Rahul Mitra

Senior Software Engineer & AI Systems Developer

rahul1991mitra@gmail.com

7000593372

Durg, Chhattisgarh 491001

Married

https://github.com/rahul-mitra

https://rahulmitra.dev

https://linkedin.com/in/rahul-mitra-2313aa17b

Profile

Full-Stack Developer and AI Systems Engineer with 7+ years of experience, including 6+ years working fully remote, designing, developing, and deploying scalable web applications across globally distributed teams. Proficient in Angular, ASP.NET Core, Node.js, and TypeScript, with a strong focus on RESTful APIs and intuitive UI/UX. Extensive experience leveraging Microsoft Azure for cloud-based solutions, including communication services, event-driven architectures, and scalable deployments. Hands-on expertise designing and delivering multi-agent LLM systems, RAG pipelines, and MCP (Model Context Protocol) servers on Azure. Deep knowledge of Azure AI Foundry, Azure OpenAI (GPT-4o, embeddings), Google Gemini/ADK, and AI coding tools including VS Code Copilot (with custom MCP servers), Claude Code, and GitHub Copilot. Strong C#/.NET Core background with proven ability to read, debug, and extend C# codebases. Adept at agile methodologies, mentoring junior developers, and contributing to open-source projects.

Education

Pune, M.H
CDAC (Post Graduation Diploma in Advanced Computing)
Sunbeam

Durg, C.G
Bachelor of Computer Science and Engineering
Chhattisgarh Swami Vivekanand Technical University

Professional Experience

11/2025 – 05/2026
Remote

Senior Software Engineer — AI & Platform
Tytan Technology Inc. (*Deployed to Quantum Capital Group*)

- Architected and delivered LPNexus, an investment operations platform for LP relationship management, with AIassisted meeting note extraction via Azure OpenAI GPT, DDQ semantic search with vector embeddings, async tearsheet generation (PDF/DOCX) over Redis streams, and DealCloud CRM + Cvent event sync.
- Designed QFI Chat Orchestrator: a multiagent AI backend using Google Agent Development Kit (ADK) and FastAPI; root agent (Gemini 2.5 Flash) delegates to subagents for Databricks/Genie structured queries, MongoDB vector document search, financial calculations (IRR, XIRR, NPV, MOIC), and a summarizer agent.
- Built and deployed a production MCP (Model Context Protocol) server exposing document search tools to VS Code Copilot, Claude Desktop, and Claude Enterprise via stdio (local) and streamable HTTP (remote, with Entra ID OAuth 2.0 peruser token validation); configured MCP client settings for all three platforms.
- Developed QFI Metadata Extraction: threestage async pipeline — Gemini OCR + LLM structured extraction → Azure OpenAI embeddings (textembedding3small) → MongoDB/CosmosDB bulk write — ingesting Oil & Gas PE documents (PDF, Word, Excel) into a searchable multifield vector knowledge base.

- Built QIQ (Quantum IQ), a companywide AI platform wrapping Azure OpenAI, with Node.js/Express TSOA APIs, Inversify IoC, MSAL/Passport.js Azure AD auth, OBO token exchange for scoped Databricks credentials, and Azure Python Functions for ETL.
- Tuned Azure OpenAI embedding deployments and designed multifield vector indexes in MongoDB Atlas with configurable score thresholds and relevance fields (summary, extended summary, chunklevel vectors).
- Set up Azure DevOps CI/CD pipelines; Dockerised all services (Node 22 Alpine, Python 3.10); deployed to Azure Container Apps / Azure Web Apps with Application Insights telemetry.

07/2021 – present
Remote from Durg, Chhattisgarh

Senior Software Engineer
InnoWise LLC

- Led development of a cloudbased, multitenant RPA platform using Angular, Node.js, and .NET Core (C#); managed microservice deployments on Azure and automated VM provisioning with Azure SDK.
- Integrated Azure Document Intelligence (Form Recognizer) for OCR and structured data extraction from scanned documents; built rolebased review workflows with multilevel validation.
- Integrated Azure OpenAI APIs to power a contextual LLM chatbot in the RPA workflow designer for guided project creation, debugging, and platform navigation.
- Built an RPA governance platform in C#/.NET Core: job queuing with prioritisation and retries, bot/machine management, Azure Blob Storage versioning, multichannel alerts (Twilio SMS, Azure Communication Services).
- Designed the .NET Core microservices execution layer: Transpiler Service (JSON → CoreWF XAML), Connection Service (Socket.IO/SignalR orchestration), and crossplatform Electron Robot desktop app.
- Received Employee of the Month award three times; mentored junior developers; conducted technical interviews.

