Evaluating CLARA (Code expLAiner & Repository Analyzer)

RESEARCH GOAL AND PROCEDURE

The goal of this study is to evaluate **CLARA**, a Chrome-based browser extension for code explanation and analysis in GitHub repositories. If you decide to participate, the study will last about 30 minutes in which you will be asked to use the three features provided by the tool (i.e., code explanation, code refactoring, and prediction of code quality attributes) and answer a set of questions about your experience using CLARA. You will answer the questions through this online questionnaire (*aka* survey).

PARTICIPATION AND CONFIDENTIALITY

You must be at least 18 years old to participate.

Any information that is obtained in connection with this study and that can be identified with you will remain confidential and will be disclosed only with your permission. Your responses to this study will be kept confidential and all data will be kept secure. Your decision whether or not to participate will not prejudice your future relations with Institute of Information Technology, University of Dhaka. If you decide to participate, you are free to withdraw your consent and to discontinue participation at any time without penalty.

CONTACT

If you have any questions, please ask us at any moment. If you have any additional questions later, Ahmed Adnan (bsse1131@iit.du.ac.bd, +8801813865290), Mushfiqur Rahman (bsse1130@iit.du.ac.bd, +8801718142322), Saad Sakib Noor (bsse1122@iit.du.ac.bd, +8801887-431603) and Dr. Kazi Muheymin-Us-Sakib (sakib@iit.du.ac.bd, +8801730-051232), will be happy to answer them.

CONSENT

YOU ARE MAKING A DECISION WHETHER OR NOT TO PARTICIPATE.
IF YOU WANT TO PARTICIPATE, PLEASE ENTER YOUR EMAIL IN THE TEXT FIELD BELOW, AND START THE SURVEY.

* Indicates required question

1. Please Enter your Email

Overview of CLARA

CLARA is a Chrome-based browser extension that assists developers and researchers into various tasks while exploring and analyzing open-source GitHub repositories. CLARA is available on the Chrome Web Store and can be easily used on any Chrome-supported browser (e.g., Google Chrome, Brave). CLARA provides the following features:

1. Code Explanation: When a user is exploring a code file in the project repository, CLARA can provide explanation of the entire code file in the context of the project repository. Additionally, if a user needs to understand a specific part/portion of the code file, he/she can mark that part and the tool will explain that part in the context of the entire code file.

2. Code Refactoring: This feature provides the refactored version of any code file within a repository.

A user can utilize this feature to view a cleaner, more efficient version of the code. Moreover, he/she can incorporate the refactored output/portion of output in their own projects more efficiently by reusing this clean, modular code.

3. Detection of Code Quality Attributes: This feature detects some useful attributes about the quality of a code file such as potential cyclomatic complexity, vulnerabilities according to CVE and CWE standards, and maintainability index to give a user an idea about the code's overall risk, readability, and ease of future modification.

Additionally, CLARA provides an **AI-based chatbot** for each of these features so that a user can ask questions, do further exploration, analysis and receive suitable contextual responses.

CLARA is a prototype that provides explanations, suggestions and guides developers and researchers in performing the above tasks effectively.

Study Overview

In this study, we aim to assess each of the CLARA's three features, i.e., Code Explanation, Code Refactoring, and Detection of Code Quality Attributes. For each feature, you will explore and analyze 2 code files on a GitHub repository and evaluate CLARA's explanations and suggestions given for that feature. Additionally, you can explore CLARA's AI-based chatbot for each of these features to satisfy your inquiries and gain further insights.

You are provided with the URL of CLARA, which you will need to install in a Chrome-based browser. Then, you will need to go to the 2 mentioned GitHub repositories to explore and analyze 6 code files using CLARA (two code files per feature). **Repository-1** will be used to assess the Code Explanation feature, while **Repository-2** will be used to assess the Code Refactoring and Code Quality Attributes Detection features. For each feature, you will assess CLARA's explanations and suggestions with the sample answers. In addition, you can gain further context by asking follow up questions and any of your inquiries to CLARA's AI-based chatbot for any of the features. After that, based on your experience with CLARA, you will answer a set of questions to give us feedback on how well the tool performs and give us suggestions to improve it.

