**Internship Report**

**(April 25-June 20)**

**Submitted by:**

**Rakesh Reddy**

**Contents**

* Internship Overview
* Learnings/Trainings
* Project-Based Technical Exposure
* Formal/Structured Technical Trainings
* Python Basics
* Angular & Spring Boot
* Cloud Computing Fundamentals (GCP)
* Generative AI
* MLOps
* DevOps – Training Summary
* Airbus KT
* Machine Learning & Deep Learning – Training Summary
* Real-time Project Tasks
* Applications Developed
  + Employee Management Application
  + FAQ Chatbot
* Velocity Fabric – Secure Cloud Foundation (SCF)
* Artificial Intelligence (AI) & GitHub Copilot
* GenWizard

**Learnings/Trainings**

During my internship, I was exposed to real-time technologies, tools, and software development practices. My learning is classified into two main categories:

* Project-Based Technical Exposure
* Formal/Structured Technical Trainings

**Technical Skills Acquired**

This part of learning came from the direct involvement in project development, where I applied core technologies to build working solutions.

**● Frontend Development with Angular**  
→ Developed responsive web interfaces using Angular framework  
→ Built reusable components, services, and managed routing  
→ Worked with reactive and template-driven forms  
→ Integrated UI with backend REST APIs for dynamic data display

**● Backend Development with Spring Boot**  
→ Created RESTful APIs using Spring Boot  
→ Performed data persistence using Spring Data JPA and connected with H2/MySQL  
→ Understood layered architecture (Controller → Service → Repository → DB)  
→ Used Postman for testing and debugging API endpoints

**● Generative AI (GenAI)**  
→ Understood concepts of Machine Learning and Large Language Models (LLMs)  
→ Learned prompt engineering, token usage, temperature setting, and context window.  
→ Explored real-world GenAI applications like summarization, chatbot creation, and content generation

**Trainings Attended**

To strengthen my foundation and broaden my understanding of emerging technologies, I also underwent several internal and self-paced trainings:

**· Python Basics:** This training covered core programming constructs such as data types, control structures, functions, and object-oriented

programming in Python, which helped enhance my programming fluency.

**· Angular & Spring Boot:** This training involved in discussing the installation of angular and spring boot along with the explanation of starter code.

**· Cloud Computing Fundamentals (GCP):**

This training introduced the **basics of Cloud Computing**, focusing on the **Google Cloud Platform (GCP)** and real-world applications.

**Cloud Types & Service Models**

* **Cloud Types**:
  + **Private** – Dedicated infrastructure
  + **Public** – Shared over the internet
  + **Hybrid** – Mix of private + public
  + **Multi-cloud** – Using multiple providers together
* **Service Models**:
  + **IaaS** – Infrastructure as a Service (e.g., Virtual Machines)
  + **PaaS** – Platform as a Service (e.g., App Engine)
  + **SaaS** – Software as a Service (e.g., Gmail, Google Drive)

**Major Cloud Providers**

* **AWS**, **Azure**, **GCP**, **IBM**, **Oracle**, **Alibaba**

Choice depends on:

* + **Cost**
  + **Available services**
  + **Security**
  + **Global reach & uptime**

**GCP Core Concepts**

* **VPC (Virtual Private Cloud)** – Creates isolated private networks
* **Subnets** – Divides VPC into smaller IP ranges
* **VMs** – Virtual Machines to run apps
* **Buckets** – For object storage (via Cloud Storage)
* **Databases**, **VPNs** – For data handling & secure connections

**Application Architectures**

* **Monolithic**:
  + Single, large app
  + One database
  + Hard to update or scale
* **Microservices**:
  + Split into smaller independent services
  + Each has its own DB
  + Scalable and easier to manage

**DevOps & Cloud Integration**

* **Tools Used**:
  + **Cloud Shell** – Browser-based CLI for managing GCP
  + **Cloud Build** – CI/CD tool for building and deploying
  + **Docker** – For containerizing apps
* **Other Concepts**:
  + Use of **YAML/Docker files** for deployments
  + **Source Control** (e.g., Git) for managing code
  + **Cloud Shell Editor** for development inside the cloud

**· Generative AI:**

This training program provided hands-on knowledge of **Generative AI (GenAI)**—from understanding fundamentals to building intelligent AI systems.

