ANNAMMACHARYA INSTITUTE OF TECHNOLOGY AND SCIENCES

Department of Artificial Intelligence and Data Science

Kadapa

INTERNSHIP REPORT

ON

"SmartSDLC – AI-Enhanced Software Development Lifecycle"

Submitted in partial fulfillment of the requirements of the

Virtual Internship Program

Organized by

SMART BRIDGE

Submitted by

G Vasanti (22HM1A3013)

Modagala Jayasri (22HM1A3026)

Thamatam Vennela (22HM1A3042)

Pullagura jashwanth(22HM1A3033)

TEAM ID: LTVIP2025TMID59168

Under the Mentorship of

Mr. M. Ganesh

Smart Bridge

June 2025

Index

BRAINSTORMING &IDEATION

Objective:

Problem Context

Governments often struggle with delayed and inefficient communication with the public. Traditional systems such as phone calls, static websites, or manual helpdesks make it difficult for citizens to get timely and relevant information regarding public services, policies, and civic issues.

Additionally, the government lacks effective tools to understand the overall mood or sentiment of citizens based on their feedback. As a result, citizen trust and engagement with digital governance platforms remain limited.

Purpose of the Project

Citizen AI was developed to solve these challenges by introducing a real-time AI assistant capable of:

- Answering citizen questions with human-like responses using IBM Granite models.
- Analyzing citizen feedback to detect sentiments such as satisfaction or dissatisfaction.
- Presenting actionable insights on a dynamic dashboard.
- Enhancing government responsiveness, transparency, and public trust through AI-powered interactions.

Key Points:

- **Problem Statement**
  - Citizens face delays in receiving information or reporting issues.

- Feedback collected by government departments is rarely analyzed in real time.
- There is no central platform to combine public service interaction, sentiment analysis, and policy responsiveness.

Proposed Solution

- **Citizens:** To ask questions, provide feedback, and get quick answers.
- **Government Officials:** To monitor public sentiment and improve services.
- **Policy Makers:** To use data insights for better governance decisions.
- **Developers/Admins:** To manage system operations and improve AI performance.

Target Users

- **City Residents** – To report civic issues easily and get eco-advice.
- **City Administrators –** To monitor feedback, analyze KPIs, and respond to anomalies.
- **Urban Planners –** To summarize long documents and make informed policy decisions.
- **Teachers & Students –** To explore sustainability practices via the Eco Tips Generator.
- **Government Departments –** For utility monitoring and forecasting resource demands

Expected Outcomes

- A responsive, intelligent chatbot available 24/7 for public queries.
- Real-time analysis of citizen feedback to detect trends and issues.
- Dashboards that visualize public mood and interaction frequency.

- Enhanced digital governance through AI, leading to improved public satisfaction and trust.

REQUIREMENT ANALYSIS

Functional Requirements:

Requirement analysis is the foundational phase of any Software Development Lifecycle (SDLC), including SmartSDLC — an AI-enhanced evolution of traditional SDLC methodologies. In SmartSDLC, Artificial Intelligence augments every stage to improve accuracy, efficiency, and collaboration. Here's how requirement analysis transforms under this intelligent framework:

🔍 1. Overview of Requirement Analysis in SmartSDLC

Definition:
Requirement analysis is the process of identifying, gathering, documenting, validating, and managing software requirements from stakeholders and transforming them into a structured format for system design.

In **SmartSDLC**, AI tools assist and automate many of these tasks, minimizing human error, reducing time-to-delivery, and increasing alignment between stakeholders.

🤖 2. AI-Driven Enhancements in Requirement Analysis

a. Intelligent Requirement Gathering

- **Natural Language Processing (NLP):** AI parses stakeholder conversations (e.g., meeting transcripts, emails, surveys) to extract functional and non-functional requirements.

- **Conversational Agents:** AI-powered chatbots or assistants interact with stakeholders to clarify and document requirements.

b. Automated Requirement Classification

- AI classifies requirements into:
    - **Functional**
    - **Non-functional** (performance, security, usability)
    - **Constraints** (regulatory, budget, timeline)

c. Requirement Prioritization and Validation

- **Machine Learning (ML)** models assess requirement feasibility and importance based on:
    - Stakeholder roles
    - Business impact
    - Historical project data
- AI also predicts conflicts or redundancies among requirements.

d. Traceability and Versioning

- AI ensures **end-to-end traceability** from requirements to test cases and code components.

- Automated version control detects changes and impacts across lifecycle artifacts.

