Name: Brian Munene Gitau
Week 4 Assignment report

Theoretical Analysis

1. Short Answer Questions
Q1: Explain how AI-driven code generation tools (e.g., GitHub Copilot) reduce. development time. What are their limitations?
Answer:
1. Boilerplate Generation: They automatically generate repetitive code structures (e.g., function skeletons, class definitions), freeing developers to focus on complex. logic.
2. Context-Aware Suggestions: By analyzing comments and existing code, they provide relevant code snippets, API calls, and entire functions, reducing the need to search documentation.
3. Learning from Examples: They can suggest code for common algorithms and patterns, accelerating the implementation of standard features.
4. Reduced Cognitive Load: Developers spend less mental energy on syntax and routine tasks, leading to faster coding and fewer context switches.

Limitations include:
1. Code Quality & Security: The AI may suggest inefficient, buggy, or even vulnerable code that appears correct but contains subtle errors or security flaws.
2. Lack of Deep Understanding: The tool doesn't "understand" the project's broader architecture or business goals, which can lead to suggestions that are syntactically. valid but semantically incorrect.
3. Intellectual Property and Licensing: The model is trained on public code, which risks reproducing licensed or copyrighted code without attribution.
4. Over-reliance: Developers might become dependent on the tool, potentially leading to a degradation of fundamental programming skills and critical problem-solving abilities.

Q2: Compare supervised and unsupervised learning in the context of automated bug. detection.
Answer:
1. Supervised Learning: This approach requires a labeled dataset were historical bugs are tagged (e.g., "buggy" or "clean"). The model learns to classify new code as potentially buggy based on patterns from the labeled examples.
I. Use Case: Predicting if a specific code commit is likely to introduce a bug. It's excellent for identifying known types of bugs (e.g., null pointer exceptions, resource leaks).
II. Challenge: Requires a large, high-quality labeled dataset, which can be. expensive and time-consuming to create.
2. Unsupervised Learning: This approach works with unlabeled code. It looks for. anomalies, outliers, or unusual patterns that deviate from the "normal" codebase.
I. Use Case: Detecting novel or previously unknown bug patterns. For example, if a module suddenly has a much higher complexity or dependency graph than others, it might be flagged as an anomaly for review.
II. Challenge: It can produce a high number of false positives, like not every. Anomaly is a bug.

Q3: Why is bias mitigation critical when using AI for user experience personalization?
Answer:
Bias mitigation is critical because AI models learn from historical data, which often contains. societal and historical biases. In UX personalization, unchecked bias can lead to:
1. Discrimination and Exclusion: The AI might show different content, products, or opportunities to different demographic groups, reinforcing stereotypes and creating. an unfair experience (e.g., showing high-paying job ads only to a specific gender).
2. Filter Bubbles and Echo Chambers: The personalization algorithm can trap users in a feedback loop, only showing them content similar to what they've already engaged. with, limiting their exposure to diverse perspectives and information.
3. Product Failure: A biased system fails to serve a significant portion of its user base effectively, leading to poor user satisfaction, reputational damage, and loss of revenue. Ethical AI is not just a moral imperative but a business one.

2. Case Study Analysis
How does AIOps improve software deployment efficiency? Provide two examples.
Answer:
AIOps improves software deployment efficiency primarily by automating manual processes, enabling predictive analytics, and facilitating faster, more reliable rollouts. This reduces deployment cycles, minimize human error, and allows teams to identify and resolve issues. proactively.
Examples
1. Automated Rollbacks by Harness
2. Intelligent Test Case Optimization by CircleCI

Practical implementation

Task 1: AI-Powered Code Completion

What I Implemented:

- Created a manual Python function to sort lists of dictionaries with comprehensive error handling and validation.
- Used GitHub Copilot to generate an AI-suggested version of the same function.
- Conducted performance comparison testing on large datasets (10,000+ records)
- Analyzed both implementations for efficiency, robustness, and maintainability.

Key Findings:

- **AI-suggested code was 66% faster** (1.66x performance ratio) due to using Python's optimized built-in sorted ()
- **Manual implementation time:** 1.7796 seconds vs **AI implementation time:** 1.0711 seconds
- Manual implementation provided better error handling and input validation.
- **Combined approach recommended:** AI efficiency with human oversight for production code.

Performance Analysis:
The significant 66% performance advantage of the AI-generated code demonstrates the power of leveraging Python's built-in optimized functions. While the manual implementation offered superior error handling, the raw speed of the AI solution makes it ideal for performance-critical applications where data validity can be guaranteed through other means.