In summary, you will need to perform the following steps (later in this study):

1. First install CLARA in a **Chromium engine-based browser (e.g., Google Chrome, Brave, Microsoft Edge)** using the following URL:

[CLICK HERE] to install **CLARA**

2. Then you just need to:

Go to **Repository-1:** used for evaluating the 'Code Explanation' feature.

Go to **Repository-2:** used for evaluating Code Refactoring and Detection of Code Quality Attributes Features**.**

3. For each feature, you will need to perform the following:

(a) Visit the mentioned code file. (detailed instructions are provided later)

(b) Perform the feature mentioned in the survey for that particular code file. This step will mimic the scenario when a user explores the code files in an open-source GitHub repository and analyze them for different purposes.

(c) Assess the explanations and suggestions suggested by CLARA. Feel free to take notes of

the suggested explanations, code quality attributes and refactored versions.

(d) Answer the questions about CLARA's behavior and functionality.

3. After that, you will need to answer some questions about your **Overall Experience** with CLARA and your feedback about its behavior. Lastly, you will need to answer some questions regarding your **Professional Background**.

Evaluating the effectiveness of CLARA's Features

Code Explainer

Repository-1 is a GitHub repository about an Image Processing Library made in pure Java. Your task will be to visit some of the code files of this repository and analyze CLARA's provided explanation summary of these code files and CLARA's explanation of the selected/marked code fragments.

PLEASE COMPLETE THE FOLLOWING INSTRUCTIONS:

Analysis of Code File 1:

1. Go to Repository-1 and open the file using the following path ***'src/main/java/com/statistics/HistogramStatistics.java'*** or simply click here Code File 1 **.**

2. After opening the code file, you will see a pop-up appear on the top-right of your screen with 2 buttons. Click the **'Explain Full Code File'** button in the popup to get an explanation summary of the code file.
[n.b.- If you close the popup or anything happens, just reload the page and the popup will appear again]

3. An explanation summary will appear in the popup. Observe the text and feel free to take notes on the explanation provided by CLARA.

4. Assess the explanation summary provided by CLARA for the given code file. Please also take the file's context in the repository scenario into consideration while assessing. A sample explanation summary and code file's context is given below for your comparison. Additionally, you can use the chatbot under the explanation summary to ask further questions or gain insights about the code file and the summary.

Context of the Code File 1 in terms of the Repository:

This Project is `Catalano Imaging Library`, which is used for processing image pixel intensity histograms.The `HistogramStatistics.java` file, which is in `src/main/java/com/statistics/`*, is a utility class providing static methods for statistical analysis of histogram data.*

A sample explanation summary for Code File 1:

Summary of Statistical Utility Functions:

This utility provides statistical analysis methods for numeric datasets (e.g., histograms). It includes functions to:

- *Calculate Entropy: Measures the disorder or uncertainty in the dataset.*
- *Determine Range Around Median: Identifies the range containing a specified percentage of values centered around the median.*
- *Compute Kurtosis: Assesses the "tailedness" of the distribution.*
- *Find Mode: Returns the most frequent value in the dataset.*
- *Calculate Skewness: Measures the asymmetry of the data distribution, with versions accepting precomputed mean and standard deviation.*
- *Calculate Standard Deviation: Measures the dispersion of values, with optional use of a provided mean.*

These methods help characterize data distributions in terms of shape, spread, and central tendency.

---

5. Now, Click '**Back**' button in the popup and please select/mark the following code fragment in the same code file using your mouse cursor [***Line 80-97] (click and hold the left mouse button from the start to the end of the code fragment in the code file to mark/select that specific portion)***:

```
public static double Kurtosis(int[] values, double mean, double stdDeviation){
    double n = 0;
    for (int i = 0; i < values.length; i++)
       n += values[i];

    double part1 = n * (n + 1);
    part1 /= ((n - 1) * (n - 2) * (n - 3));

    double part2 = 0;
    for (int i = 0; i < values.length; i++) {
       part2 += Math.pow((i - mean) / stdDeviation, 4) * values[i];
    }

    double part3 = 3 * Math.pow((n - 1), 2);
    part3 /= (n - 2) * (n - 3);
```

```
    return part1 * part2 - part3;
}
```

---

6. Now click the **'Explain Marked/Selected Code'** button. CLARA will provide an explanation of what that selected/marked code fragment does in the context of the code file. Please assess the explanation provided by CLARA. A sample answer is given below for your comparison and you can also ask follow up inquiries to the chatbot.
[n.b.- If you close the popup or anything happens, just reload the page and the popup will appear again]

