YourStory

Final Report

1. Introduction

The Interactive Storytelling and Learning Application is an innovative digital platform that aims to enhance the literacy and cognitive skills of children through personalized storytelling. In the 21st century, technology has changed the face of education by making learning more interactive and adaptive. Our project utilizes Artificial Intelligence (AI), Natural Language Processing (NLP), and Dynamic Difficulty Adjustment (DDA) in creating an engaging, customized learning environment that automatically adapts to the unique needs of each child. We are at the forefront of leveraging advanced technology to bridge the gap between traditional teaching methods and new digital learning techniques by inculcating an interest in reading and developing thinking at a tender age.

1.1 Computer Systems and Importance in the 21st Century

Computer systems have now become an integral part of education, offering new approaches to enhance learning experiences. Traditional learning methods often lack interactivity and fail to cater to diverse learning styles. With the rise of digital platforms, educational tools can now adapt to individual students, track progress, and provide instant feedback. AI, cloud computing, and mobile applications employed in educational institutes have increased the accessibility of their learning resources by manifold times. Our project essentially demonstrates this sea change in educational pedagogies through an innovative storytelling experience by using AI and NLP techniques that will be helpful in nurturing curiosity, creativity, and developing language among children. In fact, we implement interactive technologies so that instead of being information receivers, students may be the actual participants in their learning process.

1.2 Innovation Field

Technology continues to redefine the educational landscape, opening new ways for teaching and improving learning outcomes. Interactive and adaptive learning tools ensure that children receive content at the right stage in their development, thus helping them grasp concepts much more effectively. This paper has identified personalized learning as a critical innovation in this field due to its ability to dynamically adjust content to meet the unique needs of each learner. Our application, on the other hand, takes in real-time personalization along with choice-based storytelling to let students have more enhanced learning processes through critical thinking, decision-making skills, and cognitive development in active engagement.

1.2.1 Crucial Innovative Components

Our project integrates several key innovative components to ensure an engaging and effective learning experience:

- **AI-Powered Personalization:** The system adapts content based on each child's vocabulary level, learning progress, and preferences, ensuring a tailored educational experience. Unlike traditional static learning tools, our AI continuously refines and updates content based on ongoing user interactions.
- **Choice-Based Storytelling:** Children make decisions that influence the story’s progression, enhancing engagement and promoting logical reasoning skills. By giving children control over how the story unfolds, the application fosters problem-solving abilities and creative thinking.
- **Real-Time Difficulty Adjustment:** The application dynamically modifies content complexity to match learning progress, ensuring that children are neither overwhelmed nor under-challenged. This feature maintains a perfect balance between learning and entertainment, preventing frustration while keeping students motivated.

- **Parental Dashboard:** Provides real-time insights into a child's progress and learning patterns, allowing parents and educators to monitor and support development. Parents can set goals, receive performance analytics, and suggest story themes based on their child’s interests.
- **Gamification Elements:** Incorporates reward systems, achievements, and interactive elements to sustain motivation and encourage consistent engagement with the learning material. Children are rewarded for progress with virtual badges, unlockable content, and personalized story enhancements, making learning more enjoyable.

1.3 Problem Statement

The majority of conventional learning applications fail to maintain the interest of children due to the fact that they are static and one-size-fits-all. Such applications merely present the same content to everyone, regardless of a child's reading level, learning pace, or interests. Consequently, children who perceive the content to be too challenging become frustrated, and those who perceive it to be too easy become bored. Without adaptive content that changes based on a child's progress, learning is repetitive and inefficient, reducing motivation and retention.

Another serious issue is the lack of detailed progress tracking. Many programs show basic completion percentages or quiz scores but fail to provide a deeper insight into a child's strengths and weaknesses. It is difficult for parents and teachers to identify areas that need improvement. Kids will continue to struggle with certain concepts without proper intervention, ultimately inhibiting their academic progress, as teachers and parents are not provided with real-time feedback on the levels of understanding and engagement.

Second, conventional educational apps do not incorporate interesting features that facilitate fun and interactive learning sessions. The majority of them are based on passive consumption of content where kids merely read or watch but do not participate actively. But we know through research that kids learn optimally if they are able to interact with content through things like choice-based stories, gamified quizzes, and instant feedback. In the absence of these features, the level of interaction decreases, making it hard for kids to be motivated and learn.

Another drawback is the absence of personalized learning experiences. Each child has unique interests, but most apps offer set lessons or stories with no personalization. This mismatch makes it more difficult for children to relate to the content, decreasing the efficiency of the learning process.

Last, the majority of platforms are not integrated to feature AI-enabled adaptation, gamification, and monitoring of progress under one umbrella. Rather, they concentrate on solitary elements of education, thereby promoting disjointed learning experiences. Absent from a smart, adaptive system with features of content personalization, tracking of progress, and ongoing interaction, such apps do not achieve an impactful, enjoyable learning process. For overcoming these deficits, there has to be more dynamism based on integrating personalization, timely feedback, and active interaction towards greater learning accomplishments.

1.4 Background Study

There is strong empirical evidence for the effectiveness of interactive digital learning in promoting children's literacy and higher-order thinking skill development. Conventional learning approaches founded on fixed text and passive acceptance do not engage the full attention of children as learners or facilitate deep-level learning. Storytelling games, on the other hand, offer a contextual and participative experience in which children are active learners and not merely passive recipients. Research has proven that when kids have the liberty to interact with content within the framework of interactive storytelling, their retention rate is much greater. With decision-making, character interaction, and adaptive stories, these applications offer a learning experience that ignites curiosity, creativity, and critical thinking.

Follow-up research highlights the contribution of individualized digital learning supports in increasing language proficiency and literacy rates. Students with access to such adaptive learning systems have demonstrated a 27% increase in language skills, says the Joan Ganz Cooney Center. This is because interactive digital learning systems are capable of adapting content to a child's individual reading level, learning profile, and interests. In contrast to conventional models of education that adhere to a fixed system, these models adapt dynamically to present challenges at a child's current level of proficiency while maintaining their interest with pertinent and significant content. This one-to-one model avoids frustration with work that is too difficult as well as tedium with work that is too easy.

Aside from literacy, interactive storytelling applications have also proven to improve cognitive abilities by improving problem-solving, decision-making, and vocabulary. Conventional fairy tale books for children or reading software usually offer a predetermined order of events, constraining a child from discovering other possible stories or even engaging in participation to determine the plot of the story. AI-driven storytelling programs, on the other hand, offer branching stories where children can make decisions that impact the story direction, contributing to more active and stimulating engagement. As they move through various story resolutions, they develop

competencies like logical reasoning, cause-and-effect, and creative thinking, all of which cumulatively build general cognitive development.

The incorporation of AI in interactive storytelling also enhances the learning experience by enabling it to be adaptive and responsive. AI-powered features allow the system to assess a child's reading behavior, levels of comprehension, and areas of interest in real time, thus allowing it to create stories that are personalized to the individual needs of each student. This creates an extremely personalized and dynamic learning process in which every two reading sessions are different. Further, AI has the capacity to provide real-time feedback, directing the child to learn more, recommend new words, and even alter sentence structures in line with their learning trajectory. The personalization helps to keep the child engaged every step of the way while progressively improving their literacy level.

The Interactive Storytelling and Learning App goes one step ahead of all these by combining learning and entertainment to provide a fully interactive learning experience. Through AI-powered storytelling, gamified interaction, and adaptive learning methodologies, the app ensures that the kids are not just reading but are involved in the process. This not only makes learning literacy a fun experience but also more rewarding, thereby making it easy to help the kids form good reading habits while having fun. Through interactive features, adaptive difficulty, and real-time progress monitoring, the app optimizes both engagement and educational efficacy, establishing a new benchmark for learning apps.

1.5 Our Solution and Addressing Main Concerns

Our application addresses the existing limitations of educational applications by incorporating an AI-powered adaptive learning system that guarantees every child a personal experience according to his or her reading level, interests, and progress.

1.5.1 Solutions

- **Interactive Learning Paths:** Children are actively engaged in developing the story by coming to conclusions that affect the plot, resulting in increased agency and engagement.
- **Adaptive Content:** Computer algorithms with AI review comprehension when reading and modify vocabulary and sentence difficulty to correspond with the child's capability to avoid frustration or boredom.

- **Comprehensive Progress Tracking:** A detailed teacher and parental dashboard tracks learning progress, identifying strengths and weaknesses.
- **Gamification Elements:** Includes achievements, grades, and awards to encourage steady learning and engagement, offering an enjoyable and effective learning environment.

1.5.2 Addressing Main Concerns

- **Lack of Personalization:** Personalization through AI tailors content in real-time to the requirements of each child, offering the most ideal learning experience.
- **Limited Engagement:** Immersive elements such as decision-based storylines, animations, sound effects, and interactive narratives keep children actively engaged.
- **Progress Monitoring:** A parental dashboard in real-time offers feedback on reading behavior, vocabulary development, and level of engagement, allowing parents to support their child's learning process.
- **Security and Privacy:** Adherence to COPPA and GDPR guarantees the safety of user information, tackling issues of children's safety online.

1.6 Target Niche

The primary users of this application are children between the ages of four and ten, a critical period of their cognitive and linguistic development. This is the age group most responsive to interaction and visual stimulus in learning materials, and they would therefore be perfect candidates for adaptive storytelling applications. Young learners have varied reading levels, interests, and comprehension abilities, and for this reason, it is not possible to adopt one-size-fits-all approaches to education. With this platform, every child is presented with an appealing and customized way of learning-according to the child's speed and mental capability, which leads to optimal results.
Another target audience includes parents and caregivers, as they are always concerned about following the school progress of their children. The Parental Dashboard will provide them with monitoring, setting goals in learning, and personalizing content to better align with the interests of the child. Since they have the ability to track and adjust the course of their child's education, parents will continue to be more involved in learning to reinforce ideas learned through the app and continue growth.
The Interactive Storytelling and Learning Application creates added value for educators and schools as well. This platform helps teachers to enhance literacy programs, differentiate lessons, and evaluate the performance of students. Integration into the classroom opens the door for a teacher to provide direct support to students at various levels of reading. Schools looking for a more innovative solution, using technology to

help drive up literacy rates and engagement, can integrate our platform into their curriculum for seamless adaptive learning.

1.7 Business Context

The market for personalized learning tools is expanding fast because parents and teachers are trying to find new and innovative means to improve children's reading. Present learning approaches are highly structured, and hence children lose motivation. Our project is unique since it integrates interactive storytelling and personalization using AI, meaning that every child gets a customized and interesting experience.
Built with scalability in mind, the app can expand into other languages, more subjects, and integration into school curricula. This versatility enables it to reach bilingual students, expand into math and science courses, and act as an interactive supplement to classroom instruction.
In order to keep up with the EdTech sector, we focus on frequent content updating and tech innovation. Frequent updates will add new interactive features and AI updates, rendering the platform interesting, effective, and relevant to contemporary educational demands.

1.8 Technologies

1.8.1 Artificial Intelligence (AI)

It should be contextually noted that AI helps keep tabs on topics for dynamic personalization. The possible machine-learning output includes varieties associated with the level of difficulty, vocabulary, and titles, modulated by user engagement to create individualized learning experiences. We leveraged Hugging Face to integrate pre-trained AI models, enhancing the story generation process with advanced natural language understanding.

1.8.2 Mobile Applications

The app is built with Flutter, which offers cross-platform usability on Android and iOS. This offers usability for kids, parents, and teachers across devices without sacrificing performance.

1.8.3 Cloud Computing & Data Storage

By making full use of cloud technologies such as Firebase and AWS, the upper hand for real-time data synchronization enables us to layer on secure data storage and scaling for two purposes. This facilitates seamless content delivery, reliable backup over time, and user-device use cases while preserving data integrity. Additionally, Flask was used as the backend

framework to facilitate AI-driven API interactions and streamline communication between the front-end and AI models.

1.8.4 Natural Language Processing (NLP)

NLP, as well, fuels an AI-based story engine so well that our application can perceive user input, establish a story context, and initiate interactive guidance. This will make sure that the interactive storytelling process is changeable and all kid-oriented to develop narratives based on a child's reading level. Our NLP implementation utilizes Hugging Face models to ensure high-quality language processing, allowing for better comprehension and adaptive storytelling responses.

1.8.5 Security & Compliance Technologies

The app uses AES encryption, secure authentication mechanisms, and role-based access control for security, which will shield users and their information from unauthorized use. Compliance with COPPA and GDPR will ensure protection against unauthorized access to children's information and ensure regulatory compliance.

1.9 Summary

The Interactive Storytelling and Learning Application is an innovative online learning platform that is revolutionizing the way children approach literacy. By integrating AI-driven personalization with interactive storytelling and gamification, the platform provides a customized learning experience that dynamically changes with each child's reading level, cognitive skill, and interests. As opposed to a typical learning tool with static and generalized approaches, this platform will make sure that every child goes through an educational journey that is tailored to his or her specific needs.

This application makes use of advanced AI capabilities: immediate feedback to constantly readjust story complexity and gamification for making this process even more thrilling, thus compelling children to undertake a very proactive role in the learning process. Parents and teachers will also see detailed progress tracking, where the achievements can be tracked down for improvement and tailored educational support given.

With a focus on accessibility and scalability, the platform supports multiple languages and cultural adaptations, making it a valuable tool for diverse global audiences. With increasing demand for personalization in the field of digital learning solutions, our application establishes itself as an interactive education leader, opening perspectives for further research and development regarding AI-driven literacy. Our project could set the path to a brighter future in early childhood literacy and the new world standard for

digital education by addressing the current limitations of education and offering a new, rich learning environment.

2. Feasibility

Feasibility in the Interactive Storytelling and Learning App is analyzed in multiple dimensions to determine viability: strategic alignment, market conditions, technical feasibility, financial consideration, and strategies for risk management. The proposed app is quite innovative for the EdTech industry, as it changes how children will learn with the inclusion of AI-driven adaptive storytelling, real-time vocabulary development, and interactive content in one solution.

2.1 Project Charter

The Project Charter defines the key aspects of the Interactive Storytelling and Learning App. This project has been undertaken with the aim of developing an adaptive, AI-powered educational application for improving literacy in children through storytelling in an interactive manner. The call for educational needs in modern times includes artificial intelligence, natural language processing, and customized learning experiences. This application will target a group of children within the age bracket of 4-10 years. The stories would be personalized and updated according to their learning stage, interest, and progress. The development team develops this project in order to complement the shortfalls of most existing educational apps, which are unsuccessful in personalizing the learning experiences.

Key stakeholders will include schools/colleges, parents, kids, content developers, AI engineers, and any regulatory bodies that are focused on the protection of child data privacy under COPPA and GDPR. The project will be developed within six months, having milestones for initial prototyping, user testing, and final deployment.

2.2 Scope Statement

This project will involve the design, deployment, and maintenance of an interactive application for storytelling. The application shall provide personalized learning in real time, using NLP and AI models to serve the development of vocabulary, reading skills, and cognitive engagements for every child. Dynamic story paths, interactive decision-making, real-time adaptability of content difficulty, and parental progress tracking are included in the scope of features.

It will be developed as a cross-platform solution, with the stated platforms being iOS and Android. Artificial Intelligence-driven personalization, vocabulary tracking, and interactive quizzes will be part of the learning process. The scope does not include direct classroom implementation but rather provides integration capabilities with existing LMS. Continuous updating and adding new content is part of the post-launch support.

2.3 Goals and Objectives

Through an AI-supported, interactive, and adaptive narration experience, the Interactive Storytelling and Learning App will be developed to promote literacy and mental growth in children without diluting high levels of interest. The project combines learning with fun for an interesting learning experience that is personalized for each and every child separately.

1. **Personalized AI-Driven Storytelling:**
   • The goal is to develop an AI-based storytelling engine that adapts to the reading level, comprehension level, and interests of each child in real time.
   • The objective is to ensure that every child has a unique learning experience that challenges and supports literacy skills.
2. **Interactive Learning Through Story Choices:**
   • The goal is to encourage optimum participation by enabling children to engage actively with stories by making choices that impact the story.
   • The objective is to promote critical thinking, creativity, and reading ability by making children feel accountable for their own learning process.
3. **Comprehensive Parental and Teacher Dashboard:**
   • The goal is to deploy a dashboard that offers teachers and parents accurate information on a child's reading growth, vocabulary, and overall learning achievements.
   • The objective is to allow parents and teachers to monitor progress, pinpoint areas of difficulty, and target their support appropriately to improve learning outcomes.
4. **Secure and Compliant User Experience:**
   • The goal is to be fully compliant with children's data protection laws like COPPA and GDPR so that the app is safe and secure for children to utilize.
   • The objective is to win the trust of parents and teachers by adopting rigorous security protocols to protect children's personal information.
5. **Gamification Elements for Sustained Engagement:**
   • The goal is to incorporate achievements, awards, and interactive involvement so that the children are motivated to stay engaged and actively involved with the platform.
   • The objective is to make learning appear rewarding and enjoyable, increasing the potential for literacy skill development in the long run.
6. **Scalability and Future Expansion:**
   • The goal is to create a flexible and extensible architecture so as to be capable of managing future extensions, such as multilingualism and other educational content.

- The objective is to provide the platform with the ability to adapt in favor of new learning techniques, larger audiences, and compatibility with curriculums for education.

The Interactive Learning and Storytelling App aims to deliver a rich, adaptive learning experience for kids and enable them to develop essential literacy skills while putting learning at the center as an enjoyable, interactive experience.

2.4 PESTEL Analysis

2.4.1 Political

The government itself has many policy regulations on what can or cannot be made. For instance, for an educational app to even be considered for schools, strict adherence to things like the Children's Online Privacy Protection Act and General Data Protection Regulation have very specific guidelines over data collected on children under the age of 13. There are areas around the world where government funding towards EdTech projects could serve either as partnerships or financing for their initiatives. Besides, economic growth is related to global political stability, thus affecting investment in educational technologies. Changes in policy regarding data security, education reforms, and digital learning strategies might also impact the viability of this project in the market.

2.4.2 Economic

The economic landscape influences the adoption of new learning technologies directly. Growing demand for digital education solutions, together with increased household expenditure on children's learning applications, provides a very strong market opportunity. Subscription-based revenue models, freemium services, and partnerships with schools offer potential monetization strategies. In any case, the affordability issue has to be taken into consideration in order for the app to remain accessible to a wide range of users. Inflation, global recessions, and shifts in investment in technology also have a bearing on the economic viability of this project.

2.4.3 Social

Parents and educators increasingly understand the value of digital learning tools. From traditional reading habits to AI-enhanced learning, the demand for more interactive and adaptive educational solutions is very high. Stories developed should consider cultural sensitivities with respect to inclusivity and relevance of user demographics. Moreover, societal trends of early literacy and technology-assisted education will also support the acceptance of this app.

2.4.4 Technological

The technologically feasible development of the entire project will hence be very significant. AI and NLP enable the application of personalized storytelling within applications to greatly

improve user experience. This development is completely technically viable with modern mobile processing, AI-enabled learning models, and real-time analytics. The fact that it promises to be cross-platform using the Flutter framework helps in this count. This, however, requires powerful back-end infrastructure with data stored and analyzed in real time; for example, by using AWS or Firebase.

2.4.5 Environmental

Digital learning applications can have a relatively low environmental impact compared to other traditional methods, which rely so much on paper-based resources. Promoting digital education via the app increases sustainability with reduced printouts. However, energy use from operation on servers does arise, where several strategies should be put into place for energy efficiency.

2.4.6 Legal

Above all, an application for kids needs to take special care with keeping in view all the legal aspects. Accordingly, following international directives like COPPA and GDPR but also taking into consideration the education standards imposed within the specific territories does not overstep the line set by legislation anywhere. Ensure copyrights for all materials: all content, images, and sound-overs have proper protection; on top, users should agree on terms and service and privacy with parents and tutors.

2.5 Competitive Analysis

2.5.1 Current Competitors

1. **Epic!**

   Epic! is a well-known digital reading platform providing a large collection of eBooks, audiobooks, and read-along books for children. Although it possesses a formal digital library, it lacks AI-powered adaptive storytelling and real-time personalization and hence is less interactive than the Interactive Storytelling and Learning App.

2. **Reading Eggs**

Reading Eggs is an interactive reading program that teaches children to read using pre-set lessons, phonics drills, and learning games. Its preset learning tracks restrict customizability, and it fails to adapt storylines dynamically according to a child's reading behavior and comprehension levels.

3. **Tales Untold**

   Tales Untold is an audio storytelling firm that provides professionally narrated, engaging stories for children. While it provides high-quality audio content, it does not have interactive features, AI-driven content personalization, or real-time feedback mechanisms that support a child's learning process.

These competitors offer helpful educational content, though, but without real-time AI personalization, adaptive learning, and strong user engagement features, which makes the Interactive Storytelling and Learning App distinct and innovative.

2.5.2 Future Competitors

With the development of AI and NLP technologies, more and more companies will join the personalized educational space. New startups and established EdTech firms will introduce competing products, leveraging similar AI-based adaptive learning techniques. The continuous development of AI-driven personalization is at once an opportunity and a challenge for staying ahead of market trends.

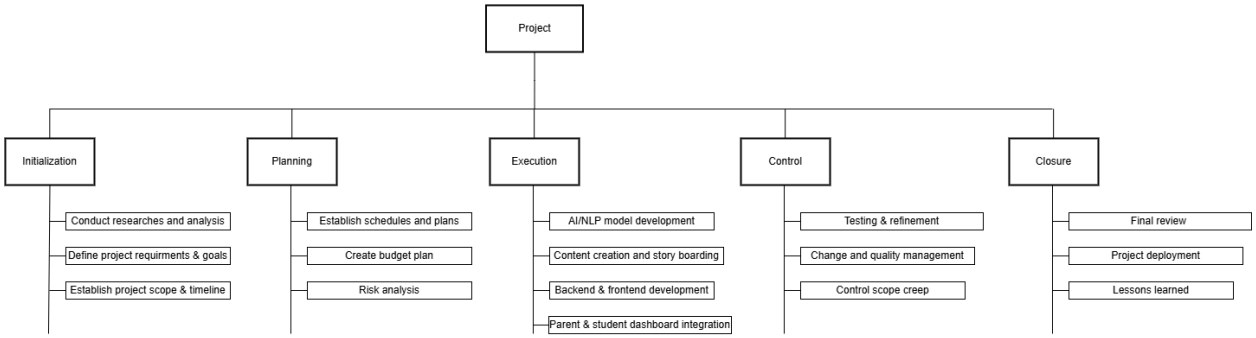
2.5.3 TechQuest's Competitive Advantage

TechQuest software is education technology since it contains a unique adaptive storytelling feature that individualizes content based on a child's reading level, comprehension ability, and interests. As opposed to other learning software with static content and minimal personalization, TechQuest individualizes stories to align with each child's unique learning trajectory so that they will stay engaged. Such flexibility provides enhanced comprehension, increased interest, and retention of literacy skills.

What sets TechQuest apart is the way it organically integrates gamification, progress monitoring, and parent reporting into one comprehensive platform to transform the learning process beyond conventional pedagogical approaches. Kids can be an integral part of crafting plots by storytelling as they advance, which makes them more astute in decision-making and critical thinking. The remaining aspects of gamification, such as achievements and rewards, are designed to increase motivation and participation so that more children form the good habit of reading and learning daily. Furthermore, the sophisticated progress tracking system provides detailed insights into children's reading frequency, words learned, and reading comprehension acquired, thereby enabling them to provide proper attention and guidance to students in conjunction with other caregivers and teachers.

Unlike most of the competitors that limit themselves to providing the content, TechQuest specializes in data-driven and rich personalized learning experiences. Blending adaptation and engagement with analytics and AI results not only in fun but also in guaranteed literacy gains through TechQuest's approach. This combination is what positions TechQuest at the forefront of the EdTech industry as the go-to solution that seamlessly combines entertainment and education in an efficient and pervasive way.

2.6 Work Breakdown Structure



2.7 Project Budget

2.7.1 Preliminary Schedule

	Month 1	Month 2	Month 3	Month 4	Month 5	Month 6	Month 7	Month 8	Rest of time

Requirements									
Specification									
Design									
Implementation									
Testing									
Deploy									
Maintenance									

2.7.2 Preliminary Budget

No	Item	Quantity	Price
1	Firebase premium	1	20.41
2	AWS Kinesis	1	87.63
3	AI gpt 2.0	1	43.28
4.	Samsung 11 inch android Tablet (rent)	1	148.00
5	C type cables	2	20.00
6	3D printing	1	300.00
		Total	619.32

2.8 Performance Measurement Baselines

The Performance Measurement Baseline played a crucial role in tracking project progress, measuring performance, and validating deliverables on schedule and to quality requirements. To remain responsible and in motion, the team developed formal task tracking, coordinating meetings, and routine communication protocols. The protocols allowed milestones to be achieved within the cost while making it easier to make changes when necessary throughout the project.

• Project tasks were divided into different components based on the area of expertise of each member and assigned accordingly. Every member documented his progress weekly in a shared document, making it transparent regarding the task status and allowing on-time adjustments if required. Deadlines were set to ensure timely delivery and prevent delays.

• Every other week Google Meet calls were planned for virtual check-ins where members discussed what was done, exchanged progress in process, and smoothed out problems. The flexible approach enabled anyone anywhere in the world to contribute and kept everyone engaged. Summary calls were recorded in a shared diary for keeping track and creating no doubt on tasks to be done.

• There were weekly face-to-face meetings at the university during class hours or breaks to allow for hands-on collaboration. The meetings consisted of checking the project goals, resolving complex tasks, debugging, and planning the next phases of development.

• WhatsApp daily updates enabled quick communication, with members of the team sharing updates, inquiring, and receiving immediate feedback. The real-time interaction helped in resolving small issues efficiently without having to wait for planned meetings.

• All face-to-face and virtual meetings were documented in the team diary, such as attendance lists, agendas, topics discussed, and action plans. Formal documentation ensured transparency, accountability, and a total record of the progress made with the project.

Through maintaining such a systematic procedure, the project could monitor performance regularly, ensuring constant monitoring of AI responsiveness, adjusting story accuracy, application uptime, and user satisfaction. Comparisons of monthly reports of actual progress against desired objectives allowed the team to see variances and take corrective action as needed. With systematic performance measurement, the project stayed on track, ensuring quality, efficiency, and adherence to cost limitations.

2.9 Milestones and Associated Dates

Task	**Completion**
Semester 1	
Project Initiation	

Market Analysis	Week 1
Feature Definition	Week 2
Define roles	Week 2
Establish project scope	Week 3
Project plan development	Week 4
Approve project plan	Week 5
Requirements & System Design	
Requirement Finalization	Week 6
Select app name	Week 6
Finalize function requirements	Week 7
Finalize non - functional requirements	Week 7
Develop user interface diagrams	Week 8
Define prerequisites	Week 6
Develop system architecture and data flow diagrams	Week 9
Create UI/UX wireframes	Week 10
Compile summary	Week 11
Risk analysis	Week 11
Semester 2	
Backend & Database Development	
Set up firebase authentication & user management	Week 1-4
Implement database structure in firebase	Week 1-4
Develop API services	Week 1-4

Implement real-time data tracking & syncing	Week 5
AI & Adaptive Learning Module	
Develop AI-based storytelling engine	Week 6-8
Implement adaptive difficulty algorithm	Week 6-8
Integrate NLP for student interactions	Week 6-8
Frontend Module	
App home page and initial design	Week 6-8
Develop student dashboard	Week 6-8
Develop teacher dashboard	Week 6-8
Develop interactive quizzes	Week 6-8
Testing & Deployment	
Review test cases	Week 8
Complete documentations	Week 8
Conduct front end tests	Week 9-10
Conduct back end tests	Week 9-10
Deploy sample version	Week 9-10
Asses the beta version	Week 9-10
Deploy final app	Week 9-10
Bug fixes and monitoring	Week 9-10

2.10 Staffing

The Storytelling App was coded by a group of committed developers with specific roles. Front-end coding and AI integration were spearheaded by Hassan Barada and Abbas Amir, maintaining an elegant user experience and AI-generated storytelling optimisation. Abdulraheem Daoud coded the back-end and AI processing, maintaining optimal data handling and system intelligence. Ajmal Abdu Razak coded the back-end, handling database transactions and server logic. Saleh Abu Zuhri was directly involved in the front-end, web development, and project planning and contributed to the application's

interface as well as a productive plan to attain the highest potential of the project.

The team created a solid, easy-to-use app as a team with customized, AI-written stories and allows teachers to monitor student work and activity in an easy way.

2.11 Division of work among team members

Project Name: Storytelling Application
Team Members: Hassan Barada, Abbas Amir, Abdulraheem Daoud, Ajmal Abdu Razak, Saleh Abu Zuhri

1. Hassan Barada - Front-End and UI Desgin

User Interface & Experience:

- Developed an intuitive and accessible UI for students and teachers.
- Ensured seamless navigation and responsive design across devices.

AI Storytelling Implementation:

- Integrated AI-driven story generation features, ensuring personalized content for students.
- Optimized AI responses for relevant and engaging storytelling experiences.

Testing and Debugging:

- Conducted rigorous front-end testing to ensure smooth functionality.
- Identified and resolved UI/UX inconsistencies to enhance user experience.

2. Abbas Amir - Back-End and AI Integration

Back-End and Database:

- Designed and implemented robust back-end architecture for the YourStory app, ensuring seamless communication between front-end and back-end components.
- Integrated Firebase Firestore for efficient data storage, structuring stories under different collections.

Performance & Security Enhancements:

- Ensured secure handling of user data with Firebase authentication and access control measures.

- Optimized database queries and API responses to handle requests efficiently,

improving app performance.

Testing and Debugging:

- Identified and resolved UI/UX inconsistencies to enhance user experience.

3. Abdulraheem Daoud - Back-End and AI Processing

API and Database Management:

- Developed and managed APIs for communication between front-end and back-end systems.
- Ensured data storage and retrieval in Firebase for seamless story access.

AI Model Optimization:

- Implemented and fine-tuned AI models to generate contextually appropriate stories.
- Enhanced AI processing for faster and more relevant content delivery.

System Security and Performance:

- Ensured secure handling of user data and compliance with best security practices.
- Optimized system performance to handle multiple concurrent story generations.

4. Ajmal Abdu Razak - Back-End Development

Server-Side Development:

- Managed the back-end logic and infrastructure for data processing.

Integration with AWS:

- Connected AWS for real-time data tracking.
- Ensured seamless logging of student engagement and progress data.

5. Saleh Abu Zuhri - Front-End, Web Development, and Planning

UI & Web Development:

- Assisted in front-end development to refine the user interface.
- Developed essential web components such as informational pages and admin

controls.

Project Planning and Coordination:

- Managed development timelines to ensure the project met key milestones.
- Facilitated team collaboration and strategic planning for project success.

System Testing & Finalization:

- Conducted final usability and functionality tests before deployment.
- Coordinated team efforts to enhance overall system efficiency and usability.

The structured division of work ensured that all aspects of the Storytelling Application were developed efficiently, with a focus on AI-driven storytelling, seamless user interaction, and robust back-end support.

2.12 SWOT Analysis

Team SWOT

Strengths:

- Developed an adaptive storytelling system with strong AI and NLP integration. competitors in the EdTech space.
- Child-friendly UI/UX focuses on younger learners to increase usability and retention.
- Concentration on highly interactive and personalized app learning sets it apart from
- Insightful and supportive real-time analytics along with teacher interaction for parents and educators.
- Increased platform accessibility and impact with scalability for diverse languages and learning levels.

Weaknesses:

- There are certain functionalities dependent on the external API providers which may delay the process because of the integration-related difficulties.
- Limited and not having an in-house team for large-scale testing would mean relying on external testers and educators in the future to do the evaluation comprehensively.