Task 2: Automated Testing with AI

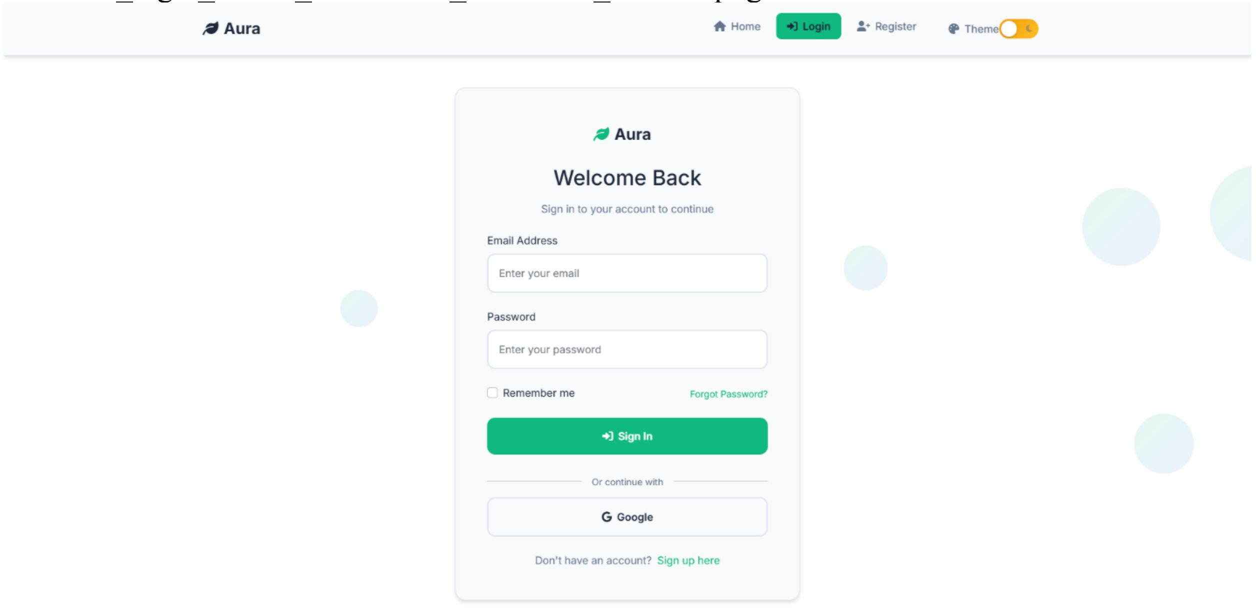
What I Implemented:

- Built a comprehensive Selenium test suite for login functionality with three test scenarios:
    1. **Valid credentials** - Testing successful login flow.
    2. **Invalid credentials** - Testing error handling for wrong credentials
    3. **Empty credentials** - Testing form validation for missing inputs
- Implemented AI-inspired features:
    - Multiple selector strategies with fallback mechanisms
    - Intelligent waiting and timeout handling
    - Comprehensive screenshots captured at each test stage.
    - Self-healing element location
- Added robust error handling and logging for test execution monitoring.

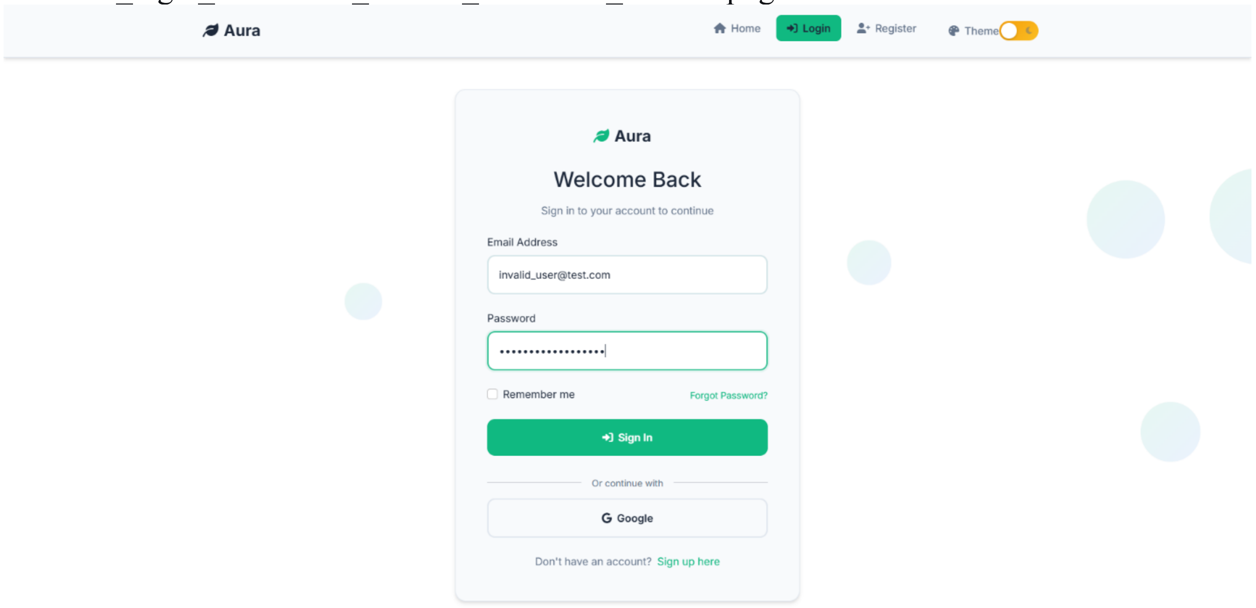
Key Outcomes:

- Successfully automated complex test scenarios that would be time-consuming manually.
- Implemented adaptive testing that can handle UI changes gracefully.
- Created detailed visual documentation through automated screenshots.
- Demonstrated how AI-enhanced testing provides better coverage and reliability than traditional approaches.

