Include Screenshots of source code and Working final project.

```
import random

from datetime import datetime

class RCAProjectPhase:

    def __init__(self, phase_title, objective, overview):

        self.phase_title = phase_title

        self.objective = objective

        self.overview = overview
```

```
        self.key_enhancements = []
        self.challenges = []
        self.outcomes = []
        self.metrics = []
        self.next_steps = ""
        self.performance_metrics = {}

    def add_enhancement(self, enhancement):
        self.key_enhancements.append(enhancement)

    def add_challenge(self, challenge, solution):
        self.challenges.append((challenge, solution))

    def add_outcome(self, outcome, description):
        self.outcomes.append((outcome, description))

    def add_metric(self, metric_name, metric_value):
        self.performance_metrics[metric_name] = metric_value

    def set_next_steps(self, next_steps):
        self.next_steps = next_steps

    def display_summary(self):
        print(f"Phase Title: {self.phase_title}")
        print(f"Objective: {self.objective}")
        print(f"Overview: {self.overview}\n")
```

```
        print("Key Enhancements for RCA:")
        for enhancement in self.key_enhancements:
            print(f"- {enhancement}")
        print()

    def display_challenges_and_solutions(self):
        print("Key Challenges and Solutions:")
        for challenge, solution in self.challenges:
            print(f"Challenge: {challenge} \nSolution: {solution}\n")
        print()

    def display_outcomes(self):
        print("Expected Outcomes:")
        for outcome, description in self.outcomes:
            print(f"{outcome}: {description}")
        print()

    def display_metrics(self):
        print("Performance Metrics:")
        for metric, value in self.performance_metrics.items():
            print(f"- {metric}: {value}")
        print()

    def display_next_steps(self):
        print(f"Next Steps: {self.next_steps}\n")


class EquipmentFailureAnalysis:
```

```
    def __init__(self, failure_id, equipment_type, failure_date, failure_reason):
        self.failure_id = failure_id
        self.equipment_type = equipment_type
        self.failure_date = failure_date
        self.failure_reason = failure_reason
        self.analytical_tools_used = []
        self.root_cause = None

    def analyze_failure(self):
        print(f"Analyzing Failure for Equipment: {self.equipment_type} ({self.failure_id})")
        print(f"Failure Date: {self.failure_date}")
        print(f"Failure Reason: {self.failure_reason}\n")

        self.analytical_tools_used = random.sample(["5 Whys", "Fishbone Diagram", "Fault Tree Analysis", "Pareto Chart"], 2)
        self.root_cause = self.identify_root_cause()

    def identify_root_cause(self):
        root_causes = ["Operator error", "Component wear", "Improper maintenance", "External factors"]
        return random.choice(root_causes)

    def display_failure_analysis(self):
        print(f"Failure ID: {self.failure_id}")
