

COMPUTER-BASED SPATIAL SKILLS TESTING

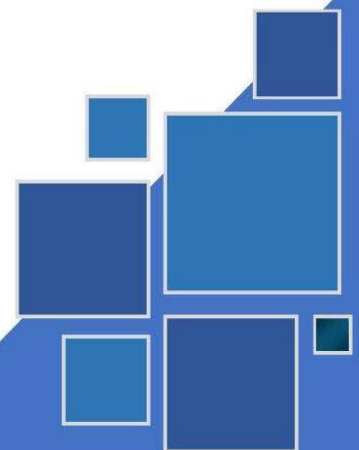
GROUP REPORT
CS 399

PREPARED BY:

Team 22:

Luke Wang - Front-end Developer
Ethan Bland - Front-end Developer
Jack Huang - Front-end Developer
Ryan La - Back-end Developer
Damon Greenhalgh - Full-stack Developer

Version Updated:
30/10/2022





Executive Summary

A 2009 Vanderbilt University review in the Journal of Educational Psychology [2] looked at over 50 years of longitudinal research on spatial ability, and concluded that “spatial ability plays a critical role in developing expertise in STEM.” Research has consistently shown that early spatial skills can predict later success in these fields. Therefore, it is crucial that education providers know their students’ spatial abilities to assist them in further succeeding in STEM-related subjects.

Traditional spatial skills tests involve students taking the test using pen and paper, which can cause many issues like being time-consuming as results are not generated instantaneously. Having to mark the test manually means that results precision can not be guaranteed. The test will most likely have to be invigilated by a supervisor. The atmosphere of taking the test in exam conditions can bring unnecessary pressure and anxiety, influencing students’ results. Traditional spatial questions are often not very interesting. They can only test for spatial perception, spatial visualisation, and mental rotation due to the limitation of question formats being static images, leaving out visuospatial working memory as it requires the test to be interactive. Most of the time, it will require interviews with participants individually.

This project aims to assist education providers in analysing strengths and weaknesses in students’ spatial skills. We have two main deliverables. Firstly, we have extended the current spatial test on coderunner by adding 29 new questions with various types, namely spatial perception, spatial visualisation, and mental rotation. Secondly, we have developed a web application that acts as a platform for creating and taking an online test. Tests are automatically marked instantly with perfect accuracy, and results are stored for viewing only by the test’s creator. An assessor can create a test with a random code. The code can then be distributed to student users to take the test. The test process works similarly to coderunner, except it has no format constraints. Our application has the capability of testing all types of spatial skills in a single test, including visuospatial working memory. We have integrated two fully interactive memory games in the question bank, where the admin can choose to add into a test along with other standard questions.

We believe our application is built in a flexible and scalable way so that clients can create their desired tests easily. Due to mainly time constraints, we could not implement some of the features that would have otherwise improved usability. For future improvements, we recommend adding a universal question bank so that new tests can be made with verified existing questions from the bank to reduce the time it takes to make a test. Users should be able to create and upload questions. Another suggestion is adding graphs and charts to the admin dashboard, allowing clients to visualise data to perform informative analysis on test results efficiently. We also suggest adding a feature where test creators can choose to make test results and answers visible to students. Lastly, we recommend implementing a dark mode for user readability preferences in terms of interface aesthetics.



Table of Contents

Executive Summary or Abstract	2
Table of Contents	3
Introduction	4
Project Background	5
Background	5
Existing Solutions	5
Project Value	5
Methods and Tools	6
Project Specification	8
User Requirements	8
Use Cases	9
Project Design	10
Design Choices	10
Home Page	10
Spatial Skills Tests	11
Testing Interface	11
Dynamic Memory Games	13
Admin Dashboard	17
Three-tier architecture	20
REST APIs structure	21
Database design	22
Data flows	23
Project Implementation	25
Git Workflow	25
Project Management	25
Interaction between client-side and REST API	26
Authentication & Authorization	27
Database implementation	27
Marking of student answers	27
Deployment	27
Unforeseen Problems	28
Results & Evaluation	30
Future Work	31
Conclusion	32
Project Aims	32
Key Findings	32
Major Outcomes	33
Declaration of Authorship	33
References	34
Appendix	35
Gantt Chart	35
Final Build	35
Project Demonstration Video	35



Introduction

Spatial skills are a very important metric in predicting the performance of individuals within STEM subjects. Our goal for this project was to develop several components that would allow teachers and educators to test and measure the performance of spatial ability for their students, which would in turn allow them to predict their performance in the scope of these subjects.