07/2020 – 06/2021
Remote from Durg, Chhattisgarh

Software Engineer
InnoWise LLC

- Built realtime applications using Angular, Node.js, .NET Web API, Socket.IO, and SignalR; developed Windows and Ubuntu services in .NET Core 5.0+.
- Designed a custom Named Pipe communication framework in C#/.NET Core for highvolume, lowlatency IPC between RPA microservices (Robot, Executor, Connection Service), with eventdriven architecture, buffered streaming, and threadsafe message handling.
- Built a secure crossplatform Script Execution Service (.NET Core) for remote PowerShell/Bash automation with JWT auth, audit logging, and rolebased access.
- Contributed to socket.ioclientcsharp open source: introduced pluggable serialisation interface (System.Text.Json / Newtonsoft.Json) to fix largepayload handling; maintained production fork across the entire RPA platform.
- Awarded Rising Star (2019–2020) and Star Performer (2020–2021).

Projects

2025 – 2026

QFI Multi-Agent AI Chat System
Multi-agent financial intelligence backend — Quantum Capital Group

Tech Stack: Python, FastAPI, Google ADK (Gemini), Redis, Azure OpenAI, MongoDB/ CosmosDB, Databricks Genie, MCP

Role: Full Stack

- Root agent (Gemini 2.5 Flash via ADK) orchestrates db_query_agent (Databricks/ Genie), doc_search_agent (MongoDB vector search), financial_calc_agent (IRR/XIRR/ NPV/MOIC), and summarizer_agent; async job queue returns task_id immediately — workers deliver results via SSE event streams.
- MCP server with dual transports — stdio for local clients (VS Code Copilot, Claude Desktop) and streamable HTTP with peruser Entra ID OAuth 2.0 for Claude Enterprise and custom integrations.
- Metadata extraction pipeline: threestage async process (Gemini OCR + structured LLM extraction → Azure OpenAI embeddings → MongoDB Atlas multifield vector indexes) ingesting PDF, Word, and Excel PE documents.
- LLM judge / model evaluation module for benchmarking Azure OpenAI vs Gemini response quality.

LPNexus — Investment Operations Platform
AI-assisted LP relationship management — Quantum Capital Group

Tech Stack: Node.js, TypeScript, Angular 17, PrimeNG, MongoDB/CosmosDB, PostgreSQL, Azure OpenAI, Redis, Azure Functions (Python), DealCloud, Microsoft Graph, Loopio

Role: Fullstack

- Call Notes: full meeting note lifecycle (draft → submit → archive) with DealCloud CRM sync, autosave per user/note, email distribution via Microsoft Graph, and failurepath draft notifications.
- DDQ Management: Azure Queuetriggered Python Function ingests DDQ emails and attachments (Word, PDF, Excel); Azure OpenAI extracts and classifies questions against an internal taxonomy using fewshot LLM prompts; Loopio vector search fetches candidate answers; QFI AI service queried per question; results bulkupserted to MongoDB.
- LP Requests (PMT): Azure Logic App captures inbound LP emails to Blob Storage; Python pipeline parses and enriches each request via Azure OpenAI; data stored in PostgreSQL with full UPSERT and duplicate detection; Angular tracker UI with inline editing, filtering, and status workflow.
- Tearsheet Generation: Redis streambased async pipeline; TearsheetPublisher Azure Function schedules bulk jobs daily; LPtearsheet_generate and RedisStreamTrigger Functions handle ondemand PDF/Word generation using DealCloud company data and OpenAI; output published via Node API.
- Azure Functions ETL layer (Python, 14 functions): scheduled DealCloud sync jobs for contacts, companies, funds, interactions, and internal users (hourly to daily); DDQ cleanup; LP requests CSV export to Blob Storage; stalerecord softdelete jobs.
- Services built: AzureOpenAIService (chat + embeddings with retry/token logging), DealCloudService, MicrosoftGraphService, PdfGenerationService (Puppeteer), WordGenerationService (docx), AttachmentService (Azure Blob Storage), RedisService.
- Azure AD authentication with MSAL (frontend) and Passport.js BearerStrategy (backend); separate admin OAuth flow; full RBAC enforcement across all modules.

