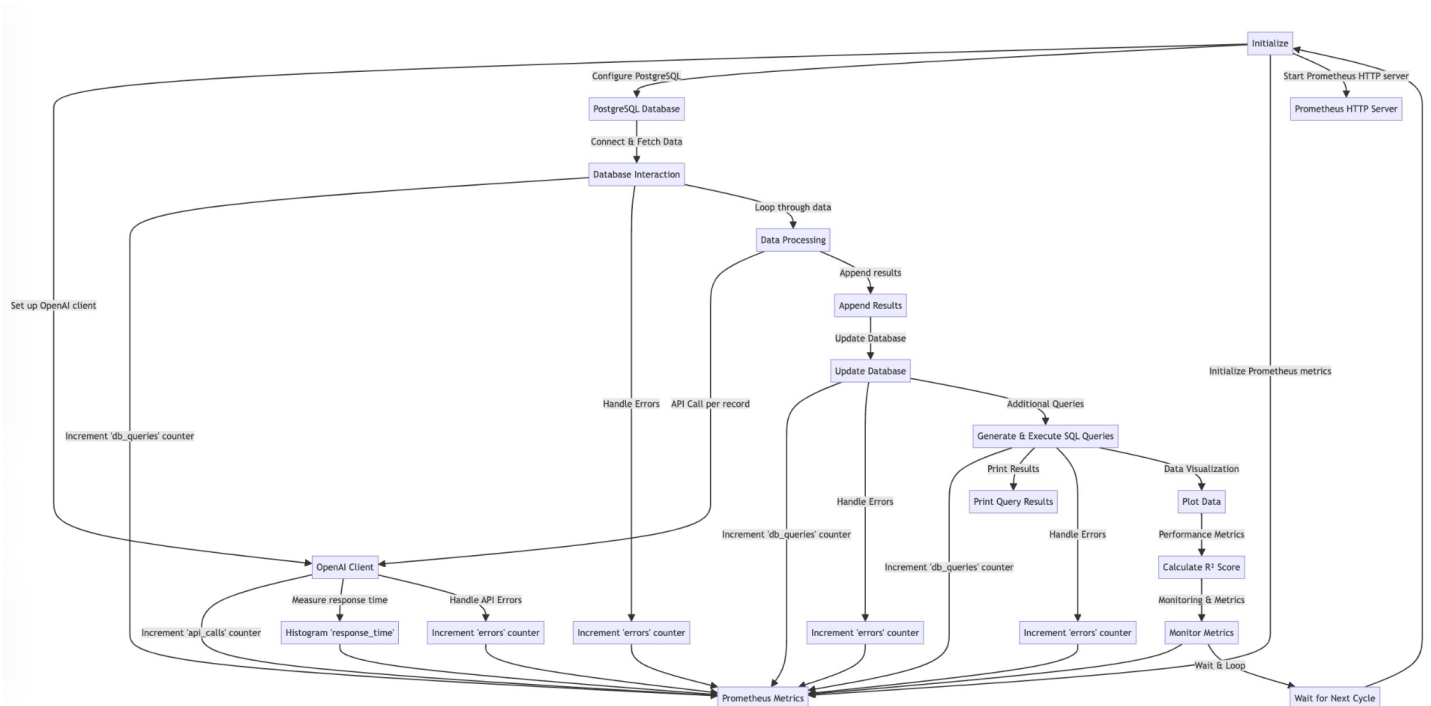
Context Diagram:



Feature Tree:



Flowchart:



3. Measurements and metrics that are used in your benchmark for the comparative analysis of the experimental results

The formula for estimated effort and object points based on dataset attributes

Estimated effort:

$$\left[\text{Estimated duration} * (\text{Dedicated Team members} + (\text{Team size} - \text{Dedicated Team members}) * 50\%) \right] * (\text{Daily working hours} * 22)$$

Actual effort:

$$\left[\text{Actual duration} * (\text{Dedicated Team members} + (\text{Team size} - \text{Dedicated Team members}) * 50\%) \right] * (\text{Daily working hours} * 22)$$

Object Points:

$$\text{Number of screens} + \text{Number of reports}$$

We have utilized machine learning techniques to evaluate and enhance accuracy and reliability. We implemented R^2 score, thus helping us evaluate the accuracy of our model in predicting effort estimation against the actual values. Further, we have plotted scatter plots, which visualize the relationship between actual and predicted values for both object points and estimated effort

R^2 Score

We implemented the coefficient of determination (R^2) to evaluate our model's ability to predict object points and estimated effort accurately. The score provides insights into the goodness of fit between predicted and actual values. A higher R^2 indicates a better model fit.

