**PROJECT DEBUGGING**

**I. Attendance Monitoring Module**

**A. System Observable Errors**

**1. Incorrect Face Recognition Results:**

The system occasionally misidentifies or fails to recognize certain faces, leading to incorrect attendance marking. This issue can occur due to variations in lighting conditions, face angles, or poor image quality captured by the camera. These inaccuracies undermine the reliability of the attendance system.

**2. Database Connectivity Issues:**

There are instances where the system fails to connect to the real-time database, resulting in delays or an inability to update or retrieve user data. This problem often arises due to network instability or incorrect database configuration, leading to data synchronization issues.

**3. Performance Lag with Larger Datasets:**

As the number of registered users grows, the system experiences noticeable performance lags. The increased data volume causes slower processing times during face recognition and database queries, affecting the overall user experience and efficiency of the attendance system.

**B. Solution to Errors**

**1. Improving Face Recognition Accuracy:**

We addressed the issue of incorrect face recognition by retraining the model with a more diverse and extensive dataset that includes variations in lighting, angles, and facial expressions. Additionally, we implemented real-time image preprocessing techniques such as histogram equalization and Gaussian blurring to enhance image quality before recognition.

**2. Enhancing Database Connectivity:**

To resolve database connectivity issues, we optimized the system’s network configuration and ensured that all database credentials and settings were correctly implemented. We also incorporated a retry mechanism with exponential backoff to manage temporary network failures, ensuring consistent data synchronization even under unstable conditions.

**3. Optimizing Performance for Larger Datasets:**

We mitigated performance lags by optimizing the database queries and implementing indexing for faster data retrieval. Additionally, we introduced batch processing for the face recognition module, allowing the system to handle larger datasets more efficiently without compromising speed or accuracy.

**II. Library Module**

**I. System Observable Errors**

1. Unassigned Variable Error: In the first code, the `barcode\_reader () ` function is called, but the result is not assigned to a variable. This leads to an `UnboundLocalError` because `id` is used without being defined or initialized.

2. Recursive Call for Error Handling: The `return\_book () ` function calls itself recursively inside the `except` block when an error occurs. This can result in a stack overflow if the error persists, eventually causing the program to crash.

3. SQL Injection Vulnerability: The code uses string interpolation to insert variables directly into SQL queries. This exposes the system to potential SQL injection attacks, which can compromise the security of the database.

4. Missing Error Handling for Database Connection: There is no specific error handling for database connection issues, meaning that if the connection fails, the system might throw an exception without giving a user-friendly response or handling the error gracefully.

5. Potential `NoneType` Error: When fetching the `student\_number` and `book\_number` from the database, if there are no results, the code attempts to access index `0` of a `None` value, which will result in a `TypeError` or `IndexError`.

**II.Solution to Errors**

1. \*\*Proper Variable Assignment\*\*: In the updated code, the result of the `barcode\_reader()` function is assigned directly to the `library\_card` and `book\_to\_return` variables. This eliminates the unassigned variable issue from the first code, ensuring the program can proceed without `UnboundLocalError`.

2. \*\*Non-Recursive Error Handling\*\*: Instead of using recursion to restart the process when an error occurs, the second code ensures that errors are handled once and the function exits gracefully. This prevents potential stack overflow issues and allows for more controlled error recovery.

3. \*\*Parameterized SQL Queries\*\*: The updated version uses parameterized queries (using `%s` placeholders) instead of direct string interpolation for SQL queries. This approach mitigates the risk of SQL injection attacks by safely handling user input and preventing malicious queries.

4. \*\*Database Connection Error Handling\*\*: The second code includes a specific exception handler for `mysql.connector.Error`, which is triggered if there is a problem connecting to the database or executing a query. This ensures that database errors are caught and handled with an appropriate error message.

5. \*\*Proper Null Check for Database Results\*\*: The second version handles potential null values or missing data more robustly by checking if `user\_name`, `borrowed\_amount`, and `ebook\_status` are valid. This prevents `NoneType` or `IndexError` exceptions when accessing the results from the database, ensuring the program can handle missing or invalid data gracefully.

**III. Object Recognition**

**A. System Observable Errors**

The existing object recognition system faced significant challenges related to detection accuracy and handling overlapping objects. The system exhibited restrictions in its ability to accurately detect and recognize multiple objects, particularly when these objects were small, less distinct, or partially obscured. This issue was largely due to the use of an older version of the YOLO (You Only Look Once) algorithm, specifically YOLOv3, which has limitations in recognizing a wide variety of objects in complex environments. Additionally, the system struggled with objects that overlapped each other, often only recognizing the object in the foreground or failing to detect both objects entirely. This inability to handle overlapping objects was a major flaw, especially in scenarios where objects were close together, reducing the system’s overall effectiveness.

