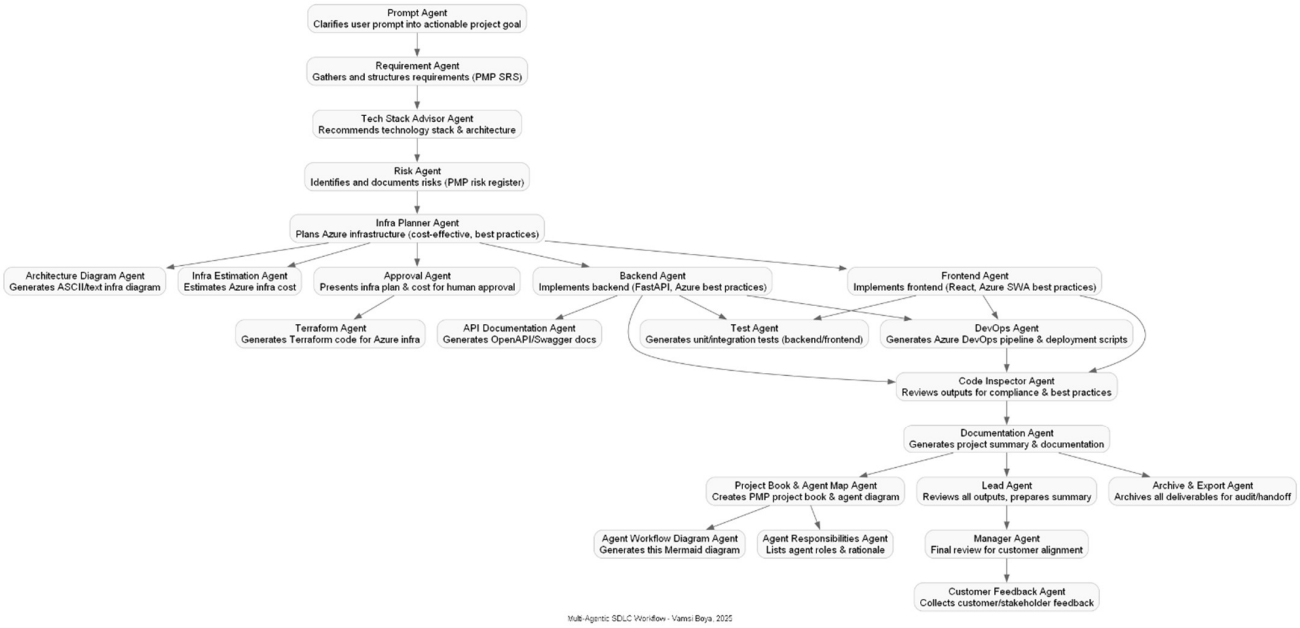
Orchestrating Software Delivery with Agentic AI: A New SDLC Paradigm *by Vamsi Boya*

1. **Introduction**
   This document introduces a cutting-edge Multi-Agentic AI Software Development Life Cycle (SDLC) platform designed to revolutionize software development. By leveraging dedicated AI agents at each stage of the SDLC, this framework ensures end-to-end traceability, adherence to Project Management Professional (PMP) and Azure best practices, and accelerated delivery from initial requirements to final release. This approach addresses the limitations of conventional SDLC methodologies by automating critical functions, reducing errors, and enabling rapid adaptation to evolving business needs.
2. **The Need for Agentic SDLC**
   Traditional SDLC methodologies often suffer from manual processes, increased error rates, and slow adaptation to changing business requirements. An agentic approach, where specialized AI agents manage each phase, eliminates these inefficiencies and enforces rigorous standards. These agents automate key functions, including:

3. **Architectural Diagram of SDLC Agentic AI System:**



4. Agentic AI SDLC - System Architecture Overview

This platform leverages **LangChain** and **LLM**, orchestrating a suite of AI agents:

4.1 Prompt Agent: Clarifies the user's intent and transforms a high-level prompt into a clear, actionable project goal, list of objectives and Success Criteria.