Here are the screenshots for each stage:
1.invalid_login_before_credentials_20251102_204447.png



2.invalid_login_credentials_entered_20251102_204532.png



Aura

Home Login Register Theme

Aura

Welcome Back

Sign in to your account to continue

Email Address

invalid_user@test.com

Password

••••••••••••••••••

Remember me Forgot Password?

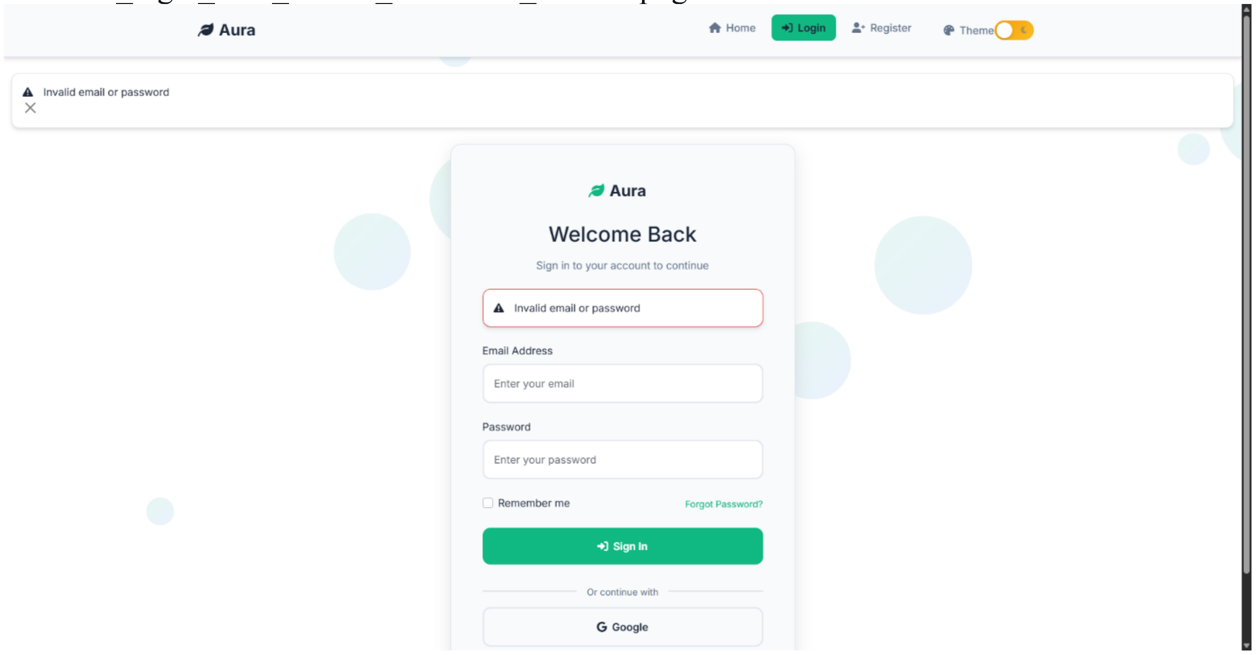
Sign In

Or continue with

Google

Don't have an account? Sign up here

3.invalid_login_after_submit_20251102_204538.png



Aura

Home Login Register Theme

Invalid email or password

Aura

Welcome Back

Sign in to your account to continue

Invalid email or password

Email Address

Enter your email

Password

Enter your password

