Here is a structured breakdown for your **AI-powered Root Cause Analysis (RCA) project** in the context of software testing / QA automation (based on the 5-phase implementation process you shared earlier, focused on reducing investigation time for test failures in CI/CD pipelines).

This is tailored to a typical enterprise setup where an **AI RCA platform** (like Virtuoso QA or similar) is integrated into existing test automation workflows.

Step 1: List of Main Actors (External Entities Interacting with the System)

Actor	Description	Type
QA Engineer / Tester	Primary user who reviews AI diagnoses, validates results, corrects if needed, and maintains test suite health.	Human user
Developer / Software Engineer	Receives enriched bug reports from AI RCA, fixes defects faster due to clear root cause + evidence.	Human user
DevOps Engineer / SRE	Configures integrations (CI/CD, monitoring), monitors platform health, handles self-healing rules.	Human user
QA / Test Manager	Views dashboards, trend analytics, release risk scores, approves RCA process changes.	Human user
CI/CD Pipeline (Jenkins/GitHub Actions/etc.)	Automatically triggers RCA on test failures, feeds execution artifacts to the platform.	External system
Test Automation Framework (e.g., Selenium, Playwright, Cypress)	Executes tests and captures raw data (logs, screenshots, DOM, network) for AI analysis.	External system
Issue Tracking System (Jira, Azure DevOps, GitHub Issues)	Receives auto-generated/enriched bug tickets from RCA platform.	External system
AI RCA Platform Admin	Configures data retention, privacy rules, model fine-tuning, user roles (rarely used).	Human user (advanced)

Core actors (most frequent interactions): **QA Engineer**, **Developer**, **CI/CD Pipeline**.

Step 2: List of Main Use Cases

These are the key functionalities the **AI RCA system** provides.

Use Case ID	Use Case Name	Brief Description	Priority
UC-01	Trigger RCA on Test Failure	System automatically starts analysis when a test fails in CI/CD.	High
UC-02	Capture & Collect Diagnostic Data	Automatically gather screenshots, DOM snapshots, console logs, network traffic, performance metrics.	High
UC-03	Perform AI-powered Root Cause Analysis	AI analyzes artifacts → classifies failure (bug / flake / env / data / timing) → identifies root cause.	High
UC-04	View AI Diagnosis & Evidence	User reviews AI explanation, visual evidence, suggested fix.	High
UC-05	Validate / Correct AI Diagnosis	QA engineer or developer marks diagnosis as correct / wrong / partial, provides feedback.	Medium
UC-06	Auto-Generate Enriched Bug Report	System creates Jira/Azure DevOps ticket with root cause, repro steps, evidence, priority suggestion.	High
UC-07	View Failure Trends & Analytics	Dashboards showing recurring patterns, flaky tests, QA capacity saved, defect escape rate.	Medium
UC-08	Configure RCA Rules & Integrations	Set up notification rules, self-healing patterns, data retention, CI/CD hooks.	Medium
UC-09	Train / Fine-tune AI Model	Provide feedback data to improve accuracy over time (supervised learning loop).	Low
UC-10	Monitor Release Risk	Use historical RCA data to score release readiness / predict high-risk areas.	Medium
UC-11	Handle Pilot / Parallel Investigation	Run manual RCA + AI side-by-side during pilot phase to build trust.	Low

Step 3: Relationships Between Actors and Use Cases

(Using standard UML notation types)

- **Association** (solid line): Actor directly interacts with / initiates the use case.
- **Include** (dashed arrow «include»): One use case always requires / includes another.

- **Extend** (dashed arrow «extend»): One use case optionally extends another under certain conditions.

Use Case	Actor(s) Involved	Relationships
UC-01 Trigger RCA	CI/CD Pipeline (primary), Test Automation Framework	Association (triggered automatically)
UC-02 Capture Data	Test Automation Framework, CI/CD Pipeline	«include» → UC-01 (always part of failure processing)
UC-03 AI Analysis	AI RCA Platform (system), CI/CD Pipeline	«include» → UC-02 (requires captured data)
UC-04 View Diagnosis	QA Engineer, Developer, Test Manager	Association
UC-05 Validate / Correct	QA Engineer, Developer	«extend» → UC-04 (optional when confidence low or wrong)
UC-06 Generate Bug Report	AI RCA Platform, Issue Tracking System	«include» → UC-03 (after analysis completes), Association to Developer/QA
UC-07 View Analytics	QA Manager, Test Manager, DevOps Engineer	Association
UC-08 Configure Rules	DevOps Engineer, Platform Admin	Association
UC-09 Fine-tune Model	QA Engineer (provides feedback), Platform Admin	«extend» → UC-05 (feedback loop)
UC-10 Monitor Risk	QA Manager, Test Manager	«include» → UC-07 (uses trend data)
UC-11 Pilot Mode	QA Engineer	«extend» → UC-01 / UC-04 (optional during rollout phase)

Step 4: Suggested Use Case Diagram Structure (Textual Description – Ready for Drawing)

You can draw **one main diagram** covering the core flow (recommended for assignment clarity), or split into 2–3 focused diagrams (e.g., Core Analysis Flow, Analytics & Management, Configuration).

Recommended Main Use Case Diagram (Core Operational Flow)

- **System Boundary**: Rectangle labeled «AI-Powered Root Cause Analysis Platform»

Inside boundary (use cases as ovals):

- Trigger RCA on Test Failure
- Capture Diagnostic Data («include» from Trigger)
- Perform AI Root Cause Analysis («include» from Capture)
- View AI Diagnosis & Evidence
- Validate / Correct Diagnosis («extend» from View Diagnosis)
- Auto-Generate Enriched Bug Report («include» from Perform Analysis)

Outside boundary (actors as stick figures, connected with solid lines):

- CI/CD Pipeline → Trigger RCA
- Test Automation Framework → Capture Data
- QA Engineer → View Diagnosis + Validate Diagnosis
- Developer → View Diagnosis + Validate Diagnosis + Receive Bug Report
- Issue Tracking System → receives Auto-Generate Bug Report

Optional Secondary Diagram: Analytics & Oversight

- Actors: QA Manager, DevOps Engineer
- Use cases: View Failure Trends & Analytics, Monitor Release Risk, Configure RCA Rules

How to Draw It (Tools)

- Use draw.io / diagrams.net (free, export as PNG/PDF)