Katabasis2
Final Project Report

Requirements, Design, Implementation & Testing

Katabasis

CSC 492 Team 45

Nomar Ledesma Gonzalez, Katie Hammer, Ryan Mikula, Laura Yang, Joey Zhou

North Carolina State University
Department of Computer Science

April 26, 2025

Executive Summary

Author(s): Katie Hammer
Reviewer(s)/Editor(s): Laura Yang

Sponsoring Organization and Business

Katabasis is a non-profit organization based in North Carolina, dedicated to bridging educational gaps through innovative software solutions. They focus on creating adaptive learning tools for underserved communities, with a current goal of developing a digital learning environment for students with Emotional and Behavioral Disorders (EBD).

Business Case

Students with EBD face significant challenges understanding the long-term consequences of their decisions, resulting in higher rates of disciplinary actions and poor academic performance. Over time, these challenges lead to concerning outcomes: high dropout rates, unemployment, limited participation in postsecondary education, social isolation, and increased involvement in juvenile and adult crime. Existing educational tools and methods fail to provide engaging and scalable solutions tailored to the specific needs of these students.

Proposed Solution

Our team is developing an interactive visual novel application specifically designed for students with EBD (ages 8-15) that helps them understand the repercussions of their classroom behavior and decisions. The app is designed to help these students understand the consequences of their behavior and decisions in classroom settings through an engaging school-based interactive storytelling experience featuring relatable scenarios. Users navigate through narratives by making choices that directly impact the storyline. The experience is personalized by initially collecting variables to determine the most relevant scenario for each student. Additionally, the application integrates seamlessly with the OpenDI framework using compatible JSON files to generate Causal Decision Diagrams (CDDs). Its branching narrative structure clearly demonstrates how individual decisions, external factors, and intermediate events collectively influence outcomes. To encourage meaningful self-reflection, the application incorporates clear and supportive feedback systems rather than explicit instruction.

Project Status and Next Steps

In our initial preparation phase, we focused heavily on requirements gathering and research. We conducted paper prototype testing with representatives from our target audience (students aged 8-15 with EBD) to clearly define core scenarios based on common classroom incidents. We set up our basic Renpy scenario and established our Renpy backend.

In our first iteration, we concentrated on enhancing the Renpy branching scenario and polishing visual and interactive effects. We developed a variable system setup that allows for more player customization and created a schema for CDD integration.

In our second iteration, we focused on refining the game's details by integrating mini-games, audio enhancements, and additional interactive effects. We implemented visualization capabilities for our CDD and generated OpenDI-compatible JSON files. Additionally, we expanded our branching scenarios and further refined the variable system to support more nuanced player interactions.

During the third iteration, we aimed to expanding the content of the game. We created and implemented new scenarios and continued enhancing the visualization of our CDD to demonstrate decision pathways and their impacts.

In our fourth iteration, we have exported the game schema for integration with external tools using OpenDI-compatible JSON formats. We're actively refining variable branching mechanics and are creating and integrating another new scenario. A thorough game review and integration process is underway to ensure seamless interaction and complete functionality.

For next steps, please refer to Suggestions for Future Teams.

Project Description

Author(s): Nomar Ledesma Gonzalez
Reviewer(s)/Editor(s): Katie Hammer

Sponsor Background

Katabasis is a non-profit 501(c)(3) organization based in North Carolina. The organization is dedicated to developing innovative educational software for underserved communities. Katabasis is committed to bridging gaps in educational accessibility, with a mission to empower students, particularly those facing unique challenges such as emotional & behavioral disorders (EBDs), by building digital learning environments that adapt to individual needs and cultivate mindsets of growth and reflection to enhance learning, inspire curiosity, and foster development. The organization works with educational institutions to create tools that support evidence-based approaches for reaching students who often struggle in traditional classroom settings. Katabasis focuses on developing adaptive learning technologies that can address diverse educational needs through innovative software solutions.

Sponsor Name & Title	**Sponsor Contact**
President of Katabasis, Bill Causey	bill.causey@katabasis.org
VP of Katabasis, Joseph B. Wiggins	joseph.wiggins@katabasis.org
Technical Lead, Jorge Parra	jorge.parra@katabasis.org
Developer, Alex Cail	alex.cail@katabasis.org
Developer, Ben Taylor	ben.taylor@katabasis.org

Problem Description

Students with emotional and behavioral disorders (EBDs) face significant challenges in managing their actions and understanding the long-term consequences of their decisions. These students experience increased rates of office discipline referrals, school suspensions, and poor academic performance. Over time, these outcomes can lead to high school dropout, unemployment, limited participation in postsecondary education, social isolation, and increased involvement in juvenile and adult justice systems.

Despite educators' efforts to support these students, existing tools and methods fail to provide engaging and scalable solutions tailored to their specific needs. Traditional behavioral interventions often lack the interactive elements necessary to capture students' attention and help them visualize the connection between their choices and outcomes.

Katabasis aims to address these challenges by developing interactive applications that empower students with EBDs to structure and reflect on their decisions. The software will incorporate principles from the Decision Intelligence (DI) framework, providing a structured approach to decision-making through Causal Decision Diagrams (CDDs). These diagrams visually represent the relationships between actions, intermediate factors, external influences, and outcomes, helping students better understand how their choices lead to consequences.

The core problem our project addresses is the lack of age-appropriate, engaging tools that can help students with EBDs:

- Recognize the factors that influence their emotional and behavioral responses
- Understand the causal relationships between their actions and outcomes
- Develop better decision-making skills through interactive reflection
- Practice these skills in a safe, consequence-free environment

Success will be measured by the system's ability to increase student engagement (tracked through interaction metrics), improve students' ability to identify and reflect on factors influencing their decisions, and provide educators with insights into students' thought processes to enable more targeted support.

Proposed Solution & Project Goals/Benefits

Our project develops an interactive visual novel application designed specifically for students aged 8-15 with emotional and behavioral disorders (EBDs). Using Ren'Py, a game engine optimized for choice-driven narratives, we're creating an immersive school-based environment where students can safely explore the consequences of classroom behaviors and decisions. The application addresses five distinct scenarios representing the most common reasons students are removed from classrooms—classroom disruption, work refusal, emotional outburst, disrespectful language, and peer conflict—based on our research with educators experienced in working with EBD populations. Through this experiential learning approach, students can develop greater self-awareness of behavioral triggers and outcomes without direct instruction.

The core features of the system will include:

- **Personalized Experience**: An introductory sequence collects user information (mood, sleep quality, previous experiences) and other responses to situations. Analysis of these will determine which of the five scenarios most closely align with the user’s situation.
- **Animal Character Design**: Custom hand-drawn animal characters to make the experience more engaging and entertaining for our target age group.
- **Branching Narrative Structure**: Each scenario offers multiple decision points where students can make choices that lead to different outcomes, allowing them to experience the consequences of their actions in a safe environment.

- **Enhanced CDD Visualization**: We're implementing a custom version of the OpenDI Casual Decision Diagram authoring tool using Cytoscape to create more age-appropriate, visually appealing representations that our target audience can easily understand.
- **Dynamic Feedback System**: Rather than explicit instruction, the application provides natural consequences within the narrative that prompt self-reflection and help students understand the relationship between their choices and outcomes.
- **Data Export Compatibility**: Generated decision data will be exported in a format compatible with OpenDI tools, allowing for integration with existing educational frameworks and further analysis.

Stakeholder Benefits

For students with EBDs, the application provides an engaging, non-threatening environment to explore behavioral choices without real-world consequences, potentially improving their self-regulation skills and classroom success. Parents gain access to a tool their children can use independently to develop critical decision-making awareness.

For educators and counselors, the system offers a supplementary intervention that can be implemented with minimal supervision, providing insights into students' decision-making patterns while reinforcing classroom behavioral expectations. The visual representations of decision processes can also serve as effective discussion tools during one-on-one interventions.

For Katabasis and the educational community, this project demonstrates how decision intelligence frameworks can be adapted for specialized populations, creating a foundation for future applications that address various learning and behavioral challenges.

The primary benefits of our solution include:

1. Increased student engagement through an interactive, game-based approach to behavioral learning
2. Development of metacognitive skills as students connect emotions, actions, and consequences
3. Reduction in classroom disruptions as students gain better self-regulation strategies
4. Valuable data collection for educators to better understand student decision-making processes
5. A scalable tool that can be implemented across multiple educational settings

Resources Needed

Author(s): Nomar Ledesma Gonzalez, Laura Yang
Reviewer(s)/Editor(s): Katie Hammer

Below is a list of the resources integral to our project.

