NexEra – AI Quiz Master

Team ID:

Sachin Pundalik Malik 1002202264

Sudipta Das 1002 189479

**PROJECT DESCRIPTION:**

NexEra – AI Quiz Master is an intelligent web app that makes studying easier and more interactive. It allows users, either students or educators to upload PDF documents, like notes, or study guides. The app then automatically generates quizzes based on the content, helping users test their understanding without manually crafting questions.

In NexEra – AI Quiz Master, every quiz will be a combination of multiple-choice and descriptive questions, rather than separating them into different assessments. This approach ensures a more balanced evaluation, where users can quickly check their factual understanding with multiple-choice questions while also engaging in deeper reflection through descriptive answers, all in the same quiz.

This design makes studying more interactive and thorough, allowing students and educators to assess both knowledge recall and critical thinking in one go. Plus, the app’s AI-driven quiz regeneration means that users can always get fresh and varied sets of questions based on the content they uploaded, keeping learning dynamic and engaging. But it doesn’t stop there, NexEra provides a personalized dashboard, where users can track their progress, see quiz history, and get insights on their strengths and weaknesses. Plus, if someone wants a different set of questions, the app uses AI to regenerate quizzes intelligently, ensuring that learning stays fresh and adaptive.

With built-in activity tracking, users can monitor their study patterns, making it a great tool for anyone who wants to stay on top of their learning goals. Whether preparing for exams or simply reinforcing knowledge, NexEra helps transform passive reading into an engaging and effective learning experience.

**PROJECT SCOPE:**

The project includes both frontend and backend development, leveraging AI capabilities to generate quiz content.

Key functionalities delivered:

*User Sign Up & Login***:** Secure authentication system for account creation and login.

*PDF Upload***:** Users can upload PDFs (up to 1MB) for processing.

*Quiz Generation***:** AI-generated quizzes (MCQs + descriptive) based on document content using Gemini AI.

*Question Count with Regeneration***:** Each quiz has 10 questions; users can regenerate new sets from the same file.

*Answer Submission & Scoring***:** Real-time quiz evaluation and score display.

*Quiz History***:** Tracks user's past quizzes (last 1 month), including filenames, dates, and scores.

*Dashboard Summary***:** Total quizzes, average score, last active date, and weekly performance graph.

*Profile Management***:**  Edit/view user details (name, email).

**PROJECT RESOURCES:**

Frontend:

* React.js, React Router
* Fetch API

Backend/AI:

* Gemini AI (Google)
* GitHub Copilot, ChatGPT (assistance)

Database & Auth:

* Postgres

Dev Environment:

* VS Code
* npm
* Git + GitHub

Testing:

* Pytest (Unit Testing)
* Claude (Debugging Assistance)

Hardware Requirements:

* Intel i5 / AMD Ryzen 5+
* 8GB RAM (16GB recommended)
* 10GB free disk space
* Stable internet connection

