Edututor ai: personalized learning with generative ai and lms

Team Name: LTVIP2025TMID31064

Team Members: TAVVA BHANUSRI

Phase – 1: Brainstorming and Ideation

Objective:

Identify the problem statement:

Edututor AI is an AI-powered personalized education platform that revolutionizes the way students learn and educators assess progress. It provides dynamic quiz generation, student evaluation, Google Classroom integration, and real-time feedback—all powered by IBM Watsonx and Granite foundation models. Designed with modular architecture, this platform streamlines personalized education and enhances learning outcomes for students across academic levels.

Define the purpose and the impact of the project

Purpose:

**EduTutor AI** aims to transform traditional education by combining **Generative AI** with **Learning Management Systems (LMS)** to deliver a highly **personalized, adaptive, and scalable learning experience**. Its primary purposes are:

**1.Personalized Learning Paths**: Tailor educational content, quizzes, and feedback based on each student's learning style, pace, and performance.

**2.Real-Time Assessment & Feedback**: Leverage AI to instantly assess student progress and provide meaningful feedback.

**3.Dynamic Content Generation**: Automatically generate quizzes, summaries, study materials, and explanations using powerful LLMs like IBM Granite.

**4.LMS Integration (e.g., Google Classroom)**: Seamlessly sync with existing classroom tools to enhance teacher and student workflows.

**5.Teacher Assistance**: Reduce manual effort in content creation, grading, and progress tracking, allowing educators to focus on mentoring.

Impact:

1.**For Students: Improved Engagement**: Interactive and relevant content tailored to their needs keeps students motivated.

* **Accelerated Learning**: Individual learning paths help students understand complex topics at their own pace.
* **Instant Support**: AI tutors provide 24/7 academic assistance, especially beneficial for remote or under-resourced learners.

2.**For Educators**

* **Efficiency Boost**: Automates routine tasks like grading, quiz creation, and student monitoring.
* **Insightful Analytics**: Helps track student performance trends and identify those needing intervention.
* **Inclusive Teaching**: Adapts to diverse learning styles and academic backgrounds, making teaching more inclusive.

3.**For Institutions**

* **Scalability**: AI scales learning without needing more human instructors.
* **Cost Reduction**: Saves costs on manual curriculum development and assessments.
* **Innovation Edge**: Positions the institution as forward-thinking and technologically advanced.

4.**For the Future of Education**

* **Equity in Learning**: Bridges gaps by offering quality education tools to all students regardless of geography or socioeconomic status.
* **Data-Driven Decisions**: Uses AI analytics to inform curriculum design and resource allocation.

**Key Points:**

**1.Problem Statement:** Edututor AI is an AI-powered personalized education platform that revolutionizes the way students learn and educators assess progress.

**2.**Proposed Solution: **EduTutor AI** is designed as a modular, AI-powered platform that integrates with existing Learning Management Systems (LMS) like **Google Classroom** and leverages **IBM Watsonx** and **Granite foundation models** to provide personalized, data-driven educational experiences. The solution dynamically generates content, evaluates student progress, and offers tailored feedback to improve learning outcomes.

**3.**Target Users:

| **User Group** | **Role** | **Key Benefit** |
| --- | --- | --- |
| **Students** | Learners at various academic levels | Personalized, AI-guided learning and feedback |
| **Educators** | Teachers, Tutors | Automated tools for teaching and assessment |
| **Institutions** | Schools, Colleges, Coaching Centers | Scalable, modern education delivery |
| **Parents** | Guardians of K-12 students | Transparent view of learning outcomes |
| **Government/NGOs** | Public education enablers | Equitable access to quality AI-driven education |

4.Expected Outcome:

| **Area** | **Expected Outcome** |
| --- | --- |
| **Students** | Better academic performance, higher engagement, tailored learning |
| **Teachers** | Time savings, actionable insights, improved instructional quality |
| **Institutions** | Scalable impact, digital transformation, measurable success metrics |
| **Education System** | Inclusive, data-driven, and future-ready learning environment |

Phase-2: Requirement Analysis

Objective:

Define technical and functional requirements.

To successfully design, develop, and deploy EduTutor AI, the following **technical components** are required, categorized under hardware, software, integrations, and infrastructure.