The R^2 scores achieved in our analysis reflect a high level of accuracy in our model predictions.

$$\text{Object Points} - R^2 \text{ Score: } 0.940763681596664$$

$$\text{Estimated Effort} - R^2 \text{ Score: } 0.9697761036280876$$

For object points, the R^2 score stands at an impressive 0.941.

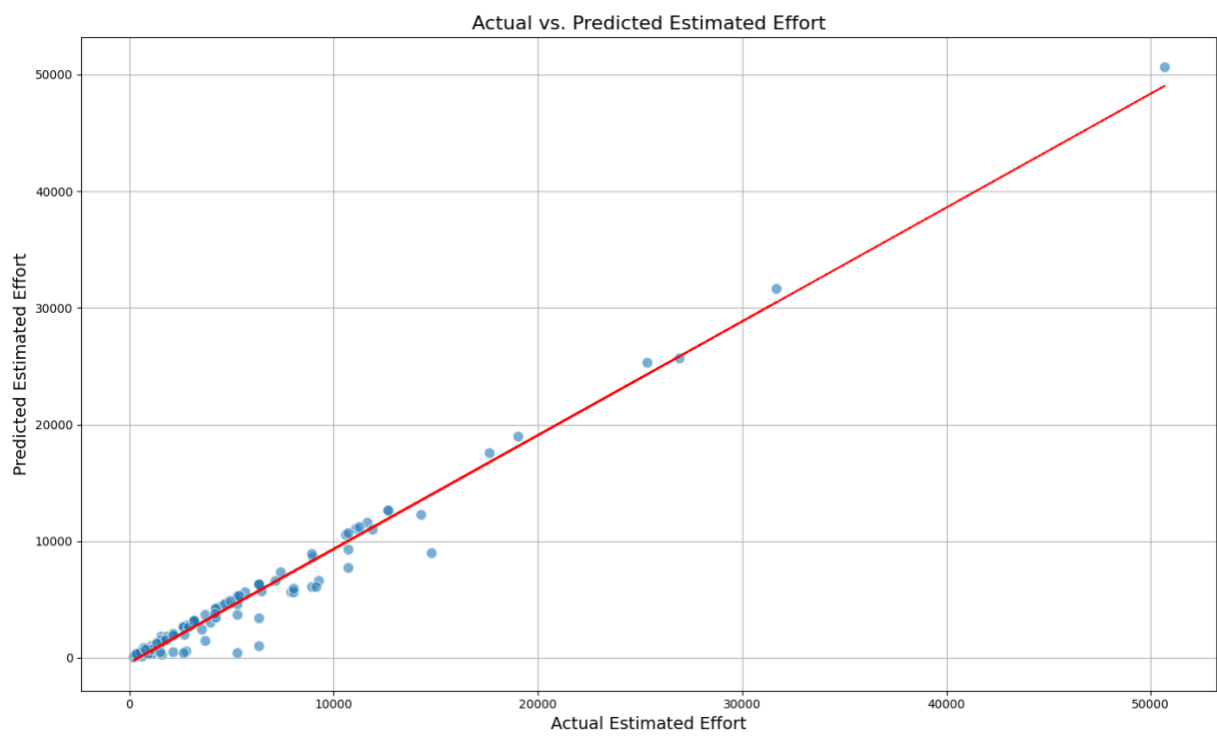
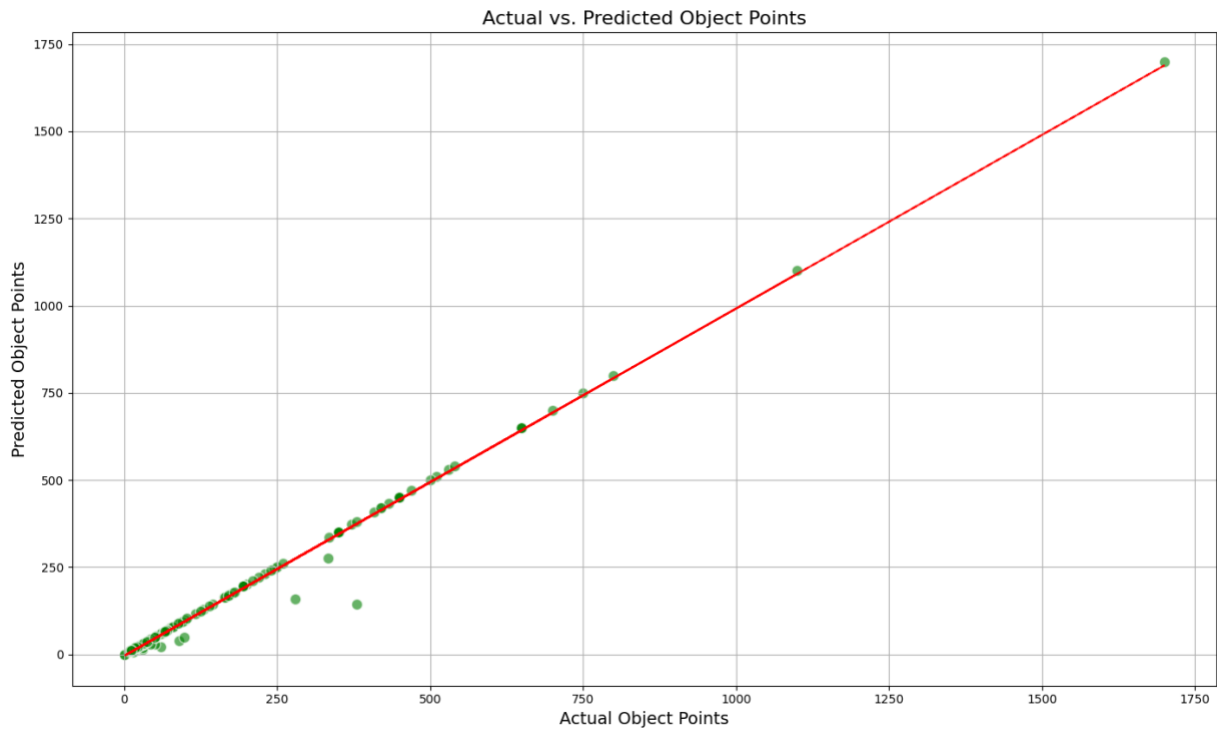
It signifies that approximately 94.1% of the variability in our predicted object points can be attributed to the model's capabilities.

