Interactive book system

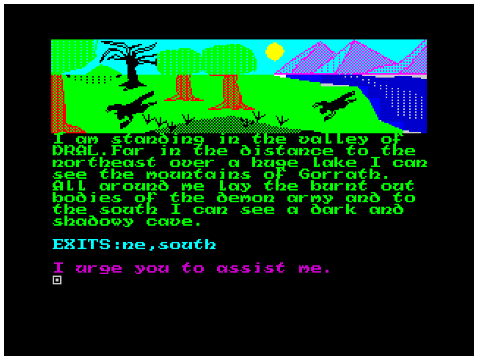
Carmel Gafa

***Abstract*—TODO**

***Index Terms*—TODO**

I. Introduction

Interactive storytelling systems have been a popular medium for delivering immersive and personalised experiences since the early days of home computing. In those early days, a genre of computer games emerged as text-based adventure games, where the story evolved based on user input through textual commands[1][5]. However, as computing power and graphical capabilities improved, the popularity of these games waned in favour of visually rich gaming experiences. Despite this decline, these games are still fondly remembered for their creativity and ability to engage.



I am standing in the valley of
DRAL.Far in the distance to the
northeast over a huge lake I can
see the mountains of Gorrath.
All around me lay the burnt out
bodies of the demon army and to
the south I can see a dark and
shadowy cave.

EXITS:ne,south

I urge you to assist me.

Fig. 1. Demon from the Darkside was released in 1986 by Compass Software for ZX Spectrum 48K[2]

Generative AI systems have the potential to introduce an innovative twist to stories by evolving based on user interaction[4] [3]. These technologies enable stories to adapt their plots in real time to align with the player's mood and style.

This paper takes a step in this direction by introducing an interactive book system—a web-based single-page application designed to ensure smooth, dynamic, and seamless user interaction. The system demonstrates the potential of Generative AI to provide engaging content that adapts to the user's age and taste, with these variables configured as system parameters.

This paper is organised into the following sections:

- Section 2 examines the system, its outputs, and navigation.
- Section 3 discusses the system architecture.
- Section 4 explores the prompts and the prompt configuration mechanism.
- Section 5 outlines potential enhancements for the system.
- Section 6 addresses the ethical considerations associated with such systems.

II. The interactive book

The interactive book system is accessible through a web browser and features a user-friendly interface to ensure readers' engaging experience.

Upon startup, the system displays the book's title page, which contains the book's name and a picture graphically aligned with the story's title, theme, and mood. The title page also contains a ”start” button, which initiates the storytelling experience when pressed.

The story will continue on the following pages, each with a segment of the story. Each page includes a speaker icon in the upper right corner, enabling a narration service to read the text to a user. This feature uses text-to-speech technology to benefit users who need or want audio output.

Every page of the story also presents two choices for users through buttons that trigger the storytelling in a particular direction. Selecting one of them influences the drama of the unfolding story, making the adventure interactive and attractive. At the end of the story, no more extended options are available, but the user can start a new story. If the user selects that option, the system resets to the title page of a newly created story.

Some examples of typical story navigation can be seen in Appendix A.

Controlling all stories are four variables. Adjusting these parameters allows the system to produce entirely different stories, making the interactive book adaptable to diverse audiences and contexts.

- Reader's Age: This parameter defines the book's target readership.
- Theme of the Book: Defines the specific theme and mood of the book and can harness personal customisation to suit user preference.
- Author Style: This is an author's name that will affect the book's tone, diction, and narrative structure.
- Number of Pages: This will determine the number of pages in the story.

Different stories suited to diverse audiences and contexts were successfully generated by changing these parameters. When writing this document, these parameters can be modified as code arguments, but this task can be done quickly enough to export it to a user interface as settings.

III. System architecture

The system utilises a three-tier architecture, with backend generative AI services and text-to-speech features coordinated by a middle layer that manages prompting and result aggregation. A frontend layer supports user interaction and displays the book's text and images. Communication among the layers occurs through RESTful APIs, which maintain a clear separation of concerns. This separation improves maintainability and scalability, allowing future expansions and integrations to be carried out quickly.

The backend application is implemented as a Python Flask application that serves as the core orchestrator for content generation and coordination. This layer integrates external APIs, including OpenAI GPT for text and image generation and Azure Cognitive Services for text-to-speech conversion, to deliver a multimodal storytelling experience. The architecture implemented in this exercise also abstracts the story parameters, prompts, and system keys into separate modules to easily extend and maintain the system.