🧠 3. Key Features in SmartSDLC Tools for Requirement Analysis

Feature	Traditional SDLC	SmartSDLC with AI
Requirement Elicitation	Manual interviews, workshops	NLP-based extraction, AI chatbots
Documentation	Manual writing (SRS)	AI-generated and updated documents
Ambiguity Detection	Manual review	AI identifies unclear/incomplete specs
Change Impact Analysis	Manual tracking	Predictive analytics using AI models
Stakeholder Feedback Loop	Scheduled reviews	Real-time AI-driven feedback analysis

---

📊 4. Benefits of AI in Requirement Analysis

- 🔄 **Continuous Alignment:** AI ensures evolving requirements remain aligned with stakeholder intent.
- 🕒 **Faster Time-to-Market:** Automation significantly reduces time spent on documentation and analysis.
- 🧩 **Better Decision Support:** AI models recommend optimal feature prioritization and cost estimation.
- 🔍 **Reduced Errors & Gaps:** Enhanced validation and requirement clarity reduce downstream bugs and rework.

.

🛠️ 5. Example AI Tools Used

Tool Type	Examples	Functionality
NLP Engines	GPT, IBM Watson NLP, BERT	Extract and classify requirements
Requirement Platforms	Requiment.AI, Visure, Modern Requirements	AI-assisted documentation, traceability
Task Automation	UiPath, Zapier	Automate stakeholder notifications
ML Analytics	Custom ML pipelines	Requirement prioritization & risk prediction

---

☑️ 6. Best Practices for AI-Enhanced Requirement Analysis

1. **Involve AI Early:** Integrate AI tools from the initial stakeholder engagement.

2. **Human-AI Collaboration:** Use AI as an assistant, not a replacement for human judgment.

3. **Maintain Data Quality:** High-quality inputs (conversations, surveys, past project data) yield better AI outputs.

4. **Iterative Refinement:** Use AI feedback loops to refine and validate requirements continuously.

PROJECT DESIGN

In SmartSDLC, the Project Design phase is elevated through Artificial Intelligence to enable smarter architecture decisions, automated modeling, and proactive risk management. This phase translates validated requirements into a blueprint for implementation — and AI plays a critical role in ensuring the design is scalable, efficient, and adaptable.

🧱 1. Overview of the Project Design Phase in SmartSDLC

Traditional Goal:
Transform requirements into a structured design, including system architecture, data models, interfaces, and user experience.

SmartSDLC Enhancement:
Incorporates AI tools to automate, optimize, and validate design decisions, reducing manual effort and minimizing design flaws early in the process.

🤖 2. AI-Enhanced Capabilities in Project Design

a. Intelligent Architecture Suggestion

- **AI recommends architecture patterns (e.g., microservices, serverless, monolith) based on:**
  - **Requirements**
  - **Non-functional constraints (scalability, latency)**
  - **Historical project data**
- **Uses trained models and rule-based systems to align design with business goals.**

b. Auto-Generated Design Diagrams

- **AI tools like Generative AI create:**
  - **System Architecture Diagrams**

•

  - **UML Class & Sequence Diagrams**
  - **Entity-Relationship (ER) Diagrams**

- **NLP-based input allows teams to describe components in plain English, which AI translates into technical visuals.**

c. Design Validation and Optimization

- **Simulation and Testing: AI simulates workloads to predict bottlenecks in system design.**
- **Static Analysis: Detects design-level anti-patterns or security vulnerabilities before implementation.**
- **Performance Forecasting: AI models estimate system performance under expected user loads.**

d. UI/UX Design Assistance

- **AI tools generate responsive UI mockups from text descriptions or sketches.**
- **Predicts user flow and suggests layout improvements based on user behavior analytics.**

🧠 3. Key Features of Project Design in SmartSDLC

Feature	Traditional SDLC	SmartSDLC with AI
Architecture Design	Manual, experience-driven	AI-recommended based on context & goals
Diagram Generation	Hand-drawn, tool-based	Auto-generated from descriptions
Security & Performance Checks	Post-implementation	Proactive AI validation at design time
UI Prototyping	Manual design with Figma/Sketch	AI-generated prototypes from user stories
Design Decision Justification	Subjective	AI-supported with rationale and risk metrics