1.Hardware Requirements:

| **Component** | **Specification** |
| --- | --- |
| **Server/Cloud** | Cloud-based VM or GPU instance (IBM Cloud, AWS, GCP) |
| **Local Dev** | Minimum: 8 GB RAM, 4-core CPU, SSD storage |
| **GPU (Optional)** | For local LLM inference (e.g., NVIDIA RTX 3060 or better) |

**2. Software Requirements**

**A. Backend**

* **Language & Framework**: Python (FastAPI or Flask), or Node.js (Express)
* **AI Libraries**:
  + Hugging Face Transformers (for IBM Granite model integration)
  + PyTorch or TensorFlow (for fine-tuning or inference)
  + IBM Watsonx SDK/API

**B. Frontend**

* **Languages**: HTML, CSS, JavaScript
* **Frameworks**: React.js (Web), Flutter (Mobile)
* **Libraries**: Chart.js, Axios, Tailwind CSS, shadcn/ui

**C. Database**

* **Relational**: PostgreSQL or MySQL (for structured data)
* **NoSQL**: MongoDB or Firebase Firestore (for user data, real-time sync)

**3**.API Integration:

| **Service** | **Purpose** |
| --- | --- |
| **Google Classroom API** | Sync courses, assignments, student data |
| **OAuth 2.0 / Google SSO** | Secure login/authentication |
| **IBM Watsonx / Granite** | Use generative AI for content and feedback |
| **Email/Notification API** | Push alerts and feedback (e.g., SendGrid) |

4.Cloud & Devop:

| **Tool/Service** | **Function** |
| --- | --- |
| **Cloud Hosting** | IBM Cloud / AWS / GCP / Azure |
| **Containerization** | Docker for microservices |
| **Orchestration** | Kubernetes (K8s) or Docker Compose |
| **CI/CD** | GitHub Actions, Jenkins, or GitLab CI/CD |
| **Monitoring** | Prometheus, Grafana, or IBM Instana |

**5. Security & Compliance**

* **Authentication**: OAuth 2.0, Google Login
* **Data Encryption**: SSL/TLS for data in transit, AES for data at rest
* **Role-Based Access Control (RBAC)**: Different permissions for students, teachers, admins
* **Compliance**: GDPR, FERPA (for education-related data privacy)

**6. Optional Add-ons (Future Expansion)**

* **Speech-to-Text APIs**: For accessibility or voice input
* **AI Chatbot Interface**: Integrated with LLM for 24/7 tutoring
* **Multilingual Support**: For diverse learners using translation APIs

Key Points:

1.Techincal Requirements: Language

| **Type** | **Language(s)** |
| --- | --- |
| **Programming** | Python, JavaScript, Dart, SQL |
| **User-Facing Language** | English (default), multilingual optional |
| **AI Model Language** | English (primarily, multilingual possible with fine-tuning) |
| **Framework:**  **Layer** | Framework(s) Used |
| **Frontend (Web)** | React.js, Tailwind CSS, shadcn/ui |
| **Frontend (Mobile)** | Flutter (Dart) (optional) |
| **Backend** | FastAPI or Flask (Python), or Express.js |
| **AI/NLP** | Hugging Face Transformers, Watsonx SDK |
| **LMS/API** | Google Classroom API Client (Python/JS) |

Tools:

| **Category** | **Key Tools** |
| --- | --- |
| **Development** | VS Code, Git, GitHub, Postman |
| **AI & ML** | IBM Watsonx, Hugging Face, PyTorch, Transformers |
| **Frontend** | React.js, Tailwind CSS, shadcn/ui, Figma |
| **Backend** | FastAPI (Python), Express.js (JS), Flask |
| **LMS & APIs** | Google Classroom API, OAuth 2.0, Firebase |
| **DevOps** | Docker, GitHub Actions, IBM Cloud / AWS, Kubernetes |
| **Monitoring** | Grafana, Prometheus, Sentry, Google Analytics |

2.Functional Requirements:

| **Category** | **Feature** |
| --- | --- |
| **Personalization** | Adaptive content, learning paths |
| **AI Integration** | Quiz generation, explanations, chatbot |
| **Assessment** | Auto-evaluation, real-time feedback |
| **Teacher Tools** | Dashboards, content curation, analytics |
| **LMS Sync** | Google Classroom integration, SSO |
| **Accessibility** | AI assistant, multilingual, mobile-ready |
| **Security** | Role-based access, secure logins |
| **Scalability** | Modular microservic |

3.Constraints and Challenges:

| **Category** | **Example Challenges** |
| --- | --- |
| **Technical** | Compute cost, integration limits, latency |
| **Security & Privacy** | Data protection, compliance, secure access |
| **Content & AI Quality** | Curriculum accuracy, bias, multilingual support |
| **Adoption & Usability** | Teacher/student onboarding, device/internet access |
| **Maintenance** | AI model updates, monitoring, scalability issues |