The frontend application is developed using React, leveraging its component-based architecture to ensure a user-friendly and visually engaging interface. As a single-page application, the system eliminates unnecessary page reloads, allowing users to navigate the story seamlessly while maintaining a dynamic and interactive experience. The frontend application also includes error-handling mechanisms to manage scenarios such as backend unavailability or invalid responses. Users are notified of such errors through messages in the user interface.

A diagrammatic representation of the concepts discussed in this section is found in the system architecture diagram in Appendix B

A. *Interfaces and external dependencies*

The backend system interacts with two services that provide large-language-model, image generation and speech capabilities:

- The OpenAI ChatGPT API generates the book title, individual page content, and the image used on the title page.
- The Azure Cognitive Services (Text-to-Speech) converts story content into audio files, allowing users to engage with the narrative audibly. This capability enhances accessibility and also provides an increased multi-modal user experience. The audio files are placed in a local folder and must be served to the front-end layer for consumption.

B. *The backend system*

As described previously, the backend system of the interactive book manages the flow of data and communication between the front end, external APIs, and internal resources. The layer is organised to introduce additional functionality without changing its underlying structure. The following aspects were taken into consideration when designing this layer.

1) Integrations with large language models and text-to-speech systems are abstracted into specialised classes that provide a specific contribution to the interactive book, such as the book's writing, the image generation or the creation of the book image. These classes handle the integration with these engines and the management of message prompt arrays responsible for maintaining the context of a given interaction, which, in this case, is the evolution of the book.
2) Prompt management was abstracted into a separate module so that each integration class would call this module to inject prompts into its respective system. This decision was made to maintain consistency between the various prompts and make prompt management easier. It is safe to assume that in producing systems with similar integrations, prompt creation and management are carried out by different teams, and prompts will be contained in specialised databases. The decision to abstract prompts is a step in this direction.
3) As explained previously, all books are based on four parameters: the theme, the author's style, the target audience's age, and the desired number of pages. These parameters are also contained in a separate module, so the interactive book type can be changed quickly without changing any code.
4) Flask API endpoints are the topmost level of this architecture and provide the most generic creation capabilities that, in most cases, consume some lower-level services. Such services include the creation of a title page or the creation of a page. The results of the endpoints are provided as JSON objects to the consumers

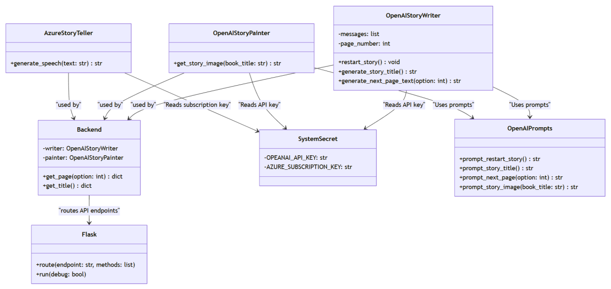

AzureStoryTeller
+generate_speech(text: str) : str
OpenAIStoryPainter
+get_story_image(book_title: str) : str
OpenAIStoryWriter
-messages: list
-page_number: int
+restart_story() : void
+generate_story_title() : str
+generate_next_page_text(option: int) : str
"used by"
"used by"
"used by"
"Reads subscription key"
"Reads API key"
"Reads API key"
"Uses prompts"
"Uses prompts"
Backend
-writer: OpenAIStoryWriter
-painter: OpenAIStoryPainter
+get_page(option: int) : dict
+get_title() : dict
SystemSecret
-OPEANAI_API_KEY: str
-AZURE_SUBSCRIPTION_KEY: str
OpenAIPrompts
+prompt_restart_story() : str
+prompt_story_title() : str
+prompt_next_page(option: int) : str
+prompt_story_image(book_title: str) : str
"routes API endpoints"
Flask
+route(endpoint: str, methods: list)
+run(debug: bool)

Fig. 2. The interactive book backend system class diagram

C. *AI integration classes*

This layer comprises three classes, each focusing on a specific interaction with OpenAI and Azure text-to-speech engines.

