Verification & Validation

Objectives

1. **Game Suite [O-1]:** Verify that our application gives users access to a minimum of six exercises that target human cognitive abilities such as memory, reasoning and speed. A suite of a minimum of four games was listed in our SDP scope, but we find that it will be necessary to implement more to cover all critical areas of cognitive decline.
2. **Clean Navigation [O-2]:** Verify that navigation between different pages on the application takes no more than 2 seconds in an ideal internet environment. A smooth and quick transition from different pages was a requirement that we listed out in both the SDP as part of the scope of the application and the RVTM making this a valid measurable for our application.
3. **User Reminders [O-3]:** Verify that the application sends reminders at the user's preferred interval (daily, weekly, bi-weekly) to prompt them to complete their goals. This was listed as a requirement in our SDP as a must have feature as it is not only a unique part of our application, but it will also encourage users to return to the application.
4. **Performance Scoring [O-4]:** Verify that each game has its own individual score and a collective score bringing the total number of visible scores to 4 (3 individual and 1 overall score). Our SDP and RVTM describe that the application will provide a complete history and results page to each user by showing them scores across each game category and throughout the application as a whole.
5. **User Goals [O-5]:** Verify that the application will provide each user with 1 predetermined goal with the ability to personally set more additional goals. A goal page that pushes the user to get better with their cognitive abilities was a requirement that was listed in both our SDP and RVTM.
6. **Usefulness for Cognitive Practice [O-6]:** Validate that users of the application find the brain games helpful for improving memory, reasoning, or cognitive speed. This will be measured through distributed surveys and direct communication with individuals that have used the application. This links to the product capability of delivering a meaningful cognitive-training application.
7. **Engagement and Return Rate [O-7]:** Validate that users are motivated to return to the application at least three times per week over a four-week period. This will be measured through usage analytics and ties to the capability for the application to support sustained cognitive training.
8. **Clarity and Learnability of Games [O-8]:** Validate that first-time users of the application can understand and start each game within one minute without external help. Testing sessions and observation will confirm that this assumption holds. This objective links to the capability for the application to deliver an accessible and user-friendly application.

Prioritization

Product Capability	Priority	Brief Justification
A suite of 6+ games focused on memory, spatial reasoning, logical reasoning, and cognitive speed	High	As the main purpose of the site is to provide ways to improve cognitive abilities, having the games and puzzles is a must have and a high priority as it is the main feature of the site.
A performance review system so that each user account can see past scores	High	One of our features to help instigate improvement is goal setting and to do that previous scores and history is important to make said goals. It is important for the users that this displays correct information.
A goal-setting system that allows users to set performance objectives and compare to the performance history to get notified if they are meeting them	Medium	A goal setting page and implementation provides users with an incentive to continue using our application and improve their cognitive skills. Notificing users whether they meet these goals is important in the overall progression of their cognitive abilities.
A simple, flexible, and clearly labeled user interface that is responsive and facilitates the user to critical portions of the program	High	Outside of the actual brain games, this is the second most important feature that the application needs to do a great job on. Making the application more appealing to our target audience will allow our application to stick out more compared to competitors.
A web-based application accessible with internet access	High	The deployment of an internet accessible application means that users will actually be able to access our application content. This should arguably be the highest priority out of all of the items that we deem within the scope of our application.
Data encryption to safely store user personal information and prevent rogue login attempts	Medium	While it is very important to protect user information and medical data the level of encryption required for functionality is up for debate and also considered to not be a main focus of the project.

User database to store personal information such as scores, and account information	High	The storing of any user personal information is a high priority to implement correctly to ensure that no violations of laws (such as HIPAA) occur. The distribution of personal information without the consent of the user is against the law and will need to be taken seriously.
Capability of pausing an exercise causing game logic to stop	Low	While this is within the scope of our application, it is very low in priority compared to other must-have features. This would be very easy to implement so we should devote time to other elements within the scope that are more critical to a fully functional application.

Full RVTM

Our RVTM has been constructed in Google Sheets and can be accessed through the link below.

RVTM Link:
https://docs.google.com/spreadsheets/d/19XLdh89-lHl3bIDnzaWUj5p8U4PFcGvq8pX8vg8rMZo/edit?gid=0#gid=0

Test Approach

Test Levels: There are three levels of testing that we will employ. These test levels are described in the table below.

