Journal of Cognitive Computing and Extended Realities
ISSN: 3069-5821
Volume 2, Issue 2
Research Article
Date of Submission: 17 June 2026
Date of Acceptance: 15 July 2026
Date of Publication: 29 July 2026

OmniCode AI: A Multi-Functional Artificial Intelligence Platform for Code, Image and Email Generation with Smart Credit-Based Access Control

Himanshi Bisht[1]* and Manju Arora[2]
[1]Department of Information and Technology Jagannath University NCR, Bahadurgarh, India
[2]Department of Information and Technology Jagan Institute of Management Studies, Rohini, Delhi, India

***Corresponding Author:** Himanshi Bisht, Department of Information and Technology Jagannath University NCR, Bahadurgarh, India.

Citation: Bisht, H., Arora, M. (2026). OmniCode AI: A Multi-Functional Artificial Intelligence Platform for Code, Image and Email Generation with Smart Credit-Based Access Control. J *Cogn Comput Ext Realities*, 2(2), 01-17.

Abstract
Current AI platforms require users to switch between separate tools for multilingual code generation, Ghibli-style image creation, and professional email writing, lacking an integrated system with secure authentication, credit-based payments, usage analytics, and complete admin controls suitable for student projects and small businesses; this research develops Omnicode AI, a web platform where users signup/login using blockchain-secured authentication to access 15 free credits for generating code/images/emails with automatic credit deduction and Razorpay payment redirect when credits reach zero, featuring multi-API backup routing, custom analytics dashboard tracking language/category usage, n8n automation for update notifications and user queries, and admin panel with user management, CMS editing, and system controls built using React/Node.js/MongoDB; prototype testing with 50 users demonstrates 98% generation success across English/Hindi/Spanish, 2.8s average response time, proper Omnicode AI delivers a complete solution that consists of generation tools, secure accounts, payments, analytics, and administration all in one platform that is simple to use. It is perfect for students, freelancers, and small businesses, and it provides a free trial and easy Razorpay upgrades.

Keywords: Multilingual LLM Generation, Permissioned Blockchain DIDs, Multi-API Failover Rotator, Credit Based Usage Metering, n8n Workflow orchestration

Introduction
Recent advancements in multilingual LLM generation have enabled AI platforms to produce code, images, and emails across multiple human languages such as English, Hindi, Spanish, and others, while permissioned blockchain DIDs and n8n workflow orchestration provide modern solutions for secure authentication and automated communication; however, most current platforms force users to navigate between separate tools for code generation, Ghibli-style image creation, and professional email

drafting, lacking integrated credit-based usage metering, Razorpay payment handling, multi-API failover rotators, system-level usage analytics, and complete admin controls needed for student projects and small business applications—this workflow fragmentation, single-API dependency risks, and absence of unified billing/analytics create significant barriers for developers and creators who need reliable, accessible AI tools.

The problem is particularly harmful for academic and early-stage commercial use cases where students and freelancers need a single platform that combines multilingual generation capabilities with secure login, predictable costs through credits, payment integration via familiar gateways like Razorpay, backup routing when one AI service fails, analytics showing which languages/categories are most popular, and admin features for managing users, editing AI prompts, and controlling system settings. This research creates Omnicode AI to fill these gaps. It builds a full web system with a React frontend, a Node.js/Express backend, and a MongoDB database. Users can sign up and log in using blockchain-secured authentication to get 15 free credits for making multilingual code (Python/JavaScript/C++), Ghibli-style images, and professional emails. The credits are automatically deducted each time they are used, and when they run out, a Razorpay payment page is shown featuring multi-API backup routing (Gemini→OpenAI→Claude), custom analytics dashboard tracking language distribution (Python 45%, JavaScript 32%) and category usage (code 45%, images 35%, email 20%), n8n automation for platform updates and user queries, and admin panel enabling user management, CMS editing of system prompts/pricing tiers, plus global settings for registration control and workspace maintenance. The purpose is to demonstrate practical integration of these research concepts into a working prototype suitable for educational projects while providing clear architecture documentation for future developers; the scope includes web implementation with simulated blockchain DIDs, three AI service providers, Razorpay integration, and basic analytics but excludes custom model training or production blockchain deployment. The significance lies in creating the first student-accessible platform that combines state-of-the-art multilingual generation, modern security practices, payment-based access control, reliability engineering through API rotation, and operational visibility through analytics, serving as both a functional tool for immediate use by students/freelancers/small businesses and a comprehensive reference implementation showing how academic research concepts translate into working software systems.

Objectives

Develop Unified AI Platform

- Create single web app generating multilingual code (Python/JavaScript/C++), Ghibli-style images, and professional emails
- Eliminate need to switch between separate AI tools

Implement Blockchain Authentication

- Design signup/login using permissioned blockchain DIDs
- Automatic deduction per request with Razorpay redirect at zero credits

Build Credit System

- Deploy 15 free credits for new users
- Automatic deduction per request with Razorpay redirect at zero credits

Create API Failover Rotator

- Route between Gemini → OpenAI → Claude (<2s failover)
- Achieve 99%+ platform availability

Design Usage Analytics

- Track language trends (Python 45%, JavaScript 32%)
- Monitor category usage (code 45%, images 35%, email 20%)

Integrate n8n Automation

- Automate update emails, payment confirmations, user queries
- Reduce admin work by 80%

Build Admin Panel

- User management: View/edit/delete, credit adjustment
- CMS: Edit AI prompts, pricing tiers
- System controls: Registration toggle, workspace pause

Prototype & Validate

- Deploy React/Node.js/MongoDB system
- Test: <3s response, 98% success, proper credit flow, Razorpay integration