* **AI Types**:
  + **Generative** – Creates (e.g., text, images)
  + **Discriminative** – Classifies (e.g., spam detection)
  + **Predictive** – Forecasts (e.g., stock prices)
  + **Conversational** – Interacts (e.g., chatbots)
* **LLMs (Large Language Models)**:  
  Powerful language models trained on massive datasets to generate human-like responses.
* **Tokens**:  
  Basic units of text that determine the **cost** and **input/output length**.
* **Temperature**:  
  Controls randomness in output.
* **Context Window**:  
  Defines how much previous input the model can remember during a conversation or task.
* **Prompt Engineering**:  
  The skill of crafting effective prompts to maximize AI performance and output quality.
* **Structured Output**:  
  Ensures AI responses follow a specific format, making them usable in real-world applications.
* **Function Calling**:  
  Allows AI to trigger external functions or access data/services beyond text generation.
* **APIs**:  
  Connect AI models to external systems for real-time data and integration.
* **Agents**:  
  Autonomous, goal-driven AI systems that can plan, reason, and take action.
* **Architecture**:  
  Design patterns and structures that ensure performance, scalability, and reliability of AI systems.

**· MLOps:**

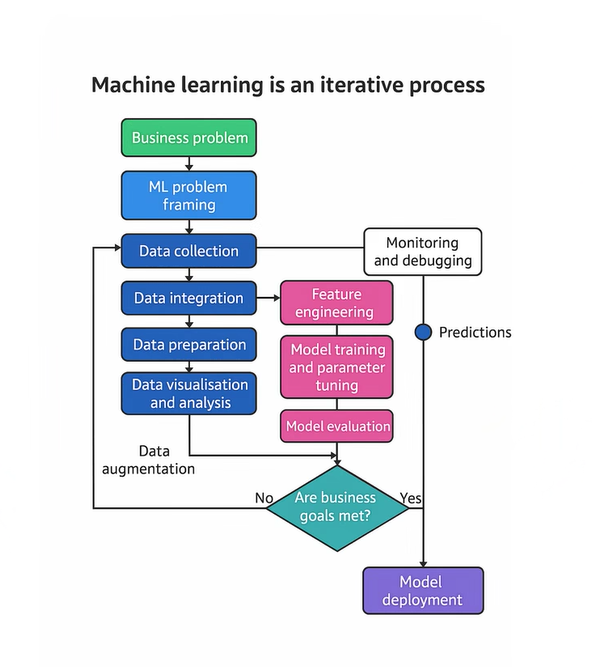
This training introduced the concept of **MLOps**, which is the combination of **Machine Learning (ML)** and **DevOps** practices to manage ML models in production.

**What is MLOps?**

* **MLOps** = ML + DevOps
* Aims to **automate and streamline** the **entire ML lifecycle** — from data preparation to model deployment and monitoring.

**Key Concepts Learned**

* **DevOps**: Automates software development & deployment
* **ML**: Builds models using data and algorithms
* **MLOps**: Bridges both to automate:
  + **Model Training**
  + **Model Versioning**
  + **CI/CD for ML models**
  + **Deployment**
  + **Monitoring & Retraining**



**Why MLOps?**

* Ensures **reliable**, **repeatable**, and **scalable** model deployments
* Reduces manual steps and human error in the ML workflow
* Helps track models, datasets, and performance over time

A screenshot of a computer

AI-generated content may be incorrect.