Test Level	Description
E2E Testing	Test the entire system
Integration Testing	Test a set of interconnected components or modules
Unit Testing	Test an individual module

General Approach: We believe that testing is a fundamental component to ensure the functionality and reliability of software driven systems. For us, untested code is “dead code”, and we must by default assume that that portion of the system is unverified. Our team must adopt a hierarchical approach to verification through testing, in which the lowest components of the hierarchy are unit tests, followed by integration tests, followed by system tests. The objective will be to make the majority of the tests automated, so that the majority of the verification can take place with a single terminal command. Ideally, the unit tests of a given category will be run before the integration tests of that category, and the integration tests before the system tests. Top level integration tests and system tests likely will have manual components to them, since the UI will become a primary interface. Overall, the test suite is designed to provide a quick, repeatable method to verify the behavior of components in a systematic way.

Unit Testing: Unit tests are the low-level tests that are designed to assess the functionality of individual modules. If a unit test fails, that module is unverified until the test is passed. For our product, unit tests will be written prior to the development of our modules in order that a clear standard for logical requirements can be pursued. Our unit tests will be rigorous to test for edge cases in logic, leaving room for higher level tests such as integration and system tests to focus on bringing the system together with correct logic. The unit tests that we generate will be categorical, pertaining to distinct subsystems. Examples of categories include the database, the game objects, or social media API’s. When creating an automated flow of test execution, all of the unit tests for a specified subsystem will be run prior to the integration tests for that subsystem.

Integration Testing: Integration tests will combine the functionality of multiple modules with the purpose of verifying that the components work together. Our team will have two types of integration tests:

1. *Intra Subsystem*: This type of integration test is purposed to validate a set of modules within a subsystem. An example test would be evaluating the behavior of multiple game objects within the game subsystem.
2. *Inter Subsystem*: This type of integration test is designed to validate the interfaces of subsystems. An example would be a test designed to evaluate how the game results get stored inside of the database. This merges the database subsystem and game logic subsystem.

Some of these integration tests will be smaller than others, with the purpose of isolating logic and parameters. Ultimately though, longer tests are necessary to verify that all of the components work together.

System Testing: The majority of system tests will be manual due to the UI’s presence. Our team will attempt to mock the UI with automated button pushes to an extent. Ultimately, we will need to visually observe to verify the screen behaves according to our UI and game window design specifications. For manual system tests, a clear and detailed procedure will be written out for the verifier to follow. A set of objectives will exist

alongside the procedure that serves as the means of evaluation. System tests will be the last tests run in the hierarchy, since in order for the system to work as a whole, the subsystems must be verified.

Non-Functional Testing:

- **Security:** Much of our security testing will be through our unit and integration testing of our authentication and encryption protocols. A system test can exist where the tester mocks a malicious user attempting to do harm to the system.
- **Scalability:** We envision two main scalability tests. One of these tests will be a database subsystem integration test, in which we introduce large amounts of synthetic data to verify storage limits and performance speed. The second test will focus on user traffic, and be designed to verify that the host platform will be able to serve an adequate number of users at one time.
- **Performance:** This metric will largely apply to the brain games themselves. We will have system tests in which the user manually plays the game and reports on lag and delays.
- **Observability:** Our system will output logs and record user data in a streamlined way that we can observe from the administrative end. These records of data can be scanned to ensure they store the correct types of data in an integration test that occurs directly after a system transaction.
- **Privacy:** This can be verified as a manual test in which a client user must find no way to observe the data of another client user.
- **Compliance/Regulatory:** Regulatory tests will be specifically tailored to organizations such as HIPPA. An example that applies to our project concerns logging database transactions with personal data. Our system will produce the required logs, and our test will verify that the logs contain the correct information.

Regression Testing: Regression testing will be utilized at the proper project checkpoints. Ideally, we will have much of our test suite automated, and can run it quickly after a feature, bug fix, or iteration. Our test suite will allow us to tweak which types of test will be run, such as all of the tests of a layer, or all of the test regarding a specific subsystem. If a bug is fixed within a specific subsystem, then only the specific subsystem (or only a portion of it) needs to be re-verified before deployment. To make our regression tests as streamlined as possible, we will connect it to our CI/CD pipeline so that they run after a commit occurs in Github.

