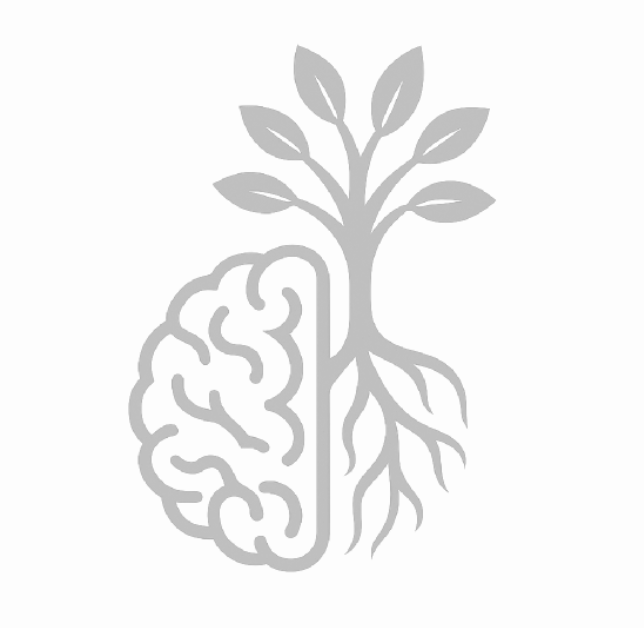
Devine Software Development Plan



ECE 49595 Open Source Software, Team 7

This is the official Software Development Plan (SDP) of team 7 in Purdue University's ECE 49595 Open Source Software senior design course. It describes the scope, priorities, requirements, and deliverables to complete the Devine project.

Authors: *Daniel Chen, Nathan Galinowski, Joshua Kirby, Colin Mcgeary*

Contents

I. Project Overview

Devine is a web-application that provides a suite of brain exercises to users experiencing early-stage symptoms of dementia or users that self-project to experience dementia as they grow older. Exercises provided by the application are designed to target fundamental cognitive functions such as memory, logical reasoning, spatial reasoning, and processing speed. Each exercise will evaluate a user's performance with a set of scores that are recorded after a game is played. A user will be able to observe the scores of a past playthrough and see cumulative statistics that reveal their performance over time for each target category (memory, logical reasoning, spatial reasoning, and speed). Users can set thresholds that act as goals with reminders to encourage and manage progress. Devine will also interface with social media platforms so that users can share their progress with their family, friends, or coworkers. Ultimately, Devine seeks to provide users with an engaging online method to impede dementia symptoms with data driven techniques and a community.

II. Scope

- **Project:** Brain games application for impeding dementia symptoms.
- **In Scope:**
    - A suite of 6+ games focused on memory, spatial reasoning, logical reasoning, and cognitive speed [ID-1].
    - A performance review system so that each user account can see past scores [ID-2].

- A goal-setting system with reminders that allows users to set performance objectives and compare to the performance history to get notified if they are meeting them [ID-3].
- A simple, flexible, and clearly labeled user interface that is responsive to varying device screen layouts and facilitates the user to critical portions of the program [ID-4].
- A web-based application accessible with internet access and a browser [ID-5].
- Data encryption to safely store user personal information and prevent rogue login attempts from malicious actors [ID-6].
- A relational database system to store personal information such as scores, and account information, with access that is private to each user regarding their own data [ID-7].
- Capability of pausing an exercise causing game logic to stop [ID-8].

- **Under Evaluation:**
  - Integration with external social media platforms to allow users to share their scores with their community [ID-9].
  - Method for medical professionals to track patients' performance on games [ID-10].
  - Adaptive difficulty system that personalizes game difficulty based on user performance over time [ID-11].
  - The ability to send data between users in multiplayer games [ID-12].
  - The ability to send emails to remind players to use the application and facilitate password resetting [ID-13].