Opportunities:

- There is a rise of AI-based learning applications, opening up their opportunities for large-scale adoption and market expansion.
- Increased acceptance of personalized learning products in schools and families

promotes the adoption of such products.
- Room is left for entering into strategic partnerships with schools and educational institutions to secure better credibility and reach.

Threats:

- Advancements in technology are so rapid that they put you in the position of needing to update and upgrade your structures constantly to keep up with the competition.
- Changing compliance requirements regarding children's privacy can create a need to constantly adjust data security policies.
- More established EdTech competitors whose resource bases are richer may raise their game and develop similar AI-influenced education apps resulting in increased competition in the market for such education apps.

The increasing demand for AI-powered educational applications reflects a favorable market and opens up opportunities for the team. Growing adoption of customized learning solutions in schools and homes will support the long-term viability of the project. On the other hand, rapid changes in technology and changes in compliance regarding children's data privacy may emerge as threats since these demand constant upgrading and adherence to regulations.

Product SWOT

Strengths:

- Real-time changes of stories are powered by AI, conforming to user interaction and learning achievements.
- Designed to provide an immersive and interactive user interface for young students to foster maximum retention and interaction.
- According to modern learning paradigms to assure the applicability and effectiveness in a formal construct of learning.
- Robust parental control and analytics features enable parents, caregivers, and educators to oversee learning progress.

Weaknesses:

- Dependence on a stable internet connection for optimal AI performance that may limit access in low-connectivity regions.
- AI computational resource heaviness that may necessitate device compatibility considerations.

Opportunities:

- Expansion into multilingual narrative to make the app available to non-English-speaking consumers.
- Possible inclusion in school curricula and educational institution adoption as

another learning tool.
- Possible partnership with publishers and content creators to expand the library of interactive narratives.

Threats:

- Competition from established EdTech players with larger market presence and resources.
- Resistance from traditional teachers who may resist AI-based learning approaches.
- Privacy and security concerns on how the information of children is processed and stored, which requires stiff compliance measures. Compared to stationary e-books or traditional storytelling apps, the application provides an experiential feature in which stories continually evolve through interaction with, and learning markers from, the child. An area of vulnerability can be relying on a good internet connection to provide optimal performance of AI features, which can limit accessibility where the connectivity may not be effective.

These opportunities include reaching multilingual story-telling possibilities, hence opening the app for users who may not speak or understand English. Besides, huge integrations and adoptions across schools and educational facilities could ensue. Its major threats could be the competitors from established EdTech companies by launching similar AI-driven learning apps in the times to come.

2.13 Risk Management Plan

Risk Levels	Description
High	Risks that significantly impact the schedule or functionality and require immediate mitigation.
Medium	Risks that allow progress but with limited solutions and potential schedule impacts.
Low	Risks with minimal impact on the schedule; alternate

	solutions are available.

Risk Levels

Sr. No.	Risks	Why is it a Potential Risk?	Risk Level	Possible Solution
1	AI Implementation Complexity	Implementing AI-driven adaptive storytelling requires fine-tuning Natural Language Processing (NLP) models for dynamic content generation.	High	Break down AI tasks into manageable components, utilize existing NLP frameworks, and conduct iterative testing.
2	Story Adaptation Accuracy	Ensuring that AI-generated stories are age-appropriate, coherent, and engaging for different reading levels.	High	Train the AI on diverse datasets, conduct real-time testing with children, and implement strict content validation rules.

3	Integration of AI and Database	Synchronizing real-time user progress with AI-driven adaptations may cause performance bottlenecks.	High	Optimize database queries, use caching techniques, and conduct stress testing for high loads.
4	Security and Compliance Challenges	Meeting COPPA and GDPR regulations for handling children's data is mandatory.	High	Conduct security audits regularly, implement end-to-end encryption, and enforce role-based access controls.
5	User Engagement and Retention	Lack of interactive features may lead to reduced user engagement over time.	High	Introduce gamification, personalized learning paths, and rewards to sustain interest.
6	Scalability Issues	Increasing the number of users may strain the current system architecture.	High	Design a scalable cloud-based backend, implement load balancing, and optimize data storage.
7	Communication Gaps Within the Team	Misalignment in development efforts could lead to inefficiencies and delays.	High	Hold regular team meetings, use centralized documentation, and establish clear project timelines.

8	Delays in Feature Integration	Ensuring that all app modules (AI, UI, database) work seamlessly together requires extensive testing.	High	Begin integration testing early and use a modular development approach to detect issues quickly.
9	Dependency on AI Expertise	Relying on one AI specialist could slow down progress if they are unavailable.	Medium	Encourage knowledge sharing within the team and provide AI training for other developers.
10	User Interface Complexity	Designing an intuitive, child-friendly UI that is both engaging and simple is resource-intensive.	Medium	Conduct user testing sessions with children and iterate UI designs based on feedback.
11	Incomplete Documentation	Missing or outdated technical documentation may delay troubleshooting and updates.	Medium	Assign team members to update documentation consistently and track changes using version control.
12	Mobile Application Performance	Potential lag or crashes on lower-end devices could affect user experience.	Low	Optimize resource usage, implement lightweight assets, and conduct performance testing across multiple devices.

2.14 Project Matter

2.14.1 Open Issues

Integration Complexity:

The most important task is to configure all the elements of the Interactive Storytelling and Learning App, that is, the mobile app, its AI learning engine, and the real-time database. Smooth communication among the components is vital to ensuring data uniformity, content personalization for every different user, and real-time performance. Absence of such communication, however, results in a compromise on quality, therefore leading to some social hassle during and after the learning process.

Data Security Concerns:

Having high data security is critical for the app as it handles sensitive information like user accounts of children and their learning activities. The threats are unauthorized access, data leakage, or breach of regulation. The app will need to secure encrypted data storage, secure transmission methods, and strict adherence to regulations such as COPPA and GDPR for the safety and trust of its users.

Limited Scalability in the Early Phase:

In the initial stages, the application will support a smaller set of users for resource limitations. As there's more adoption, the system may be scaled to support additional users. Increasing system scale would require more computational capabilities, additional storage, and an enhanced cloud base. Lack of scalability planning might result in poor performance and delayed response time with the addition of users to the load.

Hardware Limitations:

Some elements of the system, such as real-time interaction and AI processing, may require significant computing power. In the early phases of development, performance limitations on low-end devices may affect accessibility. It will be necessary to deliver a smooth experience on a variety of devices, from low-cost smartphones to inexpensive tablets, in order to maintain usability and inclusivity.

Data Overload:

The application is meant to cumulatively process a lot of learning data such as user activity and immediate feedback, and levels. Too much data, unless filtered and given importance, would

block the system and cause latency in providing suggestions or inappropriateness of suggested content. Therefore, efficient data management strategies such as optimized storage facilities and artificial intelligence-optimized data processing will be necessary.

Device Compatibility Challenges:

To ensure compatibility among various devices, operating systems, and screen sizes continues to be a significant challenge. Consequently, the performance and usability of the apps can be affected by variations in display resolution, processing capabilities, and OS versions. Following a thorough test plan will deliver a consistent experience between platforms.

2.14.2 Future Enhancements

Integration with Adaptive Learning Models:

Future enhancements could include advanced adaptive learning models driven by AI, permitting further analysis of user activity within the application and modification of learning paths in real-time. This would prove to be a more engaging situation by presenting content shaped around individual strengths and weaknesses of the child.

Emotional Intelligence Recognition:

Through AI-based emotion recognition, the app can detect children's frustration, confusion, and excitement based on facial expression and indications including voice tone. The system would, in turn, adjust difficulty levels, offer encouragement, or give hints accordingly.

Offline Learning Mode:

Future versions of the application would include an offline mode in which children could read pre-downloaded educational content and interactive books even when there was no internet access. This would serve to be of utmost help for subscribers located in bad-connectivity areas.

Integration with Educational Institutions:

The app could be upscaled for integration with teachers and schools where teachers can track student progress and provide tailored learning paths and give instant feedback. This would connect school and home learning and position the app as a very valuable blended learning tool.

Multilingual Support and Accessibility Features:

There would be the potential for the app to benefit children from different backgrounds by establishing multilingual features. Furthermore, text-to-speech, dyslexic fonts, and voice commands will make learning accessible to all.

3. Requirements

3.1 Introduction

3.1.1 Overview

This part highlights some of the key features that should be considered when designing an inclusive and effortless storytelling interactive platform. The system will enable the user to interactively participate in the learning process by means of AI powered adaptive narratives, language processing, and gamification. Primary attributes are interactivity in storytelling, vocabulary monitoring, content personalization, and detailed progress reports for parents and teachers. Therefore, the requirements focuses on fast coping and safe data privacy and mobility with different devices, whereby the application's functions are fully operational in various environments.

3.1.2 Definitions

For clarity, the following definitions are used throughout this document:

- **AI-Powered Adaptation:** The feature of the application to shift automatically storytelling parameters based on user input and level of learning.
- **Natural Language Processing (NLP):** The technological breakthrough behind the capability of the system to read and generate natural language replies to interactive narrative.
- **Dynamic Difficulty Adjustment (DDA):** A system that has the capability to dynamically modify complexity levels of content so that it is never too simple nor too complicated for the user.
- **Gamification:** Application of reward schemes, achievements, and interactive elements in an attempt to keep the user engaged and stimulated.

3.2 Frameworks

3.2.1 Machine Intelligence

This program utilizes sophisticated machine intelligence frameworks with adaptive learning capabilities. AI libraries such as TensorFlow or PyTorch are comparable to any other program for developing and training natural language processing models for interactive storytelling content. They are selected for their ability to scale, the flexibility to implement relevant changes, and performance with complex neural networks. The machine intelligence features continuously learn from any user interaction to continually refine the narratives of each child based on their interests and their emerging language proficiency.

3.3 Design Constraints

Several constraints influenced the design of the Interactive Storytelling and Learning App, including:

- **Data Privacy and Security:** The application was provisioned with encryption and access control mechanisms since it harbors sensitive user data regarding children's learning progress and profiles. Both COPPA and GDPR compliance were ensured for the continued protection of data and user privacy.

- **Cross-Platform Compatibility:** The application was designed to operate seamlessly on iOS and Android with respect to the chosen development frameworks. Most importantly, it works well, regardless of what operating system you're using, promising the best experience for end-users.

- **Performance Optimization:** This made it possible to fully optimize devices with low processing power and memory, such as budget smartphones and tablets, so the application would consist of skinny assets, efficient code, and optimized resource management in order to keep at all times responsive.

- **Budgetary Constraints:** The first development phase concentrated on the core functionality since the finances were limited. Although enhancements to storytelling via advanced AI were considered, they can be developed in a future update.

These constraints were carefully managed during development to ensure that the Interactive Storytelling and Learning App remains accessible, efficient, and high-performing for its target users.

3.4 Functional Requirements

3.4.1 AI-Driven Interactive Storytelling

Function	**AI-Driven Interactive Storytelling**

Description	**Generates dynamic, personalized story experiences based on user preferences and reading levels.**
Inputs	**User-selected themes, difficulty level, and past interactions.**
Source	**User input, AI adaptation engine, and pre-trained language models.**
Outputs	**Adaptive story with branching choices tailored to the user.**
Destination	**Displayed on the user interface and stored for future reference.**
Action	**Generate and display an interactive story while adapting responses in real-time.**
Requires	**AI processing unit, content database, and NLP engine.**
Pre-condition	**The user initiates a storytelling session and selects preferences.**
Post-condition	**Story progression is logged, and user choices are stored for learning adaptation.**
Side-effects	**Potential variance in generated content accuracy.**

3.4.2 Adaptive Learning and Progress Tracking

Function	**Adaptive Learning and Progress Tracking**
Description	**Monitors user engagement and progress, dynamically adjusting difficulty levels.**
Inputs	**User interactions, completion status, quiz results.**
Source	**User activity logs, quiz assessments, AI learning model.**
Outputs	**Performance reports, recommended adjustments, user feedback.**
Destination	**User progress dashboard, learning model database.**
Action	**Analyze engagement metrics and adapt content accordingly.**
Requires	**AI engine, data analytics module, Firebase/AWS storage.**
Pre-condition	**The user participates in learning activities.**
Post-condition	**Updated learning path stored, recommendations generated.**

Side-effects	**Data inconsistencies due to network interruptions.**

3.4.3 User Engagement and Gamification

Function	**User Engagement and Gamification**
Description	**Enhances the learning experience using gamified elements such as rewards, challenges, and leaderboards.**
Inputs	**User progress, activity levels, achievement milestones.**
Source	**Application engagement metrics, reward system database.**
Outputs	**Badges, achievements, and leaderboard rankings.**
Destination	**User profile, UI notifications.**
Action	**Reward users based on progress and encourage continued learning.**
Requires	**Gamification module, real-time data tracking.**

Pre-condition	**The user actively participates in learning sessions.**
Post-condition	**Achievements updated, new challenges unlocked.**
Side-effects	**Potential for user over-reliance on rewards rather than intrinsic learning.**

3.4.4 Parental and Educator Dashboard

Function	**Parental and Educator Dashboard**
Description	**Provides insights into user progress, strengths, and areas needing improvement.**
Inputs	**User engagement data, performance metrics, behavioral patterns.**
Source	**Database logs, AI analysis, user activity tracking.**

Outputs	**Comprehensive reports, real-time monitoring updates.**
Destination	**Admin dashboard, parental access module.**
Action	**Display real-time progress analytics and learning behavior trends.**
Requires	**Secure login system, AI-driven reporting tools.**
Pre-condition	**User activity data is logged and accessible.**
Post-condition	**Progress reports were generated and displayed.**
Side-effects	**Data privacy concerns requiring strict security measures.**

3.4.5 Security and Data Privacy

Function	**Security and Data Privacy**

Description	**Ensures user data is protected through encryption, authentication, and secure cloud storage.**
Inputs	**User credentials, activity logs, personal data.**
Source	**Authentication servers, encrypted storage, Firebase.**
Outputs	**Secure access, encrypted data storage, authentication logs.**
Destination	**User profile management, server database.**
Action	**Encrypt sensitive data, manage secure access, and prevent unauthorized breaches.**
Requires	**AES encryption, secure authentication, and access control measures.**
Pre-condition	**User authentication and data access request.**
Post-condition	**Data securely stored and accessed by authorized entities only.**
Side-effects	**Potential access delays due to authentication processes.**

3.4.6 Multi-Device Compatibility

Function	**Multi-Device Compatibility**
Description	**Ensures seamless access across various devices, including tablets and smartphones.**
Inputs	**Device type, screen resolution, OS version.**
Source	**User device specifications, app compatibility module.**
Outputs	**Optimized UI/UX, adaptive layout rendering.**
Destination	**User interface across multiple platforms.**
Action	**Adjust the interface dynamically based on device specifications.**
Requires	**Responsive design framework, cross-platform compatibility engine.**
Pre-condition	**The user accesses the application from a supported device.**
Post-condition	**Content is properly displayed with full functionality.**

Side-effects	**Minor UI discrepancies across devices.**

3.5 Non-Functional Requirements

3.5.1 Performance Requirements

For a seamless user experience, the system must ensure real-time responsiveness with low latency. It should be able to manage several sessions running at once without experiencing any performance issues. High system uptime, effective resource usage across devices, and quick data processing for AI adaptation are examples of key performance metrics. These needs will be met by implementing hardware and software optimizations.

3.5.2 Security Requirements

Given the delicate nature of the data gathered from minors, security is an essential requirement. Strong encryption protocols must be used by the application for both data transfer and storage, secure authentication must be put in place, and international data privacy laws like COPPA and GDPR must be closely followed. To guarantee continuous defense against possible attacks, regular security audits and vulnerability assessments will be carried out.

3.5.3 Reliability and Availability

High availability and dependability are crucial, especially for systems utilized in educational environments. To guarantee ongoing operation even in the case of hardware or software failures, the application needs to have redundancy and failover capabilities. To maintain dependability, system updates and routine maintenance will be planned, and extensive monitoring tools will be implemented to quickly identify and resolve problems.

3.5.4 Compatibility Requirements

The Interactive Storytelling and Learning App needs to work flawlessly on a range of mobile devices, browsers, and operating systems. The user interface must continue to be effective and consistent across platforms. To ensure that every feature functions as intended across all devices and operating systems, extensive compatibility testing will be carried out.

3.5.5 Usability and Accessibility Requirements

The user interface needs to be simple to use and easy to access, especially for the younger demographic. To meet the requirements and abilities of many users, this involves the use of

straightforward navigation, unambiguous visual cues, and adaptable display options. To make sure the system is simple to use for kids, teachers, and parents alike, the design will adhere to accepted usability standards and take user testing results into account.

3.5.6 Scalability Requirements

Future expansion must be considered when designing the system. Without sacrificing performance, its architecture ought to accommodate more features and a growing user base. Cloud-based services will be utilized to increase processing and storage capacity as required, and modular design concepts will be used to facilitate the simple incorporation of new features.

3.6 Project matters

3.6.1 Open Issues

3.6.1.1 Potential Problems

Integration Complexity

The story app has components like AWS for activity logging, Firebase for data storage, and AI-based story generation. Functionality, user experience, and maintenance have to ensure a seamless real-time flow of data between these platforms. Currently, latency or any API connection issue is sure to disrupt the accessibility of students to their generated stories. Error handling and sync with automated retries will minimize the above risks.

Data Storage and Management

It is a data-generating and storing platform with user engagements, monitored history by student readers, and stories that an AI generates. Accumulation of these datasets with time slows down retrieval and increases data storage costs. An improper data structure under Firebase can make the system inefficient. Data compression techniques and techniques of scheduling regular cleanups and keeping unused user data will ensure system efficiency.

AI Story Generation Accuracy

Accordingly, AI would produce such stories that are personalized to reading level and interests of students. Wrong suggestions from AI would create stories that may become unnecessarily complicated or may be uncategorized; thus, a large extent of user engagement is minimal. In order to put maximum quality in the contents, upgradation and updates of AI must continuously be done with machine learning improvements, varying difficulty levels, and real-time feedback.

System Performance and Load Handling

The program should handle a number of students requesting stories in real time. Story generation will thus take longer because response time would be increased under high load traffic within the system. Performance problems will render students and instructors irritated as these will be those drastically affecting user experience. Avoiding performance bottlenecks,

scalable cloud infrastructure, server-side optimizations, and load testing will be required.

User Engagement and Motivation

Engaging students on a storytelling platform is quite a task and the easiest part that will cause falling off students is when the stories become very bland and not interactive. There may also be motivation-related features lacking for serious long-term engagement, such as rewards or tracking of progress. Thus, long-term engagement will also require possible changes, especially like gamification elements, student-driven story decisions, and interactive stories.

Security and Privacy Compliance

Rigorous adherence to security and privacy laws will be obligated since the platform holds student personal-related information. Else, dire consequences will follow for illegal unauthorized access, data loss, and non-compliance to laws such as COPPA or GDPR. Access control procedures, secured authentication mechanisms, and encrypted storage of information would, therefore, be imperative to keep such student data and avoid the violations.

3.6.1.2 Future Enhancements

AI-Powered Personalization

Higher-end AI personalization can be integrated as part of future updates, where the system tailors story suggestions by analyzing student reading habits, level of interaction, and past experiences. Depending on the individual learning style of each student, the AI might adjust the subjects of the stories and the difficulty level in real time, enhancing the experience's effectiveness and worth for the students.

Offline Access to Stories

Including an offline mode will allow students to read AI-generated stories offline. With the caching of previously generated stories stored locally in their device, students can continue their reading session despite weak connectivity. This will increase accessibility, particularly for those with mobile or intermittent internet connections.

Gamification and Interactive Elements

Gamification features like leaderboard ranking, reward based on achievements, and reward based on progress will be incorporated to promote user engagement. Users will also be able to shape the story through decision-making and choice, which will lead to higher engagement and investment.

Improved Multilingual Support

The stories are generated mainly in a single language in the present approach. There can be future research work based on integrating multilingual capability that would enable students to read and compose stories in many languages. For second-language learners or bilingual students, this facility would be very helpful in reinforcing their reading and understanding abilities.

Enhanced Teacher Dashboard and Data Analysis

Adding more advanced analytics to the teacher dashboard would give teachers more accurate information about the reading activity and performance of their students. Teachers can hopefully identify children in need of additional help using more visualization tools like trend graphs and relative performance analysis. Personalized learning approaches would be further improved through automated progress reports and reading materials recommendations based on performance metrics.

Collaborative Storytelling Features

A group storytelling mode in which several students can take turns at contributing to a single story would be an enhancement for the future. With voting on plot developments and alternating composition of various points of the story, this mode would promote teamwork and imagination. This interactive method may be facilitated by a more dynamic and collaborative learning setting.

Integration with Learning Management Systems

Subsequent updates may introduce it to integrate Google Classroom and Microsoft Teams with other learning systems that are well-liked. Teachers may offer students customized reading assignments based on AI-generated observations, while tracking students' progress in the same platform and delivering AI-generated reading exercises directly via their LMS.

4. Security

The Interactive Storytelling and Learning App's extensive security features are covered in this section. Maintaining confidentiality, integrity, and availability across all system components is crucial since the program handles sensitive user data, such as children's learning progress and personal information. The technological and administrative steps taken to protect data and guarantee dependable service delivery are described in depth in the ensuing subsections.

4.1 Security and Data Privacy

Security and data privacy are the most critical concerns when developing apps that store personal data, even more so when it is about children. This is an outline of the security and privacy measures taken to ensure users' sensitive data is properly secured.

Security measures:

1. **Data encryption**

All sensitive data—whether in the form of storage on servers or in transit over networks—will be encrypted. Advanced encryption standard (AES-256) wil be used to encrypt the data at rest and Transport layer security will be used to ensure secure communication between the client and the server. This ensures any information that the system shares and users, i.e., student scores or student progress, is saved in an encrypted state and is unreadable to the incorrect hands.

2. **Authentication protocols**

OAuth 2.0 and Firebase Authentication are being used for safe user authentication. Passwords will be hashed using bcrypt, which provides great defense against brute-force attacks. Users will also have the option of logging in with a Class ID or email. This flexibility is essential for young users who might find Class IDs more convenient.

3. **Role-based access control (RBAC)**

The system utilizes RBAC in order to allow various users only access to that part of the system that fits their role. The system boasts three main roles each for different users. The first role as a student allows the user to access stories, quizzes and track their progress. Second role as a teacher allows the user to monitor student progress, view reports and generate stories. Finally the third role as the admin gives full access to the system, allows user management and configuration.

4. **Data Anonymization**

Anonymization of data helps to comply with regulatory standards like COPPA (Children's Online Privacy Protection Act) and GDPR (General Data Protection Regulation). No personally identifiable information (PII) is kept or utilized for analysis other than absolutely essential and agreed by parents or guardians.

5. **Regular Audits and Security training**

To identify potential vulnerabilities and mitigate threats, penetration testing and regular security audits are implemented for robust security. It will help make the app secure from emerging security threats. The internal teams will also review security logs to achieve any unusual usage, making sure all threats are identified in a timely manner.

Data Privacy Measures:

1. **Data Minimization**

Only necessary data will be collected from students, teachers, and parents. The app only collects educational based data like story reading, quiz results , story theme with the prior authorization from the user or parent and avoids collecting any other data other than the data authorized by the user or parent. Storing excess data can always lead to potential data leakage or compromise, thus it is important to discard or destroy sensitive data like student personal information after the retention period.

2. **Data Retention and Deletion**

After a student departs from the system or if the retention period is established then the students personal information will be discarded or erased from the system. The parents and the students can, if that's what they desire, request data erasure according to GDPR and COPPA regulation.

3. **User Consent**

The system will obtain express consent from parents (or guardians) before collecting any data from children. Users will have the option to opt-out of non-essential data collection, such as marketing preferences.

4.2 User Authentication and Access Control

User authentication is the process through which the system checks that a user is indeed who they claim to be. Access control, on the other hand, defines what any particular user can do. Strong user authentication and access control use the following methods:

Authentication Mechanisms:

1. **Password-Based Authentication:**

The system utilizes Firebase Authentication for authentication that is easy, secure, and scalable. The passwords are stored in the Firebase system encrypted using bcrypt so that they are protected. The users are prompted to enter strong passwords, which must meet security requirements (e.g., minimum length, capital letters, numbers, special characters).

2. **Class ID Authentication for Students:**

Instead of authenticating by email, pupils can opt to log in using a Class ID, a unique identification number assigned to all of them by the system. It is another option for little children who might find the Class ID login less confusing. The system verifies the Class ID against stored records in the database.

3. **Multi-Factor Authentication (MFA):**

Multi-factor (MFA) authentication is required for higher level roles like Teachers and Admin. An authentication code will be sent to their email or phone number to provide a second form of identification which adds an additional layer of security.

4. **Single Sign-On (SSO):**

SSO is supported in the app if external services like Google and Microsoft are used for

education purposes. This is an option by which the user can authenticate using their current education accounts without creating separate credentials.

Access Control Mechanisms:

1. **Role-Based Access Control (RBAC):**

Resource access is controlled by RBAC. Users are assigned to a role, and every role possesses specified permissions. For example:

Students are granted access to their own progress and can interact with the stories.

Teachers have access to student progress, can view grades, and manage classroom settings.

Admins possess maximum access, with full control of users, content, and system options.

This keeps users from accessing parts of the system that are not part of their role.

2. **Access Control Lists (ACLs):**

ACLs enable user permissions on specific resources within the system to be managed. For instance, students can view their own quiz attempts and progress, while teachers are given permission to view and manage all student data for their class.

3. **Session Management:**

Sessions will be managed securely, such as session timeouts for inactive users. Users will be logged out if they have been inactive for a while to avoid unauthorized use in the event the user leaves the system unattended.

4.2.1 Potential Vulnerabilities and Solutions

Given the complexity of the application and the sensitive nature of the data, various vulnerabilities could pose risks. Below is a table of potential vulnerabilities and corresponding solutions to mitigate these risks:

Vulnerability	Risk Level	Solution
Weak Passwords	High	Enforce strong password policies (e.g., minimum length, complexity). Use bcrypt for hashing.
Brute Force Attacks	HIgh	Implement rate-limiting and CAPTCHA after several failed login attempts.
Session Hijacking	Medium	Secure cookies with HttpOnly, use TLS for all connections.
SQL Injection	High	Since Firebase Firestore is used (NoSQL), SQL injection is mitigated by design. For other databases, use parameterized queries and input validation.
Cross-Site Scripting (XSS)	Medium	Sanitize and escape all user inputs before rendering them in the browser. Implement Content Security Policy (CSP).
Cross-Site Request Forgery (CSRF)	Medium	CSRF tokens and SameSite cookies to protect users from CSRF attacks.
Weak Encryption of Data at Rest	High	Deploy strong AES-256 encryption on every piece of data that is being stored such that sensitive user data cannot be accessed even by malicious users.
Data Breaches due to Third-Party APIs	High	Use only trusted third-party services with strong GDPR/COPPA compliance. Ensure APIs use OAuth 2.0 for secure access.

4.3 Data Privacy Risks

As the system deals with sensitive data, like educational records of children, data privacy must be handled with extreme care. The following are the key data privacy threats:

Unauthorized Access to Data:

Unauthorized access can be achieved if proper security features are not implemented. External hackers or insider threats can expose sensitive data. Encrypting data both at rest and at transit prevents these attacks. Also implementing access control like RBAC (Role-based access control) ensures that only the authorized users are given access.

Data Retention and Usage

There exists a risk of data being retained for more time than it should, or misused by third parties.

To mitigate this threat, policies for data retention will be included in the system that ensure that data is never stored indefinitely. Users (and parents) will be notified about data usage, and consent will be requested.

Third-Party API and Service Risks

When third-party services are employed, there is a possibility of information being passed on or exposed inadvertently due to bugs in outer services.

To mitigate this threat, only use GDPR-approved services and ensure that external APIs possess robust privacy policies.

4.4 Risk Assessment Table

The Risk Assessment Table evaluates and mitigates the potential security risks identified above. The table below provides a systematic approach to address these risks:

Risk	Likelihood	Impact	Mitigation Strategy
Data Breach	High	Critical	Encrypt data, apply strict access controls, use Firestore (NoSQL).

Unauthorized Access	Medium	High	Implement multi-factor authentication, enforce RBAC, session management.
Brute-Force Attacks	Medium	Medium	Implement rate-limiting, CAPTCHA after failed login attempts.
Third-Party API Failure	Low	Medium	Use failover mechanisms, ensure API providers meet compliance standards.
Data Retention Violations	Low	High	Implement retention policies, automate data deletion process.

5. High-Level Designs

5.1 Outline

A plan for implementation of the Interactive Storytelling and Learning App is established by the high-level design. It provides the foundation for detailed design and implementation by establishing the main subsystems and their interactions. The architecture has multiple layers, such as the presentation layer, where child, parent, and teacher interfaces are hosted; the processing layer, where AI, natural language processing, and adaptive content algorithms are hosted; and the data management layer, where real-time synchronization and storage are used. Beyond visibly demarcating problems for development in modules, this blueprint assures that all pertinent factors of system performance, security, and ease of use are addressed.

5.2 System Overview

It gives real-time data analysis, AI-supported content modifications, and cross-platform access, which altogether provide an engaging and interactive learning experience for children. The system consists of the following components:

- **AI-Powered Storytelling Engine:** Uses Natural Language Processing (NLP) and Machine Learning (ML) for real-time story generation and modification, tracking child learning progress, interests, and interaction.

- **User Interface:** A child-friendly interactive front end created for mobile (iOS and Android) for multi-device availability.

- **Backend Processing System:** This system provides cloud computation for AI, real-time modification of stories, tracking user interaction, and analytical reporting.

- **Data Management & Security Layer:** Secure cloud storage (e.g., Firebase or AWS) is used for storing user profiles demonstrating learning progress and insights for parents, in a COPPA and GDPR-compliant manner.

- **Parental & Educator Dashboard:** Enables tracking of progress and learning analytics, and parental controls to oversee a child's activities and performance in the app.

5.3 Design Considerations

It's always an architect of an Interactive Storytelling and Learning App that assures safety, access, and flexibility towards younger learners. The design considerations include the following:

- **User Accessibility:** A platform that is friendly to children and easy to navigate by touch interaction along with bright graphics to make it lively yet user-friendly.

- **Privacy and Security:** Since the app collects children’s learning data, stringent encryption protocols and secure authentication mechanisms are implemented to safeguard personal information in compliance with COPPA and GDPR.

- **Scalability:** The architecture of the system is modular. Therefore, there will be opportunities in the future to include additional features, for instance, other more AI-based learning models, multilingual editions, and further gamification aspects. Adaptation will be eased when applying such progressive features according to the change in educational demands.

- **Cross-Platform Compatibility:** The application runs seamlessly across iOS and Android, thereby providing a uniform experience regardless of device specifications or operating system differences.