**B. Solution to Errors**

We can the system to a more advanced version of the YOLO algorithm, like YOLOv8 or YOLOv9, to improve detection accuracy and handle more objects with precision. Additionally, we added a Multiple Object Recognition feature to address the issue of overlapping objects, enabling the system to accurately detect and distinguish between multiple objects, even in complex scenarios. These improvements significantly enhanced the system's overall performance.

**IV. Speech for Recording and Announcing School Events**

**A. System Observable Errors**

1. **Ambient Noise Interference:**

The ADA speech recognition system struggles to accurately capture spoken words in environments with significant background noise. This interference can cause the system to misinterpret or completely miss certain words or phrases. As a result, users may need to repeat themselves multiple times, leading to frustration. The system's effectiveness is reduced in noisy conditions, which limits its usability in such environments.

1. **Accents and Pronunciations:**

Users with different accents or unique ways of pronouncing words often face difficulties with the ADA system. The speech recognition may not be trained on a wide enough variety of accents, causing it to incorrectly transcribe or fail to understand the input. This issue can lead to significant misunderstandings or errors in executing commands. The lack of adaptability to diverse speech patterns limits the system's accessibility for a global user base.

1. **Recognition Errors:**

The system occasionally misinterprets spoken inputs, leading to incorrect or unintended actions. These errors can stem from a variety of factors, including ambiguous speech, rapid speaking, or similar-sounding words. Such mistakes can disrupt the user experience, requiring additional time to correct or reissue commands. Consistent recognition errors reduce the overall reliability of the system.

1. **Timeout Issues:**

When the ADA system fails to recognize a command due to noise, mispronunciation, or other factors, it can lead to delays or system timeouts. This can be particularly frustrating for users, as they may have to repeat their input or restart the interaction. The system’s inability to promptly recognize and process commands hampers its responsiveness. Frequent timeouts negatively impact user satisfaction and the overall efficiency of the system.

**B. Solution to Errors**

1. **Ambient Noise Interference:**
   * **Noise Cancellation:** Use technology that reduces background noise so the system can focus better on your voice.
   * **Microphone Setup:** Use special microphones that pick up your voice from different directions, making it easier to hear you over any background noise.
   * **Noise Detection:** Add a feature that senses how noisy your environment is and adjusts the system to better understand your speech.
2. **Accents and Pronunciations:**
   * **Better Training:** Teach the system to recognize a wider range of accents and pronunciations, so it can understand more people accurately.
   * **Personalized Training:** Let users train the system to recognize their specific way of speaking, making it more accurate for each individual.
   * **Learning Over Time:** Add a feature where the system learns from past interactions to improve its understanding of each user’s speech.
3. **Recognition Errors:**
   * **Smarter Models:** Upgrade the system to better understand the meaning behind words, reducing mistakes in recognizing speech.
   * **Error Checks:** Include a feature where the system asks for confirmation if it’s unsure about what you said, helping to avoid mistakes.
   * **Context Awareness:** Improve the system's ability to understand the context of a conversation, so it can more accurately guess what you’re trying to say.
4. **Timeout Issues:**
   * **Flexible Timeouts:** Make the system’s response time more flexible, so it can handle different speaking speeds or complex commands without unnecessary delays.
   * **Clarification Requests:** Allow the system to ask you to repeat or clarify if it didn’t catch what you said, instead of just timing out.
   * **Alternative Options:** If the system is having trouble understanding, provide other ways for you to interact, like using touch or visual inputs, to keep things running smoothly.

**V. Graphic User Interface**

**Concept Art**

Our Team combined our ideas about the concept art of ADA's GUI, the following are the ideas presented by the members of the team: Primarily, Our Group agreed to place 3 panels for ADA's GUI, one panel for ADA's Greetings. Ex. (Hi, Hello), one panel to display the users’ questions, and one panel to display ADA's answer to the users. We also came up with 4 different variations of ADA’s UI: First is the UI with face of the 1st Robot Image at the Concept design and have a sound wave that is indication that ADA is talking. Second is the Boxy Robot that was derived from the 3rd image of the ADA’s Design Concept. For the 3rd Image, it’s showing the plain Concept of ADA shown at the 2nd image at ADA’s Design Concept and the 4th Image are just an Imaginary concept derived from the GUI Team with an integrated sound waves at the middle of the body.

A drawing of a website

Description automatically generated with medium confidenceA drawing of a website

Description automatically generated with medium confidence

A drawing of a website

Description automatically generated with medium confidenceA drawing of a website

Description automatically generated with medium confidence

ADA 2.1 User Interface Sketch

A screenshot of a computer screen

Description automatically generated

ADA 2.1 User Interface Concept

Our team made ADA’s Model Design based on these Concepts.