1) **`OpenAIStoryWriter`** creates the story's narrative using OpenAI's **`gpt-3.5-turbo`** engine. The class maintains a list of messages submitted and returned from the engine required to maintain context for a book being developed and the number of pages generated for a developing book. It has three main methods:
   - **`restart_story`** starts a new story by starting a new message array to be used with OpenAI and posting a first prompt that informs the engine about the theme, author style, target age and desired number of pages for the book to be generated.
   - **`generate_story_title`** adds the book title creation prompt to the OpenAI message array and submits the message thread to the engine. The engine, in turn, returns the name of the new book.

- **`generate_next_page_text`** generates a new page for the interactive book given a prompt.

2) **`OpenAIStoryPainter`** creates the story's title page image using OpenAI's image creation capabilities. This class does not maintain any context as it is only used to create an image based on very little information, namely the book's title. It has only one method:
   - **`get_story_image`** generates a 256 by 256-pixel image based on the book's title.
3) **`AzureAIStoryTeller`** uses Microsoft Azure's cognitive services text-to-speech capabilities to generate an audio version of a page, including the options. The class has only one method:
   - **`generate_speech`** connects to Azure's text-to-speech service, providing the text for which the audio will be generated and the location where the resulting audio file should be stored. The method maps the file to a served URL, allowing any system to access the generated audio file.

D. Secrets Storage

The backend manages all sensitive information, such as API keys and authentication tokens, through the Secrets Storage module. By isolating and protecting these credentials, the system maintains secure and reliable connections to external services and mitigates potential vulnerabilities.

E. Story parameters

This module contains the four parameters required to create a book. Parameter management is an essential aspect of dynamic systems, such as interactive books, to ensure that the nature of the artefact produced can be easily adapted and extended. Although the abstraction system used in this project is trivial, it is also quite powerful, as the few parameters contained in this file can be modified to generate books with entirely different styles and themes.

F. Prompt management

The prompt management module contains prompts for components in the backend to generate the interactive book. It utilises the story parameters module to ensure consistency and relevancy in the book creation process.

Commercial projects that rely on generative AI systems depend on teams of specialised prompt engineers who will maintain their work in databases that the application uses as required. Abstracting prompts from the codebase is, therefore, an important step in such systems to ensure that multidisciplinary teams can work together,

The table in Appendix D contains all the prompts used in this system.

G. API Interface

The API layer provides the interfaces so systems can consume the backend layer. As explained previously, it is written in Flask and contains just two endpoints:

- **`/API/title`** is the narrative's entry point. This call will instruct an **`OpenAIStoryWriter`** instance to generate a story title and an **`OpenAIStoryPainter`** instance to create a story image. It will, in turn, return an object with the following information:

```
{
"title":      "book_title",
"image_url"   "image_url"
}
```

- **`/API/page/int: option`** is called to create a new page for the story, supplying the user-selected option. This call first instructs the **`OpenAIStoryWriter`** instance to generate the new page. The writer returns a string containing the information for the new page: the image text, the page options, and whether the story is completed. The page text and options are then formatted and supplied to an **`AzureAIStoryTeller`** instance to convert the text into speech. This interaction returns the location of the audio file in a URL that external systems can access. The information is then bundled into one object and returned.

```
{
"part": "The next part of the story",
"option1": "Option 1",
"option2": "Option 2",
"status": "Status",
"textURL": "URL of the aud file for page"
}
```

IV. The frontend system

The interactive book system's front end is designed to deliver an engaging user experience. It is built using ReactJS as a single-page application that communicates with the backend tier to retrieve the dynamic content. The following sections discuss the frontend tier's components and workflows in more detail.

At the core of the frontend tier is the **`Book`** component, which serves as the interactive interface through which the user interacts with the book's content. The book component has two modes of display:

- Title Page Display: If the book is at its starting point (page number = 0), the component retrieves and renders the book's title and the associated title image provided by the backend.
- Page Content Display: For all subsequent pages, the component displays the text content of the current page along with any user-selectable options for advancing the story.

In addition, this component monitors the story status for changes and re-renders to reflect the latest content retrieved from the backend. The status is particularly important when the last page of the interactive book is reached, as the book component will not display any options but prompts the user to start a new book

The frontend also streams audio content from the backend tier, which, as explained previously, is generated using Azure Cognitive Services. The speech functionality seamlessly integrates into the user interface, enabling easy playback and pausing.