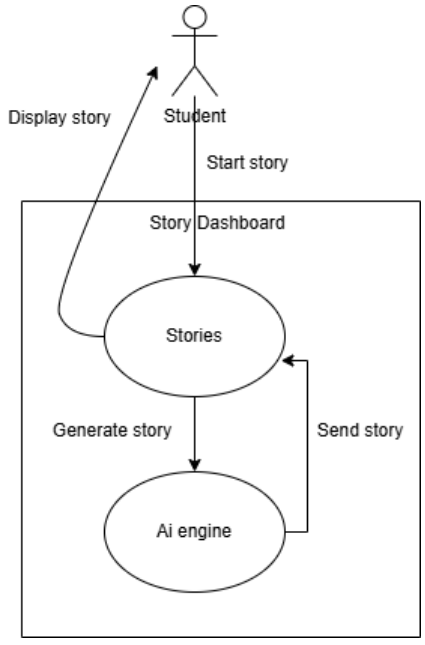
5.4 Functional Requirements

The functional requirements describe the core capabilities that the Interactive Storytelling and Learning App must deliver. These include:

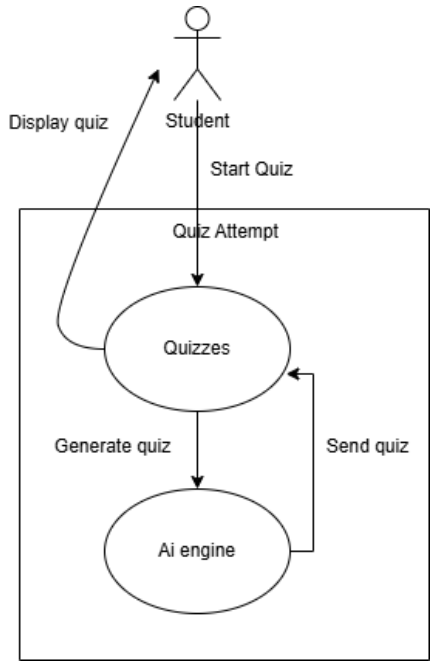
- I**nteractive Storytelling:** The app must offer stories that are not pre-defined but can be dynamically modified based on user choice. The system must be able to support many story paths that transform in real time, engaging children in decision-making.
- **Adaptive Learning:** AI algorithms combined with natural language processing scan user input to adjust the story's content and complexity. The adaptive mechanism provides each child with content based on their reading and comprehension level.
- **User Account Management:** The app must feature account creation, profile customization, and secure login for children, parents, and teachers.
- **Progress Tracking and Analytics:** The system must monitor user activity and provide for detailed reports of vocabulary growth, reading proficiency, and general learning outcomes. This information is accessible via individual parents' and teachers' dashboards.
- **Parent and Teacher Controls:** Administrators alone should have access to features which allow teachers to monitor progress, set learning objectives, and change content settings as needed.
- **Interactive Quizzes and Tests:** To enable learning, the app should include quizzes and flashcards that adapt difficulty in relation to real-time performance metrics.
- **Data Synchronization:** Progress, preferences, and performance metrics of users should be synchronized across devices using secure, cloud-based storage.

5.4.1 Use Cases

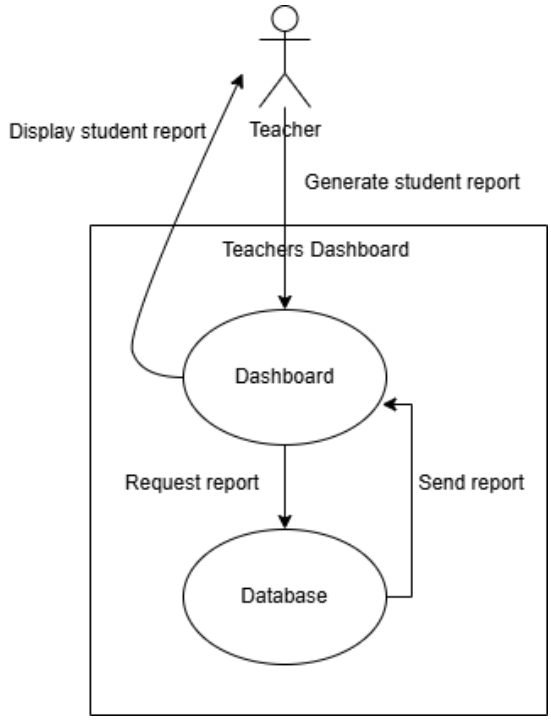
Story interaction:



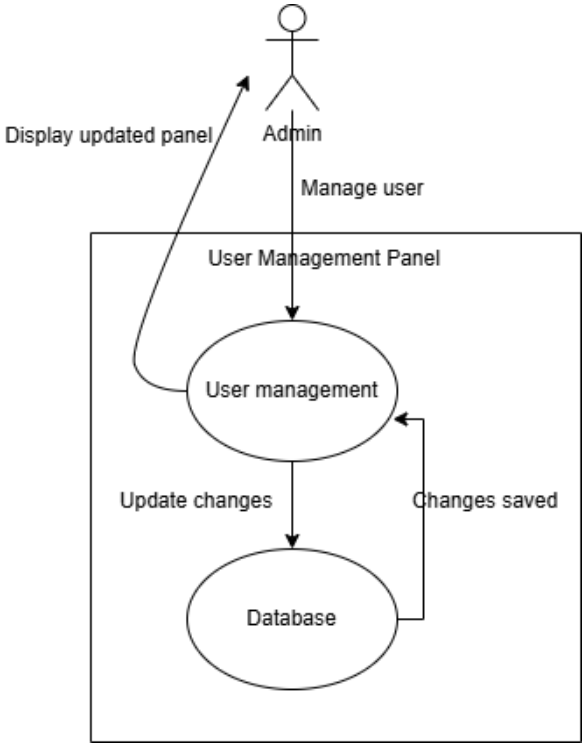
Quiz interaction:



Generate student report:

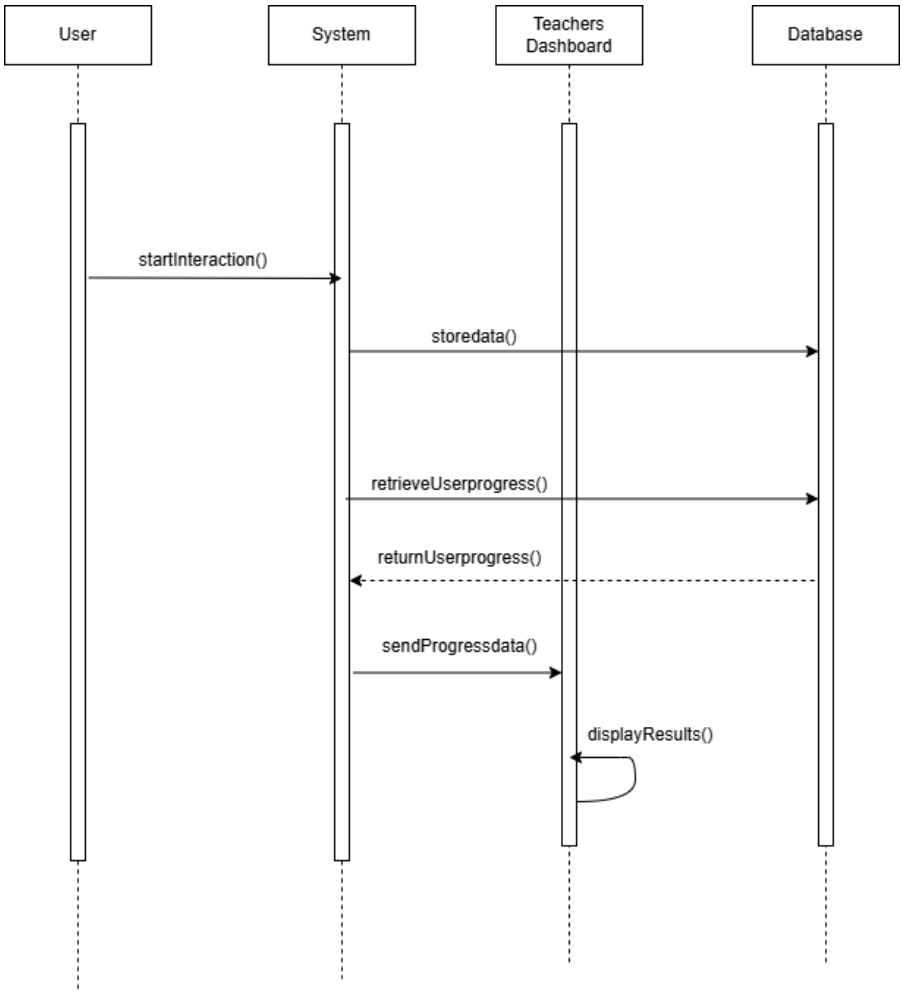


User management:

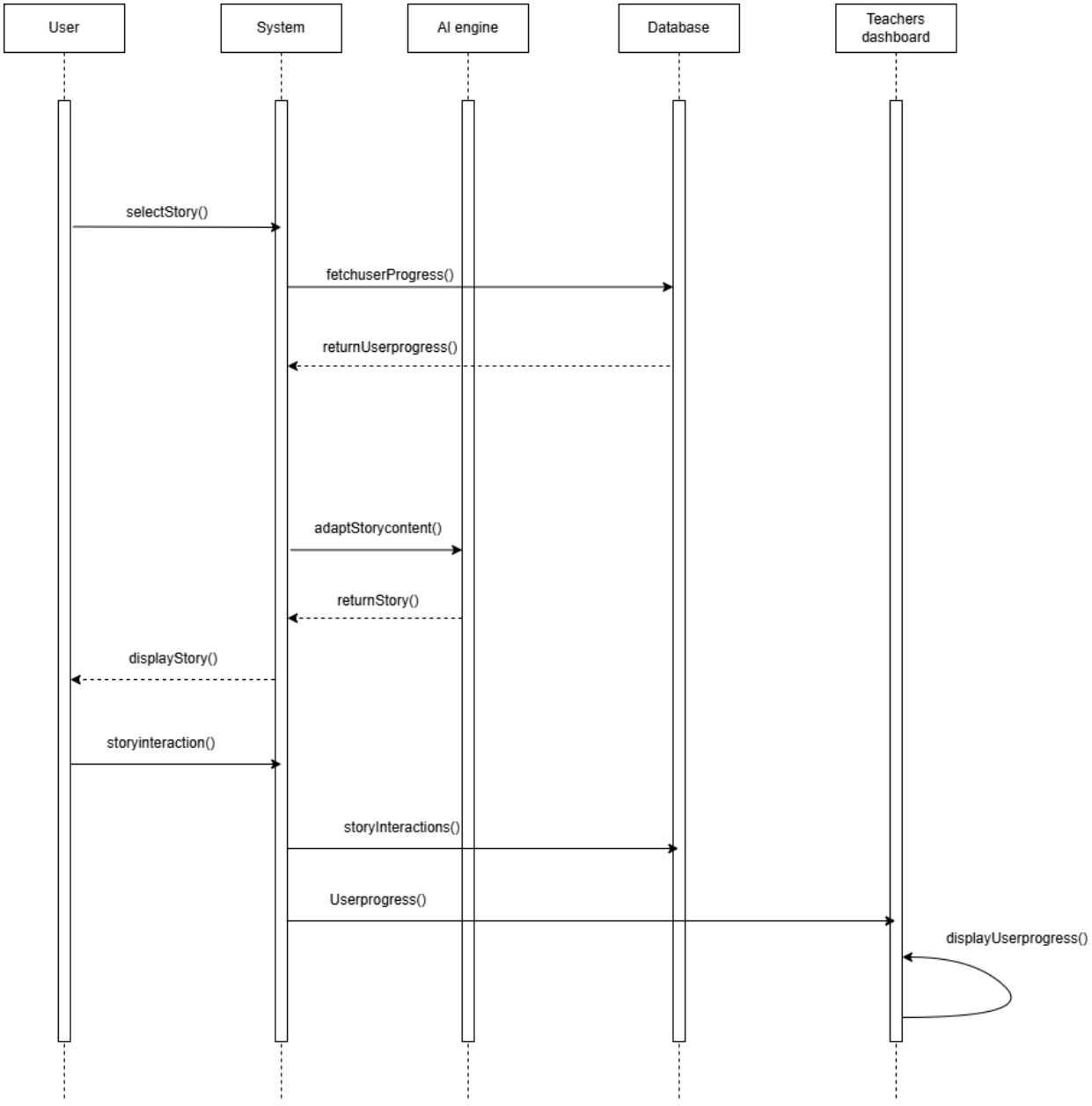


5.4.2 Sequence Diagrams

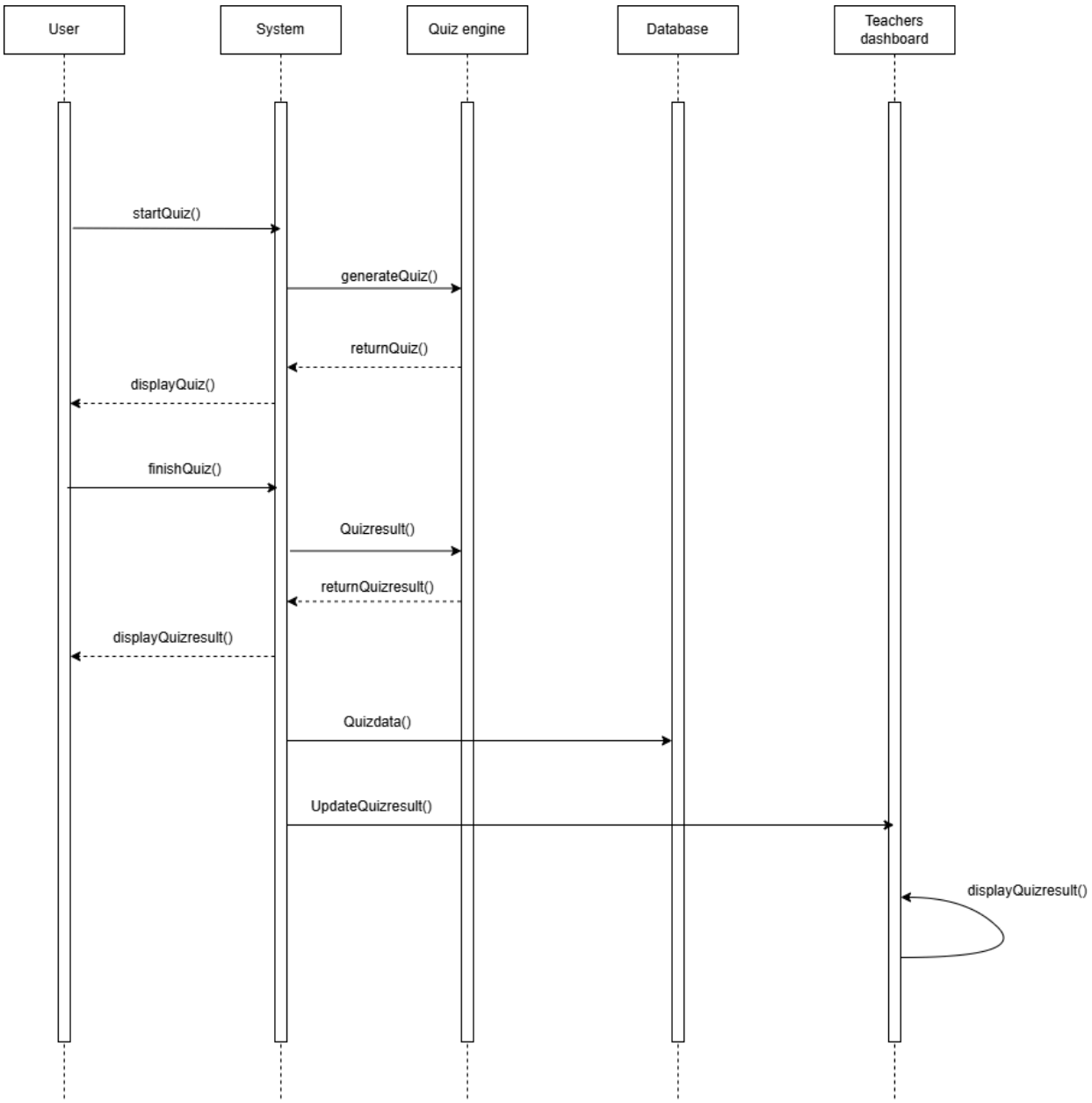
1. User Progress tracking & analytics



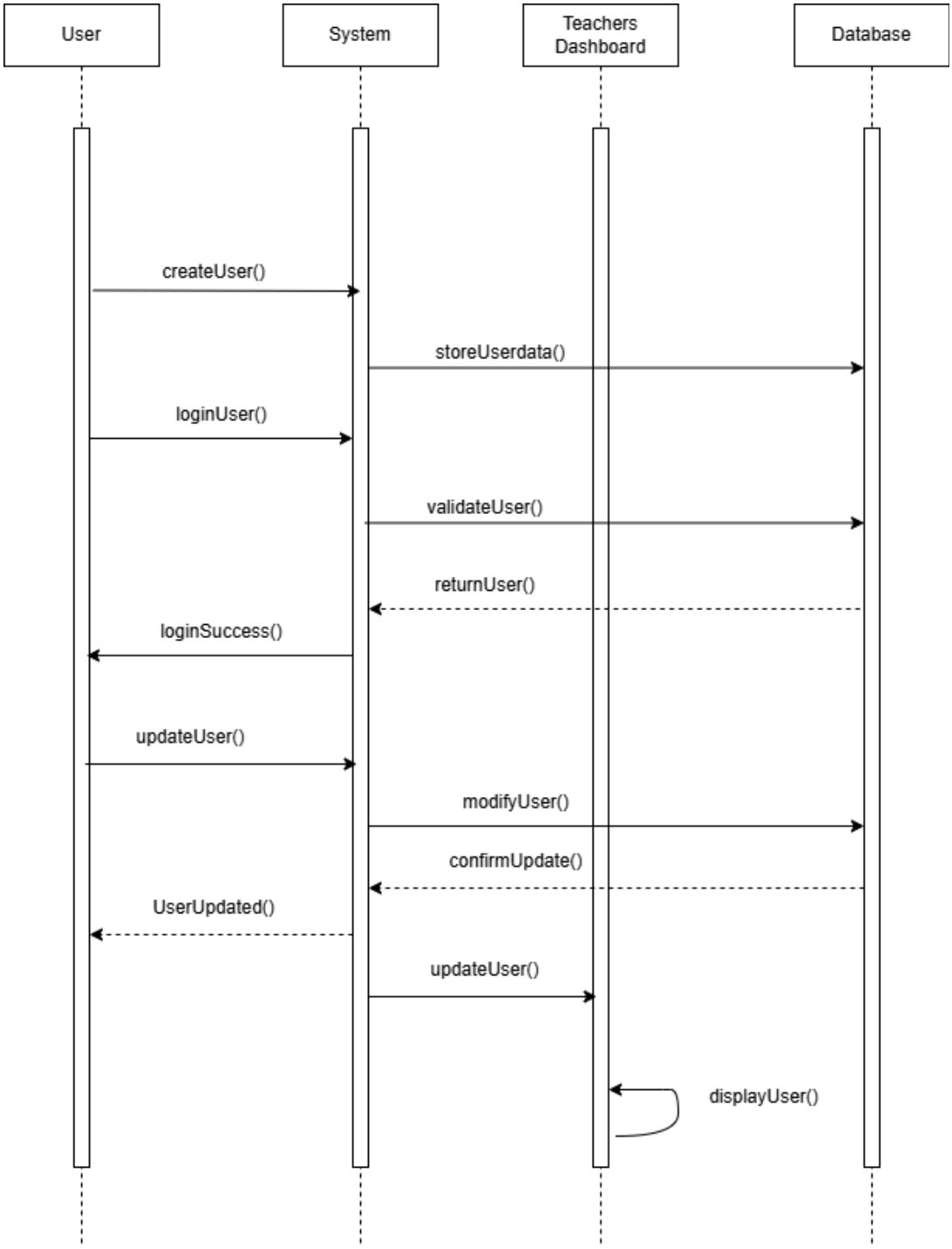
2. Adaptive learning



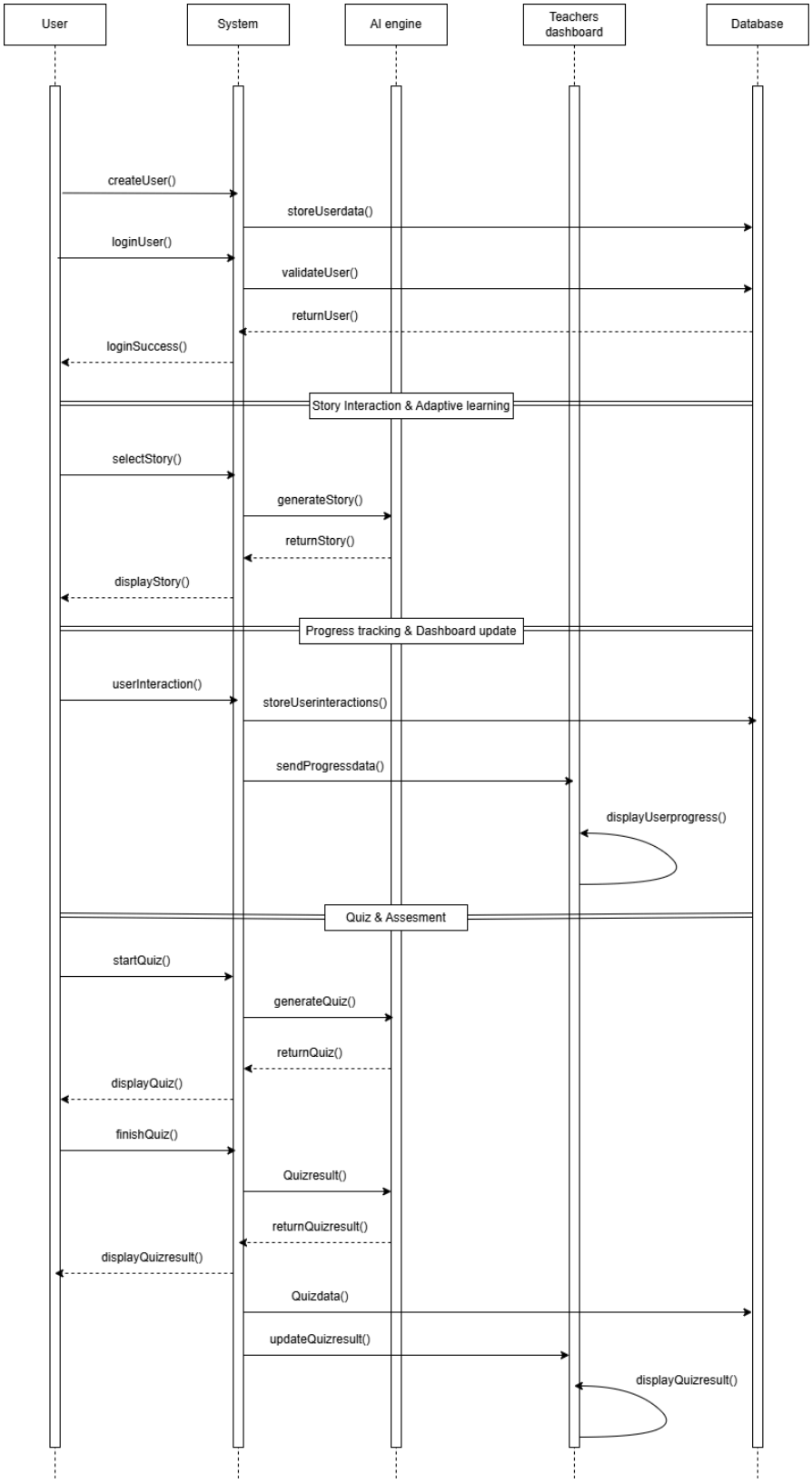
3. Interactive Quizzes and assessments



4. User Account Management



5. Overall System Flow



5.5 Assumptions and Dependencies

The section gives the assumptions and dependencies underpinning the high-level design and feeding into the running of the entire project. They are:

- Assumption that the target users (parents, children, and educators) have the contemporary, internet-enabled devices where they can comfortably execute the app.
- Availability of sufficient training data for developing and refreshing the adaptive storytelling AI and natural language models.
- Cloud infrastructure (e.g., Firebase, AWS) will manage real-time data synchronization, storage, and secure communication securely.
- Data privacy and security measures will remain under the control of regulatory frameworks such as COPPA and GDPR, and the project will remain compliant with them.
- Third-party libraries and APIs (for AI, etc.) integration will be stable and adequately supported for the duration of the project.

Strong AI platforms (like TensorFlow or PyTorch), cross-platform development platforms (like Flutter), and frequent monitoring from instructional consultants to keep the content aligned with learning goals are some of the dependencies.

5.6 Architectural Strategies

Implementation is carried out based on a modular component strategy in which each of the primary functions-AI-powered storytelling, user interaction, data management, and security-is addressed as a separate module. This enables flexibility, maintainability, and scalability.

Key architectural techniques used:

- **Service-Oriented Architecture (SOA):** Allows modularization of AI processing, user experience, and real-time data analysis for independent scaling and feature development.

- **Data Encryption and Access Control:** Adherence to COPPA and GDPR requirements on privacy and security of an application is being realized by secure authentication of users, data transmission via encryption, and role-based access to users.

- **Cloud-Based Scalability:** Utilizes cloud services (i.e., Firebase, AWS) to store data, AI processing, and real-time content synchronization with assured availability and seamless performance under varying user traffic.

- **Performance Optimization:** Employs caching, load balancing, and resource allocation mechanisms to ensure application responsiveness and stability under intense use.

This top-level architectural strategy guarantees that the Interactive Storytelling and Learning App satisfies present educational requirements while accommodating future enlargements and feature incorporations.

Its design is in a layered structure for increased efficiency and modularity:

- **Application Layer:** Manages user engagement through narrative interfaces, progress tracking dashboards, and real-time analytics for educators and parents.

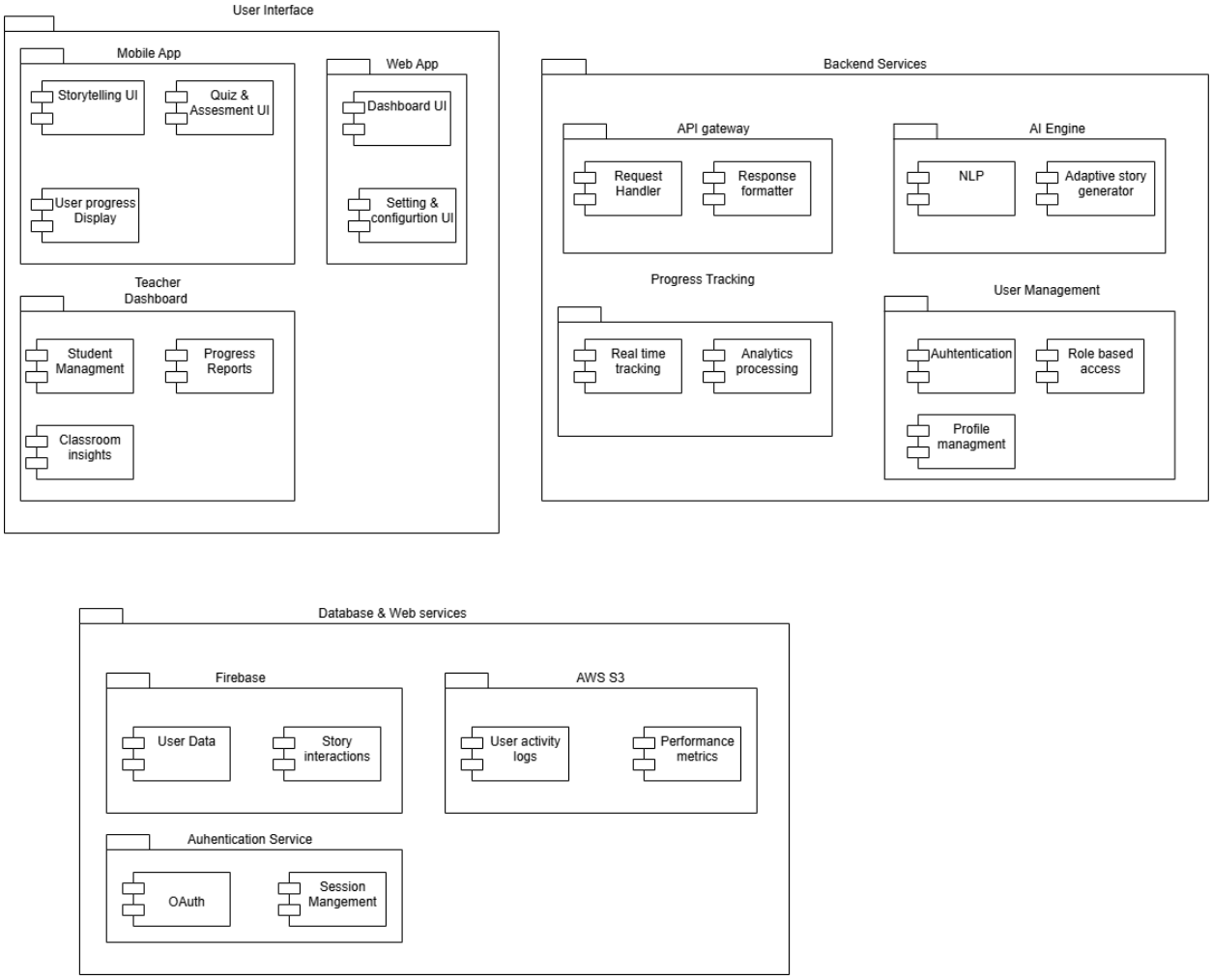
- **Software Layer:** Hosts AI-driven adaptive storytelling models, NLP engines, and gamification capabilities for engaging learning experiences. Flask powers backend API interactions, while Hugging Face enhances NLP processing for dynamic storytelling.

- **Cloud Layer:** Provides secure storage of information, AI computation, and cross-device synchronization to enable seamless cross-platform usage.

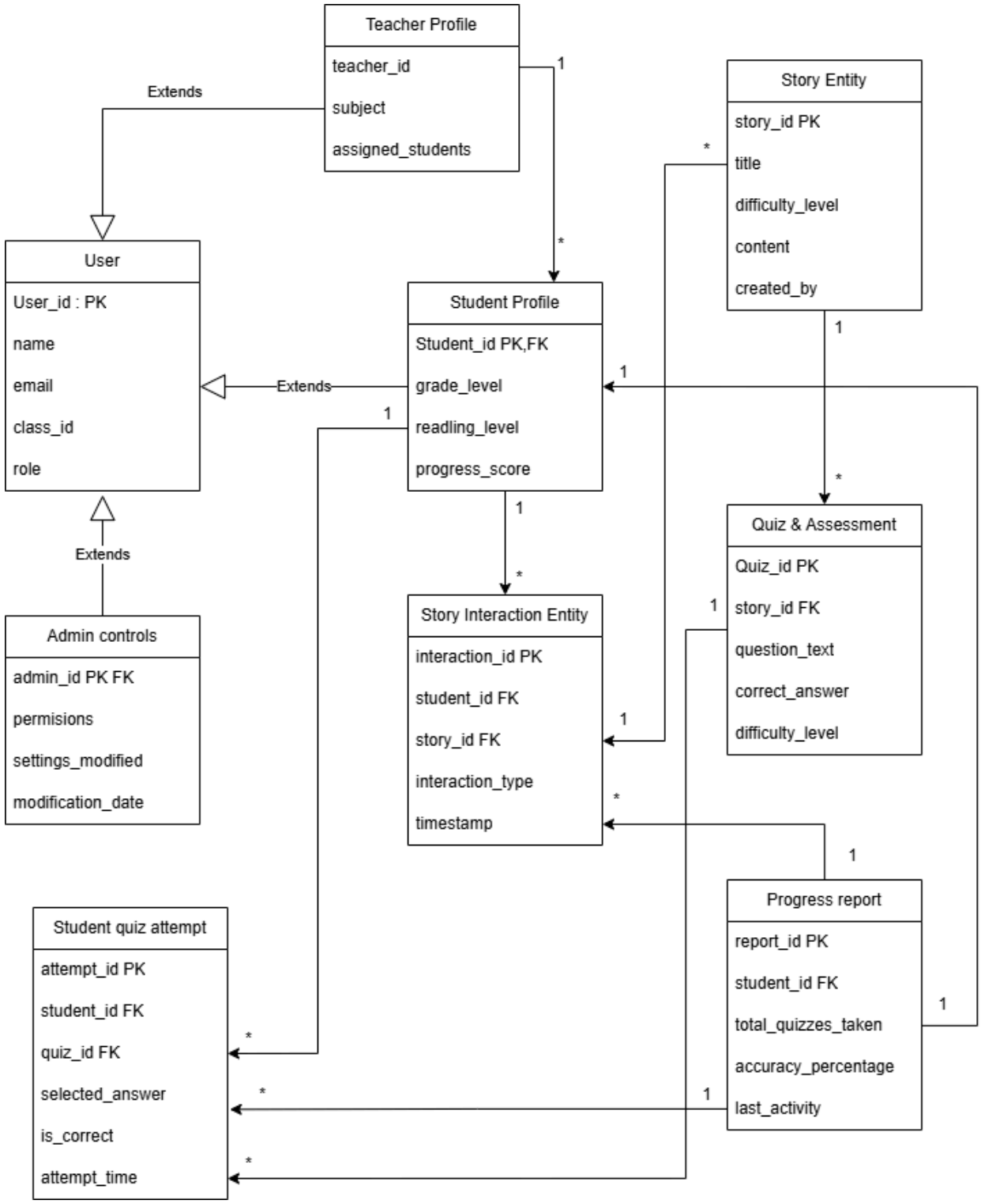
- **Security Layer:** Supports encrypted data management, user authentication, and real-time auditing to safeguard confidential information.