---

🛠️ 4. Example AI Tools for Project Design

AI Tool/Platform	Functionality
GPT + Mermaid or PlantUML	Generate architecture diagrams from prompts
Uizard, Figma AI, Framer AI	Convert text to UI designs
DeepCode, SonarQube AI	Early design flaw detection
ChatGPT (Pro)	Design guidance, best practices, diagram creation
Architecht.ai	AI-assisted cloud architecture design

📊 5. Benefits of AI in Project Design

•

- ⚡ **Faster Design Iteration: Rapidly prototype and test multiple designs.**
- 🧠 **Knowledge Reuse: AI leverages previous project knowledge to avoid reinventing the wheel.**
- 🔍 **Early Risk Detection: Spot architectural weaknesses before they cost money.**
- 🩹 **Data-Driven Decisions: AI provides objective analysis to support design choices.**

📌 6. Best Practices for AI-Driven Project Design

1. **Combine AI with Human Expertise: Use AI to enhance—not replace—design decisions.**
2. **Validate Generated Designs: Always vet AI outputs with senior architects and stakeholders.**
3. **Iterate and Simulate: Use AI simulations to stress-test designs before coding.**
4. **Ensure Model Transparency: Prefer explainable AI tools that justify their suggestions.**
5. **Document Design Rationale: Let AI help generate documentation for future reference.**

📐 7. Example Flow in SmartSDLC Design Phase

1. **Input: Requirements from the AI-validated requirement analysis phase.**
2. **AI Modeling: System architecture suggestion → visual diagrams → UI mockups.**
3. **Simulation: Load testing, latency forecasting, and risk scoring.**
4. **Output: Complete design package (architecture docs, UI/UX flows, APIs, data schema).**

PROJECT PLANNING

Objective:

Project planning is a critical phase in the Software Development Lifecycle (SDLC), where timelines, resources, costs, and deliverables are defined. In SmartSDLC, Artificial Intelligence revolutionizes planning by bringing predictive analytics, intelligent scheduling, and real-time adaptability to the forefront — ensuring projects stay on track, within budget, and aligned with goals.

🔍 1. What is Project Planning in SmartSDLC?

Traditional SDLC Planning involves:

- **Defining scope**
- **Estimating time and resources**
- **Risk management**
- **Creating work breakdown structures (WBS)**
- **Scheduling tasks**

SmartSDLC with AI enhances this by:

- **Automating estimations**
- **Forecasting risks**
- **Optimizing team allocations**
- **Adapting plans in real-time using AI-powered insights**

🤖 2. AI-Driven Enhancements in Project Planning

a. Smart Estimation of Time & Cost

- **AI models use historical project data, team performance metrics, and requirement complexity to:**
    - **Estimate timelines accurately**

-
    - **Predict effort per feature/module**
    - **Calculate cost dynamically**
- **Example: AI predicts a module will take 30 developer hours based on past similar tasks and team velocity.**

b. Intelligent Resource Allocation

- **AI recommends optimal team compositions based on:**
    - **Skill sets**
    - **Availability**
    - **Historical productivity**
- **Tools can auto-assign tasks and balance workloads across team members.**

c. Dynamic Scheduling & Forecasting

- **AI creates and adjusts schedules in real-time as:**
    - **Requirements change**
    - **Delays occur**
    - **New risks are introduced**
- **Gantt charts and sprint plans are auto-adjusted with predictive insights.**

d. Risk Identification and Mitigation

- **AI detects risks early using:**
    - **Pattern recognition from failed past projects**
    - **Code complexity analysis**
    - **Stakeholder behavior signals**
- **Suggests mitigation strategies and impact forecasts.**

e. AI-Powered Work Breakdown Structures (WBS)

- **Automatically generates task hierarchies from requirements.**
- **Maps features to epics, stories, and tasks with estimated effort levels.**

🧠 3. Key Features of Project Planning in SmartSDLC

Feature	Traditional Planning	AI-Enhanced Planning (SmartSDLC)
Time & Effort Estimation	Based on expert judgment	Data-driven, historical ML prediction
Resource Planning	Manual scheduling	AI-optimized based on team skill matrix
Risk Management	Manual checklists	Predictive analytics, real-time updates
Schedule Management	Static Gantt charts	Adaptive AI-driven timeline adjustments
Task Decomposition	Manual WBS creation	Auto-generated WBS from requirements

🛠️ 4. Tools & Technologies in SmartSDLC Project Planning

Tool / Platform	AI Capability
Jira + Advanced Roadmaps	AI sprint forecasting, resource balancing
Microsoft Project AI	Intelligent scheduling, delay prediction
ClickUp AI	Task creation, time estimation, workload planning
Monday.com AI	Timeline automation, team performance analytics
Forecast App	Time, cost, and capacity prediction using AI