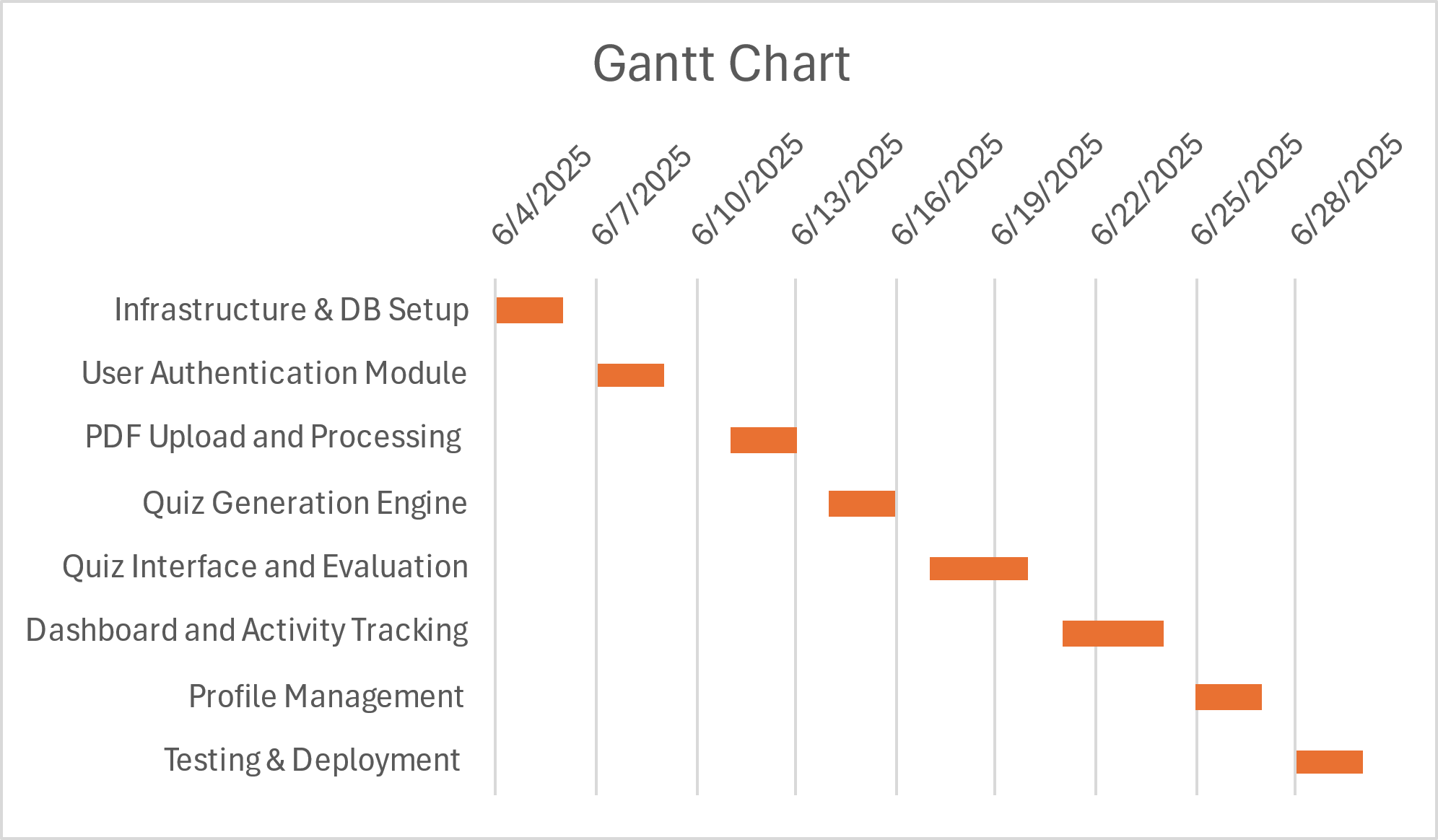
**Project Constraints**

Given that NexEra – AI Quiz Master is a classroom project with a fixed one-month timeline, there are several practical constraints we’re working within:

1. **Time Limitation**: The entire project, from planning to final testing, must be completed within 4 weeks. This requires a clear focus on core features and careful time management.
2. **Part-Time Development**: As student developers, we’re balancing this project alongside other academic commitments. Each team member can contribute roughly 20 hours per week.
3. **Limited Resources**: Since we’re working locally without cloud infrastructure, we’re keeping PDF uploads under 1MB and staying within the free tier limits of the APIs we’re using.
4. **Defined Scope**: To stay on schedule, we’re focusing strictly on the planned core functionalities. Features like timer-based quizzes or multi-language support are intentionally left out for now.
5. **AI Dependency**: The quiz generation depends on Gemini AI. If the API becomes unavailable or slow, it could affect the user experience.
6. **Local Testing Only**: Testing will be done on local machines using a limited range of browsers and devices, so broader compatibility can’t be guaranteed at this stage.

**PROJECT TIMELINE:**

***Gantt Chart:***



***WBS(Work Breakdown Structure):***

A diagram of a computer

AI-generated content may be incorrect.

***Cost and Effort Estimation Using COCOMO II:***

Cost Referenece <https://www.hyperlinkinfosystem.ca/cost-to-develop-an-app-like-quizup>

<https://theappideas.com/build-a-quiz-app/>

<https://trangotech.com/blog/ai-app-development-cost/>

To estimate the development cost of NexEra – AI Quiz Master, we explored both industry references and formal modeling methods.

We reviewed external cost benchmarks from the following sources:

Hyperlink InfoSystem – Estimates $25,000–$80,000+ for a full-featured quiz app.

The App Ideas – Provides detailed time estimates (390–590 hours) for quiz app development, leading to a cost range of $15,600–$23,600 at a $40/hour rate.

TrangoTech – Estimates $20,000–$60,000 for simple AI-integrated apps using GPT-4 or similar models.

While these are professional-grade estimates for commercial products, our project is a classroom assignment, leveraging previous development experience and pre-trained AI models (e.g., Gemini/GPT) for quiz generation. Therefore, we needed a method to align industry insights with our academic setting.