This highly organized architecture design guarantees that the system is scalable, resilient, and adaptable, with the ability to support future developments in AI-based learning.

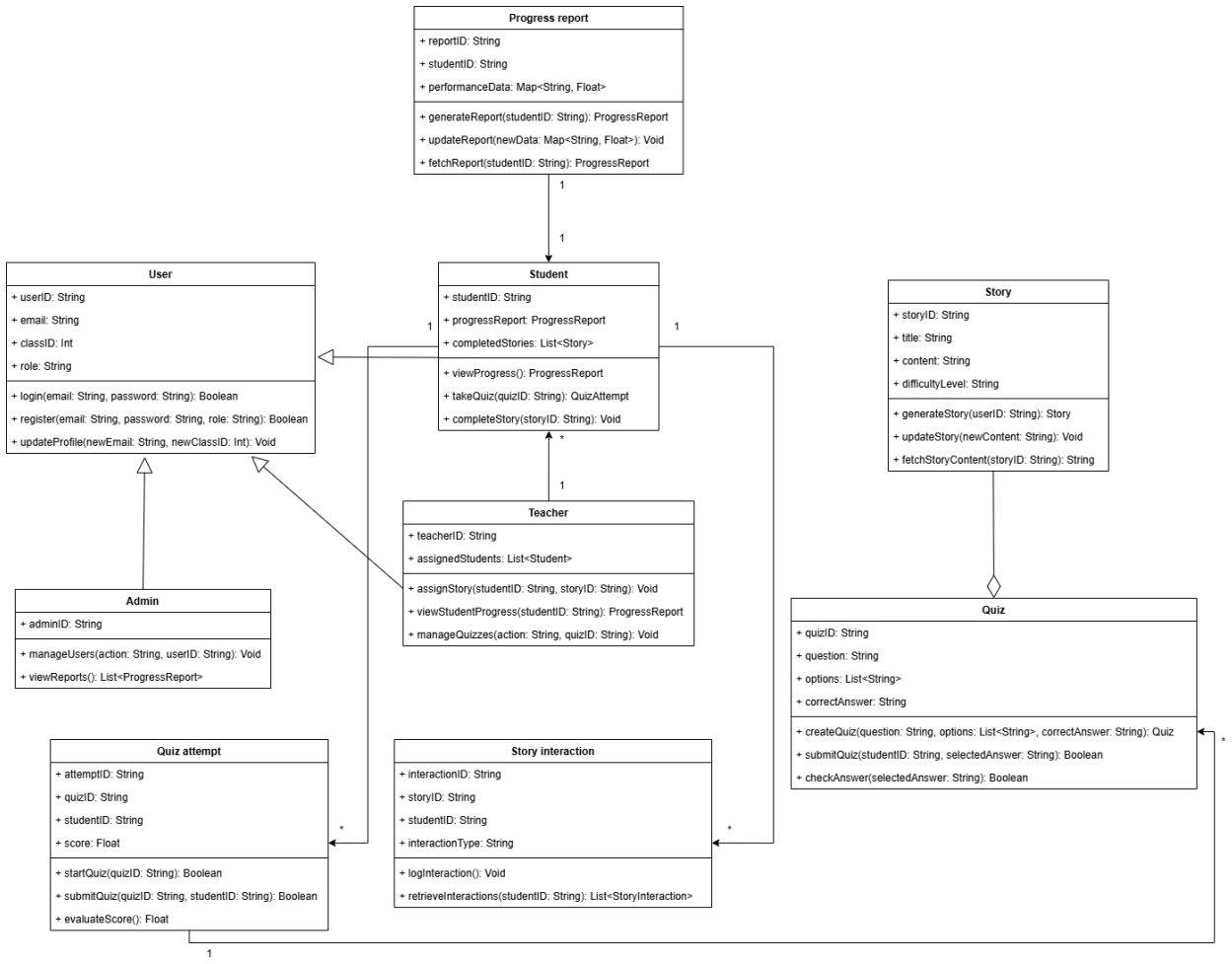
5.6.1 System Architecture



5.7 ER Diagram



5.8 Class Diagram



5.9 CRC Cards

5.9.1 Models

Stories
• Display and manage interactive stories • Track user interactions and choices • It also adapts the contents and generates a story based on the user input.
Collaborators: AI Engine, Database, Progress Tracking Module

Quizzes
● Store and manage quizzes ● Record quiz attempts and evaluate responses ● Quizzes will have different difficulty levels..
Collaborators: AI Engine, Progress Tracking Module, Notification Module

Database
● Store and retrieve all system data
Collaborators: All modules

Notification/Reminder Module
● Send reminders for quizzes and progress updates
Collaborators: Quiz Module, Teacher Interface

AI Engine
● Personalize learning content, adjust difficulty ● Generate stories ● Generates quizzes
Collaborators: Story Module, Quiz Module, Analytics Module

5.9.2 Interface

Student

• View interactive stories and quizzes • Attempt quizzes and receive feedback • Track personal progress
Collaborators: Story Module, Quiz Module, Progress Tracking Module

Teacher
• Monitor students' progress and quiz performance • Assign additional learning tasks • Send notifications/reminders
Collaborators: Progress Tracking Module, Notification Module, Analytics Module

Admin
• Manage system users (teachers, students) • Configure system settings and content • Monitor overall app performance
Collaborators: Database, Analytics Module

Authentication
• Manage user login via email or Class ID • Handle password recovery • Ensure secure access control
Collaborators: Database, Admin Interface

Content Management

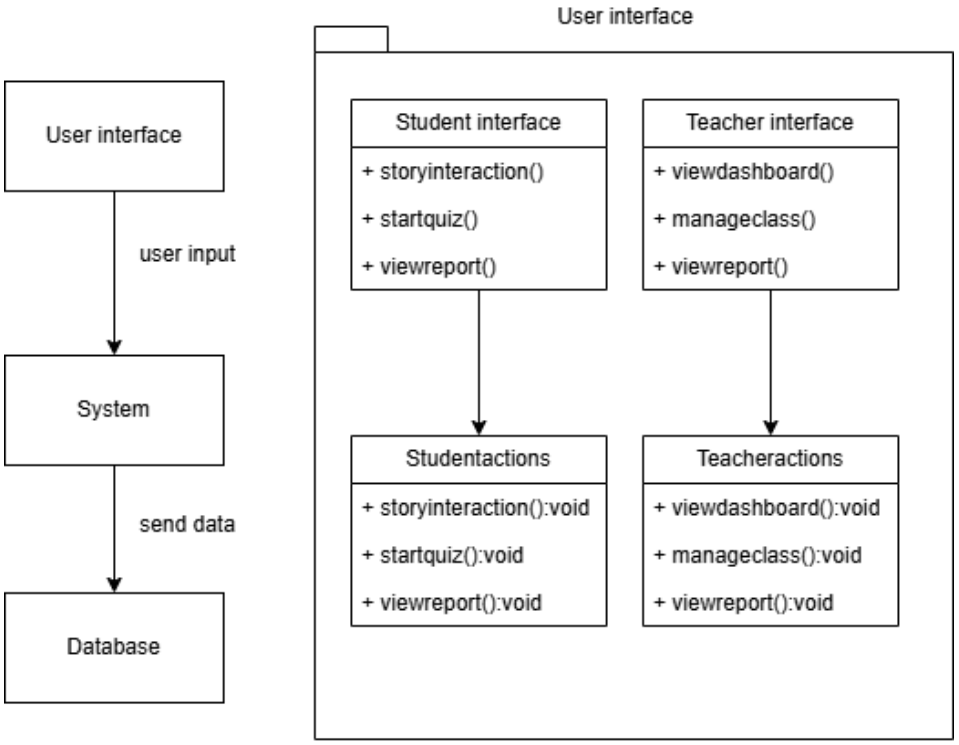
• Allow teachers/admins to create or modify stories and quizzes • Enable uploading of additional learning materials
Collaborators: Database, Story Module, Quiz Module

Notification & Alerts Interface
• Manage quiz reminders, progress alerts, and admin announcements • Send automated messages to students, parents, and teachers
Collaborators: Notification Module, Teacher Interface, Parent Interface

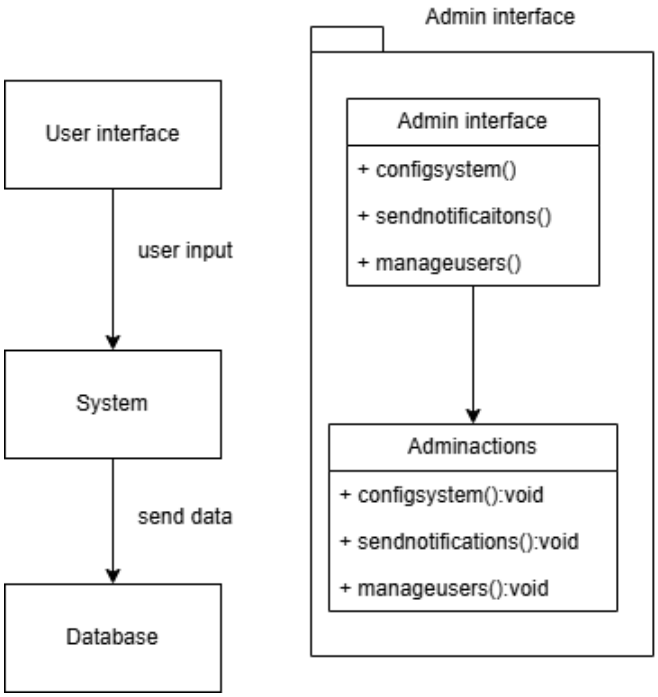
Progress & Reports
• Generate and display analytics for students and teachers • Provide real-time performance tracking
Collaborators: Progress Tracking Module, Analytics Module

5.10 User Interfaces

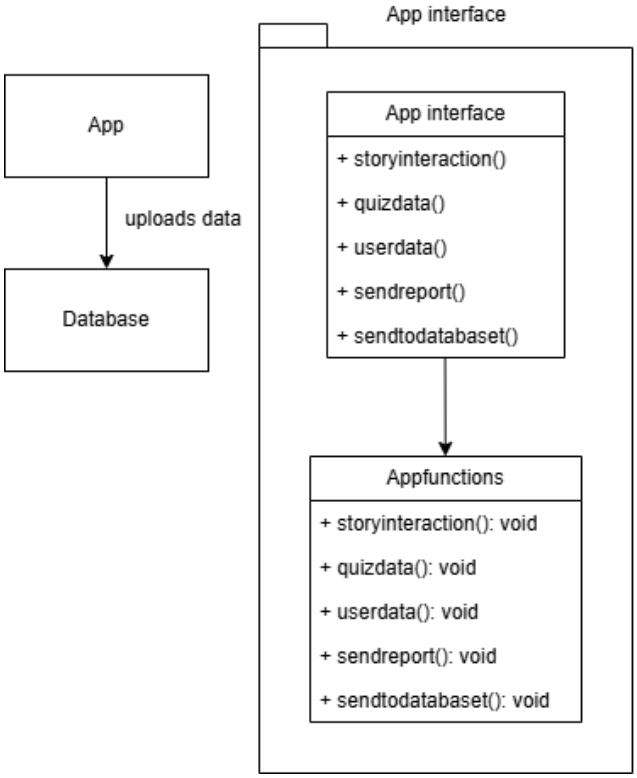
5.10.1 User Interface Diagram



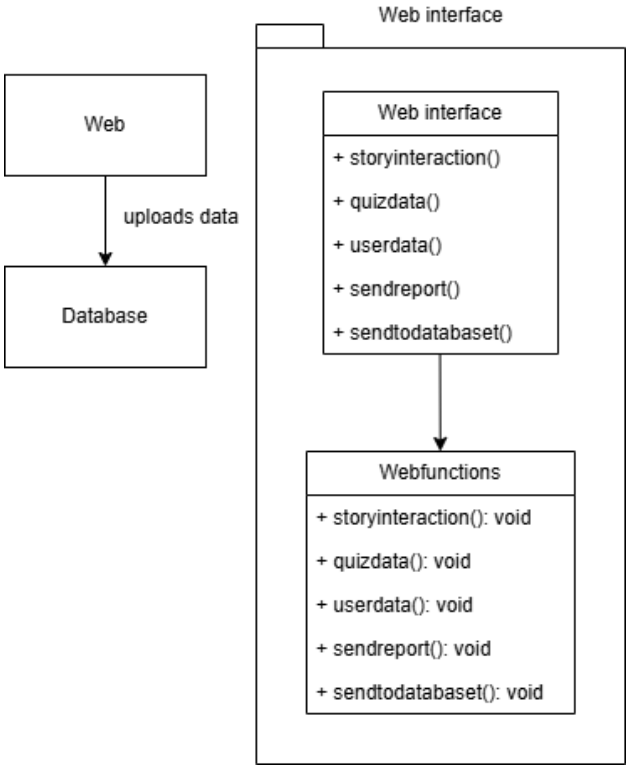
5.10.2 Admin Interface Diagram



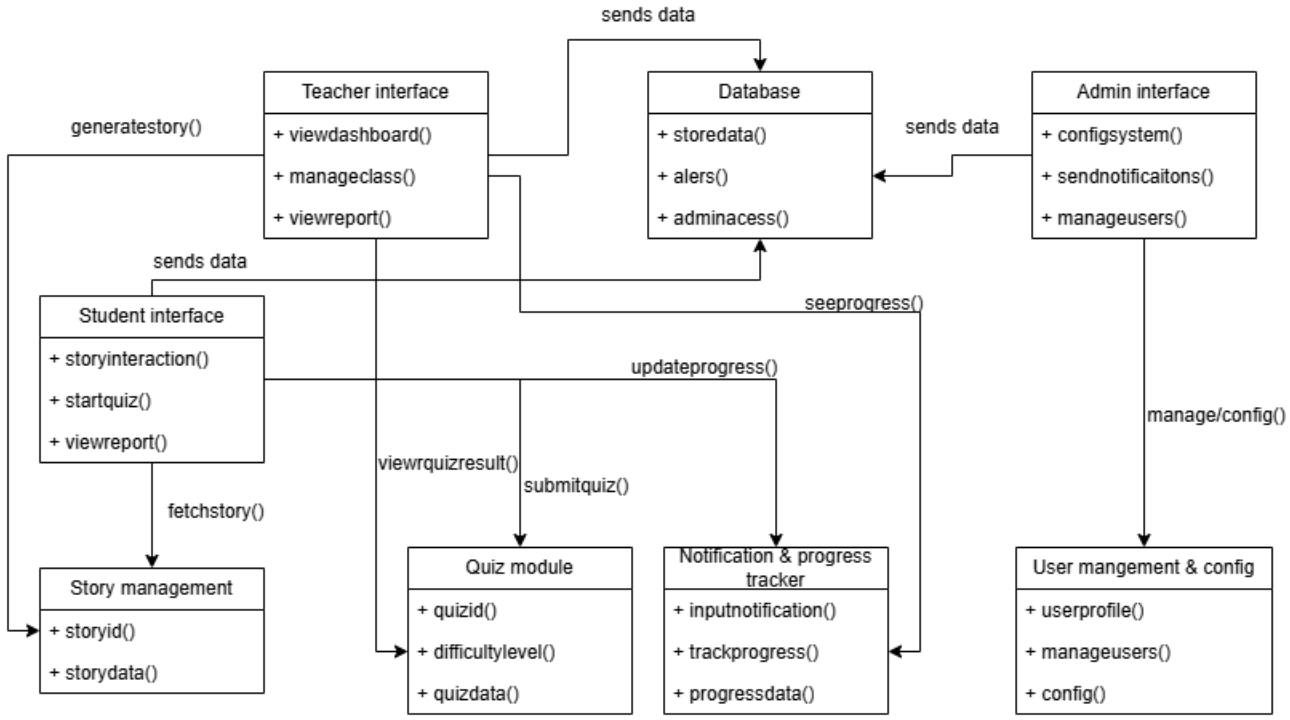
5.10.3 Mobile Application Interface



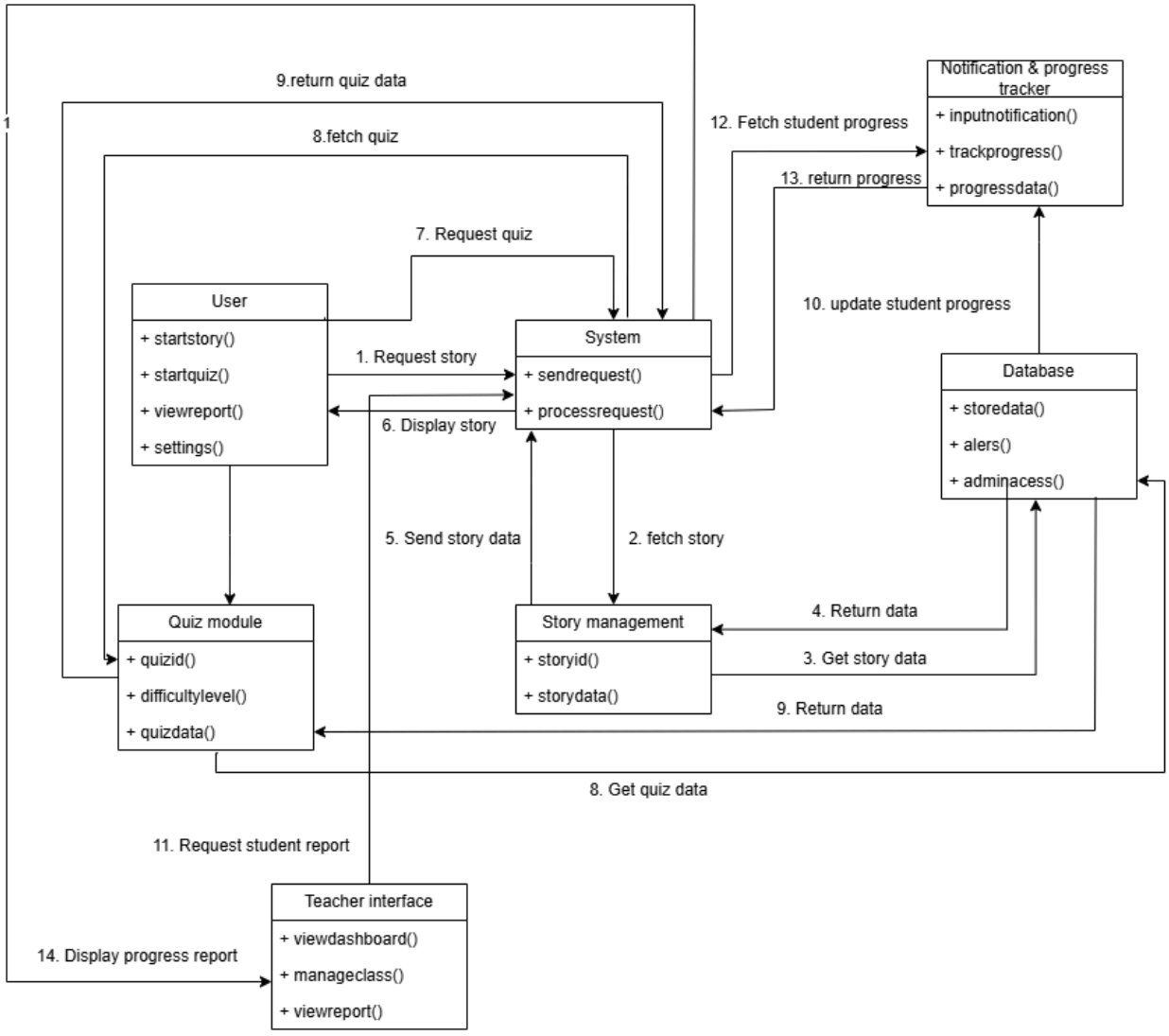
5.10.4 Web Interface



5.10.5 Overall Interface Diagram



5.11 Collaboration Diagram



6. Low-Level Design

6.1 Introduction

Interactive Storytelling and Learning Application Low-Level Design (LLD) elaborates in detail the step-by-step implementation of each feature, mapping the high-level abstractions to concrete software modules and algorithms. It specifies here the interfaces, data processing algorithm, and behavior for features such as adaptive story creation, real-time language analysis, interactive quiz examination, and progress monitoring. The design allows for simple coding and testing with the knowledge that each and every component is performance and security compatible. It describes in elaborate detail the whole design for developers, showing how various modules interact to create a unified and interactive learning experience.

6.2 Pseudocodes

6.2.1 Conversational AI for Story Personalization

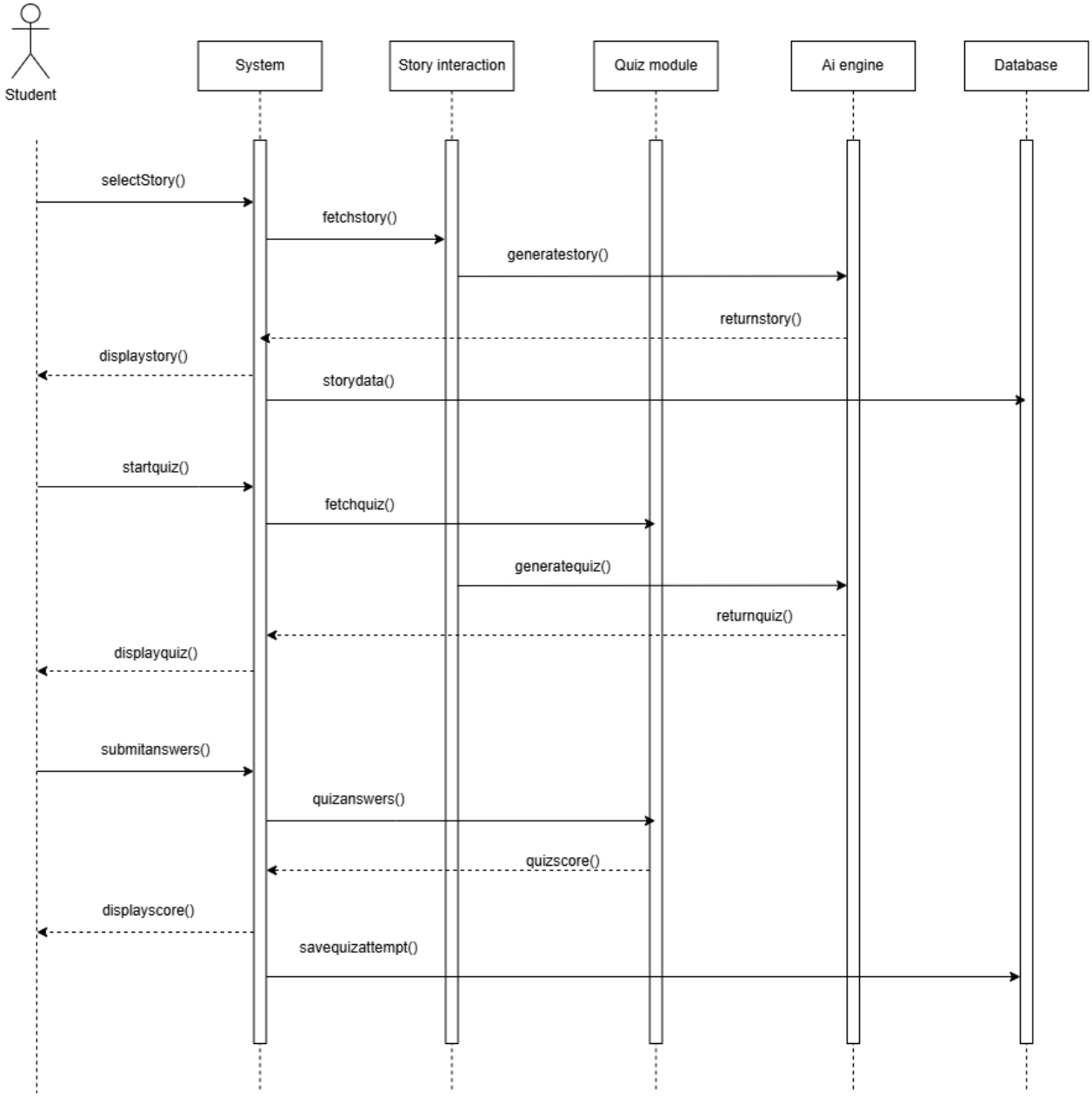
```
BEGIN
  INPUT user_id, user_age, reading_level, story_preferences

  FETCH user_profile FROM database USING user_id
  FETCH story_data BASED ON user_preferences AND reading_level

  DISPLAY "Welcome! Let's start your story!"

  WHILE story IS NOT COMPLETE
    DISPLAY current_story_segment

    IF user_input IS NOT EMPTY THEN
      PROCESS user_input USING NLP_Model
      UPDATE story_path BASED ON processed_input
    END IF

    IF story_path REQUIRES AI_GENERATION THEN
      GENERATE new_story_segment USING AI_Model
```

```
    END IF

    UPDATE user_progress IN database

  END WHILE

  DISPLAY "The End! Great job!"
  SAVE user_progress TO database

END
```

6.2.2 Adaptive Content Recommendation Engine

```
BEGIN
  INPUT user_id

  FETCH user_reading_history, performance_metrics FROM database
  ANALYZE user_engagement_level, comprehension_score

  IF comprehension_score > THRESHOLD THEN
    INCREASE story_difficulty
  ELSE
    MAINTAIN or DECREASE story_difficulty
  END IF

  FETCH recommended_stories BASED ON updated difficulty level

  DISPLAY recommended_stories TO user

  LOG recommendation_feedback IN database

END
```

6.3 Design sequence diagrams



7. Implementation

7.1 AI Storytelling System

```python
import torch
from transformers import GPT2Tokenizer, GPT2LMHeadModel
import random
import re

# Set Hugging Face cache directory
CACHE_DIR = "/tmp/huggingface"

# ------------------------
# Load Story Generation Model
# ------------------------
STORY_MODEL_NAME = "abdalraheemdmd/story-api"
story_tokenizer = GPT2Tokenizer.from_pretrained(STORY_MODEL_NAME,
cache_dir=CACHE_DIR)
story_model = GPT2LMHeadModel.from_pretrained(STORY_MODEL_NAME,
cache_dir=CACHE_DIR)

# ------------------------
# Load Question Generation Model
# ------------------------
QUESTION_MODEL_NAME = "abdalraheemdmd/question-gene"
question_tokenizer = GPT2Tokenizer.from_pretrained(QUESTION_MODEL_NAME,
cache_dir=CACHE_DIR)
question_model = GPT2LMHeadModel.from_pretrained(QUESTION_MODEL_NAME,
cache_dir=CACHE_DIR)

# Ensure tokenizers have a pad token
if story_tokenizer.pad_token_id is None:
    story_tokenizer.pad_token_id = story_tokenizer.eos_token_id
if question_tokenizer.pad_token_id is None:
    question_tokenizer.pad_token_id = question_tokenizer.eos_token_id

def generate_story(theme, reading_level, max_new_tokens=400, temperature=0.7):
    """Generates a story based on the provided theme and reading level."""
    prompt = f"A {reading_level} story about {theme}:"
    input_ids = story_tokenizer(prompt, return_tensors="pt").input_ids
    with torch.no_grad():
        output = story_model.generate(
            input_ids,
            max_new_tokens=max_new_tokens,
            temperature=temperature,
            top_k=20,
            top_p=0.7,
            do_sample=True,
            early_stopping=True,
            pad_token_id=story_tokenizer.pad_token_id,
            eos_token_id=story_tokenizer.eos_token_id,
            attention_mask=input_ids.ne(story_tokenizer.pad_token_id)
        )
    return story_tokenizer.decode(output[0], skip_special_tokens=True)
```

```python
def extract_protagonist(story):
    """
    Attempts to extract the protagonist from the first sentence by searching for
the pattern "named <Name>".
    Returns the first matched name, if available.
    """
    sentences = re.split(r'\.|\n', story)
    if sentences:
        m = re.search(r"named\s+([A-Z][a-z]+)", sentences[0])
        if m:
            return m.group(1)
    return None

def extract_characters(story):
    """
    Extracts potential character names from the story using a frequency count on
capitalized words.
    Filters out common stopwords so that the most frequently mentioned name is
likely the main character.
    """
    words = re.findall(r'\b[A-Z][a-zA-Z]+\b', story)
    stopwords = {"The", "A", "An", "And", "But", "Suddenly", "Quickly", "However",
"Well",
                 "They", "I", "He", "She", "It", "When", "Where", "Dr", "Mr"}
    filtered = [w for w in words if w not in stopwords and len(w) > 2]
    if not filtered:
        return []
    freq = {}
    for word in filtered:
        freq[word] = freq.get(word, 0) + 1
    sorted_chars = sorted(freq.items(), key=lambda x: x[1], reverse=True)
    return [item[0] for item in sorted_chars]

def extract_themes(story):
    """Extracts themes from the story based on keyword matching."""
    themes = []
    story_lower = story.lower()
    if "space" in story_lower:
        themes.append("space")
    if "adventure" in story_lower:
        themes.append("adventure")
    if "friend" in story_lower:
        themes.append("friendship")
    if "learn" in story_lower or "lesson" in story_lower:
        themes.append("learning")
    return themes

def extract_lesson(story):
    """
    Attempts to extract a lesson or moral from the story by finding sentences
    containing keywords like "learn" or "lesson". Returns the last matching
sentence.
    """
    sentences = re.split(r'\.|\n', story)
    lesson_sentences = [
        s.strip() for s in sentences
        if ("learn" in s.lower() or "lesson" in s.lower()) and len(s.strip()) > 20
    ]
    if lesson_sentences:
        return lesson_sentences[-1]
    else:
        return "No explicit lesson found."
```

```
def format_question(question_prompt, correct_answer, distractors):
    """
    Combines the correct answer with three distractors, shuffles the options,
    and formats the question as a multiple-choice question.
    """
    # Ensure exactly 3 distractors are available
    if len(distractors) < 3:
        default_distractors = ["Option X", "Option Y", "Option Z"]
        while len(distractors) < 3:
            distractors.append(default_distractors[len(distractors) %
len(default_distractors)])
    else:
        distractors = random.sample(distractors, 3)
    options = distractors + [correct_answer]
    random.shuffle(options)
    letters = ["A", "B", "C", "D"]
    correct_letter = letters[options.index(correct_answer)]
    options_text = "\n".join(f"{letters[i]}) {option}" for i, option in
enumerate(options))
    question_text = f"{question_prompt}\n{options_text}\nCorrect Answer:
{correct_letter}"
    return question_text

def dynamic_fallback_questions(story):
    """
    Generates three multiple-choice questions based on dynamic story content.
    Each question uses a randomly chosen template and shuffles its options.
    """
    protagonist = extract_protagonist(story)
    characters = extract_characters(story)
    themes = extract_themes(story)
    lesson = extract_lesson(story)

    # --- Question 1: Theme ---
    theme_templates = [
        "What is the main theme of the story?",
        "Which theme best represents the narrative?",
        "What subject is central to the story?"
    ]
    q1_prompt = random.choice(theme_templates)
    correct_theme = " and ".join(themes) if themes else "learning"
    q1_distractors = ["sports and competition", "cooking and baking", "weather and
seasons", "technology and innovation"]
    q1 = format_question(q1_prompt, correct_theme, q1_distractors)
```

```
 # --- Question 2: Primary Character ---
    character_templates = [
        "Who is the primary character in the story?",
        "Which character drives the main action in the narrative?",
        "Who is the central figure in the story?"
    ]
    q2_prompt = random.choice(character_templates)
    if protagonist:
        correct_character = protagonist
    elif characters:
        correct_character = characters[0]
    else:
        correct_character = "The main character"
    q2_distractors = ["a mysterious stranger", "an unknown visitor", "a supporting
character", "a sidekick"]
    q2 = format_question(q2_prompt, correct_character, q2_distractors)

    # --- Question 3: Lesson/Moral ---
    lesson_templates = [
        "What lesson did the characters learn by the end of the story?",
        "What moral can be inferred from the narrative?",
        "What is the key takeaway from the story?"
    ]
    q3_prompt = random.choice(lesson_templates)
    if lesson and lesson != "No explicit lesson found.":
        correct_lesson = lesson  # full sentence without truncation
    else:
        correct_lesson = "understanding and growth"
    q3_distractors = ["always be silent", "never try new things", "do nothing",
"ignore opportunities"]
    q3 = format_question(q3_prompt, correct_lesson, q3_distractors)

    return f"{q1}\n\n{q2}\n\n{q3}"

def generate_story_and_questions(theme, reading_level):
    """
    Generates a story using the story generation model and then creates dynamic,
    multiple-choice questions based on that story.
    """
    story = generate_story(theme, reading_level)
    questions = dynamic_fallback_questions(story)
    return {"story": story, "questions": questions}

# Alias for backward compatibility
create_fallback_questions = dynamic_fallback_questions
```

7.1.1 NLP Model and API Integration

To generate stories and questions, the system integrates two GPT-2 based language models via the Hugging Face Transformers API. The code below shows how the **story generation model** and a **question generation model** are loaded, along with their tokenizers, and how special tokens are handled:

```
# Load Story Generation Model

STORY_MODEL_NAME = "abdalraheemdmd/story-api"

story_tokenizer =
GPT2Tokenizer.from_pretrained(STORY_MODEL_NAME,
cache_dir=CACHE_DIR)

story_model = GPT2LMHeadModel.from_pretrained(STORY_MODEL_NAME,
cache_dir=CACHE_DIR)



# Load Question Generation Model

QUESTION_MODEL_NAME = "abdalraheemdmd/question-gene"

question_tokenizer =
GPT2Tokenizer.from_pretrained(QUESTION_MODEL_NAME,
cache_dir=CACHE_DIR)

question_model =
GPT2LMHeadModel.from_pretrained(QUESTION_MODEL_NAME,
cache_dir=CACHE_DIR)



# Ensure tokenizers have a pad token

if story_tokenizer.pad_token_id is None:

    story_tokenizer.pad_token_id = story_tokenizer.eos_token_id

if question_tokenizer.pad_token_id is None:
```

```
    question_tokenizer.pad_token_id =
question_tokenizer.eos_token_id
```

What this code does:

- **Model Loading (Lines 2-6):** Uses the Hugging Face `from_pretrained` API to load a GPT-2 tokenizer and a GPT-2 language model for storytelling. `STORY_MODEL_NAME` and `QUESTION_MODEL_NAME` are identifiers of pre-trained model repositories (in this case, likely custom fine-tuned models on Hugging Face Hub). The `cache_dir` is set to a local path (`/tmp/huggingface`) to cache the model files. This integration allows the app to fetch and use large language models without bundling the model weights with the app code. It abstracts away the details of downloading or loading model files, treating the Hugging Face Hub as a model API.

