

CS 673 SOFTWARE ENGINEERING

**Team 2**

**“My Magical Bedtime”**

Project Proposal and Planning

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Iqra Chaudhary | Team Lead | *IC* | 5/14/2025 |
| Tetiana Korchynska | Requirement Lead,  Design and Implementation Lead | *TK* | 5/13/2025 |
| Thomas Iliev | QA Lead | *TI* | 5/13/2025 |
| Grant Hovey | Configuration Lead | *GH* | 5/14/2025 |
| Hongjie Zhang | Design and Implementation Lead, Security Lead | *HJZ* | 5/12/2025 |
| Xuelin Min |  | *XLM* | 5/15/2025 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| 1 | Hongjie Zhang | 5/14/2025 | Filled the Overview, Related work, Management Plan,  Configuration Management Plan,  Quality Assurance Plan and Quality Assurance Plan |
| 2 | Tetiana Korchynska | 5/14/2025 | Filled the Functional Requirements in Proposed High level Requirements, AI Usage Log, References and Glossary sections, formatted the whole document |
|  |  |  |  |
|  |  |  |  |

[Overview](#_g6igqliy7rm)

[Related Work](#_bf21eadgjj29)

[Proposed High level Requirements](#_rgyo4hi9stmq)

[Management Plan](#_ts358bsdtbcv)

[Objectives and Priorities](#_nxeeppkjxgn4)

[Risk Management (need to be updated constantly)](#_tk7yixobah8p)

[Timeline (need to be updated at the end of each iteration)](#_iksrndohvx29)

[Configuration Management Plan](#_j5uvivmxqcsp)

[Tools](#_dzly5b9kz982)

[Deployment Plan if applicable](#_sd8zu6r3jisd)

[Quality Assurance Plan](#_vra5ptwu59qx)

[Metrics](#_vwjduhc9wuah)

[Code Review Process](#_hx3eaiwb8v3m)

[Testing](#_l9xnpmd6hh0y)

[Defect Management](#_5amsh8h9f0c7)

[AI Usage Log](#_161vmjht87fy)

[References](#_pd9euov6m4du)

[Glossary](#_ty3i2nqffhtc)

# 

# 

# 

# 

# 

# 

# 

# **Overview**

*(Please give an overview of your project. It should include the motivation, the purpose and the potential users of the proposed software system, the basic functionality of the proposed software system and the possible technology stack to be used. )*

Our "My Magical Bedtime" app is a bedtime story generator designed specifically for preschool children (ages 3-5). The app addresses parents' need for fresh, engaging bedtime content that simultaneously supports early childhood development.

**Motivation:** Many parents struggle to create new bedtime stories nightly, while educational screen time options for preschoolers are limited. My Magical Bedtime bridges this gap by providing an interactive storytelling experience that both entertains and educates.

**Purpose:** To generate age-appropriate, customizable bedtime stories that foster language development, emotional intelligence, and creativity in preschool children while making bedtime routines more engaging.

**Users:** Primary users include parents/caregivers reading with their children and preschool-aged children (3-5 years) who will interact with the app alongside adults.

**Basic Functionality:**

* Story generation with customizable characters, settings, and themes
* Simple, child-friendly interface with minimal text input required
* Read-along feature with text highlighting and audio narration
* Story saving capability for favorites
* Parent controls for content appropriateness

**Technology Stack:**

* Frontend: React for building user interfaces
* Backend: Node.js with Express
* Database: MongoDB for story templates and user preferences
* Natural Language Processing: GPT-based API for story generation

# 

# **Related Work**

*(Please describe any similar software systems that you have found through the online research, and the differences between your software and those software systems.)*

Through online research, we identified **Bedtime Story AI** (bedtimestory.ai) as a similar system in the children's storytelling space. This platform also uses AI to generate personalized bedtime stories for children.

The differences between our app and Bedtime Story AI Age-specific focus, Our app is specifically designed for preschool children (ages 3-5) with appropriate vocabulary levels and concepts, while Bedtime Story AI serves a broader age range.