**· Airbus KT:**

**Aircraft Identification**

* **MSN (Manufacturing Serial Number)**: Unique ID for each aircraft
* **Serie**: Version + Engine type (e.g., SA-A321-200-PW)
* **Model**: Aircraft family (e.g., A321-200)

**Context Builder**

* A UI to configure an aircraft using MSN and Serie.
* Selects required sections for feeding and exporting.
* Used before feeding or sending data to suppliers.

**Digital Mock-Up (DMU)**

* 3D model of an aircraft used to view, verify, and test designs.
* Helps reduce physical prototyping.

**Feeding Process**

Used to load aircraft data into the system.

**Types:**

* **Full Feeding** – All data
* **Delta Feeding** – Only new/changed data
* **Partial Feeding** – Specific parts

**Steps:**

1. **Load DMU** – Raw data stored
2. **Load Boom** – Business logic applied
3. **Load Master** – Data formatted for 3D view

**uDEX (Unified Data Exchange)**

* Used to securely send aircraft data to suppliers.
* Exports: **External** and **NATCO**
* Tabs: **Subscriptions**, **Monitoring**, **Admin**
* Uses **File Transfer Service (FTS)** for delivery.

**Tools Used**

* **Azure DevOps** – Deployment
* **Jenkins** – Task automation
* **Swagger** – API testing
* **Seq** – Log viewer
* **Trend** – Displays live data
* **adsv2/adsv3**, **nova**, **ADF** – Backend services

**Enterprise Content Management**

* **FileNet** (IBM): Used to manage and store documents securely.

**Environments**

* **Validation** – Testing
* **Production** – Live data

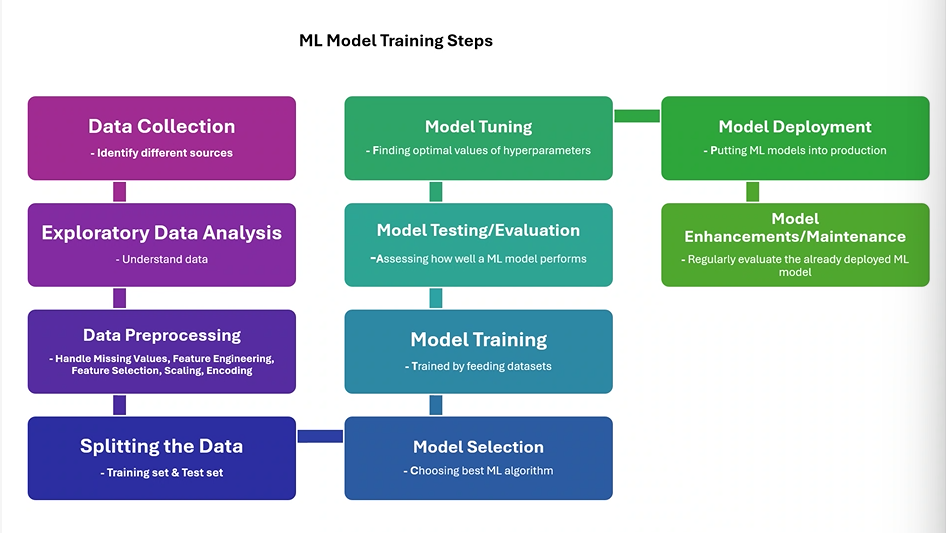
**Machine Learning & Deep Learning**

**Overview**

This training provided a foundational understanding of **Artificial Intelligence (AI)** and its subfields, including **Machine Learning (ML)**, **Deep Learning (DL)**, and **Generative AI (LLMs)**.