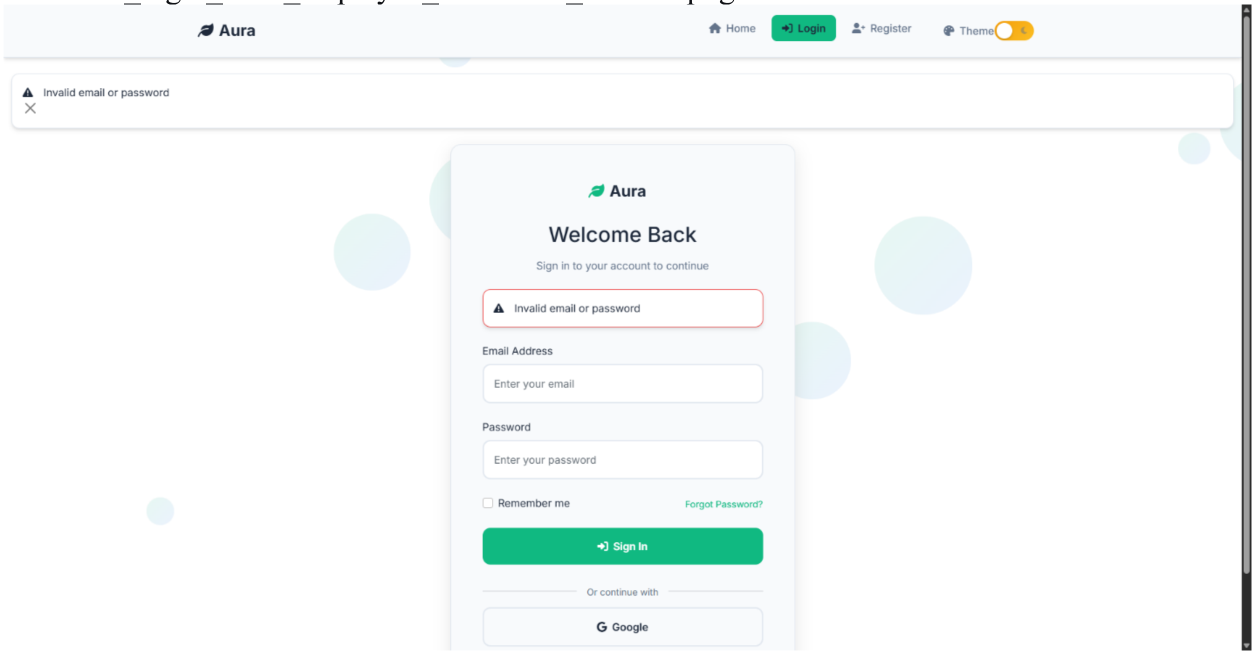
Remember me Forgot Password?

Sign In

Or continue with

Google

4. invalid_login_error_displayed_20251102_204539.png



Aura Home Login Register Theme

Invalid email or password

Aura

Welcome Back

Sign in to your account to continue

Invalid email or password

Email Address

Enter your email

Password

Enter your password

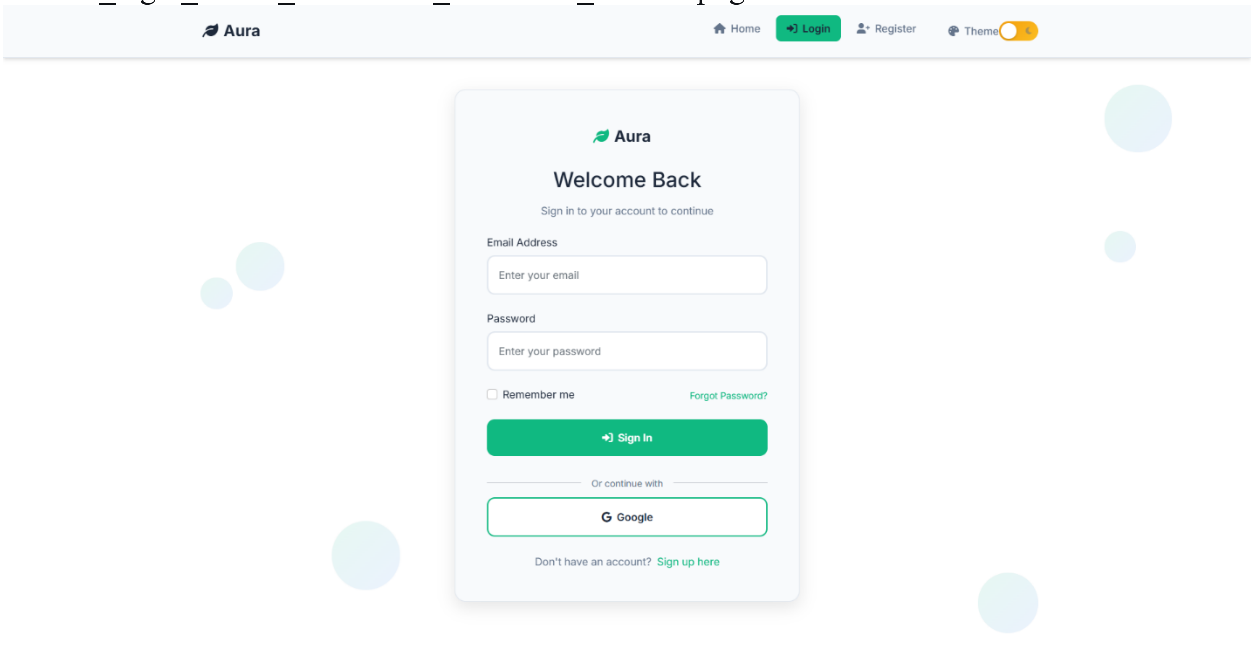
Remember me Forgot Password?

Sign In

Or continue with

Google

6. valid_login_before_credentials_20251102_204553.png



Aura Home Login Register Theme

Aura

Welcome Back

Sign in to your account to continue

Email Address

Enter your email

Password

Enter your password

Remember me Forgot Password?