- **Multiple Models:** The system loads two models: one for story text generation (`story_model`) and one intended for question generation (`question_model`). By initializing both, the app is prepared to use specialized models for different tasks (story vs. questions). In the provided code, the question generation model is loaded but not explicitly used in the generation flow – instead, a fallback method is used for questions (explained later). Still, this demonstrates the modular integration of multiple NLP models, which could be utilized or swapped as needed.

- **Pad Token Setup (Lines 8-11):** Ensures that each tokenizer has a defined *pad token*. GPT-2 by default does not have a padding token (since it was trained on unconstrained text). The code checks `pad_token_id` and if it's `None`, it assigns it to the tokenizer's end-of-sequence token (`eos_token_id`). In GPT-2, the end-of-text token `<|endoftext|>` is used as a padding surrogate. This is important for model generation calls: it prevents warnings and ensures the `attention_mask` can be correctly created so that padded positions (if any) are ignored by the model. This setup follows recommended practice for open-ended text generation with models like GPT-2.

How it supports the system:

- *Hugging Face Integration:* By loading models via `GPT2Tokenizer` and `GPT2LMHeadModel` from the Transformers library, the system seamlessly integrates state-of-the-art NLP capabilities. The use of model names (e.g. `"abdalraheemdmd/story-api"`) allows the app to pull the latest version of the model, enabling updates without changing app code. This forms the backbone of the AI storytelling system – the models generate text based on prompts.

- *Configuration Consistency:* The model's configuration (see `config.json`) defines special tokens and model architecture. Notably, the configuration indicates that both the beginning-of-sequence (BOS) and end-of-sequence (EOS) tokens are the special token ID 50256, which corresponds to `<|endoftext|>`
  file-xzwzj7vrdjjuyh4fckqzkp
  . The tokenizer configuration likewise maps `<|endoftext|>` as the `bos_token`, `eos_token`, `pad_token`, and even `unk_token`file-ppujrs5x6tqnvytymm9xfm. By setting the pad token to the EOS token in code, the integration aligns with this configuration, ensuring the model and tokenizer handle text consistently (the model will see pad tokens as end-of-text markers, which it was trained to handle). The vocabulary size is 50257 tokensfile-xzwzj7vrdjjuyh4fckqzkp, matching GPT-2's vocab, so using the correct tokenizer ensures that input text is converted to the same token IDs the model expects.

- *API Usage:* In a larger app ecosystem, these loaded models and tokenizers would typically be kept in memory (perhaps at application startup) and used by API endpoints. For example, a web API route like `/generateStory` could call `generate_story_and_questions()` (see section 7.1.6) which internally uses `story_model` and `story_tokenizer`. This avoids re-loading models for each request, making the system efficient. The integration of the models here makes it possible for the app's backend to handle user requests for story generation in real-time, leveraging the pre-trained LLMs.

In summary, this integration code brings two pre-trained GPT-2 models into the application and prepares them for use. It establishes the foundation for all subsequent AI-driven features (story creation and Q&A generation) by providing access to the language models and ensuring proper token handling according to the model's configuration.

7.1.2 Story Generation Algorithm

The core of the storytelling app is the **story generation algorithm**, which uses the loaded NLP model to produce a story based on a given theme and reading level. The function `generate_story` encapsulates this logic. Below is the code that implements story generation using the Hugging Face `generate` API:

```
def generate_story(theme, reading_level, max_new_tokens=400,
temperature=0.7):
    prompt = f"A {reading_level} story about {theme}:"
    input_ids = story_tokenizer(prompt,
return_tensors="pt").input_ids
    with torch.no_grad():
        output = story_model.generate(
            input_ids,
            max_new_tokens=max_new_tokens,
            temperature=temperature,
            top_k=20,
            top_p=0.7,
            do_sample=True,
            early_stopping=True,
            pad_token_id=story_tokenizer.pad_token_id,
            eos_token_id=story_tokenizer.eos_token_id,

attention_mask=input_ids.ne(story_tokenizer.pad_token_id)
        )
```

```
    return story_tokenizer.decode(output[0],
skip_special_tokens=True)
```

What this code does:

- **Prompt Construction (Line 2):** It first creates a textual prompt based on the inputs. The prompt template is `"A {reading_level} story about {theme}:"`. For example, if `theme="space adventure"` and `reading_level="third-grade"`, the prompt becomes: **"A third-grade story about space adventure:"**. This prompt primes the model by providing context: it explicitly instructs the model to produce a story tailored to a certain reading level and theme. The colon at the end sets up the model to continue the text as a story narrative.

- **Tokenization (Line 3):** The prompt string is converted into input tokens for the model using the GPT-2 tokenizer. `story_tokenizer(prompt, return_tensors="pt")` encodes the prompt and returns PyTorch tensors (the `.input_ids` is the tensor of token IDs). This step translates the human-readable prompt into the numeric format the model understands. The tokenizer uses Byte-Pair Encoding (BPE) as defined in `tokenizer.json` (with the GPT-2 vocabulary of size 50257 and special tokens like `<|endoftext|>` included file-c9g9nkntfoioygi7zkxen4). The `return_tensors="pt"` ensures we get a PyTorch tensor ready to feed into the model. At this point, `input_ids` might look like (for the above example prompt) a tensor of indices corresponding to tokens for "A", "third", "-", "grade", " story", " about", " space", " adventure", ":" and so forth, plus potentially special tokens (the GPT-2 tokenizer does not add a BOS token by default because `"add_bos_token": false` in the configfile-ppujrs5x6tqnvytymm9xfm, so the input begins directly with the text tokens).

- **Model Generation (Lines 4-11):** Inside a `torch.no_grad()` context (disabling gradient calculation since we are not training, just inference), it calls `story_model.generate(...)` to produce the continuation of the prompt. The parameters passed here define *how* the text is generated:

- `max_new_tokens=400`: The model will generate up to 400 new tokens (in addition to the prompt). This limits the length of the story to ensure it doesn't go on indefinitely. 400 tokens roughly correspond to a few paragraphs of text, which is suitable for a short story.

- `temperature=0.7`: This controls the randomness of the generation. A lower temperature (<1.0) makes the model more conservative and reduces the likelihood of less probable tokens, resulting in more coherent and safer outputs. At 0.7, the story will have some creativity but not be too random.

- `top_k=20` and `top_p=0.7`: These are settings for *nucleus (top-p) sampling* and *top-k sampling*. `top_k=20` means at each step, the model considers only the top 20 most probable next tokens; `top_p=0.7` means it considers the smallest set of tokens whose cumulative probability exceeds 0.7. Combining these ensures the model doesn't choose from the long tail of unlikely tokens, which improves coherence. In practice, the generator will take the intersection of these constraints (this makes the output focused on likely continuations but still with some variability). This is important for storytelling, as it helps maintain logical narrative flow while still allowing for creative word choices.

- `do_sample=True`: This enables sampling (as opposed to greedy or beam search). With sampling, each generation can be different even if the input prompt is the same, introducing variety in stories. This is essential for an interactive app so that users can get a slightly different story each time or for each theme.

- `early_stopping=True`: This parameter signals the generation to stop when it encounters the end-of-sequence token or when the story seems complete, rather than always hitting the max tokens limit. It helps prevent the model from rambling or adding irrelevant text after finishing the story. Essentially, if the model outputs the EOS token (`<|endoftext|>`), generation will stop early.

- `pad_token_id` and `eos_token_id`: Both are set explicitly to the GPT-2 end-of-text token ID (which we ensured above). EOS token defines what token marks the end of the story. Pad token ID here is also the EOS token ID (50256) as set in the tokenizer config

file-ppujrs5x6tqnvytymm9xfm
, which is used to pad the input or output if needed. Passing these avoids a warning about them not being set and ensures the model knows what constitutes padding vs. real data. The GPT-2 model config indeed uses 50256 for eos_token_idfile-xzwzj7vrdjjuyh4fckqzkp.

    - `attention_mask=input_ids.ne(story_tokenizer.pad_token_id)`: This creates an attention mask tensor that marks real tokens versus padding tokens. `input_ids.ne(...)` returns a tensor of the same shape as input_ids with `1` where the token is not equal to the pad token, and `0` where it is equal. Since our prompt has no padding (it's one continuous sequence and we typically don't pad single inputs), this mask will be all ones. However, this parameter is included as a best practice, especially if one wanted to batch multiple prompts of different lengths. It ensures the model's self-attention mechanism does not attend to any position that is just padding. In effect, it contributes to stable and expected generation results.

- **Decoding the Output (Line 12):** The raw output from `generate` is a tensor of token IDs (the prompt tokens followed by generated tokens). `story_tokenizer.decode(output[0], skip_special_tokens=True)` converts this tensor back into a human-readable string. `skip_special_tokens=True` means it will remove any special tokens like `<|endoftext|>` that might have been generated at the end. The result is the full story text in natural language. This text is then returned by `generate_story`.

How it supports the system:

- *Automated Story Creation:* This algorithm is the heart of the storytelling app. It takes a simple idea from the user (the theme and desired reading level) and produces a fully-fledged story. This allows the app to offer virtually unlimited stories on any topic, generated on demand, which is far more scalable than a library of pre-written stories.

- *Control Over Output:* The use of generation parameters (temperature, top_k, top_p, max tokens) gives the app developers control to ensure stories are appropriate in length and style. For instance, by limiting `max_new_tokens` and

using moderate temperature, the stories are more likely to be coherent and age-appropriate (important for an educational or child-focused storytelling app). These parameters can be fine-tuned further if needed (e.g., adjusting `temperature` for more or less creativity).

- *Integration with Reading Level:* Note that the prompt includes the `reading_level` (e.g., "A **second-grade** story about..."). This is a simple yet effective way to steer the model's output complexity. The model, having been fine-tuned on a corpus with prompts indicating reading level (presumably), will adapt its vocabulary and sentence structure to match the target reading level. For example, a "kindergarten story" prompt would likely yield shorter sentences and simpler words than a "middle-school story". This integration of user parameters into the prompt is a form of *prompt engineering* that guides the model's style of output.

- *Efficiency via No Gradient:* Wrapping the generation in `torch.no_grad()` ensures that the computation is efficient and uses minimal memory by not storing gradients (since we only need a forward pass for inference). This is critical for a production system serving many requests, as it keeps the resource usage per request lower.

- *Consistent Story Start:* The way the prompt is structured (with a colon at the end) and skipping special tokens on decode ensures the output starts as a continuation, making the transition from prompt to story seamless. Users reading the story would not see the prompt text; they only see the story that follows. For example, if the prompt is "A third-grade story about space adventure:", the decoded output might look like: *"Once upon a time, a group of young students built a rocket to explore the stars..."* — the prompt itself is not included in the final output because it's just the context given to the model, and the decode function by default will include the prompt text too. In this code, since they decode `output[0]` which contains the prompt tokens + story, the returned text actually **will** include the prompt prefix unless the model generated an EOS before finishing the prompt. However, because the prompt is written as if it's part of the story, it reads naturally. (If needed, one could slice the output tokens to remove the prompt, but here the prompt is phrased to act as the story's first line.)

In summary, `generate_story` uses the NLP model to turn a user's theme and reading level request into a narrative. It encapsulates the complexity of text generation behind a simple function interface, which the rest of the app can call. This algorithm is crucial for

delivering the interactive storytelling experience, as it produces the content that users come for (the stories).

7.1.3 User Progress Tracking

Was nor completed due to complication and inefficient systems.

7.1.4 AI-based Content Adaptation

AI-based content adaptation in this system refers to how the app tailors the generated content to the user's needs or context using AI. In the context of our storytelling app, the primary form of adaptation is **adjusting the story's complexity and style based on the target reading level**. This ensures that a child or user with a specific reading ability gets a story that is neither too simple nor too advanced.

The adaptation is achieved by including the reading level in the generation prompt, as seen in the `generate_story` function. Here's the relevant line (from the code above) that illustrates this:

```
prompt = f"A {reading_level} story about {theme}:"

# For example: reading_level = "second-grade", theme =
"friendship"

# Resulting prompt: "A second-grade story about friendship:"
```

How this works and what it does:

- The format of the prompt explicitly mentions the reading level (e.g., "second-grade", "third grade", "kindergarten-level", etc.). This guides the AI model to adapt its output. Essentially, the model is conditioned to the style described in the prompt. During fine-tuning of the story generation model, it was likely trained on examples where the text to generate was preceded by such phrases, teaching the model to associate certain vocabulary and sentence structures with certain grade levels or difficulties.

- When the model reads "A second-grade story about friendship:", it will attempt to continue in a way that fits a second-grade reading level. In practical terms, this means it will use simpler words, shorter sentences, and a more straightforward narrative style. For a higher reading level, the language might be more complex and the sentences longer. This is a dynamic adaptation driven entirely by the AI's understanding of the prompt context.

- This method of content adaptation is **data-driven**. There is no hard-coded vocabulary list or rule-based simplification in the code; instead, the model's neural network has learned stylistic differences. The benefit is flexibility: you can ask for a story at any level ("first grade", "middle school", "advanced", etc.) and the model will do its best to adjust.

Support from configuration and model behavior:

- The `config.json` and `tokenizer_config.json` indirectly support this feature by ensuring the model can interpret the prompt correctly. The tokenizer will break "second-grade" into tokens and the model will ingest those tokens as context. Because the model is GPT-2 (as indicated by `"model_type": "gpt2"` in config
file-xzwzj7vrdjjuyh4fckqzkp
), it has no inherent notion of reading level except what it learned from training data. The success of this adaptation relies on how the model was fine-tuned. The presence of the reading_level in the prompt is a cue that was likely part of the training format for the model (since the model name "story-api" suggests a custom fine-tuned model for stories). The model's parameters (like 12-layer Transformer, 117M parameters as per config with `n_layer:12, n_embd:768`file-xzwzj7vrdjjuyh4fckqzkp) allow it to capture subtleties of language appropriate to different age groups if given sufficient examples.

- **Generation parameters** also help ensure the content is appropriate. For instance, a slightly lower `temperature` (0.7) and limited `top_p` (0.7) reduce the chance of the model veering into very complex or unexpected language, which indirectly keeps the content in line with the expected simplicity for a given reading level. This is an AI-based way of reining in content complexity without manual intervention.

Other forms of adaptation:

- *Adaptive content length:* While not explicitly user-controlled here, the `max_new_tokens` parameter (400) combined with early stopping means the story's length is kept reasonable. If the reading level implies a younger reader, the story generated might naturally be shorter (the model might end with an `<|endoftext|>` token earlier), whereas for an older reader it might utilize more of the allowed length. The system doesn't explicitly change `max_new_tokens` per level, but the content itself could adapt in length due to how the model was trained to respond to those prompts.

- *Theme integration:* Though not exactly an "adaptation" in the sense of difficulty, it's worth noting that the model also adapts the story content to the requested theme. The prompt's second part "story about {theme}" ensures the output story is on-topic. This is AI-driven content adaptation to user interest. If the theme is "dinosaurs", the model will likely produce a story with dinosaurs; if the theme is "friendship", the story will revolve around friendship. The model's ability to stay on theme is part of its learned knowledge.

- *Post-generation analysis:* The system uses AI/NLP methods to extract information like the story's lesson or themes (see `extract_lesson`, `extract_themes` in the code). While these functions are rule-based (keyword search, etc.), they represent a form of content understanding. The extracted data is used to adapt subsequent outputs (specifically, to tailor quiz questions to the story's content). This can be seen as another layer of AI-based adaptation: the content of the story informs the content of the questions, ensuring relevance.

Contribution to the larger app ecosystem:

AI-based content adaptation is crucial for user engagement and educational value. By tailoring each story to the appropriate reading level:

- Young readers remain engaged because the vocabulary and sentence structures match their comprehension abilities. They won't be frustrated by language that is too hard.

- Parents or educators can trust that selecting a certain level will yield content suitable for that age group.

- The app can serve a wider audience range by simply switching the prompt parameter – the same underlying model can produce different styles of text, which is cost-effective (no need for separate models or hardcoded story versions).

In essence, this component leverages the flexibility of the language model to adapt tone and complexity, making the storytelling experience personalized. It's an AI-driven approach (using the model's learned language patterns) to adjust content in a way that traditionally would require writing multiple versions of a story for different reading levels.

7.1.5 Gamification and Rewards System

To make the storytelling experience interactive and educational, the app includes a **gamification** element: it generates quiz questions based on the story. This turns passive reading into an active challenge, rewarding the user for paying attention and understanding the story. The provided code shows a `dynamic_fallback_questions` function which creates multiple-choice questions from the story, and a helper `format_question` to format each question. Below are excerpts of these functions:

```
def format_question(question_prompt, correct_answer,
distractors):

    # Ensure exactly 3 distractors are available

    if len(distractors) < 3:

        default_distractors = ["Option X", "Option Y",
"Option Z"]

        while len(distractors) < 3:


distractors.append(default_distractors[len(distractors) %
len(default_distractors)])

    else:

        distractors = random.sample(distractors, 3)

    options = distractors + [correct_answer]

    random.shuffle(options)

    letters = ["A", "B", "C", "D"]

    correct_letter = letters[options.index(correct_answer)]

    options_text = "\n".join(f"{letters[i]}) {option}" for
i, option in enumerate(options))

    question_text =
f"{question_prompt}\n{options_text}\nCorrect Answer:
{correct_letter}"
```

```
    return question_text


def dynamic_fallback_questions(story):
    protagonist = extract_protagonist(story)
    characters = extract_characters(story)
    themes = extract_themes(story)
    lesson = extract_lesson(story)

    # --- Question 1: Theme ---
    theme_templates = [
        "What is the main theme of the story?",
        "Which theme best represents the narrative?",
        "What subject is central to the story?"
    ]
    q1_prompt = random.choice(theme_templates)
    correct_theme = " and ".join(themes) if themes else
"learning"
    q1_distractors = ["sports and competition", "cooking and
baking", "weather and seasons", "technology and innovation"]
    q1 = format_question(q1_prompt, correct_theme,
q1_distractors)

    # ... (Question 2 and 3 construction similarly) ...
```

```
return f"{q1}\n\n{q2}\n\n{q3}"
```

What this code does:

- **Extract Story Details (Lines 2-5 of `dynamic_fallback_questions`):** Before formulating questions, the function analyzes the story text to pull out key elements:
    - `extract_protagonist(story)`: Finds the name of the protagonist by looking for a pattern "named <Name>" in the first sentence. If the story starts with "There was a boy named Alex who...", this function will return "Alex". It uses a regex to find a capitalized name after the word "named". If found, that name is considered the protagonist.
    - `extract_characters(story)`: Scans the story for all capitalized words (which usually signify names) and uses a frequency count (excluding common words like "The", "And", "He", etc.) to list characters by frequency. The most frequent name likely is the main character. This provides a list of character names mentioned in the story.
    - `extract_themes(story)`: Checks for certain keywords in the story (like "space", "friend", "adventure", "lesson") to determine the themes. For each keyword found, it appends a corresponding theme (e.g., if "friend" is in text, theme "friendship" is added). This is a simple heuristic to guess the story's themes.
    - `extract_lesson(story)`: Attempts to find a moral or lesson by looking for sentences that contain "learn" or "lesson". It returns the last such sentence or a default message if none found. For example, if the story contains "Alice learned that honesty is the best policy.", that sentence might be returned as the lesson.
- These extraction functions use straightforward NLP techniques (regex, keyword search, frequency counts). While not as advanced as using a model to parse the story, they are efficient and ensure the questions generated are grounded in actual story content (names, themes, moral).

- **Question Template Selection (Lines 7-12 for Q1):** The code defines a few templates for each question and randomly picks one to vary the phrasing. For Question 1 (Q1) about the theme, `theme_templates` offers different ways to ask about the main theme. `random.choice(theme_templates)` selects one, so the question might be phrased as "What is the main theme of the story?" or alternatively "Which theme best represents the narrative?" etc. This adds variety to the questions so they don't appear in the same wording every time.

- **Determine Correct Answer (Lines 13-15 for Q1):** It then determines the correct answer for the theme question. `themes` (from `extract_themes`) is a list of theme keywords found. If any themes were found, it joins them with " and " (in case more than one theme is identified). If no theme was detected by the simple logic, it falls back to `"learning"` as a generic theme. This string becomes `correct_theme`, the correct answer to Q1.

- **Prepare Distractors (Lines 16-18 for Q1):** A list of plausible but incorrect options (`q1_distractors`) is provided. They are general themes that are clearly different from the actual theme (e.g., if the story was about friendship, distractors might be unrelated topics like "sports and competition"). For Q1, the distractors are hardcoded options covering a range of unrelated topics. The code will randomly sample 3 of them (or if fewer are provided, pad them with placeholder options "Option X", etc., handled in `format_question`). In this case, exactly 4 distractors are given and 3 will be chosen randomly, adding unpredictability to which wrong answers appear.

- **Format the Question (Line 19 for Q1):** It calls `format_question(q1_prompt, correct_theme, q1_distractors)`. The `format_question` function (first snippet) then handles shuffling and labeling the answers:

    - It ensures there are exactly 3 wrong options. If more were provided, it randomly picks 3 to use (line 5 of `format_question` uses `random.sample`). If fewer, it appends dummy "Option X/Y/Z" (lines 3-5) to reach 3 distractors.

    - It then combines the wrong options with the correct answer into an `options` list (line 7).

- `random.shuffle(options)` randomizes the order of the correct answer among the wrong ones (line 8). This is crucial for fairness – the correct answer shouldn't always be, say, option D.
- It prepares labels "A", "B", "C", "D" for up to 4 options (line 9) and finds which letter corresponds to the correct answer (line 10, by finding the index of `correct_answer` in the shuffled list).

It then constructs the options text, like:

```
A) [Option 1]
B) [Option 2]
C) [Option 3]
D) [Option 4]
```

(lines 11-12 use a comprehension to join each option with its letter).

Finally, it concatenates the question prompt, the options text, and a line indicating the correct answer letter (line 13). The result is a formatted multiple-choice question string.

For example, Q1 might turn into:

```
What is the main theme of the story?
A) friendship
B) sports and competition
C) cooking and baking
D) weather and seasons
Correct Answer: A
```

- (if "friendship" was identified as the theme and ends up as option A after shuffling).

- **Repeat for Q2 and Q3:** The code (not fully shown above) similarly constructs:

    - **Q2 (Primary Character):** Uses `character_templates` (questions like "Who is the primary character...?"). It decides the correct character: if `extract_protagonist` found a name, that is used; otherwise if `characters` list is not empty, the first name from that list is used; if no names found at all, it defaults to a generic "The main character". Distractors for Q2 are generic character descriptions like "a mysterious stranger", "an unknown visitor", etc. This way, even if the story doesn't have multiple named characters, the question can still be asked with filler wrong answers. Then `format_question` is used to assemble the multiple-choice question.

    - **Q3 (Lesson/Moral):** Uses `lesson_templates` (e.g., "What lesson did the characters learn...?"). For the correct answer, it takes the full sentence from `extract_lesson` if one was found and is not just "No explicit lesson found." (the code ensures it doesn't truncate it, line 42-45 in the search result shows if lesson exists, `correct_lesson = lesson`). If no lesson sentence was found in the story, it falls back to a positive, plausible generic lesson "understanding and growth". Distractors for the lesson are clearly incorrect or negative lessons like "always be silent", "do nothing", which likely do not fit the story's events. Again, `format_question` combines them.

- **Return Combined Questions (Line 20):** Finally, the function returns a string that contains Q1, Q2, Q3 separated by blank lines. Each question already includes the "Correct Answer: X" at the end for reference.

Gamification and rewards aspect:

- The generation of questions transforms the story into a mini-quiz. This is the gamification element: after reading the story, the user can answer these questions to test their comprehension. The system can use this to reward the user in various ways (even though the provided code simply outputs the questions with the correct answer marked). For example, in the app's front-end, the correct answer line might be hidden or used to verify the user's choice. If the user selects the right answer, the app could award points, stars, or progress towards a goal. This reward loop encourages users to engage more deeply with the content.

- By making questions dynamic (derived from the story's actual content), the game is **adaptive** and **relevant**. If the story was about space, the theme question will be about space (and not some unrelated theme). This relevancy is key to an educational reward system – it ensures the user is being quizzed on what they just experienced, reinforcing memory and understanding. It's a personalized quiz generated on the fly by AI.

- The use of randomness in question phrasing and distractors is another subtle gamification touch. It means that even if a user reads the same story twice, the questions might be worded differently or the wrong options might shuffle, preventing the experience from becoming too rote or predictable. It adds replay value and reduces the chance that answers can be memorized by position or phrasing.

- The **extraction functions** (protagonist, characters, themes, lesson) ensure that the content of the questions is grounded in facts from the story, which is critical for fairness and educational value. They effectively mine the AI-generated story for information to ask about. This combination of AI-generated story and rule-based extraction for question generation is a clever design: it avoids needing a more complex AI to *understand* the story (which could be error-prone) by using simple methods sufficient for the task. For example, just searching for keywords like "learned" to find a lesson is usually enough to get a moral of the story.

Contribution to larger app ecosystem:

- The gamification module increases user engagement. It turns a storytelling app into an interactive learning tool. Especially for children, turning reading into a game can motivate them to focus and enjoy the process. Parents/teachers can also use the quiz results to gauge comprehension.

- It also provides a form of **feedback loop**. If the system logs which questions users get wrong, it could infer which parts of the story the user didn't understand or which concepts might need more reinforcement. In a more advanced setting, the app could adapt future stories or questions based on past performance (e.g., focusing on themes the user struggles with, or giving easier questions if many were answered incorrectly to build confidence).

- Even though the code provided doesn't explicitly implement a scoring or tracking system for rewards, it lays the groundwork. The presence of the correct answer in the output suggests that either the frontend will parse it to give immediate feedback, or the app could hide it and only reveal it after the user attempts an answer (to simulate a quiz game). Either way, the reward could simply be the satisfaction of knowing the correct answer or something like a "Well done!" message; in a more gamified app, points could be assigned for each correct answer.

In essence, the gamification and rewards system component uses AI (the story content and some NLP parsing) to automatically create educational quizzes, making the storytelling session interactive. This increases the value of the app as both entertainment and a learning tool, and encourages repeat usage (as users might want to read more stories and answer questions to earn rewards or improve their score).

7.1.6 Large Language Model (LLM) Query Function

To streamline the usage of the AI models and algorithms in the app, a high-level **LLM query function** brings everything together. In this system, that function is `generate_story_and_questions`. This serves as a single call to generate a new story and a set of questions for that story, hiding the complexity of multiple steps. Here is the code for this function:

python

CopyEdit

```
def generate_story_and_questions(theme, reading_level):

    story = generate_story(theme, reading_level)
```

```
    questions = dynamic_fallback_questions(story)

    return {"story": story, "questions": questions}
```

What this code does:

- It accepts the `theme` and `reading_level` as input parameters (as provided by the user or calling code).
- It first calls `generate_story(theme, reading_level)`. This invokes the story generation algorithm (section 7.1.2) to produce a story text using the LLM. Under the hood, this function uses the `story_model` (GPT-2) with the prompt engineering and generation parameters we described. The result is a complete story in text form.
- Next, it calls `dynamic_fallback_questions(story)` with the generated story text. This triggers the question generation logic (section 7.1.5). It analyzes the story with the extraction functions and formulates three multiple-choice questions.
- It then returns a dictionary containing both the story and the questions: `{"story": story, "questions": questions}`. The `story` value is a string of the narrative, and `questions` is a formatted string containing all three questions (with answers).