Table 1: Project Resources

Resource Name	**Brief Purpose**	**Status**	**Version**	**Licensing Information**
OpenDI Framework	Integrates & displays Causal Decision Diagrams (CDDs) for users to reflect & modify components of their decisions.	🟢Obtained	Latest Stable (as provided by Sponsor)	Open-Source
Ren'Py	Primary engine for creating interactive visual novel experiences with branching narratives.	🟢Obtained	8.3.4	MIT License GNU Lesser General Public License Python License Jpeg/PNG License Zlib License Bzip2 License GNU GPL
Python	Core programming language to support scripting, logic, & backend integration	🟢Obtained	3.9	Python Software Foundation License
Flask (superseded)	Lightweight web framework for backend services & api endpoints	🟢Obtained	3.1.0	BSD License
Twine	Tool for storyboarding and designing branching narratives prior to integration.	🟢Obtained	2.10.0	GNU GPL v3
Firebase	Cloud-based database and	🟢Obtained	Web v9	Proprietary (Google) with

	authentication services for user data storage			Free Tier
Cytoscape	JavaScript library for graph visualization to enhance CDD displays	🟢Obtained	3.10.3	MIT License
Character Assets	Custom-designed animal characters for the visual novel	🟢Obtained	N/A	Original Work
Background Assets	School environment imagery for scene settings	🟢Obtained	N/A	Royalty-Free License
HTML	For the frontend of the new CDD tool	🟢Obtained	HTML5	No license
CSS	For the styling of the CDD tool	🟢Obtained	CSS 4.15	No license
REST API	To connect the CDD tool with the game	🟢Obtained	N/A	GitHub REST API Salesforce REST API Tableau REST API
Virtual Machine and/or Firebase	To host the program	🟢Obtained	N/A	Virtual Computing Lab OIT Microsoft Virtual Desktop Infrastructure

Risks & Risk Mitigation

Author(s): Joey Zhou, Katie Hammer
Reviewer(s)/Editor(s): Nomar Ledesma Gonzalez

Table 2: Project Risks and Mitigations

Risk	**Mitigation Strategy**
Insufficient Engagement for EBD Students - The visual novel may fail to engage students effectively or may not be tailored to their specific needs. - As a result, this can lead to low adoption & motivation towards the game and reduce the learning outcomes	- Research: Involve teachers experienced with EBD students, research online resources. - Iterative Prototyping: Playtesting with user tests including target audience to gather feedback. - Adaptive Content: Adjust content on user actions, dialogue, tailored scenarios as a result of player actions. - Usability: Ensure navigation and mechanics are simple and intuitive for our demographic.
Emotional Triggers - Certain scenes, choices, or storylines might unintentionally trigger negative emotional responses or harmful behaviors in students (i.e., bullying).	- "Safe Exit" Options: to allow students to pause, exit, or skip challenging scenarios. - Content Sensitivity Testing: Usability testing to ensure scenarios are appropriate - Collaboration & Feedback with Katabasis
Superficial Participation - Students may select choices mechanically without meaningful reflection, undermining the educational goals. - As a result this will lead to minimal improvement in decision-making skills.	- Reflections: Integrate short prompts that ask the students to articulate reasons for their choices. - Adaptive Responses: Dynamically alter storylines based on interaction patterns to encourage deeper thinking.
Complex Causal Decision Diagrams - Children may struggle to understand the CDDs if they appear too technical or abstract, limiting meaningful reflection on actions and outcomes. - Reduced comprehension of the cause-and-effect relationships. - Missed learning opportunities for increasing decision intelligence	- Updated CDD: Colorful UI with simplified visuals using intuitive icons and labels. There will be an explanation by a story-based character who explains the diagrams choices and outcomes in a way that is easy to understand. - Usability Testing: Observe how students interact with the diagrams & gather feedback on comprehension & adjust visuals or explanations accordingly.

	- Contextual Cues: Show diagram snippets at key decision points, reinforcing the links between actions, external & intermediate factors and outcomes.
Technology Integration Challenges - OpenDI framework is hard to integrate with Ren'Py - JSON data generated may not be compatible with visualization needs	- Develop new authoring tool for easier backend operation - Develop tools to generate compatible JSON schema
Ecological Validity - Without access to real EBD students during development, scenarios might not reflect authentic classroom experiences	- Expert Consultation: Regular meetings with our sponsors who have worked with EBD students to validate our authenticity. - Research: Use real-world examples to inform scenario development.
Time Constraints - Developing five complete scenarios with branching narratives may exceed available time - CDD visualization customization could require more development time than anticipated	- Phased Development: Focus on completing one high-quality scenario fully before expanding to others. - Prioritization: Identify core features vs "nice-to-have" elements and ensure essential components are completed first. - Regular Timeline Review: Check-ins with sponsors, teaching staff, and student team to make sure goals all align.

Development Methodology

Author(s): Laura Yang, Nomar Ledesma Gonzalez, Ryan Mikula
Reviewer(s)/Editor(s): Katie Hammer

Overall View

The Agile methodology was selected due to its flexibility and iterative nature, ensuring continuous improvement based on user feedback.

Moreover, our team follows an iterative process, a core part of Agile, to manage development effectively. We use sprints within each iteration to measure progress and break down iterations into tasks in more detail.

Iteration Execution

- We begin the Agile sprint by reviewing our project backlog, refining designs, estimating effort, and delegating tasks. During each sprint, we adhere to incremental development,

building features in manageable chunks. A core part of our development cycle is iterative usability testing. As we built scenarios and developed our Casual Decision Diagram we conducted usability testing sessions to gather feedback directly from players and observe their interactions with our game. This feedback directly informs the direction of our design iterations and bug fixing. While we employ automated testing for and test-driven development for foundational code where beneficial (such as grouping the player into a scenario based on their decisions), usability and playtesting were our primary methods for validating the player experience. As a team, we hold two to three meetings a week with all members of the group to discuss progress, roadblocks, upcoming tasks & collaboration.

- We conduct weekly check-ins with both Katabasis & the NCSU teaching staff where we present our latest increments, gather feedback, and refine the backlog for the next sprints. This is critical to detect issues and align with our sponsors' requirements early.
- At the end of each sprint, we hold a review to demonstrate the completed features. Additionally, each team member will present an oral progress report presentation to the NCSU teaching staff, students & Katabasis sponsors.

We considered an iterative process to best fit our development methodology as our project requires continuous input from Katabasis professionals, specialists & educators to ensure content is age appropriate and emotionally safe. An iterative approach lets us incorporate feedback quickly. Additionally, our methodology mitigates our risk & by developing in short cycles, we can validate assumptions and engagement strategies early, reducing the risk of major failures late in the project.

Collaboration and Tools

- Version Control: We utilize Git (e.g. GitHub) to manage our codebase. Each developer works off feature branches.
- Communication: We rely on various communication channels for daily messaging and asynchronous updates, ensuring the entire team remains aligned to the project goals. We have Slack channels to communicate with the teaching staff, and a Discord server for casual team discussions.
- Documentation and Shared Resources: Meeting notes, design specifications, and other documents are stored in a central drive for easy access. This approach enables transparency and collaboration.
- Design and Prototyping: We use tools like Twine for initial narrative storyboarding and wireframes, helping us visualize branching paths before implementation. Mockups and user flows may also be created through paper prototyping or other storyboarding software.
- Versioned Asset Management: Game assets are stored in dedicated folders within our GitHub repository. We track changes to these files in the same way we track code.
- Feedback Collection: We integrate user and sponsor feedback into our shared drive and discuss them during team meetings. Sprint reviews provide opportunities to reflect on the process and refine our approach toward implementation.

System Requirements

Author(s): Ryan Mikula, Nomar Ledesma Gonzalez
Reviewer(s)/Editor(s): Laura Yang, Joey Zhou

Overall View

From the perspective of our target audience (students with emotional and behavioral disorders), our project presents an interactive visual novel, story application where their in-game choices dynamically shape the narrative. The system will generate a visual Causal Decision Diagram (CDD) behind the scenes, illustrating how the student's decisions, intermediate factors, and external influences lead to specific outcomes. The primary users of this application will consist of students 8-15 years of age with EBDs & the secondary users will be educators and counselors.

- The primary users (students with EBDs) will engage with a branching story that reflects real-life decision-making scenarios in a school setting. These students will receive feedback from a CDD that highlights the cause-and-effect relationships, helping them reflect on their behavioral patterns. The existing OpenDI system lacks intuitive diagrams and explanations. Our team will make over the existing OpenDI system with an intuitive, colorful display of the CDD. This CDD will be explained in a simple manner by a character from our story.
- Our secondary users (educators & counselors) may review the generated CDDs to understand the student decision process. These insights can be used to personalize interventions or classroom strategies.

