PROBLEM STATEMENT : Develop a software system for a financial institution to detect and prevent fraudulent transactions. The project involves high-risk and complex requirements, such as real-time transaction monitoring and machine learning algorithms for fraud detection. The system needs to undergo multiple iterations of risk analysis and prototyping.

Task – 01 : Select the suited software development model defining the processes that will be used and the activities involved in the process, Prepare and document the blueprint using standard documentation template and sample templates.

The SPIRAL MODEL is best suited for this project as it requires a high-risk, complex system involving real-time monitoring, machine learning, and multiple iterations of risk analysis and prototyping.

| **Quadrant** | **Process** | **Key Activities** |
| --- | --- | --- |
| Objective Setting | Define objectives for the current phase | - Identify stakeholder needs - Set goals - Define quality targets |
| Risk Analysis & Mitigation | Identify and reduce risks | - Analyze technical, financial, ML, and security risks - Plan mitigation strategies - Evaluate prototypes |
| Development & Validation | Build and test deliverables | - Prototype development - Software coding - ML model training - Unit/Integration testing |
| Planning for Next Iteration | Plan the next phase | - Review the current cycle - Gather feedback - Adjust project plan |

Task – 02 : Identify the milestones and deliverables in each and at end of development phases.

| **Phase** | **Milestone(s)** | **Key Deliverables** |
| --- | --- | --- |
| Phase 1: Planning | Requirements Approved | - Software Requirements Specification (SRS)  - Feasibility Study Report  - Project Plan with Timeline and Budget  - Initial Use Case Diagrams  - Stakeholder Sign-off Document |
| Phase 2: Risk Analysis | Risk Assessment Completed | - Risk Management Plan  - Risk Register  - Security and Compliance Checklist  - Machine Learning Feasibility Report  - Prototype Strategy Document |
| Phase 3: Engineering | Architecture Finalized  Modules Implemented  ML Model Integrated | - Software Design Document (SDD)  - Architecture Diagrams (UML, DFDs)  - Machine Learning Code and Training Summary  - API Documentation and Data Pipeline Scripts  - UI Wireframes or Prototypes  - Source Code Repository Access |
| Phase 4: Evaluation | Prototype Validated  System Passed Tests  Stakeholder Approval | - Test Plan and Test Cases  - Test Execution Reports (Unit, Integration, Security)  - Performance Report (Latency, Accuracy)  - User Acceptance Testing Report  - Prototype Demonstration Summary |
| Final Phase: Deployment | System Delivered and Deployed | - Deployment Plan and Go-Live Checklist  - User Manuals and Training Guides  - Operations and Maintenance Manual  - Final System Handover Document  - Post-Implementation Review Report |

Task – 2.1 : Identify and design the verification & validation points and relevant methods along the development life cycle, with outputs and reports. Refer to V & V Standards 1012-1998 for selection and preparing the V & V checkpoints and methods.

| **Lifecycle Phase** | **V&V Checkpoint** | **Method** | **Output / Report** |
| --- | --- | --- | --- |
| Planning | Requirements Review | Requirement Inspection, Audit | SRS Review Report |
| Risk Analysis | Risk Management Plan Verification | Technical Review | Risk Audit Report |
| Design | Design Verification | Design Walkthroughs, Peer Review | Design Review Report |
| Implementation | Code Verification | Static Analysis, Code Review | Code Review Logs |
| ML Model Training | ML Model Validation | Cross-validation, Accuracy Metrics | ML Validation Report |
| Integration | Integration Testing | Interface Testing | Integration Test Report |
| Evaluation | System Validation & User Acceptance | Alpha/Beta Testing, Demo Review | UAT Report |
| Maintenance Prep | Documentation Verification | Checklist Audit | Document Validation Report |

## Quality Control Using ISO 9126

ISO 9126 Quality Characteristics:

1. Functionality
2. Reliability
3. Usability
4. Efficiency
5. Maintainability
6. Portability

| Quality Factor | How to Control It | Measurable Criteria | Tool/Method |
| --- | --- | --- | --- |
| Functionality | Requirement Traceability | Coverage in SRS to Implementation | Trace Matrix |
| Reliability | Load/Stress Testing | Downtime %, MTBF | JMeter |
| Usability | UI Testing, Feedback | Task success rate | User Testing |
| Efficiency | Performance Testing | Response time, throughput | Monitoring Tools |
| Maintainability | Code Modularity | Cyclomatic Complexity | Static Analysis |
| Portability | Multi-platform Support | Build Success Rate | CI/CD Logs |

Task – 03 : Define your quality target and justify the quality model solution for the problem.

| **Quality Characteristic** | **Target Metric** | **Justification** |
| --- | --- | --- |
| Functionality | 100% coverage of critical fraud scenarios in ML model and logic | Detects all high-priority fraud types |
| Reliability | ≥ 99.95% system uptime; false positive rate < 5% | Ensures trust and system availability |
| Usability | User error rate < 3%; response time < 2s | Operators can act quickly and confidently |
| Efficiency | Throughput: 1000 transactions/sec | Handles real-time transaction volumes |
| Maintainability | Modular design with ≥ 85% unit test coverage | Reduces time/cost for future updates |
| Portability | Full functionality on cloud + local server environments | Enables flexible deployment options |

**JUSTIFICATION OF QUALITY MODEL :**

* comprehensive and modular , covering technical as well as user-facing aspects
* secure and reliable
* continuous quality tracking and early defect detection
* aligns with projet goals

Task – 04 : Demonstrate how you can assure and achieve the defined targets.

| **Quality Attribute** | **Defined Target** | **Assurance Methods** | **Tools / Techniques** | **Responsible Roles** |
| --- | --- | --- | --- | --- |
| Functionality | 100% critical fraud scenario coverage | Requirement Traceability, Functional Testing | Test Coverage Matrix, JIRA, Selenium | QA Analyst, Dev Team |
| Reliability | ≥ 99.95% uptime False positive rate < 5% | Load Testing, Model Evaluation, Failover Testing | JMeter, Grafana, TensorBoard | DevOps, Data Scientist |
| Usability | Error rate < 3% Response time < 2s | User Testing, Heuristic Evaluation | Figma, Hotjar, User Surveys | UX Designer, QA Analyst |
| Efficiency | 1000 tx/sec processing | Performance Benchmarking | Apache JMeter, Prometheus, New Relic | Backend Dev, DevOps |
| Maintainability | 85%+ unit test coverage, modular code | Code Review, Static Analysis, CI/CD | SonarQube, GitHub Actions | Developers, Tech Lead |
| Portability | Run on cloud + on-prem | Environment Testing, Docker | Docker, Kubernetes, Jenkins | DevOps Team |

We will also integrate automated validation pipelines to continuously check model accuracy . Regular code reviews, static code analysis, and regression testing will help maintain software reliability and reduce bugs early. Further, a feedback loop from stakeholders and end-users will be used to refine system usability and ensure alignment with real-world needs.