A sample explanation of the code fragment:

The Kurtosis method in HistogramStatistics.java computes the kurtosis of a discrete distribution using histogram data. It accepts an integer array values, representing frequency counts for each bin, along with the distribution's mean and standard deviation. The method begins by calculating the total number of observations, n, by summing the array. It then calculates part1, a scaling factor derived from n, and part2, which accumulates the weighted fourth power of each bin's standardized distance from the mean. part3 is a correction term to adjust for bias in sample-based kurtosis. The final result, part1 * part2 - part3, yields the excess kurtosis, indicating whether the distribution is more peaked or flatter than a normal distribution.

Analysis of Code File 2:

1. Go to **Repository-1** and open the file using the following path '***src/main/java/com/math/ComplexNumber.java***' or simply click here **Code File 2** **.**

2. After opening the code file, you will see a popup appear on the top-right of your screen with 2 buttons. Click the **'Explain Full Code File'** button in the popup to get an explanation summary of the code file.
[n.b.- If you close the popup or anything happens, just reload the page and the popup will appear again]

3. An explanation summary will appear in the popup. Observe the text and feel free to take notes on the explanation provided by CLARA.

4. Assess the explanation summary provided by CLARA for the given code file. Please also take the file's context in the repository scenario into consideration while assessing. A sample explanation summary and code file's context is given below for your comparison. Additionally, you can use the chatbot under the explanation summary to ask further questions or gain insights about the code file and the summary.

Context of the Code File 2 in the Repository:

This Project is `Catalano Imaging Library`, which is used for processing image pixel intensity histograms. The ComplexNumber.java file provides a core class for representing and working with complex numbers, which encapsulates the real and imaginary parts of a complex number using double-precision floating-point values.

A sample explanation summary for Code File 2:

This utility provides functions for performing arithmetic operations on complex numbers, including,

- Magnitude: Retrieves the magnitude (absolute value) of a complex number.
- Squared Magnitude: Returns the square of the magnitude (used for performance optimization in some cases).
- Phase: Computes the phase (angle) of the complex number in radians.
- Real Part: Extracts the real component of a complex number.
- Imaginary Part: Extracts the imaginary component of a complex number.
- Swap Real and Imaginary: Interchanges the real and imaginary parts of a complex number.
- Division (Complex ÷ Complex): Returns the result of dividing one complex number by another.
- Division (Complex ÷ Scalar): Divides a complex number by a scalar value.
- Power: Raises a complex number to a given integer power.

These functions enable comprehensive manipulation and analysis of complex numbers for mathematical and engineering applications.

5.

Now, Click **'Back'** button in the popup and please select/mark the following code fragment in the same code file using your mouse cursor ***[Line 248-256]*** **(click and hold the left mouse button from the start to the end of the code fragment in the code file to mark/select that specific portion)**:

```
public void Pow(double n) {
    double norm = Math.pow(getMagnitude(), n);
    double angle = 360 - Math.abs(Math.toDegrees(Math.atan(this.imaginary / this.real)));

    double common = n * angle;

    this.real = norm * Math.cos(Math.toRadians(common));
    this.imaginary = norm * Math.sin(Math.toRadians(common));
  }
```

---

6. Now click the **'Explain Marked/Selected Code'** button. CLARA will provide an explanation of what the selected/marked code fragment does in the context of the code file. Please assess the explanation provided by CLARA. A sample answer is given below for your comparison and you can also ask follow up inquiries to the chatbot.
[n.b.- If you close the popup or anything happens, just reload the page and the popup will appear again]

A sample explanation of the code fragment:

The given code defines an instance method Pow(double n) within the ComplexNumber class. Its function is to raise the current complex number to the power of n (a real number). It then updates the object's real and imaginary components to represent the resulting complex value.

Now, please answer the following questions:

2. After reviewing CLARA's suggestions, do you think the tool provided accurate and helpful summaries and explanations for the overall code as well as for individual code fragments? *

Mark only one oval.

- Yes
- No
- Unsure

3. Please explain your answer (why yes, no, or unsure?) *

4. Do you agree that CLARA effectively captures the broader context of a code file within its repository in its explanation summary?

Mark only one oval.