## Proposed High level Requirements

1. **Functional Requirements**  
   *(For each functional requirement, please give a feature title and a brief description using the following format: As (a role), I want to (action), so that (value).)*

**Essential Features** *(the core features that you definitely need to finish): (For each essential features, please give a rough estimation in terms of person hours or an range of person hours)*

| **Feature Title** | **User Story** | **Est. Person-Hours** |
| --- | --- | --- |
| User Registration & Login | As a parent, I want to create an account and log in securely, so that I can manage my child’s story preferences and saved stories. | 12 |
| Story Generation | As a child with help from a parent, I want to generate a customized bedtime story by selecting characters, themes, and settings, so that I can enjoy a personalized reading experience. | 24 |
| Story Read-Along Mode | As a child, I want the story to be displayed with highlighted text as it is read aloud, so that I can improve my reading skills and engagement. | 12 |
| Story Saving | As a parent, I want to save my child’s favorite generated stories, so that we can read them again later. | 12 |
| Accessibility Narration Mode | As a child with sight disabilities, I want the stories to be fully audio-narrated with voice commands, so that I can enjoy the stories independently without needing to read text. | 12 |
| Parent Controls | As a parent, I want to enable content filters and limit story themes, so that I can ensure age-appropriate content for my child. | 12 |
| Secure User Data | As a developer, I want to ensure password encryption and secure data handling, so that user information is protected. | 12 |

**Desirable Features** *(the nice features that you really want to have too)*

| **Feature Title** | **User Story** | **Est. Person-Hours** |
| --- | --- | --- |
| Story Character Customization | As a child, I want to customize the names and appearances of characters, so that the story feels even more personal. | 24 |
| Theme Customization | As a child, I want to choose different settings or story themes (e.g., jungle, space), so that I can explore new adventures. | 18 |
| Audio Narration Download | As a parent, I want to download the audio version of a generated story, so that my child can listen offline. | 24 |
| Story Sharing (within app) | As a parent, I want to share a saved story with other parent accounts in the app, so that friends and family can read the same story. | 18 |

**Optional Features** (*additional cool features that you want to have if there is time)*

| **Feature Title** | **User Story** | **Est. Person-Hours** |
| --- | --- | --- |
| Visual Story Enhancements | As a child, I want to see simple illustrations or animations with the text, so that I stay more engaged. | 12 |
| Offline Mode | As a parent, I want to access saved stories without an internet connection, so that we can read during travel. | 18 |
| Reward System | ​​As a child, I want to earn badges or rewards for reading stories, so that I feel motivated to read more. | 18 |
| Usage Tracking Dashboard | As a parent, I want to see my child’s reading history and frequency, so that I can track progress over time. | 18 |
| Voice Assistant for Story Selection | As a child, I want to say “Tell me a new story” and have a new story start, so that I can use the app hands-free. | 12 |

1. **Nonfunctional Requirements**

* **Security requirements**

Our security implementation will include robust authentication with bcrypt password hashing and encryption of sensitive data at rest, while protecting API endpoints through rate limiting and strict CORS policies. We'll prevent common vulnerabilities by implementing MongoDB schema validation, anti-CSRF tokens, and proper JWT handling, with all input parameters being thoroughly validated to return appropriate error codes without exposing system details.

# 

# **Management Plan**

## Objectives and Priorities

*(Please describe your project objectives with highest priority first. Project Goals can include but not limited to complete all proposed (essential) features, deploy the software successfully, the software has no known bugs, maintain high quality, etc )*

**High Priority (Essential)**

* **Develop a functioning, child-friendly bedtime story generator app** that creates age-appropriate stories for preschool children (ages 3-5)
* **Implement educational elements** that support early childhood learning objectives including vocabulary development, emotional intelligence, and basic concepts
* **Design an intuitive interface** that both parents and preschool children can navigate easily with minimal instruction
* **Ensure content safety and appropriateness** for the target age group with content filtering and parental controls