2025 – 2026

QIQ — Quantum IQ Enterprise AI Platform
Company-wide multi-modal AI platform for Quantum Capital Group employees and applications

Tech Stack: Node.js, TypeScript, Angular, PrimeNG, CosmosDB, Mongoose, Azure OpenAI, OpenAI, Databricks, Azure Functions (Python), Redis, Agenda, Azure AD

Role: Fullstack

- Built a companywide AI platform spanning 8 chat modes: Vanilla Chat (general LLM), Document Chat (RAG over uploaded docs), Document Summary (CIM and vanilla summarization), Structured Chat (Excel/table Q&A with reasoning model support), Delta Chat (Databricks Delta Table queries), ESG Chat (combined structured + unstructured), Code Interpreter, and Deep Research (OpenAI o4minideepresearch with realtime web search and citation generation).
- Designed a pluggable multiprovider LLM layer (LLMServiceProvider) supporting Azure OpenAI, OpenAI (direct), and Databricks — with a Model Registry for perfeature model configuration and hotswappable deployments.
- Built ChatResponseOrchestrator for parallel structured + unstructured response coordination: both paths run concurrently and are combined via a secondary LLM call when both respond; supports ESG and standard document chat types.
- QueryClassificationService rewrites conversational queries into standalone searchready questions using windowed chat history and LLM prompting before routing to semantic search or structured data functions.
- Azure Functions (Python): AssistantChat (OpenAI Assistants API with thread/session management), StructuredChat (Excel table Q&A with reasoning model support and iteration limits), DeltaChat (Databricks Delta Table natural language queries), DocumentSummary, TableSummary, PdfConverter, ProcessContext.
- Semantic search and document embedding pipeline: TextEmbeddingService, DocumentService, ExcelDocumentService, document preprocessor/processor/ summary services backed by MongoDB Atlas vector search.
- Realtime streaming via WebSocket; Agenda job scheduler for background tasks; Redis for caching and sessions; Winston structured logging; Azure AD MSAL/Passport.js auth with full RBAC; TSOA OpenAPI spec autogeneration.

2020 – Present

Cloud RPA Platform

Collaborative Workflow Automation Studio

Tech Stack: Angular, Node.js, MongoDB, Redis, CoreWF, .NET Core APIs, Socket.IO , Power BI, OpenAI (LLM API)

Role: Full Stack Developer / Platform Architect

- Designed and built a **cloud-native, multi-tenant RPA platform** enabling users to create, manage, and deploy automation workflows collaboratively and at scale.
- Implemented **real-time collaboration** using **Socket.IO**, allowing multiple users to co-edit workflows simultaneously with live sync and state consistency.
- Built backend microservices using **Node.js and .NET Core**, coordinating workflow management, deployment, execution routing, and user activity.
- Integrated **Governor APIs**, allowing users to **deploy completed RPA projects directly from the cloud** to on-prem execution environments.
- Developed **dynamic JSON-based workflow authoring**, later converted into CoreWF-compatible XAML via a transpiler microservice.
- Embedded **Power BI dashboards** for live visibility into:
  - **ROI, success/failure rates, manhours saved (FTEs), and revenue generated** by RPA projects
  - Custom filtering and drill-down by tenant, bot, or time period
- Built a comprehensive **Admin Dashboard** to:
  - Provision and manage **new tenant domains** dynamically
  - Control **feature access**, manage **license tiers**, and onboard clients in real time
- Developed a **task assignment module** enabling users to:
  - Assign tasks to team members, link them to specific RPA projects, and track progress
  - View workload distribution and task status across users and teams
- Integrated **OpenAI APIs** to assist users via a contextual LLM chatbot, offering:
  - Help with project creation, debugging workflows, understanding activities, and platform navigation

- Implemented **user authentication and role-based access control**, project versioning, notifications, audit logging, and activity library customization.
- Supported real-time feedback from bots using **Socket.IO**, enabling cloud-triggered executions to reflect progress, logs, and exceptions immediately.