Phase-3: Project Design

Objective:

Create the articheture and user flow

[ Student Device ] [ Teacher/Admin Device ]

| |

|------------- Frontend Layer ------------|

| (React.js / Flutter + Tailwind) |

| Dashboards | Quizzes | Chatbot UI |

| |

| ↓ |

|---------- API Gateway / Load Balancer -----------|

|

┌──────────────┴──────────────┐

| |

┌────────▼────────┐ ┌─────────▼─────────┐

| Backend Server | | Auth Service |

| (FastAPI/Node.js)| | (OAuth, Google SSO)|

└────────▲────────┘ └─────────▲─────────┘

| |

┌─────────────┴─────────────┐ ┌─────────┴─────────────┐

| AI Engine / LLM API | | LMS Integration API |

| (IBM Watsonx / Granite LLM)| | (Google Classroom API)|

└─────────────┬─────────────┘ └─────────┬─────────────┘

| |

┌────────────▼───────────┐ ┌──────────▼────────────┐

| Quiz Generator Module | | Course Sync Module |

└────────────┬───────────┘ └──────────┬────────────┘

| |

┌────────▼─────────┐ ┌─────────▼────────┐

| Feedback Engine | | Assignment Push |

└────────┬─────────┘ └─────────┬────────┘

| |

┌──────▼──────┐ ┌─────▼─────┐

| Database | | File Store |

| (PostgreSQL /| | Firebase / |

| MongoDB) | | GCS / S3 |

**Modular microservices**: Each service (quiz generation, feedback, sync) can scale independently.

**Secure Auth**: OAuth 2.0 and Google SSO for safe, simple login.

**Real-Time Data**: LMS syncs and instant feedback mechanisms keep users constantly updated.

**AI Core**: IBM Watsonx and Granite LLM handle smart generation and responses.

4.Phase-4: Project Planning (Agile Methodlogies):

Objective:

Break down the tasks using agile Methodlogies

**1. Define High-Level Epics**

These are major components of the system:

1. **User Management**
2. **Google Classroom Integration**
3. **Course & Content Sync**
4. **Generative AI Quiz Creation**
5. **Student Evaluation & Feedback**
6. **Teacher Dashboard**
7. **AI Model Integration (Granite)**
8. **Analytics and Progress Tracking**
9. **Authentication & Authorization**

**10.Deployment & DevOps**

**2. Convert Epics to User Stories**

Here’s a sample breakdown of a few epics into **user stories** in the format:

As a <type of user>, I want to <action> so that <benefit>.

**Epic: User Management**

* *US1.1:* As a student, I want to register and log in using my Google account so that I can access my personalized learning dashboard.
* *US1.2:* As a teacher, I want to create an account so that I can manage student progress.

**Epic: Google Classroom Integration**

* *US2.1:* As a student, I want to sync my Google Classroom data so that courses and assignments are automatically available.
* *US2.2:* As a teacher, I want to import students and coursework from Google Classroom so that I don't need to add them manually.

**Epic: Generative AI Quiz Creation**

* *US4.1:* As a student, I want the system to generate quizzes based on my course topics so I can self-test my knowledge.
* *US4.2:* As a teacher, I want to review and edit AI-generated quizzes before assigning them.

**Epic: AI Model Integration (Granite Model)**

* *US7.1:* As a developer, I want to connect to the IBM Granite 3.3 2B model to enable content generation.
* *US7.2:* As a teacher, I want to customize prompt templates for quiz generation.

**3. Sprint Planning (2-week sprints)**

Here’s how you might plan out **tasks per sprint** for early development:

**Sprint 1: Project Setup & Auth**

* Setup GitHub repo, Google Colab notebook, and CI/CD base
* Implement Google OAuth login
* Create database schema for users and roles
* Build student and teacher dashboards (basic layout)

**Sprint 2: Google Classroom Sync**

* Integrate Google Classroom API
* Sync courses and assignments
* Store synced data in DB
* Display student assignments in dashboard

**Sprint 3: AI Quiz Generation**

* Connect to IBM Granite model via Hugging Face
* Build prompt system for generating questions from course topics
* Display quiz to user (MCQs, Short Answer)
* Save and evaluate responses

**Sprint 4: Feedback & Analytics**

* Implement evaluation logic (score, feedback)
* Show progress report in student dashboard
* Create analytics for teachers (heatmap of class performance)