**Medium Priority (Important)**

* **Create customizable story elements** allowing children to select characters, settings, and themes to enhance engagement
* **Implement read-along functionality** with highlighted text and optional audio narration to support early reading skills
* **Develop a story library feature** to save favorite generated stories for repeated access
* **Optimize performance** for smooth operation across multiple device types (tablets, smartphones)

**Lower Priority (Desirable)**

* **Add visual elements and simple animations** to enhance story engagement while maintaining focus on the narrative
* **Implement offline functionality** so stories can be accessed without internet connection
* **Create a parent dashboard** to track usage patterns and learning progress
* **Design a reward system** to encourage regular reading habits

## 

## Risk Management (need to be updated constantly)

(Please write a summary paragraph about the main risks your group identified and how you plan to manage these risks. Then use the separate google sheet for detailed risk management. The template is provided in the same folder with this file. Please provide the link to the sheet.)

The main risks that our group has identified are establishing a list of requirements, finding time during the week to meet as a full group, and adding new team members. One of our issues was creating a list of the tasks that we needed to complete by Thursday because we started thinking about the requirements on Tuesday. In the future, we plan to meet earlier in the week and start group meetings with a list of requirements already partially built, and then add on to it as the meeting progresses. Another issue we had was finding times to meet throughout the week. Unfortunately, our schedules clashed this week and we were only able to meet once as a full group. We plan to manage this risk by planning our meetings out a week in advance and possibly by sticking to a scheduled meeting time that we know is available for everyone. Lastly, we had two last minute additions to our team on the day before Iteration 0 of the project was due. It was difficult to integrate two new people into the team just 24 hours before a lot of work had to be finalized and this was mainly due to the fact that we started so late. For the future, we plan to start working earlier in the week so that we have more time to deal with issues that surprise us and are out of our control.

**Risk Management Sheet Link:** <https://docs.google.com/spreadsheets/d/15ZyhsUgyEfNvTP6xu0ae1PkKea6BGISsrHbtZLhMi1A/edit?usp=drive_link>

## Timeline (this section should be filled in iteration 0 and updated at the end of each later iteration)

| Iteration | Functional Requirements(Essential/Disable/Option) | Tasks (Cross requirements tasks) | Estimated/real person hours |
| --- | --- | --- | --- |
| 1 | **Frontend UI Design and Prototyping**: (60 hr)  **Backend visualization**: (36hr)  **Class Diagram**: (10hr)  **Essential Features** (96 hr) **Testing** (unit + integration + some manual testing): 24 hrs |  | 230 hours |
| 2 | **Desirable Features** (84hr**)**  **Testing** (unit + integration + some manual testing): 24 hrs |  | 160hr |
| 3 | **Optional Features** (78hr)  **Testing** (unit + integration + some manual testing): 24 hrs |  | 140hr |

# 

# **Configuration Management Plan**

1. **Tools**

*(In this project, we will use Git and Github as the version control tools. Please also specify any other tools to be used, e.g. IDE tools, CI/CD tools, container tools, SAST or DAST tools, and any other DevOps tools and* ***AI tools****)*

**Version Control:**

* Git for version control
* GitHub for repository hosting and collaboration

**Development Tools:**

* IDE: Visual Studio Code
* UI Design: Figma for interface design and prototyping
* Diagramming: Figma for class diagrams and architecture visualization

**Testing Tools:**

* Postman for API testing and documentation
* React Testing Library for frontend component testing
* Jest for JavaScript unit testing

**CI/CD and Deployment:**

* GitHub Actions for continuous integration
* Netlify for frontend hosting and deployment
* Heroku for backend hosting and deployment

1. **Code Commit Guideline and Git Branching Strategy**

*(Please briefly describe criteria for the code commitment and the branching strategy used, e.g. what are the branches to be used, how the pull request will be used etc. Here is an article to give you some basic knowledge about different git branching strategies:* [*https://www.flagship.io/git-branching-strategies/*](https://www.flagship.io/git-branching-strategies/) *)*

**Main Branches:**

* main: Production-ready code, always stable
* develop: Integration branch for ongoing development

**main** for stable, production-ready code and **dev** for ongoing development. All code changes should be committed to the **dev** branch with clear, descriptive messages, and merged into **main** through pull requests after review. This ensures code quality and keeps the production branch clean and reliable.