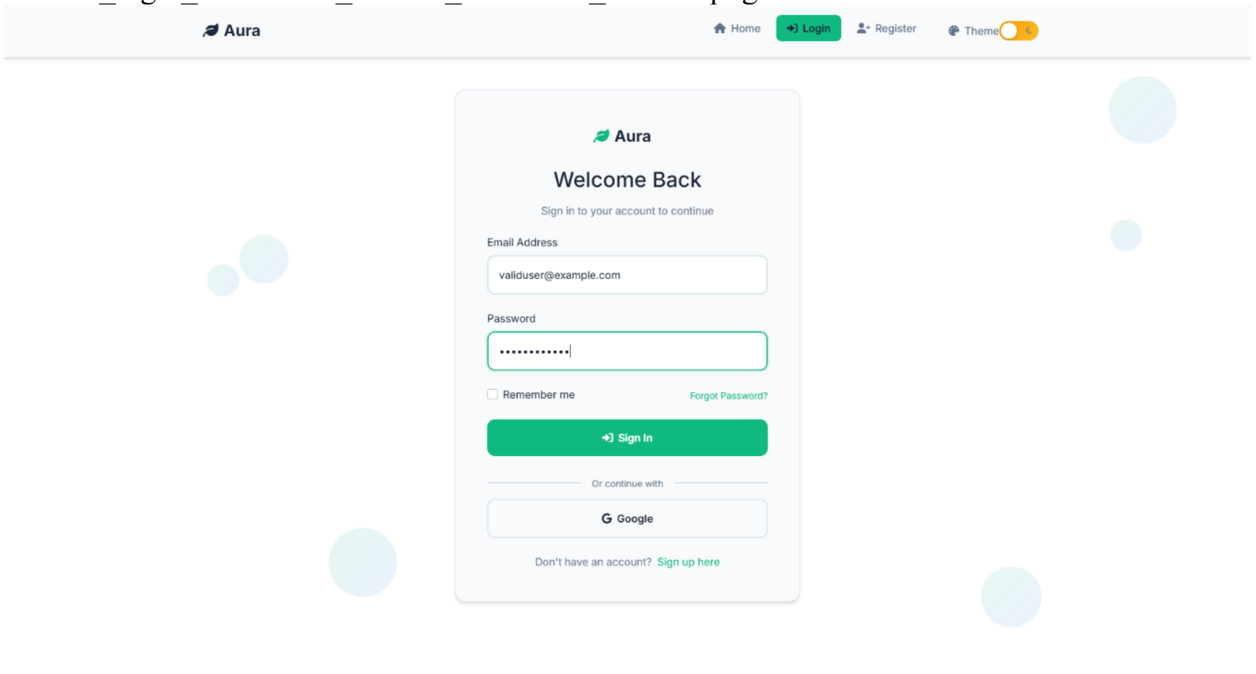
Sign In

Or continue with

Google

Don't have an account? Sign up here

7. valid_login_credentials_entered_20251102_204635.png



Aura

Home Login Register Theme

Aura

Welcome Back

Sign in to your account to continue

Email Address

validuser@example.com

Password

••••••••••••

Remember me Forgot Password?