**Sprint 5: Teacher Dashboard & Controls**

* Course overview, student list, quiz manager
* Edit and assign AI-generated quizzes
* Review student answers and give manual feedback

**Sprint 6: Testing & Deployment**

* Write unit and integration tests
* Deploy to Hugging Face Spaces or Streamlit/Gradio
* Performance optimization (batch API calls, UI responsiveness)

**4. Agile Artifacts**

| **Artifact** | **Description** |
| --- | --- |
| **Product Backlog** | List of all user stories, prioritized by business value |
| **Sprint Backlog** | Subset of backlog items selected for a sprint |
| **Burndown Chart** | Tracks remaining work in a sprint |
| **Daily Standups** | Team syncs to review progress and blockers |
| **Sprint Review** | Demo working features to stakeholders |
| **Retrospective** | Team reflects on what went well and what to improve |

**Example Agile Board (Kanban)**

| **To Do** | **In Progress** | **In Review** | **Done** |
| --- | --- | --- | --- |
| Set up DB schema | Google Login | Google Classroom sync | GitHub Setup |
| Quiz prompt design | Quiz UI | Quiz result logic |  |

Phase-5: Project Development

Objective:

Code the project and integrate components

edututor-ai/

├── app.py ← Main Gradio app

├── classroom.py ← Google Classroom integration

├── ai\_generator.py ← IBM Granite model integration

├── auth.py ← Google OAuth login

├── quiz.py ← Quiz generation & evaluation logic

├── data/ ← Store course/user data locally

└── requirements.txt

Keypoints:

1.Technology Stack Used:

| **Layer** | **Stack** |
| --- | --- |
| **Frontend** | Gradio (Python UI) |
| **Backend** | Python |
| **Authentication** | Google OAuth |
| **AI/LLM** | IBM Granite via Hugging Face |
| **LMS** | Google Classroom API |
| **Deployment** | Colab, Hugging Face Spaces, or Streamlit |
| **Package Manager** | pip (Python) |

2. Development Process:

1. **Planning & Requirements:**
   * Define user roles (students, teachers).
   * Identify core features: AI quiz generation, Google Classroom integration, performance feedback.
2. **Design:**
   * Create system architecture and UI/UX mockups.
   * Choose tech stack: React.js, Node.js, IBM Watsonx, Google Classroom API.
3. **Agile Development:**
   * Use Scrum with sprints and backlogs.
   * Break down into modules (Auth, Quiz, AI Engine, Dashboard).
4. **Implementation:**
   * Frontend: React components for dashboard, quiz, and login.
   * Backend: API services, database, AI model integration.
5. **Testing:**
   * Unit, integration, and user testing for quality assurance.
6. **Deployment:**
   * Deploy on cloud (Vercel/AWS).
   * Use CI/CD for updates and maintenance.
7. **Monitoring & Improvement:**
   * Track usage, collect feedback, and iterate features

3.Challenges and Fixes:

| **Challenge** | **Description** | **Fix / Solution** |
| --- | --- | --- |
| **1. AI Accuracy** | Quiz questions generated were sometimes too vague or irrelevant. | Refined prompts and used topic-specific context in Watsonx/Granite queries. Added manual review option. |
| **2. API Integration (Google Classroom)** | OAuth and data sync issues during integration. | Used official SDKs, implemented detailed error handling, and added refresh token logic. |
| **3. Student Data Privacy** | Risk of exposing sensitive academic data. | Enforced end-to-end encryption, OAuth2.0 for access control, and role-based authentication. |
| **4. Performance Issues** | Delays in quiz generation and dashboard loading. | Implemented caching (Redis), optimized queries, and used async job queues. |
| **5. UI Complexity** | Overwhelming dashboard for users with multiple courses. | Simplified interface, added filters, and context-aware UI rendering. |
| **6. Scalability** | Difficulty in handling simultaneous AI requests from many users. | Adopted microservices, used load balancer and horizontal scaling via cloud services. |
| **7. AI Model Cost & Latency** | High cost and delay from repeated AI model calls. | Batched AI calls, used local inference for small tasks, and introduced usage limits. |
| **8. Testing AI Outputs** | Hard to validate correctness of AI-generated quizzes. | Added human feedback loop and accuracy scoring from pilot users. |

Phase-5: Functional & Performance Testing:

Objective:

Ensure the project works as expected

Here’s a **summary** of how to **ensure EduTutor AI works as expected**:

**✅ Project Validation Summary – EduTutor AI**

1. **Functional Testing:**
   * Ensure all features (login, course sync, quiz generation, feedback) work as intended.
2. **AI Output Validation:**
   * Review quiz quality, accuracy, and relevance generated by the AI model.
3. **User Role Verification:**
   * Confirm students and teachers see only their relevant dashboards and tools.
4. **Security Checks:**
   * Test authentication, data privacy, and role-based access.
5. **Performance Testing:**
   * Check system speed and reliability under multiple user loads.
6. **CI/CD & Automation:**
   * Use automated tests and deploy through CI/CD pipelines to reduce manual errors.
7. **Monitoring & Feedback:**
   * Implement real-time error tracking and collect user feedback for continuous improvement.