- **Out of Scope:**
    - Features focusing on physical health, such as nutrition and exercise.
    - A majority of unoriginal games that have already been overdone, such as using memorization to find two matching face-down cards.
    - A translation system in which users can convert the on-screen text to their native language.
- **Assumptions:**
    - Users have not yet reached the mid-to late-stages of dementia, where mental deterioration begins to significantly impact cognitive abilities.
    - Users possess devices capable of connecting to the internet, and consequently the web page where the application will be hosted.
    - Users will be able to read and understand instructions that are written solely in English; no translations are required.
- **Constraints:**
    - Care must be taken to ensure each game is unique and does not infringe on the copyrights of competitors.
    - For regulatory purposes, the necessary preliminary research should be provided as traceable proof of each games' effectiveness in impeding dementia; competitors were fined by the FTC in the past for failing to produce sufficient scientific evidence.
    - For external APIs or cloud computing, the team's budget should not be exceeded.

III. Elicitation & Prioritization

When applying the MoSCoW method to brainstorm features for Devine, we began by listing all the features we thought it should have and our goals for the platform. Then, we slowly debated each one and decided which category to place it into. This helped us identify our must-have requirements for a minimum viable product, which was the most beneficial result of using MoSCoW. One thing we struggled with was distinguishing between a should have feature and a could have feature. Debating the difference between the two categories felt unnecessary for some features at first because we perceived that they could fit into either category. We classified features the way we did because the "could-haves" are extensions of other features whereas the "should-haves" are features that directly support the "must-haves". The key component of the MoSCoW method that was not useful was the "Won't haves" category. Our team had an intuition for the features that we desired, and we could have had a successful brainstorm by having an implied "Won't haves" category defined by what is not listed in the other three. Listing some things that we won't plan to have took more time and did not provide an exhaustive list. Priority determination is meant to help us decide on the order in which we develop features, and "Won't-haves" features will never be developed within the project timeline so they do not need to be prioritized. Overall, the MoSCoW method ultimately allowed us to clarify our priorities and develop a more focused plan for our web application.

IV. Requirements

Requirement: *User Interface [FR-1]*

- **Statement:** The system shall have a user interface with minimal buttons, optimal spacing, responsive interaction, and clear labels.
- **Rationale:** Our users need to be able to look at the screen and be confused at as few moments as possible. The screen behavior must dynamically react to the screen size, without affecting the quality of experience for the users.

- **Test Method:** Manual testing by users and tools like Cypress can be used to automate navigation throughout the application. User feedback can also be collected to make usability improvements.
- **Supporting Context:** Buttons are pressed by the user to perform an action in a game or to navigate through the webpage. Optimal spacing means that the design of the webpage allows for plenty of space to easily distinguish between different buttons. Labels are text that describe what a button does. Responsive interaction means that the webpage will navigate to different tabs and games smoothly and in a short amount of time.
- **Tracing Information:** Easy to use interface [ID-4]
- **Priority:** Must have

Requirement: *Brain Games [FR-2]*

- **Statement:** The system shall have at minimum four exercises that focus on brain development within the scope of logic, memory, or reasoning.
- **Rationale:** As the heart of the system, the exercises provide the material to improve brain functionality. Everything else in the system is designed to supplement the exercises.
- **Test Method:** Manually run each game to ensure that it launches and functions correctly without errors. Make sure that there is design consistency between the game style and the UI style. Automated unit and integration tests will be run for the game logic prior to manual testing.
- **Supporting Context:** Exercises are the brain puzzles or games that will be developed. Logic, memory, and reasoning are the skill areas of the brain which the application seeks to improve.
- **Tracing Information:** Games [ID-1]
- **Priority:** Must have

Requirement: *Multiplayer Data Transfer [FR-3]*

- **Statement:** While the system is in multiplayer mode, in-game data will be sent to other participating clients.
- **Rationale:** Sending data across a LAN is a requirement for updating the multiplayer game state.
- **Test Method:** Begin a multiplayer game with a minimum of two clients and verify that the game states are synchronized, and data updates are transmitted correctly. Ensure that gameplay is consistent for both users. Cypress can be used for web automation.
- **Supporting Context:** Multiplayer mode is a game state where a user is playing with other users of the application and not just themselves in a single player environment. Other clients are other users that use the application. This assumes that a particular user plays with another user in a multiplayer setting.
- **Tracing Information:** Multiplayer capability [ID-12]
- **Priority:** Should have