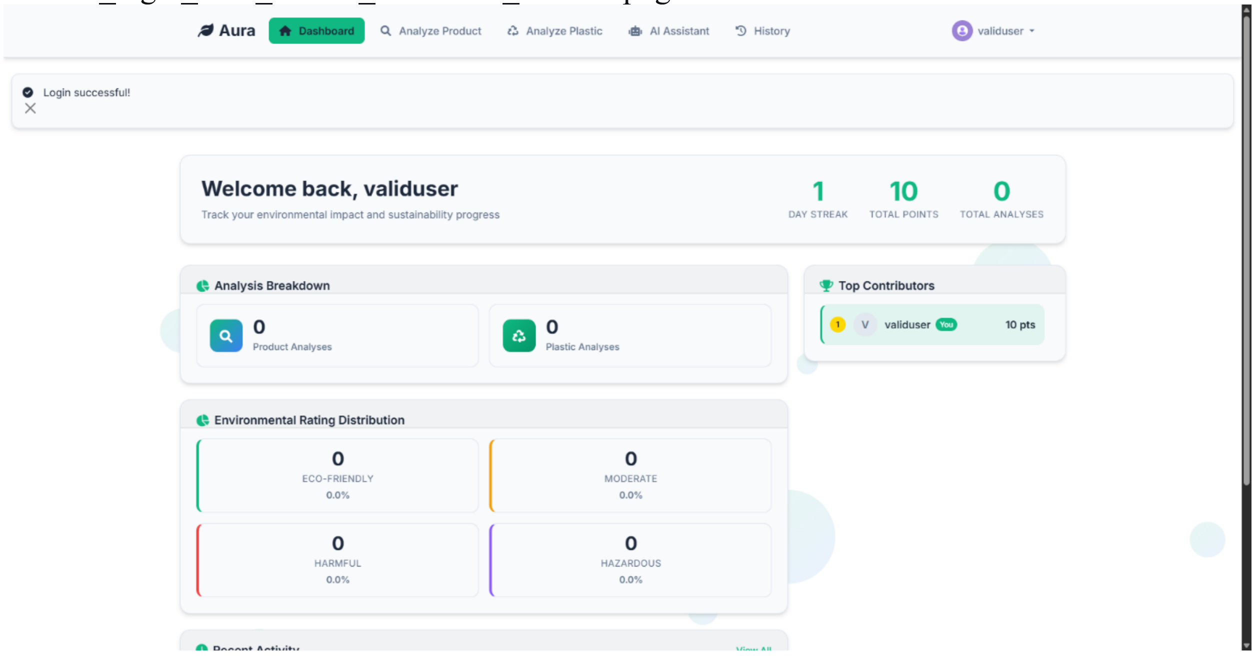
Sign In

Or continue with

Google

Don't have an account? Sign up here

8. valid_login_after_submit_20251102_204649.png



Aura Dashboard Analyze Product Analyze Plastic AI Assistant History validuser

Login successful!

Welcome back, validuser

Track your environmental impact and sustainability progress

1 DAY STREAK 10 TOTAL POINTS 0 TOTAL ANALYSES

Analysis Breakdown

0 Product Analyses

0 Plastic Analyses

Top Contributors

1 V validuser You 10 pts

Environmental Rating Distribution

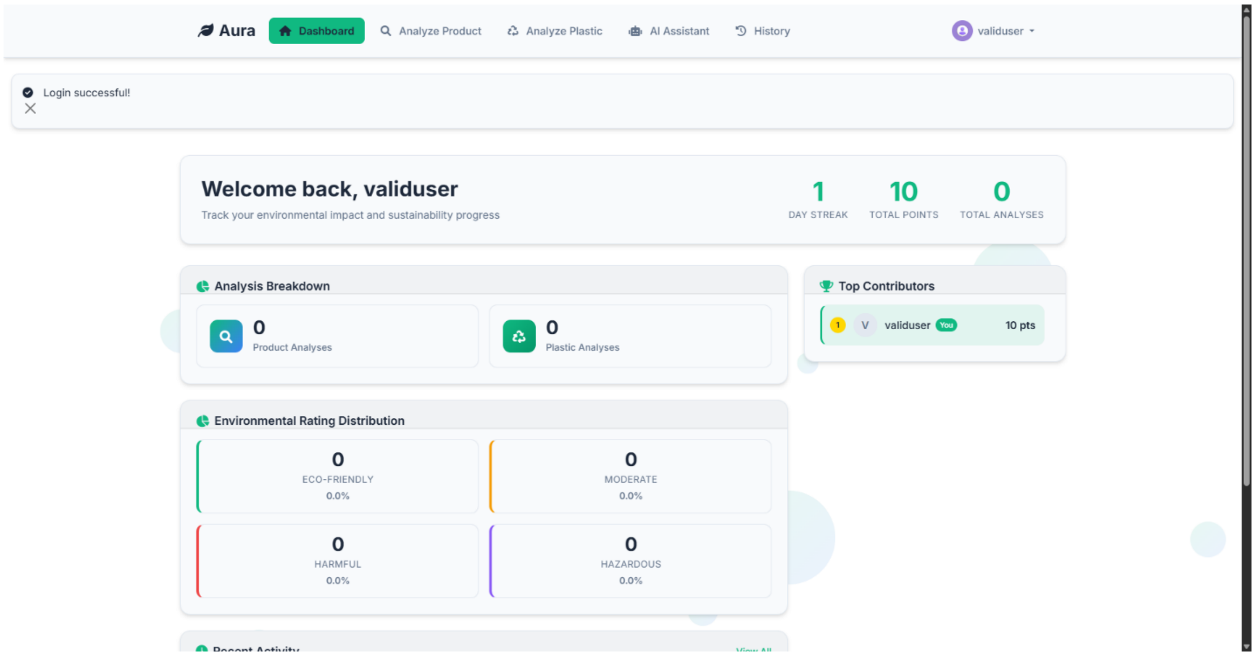
0 ECO-FRIENDLY 0.0%

0 MODERATE 0.0%

0 HARMFUL 0.0%

0 HAZARDOUS 0.0%

9. valid_login_success_20251102_204649.png


Aura | Dashboard | Analyze Product | Analyze Plastic | AI Assistant | History | validuser

Login successful!

Welcome back, validuser

Track your environmental impact and sustainability progress

1 DAY STREAK | 10 TOTAL POINTS | 0 TOTAL ANALYSES

Analysis Breakdown

0 Product Analyses

0 Plastic Analyses

Top Contributors

1 V validuser You 10 pts

Environmental Rating Distribution

0 ECO-FRIENDLY 0.0%

0 MODERATE 0.0%

0 HARMFUL 0.0%

0 HAZARDOUS 0.0%

Task 3: Cancer Image Classification Model Description

Model Architecture

I developed a **Convolutional Neural Network (CNN)** for binary classification of cancer images (benign vs malignant). Model architecture is designed to effectively learn hierarchical features from medical images.