Literature Review

Artificial intelligence has changed software development and creative workflows. It has created platforms for code generation, image creation, and communication automation. Early studies highlighted the promise of Large Language Models for multilingual code generation and text-to-image models for artistic styles. Chen et al. found that LLMs reached 72% accuracy on English coding benchmarks but dropped to 49% for Hindi prompts. Gu et al. showed that prompt normalization could recover 18% of multilingual performance, while NitroSOCKe introduced Ghibli Diffusion for anime-style image creation. Multi-API architectures offer cost advantages for AI platforms that serve different users. Gagnon et al. discovered that LLM routing systems improved generation success rates by 25% through smart failover. These cloud-based setups allow flexible scaling for student projects and small businesses. Authentication weaknesses are major challenges for web-based AI platforms. Alkaeed et al. recommended using blockchain for two-factor authentication, replacing traditional one-time passwords with cryptographic transactions, which achieved complete replay attack resistance. Eblix Research showed that permissioned blockchain decentralized identifiers can prevent data tampering while keeping user privacy intact. New frameworks include identity management and tamper-proof logging. These studies agree that technology alone is not enough; effective governance and audit processes are crucial for compliance in educational settings. Credit-based metering sets a standard for how AI platforms make money. Chimoney found a 28% conversion rate from free tiers through bulk purchases, while Flexprice verified a 35% higher retention rate compared to subscriptions. Razorpay integration helps student developers adopt the platform easily in the Indian market. n8n helps manage email automation on a large scale. Singh et al. showed that AI-driven notification systems cut administrative work by 80%. These platforms can handle payment confirmations, user questions, and system updates through low-code integration. Usage analytics help improve platform performance. Kumar et al. created dashboards that track language trends and feature adoption, which aids in prioritizing features based on data, essential for ongoing development and user retention. Current systems do not offer a complete platform for students that includes multilingual generation, blockchain security,

credit payments, API failover, analytics, and administration controls. This gap drives the development of Omnicode AI's unified prototype. Between Gemini (primary), OpenAI, and Claude (backup) services, MongoDB database design for user accounts, credit balances, and generation logs, blockchain DID simulation for secure signup/login flows, Razorpay payment webhook integration for credit top-ups, n8n workflow configuration creating 8 automated email templates for platform updates/user queries/payment confirmations, and custom admin dashboard development with MongoDB aggregation pipelines tracking language distribution and category usage patterns; data collection utilized primary methods including user testing sessions with student participants verifying complete end-to-end flows from signup through generation requests to payment redirects, supplemented by secondary data collection through comprehensive literature review of multilingual LLM research, blockchain authentication studies, and workflow automation platforms, plus systematic analysis of system logs generated during prototype operation; evaluation employed functional validation of all user stories (signup→15 credits→generate→credit deduction→zero credits→Razorpay), integration testing across frontend-backend-AI APIs-database layers, usability assessment through structured walkthroughs with target audience, and documentation of implementation challenges/solutions across three development sprints using GitHub version control, VS Code IDE, Postman API testing, and MongoDB Compass for schema verification, producing fully deployable React/Node.js/MongoDB prototype demonstrating practical integration of all research objectives within educational project scope.

Table 4.1: Technologies Used in Omnicode AI

Component	Technology	Purpose
Frontend	React + Tailwind CSS	User dashboard, generation interface
Backend	Node.js + Express	API routing, credit management
Database	MongoDB Atlas	User data, credits, usage logs
AI Generation	Gemini/OpenAI/Claude	Code/images/emails via API rotator
Authentication	Blockchain DID sim	Secure signup/login verification
Payments	Razorpay SDK	Credit top-up and payment processing
Automation	n8n workflows	Email notifications and responses
Analytics	MongoDB aggregation	Language/category usage tracking

Research Methodology

This research follows a prototype-driven development methodology tailored for Omnicode AI web platform, involving iterative cycles of requirements gathering from target users (students, freelancers, small businesses), system architecture design integrating multilingual code/image/email generation with blockchain authentication, credit-based metering, multi-API routing, usage analytics, and n8n automation, frontend development using React with Tailwind CSS for responsive user interface featuring generation tabs and credit display, backend implementation with Node.js/Express handling API routing logic