REFERENCES

[1] Will Brooker. “Filling in the gaps: Creative imagination and nostalgia in ZX Spectrum Gaming”. In: *Participations: Journal of Audience and Reception Studies* 18.2 (2021), pp. 3–43.

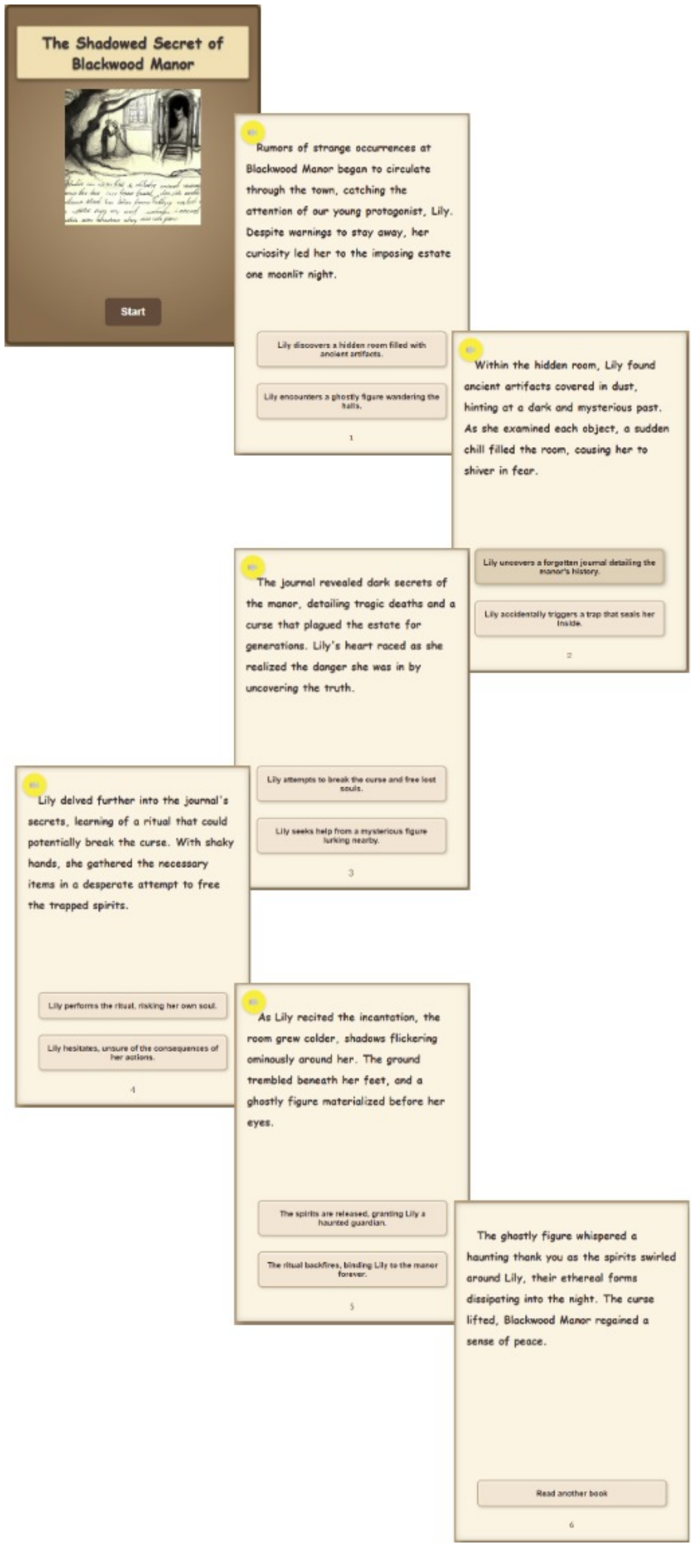
[2] *Demon from the Darkside*. URL: https://worldofspectrum.org/archive/software/text-adventures/demon-from-the-darkside-compass-software.

[3] Sonali Fotedar et al. “Storytelling AI: A Generative Approach to Story Narration.” In: *AI4Narratives@ IJCAI*. 2020, pp. 19–22.

[4] Marko Vidrih and Shiva Mayahi. “Generative AI-Driven Storytelling: A New Era for Marketing”. In: *arXiv preprint arXiv:2309.09048* (2023).

[5] Colin Woodcock. *The ZX Spectrum on Your PC*. Lulu.com, 2012.