Key Points:

1.Test case Excueted:

Scenario 1: Personalized Learning Experience

A student logs into EduTutor AI and synchronizes their courses using their Google Classroom credentials. The platform analyzes course data, generates quizzes on key topics using the Granite LLM, and assesses responses for instant feedback—creating a highly personalized and engaging learning journey.

Scenario 2: Educator Dashboard & Performance Insights

Educators can log in to view real-time quiz performance of all students. The dashboard highlights quiz history, scores, last topics attempted, and insights fetched from the Pinecone vector database. This empowers teachers to monitor learning progress and personalize their instruction based on data.

Scenario 3: Diagnostic Testing and Adaptive Quizzing

Upon registration, students undergo a diagnostic test generated by IBM Watsonx models. Based on the results, the platform adapts quiz difficulty and topic relevance, ensuring students are challenged at the right level.

Scenario 4: Google Classroom Integration

EduTutor AI syncs courses directly from Google Classroom, allowing seamless access to student data, class names, and subjects. This enables automatic quiz topic generation and helps maintain consistent alignment with the academic curriculum.

2.Bugs Fixes and Improvements:

| **Category** | **Bug / Issue** | **Fix** | **Improvement** |
| --- | --- | --- | --- |
| **Authentication** | Login fails intermittently due to expired Google tokens | Implement automatic token refresh logic | Add retry mechanism and clearer error messages |
| **Course Sync** | Some Google Classroom courses not showing | Handle pagination and permission errors in API calls | Add sync status indicator for better UX |
| **Quiz Generation (AI)** | Irrelevant or too advanced questions | Refine prompts with course-level context | Allow teacher editing and difficulty settings |
| **Quiz Submission** | Submitted answers not saving occasionally | Fix API timeout and ensure database write confirmation | Add autosave during quiz attempt |
| **Performance Dashboard** | Charts not updating in real-time | Fix state management issue in frontend (e.g., Redux bug) | Use WebSocket or polling for live updates |
| **Mobile Responsiveness** | Layout breaks on smaller screens | Adjust CSS grid and flexbox settings | Add responsive design testing for all devices |
| **Latency in AI Response** | Slow quiz generation on peak hours | Optimize API calls and use asynchronous queuing | Cache previously generated quizzes for re-use |
| **Role Access Control** | Teachers able to access student view (or vice versa) | Add stricter role-based routing checks | Implement user role middleware validation on backend |

3.Final Validation:

1. **All Features Complete:**
   * Login, course sync, AI quiz generation, dashboards, and teacher tools fully functional.
2. **Role-Based Access Tested:**
   * Students, teachers, and admins access only their authorized modules.
3. **AI Output Validated:**
   * Quiz questions are accurate, relevant, and formatted correctly.
4. **Performance Verified:**
   * Fast response time; system handles concurrent users efficiently.
5. **Security Ensured:**
   * OAuth 2.0, data encryption, and privacy compliance implemented.
6. **Responsive UI/UX:**
   * Mobile-friendly, intuitive interface with clear navigation.
7. **No Critical Bugs:**
   * Major issues fixed; full QA and regression testing completed.
8. **Monitoring Enabled:**
   * Real-time error tracking and user feedback systems in place.

4.Deployement:

1. **Prepare for Deployment:**
   * Final code tested and merged
   * Environment variables configured
2. **Deploy Components:**
   * **Frontend (React):** Deploy on Vercel or Netlify
   * **Backend (Node.js):** Deploy on Render, Heroku, or AWS
   * **Database:** Use MongoDB Atlas or PostgreSQL
3. **AI Integration:**
   * Connect backend to IBM Watsonx and Granite via API
4. **Domain & SSL:**
   * Link custom domain and enable SSL (via Vercel/Netlify)
5. **CI/CD:**
   * Set up GitHub Actions for automatic build & deployment
6. **Post-Deployment:**
   * Test live features, monitor errors, ensure AI and login work