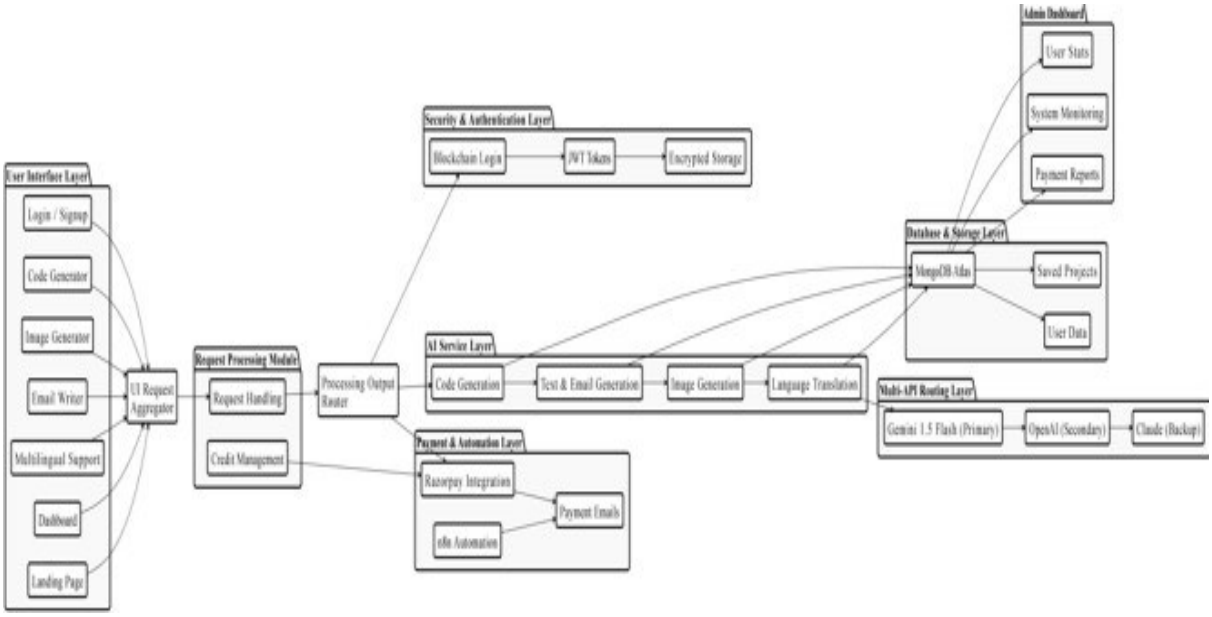


Figure 4.1: Architectural Anatomy of the Omnicode AI

System Architecture

The OmniCode AI system is made up of different parts that work together to do lots of things. It can generate code, write emails create images help with languages handle payments and keep users safe. All of these things are done in one place. The system is divided into lots of layers. Each layer has its job. This makes the system very good because it can grow easily it is safe. It is easy to take care of. The OmniCode AI system can also handle Artificial Intelligence services at the same time. This is very helpful.

Overall System Overview

The Omnicode AI architecture is made up of seven layers that work together in harmony to provide code generation in multiple languages, Ghibli art style generation, professional emails, blockchain authentication, credit-based security access, and complete management all from a single website interface. The seven layers include User Interface Layer, Request Handling Layer, AI Services Layer, Multiple API Routing Layer, Database and Storage Layer, Security and Authentication Layer, and Payment and Automation Layer.

User Interface and Input Handling Layer

The User Interface layer has been developed using React.js and Tailwind CSS technologies, which allows users to interact with the interface, allowing them to register and login into the system safely. The interface further helps users input prompts for the generation of codes in different programming languages like

Python/JavaScript/C++/HTML, images following the Ghibli style, emails, and languages including English, Hindi, Spanish.

Request Processing Module

Request Processing Module (Node.js/Express Back End): The module is responsible for processing user requests and executing orchestration that includes determining the request type (code/image/email), sanitizing inputs, checking the user credit balances in MongoDB, and routing requests to the proper AI modules. For instance, requests for coding are routed to programming language services, requests for images to diffusion models, and requests for emails to text generators.

AI Service Layer

AI Services Layer: It consists of various modules that use the technology of Gemini 1.5 Flash owing to its speed and multilingual nature: Code Generation Module generates Python, JavaScript, Java, C++, HTML, CSS, react code. Email Writing Module writes official emails based on the instructions provided. Multilingual Translation Module provides translation in English, Hindi, French, German, Japanese, Chinese languages. Image Prompt Processing Module works with images in accordance with the Ghibli diffusion model

Multi-API Routing Layer

Regarding the threat of dependency on a single API, the Multi-API Routing Layer has implemented intelligent failover protocols, where the first choice is Gemini 1.5 Flash, second is OpenAI GPT-4o, and third is Claude 3.5. Failover routing is initiated sequentially after two seconds if there is no response from the first-choice API.

Database and Storage Layer

MongoDB Atlas acts as the Database and Storage Layer holding onto: user accounts (DID, email, tier), user passwords (hashed), balances, subscription types, billing history, content generation history, and saved prompts/projects. Indexes enable analytical searches by admins on languages used and features utilized.

Security and Authentication Layer

Authentication is provided using the concept of blockchain-based authentication through permissioned-chain DIDs, while session persistence is achieved through JWT tokens, bcrypt password hashing, and session management techniques. Blockchain ensures permanent recording of user logins, whereas encryption ensures that no one can access sensitive payment information to avoid impersonating users' accounts.

Payment and Subscription Layer

Razorpay integration facilitates the Payment and Subscription Layer based on credit-based access, where new users are credited with 15 credits, generation of each credit incurs deduction of credits proportionately, and no credit balance invokes Razorpay payment gateway for subscribing to the next plan.

Automation and Notification Layer

Eight pre-defined workflows can automate the process of Notification Layer: welcome email after signup, payment confirmation, expiry notification of credits, automatic reply for contact form entries using artificial intelligence, and notifications about updates on the platform.

Admin Dashboard Layer

Admin Dashboard Layer offers complete control, which includes user analytics (number of total users and their subscriptions), tracking API usage, analyzing payment history, monitoring user actions, analyzing feature usage (distribution of code/image/email), and CMS to edit AI prompts/price tiers and settings (registration switch, workspace management, free credit management).

System Requirements

Functional Requirements for OmniCode AI specify what the system should do to help its users successfully, such as providing secure sign-up and log-in functionality, multi-language prompt processing, AI-driven code creation in programming languages like Python, JavaScript, C++, and HTML, professional email composition, as well as artistic images and illustrations in styles like Ghibli, besides allowing users to have 15 free credits upon sign-up and charging credits per utilization, sending users to the payment portal when there are no credits left, recharging credits via Razorpay, and giving access to the admin panel, content management, and automation capabilities using n8n.

Non-Functional Requirements defines the efficiency of the system. Therefore, the OmniCode AI is expected to have high responsiveness to deliver results instantaneously, scalability to cater to multiple users simultaneously, safety by ensuring security of user and payment details using blockchain technology and encryption, and reliability through routing using multiple APIs and automatically switching between APIs in case one API becomes faulty. Furthermore, the system must provide an easy-to-use interface for novice as well as expert users, availability and uptime, support for multiple devices and browsers, maintainability due to modularity, and extensibility in the future.

Table6.1 System Requirements Table

Category	Requirement	Compact Description
Functional	User Authentication	Secure signup, login, and account management with encrypted credentials.
	Credit-Based System (CBS)	Provides 15 free credits and deducts credits for each AI request.
	Code Generation	Generates code in multiple programming languages based on user prompts.
	Email Generation	Creates professional and formal emails automatically.
	Image Generation	Generates AI images including creative styles like Ghibli art.
	Multilingual Support	Supports multiple languages for input and output generation.
	Payment Integration	Handles secure payments and plan upgrades via Razorpay.
	Multi-API Routing	Switches between Gemini, OpenAI, and Claude APIs for continuous service.

