Revolutionizing Software Testing with AI and ML:

Driving Scalability and Accuracy in QA

PRESENTATATOR: Srinivasa Rao Bittla

02/06/2025

Conf42: Python 2025

Introduction to Software Testing

Software Testing: Ensures software quality and reliability.

Types of Testing:

- Manual Testing: Performed by humans step-by-step.
- Automation Testing: Scripts automate repetitive tasks.

Question: How much time does your team spend on manual testing vs. automation?

Challenges in Traditional Testing

- Manual Testing: Time-consuming, error-prone, hard to scale.
- Automation Testing: Limited to predefined test cases, high maintenance.
- Complex Applications: Require more comprehensive testing.

Question: Can current testing methods keep up with rapid software releases?

Introduction to AI and ML in Testing

- Artificial Intelligence (AI): Mimics human intelligence for problem-solving.
- Machine Learning (ML): Learns patterns from data to make predictions.
- AI/ML Testing Tools: Adapt to changing software, predict defects.

Question: How could AI/ML reduce testing bottlenecks in your projects?

Comparing Traditional vs. AI-Driven Testing

Feature	Traditional Testing	AI-Driven Testing
Test Case Creation	Manual/Scripted	AI-Generated
Script Maintenance	High Effort	Self-Healing Scripts
Defect Detection	Reactive	Predictive Analytics
Scalability	Limited	Highly Scalable
Accuracy	Human Errors Possible	High Precision

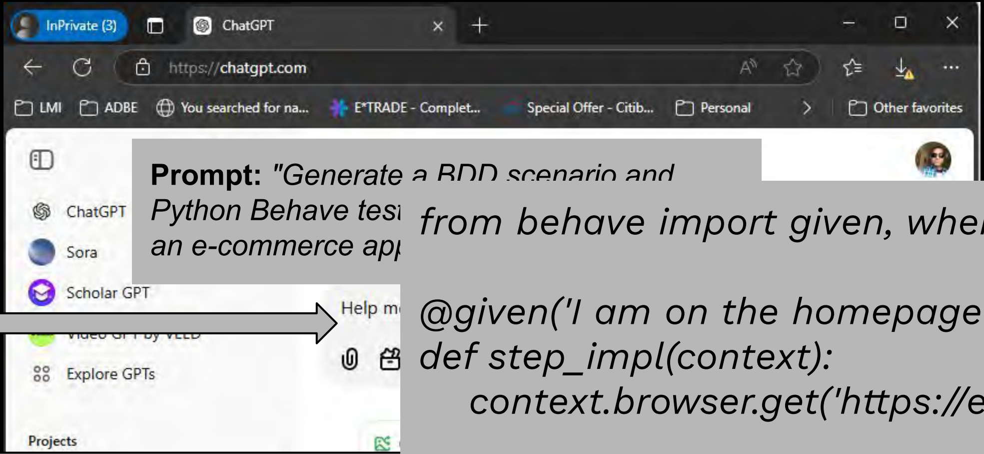
Question: Which feature do you think makes AI testing superior?

How AI and ML Enhance Testing

- Dynamic Test Case Generation: AI creates new test cases from user behavior.
- Self-Healing Scripts: ML updates test scripts automatically after code changes.
- Predictive Analytics: Forecasts high-risk areas in applications.

Question: What parts of your testing lifecycle could benefit from automation?

Demo - Dynamic Test Case Generation



Prompt: *"Generate a BDD scenario and Python Behave tesi an e-commerce app*

User Prompt

```
Feature: Product Search

  Scenario: Search for a product
    Given I am on the homepage
    When I search for "laptop"
    Then I should see results for "laptop
```

```python
from behave import given, when, then

@given('I am on the homepage')
def step_impl(context):
    context.browser.get('https://ecommerce-site.com')

@when('I search for "{product}"')
def step_impl(context, product):
    search_box = context.browser.find_element_by_name('q')
    search_box.send_keys(product)
    search_box.submit()

@then('I should see results for "{product}"')
def step_impl(context, product):
    assert product in context.browser.page_source
```

Case Study 1 – Self Healing

Problem: Frequent updates led to broken links and bugs.

Solution: AI detected UI issues and optimized workflows.

Result:

- 30% reduction in testing time.
- 45% increase in bug detection rate.

Question: How would faster bug detection impact your product delivery?

Case Study 1 – Self Healing

Automatic Learning: Healenium learns about locator changes over time and stores healed locators for future executions.

If the locator (`By.name("search")`) or `By.id("search-button")`) changes in the UI, Healenium will detect the failure and automatically search for similar elements in the DOM to replace the broken locator.

```
<dependency>
    <groupId>com.epam.healenium</groupId>
    <artifactId>healenium-web</artifactId>
    <version>3.3.2</version>
</dependency>
```

https://github.com/Srimaan/SelfHealing-Healenium

AI – Tools

Tool/Model	Test Case Generation	Code Generation	Supported Frameworks	AI-Powered
OpenAI Codex	Yes (via natural language)	Yes (step definitions)	Cucumber, Behave, SpecFlow	Yes
Testim.io	Yes (automated)	Yes	Selenium, Cypress	Yes
Functionize	Yes	Yes	Custom frameworks	Yes
SpecFlow	Manual	Yes	.NET (C#)	No (AI optional)
Test.ai	Yes	Yes	Custom for mobile apps	Yes
ChatGPT (OpenAI)	Yes	Yes	Behave, Cucumber	Yes
Mabl	Yes	Yes	Browser-based frameworks	Yes
Cucumber (AI-integrated)	Yes	Yes	Java, JavaScript, Python, Ruby	No (AI optional)

Case Study 2 – Predictive Analysis

Predictive Analysis for Defect Detection involves using historical defect data, testing metrics, and machine learning algorithms to predict potential defects in software before they occur.

