App: Study Dungeon **Group**: Green Alligators

Team: Aidan Dougherty, Murray Cadzow, Veronika Luthar

App function

The application we are building is called Study Dungeon. It is a command-line-based study game where players create flashcard decks and revise them in order to create their hand. Cards in the hand will then be assigned abilities that can be used to defeat various enemy characters for rewards. The player will be provided with a Text User Interface to navigate the game and game graphics in the form of ASCII art. The rewards will be various ASCII art items that they can use to decorate their in-game study room, a place where their player character can stand idly while the player studies out-of-game.

Theme

Study Dungeon is “back to basics” because flashcards are a fundamental and popular study technique taught to students because of its simplicity and effectiveness. Further, many classic video games originally had ASCII graphics that were displayed in a command line interface. Study Dungeon also focuses on a simple gameplay mechanic, rather than the latest and greatest graphics and story.

Development

Study Dungeon will be programmed in the C++ language. Git will be our version control system, with GitHub as the central repository (https://github.com/douai724/StudyDungeon) and GitHub Actions for our CI/CD pipeline. We will be using a unit testing framework and GitHub Actions to automate build and unit testing to ensure the app’s quality. Furthermore, we will use Doxygen for documentation. Supporting asynchronous work, we have set up a Discord server for our main line of communication, GitHub Projects for project management, and a shared Google Drive folder for additional materials.

Existing solutions & Customer interest

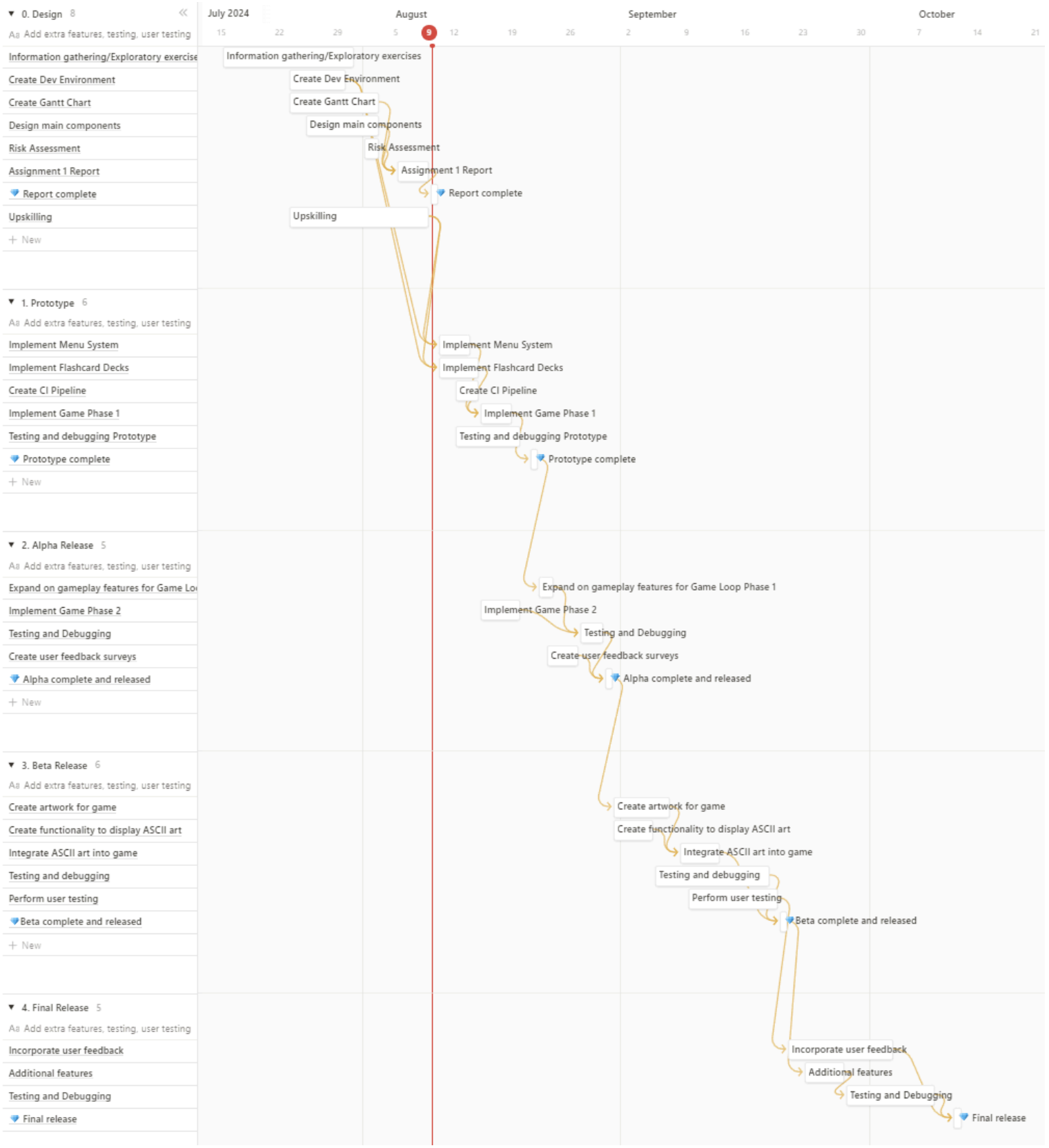
Popular digital flashcard platforms like Anki, Quizlet, Brainscape, and Memrise demonstrate a clear market demand for interactive study tools. Study Dungeon differentiates itself by combining flashcards with a deck builder game format, tapping into the growing trend of gamification in education. The success of gamified learning apps like Duolingo proves strong customer interest in educational tools that make studying engaging. By offering in-game rewards for study efforts, Study Dungeon addresses the common pain point of maintaining long-term motivation in self-directed learning and its unique command-line interface and ASCII art aesthetics also appeal to tech-savvy students and retro game enthusiasts, potentially creating a dedicated niche user base.