Requirement: *Pausing Mechanism [FR-4]*

- **Statement:** While the system is in an in-game state and is paused, all game logic shall cease to update and no data will be sent to other end hosts.
- **Rationale:** The pause state is designed to give the user(s) a break from the action. No game logic may progress when this state is reached.
- **Test Method:** Confirm that pausing a game stops all game logic and network transmission. Pausing in a multiplayer game pauses game logic for all users. Check the database to ensure that it has stopped collecting data for any user whose game is paused. This should be ultimately verified with a manual test, but an automated test can be used as well to check if the state updates on the next frame.
- **Supporting Context:** In-game state is the time when the user is actively playing a game. Game logic is the code that allows each game to function properly. End hosts would be the user's personal game history, social media platforms, and their personal doctor. This assumes that the user pauses the current game that they are playing.
- **Tracing Information:** Ability to pause [ID-8]
- **Priority:** Must have

Requirement: *Automatic Difficulty Scaling [FR-5]*

- **Statement:** When a particular game becomes too easy for the user, the system shall adjust the difficulty of the game to make it more challenging for the user.
- **Rationale:** In order to promote improvement and growth, difficulty scaling is an important aspect to make sure users don't become complacent in any given game.
- **Test Method:** Play through games, performing really well to ensure that the difficulty gets more difficult. Likewise, perform poorly to ensure that the algorithm makes the games easier. Automated test scripts can be written to simulate skill levels/difficulty levels, and to check if difficulty parameters are changing.
- **Supporting Context:** Games being the brain puzzles the user plays in the application. This assumes that games become too easy or difficult for the user.
- **Tracing Information:** Difficulty scaling [ID-11]
- **Priority:** Should have

Requirement: *Play Reminder Emails [FR-6]*

- **Statement:** If the user has yet to login to the system in several days, then the system shall send the user an email to remind them to login and play.
- **Rationale:** It is important for users to be reminded to play given that they have notifications turned on.
- **Test Method:** Verify that the application sends reminders to user emails after a period of inactivity. Automated test scripts can simulate user inactivity. Check user emails to ensure such emails are delivered and received.
- **Supporting Context:** This assumes that the user has not engaged with the application in several days.
- **Tracing Information:** email reminders [ID-13], goal reminders [ID-3]
- **Priority:** Should have

Requirement: *Postgame Scoreboard [FR-7]*

- **Statement:** When a game-end condition is met, the system shall display immediate results to the user and store the results in the database.
- **Rationale:** This gives the user feedback as to how well they performed in a certain game and saves the results as a way for the user to look back and reflect on how well they performed.
- **Test Method:** Verify that once a game completes, the scores calculated by the game logic are loaded into the relational database table through an integration test. Also, as part of a manual end-to-end test, make the appearance of a postgame scoreboard a verifying item to check off.
- **Supporting Context:** End condition means that a game/puzzle has been ended either via user completion or time expiration. This assumes that the user completes games via either of these conditions.
- **Tracing Information:** score review [ID-2], storing scores [ID-7]
- **Priority:** Must have

Requirement: *Score Sharing [FR-8]*

- **Statement:** When a share button is clicked by the user, the system shall interface with a social media service such as Facebook or Instagram, or an email service provider to share results of the game.
- **Rationale:** This provides a social interaction between different users of the application to allow for friendly competition as a way of motivation to keep using the application to get stronger in the target areas.
- **Test Method:** Click share buttons for all available social media platforms and verify that game results are sent and uploaded. Cypress can simulate the UI automation and API testing tools can be used to ensure information is shared. Unit tests can be written to test the data delivery prior to integrating with buttons.
- **Supporting Context:** This assumes that the user has linked their application profile with another external social media platform as a way of sharing their results with others. The share button usage implies that the user has already linked their profile to social media platforms.
- **Tracing Information:** social media API [ID-9], email scores [ID-13]
- **Priority:** Should have