Validation Plan

Usability & Performance Evaluations:
We will aim to ideally recruit 3-5 participants per round. Each round will focus on key flows linked to our high-priority capabilities and objectives: navigating between pages in under two seconds, viewing performance history, and setting/monitoring goals. The specific tasks to be undertaken during sessions are:

1. Creating an account and logging in
2. Starting a specific game from the menu
3. Viewing performance history and interpreting recent scores
4. Set or update weekly goals and verify that reminders are configured.

The following metrics will be used to measure the success of this particular section of the plan:

- **User Satisfaction:** After each task, participants rate ease of use on a 5-point Likert scale (1 = very difficult, 5 = very easy). We should aim to have ≥ 80% of participants rate overall ease of use ≥ 4/5. The mean satisfaction score should be ≥ 4.2/5 for navigation, game launching, and viewing performance history. An open response section will enable participants to clarify any aspects that were especially confusing.
- **Error Rates:** We will define three distinct error types for validation:
    - **Navigation errors:** going to the wrong page, needing more than one backtrack, or getting "stuck" for ≥ 30 seconds.
    - **Form/interaction errors:** invalid inputs, mis-clicks that change the wrong setting (e.g. wrong goal), or failure to submit a form.
    - **Conceptual errors:** misinterpreting performance charts (e.g. thinking a lower score means improvement when it does not).

  We should aim to have ≤ 10% of all task attempts end in a failure. Errors will be Captured via observation, screen recordings, and event logs (e.g. repeated Invalid inputs).
- **Task Completion Times:** For task 1 (account creation + login), the target median time will be three minutes. For task 2 (launching a game from the dashboard), the target median time will be 30 seconds. For task 3 (open performance review and locate last week's scores), the target median time will be 90 seconds. For task 4 (set or modify a weekly goal), the target median time will be two minutes. We will aim to have at least 90% of all participants complete all four core tasks without assistance and at or below the above thresholds in the final validation round.

Stakeholder Involvement Plan:

Our project has multiple stakeholders with distinct validation needs: primary users, loved ones, and clinicians or medical researchers who may use collected data to support care or studies. Operational risks that should be accounted for include tech illiteracy (particularly among older users) and data shortages (limited time and sample size for evaluation). Three distinct phases with varying numbers of stakeholders from each category will be defined for our project:

1. **Concept and Wireframe Review:** 1-2 clinicians and 2-3 loved ones will review low-fidelity workflows and the performance review mockups. This phase aims to validate that the tasks and terminology make sense to non-developers and align with expectations for a brain-training tool.
2. **Prototype Walkthroughs:** 4-6 representative users plus 2 loved ones will perform core tasks (play a game, view history, set goals) on the working

prototype. We will collect both subjective feedback and quantitative measures using the aforementioned metrics from the Usability section. Any high-severity concerns will trigger design changes before expanding to a larger pilot.

3. **Pilot Validation Sessions:** Near the final release of the project, a small pilot cohort (e.g. 8-12 users across all stakeholder categories) will use the application over 2-4 weeks, with usage analytics and follow-up interviews/surveys. This will mitigate the data shortage risk by gathering longitudinal evidence of usability and perceived benefit within the course timeframe.

Stakeholder-specific metrics will be collected for each phase above:

- **User Satisfaction:**
    - **Primary users:** ≥ 80% should report that the application is “easy” or “very easy” to navigate (4 or 5 on a 5-point scale). ≥ 75% should report that game instructions and goals are “clear” or “very clear.”
    - **Loved ones:** ≥ 80% should agree that they could “confidently” introduce our project to a family member through explaining its basic purpose and how to start a game.
    - **Clinicians/researchers:** ≥ 70% should agree that the performance review page provides “sufficiently clear” trends and metrics to be clinically useful as a supplementary data point.
- **Error Rates:** Primary users will use the same error targets as the general usability section. For loved ones, we should aim to have < 10% encounter errors when helping a user sign up, log in, or share results. < 15% of clinicians/researchers should report misinterpretation of charts or metrics in a post-session quiz (e.g. identifying whether a trend is improving or declining).
- **Task Completion Times:** Same target times for primary users as stated in the usability section. For loved ones, the time to help a new user sign up and launch their first game should ideally be ≤ 5 minutes in the final pilot. For clinicians/researchers, the time to locate a specific user’s recent history and summarize their last week’ performance should be ≤ 2 minutes.