How it supports the functionality:

- **Single API for content generation:** This function is likely what an API endpoint in the app would call to get results for a user request. It bundles the two major tasks (story generation and Q&A generation) into one, so the front-end or other parts of the system do not need to manage multiple calls or intermediate data. For example, when the user selects a theme and level and hits "Start", the app's backend can call `generate_story_and_questions("space adventure", "third-grade")` and get back everything needed to display the story and quiz.
- **Orchestration of LLM usage:** Notice that this function itself does not contain any heavy logic – it delegates to the specialized functions. It orchestrates the sequence: *first query the story model, then use the story to query/generate the questions.* In doing so, it ensures that the question generation has the full story available. This design decouples components (the story generator doesn't need to know about questions and vice versa), but `generate_story_and_questions` acts as the conductor ensuring data flows from one to the other in the right order.

- **Abstraction and ease of use:** For a developer using this system or for integration with a web service, `generate_story_and_questions` provides an easy interface. They don't have to remember to call `generate_story` then `dynamic_fallback_questions` – a single call handles it. This reduces potential errors (like forgetting to generate questions, or accidentally using an old story with new questions, etc.) and makes the feature accessible via one function. In a larger context, this could be part of a class or module method that the web route calls, making the codebase cleaner.

- **Consistency of data:** By generating the story and questions in one go, it can ensure they are consistent (the questions directly correspond to the freshly generated story). If these were separate calls, there's a risk (albeit small) that if the story were regenerated, the questions might not match the story. Here it's atomic in concept: get story, then questions for that story, and use them together.

Note on LLM usage for questions: The system loads a `question_model` (GPT-2 model for question generation) as seen in section 7.1.1, but the code uses `dynamic_fallback_questions` instead of an LLM to generate questions. This suggests that either:

- a) The question model could be an alternative approach (perhaps intended to take the story as input and generate questions directly via another neural network). The naming "fallback" implies that using the LLM for questions might have been the primary plan, with this rule-based approach as a fallback if the model's output wasn't reliable enough.

- b) For now, they chose the deterministic approach for questions for simplicity or consistency.

In either case, `generate_story_and_questions` is structured in a way that it could easily switch to using the question model. For instance, if one wanted to use the question generation model, this function could be modified to:

python

CopyEdit

```
questions = generate_questions_via_model(story)
```

where `generate_questions_via_model` would tokenize the story and use `question_model.generate()` to produce question text. The current design allows such a swap without changing how the rest of the app calls the function.

Contribution to the larger ecosystem:

- This function essentially **exposes the AI functionality as a service**. In a microservice architecture, one could have an endpoint that directly maps to this function. E.g., a HTTP POST to `/generate` with a JSON body `{"theme": "space", "reading_level": "third-grade"}` could internally call `generate_story_and_questions` and return the result JSON to the client. This means the complexities of tokenization, model inference, and question logic are completely hidden behind a simple API contract (input: theme & level, output: story & questions).

- It also helps with **maintainability and testing**. Each sub-component (story generation, question generation) can be tested in isolation, and this wrapper can be tested to ensure the integration works. If the team updates the story model or adjusts question templates, those changes are localized and `generate_story_and_questions` remains a stable interface.

- From a performance standpoint, combining the operations is convenient. The story generation is the heavy LLM call. The question generation here is relatively light (string processing and random choices). So the overall function call is dominated by the one model inference. If question generation were also an LLM call, this function would manage two sequential model inferences. That could double the response time, which might be a consideration. In such a case, an optimization could be to offload question generation to run in parallel if possible, or to pre-generate questions using the story context as it's being generated (advanced use). But since they opted for a non-LLM approach for questions, the performance is quite good: one model call per story request.

- Finally, this function is what truly delivers the **interactive storytelling experience** to the user: a story to read and questions to answer. It exemplifies how multiple AI components can be composed to create a richer application. Without the questions, it would just be story generation; without the story, there's nothing to quiz on. By having a single function produce both, it guarantees that whenever a story is made, the interactive part comes with it.

In summary, the `generate_story_and_questions` function is the high-level interface for the AI storytelling system. It ties together the NLP model's story generation and the gamified Q&A generation into one cohesive operation. This design makes the system easy to use (for developers and by extension the end-users) and ensures consistency and integration between the story content and the interactive elements, fulfilling the promise of an AI-powered interactive storytelling app.

7.1.7 Response Handling Functions

```python
import os
import time
import torch
from fastapi import FastAPI, HTTPException
from fastapi.responses import JSONResponse
from pydantic import BaseModel
import story_generator
from diffusers import DiffusionPipeline
from PIL import Image
import io
import base64

app = FastAPI()

# Set Hugging Face cache directories
os.environ["HF_HOME"] = "/tmp/huggingface"
os.environ["TRANSFORMERS_CACHE"] = "/tmp/huggingface"  # Deprecated but still used for now
os.environ["HF_HUB_CACHE"] = "/tmp/huggingface"

# Enable GPU if available
device = "cuda" if torch.cuda.is_available() else "cpu"

# Load image generation model
IMAGE_MODEL = "lykon/dreamshaper-8"
pipeline = DiffusionPipeline.from_pretrained(
    IMAGE_MODEL,
    torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32
).to(device)

# Define request schema
class StoryRequest(BaseModel):
    theme: str
    reading_level: str

def generate_images(prompts, max_retries=3, delay=2):
    """
    Attempts to generate images in batch. If an error related to
    "index 16 is out of bounds" occurs, it retries for up to max_retries.
    If all attempts fail, it falls back to generating images sequentially.
    """
    for attempt in range(max_retries):
        try:
            print(f"Batched image generation attempt {attempt+1}...")
            results = pipeline(
                prompt=prompts,
                num_inference_steps=15,
                height=768,
                width=768
            ).images
            return results
        except Exception as e:
            if "index 16 is out of bounds" in str(e):
                print(f"Attempt {attempt+1} failed with error: {e}")
                time.sleep(delay)
            else:
                raise e
    # Fallback to sequential generation
    print("Falling back to sequential image generation...")
    images = []
    for i, prompt in enumerate(prompts):
        try:
            print(f"Sequential generation for prompt {i+1}...")
            image = pipeline(
                prompt=prompt,
                num_inference_steps=15,
                height=768,
                width=768
            ).images[0]
            images.append(image)
        except Exception as e:
            print(f"Error in sequential generation for prompt {i+1}: {e}")
            raise e
    return images

@app.post("/generate_story_questions_images")
def generate_story_questions_images(request: StoryRequest):
    """
    Generates a story, dynamic questions, and cartoonish storybook images.
    """
    try:
        print(f"🎭 Generating story for theme: {request.theme} and level: {request.reading_level}")
        # Generate story and questions using the story_generator module
        story_result = story_generator.generate_story_and_questions(request.theme,
request.reading_level)
        story_text = story_result.get("story", "").strip()
        questions = story_result.get("questions", "").strip()
        if not story_text:
            raise HTTPException(status_code=500, detail="Story generation failed.")

        # Split the story into up to 6 paragraphs
        paragraphs = [p.strip() for p in story_text.split("\n") if p.strip()][:6]

        # Build a list of prompts for batched image generation
        prompts = [
            (
                f"Children's storybook illustration of: {p}. "
                "Soft pastel colors, hand-drawn style, friendly characters, warm lighting, "
                "fantasy setting, watercolor texture, storybook illustration, beautiful composition."
            )
            for p in paragraphs
        ]
        print(f"Generating images for {len(prompts)} paragraphs concurrently...")

        # Use the retry mechanism for image generation
        results = generate_images(prompts, max_retries=3, delay=2)

        # Convert each generated image to Base64
        images = []
        for image in results:
            img_byte_arr = io.BytesIO()
            image.save(img_byte_arr, format="PNG")
            img_byte_arr.seek(0)
            base64_image = base64.b64encode(img_byte_arr.getvalue()).decode("utf-8")
            images.append(base64_image)

        return JSONResponse(content={
            "theme": request.theme,
            "reading_level": request.reading_level,
            "story": story_text,
            "questions": questions,
            "images": images
        })
    except Exception as e:
        print(f"❌ Error generating story/questions/images: {e}")
        raise HTTPException(status_code=500, detail=str(e))

@app.get("/")
def home():
    return {"message": "🎉 Welcome to the Story, Question & Image API!"}
```

This FastAPI application integrates a story-and-image generation pipeline that combines text generation and image synthesis. The app uses a Pydantic model to validate incoming requests, which include a theme and a reading level. Based on these inputs, it invokes a separate module (story_generator) to generate a story along with corresponding dynamic questions. Environment variables are set to use a designated cache directory for Hugging Face models, and the system automatically selects a GPU if available. The generated story is then split into paragraphs, which serve as prompts for a Diffusers pipeline that produces cartoonish, storybook-style images. This structure enables a comprehensive multimedia output, integrating narrative text, comprehension questions, and visuals.

For image generation, the application employs a batched processing strategy to concurrently generate images for all story paragraphs. It includes a robust retry mechanism that catches specific errors (e.g., "index 16 is out of bounds") and retries up to three times before falling back to sequential image generation. Each generated image is converted to a Base64-encoded string for easy transport via JSON. The final response, returned as a JSON object, contains the theme, reading level, the generated story, dynamic questions, and the list of images. The application also defines a basic home route for status checks. This design not only demonstrates the integration of multiple generative models but also shows how to handle potential runtime errors gracefully in a production setting.

7.1.8 Main Execution

```python
import torch
from transformers import GPT2Tokenizer, GPT2LMHeadModel
import random
import re

# Set Hugging Face cache directory
CACHE_DIR = "/tmp/huggingface"

# ------------------------
# Load Story Generation Model
# ------------------------
STORY_MODEL_NAME = "abdalraheemdmd/story-api"
story_tokenizer = GPT2Tokenizer.from_pretrained(STORY_MODEL_NAME, cache_dir=CACHE_DIR)
story_model = GPT2LMHeadModel.from_pretrained(STORY_MODEL_NAME, cache_dir=CACHE_DIR)

# ------------------------
# Load Question Generation Model
# ------------------------
QUESTION_MODEL_NAME = "abdalraheemdmd/question-gene"
question_tokenizer = GPT2Tokenizer.from_pretrained(QUESTION_MODEL_NAME, cache_dir=CACHE_DIR)
question_model = GPT2LMHeadModel.from_pretrained(QUESTION_MODEL_NAME, cache_dir=CACHE_DIR)

# Ensure tokenizers have a pad token
if story_tokenizer.pad_token_id is None:
    story_tokenizer.pad_token_id = story_tokenizer.eos_token_id
if question_tokenizer.pad_token_id is None:
    question_tokenizer.pad_token_id = question_tokenizer.eos_token_id

def generate_story(theme, reading_level, max_new_tokens=400, temperature=0.7):
    """Generates a story based on the provided theme and reading level."""
    prompt = f"A {reading_level} story about {theme}:"
    input_ids = story_tokenizer(prompt, return_tensors="pt").input_ids
    with torch.no_grad():
        output = story_model.generate(
            input_ids,
            max_new_tokens=max_new_tokens,
            temperature=temperature,
            top_k=20,
            top_p=0.7,
            do_sample=True,
            early_stopping=True,
            pad_token_id=story_tokenizer.pad_token_id,
            eos_token_id=story_tokenizer.eos_token_id,
            attention_mask=input_ids.ne(story_tokenizer.pad_token_id)
        )
    return story_tokenizer.decode(output[0], skip_special_tokens=True)

def extract_protagonist(story):
    """
    Attempts to extract the protagonist from the first sentence by searching for the pattern "named
<Name>".
    Returns the first matched name, if available.
    """
    sentences = re.split(r'\.|\n', story)
    if sentences:
        m = re.search(r"named\s+([A-Z][a-z]+)", sentences[0])
        if m:
            return m.group(1)
    return None

def extract_characters(story):
    """
    Extracts potential character names from the story using a frequency count on capitalized words.
    Filters out common stopwords so that the most frequently mentioned name is likely the main
character.
    """
    words = re.findall(r'\b[A-Z][a-zA-Z]+\b', story)
    stopwords = {"The", "A", "An", "And", "But", "Suddenly", "Quickly", "However", "Well",
                 "They", "I", "He", "She", "It", "When", "Where", "Dr", "Mr"}
    filtered = [w for w in words if w not in stopwords and len(w) > 2]
    if not filtered:
        return []
    freq = {}
    for word in filtered:
        freq[word] = freq.get(word, 0) + 1
    sorted_chars = sorted(freq.items(), key=lambda x: x[1], reverse=True)
    return [item[0] for item in sorted_chars]

def extract_themes(story):
    """Extracts themes from the story based on keyword matching."""
    themes = []
    story_lower = story.lower()
    if "space" in story_lower:
        themes.append("space")
    if "adventure" in story_lower:
        themes.append("adventure")
    if "friend" in story_lower:
        themes.append("friendship")
    if "learn" in story_lower or "lesson" in story_lower:
        themes.append("learning")
    return themes

def extract_lesson(story):
    """
    Attempts to extract a lesson or moral from the story by finding sentences
    containing keywords like "learn" or "lesson". Returns the last matching sentence.
    """
    sentences = re.split(r'\.|\n', story)
    lesson_sentences = [
        s.strip() for s in sentences
        if ("learn" in s.lower() or "lesson" in s.lower()) and len(s.strip()) > 20
    ]
```

```python
if lesson_sentences:
        return lesson_sentences[-1]
    else:
        return "No explicit lesson found."

def format_question(question_prompt, correct_answer, distractors):
    """
    Combines the correct answer with three distractors, shuffles the options,
    and formats the question as a multiple-choice question.
    """
    # Ensure exactly 3 distractors are available
    if len(distractors) < 3:
        default_distractors = ["Option X", "Option Y", "Option Z"]
        while len(distractors) < 3:
            distractors.append(default_distractors[len(distractors) % len(default_distractors)])
    else:
        distractors = random.sample(distractors, 3)
    options = distractors + [correct_answer]
    random.shuffle(options)
    letters = ["A", "B", "C", "D"]
    correct_letter = letters[options.index(correct_answer)]
    options_text = "\n".join(f"{letters[i]}) {option}" for i, option in enumerate(options))
    question_text = f"{question_prompt}\n{options_text}\nCorrect Answer: {correct_letter}"
    return question_text

def dynamic_fallback_questions(story):
    """
    Generates three multiple-choice questions based on dynamic story content.
    Each question uses a randomly chosen template and shuffles its options.
    """
    protagonist = extract_protagonist(story)
    characters = extract_characters(story)
    themes = extract_themes(story)
    lesson = extract_lesson(story)

    # --- Question 1: Theme ---
    theme_templates = [
        "What is the main theme of the story?",
        "Which theme best represents the narrative?",
        "What subject is central to the story?"
    ]
    q1_prompt = random.choice(theme_templates)
    correct_theme = " and ".join(themes) if themes else "learning"
    q1_distractors = ["sports and competition", "cooking and baking", "weather and seasons",
"technology and innovation"]
    q1 = format_question(q1_prompt, correct_theme, q1_distractors)

    # --- Question 2: Primary Character ---
    character_templates = [
        "Who is the primary character in the story?",
        "Which character drives the main action in the narrative?",
        "Who is the central figure in the story?"
    ]
    q2_prompt = random.choice(character_templates)
    if protagonist:
        correct_character = protagonist
    elif characters:
        correct_character = characters[0]
    else:
        correct_character = "The main character"
    q2_distractors = ["a mysterious stranger", "an unknown visitor", "a supporting character", "a
sidekick"]
    q2 = format_question(q2_prompt, correct_character, q2_distractors)

    # --- Question 3: Lesson/Moral ---
    lesson_templates = [
        "What lesson did the characters learn by the end of the story?",
        "What moral can be inferred from the narrative?",
        "What is the key takeaway from the story?"
    ]
    q3_prompt = random.choice(lesson_templates)
    if lesson and lesson != "No explicit lesson found.":
        correct_lesson = lesson  # full sentence without truncation
    else:
        correct_lesson = "understanding and growth"
    q3_distractors = ["always be silent", "never try new things", "do nothing", "ignore opportunities"]
    q3 = format_question(q3_prompt, correct_lesson, q3_distractors)

    return f"{q1}\n\n{q2}\n\n{q3}"

def generate_story_and_questions(theme, reading_level):
    """
    Generates a story using the story generation model and then creates dynamic,
    multiple-choice questions based on that story.
    """
    story = generate_story(theme, reading_level)
    questions = dynamic_fallback_questions(story)
    return {"story": story, "questions": questions}

# Alias for backward compatibility
create_fallback_questions = dynamic_fallback_questions
```

his code implements an automated content generation system using pre-trained GPT-2–based models from Hugging Face. It first sets up the environment by importing necessary libraries, establishing a cache directory, and loading two models: one for story generation and another for question generation. The story-generation function constructs a prompt based on a given theme and reading level (e.g., "A beginner story about fantasy:") and generates a story using sampling parameters like temperature, top-k, and top-p. The generated tokens are then decoded into coherent text. In parallel, the code includes functions that use regular expressions and frequency analysis to extract key narrative elements from the story, such as the protagonist, other characters, thematic keywords, and the lesson or moral.

Subsequently, the system dynamically generates multiple-choice questions based on the extracted content. A dedicated function, `format_question`, combines a correct answer with three distractors, shuffles the options, and formats them into a structured question with labeled options (A–D) and the correct answer clearly indicated. The `dynamic_fallback_questions` function employs randomly selected templates to generate questions addressing the story's theme, primary character, and moral. Finally, the integrated function `generate_story_and_questions` produces a dictionary containing both the generated story and its associated questions. This integrated approach demonstrates how modern deep learning models can be combined with traditional text processing techniques to automate and enrich educational content creation.

7.2 User Engagement Analytics

7.3 WebApp

7.3.1 Admin

This is the Admin Dashboard, where administrators can navigate between user and class management sections.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <script type="module" src="firebase-config.js"></script>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Admin Dashboard</title>
    <link rel="stylesheet" href="style.css">
</head>
```

```
<body>
    <div class="container">
        <h1>Admin Dashboard</h1>
        <p>Manage users and classes efficiently.</p>
        <div class="button-group">
            <a href="admin_manage_users.html" class="button">Manage
Users</a>
            <a href="admin_manage_classes.html" class="button">Manage
Classes</a>
        </div>
    </div>
</body>
</html>
```

This CSS file styles the main page of the Admin Website for a clean and professional look.

```
@import
url('https://fonts.googleapis.com/css2?family=Inter:wght@300;400;600&displ
ay=swap');

body {
    font-family: 'Inter', sans-serif;
    background: linear-gradient(135deg, #1e3c72 0%, #2a5298 100%);
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
    color: white;
}

.container {
    text-align: center;
    background: rgba(255, 255, 255, 0.1);
    padding: 40px;
    border-radius: 12px;
    backdrop-filter: blur(10px);
    box-shadow: 0 8px 16px rgba(0, 0, 0, 0.2);
```

```
    width: 90%;
    max-width: 400px;
}

h1 {
    margin-bottom: 10px;
    font-weight: 600;
}

p {
    margin-bottom: 20px;
    font-size: 16px;
    opacity: 0.8;
}

.button-group {
    display: flex;
    flex-direction: column;
    gap: 15px;
}

.button {
    display: block;
    padding: 15px;
    text-decoration: none;
    color: white;
    background: linear-gradient(90deg, #ff512f, #dd2476);
    border-radius: 8px;
    transition: all 0.3s ease-in-out;
    font-size: 16px;
    font-weight: 600;
}

.button:hover {
    transform: scale(1.05);
    box-shadow: 0 4px 12px rgba(255, 81, 47, 0.3);
}
```

This page allows admins to add, remove, and manage users within the system.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <script type="module" src="firebase-config.js"></script>
    <script type="module" src="admin_manage_users.js"></script>

    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Manage Parents and Teachers</title>
    <link rel="stylesheet" href="admin_manage_users.css">
</head>
<body>
    <div class="container">
        <h1>Manage Parents and Teachers</h1>


        <input type="email" id="userEmail" placeholder="Enter
Parent/Teacher Email">


        <select id="userRole">
            <option value="Parent" selected>Parent</option>
            <option value="Teacher">Teacher</option>
        </select>


        <button onclick="addParent()">Add Parent/Teacher</button>


        <ul id="userList" class="scrollable-list"></ul>

        <a href="index.html" class="back-button">← Back to Home</a>
    </div>
</body>
</html>
```

This CSS file styles the Manage Users page, ensuring a structured and user-friendly interface.

```
@import
url('https://fonts.googleapis.com/css2?family=Inter:wght@300;400;600&displ
ay=swap');

body {
    font-family: 'Inter', sans-serif;
    background: linear-gradient(135deg, #1e3c72 0%, #2a5298 100%);
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
    color: white;
}

.container {
    text-align: center;
    background: rgba(255, 255, 255, 0.1);
    padding: 40px;
    border-radius: 12px;
    backdrop-filter: blur(10px);
    box-shadow: 0 8px 16px rgba(0, 0, 0, 0.2);
    width: 90%;
    max-width: 400px;
}

.container input,
.container select {
    display: block;
    width: calc(100% - 4px);
    margin: 14px 0;
    padding: 12px;
    font-size: 16px;
    border-radius: 8px;
    border: none;
    height: 48px;
    box-sizing: border-box;
```

```css
}

h1 {
    margin-bottom: 15px;
    font-weight: 600;
}

input, select, button {
    width: 100%;
    margin: 10px 0;
    padding: 12px;
    font-size: 16px;
    border-radius: 8px;
    border: none;
}

button {
    background: linear-gradient(90deg, #ff512f, #dd2476);
    color: white;
    cursor: pointer;
    font-weight: bold;
    transition: 0.3s ease-in-out;
}

button:hover {
    transform: scale(1.05);
    box-shadow: 0 4px 12px rgba(255, 81, 47, 0.3);
}


ul {
    list-style: none;
    padding: 0;
    height: 240px;
    overflow-y: auto;
    scrollbar-width: none;
    -ms-overflow-style: none;
}
```

```css
ul::-webkit-scrollbar {
    display: none;
}

li {
    background: rgba(255, 255, 255, 0.2);
    padding: 12px;
    margin: 5px 0;
    border-radius: 8px;
    display: flex;
    justify-content: space-between;
    align-items: center;
    transition: background 0.3s ease;
    cursor: pointer;
    height: 80px;
    width: 100%;
    box-sizing: border-box;
}

.remove-btn {
    background-color: #ff4444;
    color: white;
    border: none;
    padding: 10px 0;
    width: 120px;
    height: 50px;
    border-radius: 3px;
    cursor: pointer;
    text-align: center;
}

.remove-btn:hover {
    background-color: #ff2222;
}

.back-button {
    display: inline-block;
    margin-top: 15px;
    text-decoration: none;
    color: white;
```

```
    padding: 12px 20px;
    background: rgba(255, 255, 255, 0.2);
    border-radius: 8px;
    transition: all 0.3s;
}

.back-button:hover {
    background: rgba(255, 255, 255, 0.3);
    transform: scale(1.05);
}
```

This script handles user management functionalities, including adding and removing users from Firebase.

```
import { db } from "./firebase-config.js";
import { collection, addDoc, deleteDoc, doc, onSnapshot } from
"https://www.gstatic.com/firebasejs/10.7.1/firebase-firestore.js";

const userList = document.getElementById("userList");


window.addParent = async function() {
    const email = document.getElementById("userEmail").value;
    const role = document.getElementById("userRole").value || "Parent";

    if (email.trim() !== "") {
        try {

            await addDoc(collection(db, "parents"), { email, role });
            document.getElementById("userEmail").value = "";
            console.log("Parent added successfully!");
        } catch (error) {
            console.error("Error adding parent:", error);
            alert("Error adding parent: " + error.message);
        }
    }
}
```

```
window.removeParent = async function(button) {
    const parentId = button.getAttribute("data-id");

    try {

        await deleteDoc(doc(db, "parents", parentId));
        console.log("Parent removed successfully!");
    } catch (error) {
        console.error("Error removing parent:", error);
        alert("Error removing parent: " + error.message);
    }
}


onSnapshot(collection(db, "parents"), (snapshot) => {
    userList.innerHTML = "";

    snapshot.docs.forEach((doc) => {
        const parent = doc.data();
        const li = document.createElement("li");
        li.innerHTML = `
            ${parent.email} (${parent.role})
            <button class="remove-btn" data-id="${doc.id}"
onclick="removeParent(this)">Remove</button>
        `;
        userList.appendChild(li);
    });
});
```

This page enables admins to manage class enrollments, approve students, and organize class data.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <script type="module" src="firebase-config.js"></script>
    <script type="module" src="admin_manage_classes.js"></script>
```

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Manage Classes</title>
    <link rel="stylesheet" href="admin_manage_classes.css">
</head>
<body>
    <div class="container">
        <h1>Manage Classes</h1>
        <ul id="classList"></ul>
        <a href="index.html" class="back-button">← Back to Home</a>
    </div>
</body>
</html>
```

This CSS file styles the Manage Classes page, improving usability and responsiveness.

```
body {
    font-family: 'Inter', sans-serif;
    background: linear-gradient(135deg, #1e3c72 0%, #2a5298 100%);
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
    color: white;
}


.container {
    text-align: center;
    background: rgba(255, 255, 255, 0.1);
    padding: 40px;
    border-radius: 12px;
    backdrop-filter: blur(10px);
    box-shadow: 0 8px 16px rgba(0, 0, 0, 0.2);
    width: 90%;
    max-width: 400px;
```

```css
}


h1 {
    margin-bottom: 15px;
    font-weight: 600;
}


ul {
    list-style: none;
    padding: 0;
    height: 240px;
    overflow-y: auto;
    scrollbar-width: none;
    -ms-overflow-style: none;
}


ul::-webkit-scrollbar {
    display: none;
}


li {
    background: rgba(255, 255, 255, 0.2);
    padding: 12px;
    margin: 5px 0;
    border-radius: 8px;
    display: flex;
    justify-content: space-between;
    align-items: center;
    transition: background 0.3s ease;
    cursor: pointer;
    height: 80px;
    width: 100%;
    box-sizing: border-box;
}
```

```css
li:hover {
    background: rgba(255, 255, 255, 0.3);
}


button {
    background: linear-gradient(90deg, #ff512f, #dd2476);
    color: white;
    border: none;
    padding: 10px 20px;
    font-size: 16px;
    border-radius: 8px;
    cursor: pointer;
    transition: all 0.3s ease;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.2);
    height: 100%;
    margin-left: 10px;
    display: inline-block;
    min-width: 120px;
    text-align: center;
}


button:hover {
    background: linear-gradient(90deg, #ff512f, #dd2476);
    transform: scale(1.05);
    box-shadow: 0 6px 10px rgba(0, 0, 0, 0.3);
}


button:focus {
    outline: none;
    border: 2px solid #fff;
    box-shadow: 0 0 0 4px rgba(255, 34, 34, 0.6);
}


button:disabled {
    background-color: #ccc;
    cursor: not-allowed;
```

```css
}


.status {
    padding: 5px;
    border-radius: 5px;
}

.pending {
    background: orange;
    color: white;
}

.approved {
    background: green;
    color: white;
}


.back-button {
    display: inline-block;
    margin-top: 15px;
    text-decoration: none;
    color: white;
    padding: 12px 20px;
    background: rgba(255, 255, 255, 0.2);
    border-radius: 8px;
    transition: all 0.3s;
}


.back-button:hover {
    background: rgba(255, 255, 255, 0.3);
    transform: scale(1.05);
}
```

This script handles class management operations, including student approvals and removals.

```
import { db, collection, getDocs, doc, updateDoc } from
"./firebase-config.js";

document.addEventListener("DOMContentLoaded", async () => {
    const classList = document.getElementById("classList");

    if (!classList) {
        console.error("❌ ERROR: Element with ID 'classList' not found in
HTML.");
        return;
    }


    const parentsRef = collection(db, "parents");


    async function fetchParents() {
        classList.innerHTML = "";
        try {
            const querySnapshot = await getDocs(parentsRef);
            querySnapshot.forEach((parentDoc) => {
                const parentId = parentDoc.id;


                fetchChildren(parentId, parentDoc.data().children);
            });
        } catch (error) {
            console.error("❌ ERROR fetching parents:", error);
        }
    }


    async function fetchChildren(parentId, childrenData) {
        const childrenList = document.getElementById("classList");

        if (!childrenList) return;

        try {
```

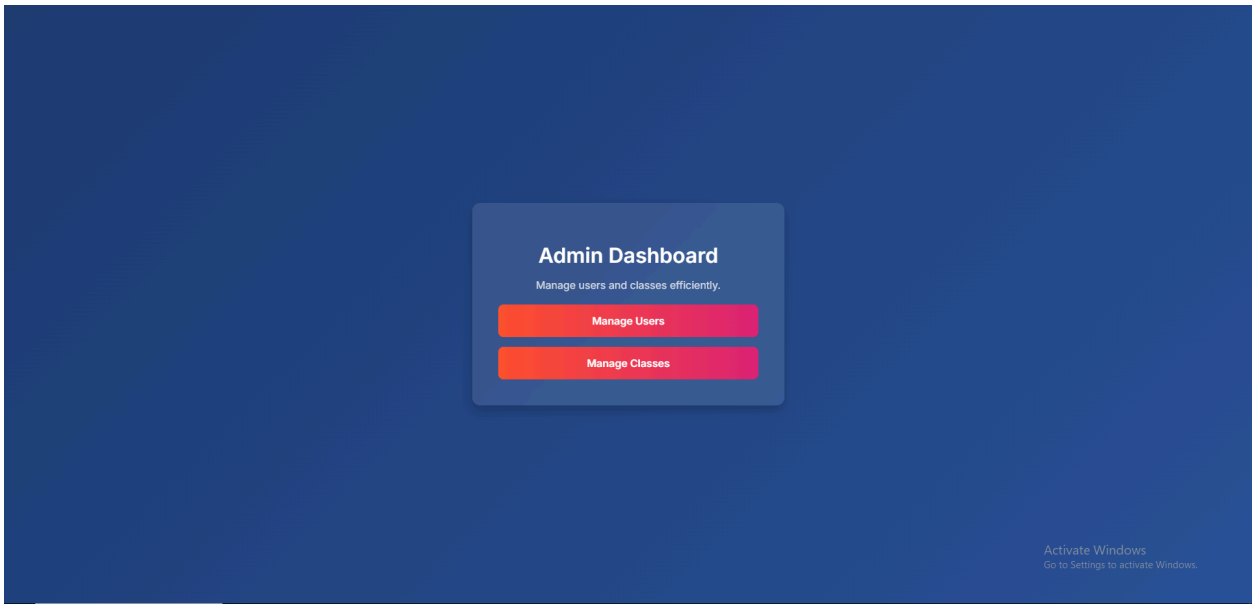
```
            for (const childKey in childrenData) {
                const childData = childrenData[childKey];
                const childName = childData.name;
                const classCode = childData.class_code;


                if (classCode) {
                    const li = document.createElement("li");

                    li.innerHTML = `${childName} (Class Code:
${classCode})
                        <button onclick="removeFromClass('${parentId}',
'${childKey}')">Remove from Class</button>`;

                    childrenList.appendChild(li);
                }
            }
        } catch (error) {
            console.error("❌ ERROR fetching children:", error);
        }
    }


    window.removeFromClass = async function (parentId, childKey) {
        try {
            const parentDocRef = doc(db, "parents", parentId);


            await updateDoc(parentDocRef, {
                [`children.${childKey}.class_code`]: null
            });


            fetchParents();
        } catch (error) {
            console.error("❌ ERROR removing class code:", error);
        }
    };

    fetchParents();
```