4.2 Requirement Agent: Based on the goal provided by the Prompt Agent, gathers, clarifies, and structures the project requirements (following PMP/Software Requirements Specifications standards:

- Project Overview
- Functional Requirements
- Non-Functional Requirements
- Constraints

- Acceptance Criteria

4.3 Tech Stack Advisor Agent: Selects technology and designs architecture for scalability and compliance which includes:

- High-Level Architecture
- Cost-Effective Resources
- Technology Choices and Rationale
- PMP Alignment and Azure Best Practices
- Key Benefits of Proposed Stack

4.4 Risk Assessment Agent: Identifies and documents delivery, security, and compliance risks which includes

- Technical Risks
- Delivery Risks
- Cost Risks
- Deployment Risks
- Compliance and Regulatory Risks

4.4 Infra Planner: Designs cost-effective, best-practice Azure infrastructure.

- High-Level Infrastructure Breakdown
- Backend Services
- AI/ML Model Hosting & Document Parsing
- Database Layer
- Real-Time Notifications for Billing Updates
- Security & Compliance
- Monitoring and Logging
- Disaster Recovery & High Availability
- Deployment & Cost Control Practices

4.5 Backend/Frontend Agents: Generate application code for both server and client sides.

- Including following:
- Database Configuration
- Authentication
- Models and Schemas
- API Endpoints - Insurance-Related
- CRUD Operations
- AI Integration
- Main Application Entry Point

a. Below is a Sample screenshot of File structure of how Agents build the code for user requested Web Application.

Backend:

i. **ai_insurance_backend/**

```
├── app/
│   ├── __init__.py
│   ├── main.py
│   ├── database.py       # Database connection and models
│   ├── auth.py           # Authentication using Azure AD B2C
│   ├── schemas.py        # Pydantic models (data validation)
│   ├── crud.py           # Database operations
│   ├── ai_module.py      # Azure Cognitive Services Integration
│   ├── endpoints/
│   │   ├── __init__.py
│   │   ├── insurance.py  # Insurance-related APIs
│   │   └── billing.py    # Billing-related APIs
│   ├── utils.py          # Utility functions
└── requirements.txt      # Python dependencies
```

Frontend:

- index.html
- Application Entry Point
- Azure AD Authentication
- API Calls
- Insurance Submission
- Policy Display Component
- Secured Dashboard
- Azure Deployment Configuration
- Deploy to Azure Static Web Apps

ai-insurance-frontend/

```
├── public/             # Contains static assets like index.html
├── src/
│   ├── components/     # Reusable UI components
│   │   ├── PolicyForm.js   # Form for submitting insurance policies
│   │   ├── PolicyList.js   # Component for displaying policies
│   │   └── ProtectedRoute.js # Protect specific routes with Azure AD login
│   ├── pages/          # Views rendered for particular routes
│   │   ├── Home.js     # Landing page
│   │   ├── Dashboard.js    # User dashboard showing insurance info
│   ├── services/       # For API calls and Azure Authentication logic
│   │   ├── auth.js     # Azure AD authentication helpers
│   │   ├── api.js      # API calls to the backend (FastAPI)
│   ├── App.js          # Main application component
│   ├── index.js        # Entry point to render React app
├── package.json        # Dependencies and scripts
└── azure-static-web-apps.config.json  # Azure SWA configuration
```

4.6 API Documentation build by AI Agents for customer requested web application:

- OpenAPI Specification (YAML)
- Key Features in the Documentation
- Azure Toolkit Compatibility

4.7 Unit and Integration Tests conducted by Test Agent: Generates the code required to perform unit and integrations tests.

4.8 DevOps Agent: Automates CI/CD pipelines for consistent, reliable delivery.

- Backend Pipeline YAML (FastAPI)
- Frontend Pipeline YAML (React.js)
- Secrets Management
- Frontend: Environment Variables
- Testing
- Deployment Script for Azure App Service (Backend)
- Deployment Script for Azure Static Web Apps (Frontend)
- Best Practices Checklist

4.9 Code Inspector Agent: Reviews all output for alignment with coding instructions and Azure best practices. Evaluates and provide the Scope.

4.9.1

FINAL EVALUATION:

Score: 8.5/10

While the provided artifacts are well-structured and PMP-compliant, there are meaningful improvements concerning secrets management, deployment slot utilization, testing rigor, and alignment with Azure's advanced deployment tools.

4.10 Documentation Agent: Compiles all deliverables into PMP-compliant documentation.

4.11 Project Book & Agent Map Agent: Produces the complete project guide and workflow diagrams.

4.11.1

Project Manager Review Checklist

- [ ] Review all generated documentation for completeness and clarity.
- [ ] Validate requirements, architecture, risks, and infrastructure plans against business needs.
- [ ] Ensure all code (backend, frontend, tests) follows company and Azure best practices.
- [ ] Confirm DevOps pipeline and deployment scripts are secure and production-ready.
- [ ] Check that all @azure coding instructions have been followed, unless a system message required otherwise.
- [ ] Approve or request changes before moving to production or client handoff.