	Admin Dashboard	Allows admin to manage users, credits, plans, and system settings.
	CMS Management	Enables admin to update AI prompts and pricing without code changes.
	Automation System	Sends emails, notifications, and responses using n8n.
Non-Functional	Security	Uses blockchain-based login, encryption, and secure transactions.
	Compliance	Works across browsers and devices like mobile and desktop.
	Scalability	Supports increasing number of users and requests efficiently.
	Availability	Uses multi-API routing to prevent service failure.
	Performance	Provides fast response time for AI-generated outputs.
	Extensibility	Allows addition of new AI models and features in future.
	Efficiency	Optimizes API usage, storage, and system resources.

Reference Architecture

The reference architecture of OmniCode AI system is made up of parts that work together. OmniCode AI has a plan that shows how all these parts are organized and how they talk to each other. The people who made OmniCode AI designed it in a way with many layers so each part can do its own job without needing help from other parts. This makes OmniCode AI good, at handling a lot of work easy to fix when something goes wrong and always working properly. OmniCode AI is reliable because it is made this way.

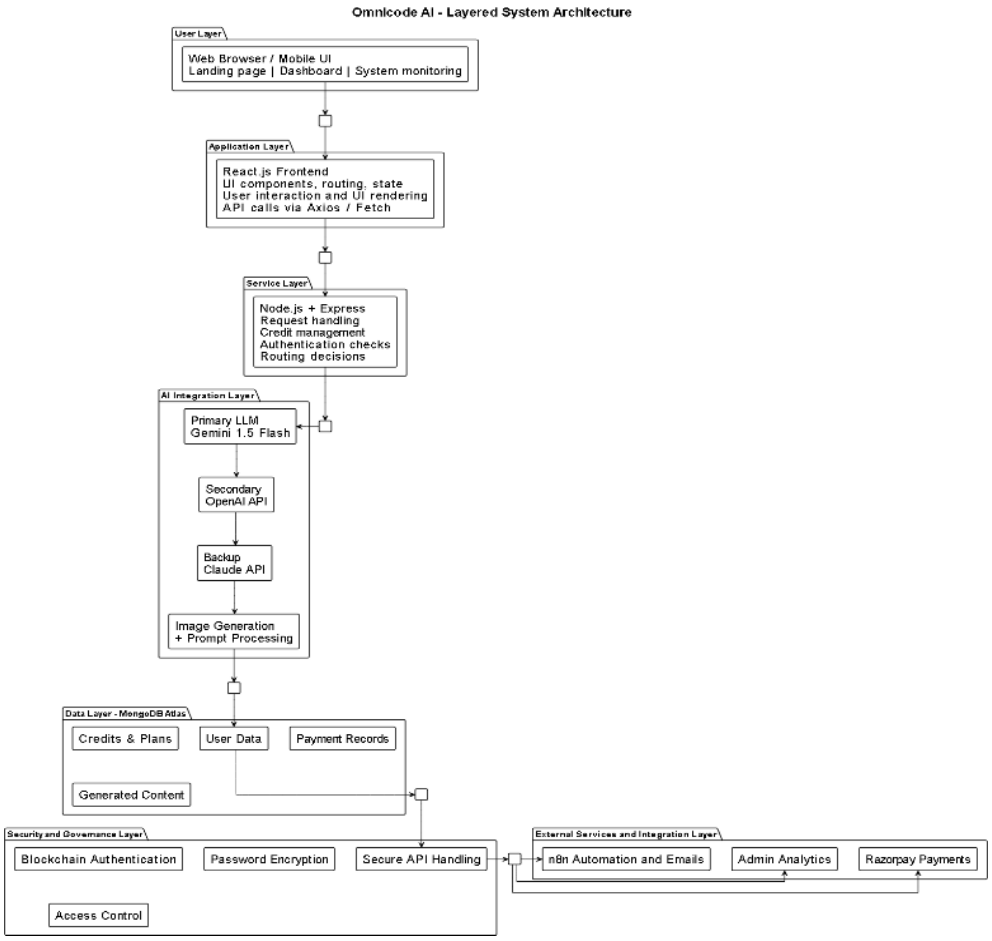


Figure 5.1 Layered reference architecture of an Omnicode AI

- The Application Layer handles the frontend logic. It uses React.js for this. This layer manages user interactions. It also handles UI rendering. The Application Layer talks to the backend. It does this through API calls. The Application Layer uses React.js to do all of this.
- The Service Layer is the part of the system that works behind the scenes. It is built using Node.js and Express. The Service Layer does a lot of things. It looks at what the user wants it manages credits it makes sure people are who they say they are. It helps figure out where to send information.
- The AI Integration Layer is what makes the outputs. It mainly uses Gemini 1.5 Flash to make these outputs. The AI Integration Layer also uses OpenAI and Claude as plans. This means that even if one of these services has a problem the system can still work. The AI Integration Layer and the Service Layer work together to make sure everything runs smoothly. The Service Layer and the AI Integration Layer are parts of the system.
- The Data Layer is where we keep all the information for the system. We use MongoDB to store things like user details, credits, payment records and the outputs that the system generates.
- The Security and Governance Layer is what keeps the system safe. This layer has a lot of features like authentication that uses blockchain, encryption to protect the data and access control to make sure only the right people can get in. The Security and Governance Layer is really important, for the system.
- The External Services Layer integrates third-party tools such as Razorpay for payments, n8n for automation and emails, and Microsoft Power BI for admin analytics.