Requirement: *Statistics History [FR-9]*

- **Statement:** The system shall provide the user with a complete history of results and statistics for their viewing.
- **Rationale:** Users should be able to view their own history and self-access themselves to determine how well they are progressing/regressing.
- **Test Method:** Have a user complete a game and then navigate to the history page to verify that the results were saved and are now visible to the user. Additionally, check the

database to see saved scores. Test scripts can be made to simulate random scores being uploaded to the database and the user's game history page.

- **Supporting Context:** This assumes that the user will have an interest in viewing their previous scores and their overall statistics of the games/puzzles.
- **Tracing Information:** Performance review [ID-2]
- **Priority:** Must have

Requirement: *Goal Setting [FR-10]*

- **Statement:** The system will provide an interface to allow the user to set goals for future cognitive improvement.
- **Rationale:** A goal system provides the user with an objective to work towards giving them a justification for using the application.
- **Test Method:** Automated unit tests can be set up to verify the logic of handling the goals and storing information in a relational database. A visual test can verify if the goal data is being displayed correctly by the UI.
- **Supporting Context:** This assumes that the user will want to not only want to impede their symptoms but actually improve their cognitive abilities.
- **Tracing Information:** Goals [ID-3]
- **Priority:** Should have

Requirement: *Data Encryption [NFR-1]*

- **Statement:** When personal data is entered, the system shall encrypt the data and store it within a relational database.
- **Rationale:** Data encryption is necessary to fulfill regulatory laws. User’s data must be inaccessible from malicious actors.
- **Test Method:** Verify that personal data is correctly entered and encrypted into the database. This can be tested via test inputs, database records, and confirmed encryption. Unit tests can validate the encryption.
- **Supporting Context:** Personal data is defined as the user’s name, email and anything else that is unique to them that the application needs to create a new account. A relational database is a place to store data in tables using keys. This assumes the user enters this personal information creating an account for the application.
- **Tracing Information:** data encryption [ID-6], database storage [ID-7]
- **Priority:** Must have

Requirement: *Responsive Navigation [NFR-2]*

- **Statement:** The system shall be able to transition between different games and website tabs quickly and smoothly.
- **Rationale:** The ability for the website to run smoothly and quickly provides a better experience for the users.
- **Test Method:** Verify that switching between pages and games on the application occurs without any noticeable delays or visual glitches that result in issues for users. Cypress can

automate clicks and movement throughout the entire application. Transitions can be timed to produce a delay score.

- **Supporting Context:** Transition is the down time between moving from one tab of the webpage to another (game to game, game to homepage, etc). This assumes that the user interacts with the system more than just logging into the application.
- **Tracing Information:** Responsive UI [ID-4]
- **Priority:** Must have

Requirement: *User Authentication [NFR-3]*

- **Statement:** If incorrect login credentials are entered on the login page, then the system will deny access and post an error message to the user.
- **Rationale:** Users need feedback if they entered the wrong login credentials. A lack of feedback usually leads to confusion, and may make the user not want to use the application.
- **Test Method:** Attempt logins with invalid credentials to verify that access is denied and the user is given an error message. Ensure that no data is leaked in the process. Automated scripts can also be used to repeat several invalid attempts.
- **Supporting Context:** Login credentials are some form of a username and password that allows the user to access their personal information on the application. This assumes that the user enters a wrong username and/or password preventing them from accessing their personal information on the application.
- **Tracing Information:** Login protection [ID-6]
- **Priority:** Must have

Requirement: *Password Reset [NFR-4]*

- **Statement:** When the user clicks the “Reset Password” button, the system will send a code by email to validate the user, and if successful provide an input box to provide a new password.
- **Rationale:** When a user forgets their password, they will have a simple way to replace that password with a new one.
- **Test Method:** By clicking "Reset Password," an email is sent and received via email so that the user can set a new password. Automated email testing can be used to ensure that users attempting to reset their password receive an email.
- **Supporting Context:** An input box prompts the user to type a new password allowing the user to re-access their profile on the application. This assumes that the user has forgotten their password.
- **Tracing Information:** password reset [ID-13]
- **Priority:** Must have

Requirement: *Reliable Web Hosting [NFR-5]*

- **Statement:** The system shall be made available on the web to multiple browsers and function cross devices.