COCOMO II-Based Estimation

To refine our estimate, we used the COCOMO II cost model, entering:

3,500 SLOC (source lines of code) as estimated application size.

A developer cost rate of $5,000/month as advised by our instructor.

Parameter Selection Rationale:

Most cost drivers were kept at Nominal, as the project is standard in terms of complexity, tooling, and team dynamics.

We selected Low for “Developed for Reusability” since the code is not intended for reuse beyond this course.

High was chosen for “Time Constraint” and “Required Development Schedule,” as we must complete the project within 4 weeks.

High was also used for Application Experience and Personnel Continuity, reflecting our prior familiarity with web and AI tools, and the stable three-member team working consistently throughout.

Results

Estimated Effort: 9.5 person-months

Estimated Schedule (1 person): 9.8 months

Estimated Cost: $47,626

If this effort were distributed across 3 full-time developers, the project would take approximately 3.2 months. Since we are working part-time (~20 hours/week per person), the timeline aligns reasonably well with our 4-week delivery target, assuming prior code reuse and efficient module division.

Conclusion

While professional development of AI-powered quiz apps ranges from $15,000 to over $60,000, our project—built with focus, academic intent, and reused components—was reasonably estimated to cost $47,626 using the COCOMO II model. This provides a realistic upper-bound estimation that reflects the effort and scope of what we are building.

A screenshot of a computer

AI-generated content may be incorrect.

**1. Risk Factor Identification**

The NexEra project is a one-month, classroom-based software development initiative involving three student developers. Given the limited duration, part-time team availability, and use of advanced components like AI integration and PDF processing, the project is subject to a variety of risks. The following risks have been identified:

* Time constraint due to fixed 4-week deadline
* Limited team availability due to overlapping academic responsibilities
* Limited prior experience with certain tools (React.js, Gemini API)
* Dependency on AI APIs (e.g., GPT/Gemini) which may have rate limits or downtime
* Inconsistencies in uploaded PDF formatting affecting parsing qualit
* Limited time and environment for thorough testing
* Scope creep due to team ambition or user suggestions
* Cross-device/local deployment issues
* Potential delays in documentation and report writing

**2. Risk Exposure Calculation**

Risk exposure is calculated as:

**Risk Exposure ($) = Probability (%) × Impact ($)**

Based on a final project cost estimated using COCOMO II at $47,626, each risk's exposure is calculated as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk Description** | **Probability (%)** | **Impact ($)** | **Risk Exposure ($)** |
| Time constraint (4-week limit) | 50 | 6000 | 3000 |
| Team availability (academic workload) | 40 | 4000 | 1600 |
| Tool familiarity (React, AI API) | 60 | 7000 | 4200 |
| AI API dependency (rate limits/downtime) | 30 | 3000 | 900 |
| PDF parsing/formatting issues | 35 | 2500 | 875 |
| Testing limitations (local-only, short time) | 45 | 4500 | 2025 |
| Scope creep (adding features mid-project) | 25 | 2000 | 500 |
| Deployment inconsistencies (local machines) | 20 | 1500 | 300 |
| Documentation/report delay | 30 | 2500 | 750 |

**Total Risk Exposure: $14,150**

**3. Risk Mitigation Plan**

|  |  |  |
| --- | --- | --- |
| Risk Description | Mitigation Strategy | Strategy Type |
| Time constraint (4-week limit) | Define MVP early, use weekly milestone check-ins | Risk Control |
| Team availability (academic workload) | Use shared Kanban board, flexible task ownership | Risk Transfer |
| Tool familiarity (React, AI API) | Assign tech leads, use official docs/tutorials | Knowledge Refinement |
| AI API dependency (rate limits/downtime) | Set API usage limits, mock fallback during dev | Risk Control |
| PDF parsing/formatting issues | Restrict to text PDFs, show file requirements early | Risk Control |
| Testing limitations (local-only, short time) | Begin testing in Week 2, test incrementally | Risk Control |
| Scope creep (adding features mid-project) | Lock scope Week 1, track extras in Phase 2 list | Risk Control |
| Deployment inconsistencies (local machines) | Standardize environments, share setup guides | Knowledge Refinement |
| Documentation/report delay | Assign writing roles early, review weekly | Risk Transfer |

This plan ensures that each risk is either controlled, reduced, or prepared for through concrete, proactive actions tailored to the scope and constraints of the NexEra classroom project.