2020 – Present

RPA Governance Platform

A cloud-based scheduling and asset management platform

Tech Stack: .NET Core, Angular, MSSQL, Azure Blob Storage, SignalR, Twilio SMS, Azure Communication Services (SMS & Email), REST APIs

Role: Full Stack Developer / Backend Lead

- Architected and developed a robust **governance platform** to centrally manage bots, machines, packages, execution queues, and environment assets for an enterprise-scale RPA solution.
- Implemented a **Queueing System** to handle job dispatching with prioritization, retries, and real-time status tracking across multiple machines and bots.
- Built a **Machine Allocation and Robot Management module**, enabling dynamic assignment of RPA microservices (bots) to available machines based on tags, capacity, and workload.
- Designed the **Package Management System**, storing published processes in **Azure Blob Storage** with versioning and metadata management.
- Developed an **Asset Management module** for secure configuration values (API keys, constants, credentials) scoped per tenant and environment.
- Integrated **real-time monitoring dashboards** to display:
  - Online, busy, or **unresponsive bots**, and live execution stats
  - Job failures, processing times, and machine health metrics
- Enabled **real-time event streaming** to UI via **SignalR** for execution logs, bot state changes, and machine health.
- Integrated **multi-channel notification system**:
  - **Twilio SMS** for bot failure alerts and completion updates
  - **Azure Communication Services** for both **SMS and email notifications** on execution status, machine downtime, and queue alerts
- Developed secure, role-based dashboards and APIs to manage tenants, projects, bots, machines, and execution reports.
- Seamlessly integrated with the **Cloud RPA Platform**, allowing direct package deployment, task assignment, and feedback routing.

2020 – Present

RPA Microservices & Execution Platform

Tech Stack: .NET Framework → .NET Core, CoreWF, Electron, WPF, XAML, Windows Services, UIAutomation, Socket.IO, SignalR, Named Pipes

Role: Full Stack Developer / Automation Platform Architect

- Started development in **.NET Framework** with a WPF-based attended bot (Robot) and supporting services. Later modernized the platform by **migrating all services to .NET Core** and rebuilding the Robot frontend as a **cross-platform Electron desktop app**.
- Designed a **microservice-driven RPA execution layer** to support JSON-defined workflows, real-time execution, and orchestration.
- Developed the following core components:
  - **Connection Service (Windows Service):** Handles real-time connections to the **cloud backend** using **Socket.IO** and the **governor backend** using **SignalR**, managing incoming workflow execution requests and dispatch coordination.
  - **Transpiler Service (Windows Service):** Converts incoming **JSON-based workflow definitions** into **executable XAML**, dynamically resolving arguments, variable types, namespaces, and CoreWF activity trees.
  - **Executor App (Console App):** A lightweight runtime invoked by the Robot app to execute workflows using CoreWF. Accepts command-line arguments such as xaml path, workflow ID, and input variables. Outputs logs and results back to the connection service via Named Pipes or IPC.

- **Robot App (Electron):** The user-facing, cross-platform desktop automation agent responsible for receiving cloud instructions, triggering executions via the Executor app, displaying logs, and managing session state.
- **Snippet Tool (WPF):** Utility to capture screen regions and return **base64-encoded image data** for use in automation scenarios like screen comparison and OCR.
- **Inspector Tool (WPF):** Generates automation-ready **UI selectors** using Microsoft UIAutomation APIs for both web and desktop applications.

- Enabled **IPC via Named Pipes** between services and executor for lightweight and secure messaging, along with **real-time messaging** via Socket.IO and SignalR to synchronize with orchestrator and UI clients.
- Fully integrated into a larger **cloud-based RPA platform** for workflow authoring, execution tracking, credential management, and orchestration.