Similarly, estimated effort indicates a remarkable 97.0% accuracy in predicting effort estimation.

We can determine from these results that our model is reliable and demonstrates precision.

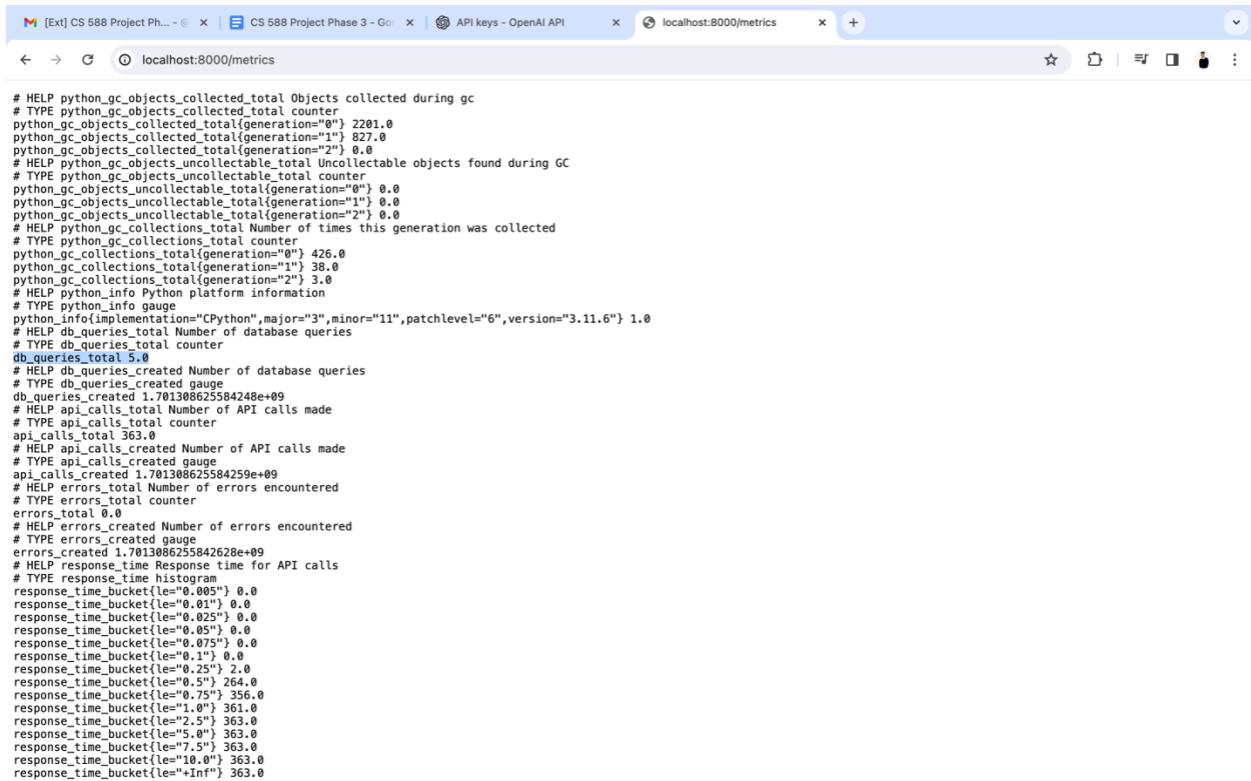
Scatter Plots

Our code generates scatter plots, which visualize the relationship between actual and predicted values for both object points and estimated effort, providing an assessment of our model's performance.