        print(f"Equipment Type: {self.equipment_type}")
        print(f"Failure Date: {self.failure_date}")
        print(f"Root Cause Identified: {self.root_cause}")
        print(f"Analytical Tools Used: {, '.join(self.analytical_tools_used)}\n")
```

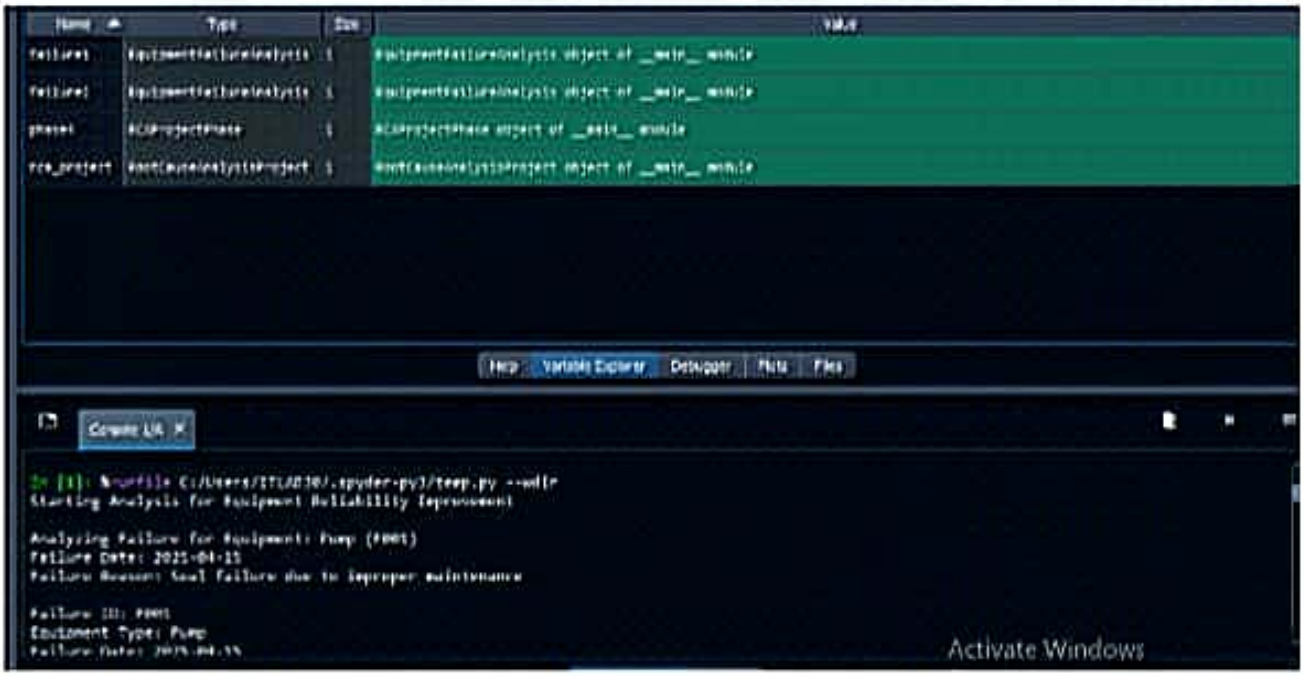
```
class ProjectMetrics:

    def __init__(self):

        self.total_failures = 0

        self.corrective_actions_taken = 0

        self.downtime_reduction = 0

        self.cost_savings = 0.0

        self.successful_rca_percentage = 0.0


    def update_metrics(self, total_failures, corrective_actions_taken,
downtime_reduction, cost_savings):

        self.total_failures = total_failures

        self.corrective_actions_taken = corrective_actions_taken

        self.downtime_reduction = downtime_reduction

        self.cost_savings = cost_savings

        self.successful_rca_percentage = (corrective_actions_taken / total_failures)
* 100 if total_failures > 0 else 0


    def display_metrics(self):

        print(f"Total Failures Analyzed: {self.total_failures}")

        print(f"Corrective Actions Taken: {self.corrective_actions_taken}")

        print(f"Downtime Reduction: {self.downtime_reduction} hours")

        print(f"Cost Savings: ${self.cost_savings:,.2f}")

        print(f"Successful RCA Percentage:
{self.successful_rca_percentage:.2f}%\n")



class RootCauseAnalysisProject:
```

```
    def __init__(self, project_name):
        self.project_name = project_name
        self.phases = []
        self.equipment_failures = []
        self.project_metrics = ProjectMetrics()

    def add_phase(self, phase):
        self.phases.append(phase)

    def add_equipment_failure(self, failure):
        self.equipment_failures.append(failure)

    def conduct_analysis(self):
        print(f"Starting Analysis for {self.project_name}\n")
        for failure in self.equipment_failures:
            failure.analyze_failure()
            failure.display_failure_analysis()

    def update_project_metrics(self, total_failures, corrective_actions_taken,
downtime_reduction, cost_savings):
        self.project_metrics.update_metrics(total_failures, corrective_actions_taken,
downtime_reduction, cost_savings)

    def display_project_summary(self):
        print(f"Project Name: {self.project_name}")
        for phase in self.phases:
            phase.display_summary()
            phase.display_key_enhancements()
            phase.display_challenges_and_solutions()
```

```
            phase.display_outcomes()
            phase.display_metrics()
            phase.display_next_steps()

        self.project_metrics.display_metrics()

rca_project = RootCauseAnalysisProject("Equipment Reliability Improvement")
phase4 = RCAProjectPhase(
    "Phase 4: Performance of the Project",
    "Identify the underlying causes of equipment failures to improve reliability, enhance operational efficiency, and reduce costs.",
    "This initiative focuses on improving RCA methods to enhance equipment reliability and reduce operational downtime."