•

📊 5. Benefits of AI in Project Planning

- **⏱️ Faster Planning: Generate entire project plans in minutes, not days.**
- **🎯 Better Accuracy: AI reduces human bias and errors in estimation.**
- **🔁 Adaptability: Real-time plan updates when scope or priorities change.**
- **💸 Cost Optimization: Smarter budget allocation based on predicted ROI.**
- **🔍 Proactive Risk Handling: Minimize project failure with early warnings.**

🧩 6. Sample AI-Assisted Planning Workflow

1. **Input: Requirements (from analysis phase)**
2. **AI Estimation: Time, effort, cost**
3. **Task Breakdown: Auto-WBS generation**
4. **Team Assignment: Based on AI-suggested resource matching**
5. **Timeline Creation: Dynamic, auto-updated Gantt/sprint charts**
6. **Monitoring: Continuous AI-based forecasting and adjustment**

📌 7. Best Practices for SmartSDLC Planning

1. **Feed Clean Historical Data: AI estimations depend on quality past project data.**
2. **Involve PMs in Final Review: Let AI suggest, but keep human oversight.**
3. **Monitor Continuously: Use AI tools to re-forecast regularly, not just once.**

4. **Integrate Across Tools: Sync planning AI with requirement, test, and dev tools.**
5. **Build Transparent Models: Use explainable AI to justify estimations and allocations.**

PROJECT DEVELOPMENT

Objective:

The objective of the Project Development phase is to implement, integrate, and test all modules of the Citizen AI platform, transforming design into a fully functional system. This includes building the chatbot, sentiment analysis module, feedback system, dashboard, and database integration, ensuring they work together seamlessly.

Key Points:

⭘ Technology Stack Used:

- **Backend Framework:** Python with Flask (for modular API routing)
  Frontend Technologies: HTML, CSS, JavaScript (with Bootstrap)
- **AI Integration:** IBM Granite LLM (for chatbot and contextual responses)
- **Sentiment Analysis:** Custom Python module using NLP techniques
- **Database:** SQLite (development) / PostgreSQL (production-ready)
- **Configuration & Secrets:** .env file and config.py for environment-based setup
- **Version Control:** Git and GitHub
- **Deployment (Optional):** Docker, cloud hosting options ⭘

Development Process:

•

The development of Citizen AI followed a structured and modular approach. The project was divided into independent yet connected components, ensuring flexibility and easier debugging. Each major feature was implemented and tested step by step, starting from backend setup to AI integration and frontend user experience. Below are the key stages of the development process:

Flask Project Initialization:

- Created the basic Flask structure with app.py, config.py, and .env for secure settings.
- Installed necessary dependencies including Flask, IBM Granite SDK, and SQLAlchemy.

Chatbot Module Implementation:

- Developed chat_routes.py to handle incoming chat messages.
- Integrated IBM Granite API through granite_interface.py to fetch AI responses.
- Designed chat.html and chat.js for real-time user interaction on the frontend.

Feedback & Sentiment Analysis:

Created a feedback form and handled submissions via feedback_routes.py.

- Analyzed sentiment using the analyse_sentiment() function in sentiment_analysis.py.
- Stored feedback and sentiment data into the database using SQLAlchemy models.

Dashboard Development:

- Built dashboard_routes.py to serve sentiment and feedback metrics.
- Designed dashboard.html with embedded charts for real-time analytics using JavaScript.
- Displayed sentiment counts and citizen engagement trends for admin users.

Contextual Response Enhancement:

- Implemented session tracking to maintain chat context.
- Used helpers.py to store previous interactions and send enriched prompts to Granite for more relevant responses.

Database Integration:

- Defined all database schemas (User, Feedback, Sentiment) in models.py.
- Created init_db.py for easy initialization and reset during testing.
- Connected the application to SQLite/PostgreSQL for persistent data handling.

Testing and Final Integration:

- Performed module-wise testing followed by full integration.
- Addressed sync issues between frontend and backend.
- Ensured end-to-end functionality from chat input to dashboard visualization.

⭘ Challenges & Fixes:

API Integration Issues:
Initial connection to IBM Granite API failed due to incorrect headers and key usage. Fixed by properly loading environment variables and validating API structure.