Project planning

Risks

	Probability	Recovery Cost	Plan for risk reduction
Team member illness	High	Medium	Reduce impact by reassigning tasks. Pair programming and regular verbal updates.
Incompatible development environments	Low	Medium	Create a centralised dev environment configuration. Devise group workflows/operating procedures. Shared configuration files. CI/CD pipeline for quality assurance
Team member turnover	Low	High	Reduce impact by ensuring all team members have a good understanding of the project, and have an equal say in its direction. Knowledge/skills sharing within the group.
Insufficient experience with programming language	High	High	Use available resources to learn and understand language.
Incomplete knowledge and inexperience may lead to incompatible design choices	High	High	Be mindful of researching limitations as much as possible before implementation
Project size underestimation	High	Medium	Allocate more time than estimated for each task. Prioritise features to be added, with must-haves completed before optional features.
Project specifications change	Medium	Medium	Ensure that tasks are frozen for set sprints.
Reliance on 3rd parties for CI\CD	Low	Low	Only use libraries built into Windows. Monitor changes to github actions. CI/CD should mirror local workflows
Features of C++ v20 and v23 haven't been finalized so there is a risk of using a language feature that doesn't get included in the final standard.	Low	Low	New language features are generally advanced and unlikely to be utilized given our knowledge

Schedule


0. Design 8
Add extra features, testing, user testing
Information gathering/Exploratory exercise
Create Dev Environment
Create Gantt Chart
Design main components
Risk Assessment
Assignment 1 Report
Report complete
Upskilling
New
1. Prototype 6
Add extra features, testing, user testing
Implement Menu System
Implement Flashcard Decks
Create CI Pipeline
Implement Game Phase 1
Testing and debugging Prototype
Prototype complete
New
2. Alpha Release 5
Add extra features, testing, user testing
Expand on gameplay features for Game Lo
Implement Game Phase 2
Testing and Debugging
Create user feedback surveys
Alpha complete and released
New
3. Beta Release 6
Add extra features, testing, user testing
Create artwork for game
Create functionality to display ASCII art
Integrate ASCII art into game
Testing and debugging
Perform user testing
Beta complete and released
New
4. Final Release 5
Add extra features, testing, user testing
Incorporate user feedback
Additional features
Testing and Debugging
Final release
July 2024
August
September
October
15
22
29
5
9
12
19
26
2
9
16
23
30
7
14
21
Information gathering/Exploratory exercises
Create Dev Environment
Create Gantt Chart
Design main components
Risk Assessment
Assignment 1 Report
Report complete
Upskilling
Implement Menu System
Implement Flashcard Decks
Create CI Pipeline
Implement Game Phase 1
Testing and debugging Prototype
Prototype complete
Expand on gameplay features for Game Loop Phase 1
Implement Game Phase 2
Testing and Debugging
Create user feedback surveys
Alpha complete and released
Create artwork for game
Create functionality to display ASCII art
Integrate ASCII art into game
Testing and debugging
Perform user testing
Beta complete and released
Incorporate user feedback
Additional features
Testing and Debugging
Final release

Requirements

We have elicited the main requirements for the application, and prioritised them using the MoSCoW method (**Must** have, **Should** have, **Could** have, **Won't** have).

Functional

General

- The system will **display an ascii art splash screen** (Could)
- The system will have a **menu** with options for Configuration and Gameplay (Must)
- The system will have a screen to display ASCII art rewards from game wins. (Should)

Card/Deck Management

    - Users will be able to **create a flashcard deck** by either reading from files, or through a deck creator using user input in the program (Must)
    - Users will be able to **edit flashcards** (Could)
    - The system will **save the user inputted flashcards to disk** (Should) ensuring successful file writing, and path/filename restrictions are met
    - Users will be able to **browse their flashcard decks and select the one they want to revise** (Must)
    - The cards in the game will be drawn with **ASCII art** (Could)

Gameplay

- The system will **save/load the program game state to/from disk** and provide game continuation upon load. (Could)
- Users will be able to use a **countdown timer to time game rounds and pomodoro sessions** (Should)
- Users will be able to play the **game to revise flashcards** (Must) This game will consist of three phases:
    - Phase 0 – Where the user will configure the game by selecting a card "deck", assigning stats to the flashcards (Card value, card abilities), setting the number of cards to draw and the number of rounds they wish to play during their session.
    - Phase 1 – System will shuffle the deck and present one card at a time. Where the user picks between whether they got the answer correct (card added to *hand* for Phase 2) or incorrect (Discarded). After the user reviews all cards in the current round, the program moves to the last phase.
    - Phase 2 – System assigns stats/abilities to the *hand*, the user then uses this hand to "battle" an enemy.
- "Rewards Room" of ASCII art with pomodoro countdown timer (Should)
- Play music and sound effects during game play (could)

Non-functional

- Run in the Windows 10 and 11 Command Prompt
- The game graphics will be done in ASCII art
- Single threaded application
- Compiled for x86 or x64_86 using the CL compiler
- Adapt to resolution/window resizing
- Developed using C++latest/C++23

Initial Gameplay Schematic