4.12 Leads/Managers/Customer Feedback Agents: Layered review—technical, compliance, business requirements.

4.13 Compliance & Evaluation Summary for Manager Agent

4.14 Observed Gaps & Issues

4.14.1

Summary of Compliance

Area	Compliant?	Notes
Requirements (SRS)	✅	Detailed functional and non-functional specs.
Architecture	✅	Aligns with Azure microservices design best practices
Backend Code	✅	Modular, adherent to Azure Functions guidelines
Frontend Code	✅	Optimized React app, WCAG accessibility standards
DevOps Pipeline	✅	CI/CD Pipelines PMP-compliant and feature security measures
AI/ML and Cognitive Services	⚠️	Requires clarification on scaling AI inference models.
Testing Coverage	⚠️	Performance metrics (e.g., dashboard latency) need tests.
Risk & Mitigation	✅	Addresses major Azure-specific risks proficiently
Documentation	✅	Comprehensive PMP-compliant with acceptance criteria

4.15 Manager Review Agent: Layered review—technical, compliance, business requirements.

4.16 Customer Goals Alignment like shown in the screenshot.

4.17 Key Deliverables Verification

4.18 Analysis of Gaps and Suggested Improvements

Customer Goals Alignment

The project addresses the customer's defined goals and success criteria effectively:

- **Patient Insurance Submission:** A secure, user-friendly interface has been implemented.
- **Real-Time Bill Visibility:** A secure portal for real-time bill tracking is in place.
- **AI-driven Policy Parsing and Matching:** AI implementations integrate insurance policy parsing and claim tracking.
- **Discharge Summary Automation:** Automated generation of discharge summaries is included.
- **Claim Approval and Flagging:** Effective AI-driven claim handling is implemented.
- **Early Dispute Detection:** Dispute detection mechanisms are in place, enhancing efficiency.
- **Compliance:** Adherence to HIPAA and other relevant standards is assured.

4.19

Summary of Alignment

The project demonstrates strong adherence to the customer's goals and provides solutions likely to succeed in reducing manual processing, minimizing disputes, and complying with data protection regulations. However, the identified gaps need to be addressed to ensure full scalability, integration, and performance testing—ultimately providing a robust and future-ready solution.

Area	Compliant?	Notes
Requirements (SRS)	✅	Fully aligns with functional/non-functional requirements.
Architecture	✅	Meets Azure best practices for modular architecture.
Backend Code	✅	Cleanly coded and optimized for Azure Functions.
Frontend Code	✅	WCAG-compliant and optimized for Static Web Apps.
DevOps Pipeline	✅	Security-compliant pipelines with rollback strategies.
AI/ML and Cognitive Services	⚠️	Scalability needs clarification for high-load scenarios.
Testing Coverage	⚠️	Performance testing for NFR metrics is missing.
Risk & Mitigation	✅	Covers critical risks with proactive mitigations.
Documentation	✅	Comprehensive and PMP-compliant deliverables.

Actionable Requests for Final Review

Before requesting human confirmation for approval, the following actions are recommended:

1. **AI Scalability:** Provide scaling details for AI inference services, such as AKS or auto-scaling settings.
2. **Integration Testing:** Define and implement API validation strategies for the integration of third-party insurance systems.
3. **Performance Testing:** Add load and performance testing scripts to the CI/CD pipeline for validating NFR compliance.
4. **Cosmos DB Partitioning:** Confirm a cost-efficient scalability plan for query-heavy Cosmos DB usage.

4.20

4.21 **Here comes the Human in Loop for Approvals.**

Decision: Prompt Human-in-the-Loop for Approval

The deliverables are of high quality and meet the majority of the project goals. However, given the identified gaps in AI scalability, integration testing, performance testing, and Cosmos DB scaling, human input is recommended before final approval to ensure these gaps are adequately addressed.

Would you like to request refinements to the deliverables before final approval? If so, provide feedback on specific areas or confirm alignment with the improvement plan. Let me know how you'd like to proceed!