- Completely agree
- Somewhat agree
- Neutral
- Somewhat disagree
- Completely disagree

5. How easy is it to understand CLARA’s explanations of the code files and the selected code fragments within the files? *

Mark only one oval.

- ◯ Very easy
- ◯ Moderately easy
- ◯ Neutral
- ◯ Moderately difficult
- ◯ Very difficult

6. Please elaborate on your answer (Optional):

Evaluating the Effectiveness of CLARA's Features

Code Refactoring

Repository-2 is a GitHub repository that contains a couple of code files. Your task will be to visit some of the code files of this repository and analyze their refactored version provided by CLARA and evaluate their potential of reusability.

PLEASE COMPLETE THE FOLLOWING INSTRUCTIONS:

Analysis of Code File 3:

1. Go to

Repository-2 and open the file using the following path ***'original_refactor1.java'*** or simply click **Code File 3**.

2. After opening the code file, please click on the **'Refactor the Code'** button in the popup.
[n.b.- If you close the popup or anything happens, just reload the page and the popup will appear again]

3. A clean and refactored code will appear in the popup. The code will contain explanation comments on refactored areas, which will give you a better idea about how refactoring was done and where it happened. Observe the code and the comments and feel free to take notes.

4. Assess the refactored code and comment details provided by CLARA. A sample refactored version of the code is given below with refactoring details for your comparison. Additionally, you can use the chatbot under CLARA's provided answer to ask further questions or gain insights about the code and the refactoring process.

A sample refactored version for Code File 3:

```
public class StatisticsUtil {

  public static List<ResultRow> executeQuery(String template, Map<String, String> params) {
    StringSubstitutor stringSubstitutor = new StringSubstitutor(params);
    String sql = stringSubstitutor.replace(template);
    return execStatisticQuery(sql);
  }

  public static void execUpdate(String template, Map<String, String> params) throws Exception {
    StringSubstitutor stringSubstitutor = new StringSubstitutor(params);
```

```
        String sql = stringSubstitutor.replace(template);
        // Add logic to execute the update query here
    }

    public static List<ResultRow> execStatisticQuery(String sql) {
        if (!FeConstants.enableInternalSchemaDb) {
            return Collections.emptyList();
        }

        try (AutoCloseConnectContext r = buildConnectContext()) {
            if (Config.isCloudMode()) {
                r.connectContext.getCloudCluster(); // Placeholder for actual usage
            }
            // Add logic to execute the query and return results
            return Collections.emptyList(); // Placeholder
        }
    }

    public static AutoCloseConnectContext buildConnectContext() {
        return buildConnectContext(false);
    }

    public static AutoCloseConnectContext buildConnectContext(boolean limitScan) {
        ConnectContext connectContext = new ConnectContext();
        SessionVariable sessionVariable = connectContext.getSessionVariable();
        sessionVariable.internalSession = true;

        connectContext.setQualifiedUser(UserIdentity.ROOT.getQualifiedUser());
        connectContext.setCurrentUserIdentity(UserIdentity.ROOT);
        connectContext.setStartTime();

        return new AutoCloseConnectContext(connectContext);
    }
}
```

What refactoring changes were made:

- ***Removed duplicate and malformed method declarations (execUpdate, deserializeToHistogramStatistics, buildConnectContext).***
- ***Organized method bodies properly and completed unfinished logic.***
- ***Ensured single responsibility per method and applied naming clarity.***
- ***Fixed structural and syntax errors for compilation.***

Analysis of Code File 4:

1. Go to **Repository-2** and open the file using the following path ***'original_refactor2.java'*** or simply click **Code File 4**.

2. After opening the code file, please click on the **'Refactor the Code'** button in the popup.
[n.b.- If you close the popup or anything happens, just reload the page and the popup will appear again]

3. A clean and refactored code will appear in the popup. The code will contain explanation comments on refactored areas, which will give you a better idea about how refactoring was done and where it happened. Observe the code and the comments and feel free to take notes.

4. Assess the refactored code and comment details provided by CLARA. A sample refactored version of the code is given below with refactoring details for your comparison. Additionally, you can use the chatbot under CLARA's provided answer to ask further questions or gain insights about the code and the refactoring process.