)
phase4.add_enhancement("Failure Data Analysis")
phase4.add_enhancement("Streamlined RCA process for faster identification")
phase4.add_enhancement("Integration with IoT sensors and CMMS")
phase4.add_enhancement("Improved diagnostic accuracy with advanced tools like FTA, FMEA")
phase4.add_challenge("Inadequate Data Collection", "Use IoT sensors and CMMS to gather real-time and historical data.")
phase4.add_challenge("Lack of Proper Training", "Provide training on various RCA methodologies.")
phase4.add_challenge("Human Error and Bias", "Encourage open communication and involve cross-functional teams.")
phase4.add_challenge("Overloading with RCA Investigations", "Prioritize investigations based on downtime impact and safety concerns.")
phase4.add_outcome("Enhanced Equipment Reliability", "Reduce unexpected breakdowns and ensure consistent performance.")
phase4.add_outcome("Cost Savings", "Reduce repair and replacement costs.")
phase4.add_outcome("Reduced Downtime", "Increase operational uptime and efficiency.")
```

```
phase4.add_outcome("Enhanced Decision-Making", "Utilize RCA insights for better resource allocation and future planning.")

phase4.set_next_steps("Address identified issues with corrective actions and implement improvements for future analysis.")

rca_project.add_phase(phase4)

failure1 = EquipmentFailureAnalysis("F001", "Pump", "2025-04-15", "Seal failure due to improper maintenance")

failure2 = EquipmentFailureAnalysis("F002", "Compressor", "2025-04-20", "Component wear due to aging parts")

rca_project.add_equipment_failure(failure1)

rca_project.add_equipment_failure(failure2)

rca_project.conduct_analysis()

rca_project.update_project_metrics(total_failures=2, corrective_actions_taken=2, downtime_reduction=48, cost_savings=1200.50)

rca_project.display_project_summary()
```

output:


Name
Type
Size
Value
failure1
failure2
phase1
rca_project
Help
Variable Explorer
Debugger
Plots
Files
Console 1/A
Starting Analysis for Equipment Reliability Improvement
Analyzing Failure for Equipment: Pump (F001)
Failure Date: 2025-04-15
Failure Reason: Seal failure due to improper maintenance
Failure ID: F001
Equipment Type: Pump
Activate Windows

```
In [1]: %runfile C:/Users/ITLAB30/.spyder-py3/temp.py --wdir
Starting Analysis for Equipment Reliability Improvement

Analyzing Failure for Equipment: Pump (F001)
Failure Date: 2025-04-15
Failure Reason: Seal failure due to improper maintenance

Failure ID: F001
Equipment Type: Pump
Failure Date: 2025-04-15
```



```
Analyzing Failure for Equipment: Pump (F001)
Failure Date: 2025-04-15
Failure Reason: Seal failure due to improper maintenance

Failure ID: F001
Equipment Type: Pump
Failure Date: 2025-04-15
Root Cause Identified: Operator error
Analytical Tools Used: 5 Whys, Fault Tree Analysis

Analyzing Failure for Equipment: Compressor (F002)
```

```
Equipment Type: Compressor
Failure Date: 2025-04-20
Root Cause Identified: External factors
Analytical Tools Used: Pareto Chart, Fishbone Diagram

Project Name: Equipment Reliability Improvement
Phase Title: Phase 4: Performance of the Project
Objective: Identify the underlying causes of equipment failures to improve reliability, enhance operational efficiency, and reduce costs.
Overview: This initiative focuses on improving RCA methods to enhance equipment reliability and reduce operational downtime.
```

```
Expected Outcomes:
Enhanced Equipment Reliability: Reduce unexpected breakdowns and ensure consistent performance.
Cost Savings: Reduce repair and replacement costs.
Reduced Downtime: Increase operational uptime and efficiency.
Enhanced Decision-Making: Utilize RCA insights for better resource allocation and future planning.

Performance Metrics:
```

```
Performance Metrics:

Next Steps: Address identified issues with corrective actions and implement improvements for future analysis.

Total Failures Analyzed: 2
Corrective Actions Taken: 2
Downtime Reduction: 48 hours
Cost Savings: $1,200.50
Successful RCA Percentage: 100.00%
```