This application will meet the needs of our sponsor Katabasis in several ways: targeting underrepresented communities, providing a scalable educational app, and empowering the connection between educators and students.

Glossary:

- EBD: Emotional and Behavioral Disorders
- OpenDI Framework: An open-source toolkit for creating and displaying CDDs.

Functional Requirements

Gamified Elicitation Interface

FR-1: The interface shall present questions and prompts as a form of elicitation to gather inputs.

FR-2: Positive reinforcement through encouraging and congratulatory messages upon completion of levels, and actions. For instance developing a positive relationship with other characters.

Decision Intelligence Framework

FR-3: The system shall produce OpenDI-compatible app to structure decisions and generate CDDs

FR-4: The system shall identify and represent:

i. Levers/Actions: Choices or actions a student can make
ii. External Factors: Environmental influences
iii. Intermediate Factors: Steps follow the actions
iv. Outcomes: Results of player actions at the end of their route

Cause & Effect Relationships

FR-5: The system will provide a clear, visual representation of the cause and effect relationships taken throughout the game using a causal decision diagram.

FR-6: The diagram will be modifiable to represent various levers with their intermediates & outcomes. The result shall be updated automatically.

FR-7: System shall collect & store all student Question Responses

FR-8: Collection of Student Actions. System shall interpret these actions as levers to properly configure a CDD with details.

Adaptive Story Progression

FR-9: The system shall manage branching paths, ensuring narrative evolves based on user decisions and reflects their behavior within the game context.

Non-Functional Requirements

NFR-1: The application operates consistently across common web browsers (e.g., Chrome, Firefox, Safari, Edge)

NFR-2: The system will promote accessibility ensuring text, visuals, & interactive elements are perceivable and operable by students with diverse needs.

NFR-3: The user interface must be responsive, adjusting layout and scaling for varied screen sizes and resolutions without compromising functionality.

NFR-4: All user-generated data or sensitive information must be stored securely

NFR-5: System shall utilize OpenDI framework for CDD generation in accordance with sponsor-provided data standards. OpenDI framework usage should be documented.

NFR-6: The system must provide a gamified, interactive interface to guide students through decision-making scenarios. The system shall run in Ren'Py with frequent decisions and reflections that dynamically update the intermediates, and outcomes.

Constraints

- The system must integrate with the OpenDI framework, specifically for gathering and displaying Causal Decision Diagrams (CDDs).
- The project must run on lower-end systems to maximize accessibility
- Dialogue, interface, content must be age appropriate for the average middle schooler.
- The team must maintain a consistent art style and theme throughout the application.
- The code must adhere to unit and acceptance testing standards and ensure sufficient coverage and quality.
- Application must be demonstrable in real time via HTTP tunneling
- Multi-language support: Depending on the target user group, consider adding multi-language interfaces to facilitate the use of students and teachers from different language backgrounds.

Gameplay Requirements

Author(s): Nomar Ledesma
Reviewer(s)/Editor(s):

We've adopted a gameplay requirements approach for the interactive visual novel portion of the project. This section outlines the core characteristics and mechanics of the game.

Genre: Interactive Visual Novel

Playable Characters:

- Player character: The student embodies the primary character as themselves navigating the scenarios. The player's background and initial characteristics are influenced by the elicitation phase.
- Non-Player Characters: Supporting characters within the narrative include:
    - Mousy, Crusher & Bandit (Peers)
    - Ms. Johnson (Teacher) & Donkey the Assistant Teacher (Administrator)

Objectives:

- Primary Objective from the player perspective is to progress through the narrative scenarios, make decisions, and explore the resulting storylines and outcomes.
- Educational Objectives:
    - Immerse student in relatable school-based decision-making scenarios
    - Prompt student reflection on the relationship between their decisions, influencing factors and outcomes.
    - To facilitate understanding of causal relationships in decision-making through visualization of Causal Decision Diagrams.
    - Collect data on student choices & paths for educational insight & personalization.

Game Mechanics & Rules

- Narrative Progression: The game follows a story structure presented by dialogue boxes, character sprites, background images and advancing my making actions or dialogue choices.
- Decision Points: At specific portions in the narrative, players are presented with explicit choices or actions to take.
- Branching Narrative: Player choice influences path of the story leading to various dialogues, scenes, and factors of the CDD (Levers, Intermediate factors & Outcomes)
- Elicitation Phase: Initial interactive opening sequence to gather information about the player's

mood, experiences, or responses to initial situations to inform scenario selection & personalized dialogue.

- Scenario Selection: Based on data collected from elicitation phase, the system will dynamically select and present the most relevant and tailored follow-up scenarios capturing one of five behaviors: classroom disruption, elopement, work refusal, emotional outburst, disrespectful language.
- In-Game Feedback: Coupled with decision points the narrative itself will provide consequences both immediate and delayed that reflect the impact of the players decisions.
- CDD Feedback: Upon reaching a significant outcome or conclusion within a scenario path, the system will present a visual Causal Decision Diagram illustrating the players specific path, highlighting their chosen actions, intermediate factors, external influences and final outcome.
- CDD Interaction: Our revised visual representation of the CDD is designed for our target audience with simplified visuals and familiar icons.

Design

Author(s): Ryan Mikula, Katie Hammer
Reviewer(s)/Editor(s): Laura Yang

High-Level Design

Our high-level design illustrates an interactive framework where user actions feed into a branching gameplay to provide an enhanced experience. The details are shown in Figure 2:

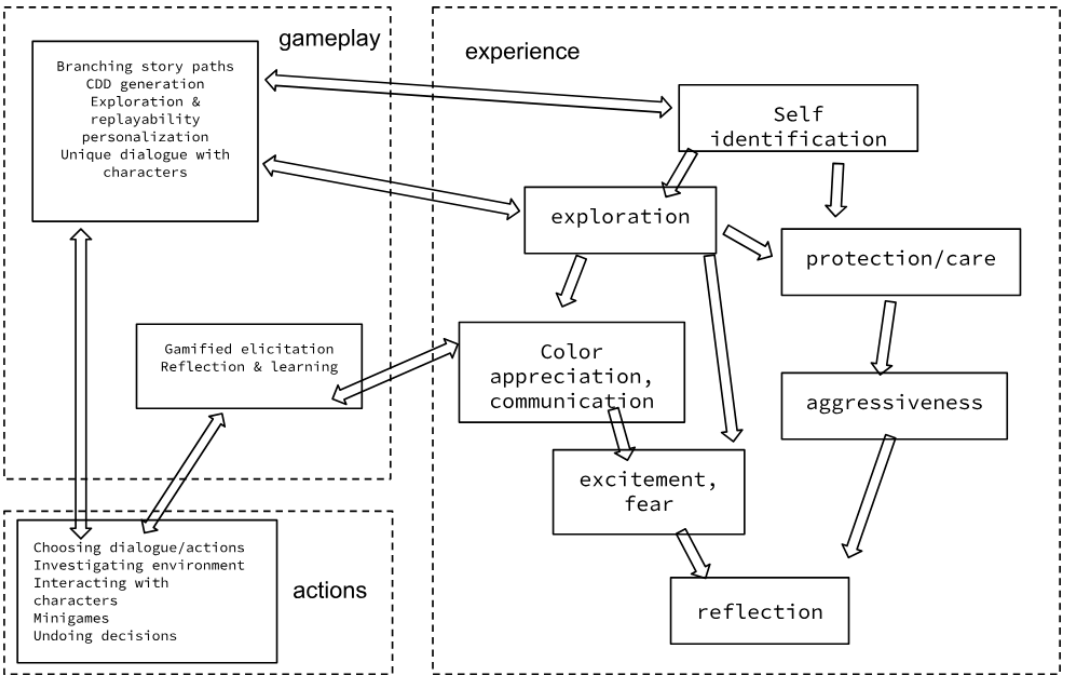

Figure 2: AGE Diagram of High-Level Design

Our project implements an interactive educational system designed to help middle school students with emotional and behavioral disorders (EBD) understand the consequences of their actions and develop better decision-making skills. The architecture consists of two primary interconnected components:

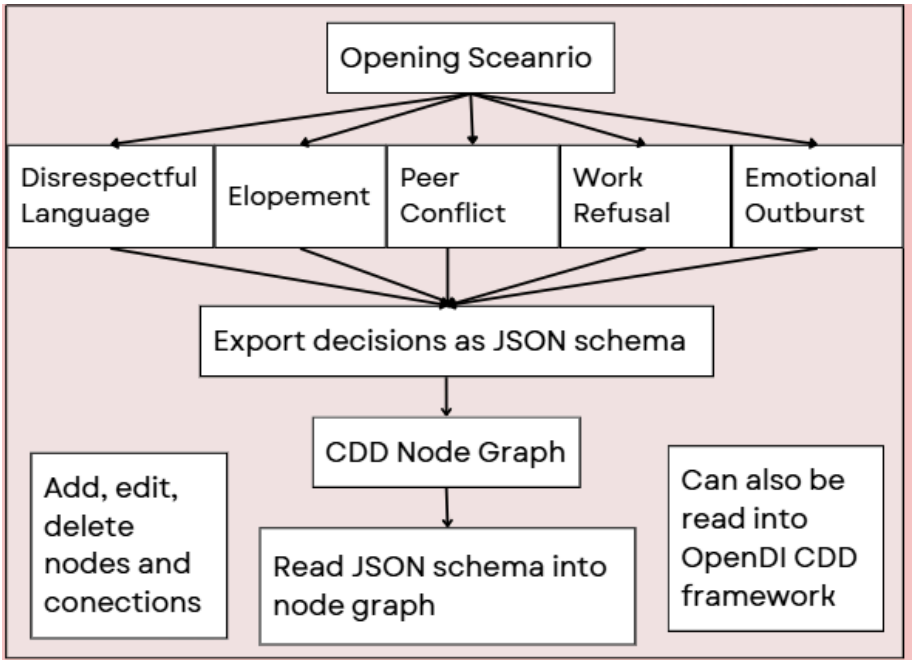

Figure 3: High-Level Design of Ren'Py with OpenDI Integration

1. **Interactive Storytelling Engine (Ren'Py)**: Provides the narrative framework where students explore classroom scenarios related to behavioral issues, particularly focusing on situations that might lead to being removed from a classroom. This component handles user input, branching storylines, and personalization based on individual factors.
2. **Decision Intelligence Framework (OpenDI)**: Translates user choices and narrative paths into Causal Decision Diagrams (CDDs) that visualize the relationship between actions, intermediates, and outcomes, helping students understand the causal connections in their decision-making process.

As illustrated in our architecture diagram, the system is organized into three major domains:

- **Gameplay**: Contains the core interactive elements, including branching narrative paths, CDD generation capabilities, story exploration mechanics, replayability features, personalization options, and unique character dialogues. This domain leverages Ren'Py's visual novel capabilities for delivering engaging, age-appropriate content.
- **Experience**: Represents the psychological and educational elements of the system, including self-identification, exploration, protection/care dynamics, aggressiveness triggers, excitement/fear responses, reflection opportunities, and communication mechanisms. This domain is structured to support the emotional learning objectives.
- **Actions**: Encompasses the concrete interaction points, including dialogue/action choices, environmental investigation, character interactions, mini-games, and decision reversal options.

Data flows bidirectionally between these domains as students navigate through scenarios. The system collects user input (name, mood states, response choices) and generates personalized narrative paths and corresponding decision diagrams. The OpenDI framework processes the decision pathways to create visual representations that help students recognize patterns in their decision-making without explicitly telling them what they should have done.

This architecture supports our pedagogical approach of guided discovery, allowing students to gain insights into their behavioral patterns and emotional responses through interactive storytelling rather than direct instruction.

Low-Level Design

The low-level design of our system can be broken down into several different components. For instance, the system contains Ren'Py novel creation, OpenDI schema creation, CDD node graph visualization, as well as research pathways into OpenAI GPT integration and Flask backend. Some architecture components (e.g. OpenAI and Flask have been superseded by a simpler design, but still remain useful and active research pathways).

1. Core System Architecture

   1.1 Ren'Py Engine Integration

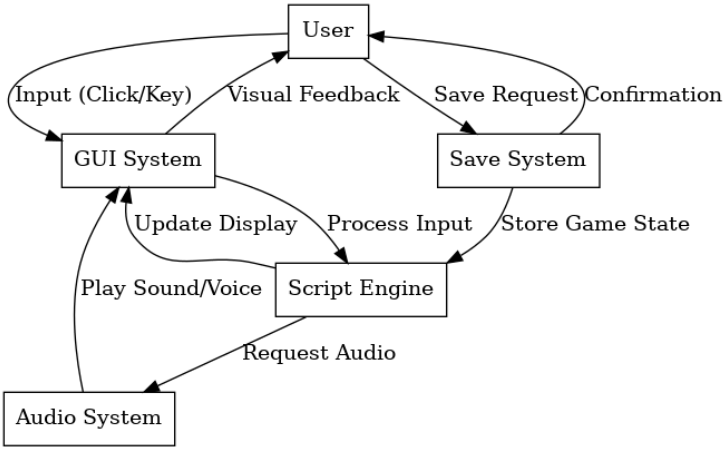

Figure 3.1: Core System Data Flow

The Ren'Py engine has allowed a seamless integration of our storyboarded scenarios. The engine can be described broadly as having the following systems:

1. User
2. GUI System
3. Save System
4. Script Engine
5. Audio System

Each system interacts with a different system through updates, input processing, asset requests, etc. The engine automatically abstracts each system and provides a simple way to create games in a modified version of Python, .rpy.

2. OpenDI Schema Exportation

   2.1 Integration of our own API

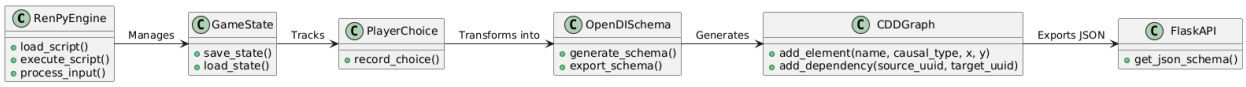

Figure 3.2: Schema Export API

We created our own custom API that allows the seamless creation of JSON schema in the OpenDI format:

The game begins by defining a global game schema. This game schema is set as a constant placeholder in the OpenDI format. We then add to this global schema through the CDDGraph API we created.

This API consists of two main features: add_element and add_dependency. Developers can call add_element to create a node in the graph. During the creation of this node, developers can pass name, causal_type, and the x, y coordinates of this node. This feature allows programmers to go through each scenario in Ren'Py and construct the graph by calling these functions. Developers can call add_dependency and pass two uuid's of the nodes. The first uuid parameter is the source while the second is the target. This would create an arrow pointing from the source node to the target.

Example usage:

```python
label second_choice:
	menu:
	"Prioritize Environmental Impact":
		$ uuid2 = add_element("Environmental Impact", "Intermediate", 450, 490)
	"Prioritize Worker Wages":
		$ uuid2 = add_element("Worker Wages", "Intermediate", 440, 80)

	"This decision affects your initial choice."

	$ add_dependency(uuid1, uuid2)

	jump third_choice
```

3. Causal Graph Visualization Framework

The graph below shows how the user interacts with the CDD tool system. It incorporates detailed design such as parsing schemas and processing user actions. This design emphasizes efficiency and clarity, ensuring that user inputs are quickly processed and reflected in the interface to support an intuitive and responsive experience.

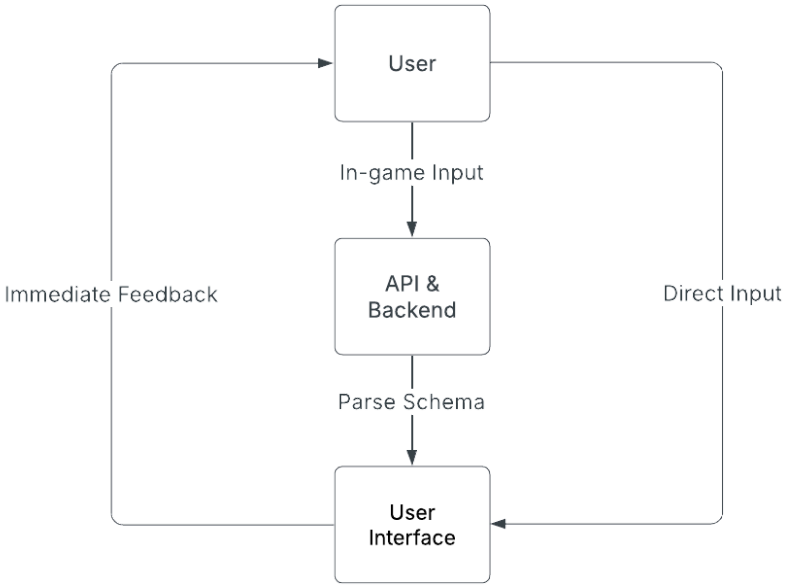

Figure 4.1: Framework Diagram

3.1 Overview

The causal graph visualization utilizes Cytoscape for nodes, and JavaScript / HTML / CSS as a vanilla frontend web development stack. We begin with the user, who creates in-game input. This input prompts an API to add to the schema. That schema is then constructed into the user interface, providing immediate feedback to the user via direct user input.