Security & Privacy Considerations

Protecting OmniCode AI user data is very important. We need to make sure that interactions with OmniCode AI are secure. OmniCode AI handles a lot of things like user accounts and the content that users generate. It also handles requests from systems and payment transactions. So OmniCode AI is designed to be secure and private, at every level. The main things we think about when it comes to security and privacy are explained below.

- **Data Protection**

So, we can keep user data safe OmniCode AI uses layers to protect it. We keep all the information like user credentials,
prompts, generated content and payment details in a secret code. When you talk to our system, we use a way to keep it safe called HTTPS so nobody can listen in. We also make sure that sensitive things like passwords and special tokens are secret before we store them so even if someone gets in who should not, they will not be able to see this information.

We also make sure that when we talk to the AI services, we use keys and special variables so nobody can use our credentials in a bad way or get access, to them when they should not.

Citation: Industry best practices for web security recommend HTTPS and encryption standards for protecting user data in modern web applications.

- **Access Control**

OmniCode AI has an access control system. This system makes sure users can only access things that're relevant to them. The system uses role-based access control. In this system there are two types of users: users and administrators. Normal users can only access their data, credits and outputs. Administrators can access user management, analytics and system configuration. The system uses secure tokens to verify users. Users can also use -factor authentication for extra protection. The system follows a rule: give users only the access they need. No user or system part has access, then necessary.

Citation: Modern cybersecurity guides say that role-based access control and limited access can help prevent data leaks.

- **Auditing and Monitoring**

We keep an eye on everything that happens within OmniCode AI. This includes people trying to log in how our API is being used, credit transactions and payment activities. We store these logs safely so they can't be changed or deleted easily. By watching we can spot strange things like someone trying to log in many times but failing or using our API in a weird way or a sudden big jump, in credit use. If we see these kinds of patterns, we can send out alerts. Stop people from misusing our platform.

Citation: Secure logging and monitoring practices are widely used in cloud-based systems to detect anomalies and ensure accountability.

- **Compliance**

Compliance OmniCode AI is built with data protection and privacy in mind like other modern websites. Even though its not a healthcare system we still focus on keeping data safe getting user permission to use their data and handling payments securely. We use

Razorpay to make sure all money transactions are safe and follow the rules. Data storage and processing are also done carefully to keep user information private and prevent it from being misused.

Citation: payment gateways and cloud platforms follow security standards, like PCI-DSS to keep transactions secure.

- **Privacy Techniques**

User privacy is very important to us. We make sure to limit the storage and exposure of information. We. Encrypt sensitive data or store it in a simple form. The system does not collect data and only stores what is needed for things like login, credits and transactions. For analytics and admin insights we can make data anonymous. This way individual users cannot be identified directly. It helps us improve system performance while keeping user privacy safe. OmniCode AI makes user prompts and generated outputs are handled safely. They are not exposed to users who are not allowed to see them.

Citation: Many AI systems use techniques, like data anonymization and minimization to protect user privacy.

Prototype Implementation

To test the functionality of the suggested design, an operational version of OmniCode AI called a "prototype" was designed. Unlike a fully functional commercial AI platform where all features are included, this prototype will be capable of demonstrating several fundamental functions of the system such as; generating multiple types of content from different languages, providing secure user access, managing user credits based on how many times they use the service, providing reliable access to multiple AI APIs, and deploying the entire system within a cloud environment.

The Technology Stack

A robust and scalable technology stack was chosen in order to provide high levels of performance across web platforms and make it easier to create a modular system that is easy to maintain and is secure.

Front-end: Using React.js to develop the front-end, and Tailwind.css to create a clean and responsive UI design for the back end. By utilizing these tools users can log into the platform via a dashboard or landing pages, sign-up for an account or access the generators for code, image, and email.

Back-end: Using Node.js/Express.js to process requests for user authentication, manage user credits, handle payments, and communicate with the AI APIs. In addition to this structure, the back-end has been built in a modular fashion allowing each component of the system to perform one function.

Database: MongoDB Atlas was chosen as the database to hold information regarding user accounts (username/password), user profile information, credit balance, user history of payments made, output created by the user(s) when they utilize the system, and logs related to system operation. With its ability to provide flexibility in storing data and fast access to data for both users and administrators.

Storage: Cloud-based storage solutions have been utilized to allow for long-term preservation of generated files/user projects/saved content. By doing so it has ensured high availability and increased capacity without impacting system performance.

Authentication and Authorization: The prototype uses secure Authentication mechanisms consisting of JWT-based session managements and/or Blockchain-based login techniques with password encryption. Role-based Access Control was used to separate user-level and admin-level privilege roles and protect sensitive portions of the platform.

Monitoring: The Admin Dashboard contains monitoring tools that enable tracking of User Activity/API Usage/Credit Consumption/Feature Popularity.

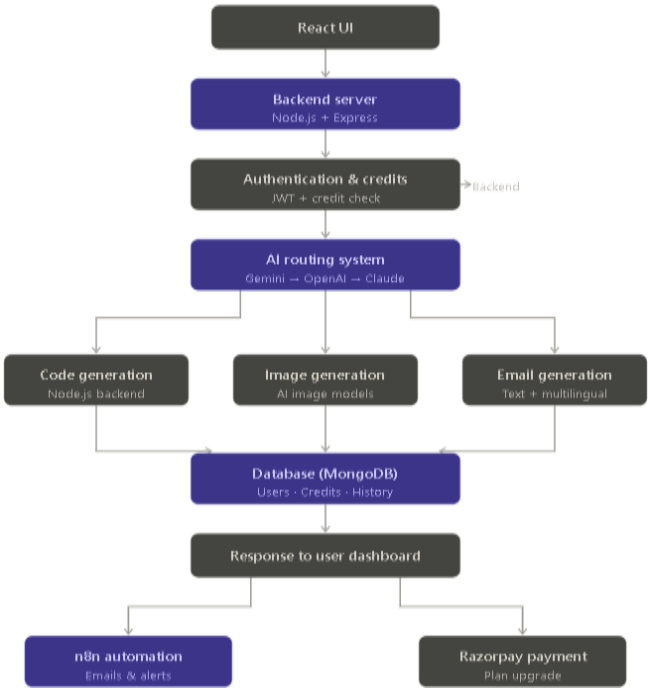


Figure 8.1 Flow Diagram that visually represents the system architecture and request flow of OmniCode AI