A sample refactored version for Code File 4:

```
public class NereidsSqlCacheManager {
    // value: SqlCacheContext
    private volatile Cache<String, SqlCacheContext> sqlCaches;

    public NereidsSqlCacheManager(int sqlCacheNum, long cacheIntervalSeconds) {
        sqlCaches = buildSqlCaches(sqlCacheNum, cacheIntervalSeconds);
    }

    public static synchronized void updateConfig() {
        Cache<String, SqlCacheContext> updatedCaches = buildSqlCaches(
                Config.sql_cache_manage_num,
                Config.cache_last_version_interval_second
        );
        updatedCaches.putAll(sqlCacheManager.sqlCaches.asMap());
        sqlCacheManager.sqlCaches = updatedCaches;
    }

    private static Cache<String, SqlCacheContext> buildSqlCaches(int sqlCacheNum, long
cacheIntervalSeconds) {
```

```
        sqlCacheNum = sqlCacheNum < 0 ? 100 : sqlCacheNum;
        cacheIntervalSeconds = cacheIntervalSeconds < 0 ? 30 : cacheIntervalSeconds;

        return Caffeine.newBuilder()
                .maximumSize(sqlCacheNum)
                .expireAfterAccess(Duration.ofSeconds(cacheIntervalSeconds))
                .softValues() // Auto-evict when JVM memory is low
                .build();
    }

    public Optional<LogicalSqlCache> tryParseSql(ConnectContext connectContext, String sql) {
        // Placeholder for implementation
        return Optional.empty();
    }

    private boolean tablesOrDataChanged(Env env, SqlCacheContext sqlCacheContext) {
        long latestPartitionTime = sqlCacheContext.getLatestPartitionTime();
        long latestPartitionVersion = sqlCacheContext.getLatestPartitionVersion();

        if (sqlCacheContext.hasUnsupportedTables()) {
            return true;
        }

        for (ScanTable scanTable : sqlCacheContext.getScanTables()) {
            long cacheTableTime = scanTable.latestTimestamp;
            long cacheTableVersion = scanTable.latestVersion;

            OlapTable olapTable = env.getTable(scanTable.tableId);
            long currentTableTime = olapTable.getVisibleVersionTime();
            long currentTableVersion = olapTable.getVisibleVersion();

            if (currentTableTime > cacheTableTime ||
                    (currentTableTime == cacheTableTime && currentTableVersion > cacheTableVersion))
{
                return true;
            }

            for (Long scanPartitionId : scanTable.getScanPartitions()) {
                Partition partition = olapTable.getPartition(scanPartitionId);
                if (partition == null || partition.getVisibleVersionTime() > latestPartitionTime ||
                        (partition.getVisibleVersionTime() == latestPartitionTime &&
                                partition.getVisibleVersion() > latestPartitionVersion)) {
                    return true;
                }
```

```
            }
        }
        return false;
    }
}
```

What refactoring changes were made:

- ***Removed duplicate and incomplete method declarations (updateConfig, tablesOrDataChanged, tryParseSql).***
- ***Organized method scopes and logic blocks properly for readability and correctness.***
- ***Applied clear method separation and consistent formatting.***
- ***Ensured safe use of shared state (sqlCaches) and added null/validation safety hints.***

Now, please answer the following questions:

7. After reviewing CLARA's suggestions, do you think the tool provided useful and reusable refactored version of the code? *

   Mark only one oval.

   - ◯ Yes
   - ◯ No
   - ◯ Unsure

8. Please explain your answer (why yes, no, or unsure?) *

9. How clear and understandable do you find the refactored code provided by CLARA compared to the original version? *

Mark only one oval.

- ◯ Very clear and understandable
- ◯ Moderately clear and understandable
- ◯ Neutral
- ◯ Moderately unclear and difficult to understand
- ◯ Very unclear and difficult to understand

10. Please elaborate on your answer (Optional):

Evaluating the Effectiveness of CLARA's Features

Detection of Code Quality Attributes

Repository-2 is a GitHub repository that contains a couple of code files. Your task will be to visit some of the code files of this repository and analyze the code quality attributes detected by CLARA and evaluate how these metrics can assist users in a safer reuse of code and better code maintainability.

PLEASE COMPLETE THE FOLLOWING INSTRUCTIONS:

Analysis of Code File 5:

1. Go to Repository-2 and open the file using the following path ***"file_1.c'*** or simply click Code File 5.

2. After opening the code file, please click on the **'See Code Quality Attributes'** button in the popup.
[n.b.- If you close the popup or anything happens, just reload the page and the popup will appear again]

3. Cyclomatic complexity, Maintainability Index, Classification of Vulnerabilities will appear on the popup. Observe the values and identified categories and feel free to take notes.