Case Study 2 – Predictive Analysis Process

Data Collection:

Defect Reports
Code Metrics
Test Metrics
Release Data
Developer Activity

Data Preprocessing:

- Handle Missing Values
- Feature Engineering
- Encoding
- Normalization

Exploratory Data Analysis (EDA):

Correlation
High-risk areas
Visualization

Feature Selection:

Code Complexity Metrics
Change Metrics
Test Coverage.
Developer Activity.

Model Selection:

Classification Models
Clustering Models
Deep Learning Models

Model Training and Evaluation:

- Split data into **training** and **test** sets
- Train the selected model
- Evaluate using metrics:**Accuracy, Precision & Recall, F1-Score, ROC-AUC**

Prediction and Defect Risk Analysis

- Predict which features are prone to defects.
- Prioritize high-risk modules for intensive testing.
- Use predictions to allocate testing resources effectively.

Data Collection:

-Continuous Learning
Continuously update models with new defect data.
-Implement feedback loops for model improvement.

Case Study 2 – Predictive Analysis Tools

There are several tools and frameworks designed to implement **Predictive Analysis for Defect Detection** using historical defect data, testing metrics, and machine learning algorithms.

These tools help predict potential defects in software before they occur, leading to proactive quality assurance and faster delivery cycles.

Tool	Integration with CI/CD	Data Analysis Capability	ML/AI Support	Use Case Focus
Azure Machine				dictions
IBM SPSS Mode				nalysts
H2O.ai (AutoML				kflows
Google Cloud AI				models
SonarQube (ML				prediction
Jenkins (Predict				
Seerene				rediction
Kibana + Elasticsearch (Custom)	Yes	High	Customizable	Real-time defect monitoring
Test.ai	Yes	Low	Strong	UI/UX defect prediction
Jira Predictive Plugins	Yes	Moderate	Medium	Project-level defect trends

Case Study 2 – ML in Mobile App Testing

Problem: Device fragmentation caused inconsistent user experiences.

Solution: ML identified device-specific issues and optimized testing.

Result:

- 50% reduction in app crashes.
- 60% increase in device compatibility.

Question: Do you face challenges testing across multiple devices?

Metrics That Matter

Speed: AI reduces test execution time by up to 70%.

Coverage: ML expands test coverage by 60%.

Accuracy: AI improves defect prediction accuracy by 40%.

Question: Which metric is most critical for your team's success?

Tools and Technologies in AI/ML Testing

Popular Tools:

- Selenium with AI plugins (Healium)
- Test.ai
- Applitools (Visual AI)
- Mabl (Intelligent Testing)

Question: Which AI tool would you like to explore further?

Technologies:

- Natural Language Processing (NLP)
- Predictive Analytics
- Computer Vision

Comparing AI Models and Tools for BDD Test Generation

Tool/Model	Test Case Generation	Code Generation	Supported Frameworks	AI-Powered
OpenAI Codex	Yes (via natural language)	Yes (step definitions)	Cucumber, Behave, SpecFlow	Yes
Testim.io	Yes (automated)	Yes	Selenium, Cypress	Yes
Functionize	Yes	Yes	Custom frameworks	Yes
SpecFlow	Manual	Yes	.NET (C#)	No (AI optional)
Test.ai	Yes	Yes	Custom for mobile apps	Yes
ChatGPT (OpenAI)	Yes	Yes	Behave, Cucumber	Yes
Mabl	Yes	Yes	Browser-based frameworks	Yes
Cucumber (AI-integrated)	Yes	Yes	Java, JavaScript, Python, Ruby	No (AI optional)

Implementing AI/ML in QA

Step 1: Identify repetitive testing tasks.

Step 2: Choose suitable AI/ML tools.

Step 3: Start with pilot projects.

Step 4: Scale AI integration across workflows.

Question: What’s the first step your team can take toward AI testing?

Challenges in Adopting AI/ML

Initial Costs: Tool acquisition and training.

Data Dependency: Requires quality data for learning.

Change Management: Resistance to adopting new methods.

Question: What challenges could your team face when adopting AI in testing?

Future of QA with AI/ML

Autonomous Testing: AI will handle testing end-to-end.

Real-Time Defect Prediction: Faster issue resolution.

Continuous Learning: AI adapts to new technologies.

Question: How do you envision the future of software testing?

Key Takeaways

AI/ML: Boosts testing speed, accuracy, and scalability.

Real Results: Proven case studies show measurable improvements.

Adoption: Start small, scale with confidence.

Question: What key insight will you apply to your QA process?

Q&A

Thank you! Let’s discuss your thoughts and questions.

Contact Information:

Srinivasa Rao Bittl

sbittla@gmail.com

https://www.linkedin.com/in/bittla/

https://www.bittla.me/

Question: What would you like to explore more in AI-driven QA?