2023 – 2023

Data Validation Platform

Intelligent Document Processing & Review System

Tech Stack: .NET Core, Azure Document Intelligence (Form Recognizer), SQL Server, MailKit, REST APIs

Role: Backend Developer

- Developed backend services for an **RPA-integrated document validation platform** where automation bots upload scanned PDFs for AI-based extraction and human validation.
- Integrated **Azure Document Intelligence** (formerly Form Recognizer) to extract structured data from PDFs and auto-populate database entries.
- Built APIs to support end-to-end document lifecycle:
  - **Initial upload & extraction**
  - **Manual review and data correction by users**
  - **Role-based resubmission** to alternate extraction models (custom Azure models) when needed
- Implemented a **review system** allowing submission of document changes, rejection tracking, and multi-level validation based on user roles.
- Developed **time-tracking logic** to calculate duration taken by users to validate each document — enabling productivity measurement and audit.
- Sent **automated email notifications** using **MailKit**, triggered on document assignment, submission, or review completion.
- Designed the backend with **role-based access**, ensuring only authorized users could view/edit specific document batches and data fields.
- Optimized DB operations for large document sets using indexing and asynchronous batch processing.

08/2023 – 08/2023

Open Source Contribution

socket.io-client-csharp (*Unofficial Fork*)

GitHub: doghappy/ socket.io -client-csharp

Tech Stack: C#, Newtonsoft.Json, System.Text.Json

- Identified and resolved a **serialization bottleneck** in the official C# Socket.IO client library which failed to handle **large payloads** during real-time communication.
- Introduced a **custom serialization interface** to allow seamless switching between System.Text.Json and Newtonsoft.Json, enabling full compatibility with large, nested payloads and flexible settings.
- Refactored message handling logic to support dynamic JSON parsing without breaking existing type-safe contracts.
- Updated and passed **all unit tests** in the original repo to ensure backward compatibility and maintain stability.
- Maintained and used this fork **across the entire RPA platform** in production for robust real-time collaboration and bot communication.
- Although the pull request was not merged, the contribution is actively used in a production-grade RPA cloud infrastructure.

12/2022 – 12/2022

Modernized Fork of ngx-drag-and-drop-lists
(*Private NPM Package*)
Tech Stack: Angular, TypeScript, NPM
Role: Full Stack Developer (Cloud RPA Platform)

- Forked and modernized the outdated ngx-drag-and-drop-lists Angular library, which was no longer compatible with **Angular 13+** and lacked active maintenance.
- Refactored the internal structure to fix **zone.js compatibility issues**, deprecated lifecycle methods, and updated the package to align with Ivy, strict mode, and AoT compilation.
- Created and published a **private internal NPM package** used within the **Cloud RPA Platform Designer**, enabling **workflow steps to be rearranged via drag-and-drop UI**.
- Enhanced component flexibility, styling, and performance for better responsiveness in large-scale collaborative workflows.
- Maintained long-term compatibility and proactively updated the package for use across multiple cloud environments without dependency on third-party updates.

Custom Named Pipe Communication Framework for High-Volume IPC
Tech Stack: .NET Core, C#, System.IO .Pipes, Serialization (JSON, Binary)
Role: Systems Developer / Infrastructure Engineer

- Designed and implemented a **custom Named Pipe wrapper** for both server and client, overcoming native .NET limitations in **handling large data payloads** over pipes.
- Developed internal libraries/packages that handled:
  - **Efficient serialization/deserialization** (supporting both JSON and binary formats)
  - **Buffered streaming** to prevent data loss or corruption during large data transfers
- Built a custom **event-driven architecture** over raw pipe streams, adding events such as:
  - OnClientConnected, OnClientDisconnected, OnDataReceived, OnDataSent
- Used across the **RPA Microservices platform** to enable secure, low-latency inter-process communication between the Robot app, Executor, and Windows Services.
- Ensured **thread-safety, message integrity**, and resource cleanup in high-load and concurrent scenarios.
- Packaged the solution as a reusable module integrated into multiple automation subsystems.