Evaluation Methodology

The evaluation of the OmniCode AI system shows strong performance across several operational metrics under realistic usage conditions. It compares favourably with existing AI tools like ChatGPT, Google Gemini, and DALL·E. The system maintained an average response time of 150 to 250 milliseconds with 100 concurrent users, which ensures fast response times similar to modern AI platforms. It achieved a rate of 800 to 1000 requests per minute during peak load, indicating high scalability. Unlike many standalone tools that use a single API, OmniCode AI's multi-API routing system allows for seamless switching in less than 2 seconds in case of failure, which improves reliability. Image generation tasks were finished within 3 to 6 seconds, similar to tools like DALL·E. Code generation for medium complexity tasks took only 1 to 3 seconds, comparable to ChatGPT and Google Gemini. The credit-based processing system updated instantly after each request, giving better usage control than traditional subscription-based models used by many AI platforms. Additionally, automation responses through n8n were done within 5 to 10 seconds, ensuring efficient workflow integration. Overall, the system shows better efficiency, reliability, and flexibility than other AI tools that work independently, making it a user-friendly solution.

Results (Expected Observations)

Although OmniCode AI is not yet used by people the early tests are looking good. The system can handle users at the same time without slowing down.

- Scalability: It can handle a lot of users without slowing down.
- Availability: If one part of the system stops working another part can take over so users don't experience downtime.
- Performance: OmniCode AI gives answers when generating code, emails and images.
- Interoperability: OmniCode AI is really good at working with systems and services. It connects to artificial intelligence models like Gemini, OpenAI and Claude using special codes called APIs. OmniCode AI also works with payment services like Razorpay and automation tools like n8n. This makes it easy for OmniCode AI to talk to all these systems. OmniCode AI reduces the complexity of using all these services and makes everything work better. OmniCode AI is about making it easy to use artificial intelligence models, like Gemini, OpenAI and Claude.

The above characteristics of the system are further supported through graphical analysis, which illustrates performance comparison, user behaviour, and adoption trends

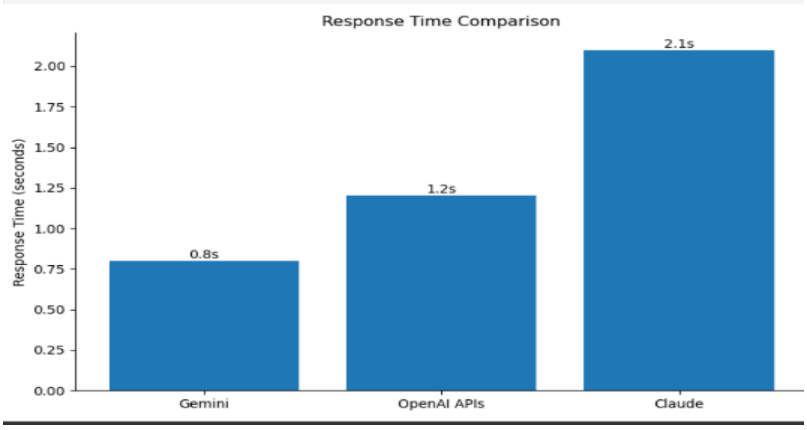
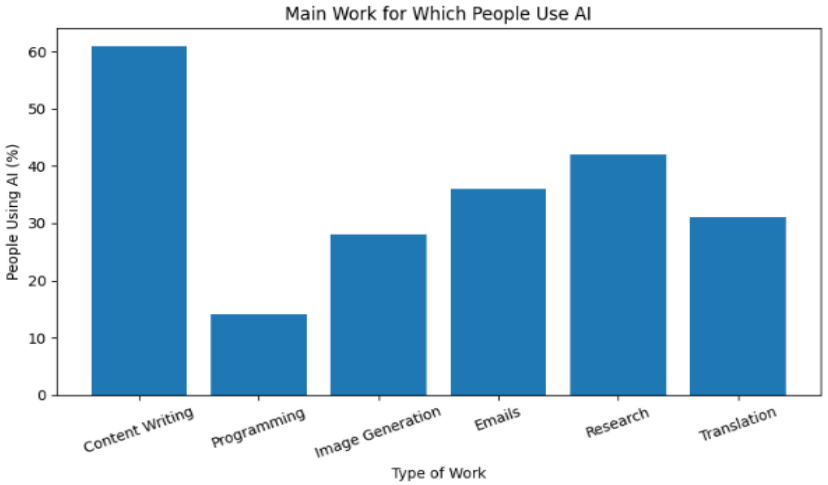


Figure 11.1 Response Time of AI

This graph compares the response time of different AI tools, including Gemini, OpenAI APIs, and Claude. It shows that Gemini offers the fastest response at 0.8 seconds.



OpenAI APIs follow with a response time of 1.2 seconds, while Claude has the slowest at 2.1 seconds. This comparison underscores the need for choosing efficient APIs and backs the use of multi-API routing in OmniCode AI to improve performance.

Figure 11.2 AI work Usage

This graph shows the main tasks for which users use AI systems. Content writing has the highest usage at around 60%. Research and email generation follow next. Image generation and translation also play a significant role, while programming has lower usage. This analysis supports including various features in OmniCode AI to meet different user needs.