**🔁 AI to LLM – Evolution**

Artificial Intelligence → Machine Learning → Deep Learning → Large Language Models (LLMs)



**Types of AI**

1. **Reactive Machines** – No memory, responds to present inputs
2. **Limited Memory** – Uses past data (e.g., self-driving cars)
3. **Theory of Mind** – Understand emotions and intentions (future goal)
4. **Self-awareness** – Conscious AI (theoretical)

**Stages of AI**

* **ANI** – Artificial Narrow Intelligence (current systems)
* **AGI** – Artificial General Intelligence (human-level intelligence)
* **ASI** – Artificial Super Intelligence (beyond human capabilities)

**Machine Learning Types**

* **Supervised Learning** – Uses labeled data (e.g., Linear Regression, KNN)
* **Unsupervised Learning** – Finds patterns in unlabeled data (e.g., Clustering)
* **Reinforcement Learning** – Learns via rewards & penalties (e.g., Q-learning)

**Deep Learning**

* A subfield of ML using **neural networks with multiple layers**
* Best for large-scale data and complex tasks (e.g., image recognition, NLP)
* A diagram of a neuron system

  AI-generated content may be incorrect.

**Generative AI & LLMs**

* **LLMs (Large Language Models)** like ChatGPT, Gemini, Bard
* Learn from large text data to generate human-like responses
* **Key Components**: Pre-training, fine-tuning, tokenization, attention mechanisms

**Challenges & Threats in AI**

* Job Displacement
* Bias & Fairness Issues
* Privacy & Security Concerns
* Lack of Accountability
* Ethical Concerns
* High Resource Requirements
* Dependence on AI / Loss of Control
* Existential Risks

**Responsible AI**

Key principles to develop ethical and safe AI systems:

* Fairness
* Transparency
* Accountability
* Privacy & Security

**Real-time project tasks**

As part of my internship, I worked on two real-time tasks that gave me practical experience with document management systems, source code analysis, and data verification processes used in enterprise environments.

**1. Identifying Document Owners' Valid Email IDs**

In this task, I was responsible for finding and verifying the correct email addresses of document owners from the Production Department. These documents were stored in the Enterprise Content Management (ECM) system, which is used to securely manage and organize official content across the organization.

The process involved:

· Searching for the owners' email addresses using multiple internal tools such as:

* IAM Data Checker
* My Access
* Google Contacts

This task helped me understand how large organizations manage identity and access information and gave me exposure to internal tools used for data validation.

**2. Filenet upgrade Code Module Validation**

The second task I worked on involved comparing Java archive (JAR) files—specifically:

* JAR files that were deployed in production about a year ago, and
* The latest modified JAR files.

I extracted the source code from both versions, performed a detailed comparison, and documented the differences in an Excel sheet, which included:

* File name
* Old code lines (from the production JAR)
* New code lines (from the modified JAR)

This task improved my understanding of version control, software deployment practices, and the importance of traceability in software changes.

Virtual Environment Setup – DEREX

To begin working on the project, I had to set up a virtual machine called DEREX using my Airbus credentials. This secure virtual environment allowed me to access internal systems, tools, and data safely. It was an essential part of ensuring that all tasks were performed in compliance with company security standards.

**Applications Developed**

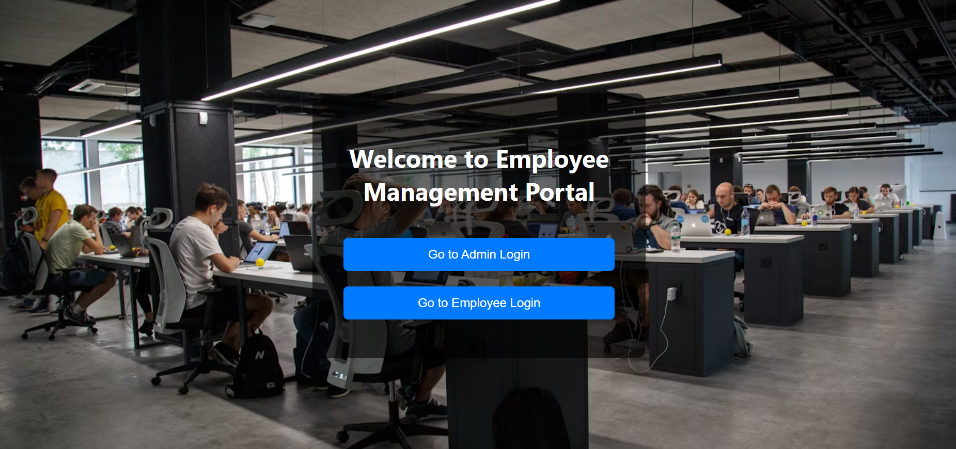
During my internship, I designed and developed two applications that helped me apply the technical skills I learned in real-world scenarios. These applications were built using modern frameworks and demonstrate my understanding of both frontend and backend development, as well as working with machine learning models.