- **Rationale:** The web is a standard way to deploy applications, and it will make the product accessible to any internet accessing device with a standard OS and screen. This will allow mobile users to play as well, enhancing portability. A specific device or browser will not be required, granting the potential for a larger user base.
- **Test Method:** A monitoring application can be applied to verify the status of the webpage. The webpage's existence can be tested by accessing it from multiple networks, browsers, and devices.
- **Supporting Context:** This is opposed to hosting the application on an app that would be downloaded from the appstore or google play store.
- **Tracing Information:** Web-application [ID-5]
- **Priority:** Must have

Requirement: *PDF Statistics Download [NFR-6]*

- **Statement:** When users would like to share their game statistics with their doctor, then the system shall allow doctors to access their information via a pdf.
- **Rationale:** It is important for users and doctors to see progression/regression in user cognitive abilities to assess next steps in treatment.
- **Test Method:** Use automated scripts to generate user statistics to place on a downloadable PDF. Verify that the information on the PDF correctly lines up with the data in the database and on the user history page. Use PDF paging libraries for content validation.
- **Supporting Context:** This assumes that the user is willing to share their personal scores and progress with their doctor. This assumes that a pdf is the best way for the doctor to view user progress.
- **Tracing Information:** Doctor view [ID-10]
- **Priority:** Should have

V. Deliverables

Devine Web Application [D-1]

- **Description:** The Devine web application must be hosted on a server 24/7 so that anyone with a browser can access it and users can play exercises with an account.
- **Requirements:** This deliverable fulfills all of the requirements since the requirements are for the app itself.

Devine Frontend Source Code [D-2]

- **Description:** The source code for the frontend of Devine will be released for developers to improve upon within a public Github repository.
- **Requirements:** User Interface [FR-1], Brain Games [FR-2], Pausing Mechanism [FR-4], Postgame Scoreboard [FR-7], Responsive Navigation [NFR-2]

Devine Frontend Documentation [D-3]

- **Description:** Documentation for the frontend source code will be provided as a supplementary resource within the Github repository.
- **Requirements:** The documentation for the frontend satisfies no requirements in itself, rather it helps other developers interpret our source code.

Automated Test Suite [D-4]

- **Description:** An automated test suite will be released within the Github repository for developers to validate their changes.
- **Requirements:** The test suite is designed to verify all frontend and backend components, therefore it helps satisfy all requirements.

Test Manuel [D-5]