3.2 Reading the Schema into Cytoscape

The JSON schema that is exported from Ren'Py is then imported into Cytoscape. We use JavaScript to parse the schema and construct nodes based on the content in that JSON file.

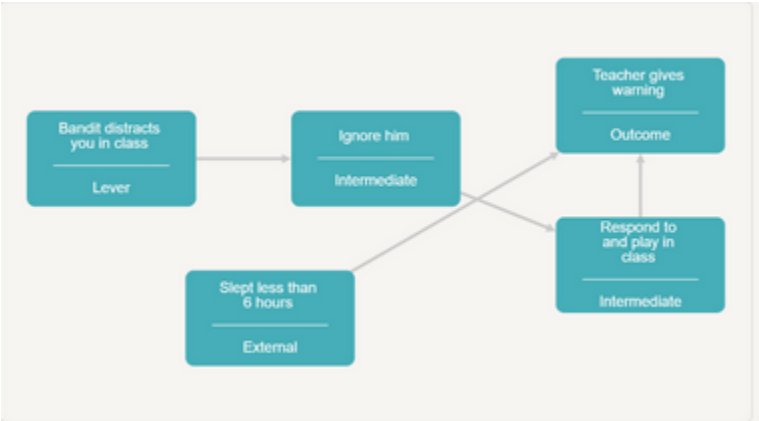

Figure 4.2: JSON imported into Cytoscape

Using figure 4.2 as reference, you can see how the data had been imported. Values such as name and causal type are displayed to the user after being read from the schema.

4. Ren'Py In-game Design

4.1 Scenario Design

This design shows specifically how the visual novel brings the player into different scenarios. The design deploys a “point system” which evaluates the player’s decisions and takes them to different scenarios.

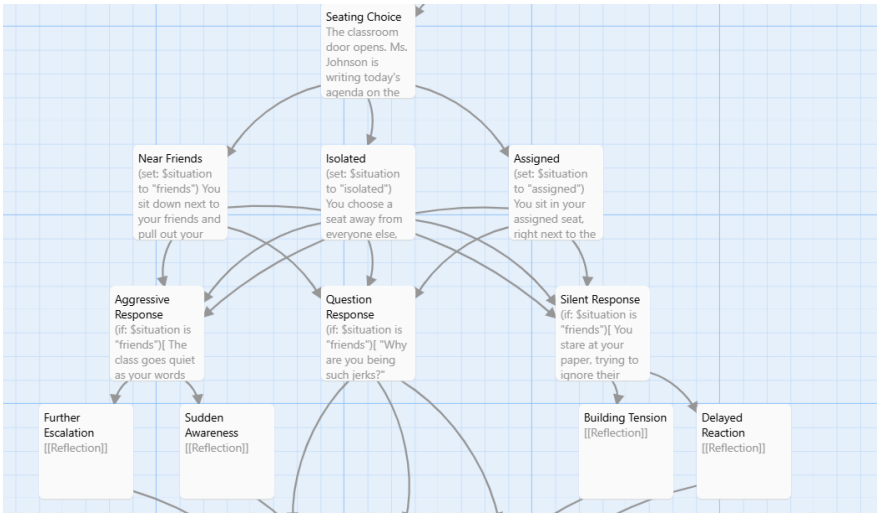

Figure 4.1: Design on how to divide into scenarios

4.2 Character Design

The characters are designed as animals to encourage player empathy and engagement. Figure 4.2 shows the original design of some characters, as well as the background setting, which we decided to use existing sources online later in our implementation phase.

Figure 4.2: Design of characters and other assets

4.3 Paper Prototype

Our sponsors required the team to paper prototype our design ideas to communicate design ideas and make adjustments at the early process. Figure 4.3 showcases examples of paper prototyping.

Figure 4.3: Paper prototypes

5. Research Pathways
    a. AI Integration
        - We initially explored the possibility of using AI to generate dynamic story content for our visual novel. While this approach offered the

potential for highly personalized narratives, our research revealed several limitations:

- Manual story implementation provided significantly more control over the educational content and therapeutic messaging
- Maintaining narrative coherence proved challenging with AI-generated content
- Ensuring age-appropriate language and situations for our 8-15 year old target audience required extensive oversight
- The therapeutic objectives for students with EBD demanded precise narrative design that AI generation couldn't reliably deliver

b. Flask Backend Integration (research pathway, superseded)

- We investigated implementing a Flask backend to handle dynamic content generation and data processing:
- Manual story implementation ultimately gave us more control over the narrative flow
- We encountered significant integration challenges:
- The latency introduced by sending requests to the backend disrupted the story flow
- Real-time dialogue generation created inconsistent user experiences
- Precomputing dialogue options reduced system versatility and limited adaptation to user choices
- The additional complexity didn't justify the marginal benefits for our specific use case

GUI Design

Wireframe sketches for the CDD tool. This design envisions the user to engage with the CDD tool on the right, and on the left side, an animated character would explain each decision and nudge the user to think about changes in their action.

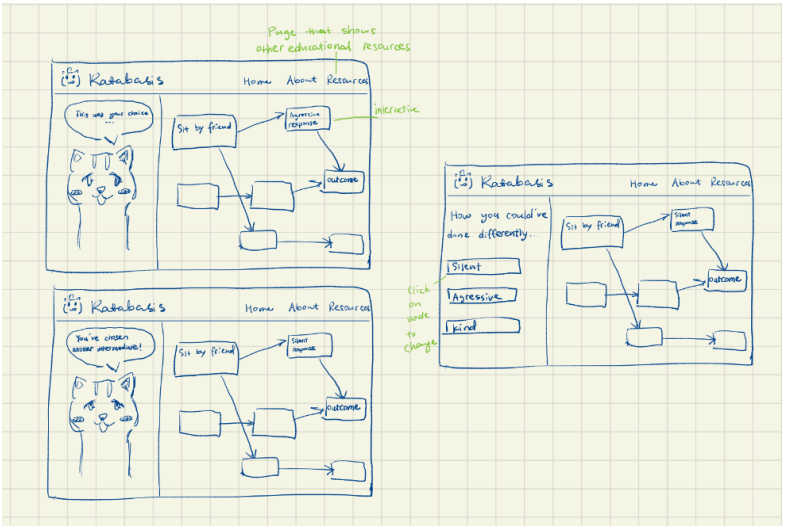

Figure 5.1: GUI of the CDD tool

The design for the CDD tool's landing page. This design is later disregarded since it greatly broadened the scope to an unmanageable degree.

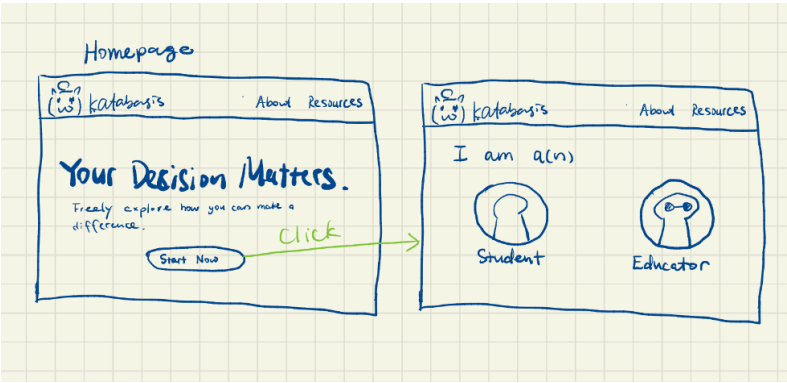

Figure 5.2: Envisioned landing page of the CDD tool

Screenshots of our design from the web

Figure 5.3: Landing screen of Ren'Py game

Figure 5.4: In-game screenshot of player talking to a character

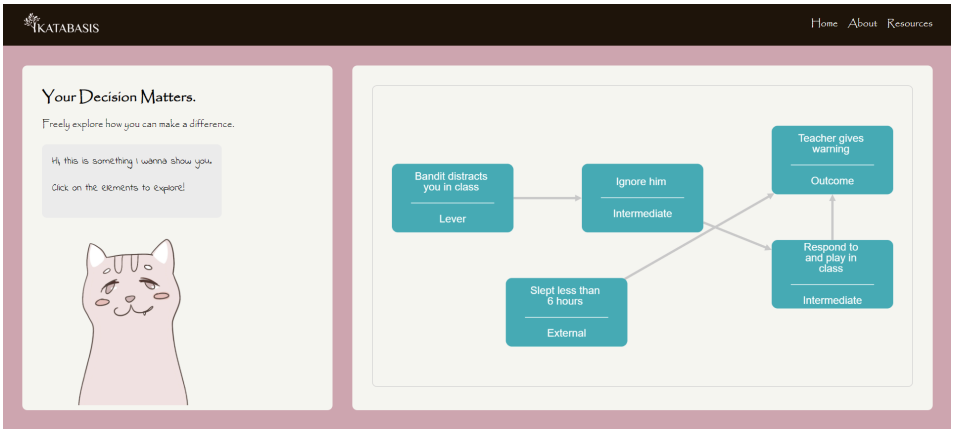

Figure 5.5: Implemented CDD tool

Implementation

Author(s): Joey Zhou
Reviewer(s)/Editor(s): Katie Hammer, Laura Yang

Iteration 0: Research, Planning & Initial Setup

- Dates: January 15, 2025 – February 1, 2025
- Features Implemented:
    - Conducted extensive research on appropriate tools and technologies for our target audience
    - Performed paper prototyping of key interactions to validate design concepts
    - Researched characteristics and needs of students with EBD to ensure appropriate content design
    - Collaborated with sponsor to align project goals with therapeutic objectives
    - Defined main characters (Mousy, Friend, Ms. Johnson) along with their corresponding image assets
    - Set up global variables for background weather and color
    - Configured images and transition effects (e.g., ImageDissolve for scene transitions)
    - Established the basic interactive narrative flow, including the input mechanism for the player's name