**DEVELOPMENT METHODOLOGY:**

We have used Kanban because it is a simple, flexible, and visual approach to managing tasks, making it ideal for the tight 4-week schedule.

* *No rigid deadlines or sprints*: Unlike Scrum, which requires fixed time blocks (sprints) to complete work, Kanban allows tasks to be moved at their own pace, keeping priorities clear. This makes it easier to adapt if something takes longer or requires adjustments.
* *Visual clarity* : A Kanban board with columns (such as To Do, In Progress, Done) provides an instant overview of what needs attention, what is being worked on, and what has been completed. This helps maintain organization without the need for complex task lists.
* *Prevents overload*: Since Kanban limits the number of tasks in progress, it ensures that the workload remains manageable, reducing confusion and improving workflow efficiency.
* *Easy to adjust priorities*: If new tasks emerge or a shift in focus is required, priorities can be adjusted without disrupting the workflow. There is no need to wait for a sprint to end before making changes.

For this project, each week's work has been clearly planned, but without the restrictions of rigid timeframes. If quiz generation takes longer than expected, adjustments can be made without affecting the overall progress. This approach ensures that the team stays efficient and adaptable while maintaining a structured development process

**Software Quality Attributes in NexEra – AI Quiz Master**

To ensure the effectiveness and maintainability of NexEra within the scope of our classroom project, we are focusing on a selected set of software quality attributes that are both achievable and meaningful for our users and team structure. These include:

**1. Usability**

We want users—both students and educators—to be able to interact with the platform effortlessly. To maintain usability:

* The interface will be kept clean and intuitive using React with clear navigation.
* Key actions like uploading PDFs or starting a quiz will be just one or two clicks away.
* We'll perform informal user testing among classmates to gather quick feedback.

**2. Reliability**

Given that the app will be used for self-assessment, users must trust the system to behave consistently. To ensure reliability:

* We'll validate inputs and implement error-handling for quiz generation and PDF uploads.
* We plan to test core features like login, PDF parsing, and quiz regeneration thoroughly before the final deployment.

**3. Modularity**

We’re designing the system as a set of independent modules—authentication, quiz generation, scoring, and dashboard. This:

* Makes the code easier to maintain and debug.
* Allows different team members to work in parallel without interference.
* Supports future extensions like file-type expansion or adding new AI models.

**4. Understandability**

Since this is a team project, it’s important that all team members can understand and contribute to each other's code. To promote understandability:

* We will use consistent naming conventions and leave inline comments.
* Each module will be documented briefly so another team member can pick it up if needed.

**5. Testability**

To prevent bugs and support future refactoring, we’re writing simple unit tests:

* Functions related to quiz generation, scoring, and PDF validation will be tested.
* We'll rely on tools like React Testing Library and mock data for key flows.

**6. Learnability**

If new developers or contributors want to understand the app later (e.g., in an academic showcase or open-source version), the system should be easy to pick up. We’ll support this by:

* Writing a brief README file explaining how to run the app.
* Structuring the frontend/backend files clearly and using standard React/Firebase patterns.

2. Kanban board :

Several pieces of paper

AI-generated content may be incorrect.**RESOURCE ALLOCATION:**

***Sachin:*** Quiz UI, dashboard, score tracking, testing & deployment.

***Sudipta:*** PDF upload, quiz generation, profile management.

**CONCLUSION:**

NexEra – AI Quiz Master is a focused and achievable software project designed to provide an intelligent, user-friendly platform for generating and evaluating quizzes from uploaded PDF content. To ensure the project remains maintainable and adaptable in the future, we are adopting modular and scalable architecture. Each core functionality, such as authentication, PDF handling, quiz generation, and user analytics will be developed as independent components. This modular approach allows for easy integration of new features, such as additional file types, new question formats, or expanded analytics, without requiring significant changes to the existing codebase.

Overall, this project not only reinforces key software engineering practices but also demonstrates our ability to design scalable, real-world applications that integrate modern tools and AI services effectively.

**REFERENCE:**

Pressman, R. S., & Maxim, B. R. (2014). *Software Engineering: A Practitioner's Approach* (8th ed.). McGraw-Hill Education.

Sommerville, I. (2015). *Software Engineering* (10th ed.). Pearson.

Google Gemini AI Overview: <https://ai.google>

GitHub Docs: <https://docs.github.com>

Kanban – Atlassian: <https://www.atlassian.com/agile/kanban>

Jest Documentation: <https://jestjs.io>

Visual Studio Code Docs: <https://code.visualstudio.com/docs>