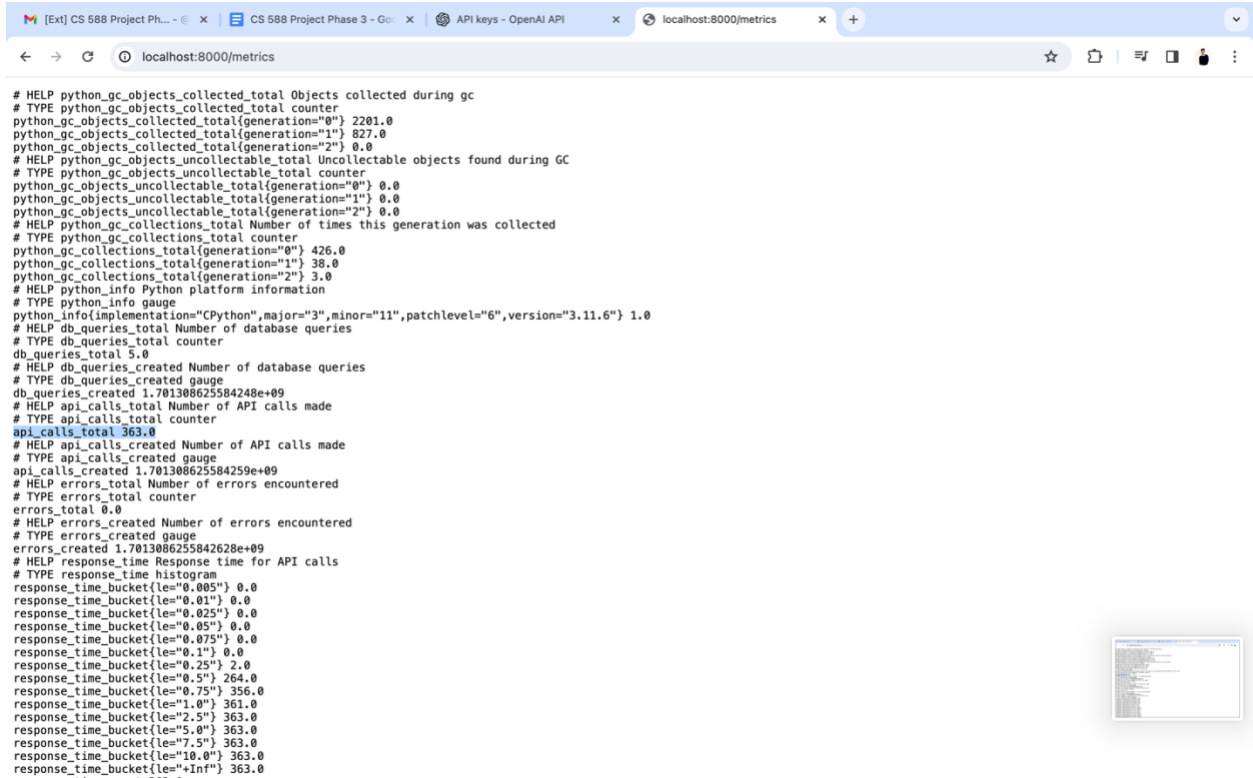


Monitoring with Prometheus

It ensure the reliability of our system, we have integrated Prometheus for monitoring OpenAI GPT-3 API calls, as also taught in the course. Prometheus enabled real-time tracking of key metrics, such as response time, error counts, API calls, and the count of SQL queries.



```
# HELP python_gc_objects_collected_total Objects collected during gc
# TYPE python_gc_objects_collected_total counter
python_gc_objects_collected_total{generation="0"} 2201.0
python_gc_objects_collected_total{generation="1"} 827.0
python_gc_objects_collected_total{generation="2"} 0.0
# HELP python_gc_objects_uncollectable_total Uncollectable objects found during GC
# TYPE python_gc_objects_uncollectable_total counter
python_gc_objects_uncollectable_total{generation="0"} 0.0
python_gc_objects_uncollectable_total{generation="1"} 0.0
python_gc_objects_uncollectable_total{generation="2"} 0.0
# HELP python_gc_collections_total Number of times this generation was collected
# TYPE python_gc_collections_total counter
python_gc_collections_total{generation="0"} 426.0
python_gc_collections_total{generation="1"} 38.0
python_gc_collections_total{generation="2"} 3.0
# HELP python_info Python platform information
# TYPE python_info gauge
python_info{implementation="CPython",major="3",minor="11",patchlevel="6",version="3.11.6"} 1.0
# HELP db_queries_total Number of database queries
# TYPE db_queries_total counter
db_queries_total 5.0
# HELP db_queries_created Number of database queries
# TYPE db_queries_created gauge
db_queries_created 1.701308625584248e+09
# HELP api_calls_total Number of API calls made
# TYPE api_calls_total counter
api_calls_total 363.0
# HELP api_calls_created Number of API calls made
# TYPE api_calls_created gauge
api_calls_created 1.701308625584259e+09
# HELP errors_total Number of errors encountered
# TYPE errors_total counter
errors_total 0.0
# HELP errors_created Number of errors encountered
# TYPE errors_created gauge
errors_created 1.7013086255842628e+09
# HELP response_time Response time for API calls
# TYPE response_time histogram
response_time_bucket{le="0.005"} 0.0
response_time_bucket{le="0.01"} 0.0
response_time_bucket{le="0.025"} 0.0
response_time_bucket{le="0.05"} 0.0
response_time_bucket{le="0.075"} 0.0
response_time_bucket{le="0.1"} 0.0
response_time_bucket{le="0.25"} 2.0
response_time_bucket{le="0.5"} 264.0
response_time_bucket{le="0.75"} 356.0
response_time_bucket{le="1.0"} 361.0
response_time_bucket{le="2.5"} 363.0