4. Assess the information provided by CLARA. Additionally, you can use the chatbot under CLARA's provided answer to ask further questions or gain insights about the calculation of these metrics.

A sample output for Code File 5:

Cyclomatic Complexity: 3
Vulnerabilities Classification: Overflow
Maintainability index: 80

Analysis of Code File 6:

1.

Go to Repository-2 and open the file using the following path ***"file_2.c'*** or simply click Code File

6.

2. After opening the code file, please click on the **'See Code Quality Attributes'** button in the popup.
[n.b.- If you close the popup or anything happens, just reload the page and the popup will appear again]

3. Cyclomatic complexity, Maintainability Index, Classification of Vulnerabilities Score will appear on the popup. Observe the values and identified categories and feel free to take notes.

4. Assess the information provided by CLARA. Additionally, you can use the chatbot under CLARA's provided answer to ask further questions or gain insights about the calculation of these metrics.

A sample output for Code File 6:

Cyclomatic Complexity: 3
Vulnerabilities Classification: Information Leak
Maintainability index: 75

Now, please answer the following questions:

11. After reviewing CLARA's suggestions, do you think the tool provided accurate predictions of code quality attributes? *

Mark only one oval.

- ◯ Yes
- ◯ No
- ◯ Unsure

12. Please explain your answer (why yes, no, or unsure?) *

13. How helpful do you find the detected code quality attributes provided by CLARA? *

Mark only one oval.

- Very helpful
- Moderately helpful
- Neutral
- Moderately unhelpful
- Very unhelpful

14. Please elaborate on your answer (Optional):

Evaluating the Overall Experience with CLARA

15. How easy or difficult did you find CLARA to use, in terms of its overall user-friendliness? *

Mark only one oval.

- ◯ Very easy
- ◯ Moderately easy
- ◯ Neutral
- ◯ Moderately difficult
- ◯ Very difficult

16. Please provide any suggestions for improving CLARA's Graphical User Interface (If any)

17. Overall, how accurate were the suggestions for CLARA's three features? *

Mark only one oval.

- ◯ Very accurate
- ◯ Moderately accurate
- ◯ Neutral
- ◯ Moderately inaccurate
- ◯ Very inaccurate

18. Please explain your answer (Optional):

19. How helpful do you think CLARA is to assist developers and researchers in exploring and analyzing open-source codebases? *

Mark only one oval.

- Very helpful
- Moderately helpful
- Neutral
- Moderately unhelpful
- Very unhelpful

20. Please specify the reason for your answer: *

21. Do you agree that CLARA can help developers and researchers save time and effort in analyzing code files and understanding code explanations while exploring open-source GitHub repositories, compared to using third-party generative AI models or IDE-based tools? *

Mark only one oval.

- Completely agree
- Somewhat agree
- Neutral
- Somewhat disagree
- Completely disagree

22. Do you agree that the combination of the three features in CLARA simplifies the code exploration and analysis workflow in open-source GitHub repositories? *

Mark only one oval.

- Completely agree
- Somewhat agree
- Neutral
- Somewhat disagree
- Completely disagree

23. How responsive is CLARA in providing suggestions for the three features? *

Mark only one oval.

- ◯ Very responsive
- ◯ Moderately responsive
- ◯ Neutral
- ◯ Moderately unresponsive
- ◯ Very unresponsive

24. What additional functionality/feature (if any) would you like to see in CLARA * in the future?

Check all that apply.

- ☐ AI-assisted bug detection in the code files & potential fixes
- ☐ Unit test generation
- ☐ Code documentation generation
- ☐ Analysis and summarization of commits and pull requests
- ☐ Cross-file dependency & reference mapping
- ☐ Other: _____

25. Please provide any (other) recommendations to improve CLARA (Optional):

_____

_____

_____

_____

_____

Demographic Questions

26. Please indicate your current professional affiliation or role: *

Check all that apply.

- [ ] Academic Researcher (e.g., professor, postdoctoral fellow, PhD/MSc student)
- [ ] Industry Developer / Practitioner (e.g., software engineer, technical lead)
- [ ] Other: _____

27. How many years of professional experience do you have in this role? *

_____