Appendix A
Story Navigation


The Shadowed Secret of Blackwood Manor
Start
Lily discovers a hidden room filled with ancient artifacts.
Lily encounters a ghostly figure wandering the halls.
1
Lily uncovers a forgotten journal detailing the manor's history.
Lily accidentally triggers a trap that seals her inside.
2
Lily attempts to break the curse and free lost souls.
Lily seeks help from a mysterious figure lurking nearby.
3
Lily performs the ritual, risking her own soul.
Lily hesitates, unsure of the consequences of her actions.
4
The spirits are released, granting Lily a haunted guardian.
The ritual backfires, binding Lily to the manor forever.
5
Read another book
6

Rumors of strange occurrences at Blackwood Manor began to circulate through the town, catching the attention of our young protagonist, Lily. Despite warnings to stay away, her curiosity led her to the imposing estate one moonlit night.

Within the hidden room, Lily found ancient artifacts covered in dust, hinting at a dark and mysterious past. As she examined each object, a sudden chill filled the room, causing her to shiver in fear.

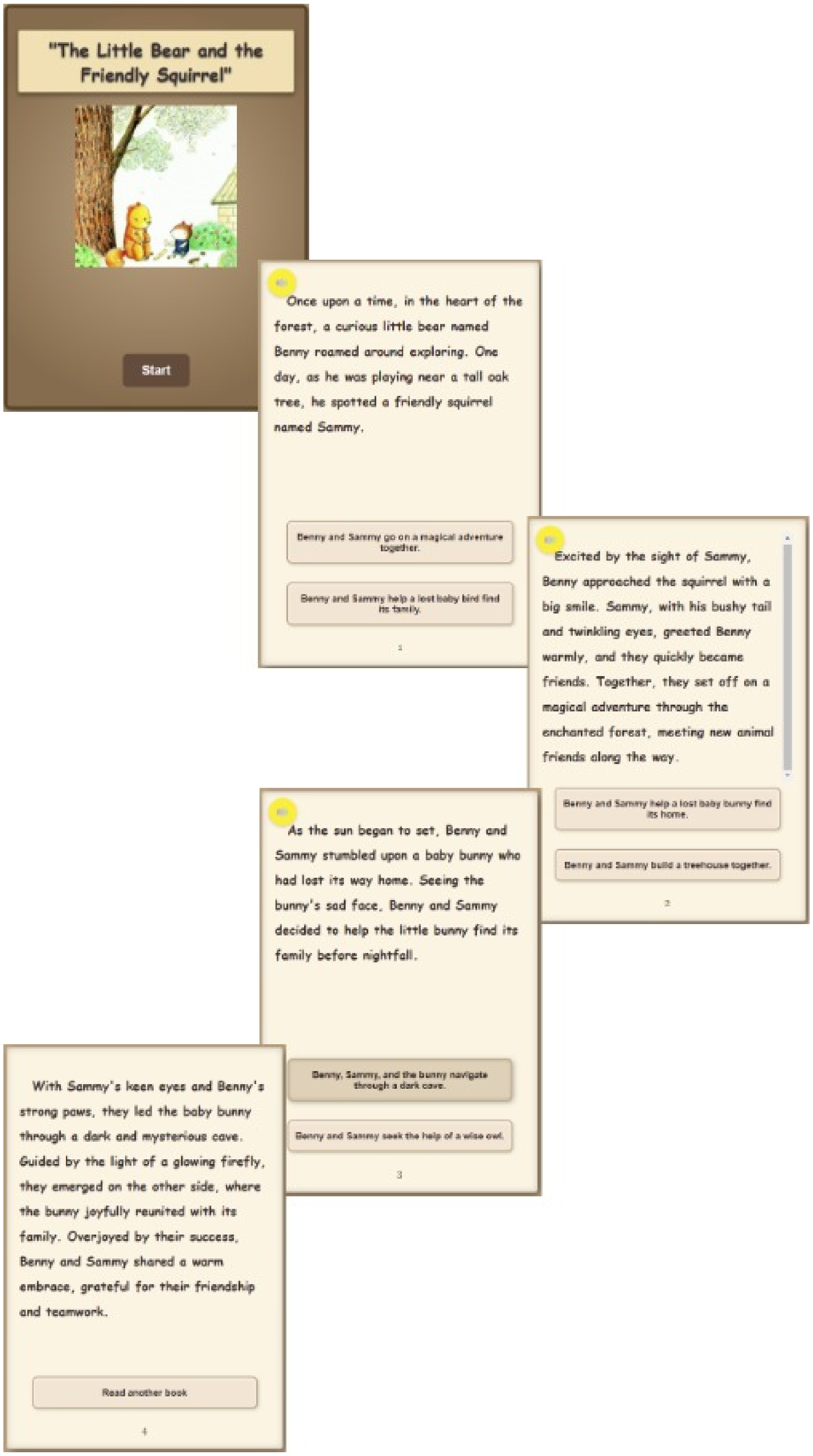
The journal revealed dark secrets of the manor, detailing tragic deaths and a curse that plagued the estate for generations. Lily's heart raced as she realized the danger she was in by uncovering the truth.