response_time_bucket{le="5.0"} 363.0
response_time_bucket{le="7.5"} 363.0
response_time_bucket{le="10.0"} 363.0
response_time_bucket{le="+Inf"} 363.0
```



```
# HELP python_gc_objects_collected_total Objects collected during gc
# TYPE python_gc_objects_collected_total counter
python_gc_objects_collected_total{generation="0"} 2201.0
python_gc_objects_collected_total{generation="1"} 827.0
python_gc_objects_collected_total{generation="2"} 0.0
# HELP python_gc_objects_uncollectable_total Uncollectable objects found during GC
# TYPE python_gc_objects_uncollectable_total counter
python_gc_objects_uncollectable_total{generation="0"} 0.0
python_gc_objects_uncollectable_total{generation="1"} 0.0
python_gc_objects_uncollectable_total{generation="2"} 0.0
# HELP python_gc_collections_total Number of times this generation was collected
# TYPE python_gc_collections_total counter
python_gc_collections_total{generation="0"} 426.0
python_gc_collections_total{generation="1"} 38.0
python_gc_collections_total{generation="2"} 3.0
# HELP python_info Python platform information
# TYPE python_info gauge
python_info{implementation="CPython",major="3",minor="11",patchlevel="6",version="3.11.6"} 1.0
# HELP db_queries_total Number of database queries
# TYPE db_queries_total counter
db_queries_total 5.0
# HELP db_queries_created Number of database queries
# TYPE db_queries_created gauge
db_queries_created 1.701308625584248e+09
# HELP api_calls_total Number of API calls made
# TYPE api_calls_total counter
api_calls_total 363.0
# HELP api_calls_created Number of API calls made
# TYPE api_calls_created gauge
api_calls_created 1.701308625584259e+09
# HELP errors_total Number of errors encountered
# TYPE errors_total counter
errors_total 0.0
# HELP errors_created Number of errors encountered
# TYPE errors_created gauge
errors_created 1.701308625584262e+09
# HELP response_time Response time for API calls
# TYPE response_time histogram
response_time_bucket{le="0.005"} 0.0
response_time_bucket{le="0.01"} 0.0
response_time_bucket{le="0.025"} 0.0
response_time_bucket{le="0.05"} 0.0
response_time_bucket{le="0.075"} 0.0
response_time_bucket{le="0.1"} 0.0
response_time_bucket{le="0.25"} 2.0
response_time_bucket{le="0.5"} 264.0
response_time_bucket{le="0.75"} 356.0
response_time_bucket{le="1.0"} 361.0
response_time_bucket{le="2.5"} 363.0
response_time_bucket{le="5.0"} 363.0
response_time_bucket{le="7.5"} 363.0
response_time_bucket{le="10.0"} 363.0
response_time_bucket{le="+Inf"} 363.0
```