**1.Employee Management Application**

This is a full-stack web application designed to manage employee information, leave requests, and announcements. It supports two roles – **Admin** and **Employee**, each with distinct privileges and views.

**Key Features:**

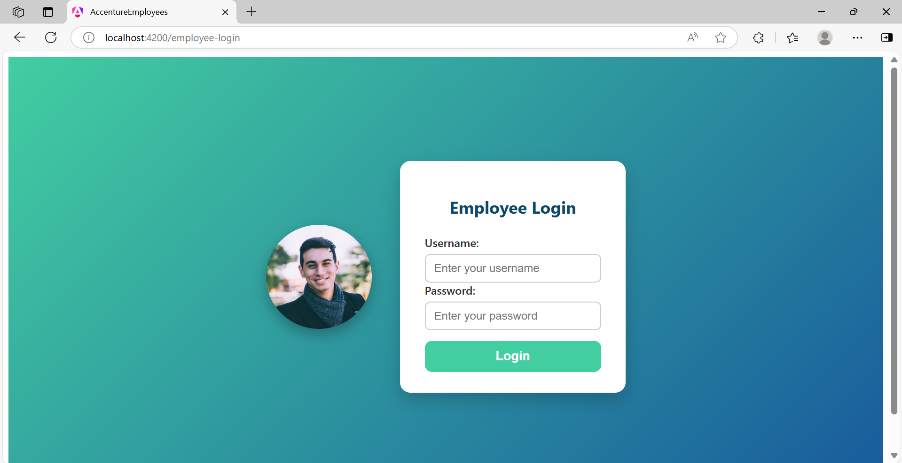
* **Employee and Admin Login**



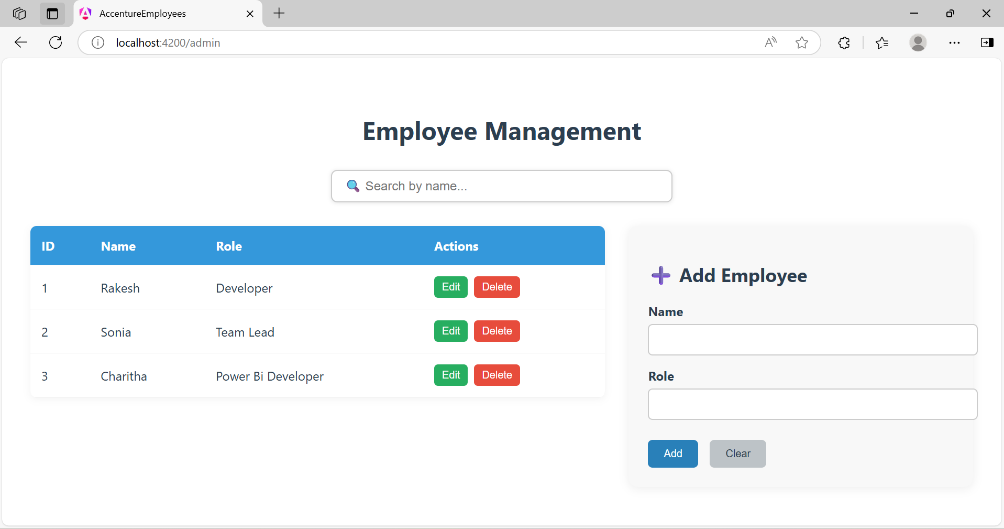
* **Role-Based Access Control**:
  + Admin can view and manage all employee profiles.
  + Employees can view and edit only their own profile.