Lily delved further into the journal's secrets, learning of a ritual that could potentially break the curse. With shaky hands, she gathered the necessary items in a desperate attempt to free the trapped spirits.

As Lily recited the incantation, the room grew colder, shadows flickering ominously around her. The ground trembled beneath her feet, and a ghostly figure materialized before her eyes.

The ghostly figure whispered a haunting thank you as the spirits swirled around Lily, their ethereal forms dissipating into the night. The curse lifted, Blackwood Manor regained a sense of peace.

Fig. 3. Interactive book navigation for a horror story



"The Little Bear and the Friendly Squirrel"

Once upon a time, in the heart of the forest, a curious little bear named Benny roamed around exploring. One day, as he was playing near a tall oak tree, he spotted a friendly squirrel named Sammy.

Benny and Sammy go on a magical adventure together.
Benny and Sammy help a lost baby bird find its family.
1

Excited by the sight of Sammy, Benny approached the squirrel with a big smile. Sammy, with his bushy tail and twinkling eyes, greeted Benny warmly, and they quickly became friends. Together, they set off on a magical adventure through the enchanted forest, meeting new animal friends along the way.

Benny and Sammy help a lost baby bunny find its home.
Benny and Sammy build a treehouse together.
2

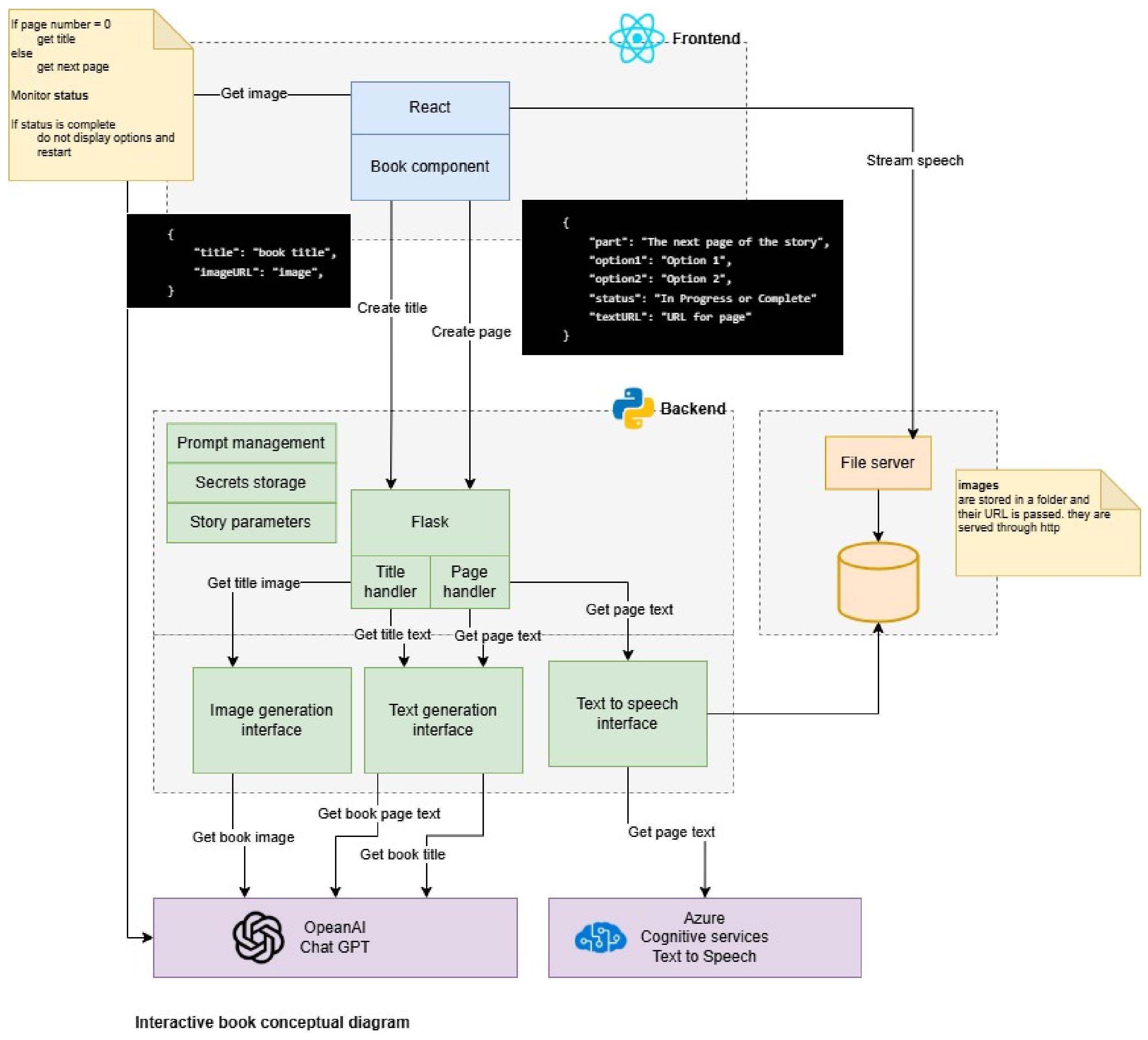
As the sun began to set, Benny and Sammy stumbled upon a baby bunny who had lost its way home. Seeing the bunny's sad face, Benny and Sammy decided to help the little bunny find its family before nightfall.