- **Slow Chat Response:**
  LLM responses were delayed due to large prompts. Solved by optimizing the prompt format and reducing input token length.
- **Frontend and Backend Sync:**
  Chat UI failed to reflect real-time responses. Resolved by properly handling async calls and using consistent JSON structures.
- **Sentiment Misclassification:**
  Feedback was inaccurately tagged. Improved model accuracy by refining classification logic and preprocessing input text.
- **UI Freezing & Form Reloads:**
  Long LLM calls caused UI lag. Handled with asynchronous JS and form handling improvements.
- **Database Connection Errors:**
  Faced schema mismatches during early testing. Resolved by standardizing model definitions and resetting the database using init_db.py.

FUNCTIONAL & PERFOMANCE TESTING

Objective:

Testing is a crucial phase in the SDLC that ensures software **meets its requirements** (functional testing) and **performs under expected conditions** (performance testing). In **SmartSDLC**, AI-enhanced testing transforms these traditional activities into **automated, intelligent, and continuous** processes — reducing effort, increasing coverage, and accelerating delivery.

🧪 1. Overview: Testing in SmartSDLC

Type	Goal
Functional Testing	Validate that the software behaves as expected (features, UI)
Performance Testing	Assess system behavior under load (speed, stability, scalability)

With **AI integration**, SmartSDLC augments these processes to:

- Auto-generate test cases
- Predict test coverage gaps
- Optimize test execution
- Detect performance issues early

🤖 2. AI-Driven Functional Testing

a. Test Case Generation

- AI (using NLP & ML) analyzes:
    - Requirements
    - User stories
    - Acceptance criteria
- Then it **automatically generates** test scenarios and edge cases.

b. Test Optimization & Maintenance

- AI detects **redundant or obsolete test cases**.
- Auto-updates tests when code or requirements change (reduced test maintenance effort).

c. Predictive Bug Detection

- ML models flag likely failure points in the application before running tests.
- Prioritize high-risk features using historical defect data.

d. Visual and UI Testing

- AI validates UI/UX behavior across devices and browsers.
- Can detect visual regressions using **image recognition algorithms**.

🚀 3. AI-Driven Performance Testing

a. Smart Load Testing

- AI simulates **realistic user traffic patterns** based on production usage.
- Automatically adjusts test loads to mimic peak vs. normal usage.

b. Anomaly Detection

- AI models analyze performance logs (latency, throughput, memory usage) to:
    - Detect anomalies
    - Predict future bottlenecks

c. Performance Tuning Suggestions

- AI recommends changes in:
    - Infrastructure (e.g., autoscaling strategies)
    - Code (e.g., caching, DB optimization)
    - API usage

🛠️ 4. Tools Supporting AI-Enhanced Testing

Tool	Functional Testing Feature	Performance Testing Feature
Testim / Mabl	AI test generation, auto-maintenance	Simulated user flow testing
Applitools	AI-powered visual testing	–

Tool	Functional Testing Feature	Performance Testing Feature
Functionize	NLP-based test creation, smart regression	–
Katalon Studio AI	Smart test case generation, risk-based testing	Load testing with predictive analysis
Dynatrace / New Relic AI	–	AI-powered anomaly detection & forecasting
JMeter + AI Plugins	–	Intelligent test data generation

🧠 5. Benefits of AI in Functional & Performance Testing

- 🔁 **Test Automation at Scale:** Continuous test generation and execution
- 🎯 **Higher Accuracy:** AI reduces false positives and missed edge cases
- 📉 **Faster Feedback:** Real-time insights into performance degradation
- 💻 **Resource Efficiency:** Predictive models optimize test infrastructure use
- 🔍 **Better Coverage:** AI identifies gaps in testing and expands coverage

---

🧱 6. Best Practices for Smart Testing

1. **Use Requirement-Based Test Generation:** Feed AI with updated requirement docs.
2. **Integrate Testing in CI/CD:** Automate AI-enhanced testing in your pipeline.
3. **Track Test Intelligence Metrics:** Coverage, flakiness, maintenance needs.

4. **Monitor Real-User Behavior:** Use AI insights from real users to fine-tune test scenarios.
5. **Apply Shift-Left Testing:** Begin AI-assisted tests early in development to catch defects sooner.

🧩 7. Testing Workflow in SmartSDLC

vbnet

CopyEdit

Requirements → AI Analysis → Auto Test Case Generation → Continuous Execution →

AI Evaluation of Results → Defect Prediction & Root Cause Analysis →

Optimized Regression Test Suites → Smart Performance Simulation