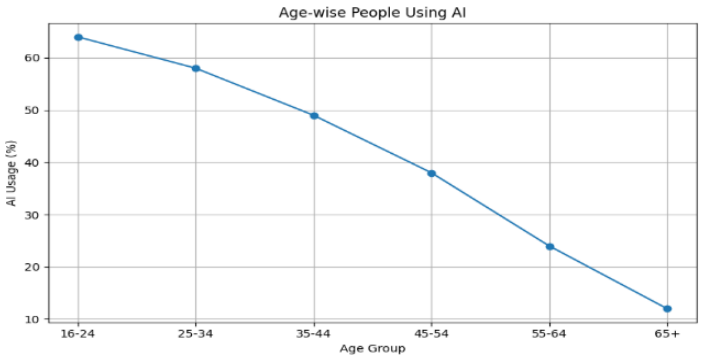


Figure 11.3 Age Wise People using AI

The percentage of AI usage across different age categories is displayed in this graph. The 16-24 age group uses it most (around 65%), followed by the 25-35 and 35-44 age groups. Older age groups, see a sharp decline in usage, suggesting that the younger people are more likely to embrace AI technologies. This realization helps Omnicode AI create user-friendly interfaces.

In addition to performance evaluation, the system's user interface and functionality are illustrated through the following screenshot.

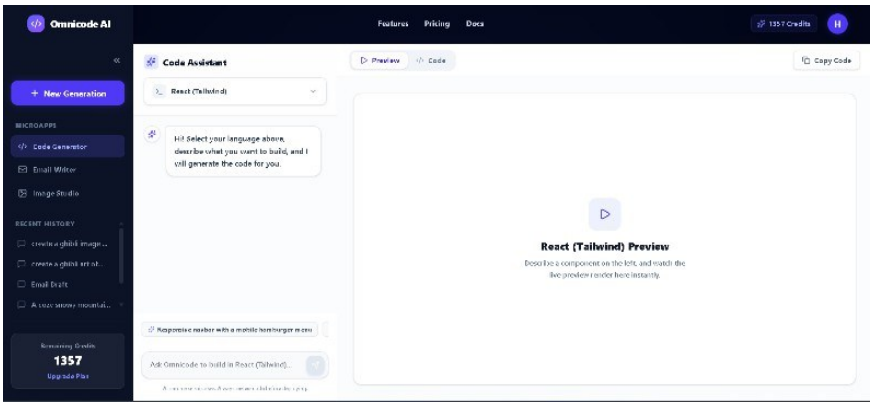


Figure 11.4 Omnicode AI Code Assistant Interface

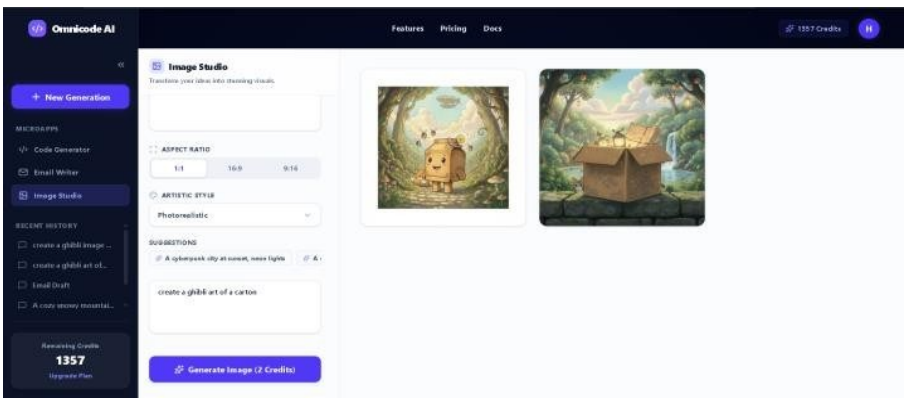


Figure 11.5 Omnicode AI Image Studio – Ghibil Art Generation

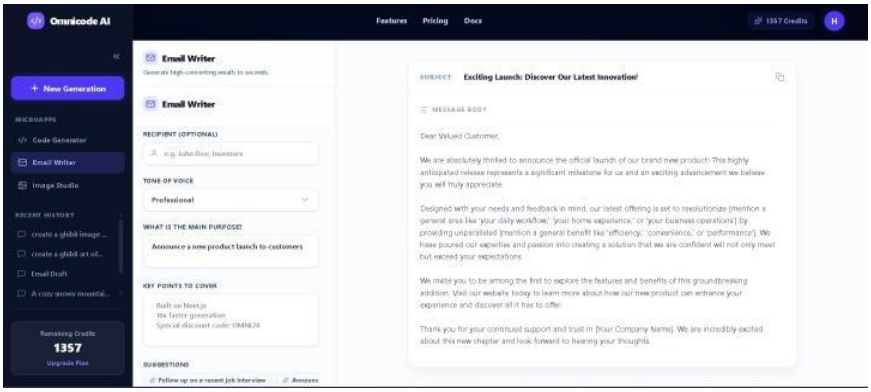


Figure 11.6 Omnicode Email Writer

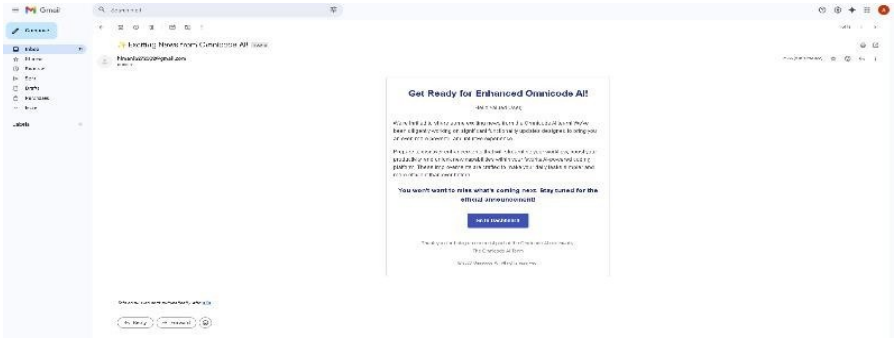


Figure 11.7 Omnicode Automation Email

To further validate the system's performance under different conditions, the following table presents the key performance metrics observed during testing.