Iteration 1: CDD Schema & Narrative Framework

- Dates: February 1, 2025 – February 20, 2025
- Features Implemented:
    - Developed the schema for CDD integration
    - Created scenario outlines for all five classroom incident types
    - Added multiple branching dialogues based on player choices:
        - Decision on whether the player remembers Mousy ("Yes, of course" or "Not really")
        - Sleep duration selection with corresponding changes in Mousy's expression (happy or sad)
        - Mood selection that alters Mousy's emotional state and dialogue responses
        - Additional class excitement level choices, reusing the mood variable for emotional feedback
    - Implemented the locker and sticky note scene, where the locker is opened and a sticky note with a teasing message ("you smell like a dog!") appears, accompanied by corresponding changes in Mousy's expression
    - Addressed implementation challenges related to maintaining narrative coherence across branching paths

Iteration 2: Enhanced Interaction & Added Effects

- Dates: February 20, 2025 – March 5, 2025
- Ren'Py Features Implemented:
    - Mini-game Integration: A tic-tac-toe mini-game has been introduced within the disruptive behavior scenario, offering branching outcomes based on the game's result

    - Full Audio Integration: Added audio assets for all characters, ensuring that each role has dedicated sound clips to enhance character immersion
    - Vibration Effects: Implemented vibration effects for key narrative moments (e.g., during the disruptive scene when Bandit playfully punches the player)
    - Disruptive Behavior Scenario: Developed a new scene focusing on classroom disruption with dynamic adaptation based on player choices, mood, and prior interactions
    - User Role Audio/Text Sound Selection: Added option to choose between audio-based or text-based sound experience for character interactions
    - Auto Highlight for Characters: Integrated auto-highlight functionality that emphasizes the speaking character's image during dialogue
- CDD Visualization Tool Implemented:
    - Frontend using Javascript, CSS, HTML, and Cytoscape.
    - OpenDI-compatible JSON file generator using Python

Iteration 3: Scenario Completion & Project Finalization

- Dates: March 5, 2025 – April 25, 2025
- Features Implemented:
    - Completed development of the remaining four classroom incident scenarios:
        - Scenario 2: Work refusal
        - Scenario 3: Emotional outburst
        - Scenario 4: Disrespectful language
        - Scenario 5: Peer conflict
    - Refined CDD generation to accurately reflect complex decision pathways
    - Conducted user testing with target demographic and implemented feedback
    - Prepared final documentation for project handoff to sponsor
    - Created comprehensive poster presentation for project showcase
    - Compiled final project report documenting all aspects of the development process

Security Considerations

Author(s): Katie Hammer, Joey Zhou
Reviewer(s)/Editor(s):

Confidentiality: Our application will collect and store sensitive information about students with EBDs, including their emotional states, behavioral patterns, and decision-making process. We need to export this data to be able to use it in our CDD authoring tool. This data requires protection from unauthorized access. While our current prototype doesn't implement full encryption, it would be a good measure for the future.

Integrity: The integrity of scenario paths and decision diagrams is crucial for providing consistent educational outcomes.

Availability: For students with EBDs, consistent access to supportive tools is essential for development. Our application is designed to be hosted to the web, so no prior installation of Ren'Py is required for gameplay.

Identification & Authentication: Our application will implement secure user identification and authentication measures to ensure that only authorized users can access sensitive information. This includes a secure login process with strong password policies. User sessions will be monitored and timed out after periods of inactivity to further protect against unauthorized access.

Accountability: To maintain accountability, our system will log all user actions, particularly any data access or modifications related to students' sensitive information. These logs will be securely stored and monitored for suspicious activity, allowing administrators to conduct audits and track any unauthorized attempts to access or alter decision scenarios.

Privacy: Privacy is of utmost importance given the sensitive nature of the data collected (e.g., emotional states, behavioral patterns, and decision-making processes). Data will be handled according to applicable privacy laws and best practices, ensuring that only the minimum necessary information is stored and that all access is strictly controlled. User consent will be obtained where required, and access to data will be role-based to minimize unnecessary exposure.

Future Security Roadmap:

Although our current prototype does not implement full encryption, we are committed to enhancing our security measures. The roadmap includes:

- Encryption: Implementing full encryption for data both at rest and in transit, ensuring that sensitive information is protected from unauthorized access.
- Enhanced Authentication: Introducing multi-factor authentication (MFA) and advanced password policies.

Project Folder Structure

Ren'Py:

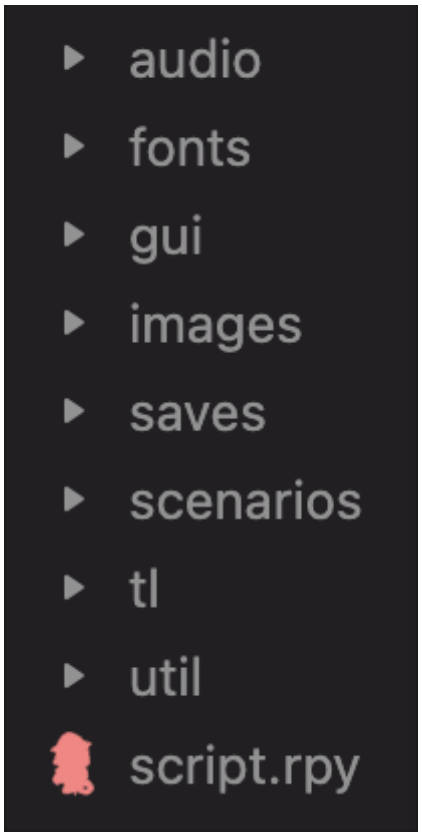

Figure 6: Ren'Py directories

1. **Subdirectories**:
    - **audio**: Stores audio files or scripts related to audio processing.
    - **game**: This folder contains game-related assets or scripts.
    - **fonts**: Stores font files used in the project.
    - **gui**: Contains files related to the graphical user interface, such as gui.ryy.
    - **images**: Stores image assets used in the project.
    - **saves**: Contains save files or data related to game progress.
    - **scenarios:** Contains the scripts for the main scenes of our game.
        - intro_scene.py: manages the introductory scene of the game or application.
        - locker_scene.py: handle a specific scene related to a locker.
        - disruptive_scene.rpy: the disruptive behavior scenario
        - elopement_scenario.rpy: elopement (skipping class) scenario
        - emotional_outburst_scene.rpy: emotional outburst scenario
    - **util**: This folder contains various Python scripts, possibly for handling different scenes or functionalities:
        - 00auto-highlight.py: It is related to automatic text highlighting.

        - add_voice.py: It handles adding voice overs or audio to dialogues.
        - dialogue_effects.py: manages effects applied to dialogues.
        - disruptive_scene.py: handle a specific scene that disrupts the normal flow.
        - generate_audio.py: responsible for generating audio files.
        - options.py: manage user options or settings.
        - screens.py: handles different screens or views in the application.
        - script.py: the main script for the project.
        - tictactoe.py: Is a mini-game or a specific feature.
        - tts_settings.py: contains settings specific to the project.
        - **minigames: folder with minigame scripts and assets**
            - **images: folder with game assets**
            - characterracing.rpy: race minigame with Crusher, Bandit & Mousy
            - memorycard.rpy: mini game with cards flipped upside down, purpose is to reveal 2 cards at a time and match them
            - reactiongame.rpy: whack a mole
    - **venv**: This is a Python virtual environment, containing dependencies and configurations specific to the project.
2. **Error Files**:
    - errors.txt: logs errors encountered during the execution of the project.
    - log.txt: a general log file for tracking events.
    - traceback.txt: contains detailed traceback information for debugging purposes.