**Commit message guidelines:**

**Commit messages** should follow a standard format to ensure that they convey useful information instead of something low information like ‘wip’

A common standard that can be enforced by tools such as commitlint as as follow <type>[optional scope]: <description>

[optional body]

[optional footer(s)]

Where type is something like feat or fix to denote if something is a user story or a bug fix.

Scope refers back to a user story number or jira ticket, and description provides a brief summary of the changes that have occured. The optional body can be used to provide more details about what was implemented or fixed beyond the brief summary in the description. The footer is optional and is often used for a link to the item being closed in Git/Jira/Rally etc.

## CI/CD Plan

*(Briefly describe how you plan to continuously integrate and deploy your application).*

Our application will utilize GitHub Actions for automated testing and deployment. Every push to the main branch triggers test execution, and successful builds are automatically deployed to our staging environment.

# **Quality Assurance Plan**

## Metrics

*(Describe the metrics to be used in the project to measure the quality of your software. Each metric should be measurable and quantifiable. Examples of metrics include product complexity (LOC, # of files, # of classes, # methods, cyclomatic complexity, etc.) , defect rate (# of defect per KLOC), # of test cases, test case pass rate, cost (# of person hours used), # of user stories completed, etc.* ***The result of these metrics should be reported in the progress report/ iteration summary sheet.****)*

| **Metric Name** | **Description** |
| --- | --- |
| Bundle Size | Size of compiled frontend assets (target: <500KB) |
| API Response Time | Average response time for API endpoints (target: <200ms) |
| Test Pass Rate | Percentage of passing automated tests (target: 100%) |
|  |  |
|  |  |

1. **Coding Standard**

*(Describe any coding standard to be used)*

**General:**

* Follow Airbnb JavaScript Style Guide
* Use ESLint and Prettier for code formatting and style enforcement
* Maintain consistent naming conventions (camelCase for variables/functions, PascalCase for components/classes)

**Frontend (React):**

* Follow React recommended patterns and hooks
* Use functional components with hooks instead of class components
* Implement proper component structure and separation of concerns

**Backend (Node/Express):**

* Follow RESTful API design principles
* Implement proper error handling and validation
* Use async/await for asynchronous operations
* Implement authentication and authorization middleware

**Database (MongoDB):**

* Create proper schema validation
* Implement data validation at database level
* Follow naming conventions for collections and fields

## Code Review Process

*(Everyone should review all documents to be submitted. Here you will mainly describe how the code review will be done. Who will review the code, e.g. design or implementation leader will review all code or team members review each other’s code. Do you use pull requests for the code review? Is there a checklist to help review? What feedback should the reviewer provide?)*

All team members will receive notifications of code changes via Discord. Automated checks required to pass before human review begins. The tech lead must review all critical path code, Reviewers should provide constructive, specific feedback focused on adherence to coding standards, security requirements, and functional completeness rather than stylistic preferences.

## Testing

*(Both manual testing and automated testing should be considered. Both unit testing and integration testing should be considered. Briefly describe the testing tools/framework to be used, the personnel involved (e.g. the QA leader will focus on the integration testing and each developer will unit test their own code), when and what types of testing will be performed, the testing objectives, etc)*

Writing Jest unit tests for both React components and backend services to achieve 80% code coverage. Integration testing to verify API functionality using Supertest, while end-to-end testing will be implemented with Cypress for critical user flows across multiple browsers and devices. Manual exploratory testing will complement automated tests by verifying visual design against Figma specifications and catching edge cases.

## Defect Management

*(Describe the tool to be used to manage the defect (e.g github issues). The types of defects to look at. The actions or personnel for defect management. )*

We use **GitHub Issues** to track and manage defects, with each issue including a description, reproduction steps, severity, and any relevant screenshots or logs. Common defects include functional bugs, UI/UX issues, performance problems, and integration failures. Defects are reported by developers or QA, triaged by the project or QA lead, and assigned to developers for resolution. All fixes are tested in the **dev** branch before being merged into **main**.

