Rahul Mitra *Senior Software Engineer*

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 with over 6 years of experience, including 4+ years working fully remote in 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 creating RESTful APIs and intuitive UI/UX. Extensive experience leveraging Microsoft Azure for cloud-based solutions, including communication services, event-driven architectures, and scalable deployments. 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

07/2021 – present
Remote from Durg, Chhattisgarh

Senior Software Engineer
InnoWise LLC

- Developed and maintained multiple microservices for a cloud-based Robotic Process Automation (RPA) platform using Angular, Node.js, and .NET Core, ensuring scalability and maintainability.
- Designed and implemented responsive user interfaces with Angular and TypeScript, enhancing user experience.
- Managed and deployed microservices on Azure Cloud Services, optimizing the RPA platform's performance.
- Automated Azure VM instance creation using Azure SDK for dynamic resource allocation in projects.
- Created Windows Services installers using Visual Studio Installer Projects and InnoSetup for seamless deployments.
- Designed and integrated dashboards for document understanding and data extraction with Azure Cognitive Services for OCR and text analysis.
- Implemented real-time features using Socket.IO to enable collaborative workflow design and execution.
- Utilized Azure Communication Services for real-time communication and collaboration functionalities.
- Contributed to new feature development and bug fixes across modules, maintaining high code quality and adhering to agile methodologies.
- Mentored new employees and conducted technical interviews.
- **Achievements:**
- Received Employee of the Month award three times.
- Received client recognition for developing an RPA governance system using Angular and Web API

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

Software Engineer
InnoWise LLC

- Developed Node.js Express TypeScript servers with real-time capabilities.
- Engineered the user interface for "InnoMeet," a Group Meeting application, utilizing Angular.
- Built real-time applications using Angular, Socket.IO, SignalR, and Web API.
- Developed Windows services with and without API integrations.
- Established an IPC framework using Named Pipes and TCP sockets in .NET and .NET Core.
- Created Ubuntu services based on .NET Core 5.0+ with RESTful APIs.
- Conducted technical interviews and provided mentorship to new employees.
- **Achievements:**
- Recognized for enhancing "InnoBots," a cloud automation platform, by integrating Angular with backend technologies like Web API, Mongoose ODM, MongoDB, and Node (for Socket.IO server), serving as the sole developer on the project.
- Received the Star Performer Award for 2020-2021 for contributions across multiple projects.

03/2019 – 06/2020
Pune, M.H

Associate Software Engineer
InnoWise LLC

- Developed user interfaces for a cloud-based RPA system.
- Created a Node.js backend server using Socket.IO for real-time communication, including sending notifications to online users.
- **Achievements:**
- Received the Rising Star Award for 2019-2020 for developing "InnoBots," a Cloud-Based RPA platform in Angular.

Projects

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

Angular
Worked on Angular 6 till 20

Javascript

Dot Net Framework (C#)
Created Windows Services, WWF activities, WPF

MSSQL
Used MSSQL on several projects

PostgreSQL

React
Collaborated on some projects for component data design

Socket.IO
Contributed to open source project for C# Socket.IO client, used Socket.IO on several projects for Real-Time communication

NodeJs

Dot Net Core (Using C#)
Worked on Web API, Windows Services, Kestral,

Typescript

MongoDB
Worked with MongoDB on several projects

Mongoose ODM

Entity Framework & Entity Framework Core ORM

Web API
Using Dot Net (Core and Framework) or ExpressJS

SignalR