Project Configuration/Settings

The project is configured using a combination of hardcoded constants within the Ren'Py script and external configuration files that allow for adjustments when deploying in different environments. Key configuration details include:

- **File Paths & Assets:**
  The script references various asset files such as images (e.g., "images/classroom.png", "images/bg lockers.png") and audio files (e.g., "audio/_Wake up The day jus.MP3"). These paths are defined directly in the script, and if the project is moved or deployed on another system, updating these paths in a central configuration file or module can simplify the process.

- **Global Variables & Constants:**
  Variables such as `bg_weather` and `bg_color` are set as defaults within the script. Other constants, like transition effects (e.g., `ImageDissolve` parameters), are similarly embedded. Making these configurable through an external settings file would allow adjustments without needing to modify the main script.

- **Environment-Specific Settings:**
  In a deployment scenario, you might need to adjust parameters such as:

    - **Asset Paths:** Adjust file paths if the directory structure changes.
    - **Audio/Video Settings:** Volume levels or playback configurations.

    - **Debugging and Logging:** Options to enable verbose logging during development, which can be disabled in production.
    - **Mini-Game Settings:** Parameters for game mechanics (e.g., tic-tac-toe win conditions, time limits) can be made configurable to tweak gameplay.
- **Example Configuration File:**
  Below is an example of a JSON configuration file (`config.json`) that could be used to manage these parameters:

```json
"asset_paths": {
    "images": {
        "classroom": "images/classroom.png",
        "bg_classroom": "images/bg classroom.png",
        "lockers": "images/lockers.png",
        "lockers_blurred": "images/lockersblurred.png",
        "sticky_note": "images/sticky_note.png"
    },
    "audio": {
        "wake_up": "audio/_Wake up The day jus.MP3",
        "mousy_greeting": "audio/_Hiya it s me Mousy .MP3",
        "vibration": "audio/vibration.mp3"
    }
},
"global_settings": {
    "bg_weather": "Sunny",
    "bg_color": "Green",
    "transition_effect": {
        "in_duration": 3.0,
        "out_duration": 3.0
    }
},
```

Figure 7: JSON configuration file

- **Usage in the Project:**

  The script can be modified to load and parse this configuration file at startup, dynamically setting the asset paths, global variables, and gameplay parameters accordingly. This makes the project more adaptable to different deployment environments or user preferences.

In summary, by centralizing configuration in an external file like the JSON example above, the project becomes easier to maintain and deploy, with environment-specific settings such as asset paths, gameplay constants, and debugging options clearly defined and modifiable without altering the core codebase.

Testing

Author(s): Nomar Ledesma Gonzalez
Reviewer(s)/Editor(s): Joey Zhou

Overall View

Our game system will require testing for both the backend logic verification, certain API/data validations, and human centered around manual tests and usability testing to ensure the gameplay experience meets our sponsor's requirements (i.e., engagement, aligned with EBD needs)

We will conduct Acceptance Testing to verify implemented features meet the requirements outlined in our system design, including gamified interfaces, CDD generation and cross-browser compatibility.

In addition to Acceptance Testing, we will also conduct Technical Testing to ensure our systems performance, a proper integration between our game and OpenDI, and correctness of logic such as CDD generation, branching narrative flow and variable persistence.

Our game runs on the Ren'Py engine and is integrated with OpenDI. For web-based access and demonstrations we host the application on an NCSU VM and use HTTP tunneling to allow external connections. We perform tests on our local machines and plan to target common web browsers when portions of the game (such as causal decision diagram visual) go web-based. All test configurations will be documented in the project's README to allow testers and developers to replicate the environment.

Acceptance Testing

Functionality

Our acceptance testing on functionality will verify that functional requirements (e.g., "the system must present a gamified interface" & "students can see their CDD" and non-functional requirements (e.g., "diverse browser compatibility") are satisfied.

Test ID	Feature	Scenario	Preconditions	Test Steps	Expected Results	Actual Results
FR2-AT1	Player Rewards	Player chooses "positive" options repeatedly	Game scenario loaded with multiple decision points; position options available	1. Consecutively choose positive responses 2. Check visual indicator for live point-tracking	Points or badges increment for each positive choice	Points or badges increment for each positive choice
FR9-AT2	Decision Selection	Player makes choices in branching dialogue. Visits other environments in school.	Player starts with an elopement scenario. "Skip class" or go to class options available.	1. Player demonstrates elopement behavior and chooses option to leave class 2. Observe if story transitions accordingly	Game transitions to the next appropriate scene	Game transitions to the next appropriate scene

FR1 - AT3	Story Display	Player navigates from the main menu to the first scenario.	Game is launched. On the home screen, with the Start button.	1. Launching the game in Ren'Py or in web browser 2. Clicking "Start Game". 3. Observe intro dialogue and first scenario	Intro text and scenario are displayed; no errors or UI glitches	Intro text and scenario are displayed; no errors or UI glitches
FR4 - AT4	CDD Generation	Player completes one scenario to trigger CDD generation.	Play a route through the game.	1. Finish scenario route 2. Access OpenDI framework	System generates a diagram reflecting actions, intermediate factors, and outcomes. An updated UI with a game character will explain the CDD to show students how decisions link.	System generates a diagram reflecting actions, intermediate factors, and outcomes. An updated UI with a game character will explain the CDD to show students how decisions link.
FR-6 AT5	Editable Decisions	Player revisits their previous decision.	Scenario complete. Visiting CDD site.	1. Navigate to edit option 2. Modify a prior decision	Scenario adjusts, updated points are reflected, CDD is regenerated accordingly	Scenario adjusts, updated points are reflected, CDD is regenerated accordingly
FR3 - AT6	Browser Compatibility	Run the game or playable build on Chrome, Firefox, Safari, etc.	Game build is hosted.	1. Open the game or relevant link in multiple browsers. (Important to test variety of hardware specifications) 2. Test main gameplay	Game is functional and consistent across tested browsers	Game is functional and consistent across tested browsers

				loop	(no layout breaks, minimal to no performance issues)	(no layout breaks, minimal to no performance issues)

Acceptance Test Status

- AT1 is yet to be implemented. Our team is currently focused on scenario development and CDD updates.
- We have completed initial tests (AT2, AT3) with positive results–scenarios, UI, dialogue are all displayed with no errors.
- Software for CDD generation and editing (AT4, AT5) is under development and testing will commence when implementation is completed.
- Browser compatibility tests (AT6) are planned once we finalize the build for web deployment.

Scenarios

Our acceptance testing on scenarios will verify that variable branching is working expectedly and all scenarios are accessible to the player. All preconditions for these tests will be "start a new game."

Test ID	Menu Choices	Expected Result	Actual Result
S1 **Conflict classroom scene**	1. About 6-8 hours 2. Excited! 3. Class is so boring! I always get called on for no reason! 4. Throw 5. Who did that? 6. You have to! Tell the teacher, that is not okay. 7. I wouldn't let it go! I'd have to say something back! 8. Yeah, see you later!	Scenario starts with "You sit quietly in the classroom, focusing on your textbook and notes."	Scenario starts with "You sit quietly in the classroom, focusing on your textbook and notes."

S2 **Emotional Outburst scene**	1. Less than 6 hours 2. I'm already so over it 3. Class is so boring! I always get called on for no reason! 4. Throw 5. Why would someone do that? 6. You have to! Tell the teacher, that is not okay. 7. Depends on my mood 8. Yeah, see you later!	Scenario starts with "After the incident at the lockers, I make my way to Ms. Johnson's math class."	Scenario starts with "After the incident at the lockers, I make my way to Ms. Johnson's math class."
S3 **Disruptive scene**	1. About 6-8 hours 2. I'm already so over it 3. Class is so boring! I always get called on for no reason! 4. Ignore 5. Who did that? 6. No way! That'll make it worse, better to handle it yourself 7. I think I would just deal with it 8. I'm gonna take my time!	Scenario starts with "As I walk through the empty hallway, the sound of my footsteps echoes against the lockers. The conversation with Mousy still lingers in my mind."	Scenario starts with "As I walk through the empty hallway, the sound of my footsteps echoes against the lockers. The conversation with Mousy still lingers in my mind."
S4 **Elopement scene**	1. More than 8 hours 2. I'd rather be anywhere else 3. I'm thinking of	Scenario starts with "I walk past the classroom I'm supposed to be in. Ms. Johnson's voice fades as I continue	Scenario starts with "I walk past the classroom I'm supposed to be in. Ms. Johnson's voice fades as I continue

	just not going 4. Ignore 5. Who did that? 6. You have to! Tell the teacher, that is not okay. 7. I'd try to avoid them entirely 8. I might just skip this one	down the hallway."	down the hallway."