4.21.1

4.22 **Customer/Stakeholder Feedback**

4..1 Summary of Deliverables vs. Your Goals:

4..2 Identified Gaps and Recommended Refinements:

4..3 Decision Point: Refinements or Approval

4.23 **Archive & Export Agent:** Secures all deliverables for audit and handoff in standard formats.

```
Saved: requirements.md
Saved: tech_stack.md
Saved: risks.md
Saved: infra_plan.md
Saved: backend_code.md
Saved: frontend_code.md
Saved: api_docs.md
Saved: tests.md
Saved: devops_pipeline.md
Saved: project_documentation.md
Saved: coding_instructions.md
Added to archive: requirements.md
Added to archive: tech_stack.md
Added to archive: risks.md
Added to archive: infra_plan.md
Added to archive: backend_code.md
Added to archive: frontend_code.md
Added to archive: api_docs.md
Added to archive: tests.md
Added to archive: devops_pipeline.md
Added to archive: project_documentation.md
Added to archive: coding_instructions.md
```

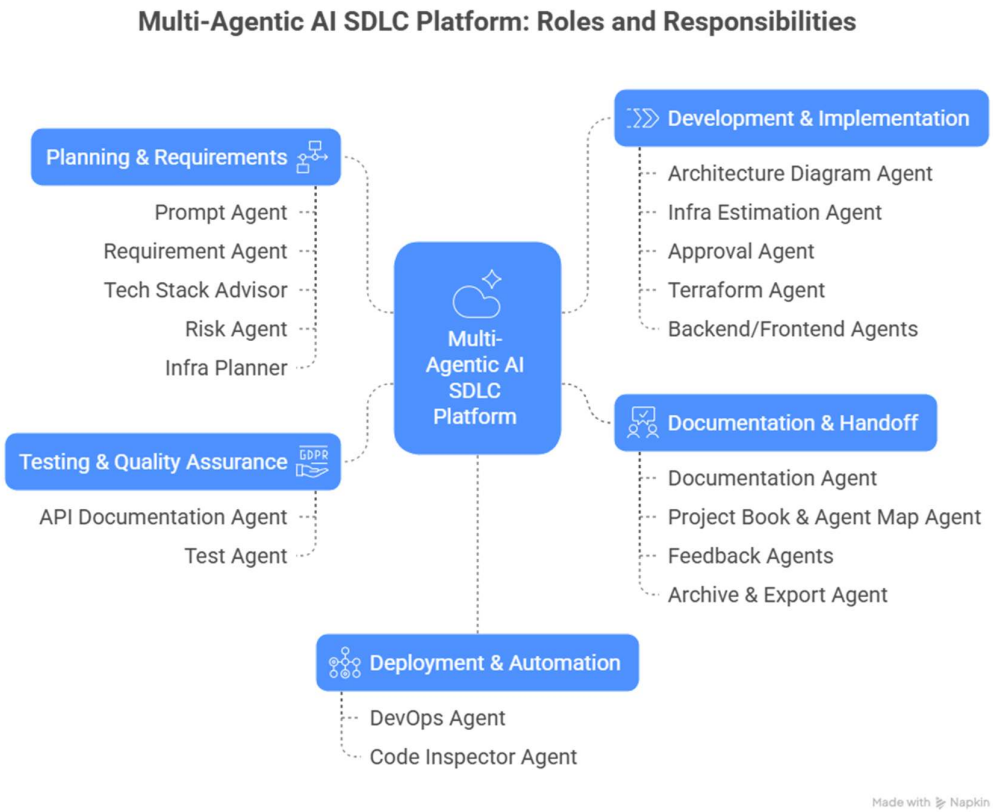
4.23.1 `Archive created: sdlc_project_deliverables.zip`

5 Below is an Example Solution through Multi Agent SDLC following PMP standards:

5.1 Step-by-Step Example: Insurance Verification System

5.1.1 User Prompt (Example):

"Build an AI-powered insurance verification and billing coordination system for hospitals..."



5.2

5.3 How Each Agent Contributes:

5.3.3 **Prompt Agent:** Refines the prompt into a clear, actionable goal:

5.3.4 **5.3.2 Requirement Agent**: Structures PMP-compliant SRS, capturing user stories, HIPAA compliance, system constraints.

5.3.5 **Tech Stack Advisor:** Recommends FastAPI, React, Azure SQL, Azure AD. Delivers architecture diagram and rationale.

5.3.6 **Risk Agent:** Identifies technical, security, and delivery risks (e.g., HIPAA, privacy, integrations) with mitigation strategies.

5.3.7 **Infra Planner:** Designs cost-effective infrastructure (App Service, SQL, Storage) using minimal/Azure-free SKUs, and plans resources.

5.3.8 **Architecture Diagram Agent:** Produces clear diagrams illustrating the solution's components and data flows.