A screenshot of a computer

AI-generated content may be incorrect.



* **Employee Profile Management**:
  + CRUD operations for employee details (Name, Role, Email).

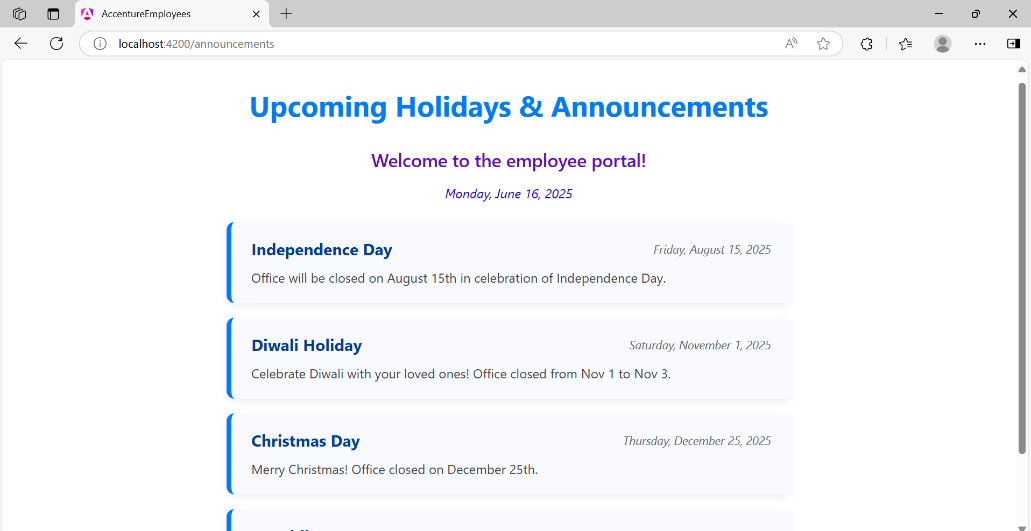


* **Leave Management**:
  + Employees can apply for leaves.
  + Admin can view leave requests

A screenshot of a computer

AI-generated content may be incorrect.

* **Announcements**:
  + Admin can create announcements.
  + Employees can view announcements relevant to them.



**Login Page:**

* Secure login for both Admin and Employee roles using hardcoded credentials (can be extended to use authentication service).

**Admin Dashboard:**

Admins can:

* View all employees
* Edit employee details
* Delete or add employees
* Navigate to employee profile or leave requests
* Post announcements

**Employee Profile Page:**

Employees can:

* View and update their personal details
* Apply for leave
* View announcements posted by the admin

**Announcement Page:**

Displays:

* Title
* Message
* Date and Time

**Technologies Used:**

* **Frontend**: Angular
* **Backend**: Spring Boot
* **Database**: H2 (In-Memory Database)

**Learning Outcome:**

This project helped me understand **full-stack development**, including:

* REST API integration between frontend and backend
* Angular component communication and routing
* Role-based conditional rendering
* Form handling and validation
* Managing in-memory databases with Spring Boot

**2. FAQ Chatbot**

The second application is an AI-powered chatbot that answers questions based on the content of a document uploaded by the user.

Functionality:

* + Users can upload a document (PDF or text)
  + The chatbot processes the content and answers questions asked by the user
  + It uses two machine learning models from Hugging Face:
* One for Question Answering
* Another for Converting document chunks into vectors for better context understanding

**Technologies Used:**

* + Python
  + Hugging Face Transformers
  + NLP Techniques for chunking and vectorizing text

This project gave me practical experience with natural language processing (NLP), using pre-trained models, and working with real-time document-based question answering systems.

A screenshot of a document

AI-generated content may be incorrect.