Usability Testing

Methodology (Test Script)

We conducted structured usability testing sessions with friends and family and made notes of the responses.

- Introduction and warm-up (collecting demographic information and gaming background)
- Observed gameplay with a think-aloud process where participants verbalized their thoughts while interacting with the system
- Post-play structured interview to gather detailed feedback on specific features and overall experience

Key Findings

The usability testing revealed several important insights:

- Controls were intuitive for most participants, with minimal instruction needed to navigate the visual novel interface
- Character designs received positive feedback, with participants noting they felt relatable and age-appropriate
- Users Requested:
    - Clearer guidance on text input length when providing personal responses
    - More character background/introduction at the beginning of scenarios
    - Improved scene transitions for better narrative flow
    - Additional visual cues to indicate interactive elements
    - More explicit feedback when their choices affected story outcomes

These findings informed several adjustments to our implementation during Iterations 3 and 4, particularly around improving character introductions and providing clearer visual indicators for interactive elements.

A copy of our testing script is included in the appendix.

Technical Testing

Testing Approach

In parallel with usability testing, we implemented a comprehensive technical testing strategy focusing on:

- CDD generation verification: Ensuring that user choices correctly map to appropriate causal decision diagram elements

- Branch logic testing: Validating that all narrative branches function as intended without dead ends or logical inconsistencies
- Story flow validation: Confirming that variables persist correctly throughout the narrative and affect subsequent scenes appropriately
- Technical performance validation: Testing loading times, memory usage, and overall system stability across different devices

Current Status

Our technical testing efforts are ongoing, with several key components:

- Developing CDD output validation tools to verify the correctness of generated decision diagrams
- Developed comprehensive narrative flow test cases that exercise all possible decision paths
- Testing custom input validation to ensure robust handling of user-provided text
- Implementing automated regression testing to maintain stability throughout development iterations

The integration between the Ren'Py visual novel system and the OpenDI framework has required particularly thorough testing to ensure data flows correctly between these components. Throughout our testing process, we've maintained close communication with our project sponsor to ensure that the application not only functions correctly but also effectively serves its therapeutic and educational purpose for students with EBD.

Task Plan

Author(s): Katie Hammer
Reviewer(s)/Editor(s): Laura Yang

Task	Owner(s)	Due Date	Status
Iteration 0			
Requirements Gathering & Research	All	2/1	done
Paper prototype testing with target audience representatives	All	2/1	done
Define core scenarios based on common classroom incidents	Katie & Nomar	2/1	done
Ren'Py basic scenario setup	Laura & Joey	2/1	done
VM & Ren'Py backend setup	Ryan	2/1	done
Iteration 1			
Ren'Py branching scenario & polish effects	Joey & Nomar	2/20	done
Variable system setup for player customization	Katie	2/20	done
Develop the schema for CDD integration	Laura & Ryan	2/20	done
Iteration 2			
Polish game details (Mini-game, audio, effects)	Joey & Nomar	3/5	done
CDD visualization	Laura	2/20	done
Generate OpenDI-compatible JSON	Ryan	2/20	done
Branching of scenarios & Variable system	Katie	2/20	done
Iteration 3			
Create a new scenario	Katie	3/15	done
Create a new scenario	Nomar	3/15	done
Implement scenario	Joey	3/15	done
CDD visualization	Laura	3/15	done

OpenDI - JSON	Ryan	3/15	done
Iteration 4			
Export the game as the schema	Ryan	4/22	done
Variable branching	Katie	4/22	done
Create & implement a new scenario	Joey & Nomar	4/22	done
Game review & integrating	Laura	4/22	done

Team Contact Information

Name	**Main Roles**	**Email**
Nomar Ledesma Gonzalez	Front-End Developer, Testing Leader	nledesm@ncsu.edu
Katie Hammer	Narrative Developer, Quality Assurance	kahammer@ncsu.edu
Ryan Mikula	Full Stack Developer, Integration Specialist	rmmikula@ncsu.edu
Laura Yang	Asset Developer, Authoring Tool Supervisor	lnyang2@ncsu.edu
Joey Zhou	Lead Code Developer, Testing	yzhou39@ncsu.edu

Suggestions for Future Teams

Author(s): Laura Yang
Reviewer(s)/Editor(s):

Project Management

To take over our project, first define the scope of the project at the early stage of designing and create task plans and deadlines according to your chosen development cycle. We refactored Renpy code for increased readability and commented on the purposes of methods, so we suggest future teams to read our comments and notice the naming conventions as well as the organization of files.

AI Integration

With the purposes of more natural interaction between the game and the players, we experimentally implemented a flask backend on our github branch “flask_ai_example” so that AI would generate conversations and help the player to think about decision-making. We suggest future teams to build upon our existing flask backend and AI demo.

UI

Our initial design of the UI is to have the character Mousy on the left side, and as the player interacts with the nodes, Mousy would talk to the player about their new decision and what impact that could have on the main character. This functionality is yet to be implemented, and to pick up our work, we suggest future teams to pinpoint the export of JSON schema - an efficient way of parsing the schema with some additional library usage could bring this blueprint into reality.

Appendix

Usability Testing Script

Katabasis
USABILITY TESTING SCRIPT

INTRODUCTION:

- Greet the playtester. Introduce yourself
- Explain Katabasis project game:
    - We’re working with Katabasis to build a visual interactive novel game designed for students with emotional and behavioral disorders, aged 8-15. The students will ultimately be playing a game of self-identification and reflection. This game is set in a school environment where the player will navigate through scenarios interacting with classmates, teachers, and objects to create actions that will ultimately lead to a final outcome. The final outcome of this game will be represented by a diagram which will explain to the students each of the significant details which led them to their outcome.

WARM-UP DISCUSSION:

- General Demographic Questionnaire:
    - How old are you?
    - What is the highest level of schooling you’ve completed?
- Gaming Questionnaire:
    - Do you play video games?
        - What type of games do you enjoy the most?

        - Which is your favorite game and why?
        - Have you ever played an interactive novel/branching narrative game?
- Think-Aloud Practice
    - Exercise to practice thinking aloud. Solve 257 * 83 on paper.

PLAY SESSION:

- Explain the purpose of playtesting:
    - The purpose of this playtest is to test the game and not the playtester. There are no wrong choices or comments and we would like to see how you naturally interact with the game.
- Explain to the user that—if anything is confusing or frustrating, please let us know.
- What the playtester will do:
    - As the playtester, you will play the game for about 15 minutes while thinking aloud while playing. Please let us know how you are feeling, as well as the rationale behind the choices you make.
- While you play, I will look out for:
    - How you interact with the game.
    - Visual and audible emotions elicited.
    - Confusions and frustrations with the game.

- (IF NEEDED) Explain the basic controls of Ren'Py
    - LMB(Left Mouse Button)/SPACE advances the scene or makes a choice
    - RMB(Right Mouse Button) enters the game menu. (This can be accidentally prompted, often by a misclick.)
    - BACK button undoes the previous action.
    - It is generally not encouraged unless the playtester is an experienced gamer, they are free to change text options of game client (i.e., text speed, bindings)
- While the playtester is playing, encourage them to verbalize their thoughts.
    - Why did you choose that option?
    - What do you think will happen next?
    - What are you trying to do right now?
    - Was that what you expected?
    - What is confusing or frustrating you?
- If the playtester is stuck, observe why they are stuck & assist with their confusion (i.e., unclear instructions, difficulty understanding choices, etc.).
    - Look out for and make note of:
        - Difficulty & frustrations
        - Engagement level
        - Emotional reactions
        - Unintended player behavior
        - Game controls
        - Narrative Clarity
        - Parts of the game users really like
        - Times when users needed help to continue (and why)

DISCUSSION OF PLAY EXPERIENCE:

- Questionnaire:
    - Overall Experience:
        - What did you think of the game overall?
        - Did you enjoy playing it? Why or why not?
        - Was it easy or hard to learn how to play the game?
    - Gameplay & Mechanics
        - How would you explain this game to someone who's never played it before?
        - What was your favorite part of the game?
        - What was the most difficult or frustrating part?
        - Was anything confusing?
    - Story & Choices
        - What do you feel like would've happened if you choose (x or y/specific choice in their story)?
        - What was the scenario you experienced and does it feel realistic?
        - How do you make decisions? (i.e., "how i would in real life", "embodied a new character")
    - Feedback & Improvements
        - What would you change about the game to make it better?
        - Would you want to play a longer version of this game?

WRAP-UP:

- Thank the playtester for their time and thoughts.
- Ask if there are any final thoughts before the conclusion of usability testing.
- Let them know their feedback is very helpful and will help us make the game better