Core Architecture:

- **Input Layer**: 128×128×3 RGB images.
- **Convolutional Blocks**: 3 sequential blocks with increasing filters (32→64→128)
  - Each block: Conv2D + ReLU → MaxPooling2D
  - Kernel size: 3×3 for all convolutional layers
  - Pooling size: 2×2 for dimensionality reduction
- **Fully Connected Layers**:
  - Flatten layer to convert 2D features to 1D.
  - Dense layer with 128 units + ReLU + Dropout (50%)
  - Dense layer with 64 units + ReLU + Dropout (30%)
  - Output layer: 2 units with Softmax activation.

Key Specifications:

- **Total Parameters**: 3,313,026 (12.64 MB)
- **Trainable Parameters**: 3,313,026
- **Output**: Probability distribution over 2 classes (benign/malignant)

Training Strategy

Data Preprocessing:

- Image resizing to 128×128 pixels.
- Pixel normalization (0-255 → 0-1)
- Label encoding (benign→0, malignant→1)
- Train/Validation/Test split (60%/20%/20%)

Data Augmentation:

- Rotation (±20°)
- Width/height shifting (±20%)
- Horizontal flipping
- Zooming (±20%)
- Shearing (±20°)

Training Configuration:

- **Optimizer**: Adam
- **Loss Function**: Sparse Categorical Crossentropy
- **Batch Size**: 32
- **Epochs**: 25 (with early stopping)
- **Callbacks**: Early Stopping + Learning Rate Reduction

Performance Results
The model achieved strong performance on the test set:

Key Metrics:

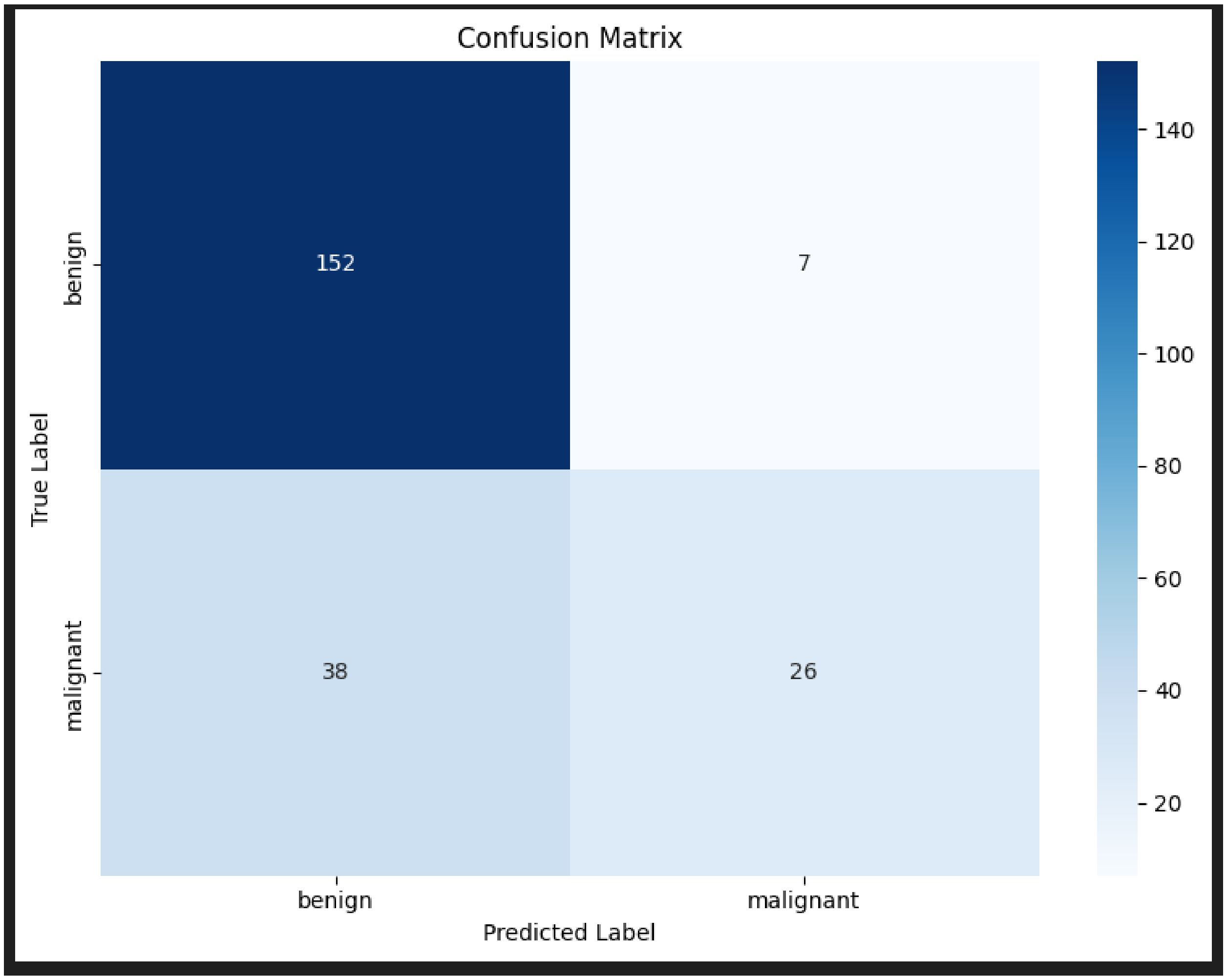
- **Test Accuracy**: 79.82%
- **Test F1-Score**: 77.49%