A screenshot of a phone

AI-generated content may be incorrect.

**Velocity Fabric – Secure Cloud Foundation (SCF)**

**Velocity Fabric** is a platform co-developed by **Accenture and AWS** to accelerate cloud migration by automating infrastructure setup and reducing risk, cost, and time.

**⮕ Secure Cloud Foundation (SCF)**

**SCF** is a pre-built, automated, and secure solution to set up AWS infrastructure using reusable components called **blocks**, developed with **AWS CDK (Cloud Development Kit)**.

It simplifies and automates key infrastructure tasks such as:

* Creating AWS accounts
* Setting up secure networking, firewalls, and permissions
* Enabling encryption, logging, and monitoring tools

SCF follows AWS best practices and supports both **greenfield** (new) and **brownfield** (existing) environments.

**⮕ Key Benefits**

* Fast and consistent deployment of cloud environments
* Reduced manual effort and cost
* Built-in security features like encryption, IAM, and monitoring
* Scalable plug-and-play integration with enterprise tools

SCF helps enterprises modernize IT systems securely and efficiently, with infrastructure that is ready for production from day one.

**AI & Github Copilot:**

**Artificial Intelligence (AI)**

* **Definition**: AI refers to the ability of machines to perform tasks that usually require human intelligence.
* **Key Capabilities**:
  + Understanding natural language (speech/text)
  + Image and sound recognition
  + Learning from data (Machine Learning)
  + Decision-making and predictions
  + Problem-solving abilities (e.g., in games or driving)
* **Types of AI**:
  + **Narrow AI**: Focused on specific tasks (e.g., Siri, ChatGPT)
  + **General AI**: Human-level intelligence (still theoretical)
  + **Superintelligent AI**: Beyond human intelligence (hypothetical future concept)
* **Takeaway**: AI helps systems mimic human-like thinking and behavior to varying extents.

**GitHub Copilot**

* **Overview**: GitHub Copilot is an AI-based coding assistant developed by GitHub and OpenAI.
* **Functionality**:
  + Suggests relevant code snippets in real-time
  + Works within popular IDEs like VS Code and JetBrains
* **Working**: Powered by machine learning models trained on vast code repositories.
* **Installation**: Simple plugin/extension setup in supported IDEs.
* **Purpose**: Enhances developer productivity and reduces time spent on repetitive coding tasks.

**Gen Wizard:**

**GenWizard** is Accenture’s patented platform designed to drive end-to-end automation across the **Technology Delivery Life Cycle (TDLC)** using **Generative AI** at its core. It accelerates enterprise transformation by delivering scalable, secure, and intelligent automation solutions.

**⮕ Key Capabilities**

* Integrates scripting, diagnostic AI, predictive AI, and GenAI.
* Supports 150+ TDLC use cases and leverages 350+ patented assets.
* Built with **multi-LLM Switchboard architecture** for scalability and flexibility.
* Includes project-contextual knowledge engine, orchestration, and prompt generation.
* Follows secure, compliant, and responsible AI practices.

**⮕ Value to Clients**

* Acts as an entry point to GenAI at scale and speed.
* Offers prebuilt GenAI use cases for AMS-AM, IMS, AD, and SI.
* Seamlessly integrates with existing client tools and Accenture platforms like **myWizard**, **myNav**, and **myConcerto**.

**⮕ Platform Benefits**

* Living knowledge base and hyper-automated development workflows.
* Code generation, refactoring, translation, and test script automation.
* Supports modernization and portfolio rationalization.

**⮕ Reverse Engineering Support**

* Designed for legacy systems with missing or outdated documentation.
* Extracts knowledge from source code using LLMs and stores results in vector/graph databases.
* Enables both reverse and forward engineering workflows.

**⮕ Deployment Models**

* SaaS (API-based, Accenture-hosted)
* Dedicated (Accenture or client-managed)
* Custom (GenLite) for client-specific open-source integration