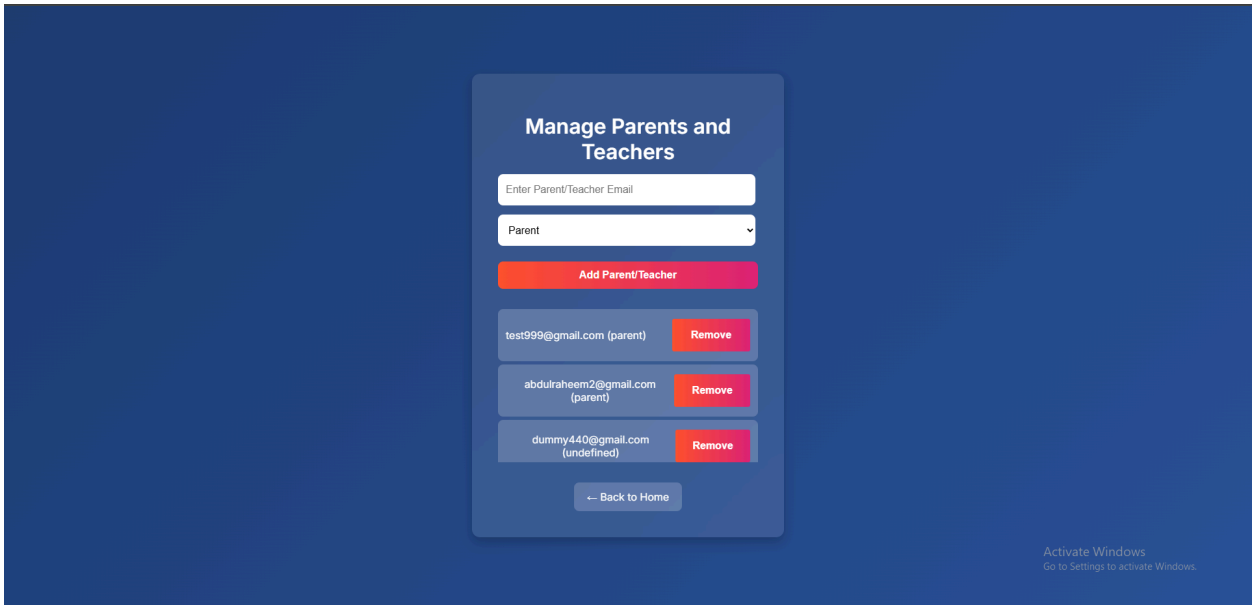
```
});
```

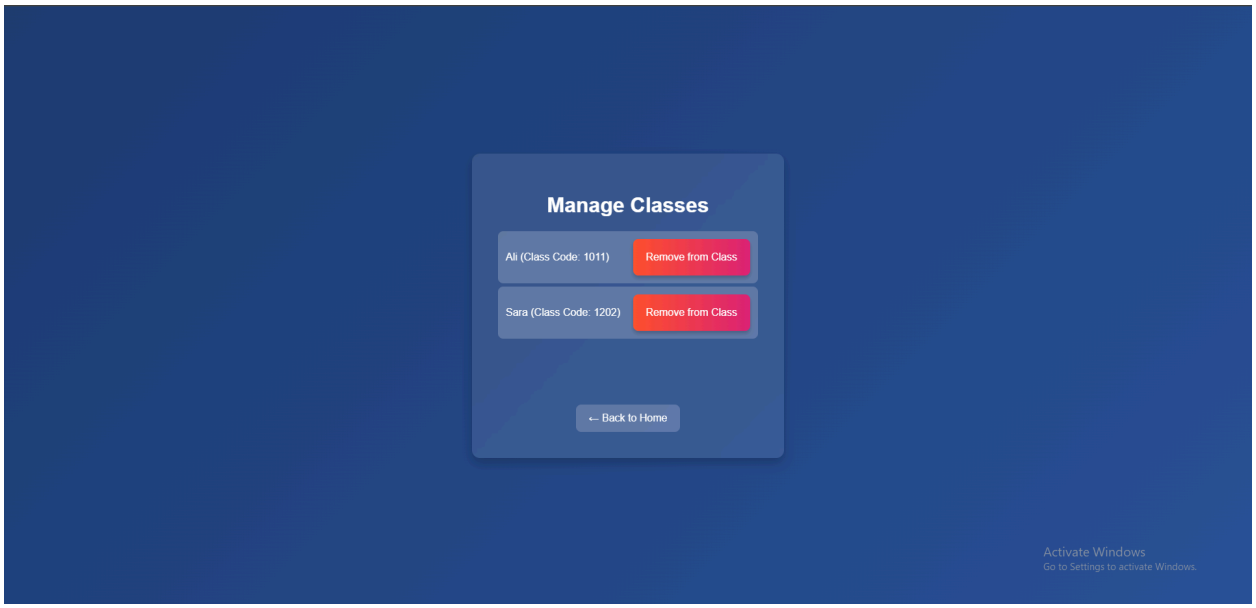
This file contains Firebase configuration settings to connect the Admin Website to the database.

```
import { initializeApp } from
"https://www.gstatic.com/firebasejs/10.7.1/firebase-app.js";
import { getFirestore, collection, addDoc, getDocs, updateDoc, deleteDoc,
doc } from
"https://www.gstatic.com/firebasejs/10.7.1/firebase-firestore.js";

// web app's Firebase configuration
const firebaseConfig = {
  apiKey: "AIzaSyDzU-whegyvaC7dTkmfkzr5MOu5So9HAFc",
  authDomain: "your-story-43a73.firebaseapp.com",
  databaseURL: "https://your-story-43a73-default-rtdb.firebaseio.com",
  projectId: "your-story-43a73",
  storageBucket: "your-story-43a73.appspot.com",
  messagingSenderId: "918519664641",
  appId: "1:918519664641:web:b7fd0d2b1fd2741fb8f000",
  measurementId: "G-XQH82XFN1F"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
const db = getFirestore(app);

// Export Firestore functions
export { db, collection, addDoc, getDocs, updateDoc, deleteDoc, doc };
```

These are screenshots of the Admins Website







7.3.3 LearnMoreAboutUs

This is the HTML structure for the "Learn More About Us" page, defining its layout and content.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=0.5">
    <title>Learn More About Us - Tech Quest</title>
    <link rel="stylesheet" href="lmauu.css">
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link
href="https://fonts.googleapis.com/css2?family=Luckiest+Guy&family=Raleway
:ital,wght@0,100..900;1,100..900&display=swap"
        rel="stylesheet">
</head>

<body>

    <header>
```

```
        <h1>Learn More About Us</h1>
    </header>

    <section class="about container left">
        <div class="text">
            <h2>Who We Are</h2>
            <p>We are Tech Quest, a team of five passionate Computer
Science students from the University of Wollongong in
                Dubai. Our journey began with a shared vision: to leverage
technology to create meaningful, engaging, and
                impactful educational experiences. As part of our
final-year project, we developed an AI-powered interactive
                storytelling and learning platform designed to
revolutionize digital education for young learners.</p>
            <p>With backgrounds in system architecture, AI integration,
software development, and user experience, we
                combined our expertise to craft a seamless learning
environment that adapts to each user's needs. Our
                application fosters personalized learning by integrating
artificial intelligence, natural language
                processing, and dynamic difficulty adjustment to create
immersive and adaptive educational
                experiences.</p>
        </div>
        <div class="image">
            <img src="who_we_are.jpg" alt="Who We Are">
        </div>
    </section>

    <section class="team">
        <h2>Meet Our Team</h2>
        <div class="team-container">
            <div class="member">
                <img src="hassan.png" alt="Hassan Barada">
                <h3>Hassan Barada</h3>
                <p>Backend, Frontend & AI Developer</p>
                <p>Email: hbarada2005@gmail.com</p>
                <p><a
href="https://www.linkedin.com/in/hassan-barada-45238b211">LinkedIn</a></p
>
```

```
            </div>

            <div class="member">
                <img src="abbas.jpg" alt="Abbas Amir">
                <h3>Abbas Amir</h3>
                <p>Backend, Frontend & AI Developer</p>
                <p>Email: abbasamir179@gmail.com</p>
                <p><a
href="https://www.linkedin.com/in/abbasamirr">LinkedIn</a></p>
            </div>

            <div class="member">
                <img src="ajmal.jpg" alt="Ajmal Abdu Razak">
                <h3>Ajmal Abdu Razak</h3>
                <p>Backend Developer</p>
                <p>Email: ajmal.razak8@gmail.com</p>
                <p><a
href="http://www.linkedin.com/in/ajmal-ar-648aa5220">LinkedIn</a></p>
            </div>

            <div class="member">
                <img src="abdulraheem.jpg" alt="Abdulraheem Daoud">
                <h3>Abdulraheem Daoud</h3>
                <p>Backend & AI Developer</p>
                <p>Email: abdalraheemdmd@gmail.com</p>
                <p><a
href="https://ae.linkedin.com/in/abdulraheem-daoud-a87701305">LinkedIn</a>
</p>
            </div>

            <div class="member last">
                <img src="saleh.jpg" alt="Saleh Abu Zuhri">
                <h3>Saleh Abu Zuhri</h3>
                <p>Web Development, Content Writing & App Design</p>
                <p>Email: saleh-wael@hotmail.com</p>
                <p><a
href="https://ae.linkedin.com/in/saleh-abuzuhri">LinkedIn</a></p>
            </div>
        </div>
    </section>
```

```
    <section class="vision container left">
        <div class="image">
            <img src="vision.webp" alt="Our Vision">
        </div>
        <div class="text">
            <h2>Our Mission & Vision</h2>
            <p>At Tech Quest, we believe that education should be
engaging, personalized, and accessible to all. Our mission
                is to bridge the gap between traditional learning methods
and modern digital advancements. By integrating
                AI-driven storytelling with adaptive learning, we empower
students to take control of their learning
                journey in a way that is both fun and effective.</p>
            <p>Our vision is to create an inclusive, scalable, and
innovative educational platform that caters to diverse
                learning styles. We aim to inspire the next generation by
making learning a journey of curiosity, creativity,
                and discovery.</p>
        </div>
    </section>
    <section class="journey container left">
        <div class="text">
            <h2>Our Journey</h2>
            <p>From brainstorming sessions in university classrooms to
late-night coding marathons, our journey has been
                fueled by dedication, teamwork, and an unyielding passion
for innovation. Each of us brought unique
                skills to the table, ensuring that our project was built
on a strong foundation of research, development,
                and rigorous testing.</p>
        <p>Through months of collaboration, we transformed an idea into a
functional product that is ready to make a
            difference in the world of education. Our experience has not
only strengthened our technical skills but also
            reinforced the importance of teamwork, resilience, and a
shared commitment to excellence.</p>
    </div>
    <div class="image">
        <img src="journey.webp" alt="Our Journey">
```

```
    </div>
    </section>
    <section class="contact container left">
        <div class="image">
            <img src="contact.webp" alt="Get in Touch">
        </div>
        <div class="text">
            <h2>Get in Touch</h2>
            <p>We love connecting with like-minded individuals who share
our passion for technology and education. If
                you'd like to learn more about our project, collaborate,
or provide feedback, feel free to reach out to
                us!</p>
            <p>Email: TechQuest@hotmail.com</p>
            <p>Follow us on <a
href="https://www.instagram.com/yourstory_edu?igsh=OTRhYXN2OGlrbW16&utm_so
urce=qr">Instagram</a></p>
        </div>
    </section>

</body>

</html>
```

This CSS file styles the "Learn More About Us" page, ensuring a visually appealing design and responsiveness.

```
body {
    font-family: 'Raleway', sans-serif;
    margin: 0;
    padding: 0;
    background: white;
    color: #333;
    text-align: center;
    overflow-x: hidden;
}
```

```css
header {
    background: #f77d3e;
    color: #f5d9c6;
    padding: 20px 20px;
    font-size: 50px;
    font-weight: 900;
    text-transform: uppercase;
    text-shadow: 4px 4px 15px rgba(0, 0, 0, 0.5);
    letter-spacing: 4px;
    text-align: center;
    animation: fadeIn 0.8s ease-in-out;
    font-family: 'Luckiest Guy', cursive;
}


.container {
    display: flex;
    align-items: center;
    justify-content: space-between;
    max-width: 1200px;
    margin: 30px auto;
    padding: 10px;
    flex-wrap: wrap;
}

.container.left {
    flex-direction: row;
}

.container.right {
    flex-direction: row-reverse;
}


@media (max-width: 768px) {
    .container {
        padding: 10px;
        margin: 10px auto;
    }
```

```css
}


.container.flip {
    flex-direction: row-reverse;
    perspective: 1500px;
    transition: transform 0.5s ease-in-out;
}

.container.flip:hover {
    transform: rotateY(180deg);
}

.text {
    flex: 1;
    text-align: left;
    padding: 20px;
    backface-visibility: hidden;
}

.image {
    flex: 1;
    display: flex;
    justify-content: center;
    backface-visibility: hidden;
}

.image img {
    max-width: 100%;
    height: auto;
    border-radius: 20px;
    box-shadow: 0 10px 20px rgba(0, 0, 0, 0.5);
    max-height: 400px;
}


h2 {
    font-family: 'Luckiest Guy', cursive;
    font-size: 2rem;
    font-weight: 800;
```

```
    text-transform: uppercase;
    letter-spacing: 3px;
    text-shadow: 2px 2px 12px rgba(255, 255, 255, 0.3);
}


p {
    font-size: 1rem;
    max-width: 800px;
    line-height: 1.4;
    opacity: 0.95;
}


.contact {
    text-align: center;
    padding: 40px 20px;
    font-size: 1.2rem;
    font-weight: bold;
    opacity: 0.95;
    text-shadow: 1px 1px 8px rgba(255, 255, 255, 0.3);
}

.contact a {
    color: #fcce33;
    text-decoration: none;
    font-weight: bold;
    transition: color 0.3s ease, text-shadow 0.3s ease;
}

.contact a:hover {
    color: #37bcf4;
    text-shadow: 0px 0px 12px rgba(55, 188, 244, 0.8);
}


@keyframes fadeIn {
    from {
        opacity: 0;
        transform: translateY(-20px);
```

```
    }

    to {
        opacity: 1;
        transform: translateY(0);
    }
}

@keyframes gradientShift {
    0% {
        background-position: 0% 50%;
    }

    100% {
        background-position: 100% 50%;
    }
}


.team {
    padding-bottom: 50px;
}

.team-container {
    display: flex;
    justify-content: center;
    flex-wrap: wrap;
    gap: 20px;
    margin-top: 50px;
    margin-bottom: 20px;
    padding: 20px;
    animation: slideIn 0.8s ease-in-out;
}

.member {
    width: 280px;
    box-sizing: border-box;
    padding: 20px;
    border-radius: 20px;
    box-shadow: 0 8px 15px rgba(0, 0, 0, 0.2);
```

```
    text-align: center;
    background: rgba(255, 255, 255, 0.9);
    margin: 10px;
}

.member img {
    width: 140px;
    height: 140px;
    border-radius: 8px;
    margin-bottom: 15px;
    border: 4px solid white;
    object-fit: cover;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
}

.member h3 {
    margin-top: 10px;
    margin-bottom: 5px;
    font-weight: 600;
}

.member p {
    font-size: 0.9rem;
    margin-bottom: 8px;
}

.member a {
    color: #37bcf4;
    text-decoration: none;
    font-weight: 500;
}

.member a:hover {
    text-decoration: underline;
}

@media (max-width: 600px) {
    .member {
        width: 90%;
        margin: 10px auto;
```