4. List of tools used to conduct/develop the experiments

Development Environment

Visual Studio Code:

We used it for our code development, as it provides an integrated development environment.

Database Management

PgAdmin:

We utilized PostgreSQL administration and management, enabling us to have seamless database interactions for our model. It stores the predicted values for effort and object points, and furthermore, our model fetches data for various prompts.

PostgreSQL:

Serving as the relational database management system for storing and retrieving experiment data,

Machine Learning and Natural Language Processing

OpenAI GPT-3 API:

We integrated it to leverage its advanced natural language processing capabilities for project analysis and estimation. Also, our project is based on using generative AI.

Data Processing and Analysis

Pandas:

It is a data manipulation library in Python; we utilized it for loading, processing, and analyzing our data.

SQL Query Generation and Execution

SQLAlchemy:

We created an SQLAlchemy engine for interacting with PostgreSQL, enabling the execution of our SQL queries.

Visualization

Matplotlib:

We implemented it for data visualization, generating scatter plots to depict relationships between actual and predicted values.

Metric Tracking and Monitoring

Prometheus:

We have integrated metrics for real-time monitoring of API calls, allowing the tracking of key metrics such as response time, SQL query calls, and error counts.

5. Data sources that you have used for your final project.

The SEERA Software Cost Estimation Dataset - <https://zenodo.org/records/3987969>

We decided to use the SEERA Software Cost Estimation Dataset as the foundation for this project. The dataset has a comprehensive collection of data from 120 software development projects. The dataset provides a well-established foundation for our analysis, and with the given attributes, it suits our project requirements the best.

One factor influencing our choice was also the given actual and estimated effort values for each project. The dataset enabled us to conduct a thorough exploration of estimation. The presence of formulas and estimates within the dataset further enhances its usability, allowing us to derive meaningful insights and develop models for accurate project estimation.

Following are the general attributes present in the dataset

							General Information	
ProjID	Year of project	Organization id	Organization type	Role in organization	Size of organization	Size of IT department		
1	2015	1	1	1	16	7		
2	2016	25	5	5	2	1		
3	2008	2	5	3	2	2		
4	2009	42	4	2	3	2		
5	2016	42	4	2	3	2		
6	2012	42	4	2	3	2		
7	2016	42	4	1	3	1		

projID	Year of project				Organization id	Organization type	Role in organization
	Contract Date	Contract software delivery date	Actual software development	Year of project (Main)			
1	?	?	00/2015	2015	1	1	1
2	05/2016	05/2016	05/2016	2016	25	5	1
3	06/2008	03/2009	09/2008	2008	2	5	3
4	06/2010	01/2010	06/2009	2009	42	4	2
5	02/2016	2/1/2018	02/2016	2016	42	4	2
6	12/2012	05/2013	12/2012	2012	42	4	2
7	00/2016	00/2016	00/2016	2016	42	4	1
8	07/2018	12/2018	07/2018	2018	42	4	4
9	00/2018	00/2018	00/2018	2018	42	4	4