Tools, Environments & Test Data Sets

Testing Tools:

- **Github Actions**: This can be used to include automated test runs to our CI/CD pipeline after commits.
- **React Testing Library (RTL):** This is a library that can be used to test our UI components in an automated fashion. It is built on top of DOM Testing Library and is specialized for React frontend applications.
- **pytest:** This python library can be used for backend unit and integration testing. It displays test coverage metrics, and is practical for scaling an autonomous test suite.

Simulation or Emulation Environments:

- **Docker:** We will create a docker image and container as a uniform environment for our testing. We will have to vary the environment after initial product development to ensure consistency of correct behavior across platforms.

- **Cypress:** This tool is able to test web-application behavior externally without referencing any objects created in code. It provides a user perspective, but acts as a bot surfing a site according to rules that our team can set up. Automated system testing will be where this tool will thrive for verification.
- **JSDOM:** This tool is able to simulate a DOM without functioning as a real browser. It is commonly used with RTL.

Data Sets Used:

Three categories of data sets exist that will be considered for testing purposes. These categories are:

1. **Realistic:** A dataset that shares the same schema, statistical distributions, noise levels as the actual data while having values that feel real.
2. **Anonymized:** A dataset that has real data but has been preprocessed as to remove personally identifying information.
3. **Representative:** A dataset that could be real or synthetic but represents the actual distribution well.

For our project, we will have a data table for users, stats per user per game, stats per user, and connections. Each of these sets will be a blend between realistic and representative, since the schemas must match for testing and because there is guaranteed privacy with nonreal data. We will be able to synthesize values with realistic data types and even generate testing accounts that can be worked with. The distributions may not be correct in our synthetically generated data, because we have no standard to compare how individuals will score until we perform user testing. The tests then will be used not to establish statistical claims, but to validate the functionality, reliability, durability, and scalability of the data pipeline.

Version Control & Test Management Tools:

- **Github:** Github is the industry standard version and branch control tool for maintaining a codebase history and updating the current state of the project.
- **LambdaTest:** A cloud based testing platform that manages tests on actual mobile devices rather than just emulators.

Test Configuration:

- Github is set up by creating a repository and adding users to it. Github Actions can be configured by turning on actions and building a workflow.
- Data sets can be created with a python script that generates random or distributed data, then ingested to the database files within a command or automated test.
- Docker can be set up by downloading the desktop client, creating a docker-compose file, building the docker image, then running the docker container. This will involve including all of the dependencies for the project environment within the image.
- For Vite, RTL can be installed with the following command: npm install --save-dev @testing-library/react @testing-library/jest-dom @testing-library/user-event. The testing

library can be included with the line: “import '@testing-library/jest-dom';”
- JSDOM is installed with Jest (JavaScript testing framework) with the command “npm install --save-dev vitest jsdom”, configured in an environment file, then run using npm.
- Cypress must be installed with “npm install --save-dev cypress” then the directory structure must be built with “npx cypress open”.
- LambdaTest requires users to create an account, then they can plug in other testing frameworks such as Cypress, and see results on their interface.
- Pytest can be installed through pip package manager and imported like a standard python library. Each file and directory must either be prepended with “test_” or appended with “_test” for the framework to function correctly.

Test Archival: Our product has varying components involved such as a frontend, backend, database, and game suite. Therefore, different combinations of these tools listed above will be used to verify functional and non-functional traits of the program. To organize tests well, every test script and file will be located within a “tst” directory. As for documenting the purpose of the tests, another testing document will describe each test and its purpose so that the development team will have a clear and shared understanding of the test purpose.

Works Cited

- **React Testing Library (RTL):** https://testing-library.com/docs/react-testing-library/intro/
- **Cypress:** https://www.cypress.io/#create
- **JSDOM:** https://www.testim.io/blog/jsdom-a-guide-to-how-to-get-started-and-what-you-can-do/
- **pytest:** https://docs.pytest.org/en/stable/
- **LambdaTest:** https://www.lambdatest.com/lp/automation-testing