Benny, Sammy, and the bunny navigate through a dark cave.
Benny and Sammy seek the help of a wise owl.
3

With Sammy's keen eyes and Benny's strong paws, they led the baby bunny through a dark and mysterious cave. Guided by the light of a glowing firefly, they emerged on the other side, where the bunny joyfully reunited with its family. Overjoyed by their success, Benny and Sammy shared a warm embrace, grateful for their friendship and teamwork.

Fig. 4. Interactive book navigation for a children's story

APPENDIX B
SYSTEM ARCHITECTURE DIAGRAM



```
If page number = 0
      get title
else
      get next page

Monitor status

If status is complete
      do not display options and
      restart
```

```
{
     "title": "book title",
     "imageURL": "image",
}
```

```
{
     "part": "The next page of the story",
     "option1": "Option 1",
     "option2": "Option 2",
     "status": "In Progress or Complete"
     "textURL": "URL for page"
}
```

Fig. 5.

APPENDIX C
INTERACTIVE BOOK PROMPTS

Prompt name	Prompt text
`prompt_restart_story`	You will create a story for a **`READER_AGE`**-year-old using the style of **`BOOK_STYLE`**. The story should have a simple structure with an introduction, a conflict, and a resolution. Each 'page' of the story should be two sentences long, except for the final page,which should contain only the words **`'The End'`** and nothing else. Make it engaging and age-appropriate but make sure that there is **`BOOK_THEME`**.The story should be NOT MORE THAN **`MAXIMUM_PAGES`** pages long. Introduce the main character of the story in the first page.
`prompt_story_title`	Give me only the title of the story.
`prompt_next_page`	Given that I selected option **`OPTION`**, give me the next part of the story in a paragraph that extend the story so far and on the option I selected.Give me also two possible ways to evolve the story, that should be not more than ten words; unless you are on the final page. In this case, the text will be **`'The End'`**. Label them **`'Option 1'`** and **`'Option 2'`**. Give me also a status for the story, which can be one of the following: **`'In Progress'`** or **`'Complete'`** .Package your reply in a JSON object with the following format: **`{"part": "The next part of the story", "option1": "Option 1", "option2": "Option 2", "status": "Status" }`**
`prompt_story_image`	A pencil-drawn illustration evoking a storybook atmosphere of **`BOOK_THEME`**. The drawing style is aligned with the writing of **`BOOK_STYLE`**. The title of the story is **`TITLE`**