5.3.9 **Infra Estimation Agent:** Estimates Azure costs using real data from the Azure Pricing Calculator for transparency.

5.3.10 **Approval Agent:** Presents infrastructure plan and cost for explicit human sign-off before code is generated.

5.3.11 **Terraform Agent:** Generates modular, best-practice Terraform code aligning with your permissions and security instructions.

5.3.12 **Backend/Frontend Agents:** Implements backend logic (FastAPI, Azure SQL, AD integration), and React frontend (modern UI, secure auth).

5.3.13 **API Documentation Agent:** Creates OpenAPI/Swagger documentation for every endpoint.

5.3.14 **Test Agent:** Generates unit and integration tests for backend (pytest) and frontend (React Testing Library).

5.3.15 **DevOps Agent:** Builds secure, automated Azure DevOps pipelines and deployment scripts.

5.3.16 Code Inspector Agent: Reviews all output for alignment with coding instructions and Azure best practices.

5.3.17 Documentation Agent: Compiles all deliverables into PMP-compliant documentation.

5.3.18 Project Book & Agent Map Agent: Produces the complete project guide and workflow diagrams.

5.3.19 Leads/Managers/Customer Feedback Agents: Layered review—technical, compliance, business requirements.

5.3.20 Archive & Export Agent: Secures all deliverables for audit and handoff in standard formats.

5.4 Project Management & Compliance

All documentation and procedures strictly follow **PMP standards**.

Human-in-the-loop is enforced at key milestones for approval and transparency.

System outputs are audit-ready for every project phase, ensuring peace of mind during reviews or handoff.

5.4.1 Security, Risk, and Cost Management

Every stage incorporates risk identification, assessment, and mitigation.

Infrastructure is architected for cost optimization—making use of free/minimal Azure resources whenever feasible.

Security enforcement includes Azure AD, HIPAA/data privacy, and least-privilege principles.

5.4.2 DevOps and Automation

Automated CI/CD with Azure DevOps pipelines accelerates releases while eliminating manual errors.

Infrastructure as Code (Terraform) for reproducible deployments and rapid environment spin-up.

Integrated unit/integration testing and code auditing by agentic inspectors bolster quality.

5.4.3 Documentation and Handoff

Every output—code, infrastructure, tests, and process documentation—is archived and exportable as Markdown, PDF, or ZIP.

Comprehensive project book and visual agent workflow diagrams included by default.

Handoff is streamlined, making client transitions, audits, or future enhancements secure and straightforward.

5.4.4 Conclusion & Future Directions

The Multi-Agentic AI SDLC platform delivers a scalable, compliant, and high-velocity approach to software delivery. Future enhancements may include support for additional cloud providers, built-in analytics, and release of open-source agent templates to the community.

6 Author's Note: Origin, Vision, and Call to Action

This platform is a result of my drive to blend automation with industry standards. Drawing from years of experience and a passion for engineering excellence, I created this end-to-end, agent-driven system to address modern SDLC pain points and empower others to build with confidence.

6.1 Technologies Used:

Python, Jupyter Notebook

LangChain for agent orchestration

Azure OpenAI for natural language and code generation

Azure DevOps, Terraform, FastAPI, React (artifact generation)

Markdown & Mermaid for unified documentation

6.2 Best Practices Built In:

Azure & PMP standards

Full risk, security, and cost controls

Modular, extensible, and future-ready design

7. Appendix & References

7.1 Official Documentation & Standards

Project Management Institute (PMI) - PMP Standards

Azure Documentation

Azure OpenAI Service

Azure Functions Documentation

Azure Static Web Apps

Azure SQL Database

Azure Key Vault

Azure DevOps Pipelines

Azure Cognitive Services

Azure Event Grid

Azure Application Insights

Azure Active Directory B2C

Azure Logic Apps

Azure Security Center

Azure Monitor

Azure Blob Storage

Azure Machine Learning

- Azure Pricing Calculator

7.2 Open Source Libraries & Tools

- Python
- FastAPI
- React
- LangChain
- OpenAI Python SDK
- python-dotenv
- pytest
- Jest
- React Testing Library
- Terraform
- Graphviz
- Jupyter Notebook

7.3 Other References

- HIPAA Compliance Overview
- Mermaid Diagrams
- Pydantic

Note:
All code, architecture, and process recommendations in this document are based on the official documentation and best practices from the above sources.
Please refer to the respective URLs for the most up-to-date and detailed information