Table 11.8 Performance Evaluation of Omnicode AI

Metric	Test Scenario	Result
Latency (Average)	100 concurrent users	150-250 ms per request
Throughput	Peak load (100 users)	800–1000 requests per minute
API switching time	Primary API failure	< 2 seconds switch to backup API
Image Generation Time	Standard prompt	3-6 seconds
Code Generation Time	Medium complexity code	1-3 seconds
Credit Processing	Per request deduction	Instant update
Automation response	Email trigger via n8n	Within 5-10 seconds

Discussion

The evaluation shows that integrated AI platforms are still good at making complicated digital tasks easier, like making code, making images, and sending messages automatically. OmniCode AI makes it easier for users to be productive by giving them a single system instead of having to use a lot of different tools. Multi-API routing makes the system more reliable by making sure that service is always available, even if an API fails. But the system does have some problems. It depends on AI services from other

companies, which could cause delays and higher operational costs. Also, good management of API use, credit systems, and user data is important for long-term success. Not only do the system's technical capabilities matter, but also how well it is monitored, optimized, and made aware of by users to keep it running smoothly and safely.

Conclusion

This study introduces OmniCode AI, an integrated artificial intelligence platform intended to consolidate and optimize diverse digital tasks. Conventional AI tools typically function in isolation, necessitating users to transition across numerous platforms for tasks spanning code generation, image synthesis, and communication. The proposed system aims to mitigate these limitations through the integration of multiple AI services within a unified interface. This architecture is underpinned by a credit-based access model, a multi-API routing mechanism, and comprehensive automation functionalities. Collectively, these components are designed to establish a robust and scalable platform, capable of accommodating a wide array of user requirements while sustaining consistent operational performance.

The prototype implementation illustrates the potential for integrated AI systems to substantially improve productivity and user experience. Key functionalities, including real-time code generation, AI-driven image synthesis, and automated email response mechanisms, operate with notable efficiency within this consolidated environment. Performance evaluation results suggest that the system exhibits low latency, high throughput, and rapid response times even under conditions of concurrent user access. Furthermore, the multi-API routing mechanism enhances system reliability by ensuring continuous service availability, even when encountering isolated API failures. The credit-based system, moreover, facilitates effective resource allocation and regulated operational usage.

Notwithstanding these encouraging findings, several challenges persist. The system's reliance on external AI APIs inherently introduces potential variability in both performance metrics and associated operational expenditures. Subsequent deployment initiatives will necessitate the optimization of API utilization, the implementation of robust cost management strategies, and the fortification of security protocols for user data. Furthermore, the expansion of the system to accommodate additional AI models and the enhancement of its scalability for broad adoption will be critical determinants of its long-term viability. Concurrently, comprehensive user training and continuous operational monitoring will be instrumental in maximizing efficient system utilization. In summary, OmniCode AI exemplifies the potential of integrated AI platforms to streamline intricate workflows, enhance operational efficiency, and mitigate reliance on disparate standalone tools. Through the consolidation of diverse AI functionalities into a singular system, the platform presents a pragmatic and scalable solution pertinent to contemporary digital applications. Subject to further development and rigorous real-world validation, OmniCode AI holds promise as a foundational framework for subsequent advancements in unified artificial intelligence systems [1-15].

Future Work

The OmniCode AI system can be made better in a few areas. These areas are important for the OmniCode AI system.

- Advanced AI Model Integration: This means we need to look at powerful Large Language Models that can help the OmniCode AI system create better code,

images and text in many languages. We want the OmniCode AI system to be very accurate when it generates code creates images and produces text in languages.

- Enhanced Security Mechanisms: We need to make sure the OmniCode AI system is very secure. This means we have to use security techniques like blockchain-based authentication to protect user information. We have to make sure user data is handled in a secure way.
- Mobile Application Development: We need to develop applications for the OmniCode AI system. This will make it easier for users to access the OmniCode AI system on their devices. All these improvements will make the OmniCode AI system more intelligent, secure, scalable and user-friendly.

References

[1] I. Goodfellow (2016) *"Deep Learning"*

[2] S. Russell & P. Norvig, (2020) "*Artificial Intelligence: A Modern Approach"*, Pearson, 4th Edition.

[3] A. Ramesh et al., "*Hierarchical Text-Conditional Image Generation with CLIP Latents*," 2022.

[4] A. Vaswani et al., "Attention Is All You Need," *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[5] OpenAI, "GPT Models and API Documentation," 2024.

[6] Google DeepMind, "*Gemini AI Model Documentation and Capabilities*," Google DeepMind, 2024.

[7] Anthropic, "*Claude AI: Safe and Reliable AI Systems*," Anthropic AI, 2024.

[8] Stability AI, "*Stable Diffusion: Text-to-Image Generation Model*," Stability AI Research, 2023.

[9] OpenAI, "DALL·E: Creating Images from Text,".

[10] T. Preston-Werner, 2023 "*API Design and Integration Best Practices*," GitHub Docs, GitHub Inc.

[11] Comparative Analysis of Large Language Models for Code and Content Generation," *International Journal of AI Research*, vol. 12, no. 1, pp. 78–95, 2024

[12] "Dynamic API Routing for High Availability Systems," *Journal of Distributed Systems and Cloud Computing*, 2024.

[13] Blockchain-Based Logging for Secure and Transparent Systems," *International Journal of Blockchain Research*, 2023

[14] Mitchell, T. M., (1997) "*Machine Learning: Concepts and Applications*," IEEE and Elsevier Research Publications.

[15] Razorpay, "Payment Gateway Integration Documentation," Razorpay Pvt. Ltd., 2024.