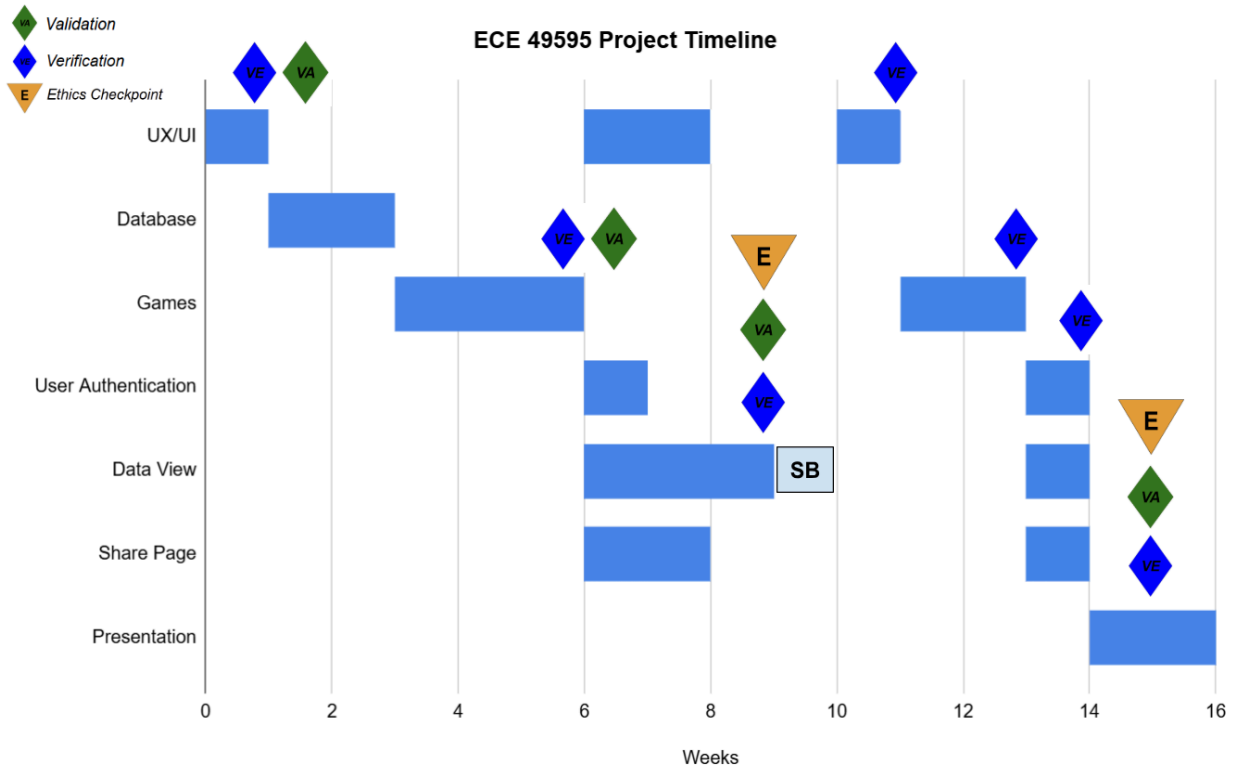
- **Description:** A testing manual will be released with the tests in the Github repository so developers understand the best practices for running tests.
- **Requirements:** The test manual does not satisfy any requirements on its own, but guides developers to help them understand how to run provided tests to verify the product's functional and non-functional requirements.

Devine Usage Video [D-6]

- **Description:** An explanatory video embedded within the application and stored on YouTube will demonstrate the best way to use Devine.

- **Requirements:** The test video satisfies no requirements for the product, but is a teaching medium used to advertise the product and demonstrate the best use cases to site visitors.

VI. Project Management



https://docs.google.com/spreadsheets/d/17_YLmnD18Ab2ZlxQVeTD7aDCEPIQ53iq/edit?usp=sharing&ouid=107941678547125665435&rtpof=true&sd=true

Gantt Chart Description:

Our gantt chart demonstrates a cohesive timeline for the spring 2026 semester project build. First, everything initially depends on if a user interface and skeleton exists for all of the content to be displayed on. Therefore, the first week of the semester will be dedicated to complete this skeleton and apply base formatting standards. Not every aspect of the UI will be created in this step, since the backend data and services need to be configured to make the UI

wholly complete. Other core features heavily rely on the database, which will be developed in the next two weeks. This sprint, like the UI, will be a full team effort so that all members know how to modify and use it as needed. The games are designated three weeks since they make up the core content of the project's purpose. They depend on the database, since the scores will be exported to a table upon the completion of an exercise. Each member of the team will work on games individually to divide and conquer effectively. The next sprint includes the completion of communication, security, and data-centric services. They all depend on the database, but the data view and share page depend on the game scores, which is why it is best to implement them after the games are created. After these three functionalities are implemented, the initial build will be complete. Spring break will commence directly after the initial build.