# **AI usage Log**

*You are allowed and even encouraged to use AI tools to help you generate the project idea, plan it and build it, but you need to clearly describe 1) What tools were used? 2) for what specific tasks and 3) Is it helpful? 4) how did you evaluate or modify AI-generated content? Additionally, you should submit the exported AI chat history as an appendix or share that with the instructor and facilitators.*

| Tools | Who | Tasks | helpful | Evaluation/modification | links |
| --- | --- | --- | --- | --- | --- |
| ChatGPT | Tetiana Korchynska | To suggest the title for our app | yes | The model suggested titles for the application, reducing our brainstorming time and providing a usable option. | <https://chatgpt.com/share/6824d037-a980-8002-8c10-c732d4b371cb> |
| Claude | Hongjie Zhanng | For Technical Implementation Planning and Development Environment Evaluation | yes | Reduced technical planning time through automated documentation and requirements analysis. | <https://claude.ai/new> |
| ChatGPT | Tetiana Korchynska | To get a clear idea how children’s user story might look like | yes | The model provided some powerful insights that, with some edits and rewrites, came up as a base for the concrete user stories for our app. | <https://chatgpt.com/share/68255982-fa24-8002-9615-211dea868da5> |

# 

# **References**

*(Any references/citations that you have used)*

1. **Bedtime Story AI**. (2024). *Bedtime story generator for children*. Retrieved from [https://www.bedtimestory.ai](https://www.bedtimestory.ai/)
2. **React**. (2025). *JavaScript library for building user interfaces*. Meta Platforms, Inc. Retrieved fromhttps://react.dev/
3. **Node.js**. (2025). *JavaScript runtime built on Chrome's V8 JavaScript engine*. OpenJS Foundation. Retrieved from [https://nodejs.org](https://nodejs.org/)
4. **MongoDB**. (2025). *NoSQL document database*. MongoDB, Inc. Retrieved from [https://www.mongodb.com](https://www.mongodb.com/)
5. **Express**. (2025). *Fast, unopinionated, minimalist web framework for Node.js*. OpenJS Foundation. Retrieved from [https://expressjs.com](https://expressjs.com/)

# **Glossary**

*(Any acronym used in the document should be explained here)*

| **Acronym** | **Explanation** |
| --- | --- |
| AI | ***Artificial Intelligence***  - the capability of computational systems to perform tasks typically associated with human intelligence, such as learning, reasoning, problem-solving, perception, and decision-making. |
| API | ***Application Programming Interface***  - a set of rules and specifications that allows different software programs to communicate and interact with each other. |
| CORS | ***Cross-Origin Resource Sharing***  - a browser-based security mechanism that allows a web page on one domain to access restricted resources from a server on a different domain, thus bypassing the same-origin policy. |
| CSRF | ***Cross-Site Request Forgery***  - a web security vulnerability that allows attackers to trick users into performing actions they don't intend on a website where they are already logged in. |
| CI/CD | ***Continuous Integration / Continuous Deployment***  - a set of practices and tools that automate the software development process, enabling developers to build, test, and deploy code changes more frequently and reliably. |
| IDE | ***Integrated Development Environment***  - a software application that helps programmers develop software code by providing a comprehensive set of tools in one interface. |
| JWT | ***JSON Web Token*** (JSON - JavaScript Object Notation)  - a proposed Internet standard for creating data with optional signature and/or optional encryption whose payload holds JSON that asserts some number of claims. |
| KLOC | ***Kilo Lines of Code***  - a metric used in software development to estimate the size of a program or project. |
| LOC | ***Lines of Code***  - a metric used to measure the size or complexity of a software program by counting the number of lines in the source code. |
| REST | ***Representational State Transfer***  - a software architectural style that was created to describe the design and guide the development of the architecture for the World Wide Web. |
| UI | ***User Interface***  - the space where interactions between humans and machines occur. |