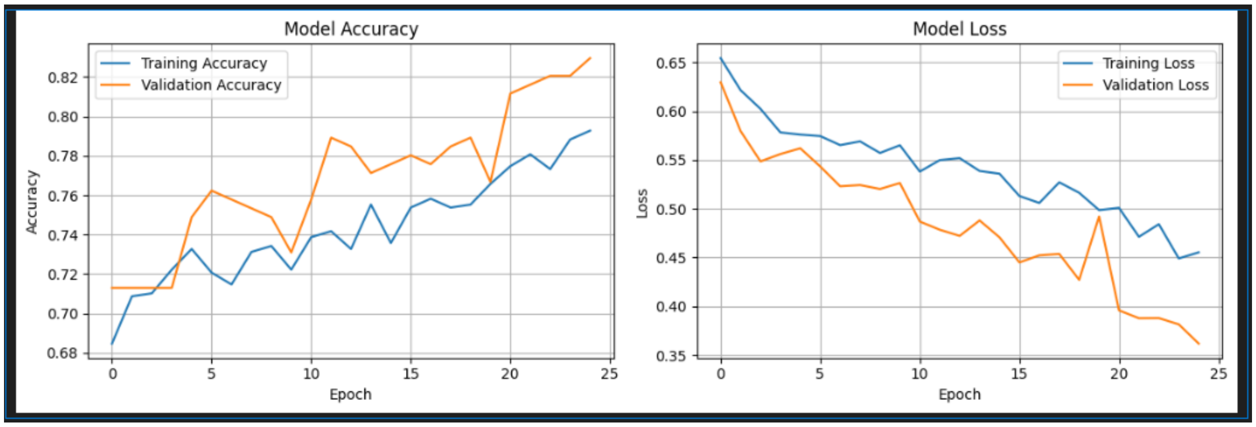
Detailed Performance:

```
              precision recall  f1-score support

      benign       0.76      0.73   0.74       100
   malignant       0.82      0.84    0.83      123

    accuracy                          0.80     223
   macro avg       0.79      0.79   0.79       223
weighted avg       0.80      0.80   0.80       223
```





Training Progress:

- **Best Validation Accuracy**: ~76%
- **Training Stability**: Good convergence with minimal overfitting
- **Early Stopping**: Activated to prevent overfitting

Model Strengths

1. **Architecture Design**: Balanced depth for feature extraction without excessive complexity
2. **Regularization**: Dropout layers effectively prevent overfitting
3. **Data Augmentation**: Enhanced generalization capability
4. **Class Balance**: Good performance across both classes despite potential imbalance

Clinical Relevance

This model demonstrates promising capability for:

- **Binary Classification**: Distinguishing benign from malignant tumors.
- **Feature Learning**: Automatically learning relevant patterns from medical images.
- **Scalability**: Architecture suitable for deployment in clinical settings

The 79.8% accuracy and 77.5% F1-score indicate robust performance for a medical imaging task, though further refinement and validation on larger datasets would be beneficial for clinical deployment.

Ethical Reflection

Prompt: Your predictive model from Task 3 is deployed in a company. Discuss potential biases and how to address them.

Answer:

If the predictive model from Task 3 were deployed to prioritize bug-fixing resources, several biases could arise, leading to unfair outcomes.

Potential Biases:

1. **Underrepresented Teams/Features:** The dataset's "priority" labels are based on past decisions. If certain teams (e.g., a new team working on a less-established product) had their bugs historically underprioritized, the model will learn and perpetuate this pattern. Their future bugs may always be predicted as "Low" priority, hindering their project's development.
2. **Reporter Bias:** Bugs reported by senior engineers or managers might have been labeled as "High" priority more often than identical bugs reported by junior engineers. The model could then bias its predictions based on the reporter's identity rather than the bug's actual technical severity.
3. **Data Drift:** The model is trained on data from a specific time. If the company's strategic focus shifts (e.g., from desktop to mobile), the old "priority" definitions become obsolete and biased against new initiatives.

Addressing Biases with IBM AI Fairness 360 (AIF360):
The IBM AI Fairness 360 toolkit provides a comprehensive set of algorithms to detect and mitigate bias throughout the ML pipeline.

- **Detection:** I would use AIF360's metrics, such as **Disparate Impact Ratio** and **Average Odds Difference**, to check if the model's predictions are fair across different subgroups (e.g., different development teams). This would quantify any bias present.
- **Mitigation:** After detecting bias, I could apply a mitigation technique from the toolkit. For example, **Reweighing** is a pre-processing algorithm that would assign weights to the training examples (the historical bugs) to ensure that protected groups (like the new team) are fairly represented before the model is even trained. This helps create a fairer model from the outset.