03/2021 – 03/2021

Script Executor Service
Cross-Platform Remote Script Runner for System Automation
Tech Stack: .NET Core, Linux (Ubuntu), Windows, PowerShell, Bash, JWT/Auth
Role: Backend Systems Developer

- Built a secure, cross-platform **script execution service** in .NET Core capable of running **PowerShell (Windows)** and **Bash (Linux)** scripts remotely.
- Used for automation of **privileged system tasks** such as **user creation**, service management, log collection, system updates, and environment configuration.
- Integrated **secure authentication mechanisms** (JWT/API key-based) to identify and authorize users of the service.
- Enabled all command-line-level operations accessible to a system administrator while maintaining **audit logging**, **role-based access**, and **command traceability**.
- Designed to run on both **Ubuntu servers** and **Windows environments**, facilitating consistent automation workflows across platforms.
- Returned structured output, status codes, and execution logs via RESTful APIs, enabling easy integration with other services and dashboards.
- Used in conjunction with the **RPA platform** to perform headless system tasks as part of bot deployment, remote diagnostics, and environment provisioning.

2020 – 2020

Innomeet
Web-Based Group & 1-on-1 Video Calling UI
Tech Stack: Angular, WebRTC, Socket.IO
Role: Frontend Developer

- Developed the UI for **Innomeet**, a web-based video calling app designed to support **1-on-1 and group calls** during the COVID-19 lockdown period.
- Implemented **real-time communication features** using WebRTC and Socket.IO, including call join/leave, mute/unmute, and dynamic participant layout.
- Focused on clean, responsive UI design for both mobile and desktop browsers.
- Although the project was ultimately scrapped, it contributed valuable real-time collaboration experience later applied in production RPA systems.

2017 – 2018

Swing
Cab Aggregation Platform
Tech Stack: Python (Flask), REST APIs, Paytm Payments, Uber, Ola, Meru, Jugnoo
Role: Backend Developer

- Developed backend services for a **cab price comparison platform** that allowed users to view, compare, and book rides across multiple providers based on location, cab type, and real-time availability.
- Integrated major cab provider APIs such as **Uber, Ola, Meru** and **Jugnoo** to fetch dynamic pricing, ETA, and cab availability.
- Implemented **Paytm Payment Gateway** integration to support wallet top-ups, in-app payments, and fare settlements.
- Designed RESTful APIs for mobile/frontend apps with secure token-based authentication and **location-based fare estimation** logic.
- Handled **API rate limits, error handling**, and **fallback logic** to ensure seamless user experience despite provider-level outages or delays.
- Contributed to **modularizing third-party integrations**, making it easy to plug in or remove cab providers based on regional availability.

Skills

AI / LLM
Azure OpenAI (GPT-4o, embeddings), Azure AI Foundry, Google Gemini / ADK, RAG Systems, Multi-Agent Orchestration, MCP (Model Context Protocol), VS Code Copilot (custom MCP servers), Claude Code, GitHub Copilot, Prompt Engineering, Vector Embeddings & Similarity Search

JavaScript / TypeScript
JavaScript, TypeScript, Node.js, Express, TSOA, Inversify IoC, Passport.js

Frontend
Angular 6–20 (PrimeNG, Material, RxJS), React

Python
FastAPI, Azure Functions (Python)

DevOps / Tools
Docker, Azure Pipelines CI/CD, Git, Azure CLI, Puppeteer, Swagger / OpenAPI, Mocha / Chai

Azure
Azure AI Services, Azure AD / MSAL / Entra ID, Azure Cosmos DB / MongoDB vCore, Azure Functions, Azure Container Apps, Azure Application Insights, Azure DevOps / Pipelines, Azure Blob Storage, Azure Document Intelligence, Azure Communication Services, Azure Key Vault, Azure OpenAI Service

C# / .NET
.NET Core (Web API, Windows Services, Kestrel), .NET Framework (Windows Services, WWF / CoreWF activities, WPF), Named Pipes IPC, Electron desktop integration, Entity Framework, Entity Framework Core, C#

Real-Time & Messaging
Socket.IO, SignalR, WebSockets, Named Pipes

Databases
MongoDB / CosmosDB (vector search & Atlas Search), Mongoose ODM, MSSQL, Entity Framework, PostgreSQL, Prisma ORM, Redis (streams, queues, caching)