Multiple markers are positioned on the gantt chart to represent key verification and validation events. For verification, our team's standard is to write the majority of tests prior to the development of modules. After a sprint is over, the automated unit tests will be run for the first level of verification. Bug fixes will be resolved before the sprint is closed. Depending on which sprint is run, integration or system tests may be run. For example, the UI will only have unit tests at first because no other parts of the system will have been built. The database will have some integration tests with the UI, since they are interconnected components. After the sprint for user authentication, the share page, and the data view page, system tests will be generated and run so that the first version of the product will be verified. Due to this structure, integration efforts will be completed before every verification checkpoint when applicable. Validation will occur after various sprints, according to which are most useful to collect user feedback. The UI will require a great amount of validation testing, so it will be tested through surveying directly after its sprint. The games will also need large amounts of surveying and user tests, so the second marker is placed directly after the games sprint. A third validation test will be needed for determining how usable seamless the data view, authentication, and share pages are. This user test will also encompass fixes from the last tests' results.

A second main cycle of development will take place starting in week 10, which will be used to refine many of the insufficiencies from the third validation test. The order of development is similar, going from the UI to the games to the secondary features. The database will already be complete, so unless another feature changes dramatically there will be no need to implement dedicated or significant revision. While there will be revisions according to the user test results throughout the first cycle of development, this second cycle can be seen as a clean slate for innovation that seeks to truly iterate and find new ways to meet the customers needs. A final validation test will be performed at the end of this second cycle, and bug fixes and last minute additions will be implemented in week 14, which serves as an "open period". A final verification test will be run to make sure the project works for the presentation, and the presentation will be given at the end of the semester. This section is highly flexible, as some components will need more iteration than others and will be modified accordingly. Also, while it is explicitly mentioned as the "second main cycle of development" it will not actually be the literal second cycle. This is because while other features are being built, some time can be used to make improvements on the previously completed features. This implies that while our sprints will have a main focus, not every amount of labor during that sprint must be relevant to

the topic. For example, improvements could be made on the games while the social media page is being built. This approach provides multiple minor cycles of iteration, but there will be a large push for iteration after the product is completely developed at week 10.

Task Distribution & Roles:

Note: These roles do not limit the scope of development across team members, but distribute primary responsibility among items.

1. UI Specialist

- UI/UX Design
- Component integration

2. Game Manager

- Game content and design
- Game stats

3. Security Architect

- User authentication strategy
- Data encryption
- User base

4. Analytics Engineer

- Game stat analytics
- Goals
- Social media
- Website monitoring

Development Methodology:

Our team will use a combination of scrum and kanban methodologies to complete Devine. We have structured our gantt chart so that large items are completed in sprints on a macro scale; however, we will use a kanban board to organize micro tasks within each sprint. This is so that we have a clear plan for the semester as a whole while also maintaining an organized set of items to complete during a given time. For the kanban board, we will use a software called Clickup, which allows for multiple views providing dynamic capabilities for project management. We will designate the micro tasks to team members within the software and link which tasks are dependent upon the completion of others. We chose to implement sprints on

a macro scale because we value the centralized effort on particular features so our team will maintain a similar mindset. We are not going to establish formal roles for project management such as a scrummaster or a technical product owner. This is because we will all take on the developer role but also have administrative access to task plan and management. For each sprint, we will have an initial meeting where we finalize the kanban board items to complete so we have a clear understanding of who is completing what items. Also, we will have an end-of-sprint meeting where we apply a formal verification to the items that we had planned. Ultimately, this strategy seeks to combine a centralized macro-approach with a detailed flow within sprints.

Ver/Val Plan:
https://github.com/jkirbs4/Devine/blob/11ea623547b5463ad7ac000670d3ce4a27f1051a/docs/Verification%26Validation_Response.pdf

---

VII. Works Cited

"What is MoSCoW Prioritization? | Overview of the MoSCoW Method." *ProductPlan*, https://www.productplan.com/glossary/moscow-prioritization/. Accessed 26 November 2025.