```
    }
    .image img{
        max-height: 350px;
    }
}

@media (max-width: 480px) {
    .image img{
        max-height: 300px;
    }
}

p {
    word-break: break-word;
    hyphens: auto;
}


@media (max-width: 768px) {
    p {
        font-size: 0.95rem;
    }
}
```

8. Test Plan

8.1 Testing Goals

The testing phase's main objective is to ensure the storytelling application functions as expected, providing personalized AI-generated stories for students and enabling teachers to track engagement and progress. Testing will validate system performance, functionality, and integration with Firebase and AWS.

Specific objectives are:

- Ensuring AI-generated stories match the selected themes and student reading levels.
- Making sure that Firebase is properly storing generated stories and user interactions.

- Verifying AWS logs and records student reading progress.
- Testing usability and accessibility for teachers and students.

8.2 Test Plan Scope

In Scope

The following will be examined:

- Story Generation: Testing the ability of AI to generate stories based on student reading levels and themes.
- Data Storage: Making sure that Firebase saves generated stories beneath the appropriate student profiles.
- User Activity Logging: Verifying AWS logs for monitoring story interactions, read time, and student progress.
- Performance Testing: Testing response time for story generation under various network conditions.
- Integration Testing: Providing effortless data migration from Firebase to AWS.
- Usability Testing: Evaluating user experience for both students and educators, focusing on navigability and accessibility.

Out of Scope

Others are outside of the scope of the present test:

- Third-party AI Models: External AI model performance outside of the system's AI generation.
- Advanced Data Analytics: Any predictive analytics beyond simple progress monitoring.
- Offline Capability: The system is implemented to be used online; there is no offline support.

8.3 Test Plan Assumptions and Constraints

Test Plan Assumptions

- The AI model is pre-trained and doesn't need further development.
- Firebase and AWS are set up and running.
- Both teachers and students are given stable internet connections.

Testing Plan Constraints

- Variability in AI-generated content may affect consistency.
- Network delay can influence response times across many environments.
- Usability testing can involve time constraint scheduling with both students and teachers.

8.4 Test Plan Roles and Responsibilities

Specification	Plan Roles	Responsibilities
AI Story Generation	Ensure AI produces relevant, theme-based stories for students.	Abdul Raheem, Abbas
Database Storage	Validate Firebase data storage for user-generated content.	Abbas
User Interaction Logging	Ensure AWS logs track reading progress and engagement.	Ajmal
Performance Testing	Test story generation speed and system scalability.	Abbas, Abdul Raheem
UI/UX Testing	Improve accessibility for students and teachers.	Hassan

8.5 Test Entry Criteria, Approach and Tools

Test Entry Criteria

- The basic application features must work.
- Integrations with Firebase and AWS must be completed.

- Test cases need to be documented and revised.

Test Strategy

- Unit Testing: Test every system component separately like story generation and data storage.
- Integration Testing: Make sure Firebase syncs properly with AWS.
- System Testing: Test real user environment end-to-end functionality.
- User Acceptance Testing (UAT): Collect feedback from students and teachers.

Testing Tools

Tool	Purpose
Postman	API testing to validate data transfer
Firebase Console	Test story storage and retrieval
AWS S3 Logs	Maintain accurate documentation of students progress
Google Forms	Gather usability input educators and students
Xcode	Test iOS app performance, UI responsiveness, and stability

8.6 Testing Forms

Field	Details
Test Case ID	A unique identifier is assigned to each test case (e.g., ST001 for AI-generated story adaptation).
Test Objective	Defines the functionality or feature being evaluated in the test.
Preconditions	Specific conditions that must be met before executing the test.

Test Steps	A sequential list of actions required to perform the test case.
Expected Outcome	The anticipated system response or behavior.
Actual Outcome	The observed behavior during test execution.
Test Status	Indicates whether the test case Passed, Failed, or was Blocked.
Remarks	Additional notes, including identified issues or observations.

8.7 Traceability Matrix

Requirement ID	Test Case ID(s)	Description	Status	Remarks
R001	ST001, ST002	Story generation accuracy and relevance.	Tested (Pass)	AI-generated content matched expected themes.
R002	ST002, ST003	Data storage and retrieval via Firebase.	Tested (Pass)	Stories stored under correct student profiles.
R003	ST003, ST004	Logging and tracking of student interactions.	Tested (Pass)	AWS logs accurately captured progress data.
R004	ST005	System performance under high load.	Tested (Pass)	Response time remained within acceptable limits.
R005	ST006	Role-based access control validation.	Tested (Pass)	Access restricted appropriately based on user roles.

9. Maintenance

In order to keep this interactive storytelling application efficient, scalable, and up to date with changing technology, a maintenance strategy has been laid out, which will help the development team in handling post-lunch updates, bug fixes, and enhancements within the features. Maintenance would include system optimizations, fixes for bugs, and regular updates to enable users to have a smoother experience and long-term functioning. The next section will provide a general overview of the various maintenance activities that are about to be undertaken for keeping and improving the application in future.

9.1 Adaptive Maintenance.

Adaptive maintenance is essential in ensuring that the application remains compatible with technological advancements, external system changes, and evolving user demands. As software environments are constantly changing, one needs to continue modifying the application in order to maintain it at its optimal performance. This includes updates to remain current with Firebase and AWS changes for seamless data storage and retrieval and changes required to operate on new versions of mobile operating systems. The AI algorithms used for story personalization and learning adaptation also need to be updated from time to time to maximize engagement and pedagogic impact. Security practices will also be updated based on latest industry benchmarks so that the information of the users will not remain at risk due to more contemporary threats. In this manner, the application will continue to provide an optimum-quality, smooth, and effortless experience despite inevitable changes from the outside world.

9.2 Perfective Maintenance

Perfective maintenance is all about refining and fine-tuning the usability, functionality, and performance of the application based on users' feedback and analytics. While the teachers and students utilize the platform, information will be gathered to ascertain how to streamline navigation to make it more efficient, improve accessibility, and make reading easier in general. Story suggestions by AI will be continuously refined to deliver more profound and engaging content that responds to the individual user. In addition, features in the user interface will be continually updated to achieve maximum readability, reduce interactions, and facilitate the application to be easily accessible for users of all ages. New interactive story elements will also be added in the future to update the content and provide it with relevance, thereby keeping the students interested. System performance improvements, such as database query optimizations and reduced processing latency, will also be pursued to offer faster response times and

more responsive user interactions. By continuous upgrading and system adjusting of the program, it will remain flexible and be capable of catering to changing students, teachers, and parents' needs.

9.3 Corrective Maintenance

Corrective maintenance is required to detect and rectify system errors, bugs, and functional inconsistencies during the use of the application. It's a systematic process of finding, diagnosing, and fixing software faults that can impact performance, reliability, or security. Routine corrective maintenance activities include debugging AI-generated content to be more relevant and sensible, debugging API issues that affect Firebase-AWS data sync, and debugging user-reported problems such as login problems, response time, or incomplete story content. Routine security vulnerability scanning will also be conducted to discover potential vulnerabilities and implement necessary patches to avoid attacks. Periodic log monitoring will assist in the early identification of anomalies before they turn into critical issues so that they can be resolved by the development team within reasonable time frames. Through the preemptive issue resolution mechanism, the application will deliver a secure, safe, and uninterrupted experience to the users without presenting any form of interruption to learning and interaction.

9.4 Preventative Maintenance

Preventative maintenance is aimed at actively eliminating system failures, performance degradation, and security vulnerabilities before they have a chance to become disruptive. By using regular system audits, scheduled database purges, and automated performance assessments, the application's performance and stability can be ensured in the long run. Preventative solutions will include regular security audits for compliance with data protection requirements, occasional system performance testing to identify inefficiencies, and automated test regimes to identify and address potential software issues prior to their affecting users. Further, monitoring tools will be used to analyze usage patterns and identify anomalies that may indicate underlying issues, facilitating early intervention. By adhering to a preventative maintenance schedule, there will be no likelihood of unscheduled downtime, data leakage, and performance problems, which will make the application highly reliable and secure for all the users accessing it.

9.5 Maintenance Process

Maintenance of Interactive Storytelling and Learning App is carefully crafted to be

long-lasting, scalable, and ideal for the user. The systematic process is designed to cope with updates, fix likely issues, and apply upgrades depending on comments from users and technology advancement. Maintenance is divided into four broad categories: adaptive, perfective, corrective, and preventive maintenance, all working towards the security, stability, and long usability of the platform.

1. **Scheduled Maintenance and Documentation:**
   Planned maintenance activities are conducted for the app's performance, security, and updates. Everything that has been updated, bug-fixed, and optimized is logged in a central system, making it easy to track and make long-term maintenance decisions.

2. **Real-Time System Monitoring and Logging:**
   The app employs monitoring tools to track system performance, user activity, and security breaches. The logs are scanned to reveal trends, reveal anomalies, and fix performance problems before they impact users.

3. **Detailed Testing and Quality Assurance:**
   Each alteration, whether to enhance features, apply security updates, or perform compatibility fixes, is thoroughly tested prior to deployment. This comprises unit testing, integration testing, and user acceptance testing to validate that changes don't introduce new bugs or affect the user interface in a detrimental way.

4. **Phase Rollout for Major Updates:**
   Phased rollout is employed for significant upgrades, such as AI model changes, content enrichment, or security patches. Upgrades are first introduced to a test subset of users for trials and feedback collection before complete release. This reduces risks and allows interruption-free services to all the users.

Through this systematic maintenance process, the Interactive Storytelling and Learning App is maintained in a robust, secure, and continuously improved form for parents, teachers, and students to benefit from. The systematic process ensures that all maintenance dimensions—adaptive, perfective, corrective, and preventive—are handled systematically to ensure the longevity, security, and functionality of the platform.

10. Appendices

10.1 References

Cooney Center, J.G., 2014. *Learning through interactive storytelling: Educational impacts*. New York: Joan Ganz Cooney Center.

Tamim, R.M., Bernard, R.M., Borokhovski, E., Abrami, P.C. and Schmid, R.F., 2011. What forty years of research says about the impact of technology on learning: A second-order

meta-analysis and validation study. *Review of Educational Research*, 81(1), pp.4-28.

Sommerville, I., 2015. *Software Engineering*. 10th ed. Boston: Pearson Education.

Vygotsky, L.S., 1978. *Mind in Society: The Development of Higher Psychological Processes*. Cambridge: Harvard University Press.

Lee, K. and Hutton, J., 2020. AI-driven personalization in education: Theoretical foundations and applications. *Journal of Educational Technology*, 12(3), pp.150-167.

Google Developers, 2023. *Firebase Realtime Database Documentation*. Available at: https://firebase.google.com/docs/database [Accessed 1 November 2024].

Amazon Web Services (AWS), 2023. *AWS Cloud Services Documentation*. Available at: https://aws.amazon.com/documentation/ [Accessed 1 November 2024].

OpenAI, 2024. *Natural Language Processing and Machine Learning with OpenAI Models*. Available at: https://openai.com/resources/ [Accessed 1 November 2024].

European Union, 2018. *General Data Protection Regulation (GDPR)*. Available at: https://gdpr-info.eu/ [Accessed 1 November 2024].

U.S. Federal Trade Commission, 2020. *Children's Online Privacy Protection Rule (COPPA)*. Available at: https://www.ftc.gov/legal-library/browse/rules/childrens-online-privacy-protection-rule-coppa [Accessed 1 November 2024].

1. Author links open overlay panelRichard F. Schmid a, et al. "The Effects of Technology Use in Postsecondary Education: A Meta-Analysis of Classroom Applications." *Computers & Education*, Pergamon, 15 Nov. 2013, www.sciencedirect.com/science/article/abs/pii/S0360131513003072.

Getting a Read on the App Stores, www.joanganzcooneycenter.org/wp-content/uploads/2015/12/jgcc_gettingaread.pdf. Accessed 20 Oct. 2024.

Common Sense Media. (2023). *Epic! Digital Reading Platform for Kids - App Review*. Common Sense Media. Retrieved October 2024, from https://www.commonsensemedia.org.

Tales Untold. (2023). *Tales Untold - Children's Stories*. Tales Untold. Retrieved October 2024, from https://www.talesuntold.com.

StoryJumper. (2023). *Create & Share Your Own Storybooks*. StoryJumper. Retrieved October 2024, from https://www.storyjumper.com.

Smith, J. (2023). *Adaptive Learning in Early Childhood Education: Benefits and Challenges*. *Journal of Educational Technology & Society*, 26(2), 45-60.

Flutter Documentation. Official documentation for Flutter, which will help you with cross-platform app development. https://flutter.dev/docs. Accessed 23 Oct. 2024.

Android Studio. Android Studio is the official IDE for Android development, including for Flutter-based projects. https://developer.android.com/studio. Accessed 23 Oct. 2024.

Firebase Documentation. Firebase provides backend services like authentication, real-time databases, and analytics. https://firebase.google.com/docs. Accessed 23 Oct. 2024.

Google Cloud Platform. Google Cloud offers services like AI/ML integration and cloud storage for scalable backend solutions. https://cloud.google.com/docs. Accessed 23 Oct. 2024.

TensorFlow Documentation. TensorFlow is an open-source platform for machine learning, useful for training AI models. https://www.tensorflow.org/. Accessed 23 Oct. 2024.

Natural Language Processing (NLP) with Hugging Face. Hugging Face offers state-of-the-art NLP models for integrating personalized content. https://huggingface.co/. Accessed 23 Oct. 2024.

PyTorch Documentation. PyTorch is another popular library for building AI models in adaptive storytelling. https://pytorch.org/. Accessed 23 Oct. 2024.

General Data Protection Regulation (GDPR) Guidelines. GDPR outlines privacy laws for handling user data securely. https://gdpr.eu/. Accessed 23 Oct. 2024.

Children's Online Privacy Protection Act (COPPA) Compliance. COPPA governs data privacy for apps targeting children in the U.S. https://www.ftc.gov/business-guidance/privacy-security/children's-privacy. Accessed 23 Oct. 2024.

Material Design Guidelines. Material Design provides guidelines for building intuitive and accessible user interfaces. https://material.io/design. Accessed 23 Oct. 2024.

UserTesting Platform. UserTesting provides a platform for gathering user feedback to improve app features. https://www.usertesting.com/. Accessed 23 Oct. 2024.

For further reference and deeper analysis, the following reports provide essential background and supporting details:

- Planning & Feasibility Analysis
- Design Document (High & Low-Level Design)
- Software Requirements Specification
- Test Case Report

10.2 Skills Tracker

10.2.1 Team Members and Responsibilities

Name	Skill/Role	Tasks	Timeline	Comments
Hassan Barada & Abbas Amir	Front-End Development, AI Integration	Developed intuitive UI, integrated AI-driven story generation, optimized user experience and Firebase Integration	Week 2 – Week 12	Ensured seamless storytelling functionality across platforms.
Abdulraheem Daoud	Back-End & AI Processing	Developed API interactions, optimized AI content generation, and handled system security.	Week 2 – Week 12	Improved AI-driven story adaptation for real-time responses.
Ajmal Abdu Razak	Back-End Development	Managed database transactions, server-side processing using AWS.	Week 4 – Week 12	Optimized data handling for efficient student progress tracking.
Saleh Abu Zuhri	Front-End, Web Development, Planning	Assisted in UI development, led project planning, developed the admin website, and compiled the final project report.	Week 3 – Week 12	Ensured a responsive design, implemented user management features, and finalized project documentation.

10.2.2 Skill Development Progress

Skill Area	Team Member	Training Resources	Completion Status	Notes

AI Storytelling Development	Abbas/Abdul Raheem	OpenAI GPT Docs, NLP & Machine Learning Courses	Completed (Week 8)	Final AI-driven storytelling engine deployed.
Adaptive Learning & NLP Optimization	Abdul Raheem	TensorFlow NLP Guides, Adaptive Learning Research	Completed (Week 9)	Enhancing real-time difficulty adjustments.
Gamification & Rewards System	Abbas	Game Development, Gamification Research	Completed (Week 10)	Testing reward mechanics for better engagement.
Parental & Educator Dashboard	Hassan	API Documentation	Completed (Week 9)	Ensuring smooth data visualization for parents & teachers, including user management tools.
Cross-Platform UI Development	Abbas/Hassan /Saleh	Flutter Docs, Mobile Optimization Resources	Completed (Week 10)	Fully integrated responsive UI for mobile.
Admin Website Development	Saleh	Firebase Docs, Web Development Resources	Completed (Week 10)	Implemented features for adding/removing parents and teachers, and managing student enrollments in classes.

10.2.3 Function Development Progress

Function	Team Member	Status	Notes
AI Story Generation	Abdul Raheem/Abbas	Completed	AI-driven storytelling engine generates adaptive narratives.
User Progress Tracking	Abbas	Completed	Tracks user learning patterns and adjusts content accordingly.

Real-time Story Adaptation	Abdul Raheem	Completed	AI dynamically modifies story elements based on user choices.
Parental Dashboard	Hassan	Completed	Provides real-time learning insights and analytics for parents.
Gamification & Rewards System	Abbas	Completed	Integrated achievement tracking.
Cross-Platform UI Optimization	Abbas/Hassan/Saleh	Completed	Ensuring a seamless user experience across mobile platforms.
Admin Website Management	Saleh	Completed	Allows admins to add/remove parents and teachers and manage student enrollment in classes.

10.3 Meeting Minutes

Semester 1

Week 1

Week 1 (September 18, 2024 – 24 September 2024)

Meeting Details

- **Date & Time:** September 18, 2024, 3:30PM
- **Type:** Online Meeting
- **Duration:** Approximately 1 hour
- **Attendees:**
    - Hassan Barada - Leader
    - Abbas Amir – Scribe
    - Rami Al Sabek – Member
    - Abdul Raheem – Member

Agenda

1. Form the project group and assign roles.

2. Discuss team goals and interests for the capstone project.
3. Plan a structured approach for project idea generation.
4. Set key deadlines and establish expectations for upcoming weeks.

Discussion Points

1. Group Formation and Role Assignment

- The group was officially formed, and roles were assigned as follows:
    - **Hassan Barada:** Leader
    - **Abbas Amir:** Scribe
    - **Rami Al Sabek:** Member
    - **Abdul Raheem:** Member

2. Team Goals and Interests

- Members discussed their individual academic and career interests to align project objectives.
- Key focus areas included creativity, innovation, and practical relevance to the team's strengths.

3. Initial Project Planning

- The team agreed on a phased approach for the project:
    - **Week 1:** Form the group and establish roles.
    - **Week 2:** Brainstorm potential project ideas.
    - **Week 3:** Narrow down ideas and finalize project selection.

4. Structured Approach for Idea Generation

- Each member will research independently and propose multiple project ideas.
- Ideas will be discussed in the next meeting to identify three viable options.
- Members will choose one idea for detailed research.

5. Key Deadlines and Expectations

- All members complete their initial research by the end of Week 2.
- Prepare to present findings and brainstorm during the in-person meeting on September 25.

Week 2

Week 2 (September 24, 2024 – October 1, 2024)

Meeting Details

- Date & Time: September 25, 2024, 3:00 PM
- Type: In-Person Meeting (During Lecture)
- Duration: Approximately 2 hours
- Attendees:
    - Hassan Barada (Leader)
    - Abbas Amir (Scribe)
    - Rami Al Sabek (Member)
    - Abdul Raheem (Member)
    - Saleh Abu Zuhri (Member)

Agenda

1. Welcome new member, Saleh Abu Zuhri, to the team.
2. Share and discuss project ideas researched by each team member.
3. Evaluate and shortlist the proposed ideas.
4. Assign research roles for the shortlisted ideas.

Discussion Points

1. Team Update

- Saleh Abu Zuhri was officially added to the group, contributing to the team's diverse skill set and resources.

2. Presentation of Ideas

- Each member shared their initial project ideas:
    - A total of 10 ideas were presented, covering areas such as education, sustainability, cybersecurity, and AI.

3. Evaluation and Shortlisting

- The team evaluated the proposed ideas using the following criteria:
    - Relevance: Alignment with academic goals and team strengths.
    - Feasibility: Resource availability and timeline.
    - Impact: Potential to solve real-world problems innovatively.
- **Shortlisted Ideas:**
    - **Strategic Management Portal**: A platform for streamlining strategic decision-making in organizations.
    - **Sports Club Management Portal**: A system to help manage club activities, schedules, and member records.
    - **Document Version Control System**: A solution for tracking changes and managing versions of documents collaboratively.

5. Assignment of Roles

- **Each member selected a project to research further:**
    - Hassan Barada: Strategic Management Portal
    - Abbas Amir: Sports Club Management Portal
    - Rami Al Sabek: Document Version Control System
    - Abdul Raheem: Document Version Control System
    - Saleh Abu Zuhri: Research on feasibility and technology for all three ideas to provide support.

Week 3

Week 3 (October 2, 2024 – October 8, 2024)

Meeting Details

First Meeting

- **Date & Time:** October 2, 2024, During Class
- **Type:** In-Person (Class Discussion)

Second Meeting

- **Date & Time:** October 4, 2024, 7:00 PM
- **Type:** Online Meeting via Google Meet
- **Duration:** 1 hour

Agenda

1. Present finalized project ideas to the professor for feedback.
2. Brainstorm ways to refine and add clarity to the project ideas based on feedback.
3. Prepare improved project proposals for further review.

Discussion Points

1. Finalization of Project Ideas (October 2, 2024)

- The team finalized three project ideas for presentation:
    - **Strategic Management Portal**
    - **Sports Club Management Portal**
    - **Document Version Control System**

- Members prepared their respective parts of the presentation.

2. Feedback from Professor (October 2, 2024)

- The professor reviewed the proposed ideas and provided constructive feedback, noting that:
    - The ideas were promising but lacked clarity in their scope and objectives.
    - More details were needed to make them viable and innovative.

3. Refinement Session (October 2, 2024)

- The team held an in-class brainstorming session immediately after receiving feedback.
- Key improvements discussed:
    - Adding specific use cases for each project.
    - Highlighting the technical and innovative aspects.

4. Follow-Up Meeting (October 4, 2024)

- The team reconvened online to finalize improved versions of the project proposals.
- Tasks were distributed to ensure each idea had clear objectives and practical implementation plans.

Week 4

Week 4 (October 9, 2024 – October 15, 2024)

Meeting Details

First Meeting

- **Date & Time:** October 9, 2024, During Class
- **Type:** In-Person (Class Presentation & Discussion)

Second Meeting

- **Date & Time:** October 10, 2024, 7:00 PM
- **Type:** Online Meeting via Google Meet
- **Duration:** 1.5 hours

Agenda

1. Present improved project ideas to the professor for evaluation.
2. Brainstorm new project ideas after rejection of the initial proposals.

3. Discuss and refine newly proposed ideas.

Discussion Points

1. Presentation to Professor (October 9, 2024)

- The team presented the revised project ideas to the professor:
    - **Strategic Management Portal**
    - **Sports Club Management Portal**
    - **Document Version Control System**
- **Outcome:**
    - All three ideas were rejected by the professor, citing insufficient originality and alignment with project goals.

2. Immediate Brainstorming Session (October 9, 2024)

- After the rejection, the team held an in-class session to generate new ideas.
- Several preliminary concepts were discussed but required further exploration.

3. Follow-Up Meeting (October 10, 2024)

- A detailed online meeting was conducted to refine the new project ideas.
- Focus was placed on innovation, feasibility, and alignment with team interests and academic objectives.
- New ideas shortlisted for further research included:
    - **Interactive Storytelling and Learning App**
    - **Food Waste Reduction Platform**
    - **AI-Powered Phishing Detection System**

Week 5

Week 5 (October 16, 2024 – October 22, 2024)

Meeting Details

First Meeting

- **Date & Time:** October 16, 2024, 7:00PM
- **Type:** In-Class
- **Duration:** 30 mins

Second Meeting

- **Date & Time:** October 17, 2024, 6:00 PM
- **Type:** Online Meeting via Google Meet
- **Duration:** 1 hour

Third Meeting

- **Date & Time:** October 20, 2024, 7:00 PM
- **Type:** Online Meeting via Google Meet
- **Duration:** 30 mins

Agenda

1. Discuss changes in group composition and welcome the new member.
2. Confirm the professor's approval of the Interactive Storytelling App idea.
3. Start planning and feasibility analysis for the approved project.

Discussion Points

1. Group Composition Update

- **Change:**
    - Rami Al Sabek left the group.
    - Ajmal Abu Razak was introduced as the new member.

2. Project Approval

- The professor officially approved the **Interactive Storytelling and Learning App for Kids** on October 16.
- Feedback from the professor emphasized:
    - The need for detailed feasibility analysis.
    - A clear plan outlining milestone, technical requirements, and expected outcomes.

3. Planning and Feasibility Analysis

- The team discussed key aspects of the feasibility analysis:
    - **Technical Feasibility:** Researching AI and NLP tools (e.g., TensorFlow, Keras).
    - **Market Feasibility:** Understanding demand for interactive learning tools for kids.
    - **Resource Requirements:** Evaluating hardware, software, and team skills needed for implementation.

Initial brainstorming for project milestones and division of tasks began.

Week 6

Week 6 (23 October, 2024 – 29 October, 2024)

Meeting Detailsbrand

Date & Time: October 23, 2024
Type: Online Meeting via Google Meet
Duration: 2 hours

Date & Time: October 24, 2024
Type: Online Meeting via Google Meet
Duration: 2 hours

Date & Time: October 26, 2024
Type: Online Meeting via Google Meet
Duration: 1 hour

Agenda

1. Finalize the main details of the Interactive Storytelling and Learning App.
2. Decide on the project logo and branding.
3. Distribute work to each member.
4. Discuss any feedback or revisions needed based on the Planning & Feasibility Analysis submission.

Discussion Points

1. Finalizing Project Details

- The team discussed the primary features and functionalities of the **Interactive Storytelling and Learning App for Kids**.
    - **Core Features:**
        - Personalized story content using AI and NLP.
        - Dynamic Difficulty Adjustment (DDA) to tailor learning experiences.
        - User dashboards for children and teachers.
    - **Target Audience:**
        - Primary school-aged children (5-10 years).
        - Teachers and parents who will monitor progress.

2. Logo and Branding

- The team brainstormed logo concepts that would reflect the fun, interactive, and educational nature of the app.
- After several iterations, the final logo was decided. The design features playful, vibrant colors and icons that represent storytelling and learning (e.g., an open book with a lightbulb).

3. Planning & Feasibility Analysis Document

- The team discussed the feedback on the **Planning & Feasibility Analysis Document**, submitted on **October 23, 2024**.

Week 7

Week 7 (30 October, 2024 – 5 November, 2024)

Meeting Details

Date & Time: November 1, 2024
Type: Online Meeting via Google Meet
Duration: 1 hour

Date & Time: November 4, 2024
Type: Online Meeting via Google Meet
Duration: 1 hour

Agenda

1. Discuss feedback from the professor on the **Planning & Feasibility Analysis Document**.
2. Start planning and work distribution for the **Requirements Analysis Document**.
3. Identify and assign specific roles for the next steps in document preparation.

Discussion Points

1. Feedback on Planning & Feasibility Analysis Document

- **Feedback Overview:**
    - The professor provided constructive feedback on the **Planning & Feasibility Analysis Document**, focusing on:
        - **Technical Clarity:** Further details were needed on the integration of AI tools and algorithms.
        - **Market Feasibility:** Suggestions were made to enhance the analysis by including competitor research and user personas.
        - **Resource Assessment:** More precise details about hardware and software requirements were requested.
- **Action Plan:**
    - The team agreed to refine the document by incorporating the professor's feedback, especially focusing on technical aspects and market analysis.

2. Work Distribution for Requirements Analysis Document

- The team began discussing the structure and content of the **Requirements Analysis Document**.
    - Key areas to be covered:
        - **Functional Requirements:** Defining user stories and key app features.
        - **Non-Functional Requirements:** Performance, scalability, and security.
        - **System Architecture:** Defining the backend and frontend structure.
- The team outlined the responsibilities for drafting the document:
    - **Functional Requirements**: **Hassan Barada**
    - **Non-Functional Requirements**: **Abdul Raheem**
    - **System Architecture**: **Saleh Abu Zuhri**
    - **AI & Technical Integration**: **Abbas Amir**
    - **Risk Analysis**: **Ajmal Abu Razak**

3. Next Steps & Timeline

- The team discussed a timeline for completing the **Requirements Analysis Document**, with an agreed-upon submission date of **November 6, 2024**.
- The document was submitted via mail on November 6, 2024 midnight.

Week 8

Week 8 (6 November, 2024 – 12 November, 2024)

Meeting Details

Date & Time: November 6, 2024
Type: In-Class Meeting
Duration: 2 hour

Date & Time: November 11, 2024
Type: In-Class Meeting
Duration: 1 hour

Agenda

1. Discuss and outline the structure for the **Presentation with Panel**.
2. Assign tasks and deadlines for preparing the presentation.
3. Review the required content sections for the presentation.

Discussion Points

1. Presentation Outline

The team started working on preparing the **Presentation with Panel**, which will be presented on **November 13, 2024**. The following sections were agreed upon for the presentation outline:

1. **What problem are you trying to solve?**
    - A clear explanation of the problem that the app aims to address.
2. **Why is this problem interesting to solve?**
    - Emphasizing the significance and impact of solving this problem, particularly in the context of children's learning.
3. **The Solution proposed and its key features.**
    - A detailed description of the **Interactive Storytelling App for Kids** and its unique features.
4. **Competitors Analysis Table.**
    - A table comparing the proposed solution with existing solutions in the market.
5. **High-level System Architecture.**
    - Visualizing the app's architecture, showcasing the relationship between the front-end and back-end components.
6. **Hardware and Software Requirements.**
    - Listing the hardware and software needed for the development and testing of the app.
7. **Timeline for the full project.**
    - Presenting a timeline for the entire project, highlighting key milestones.
8. **Team Members' Skills and Roles in the Project.**
    - Detailing each team member's role and the skills they bring to the project.

2. Work Distribution & Deadlines

Each team member was assigned specific sections of the presentation to work on, with a deadline of **November 12, 2024**, for finalizing the presentation.

- **Hassan Barada**: "What problem are you trying to solve?" & "Why is this problem interesting to solve?"
- **Abbas Amir**: "The Solution Proposed and its Key Features" & "Competitors Analysis Table"
- **Abdul Raheem**: "High-level System Architecture"
- **Saleh Abu Zuhri**: "Hardware and Software Requirements"
- **Ajmal Abu Razak**: "Timeline for the Full Project" & "Team Members' Skills and Roles in the Project"

Week 9

Week 9 (13 November, 2024 – 19 November, 2024)

Meeting Details

Date & Time: November 13, 2024
Type: In-Class Meeting
Duration: 1 hour

Date & Time: November 17, 2024
Type: Online Meeting Via Google Meets
Duration: 2 hour

Agenda

1. Review feedback received from the **Professor Panel**.
2. Discuss and implement the feedback into the project.
3. Plan and distribute tasks for the **Design Document** (High & Low-Level Design).
4. Set clear deadlines for completing the Design Document.

Discussion Points

1. Feedback from Professor Panel

- The team reviewed the feedback from the professor panel after the presentation on **November 13, 2024**.

- The feedback was generally positive, with suggestions for further clarification on certain sections of the project, such as the user flow and interface design.
- **Action Taken:** The feedback was immediately incorporated into the project, especially regarding the **user interface design** and **system architecture**. Minor adjustments were made to reflect the feedback and ensure clarity.

2. Design Document Discussion

- The team focused on planning and discussing the next phase: the **Design Document** (High & Low-Level Design).
- The **High-Level Design** will outline the overall structure and flow of the app, including the key components and their interactions.
- The **Low-Level Design** will go deeper into the details of each component, providing a blueprint for developers to follow during implementation.
- **Action Plan:**
    - The team divided the tasks for the **Design Document**, ensuring all aspects (UI design, backend structure, database design, etc.) were covered.
    - **Deadline for Submission: November 27, 2024**.

3. Work Distribution & Deadlines

Each team member was assigned a section of the **Design Document**.

- **Hassan Barada**: High-Level Design (UI/UX flow and components interaction)
- **Abbas Amir**: Software Interface Design & Testing
- **Abdul Raheem**: System Architecture
- **Saleh Abu Zuhri**: Plan and prepare System Overview
- **Ajmal Abu Razak**: Component Designing

Week 10/11

Week 10 & 11 Diary/Meeting Minutes

Project: Capstone Project 2024
Week 10 (November 20, 2024 – November 28, 2024)

Meeting Details

Date & Time: November 21, 2024
Type: Online Meeting via Google Meet
Duration: 2 hours

Date & Time: November 25, 2024
Type: Online Meeting via Google Meet of Hassan (Leader) with Abdullah Al Nokiti (Mentor)
Duration: 30 minutes

Date & Time: November 25, 2024
Type: Online Meeting via Google Meet
Duration: 1 hour

Agenda

1. Review of each member's work on the Design Document.
2. Provide feedback on each other's work.
3. Introduction of mentor **Abdullah Al Nokiti** and feedback session.
4. Implement changes based on mentor and team feedback.

Discussion Points

1. Review of Work

- The team went over the work completed so far for the **Design Document**.
- Each member reviewed their respective sections and presented them for feedback.
- **Feedback Points:**
    - Emphasis was placed on clarity in the **System Overview** and **Component Design** sections.
    - Suggestions for more detailed explanations on specific components in the **Software Interface Design** were made.

2. Feedback Among Team Members

- Each member provided constructive feedback on others' work, especially focusing on consistency in design elements and logical flow across sections.
- Changes and improvements were noted for further refinement.

3. Mentor Feedback

- The team introduced a mentor, **Abdullah Al Nokiti**, who reviewed the **Design Document** and gave detailed feedback.
    - He suggested improving the **UI/UX flow** for a smoother user experience.
    - Recommended clarifying the relationships between components in the **System Architecture** section.
    - Advised on better integration of features in the **Component Design** section.

4. Implementing Changes

- The team agreed to implement the feedback given by both the team members and **Abdullah Al Nokiti**.
- The necessary revisions were made in each section to address clarity, integration, and flow.

Semester 2

Week 1

Week 1 (January 6, 2025 – January 12, 2025)

Meeting Details
Date & Time: January 7, 2025
Type: Online Meeting via Google Meet
Duration: 1 hour 30 minutes

Agenda

1. Kickoff of the second semester: Overview of remaining tasks.
2. Review of feedback from the first semester.
3. Assignment of new tasks and roles for the second semester.
4. Setting milestones for project completion.

Discussion Points

1. Kickoff of the Second Semester
    - The team reviewed the timeline and upcoming deadlines.
    - A clear division of work for the remaining tasks was established.
2. Review of Feedback
    - Each team member presented their work from the first semester and discussed the feedback received from mentors and peers.
    - Positive feedback on the clarity of the Design Document and suggestions for better implementation of UI/UX features.
3. New Tasks and Roles
    - The project manager (Hassan) outlined new roles based on team strengths.
    - Tasks were reassigned to ensure efficiency for the second semester, particularly focusing on the development and integration phase.
4. Milestone Setting
    - The team agreed on specific milestones:
        - Completion of the software architecture model by Week 4.

- Implementation of major system features by Week 6.

Week 2

Project: Capstone Project 2025
Week 2 (January 13, 2025 – January 19, 2025)

Meeting Details
Date & Time: January 16, 2025
Type: In-Person Meeting in University
Duration: 2 hours

Agenda

1. Review of system architecture and initial software models.
2. Discuss UI/UX feedback implementation.
3. Plan integration of system components.
4. Adjust timeline for next milestones.

Discussion Points

1. System Architecture Review
    - Abdul Raheem presented the updated system architecture.
    - Team members provided feedback on improving modularity and scalability.
2. UI/UX Design Feedback
    - Abbas Amir showcased progress on the UI/UX design, incorporating mentor feedback from the first semester.
    - Suggestions were made for a smoother user interface and user journey.
3. Component Integration
    - Discussed initial integration of various components in the system.
    - Saleh Abu Zuhri outlined the first steps of module development.
4. Timeline Adjustments
    - Adjusted the milestones to accommodate additional tasks.
    - The final submission deadline for the second semester was emphasized.

Week 3

Project: Capstone Project 2025
Week 3 (January 20, 2025 – January 26, 2025)

Meeting Details

First Meeting

- **Date & Time:** January 21, 2025, 9:00 PM
- **Type:** Online Meeting via Google Meet
- **Duration:** 1 hour

Agenda

1. **Review progress on story generation logic.**
2. **Final testing of UI/UX design.**
3. **Conduct tests for Firebase integration.**
4. **Discuss new features for improved story coherence.**

Discussion Points

1. Story Generation Logic Review
    - Abbas Amir demonstrated the progress of the refined story coherence algorithms.
    - Suggestions were made for improving the transition between story themes.
2. UI/UX Design Testing
    - Hassan Barada presented the completed UI/UX design for the app.
    - The team conducted a round of internal testing for the interface, providing feedback for final tweaks.
3. Firebase Integration Test
    - Ajmal Abu Razak demonstrated the first successful integration of dynamic themes through Firebase.
    - Some small bugs were identified, especially related to theme updates during story generation.
4. New Features Discussion
    - The team brainstormed ideas for additional features, including story branching and user-driven prompts for dynamic story progression.

Week 4

Project: Capstone Project 2025
Week 4 (January 27, 2025 – February 2, 2025)

Meeting Details
Date & Time: January 27, 2025
Type: Online Meeting via Google Meet
Duration: 1 hour 30 minutes

Agenda

1. Review internal testing results of the story generation app.
2. Finalize integration of themes with Firebase.
3. Discuss improvements to story coherence.
4. Update project timeline and plan next steps.

Discussion Points

1. Internal Testing Results
    - Ajmal Abu Razak presented the results of the internal testing, highlighting issues with theme accuracy during story transitions.
    - The team reviewed the feedback and made adjustments to improve theme loading times and relevance.
2. Firebase Integration Finalization
    - Saleh Abu Zuhri and Ajmal Abu Razak finalized the integration with Firebase for dynamic theme updates, ensuring smoother transitions.
3. Story Coherence Improvements
    - Abdul Raheem discussed improvements to the story coherence algorithms, ensuring better alignment of story elements.
    - The team worked on refining the transition logic for story sections to improve flow and user engagement.
4. Project Timeline and Next Steps
    - The team adjusted the timeline to focus on final testing and bug fixes.
    - The focus now shifts to preparing for full-scale testing and final documentation for submission.

Week 5

Week 5 (February 3, 2024 – February 9, 2024)

Meeting Details

First Meeting

- **Date & Time:** February 9, 2025, 4:00PM
- **Type:** In-Class
- **Duration:** 2 hours

Agenda

1. Review progress on the API development for story generation.
2. Test the app's integration with the new API.
3. Finalize user interface updates for dynamic content.
4. Discuss final bug fixes and next steps for project completion.

Discussion Points

1. API Development for Story Generation
    - The team discussed the creation of a REST API to handle story generation requests from the app.
    - Hassan Barada presented the initial design of the API, explaining the data flow between the app, backend, and Firebase for dynamic theme retrieval.
    - Key decisions were made regarding the API endpoints, including a POST /generate-story endpoint for sending story prompts and receiving generated content.
    - Abbas Amir and Abdul Raheem reviewed the API structure and discussed how to ensure scalability for future expansion.
2. App Integration with the API
    - Ajmal Abu Razak demonstrated the integration between the app and the new API.
    - Early tests showed success in generating stories, but some adjustments were needed to handle API latency issues, which were causing slight delays in story loading.
3. UI Updates for Dynamic Content
    - Abbas Amir showed off the latest UI updates, focusing on the integration of dynamically generated story content.
    - The team provided feedback on how to improve the readability and interaction of the dynamic content in the app’s UI.
4. Final Bug Fixes and Next Steps
    - Saleh Abu Zuhri led the discussion on bug fixes. The team prioritized resolving minor issues related to story transitions and Firebase integration.
    - Plans were made to finalize all development work by Week 6 and prepare for the final user testing phase.

Week 6

Project: Capstone Project 2025
Week 6 (10 February, 2025 – 16 February, 2025)

Meeting Details

Date & Time: February 13, 2025
Type: In-Person
Duration: 1 hours

Date & Time: February 16, 2025
Type: Online Meeting via Google Meet
Duration: 2 hours

Agenda

1. Conduct final testing of the story generation API.
2. Review the complete app integration with Firebase and the API.
3. Finalize UI/UX for the app with real-time story generation.
4. Discuss project submission details and documentation.

Discussion Points

1. Final Testing of the API
    - The team conducted a thorough round of testing on the story generation API, ensuring that all endpoints functioned as expected.
    - Ajmal Abu Razak provided an overview of optimizations to reduce latency and improve the speed of story loading. Minor bugs were identified and quickly resolved, including issues with data synchronization between the app and Firebase.
2. App and Firebase Integration
    - Saleh Abu Zuhri tested the app's full integration with Firebase, ensuring real-time theme and story updates were being pulled correctly.
    - Final adjustments were made to handle concurrent users more effectively, especially when multiple users accessed the story generation at the same time.
3. UI/UX Finalization
    - Abbas Amir presented the final version of the app's UI/UX, focusing on the seamless integration of dynamically generated stories and user interactions.

The team agreed on final touch-ups for better user experience, including smoother animations for transitioning between different story themes.

Week 7

Project: Capstone Project 2025
Week 7 (10 February, 2025 – 16 February, 2025)

Date & Time: February 13, 2025
Type: In-Person
Duration: 1 hours

Date & Time: February 16, 2025
Type: Online Meeting via Google Meet
Duration: 2 hours

Agenda

1. Re-test Firebase integration due to persistent issues.
2. Review API performance and response times.
3. Identify and fix UI inconsistencies reported during testing.
4. Plan strategy for final bug fixes and optimizations.

Discussion Points

1. Firebase Integration Issues
    - Several issues from previous weeks persisted, despite being marked as resolved.
    - The team had to redo much of the Firebase integration work that Ajmal previously handled.
    - Major problems included real-time updates not reflecting correctly and conflicts with concurrent users.
2. API Performance
    - Hassan Barada worked on API improvements, focusing on reducing response time.
    - Load testing was conducted to evaluate system efficiency under stress conditions.
3. UI Inconsistencies
    - Abbas Amir identified inconsistencies in the UI due to improper handling of dynamic content.
    - Adjustments were made to ensure smoother user experience across different story themes.
4. Final Bug Fixes and Optimizations
    - The team compiled a final list of bugs to be addressed before submission.
    - A new testing plan was outlined to validate fixes before release.

Week 8

Project: Capstone Project 2025
Week 8 (10 February, 2025 – 16 February, 2025)

Date & Time: February 13, 2025
Type: In-Person
Duration: 1 hours

Date & Time: February 16, 2025
Type: Online Meeting via Google Meet
Duration: 2 hours

Agenda

1. Conduct full-scale testing of Firebase real-time updates.
2. Review and finalize API structure.
3. Prepare documentation for project submission.
4. Assign tasks for final presentation preparation.

Discussion Points

1. Firebase Full-Scale Testing
    - A dedicated testing session was held to simulate real-world app usage.
    - Issues persisted, leading to further investigation and fixes.
2. API Finalization
    - Hassan Barada reviewed the final API structure, making minor adjustments for improved efficiency.
    - Endpoints were validated against expected outputs.
3. Documentation
    - Abbas Amir led the documentation effort, ensuring all system components were well-documented.
    - Special focus was placed on justifying architecture and implementation choices.
4. Presentation Preparation
    - The team divided tasks for the upcoming project presentation.

Week 9

Project: Capstone Project 2025
Week 9 (10 February, 2025 – 16 February, 2025)

Date & Time: February 13, 2025
Type: In-Person
Duration: 1 hours

Date & Time: February 16, 2025
Type: Online Meeting via Google Meet
Duration: 2 hours

Agenda

1. Review and refine presentation materials.
2. Conduct final bug testing.
3. Prepare for project submission.

Discussion Points

1. Presentation Review
    - Team members practiced their respective sections.
    - Adjustments were made based on feedback.
2. Bug Testing
    - A final sweep of the app was conducted to catch any remaining issues.
    - Firebase data sync remained a concern, requiring yet another fix.
3. Submission Preparation
    - Documentation and code repositories were organized for submission.

Week 10

Project: Capstone Project 2025
Week 10 (10 February, 2025 – 16 February, 2025)

Date & Time: February 13, 2025
Type: In-Person
Duration: 1 hours

Date & Time: February 16, 2025
Type: Online Meeting via Google Meet
Duration: 2 hours

Agenda

1. Present the project.
2. Answer panel questions.
3. Receive feedback and discuss improvements.

Discussion Points

1. Project Presentation
    - The team delivered the final presentation.
    - Judges provided feedback on technical and design aspects.
2. Panel Questions
    - Questions focused on AI model efficiency and Firebase integration challenges.
3. Feedback & Improvements
    - Suggestions were made to improve scalability and user experience.

Summary Week 10 marked the final presentation. The project was well-received, though Firebase issues were noted as a recurring challenge.

Prepared by: Abbas Amir (Scribe)

Week 11

Project: Capstone Project 2025
Week 11 (10 February, 2025 – 16 February, 2025)

Meeting Details Date & Time: March 21, 2025 Type: Wrap-Up Meeting Duration: 1 hour

---

Agenda

1. Review project performance.
2. Discuss lessons learned.
3. Plan next steps for deployment (if applicable).

Summary Week 11 served as a wrap-up session, reflecting on challenges and achievements.

10.4 Milestone Report

Task	Completion
Semester 1	
Project Initiation	
Market Analysis	Week 1
Feature Definition	Week 2
Define roles	Week 2
Establish project scope	Week 3
Project plan development	Week 4
Approve project plan	Week 5

Requirements & System Design	
Requirement Finalization	Week 6
Select app name	Week 6
Finalize function requirements	Week 7
Finalize non - functional requirements	Week 7
Develop user interface diagrams	Week 8
Define prerequisites	Week 6
Develop system architecture and data flow diagrams	Week 9
Create UI/UX wireframes	Week 10
Compile summary	Week 11
Risk analysis	Week 11
Semester 2	
Backend & Database Development	
Set up firebase authentication & user management	Week 1-4
Implement database structure in firebase	Week 1-4
Develop API services	Week 1-4
Implement real-time data tracking & syncing	Week 5
AI & Adaptive Learning Module	
Develop AI-based storytelling engine	Week 6-8
Implement adaptive difficulty algorithm	Week 6-8
Integrate NLP for student interactions	Week 6-8
Frontend Module	

App home page and initial design	Week 6-8
Develop student dashboard	Week 6-8
Develop teacher dashboard	Week 6-8
Develop interactive quizzes	Week 6-8
Testing & Deployment	
Review test cases	Week 8
Complete documentations	Week 8
Conduct front end tests	Week 9-10
Conduct back end tests	Week 9-10
Deploy sample version	Week 9-10
Asses the beta version	Week 9-10
Deploy final app	Week 9-10
Bug fixes and monitoring	Week 9-10

10.5 Requirements Matrix

Requirement ID	Requirement Description	Type	Stakeholder	Linked Use Cases	Priority	Status
R1	The system shall generate AI-driven interactive stories based on user preferences.	Functional	Student, Parent	AI Story Generation	High	Implemented
R2	The system shall track user progress and adapt content accordingly.	Functional	Student, Teacher, Parent	Adaptive Learning	High	Implemented

R3	The system shall allow teachers and parents to monitor student learning analytics.	Functional	Teacher, Parent	Progress Tracking	High	Implemented
R4	The system shall integrate a gamification and rewards system to enhance engagement.	Functional	Student	Gamification	Medium	In Progress
R5	The system shall enable real-time parental control and content monitoring.	Functional	Parent	Parental Dashboard	Medium	In Progress
R6	The system shall allow administrators to manage users and class enrollments.	Functional	Admin	User & Class Management	Medium	Implemented
R7	The system shall support cross-platform accessibility on mobile platforms.	Non-Functional	All Stakeholders	Applicable to all	High	Implemented
R8	The system shall implement encryption and secure authentication to protect user data.	Non-Functional	All Stakeholders	Applicable to all	High	Implemented
R9	The system shall provide a user-friendly and engaging interface for children.	Non-Functional	Student	Applicable to all	High	Implemented