The attributes we used for estimating object points

projID	Object Points			Other Sizing Method	Estimated Size
	Number of screens	Number of reports	Object Points (Main)		
1	80	20	100	2	N/A
2	11	8	19	2	N/A
3	64	0	64	3	50
4	150	96	246	5	70
5	310	110	420	5	120
6	110	67	177	5	76
7	20	34	54	5	5
8	40	31	71	5	50
9	10	4	14	5	5
10	65	270	335	4	6

The attributes we used for estimating the effort

projID	Estimated effort						Actual effort	
	Estimated duration	Team size	Dedicated team	Daily working hours	Monthly working hours	Estimated Effort (Main)	Actual Duration	Actual effort (Main)
1	2	6	6	8	176	2112	3	3168
2	2	3	3	8	176	1056	3	1584
3	3	6	6	8	176	3168	5	5280
4	6	6	4	8	176	5280	6	5280
5	12	9	9	8	176	19008	24	38016
6	6	7	7	8	176	7392	8	9856
7	6	5	5	8	176	5280	9	7920
8	4	5	5	10	220	4400	4	4400
9	6	4	4	8	176	4224	6	4224
10	12	4	3	7	154	6468	45	24255

Below is our database as implemented in PgAdmin as Testing data

organization	numberscreens	numberofreports	estimatedduration	teamsize	dedicatedteammembers	dailyworkinghours	monthlyworkinghours	actualobjectpoints
16	30	20	3	3	3	8	176	
4	150	200	4	6	6	12	264	
4	25	25	12	6	2	6	132	
16	50	30	1	2	1	8	176	
1	130	70	3	4	1	7	154	
1	180	50	5	2	2	5	110	
1	80	140	5	2	0	10	220	
2	55	15	20	2	1	8	176	
2	25	15	3	2	1	5	110	
2	37	25	7	1	0	8	176	
6	441	70	7	10	5	8	176	
4	50	26	4	3	3	8	176	
17	5	2	6	3	3	4	88	
3	50	10	4	2	1	8	176	
16	180	350	12	12	12	8	176	
16	200	150	6	9	8	8	176	
16	150	230	6	10	10	8	176	
13	46	24	2	3	3	8	176	
16	20	10	12	5	0	8	176	
4	150	200	3	6	6	12	264	
3	350	83	6	2	0	8	176	
3	30	0	7	4	4	5	110	
3	90	40	3	5	5	5	110	
16	10	5	6	5	5	7	154	
16	44	7	6	12	12	7	154	

Total rows: 120 of 120 Query complete 00:00:00.059 Ln 1, Col 1

6. Executive summary of your final research report

Our final project represents our efforts to integrate generative AI, particularly OpenAI's GPT-3 API, with our focus area of project management.

Our primary objective was to use the power of generative AI to automate various parts of project management, including the estimation of project plans, design documents, source code, test plans, test cases, and user API documentation.

Our project utilized the SEERA Software Cost Estimation Dataset to use the capabilities of our generative AI model. The integration of PostgreSQL as a database provided a foundation for storing and managing project data.

Key Features and Processes

Data Source:

The SEERA dataset served as our primary data source. Preprocessing and analysis ensured the quality and relevance of the data for training and evaluation.

OpenAI Integration:

We seamlessly integrated OpenAI's GPT-3 into our project, leveraging its advanced natural language processing to generate human-like text based on prompts.

Prompt Engineering:

The generative AI model underwent training on historical data, learning patterns, and relationships. Prompt engineering involved crafting effective prompts for accurate information extraction.

Estimation:

Our model successfully predicted estimates for the duration and object points.

Accuracy and Evaluation:

R^2 scores for both object points and estimated effort. The high R^2 scores (0.941 and 0.970, respectively) underscore the model's precision and reliability.

Monitoring and Metrics:

Prometheus integration ensured real-time monitoring of key metrics, including OpenAI GPT-3 API calls, response time, error counts, and SQL queries.