We aimed to develop three main components. Firstly, a Coderunner based spatial skills test, which would contain a large repository of questions for teachers to utilise for spatial skills testing purposes. Secondly, we planned to develop a spatial skills web application, which would allow students to take spatial skills tests in a dedicated application for the purpose of measuring spatial ability. Thirdly, a game based in the Unity engine, which would measure spatial ability based upon the content within the game, and would provide an illusory element that would make students playing the game be unaware that they are being tested. This in turn would allow students to take a “test” with elements that made it more engaging and fun and would likely yield a higher completion rate.

Each individual component of the project contained multiple features which we planned to implement, and some others that would eventually become revealed as the project furthered in its development. For the Coderunner test, the objectives were quite straightforward; create a repository of spatial questions researched and found online, and then implement these questions accordingly within Coderunner, with a provided source for each question so our client would have the ability to detect for any copyright issues related to the question. Our web-based application would have a consistently expanding scope as the project progressed, but our primary objectives were the following:

1. A quiz system for students to navigate through and answer questions.
2. A database to store quizzes and questions, along with other information relevant to taking the test.
3. A dashboard which would allow for the creation of tests and questions.

In regards to the spatial skills game, we had a general idea that the game would need to be straightforward, easy to learn, and have a defined beginning and end with a score displayed for teachers to record.

For carrying out this project, our approach was to use GitHub in order to manage our collaboration, regularly scheduled meetings to ensure we were on track, and in general a form of collaboration which emphasised communication and organisation.

As a result of these plans, our outcomes were the following:

1. A Coderunner based test with a large array of spatial questions, sectioned into their respective categories.
2. A web app which would allow for the testing of spatial skills, as well as the creation of tests and questions.
3. A game-based spatial test which would test students without their knowledge.



Project Background

Background

Spatial skills are our ability to “generate, retain and manipulate abstract visual images” [1]. There are four primary types of spatial ability; visual perception, mental rotation, spatial visualisation and spatial working memory. The Journal of Educational Psychology published a study in 2009 [2] has shown that spatial skills are strongly correlated with performance in a STEM subject. That is, students with good spatial skills are more likely to perform better academically and build a career in STEM fields. Educators in these STEM fields may want to identify the spatial ability of their students in order to assess their academic strength and weaknesses, and potentially alter their teaching to aid those students.

Existing Solutions

In academics, the traditional method of assessing spatial skills involves physical paper tests [3, 4], typically with multichoice questions such as the Mental Cutting Test or Revised Purdue Spatial Visualisation Tests. Paper tests however have various issues. Logistically, the assessment process is very laborious and time consuming. It requires printing, test administering, marking and storing results, all while trying to avoid human error which could lead to inaccurate results. Paper tests are also unable to assess spatial working memory due to the paper being static; it cannot change and force the student to recall information. Finally, these traditional tests are not engaging or fun to take, particularly for younger students, which could lead to demotivation and a worse result.

Online testing platforms for spatial skills also exist, such as 123test.com or assessmentday.com [5, 6]. While these platforms help alleviate the issues of paper tests, they still aren't perfect. The questions remain unengaging and spatial working memory often remains untested. These platforms typically force you to use their provided tests, thus causing the test to be uncustomizable. Finally, these platforms often require account creation and usage to take tests and store results, limiting accessibility.

Project Value

Our project, Visuo, solves the core issues that come with paper tests by digitising the testing format and automating tedious tasks (such as marking results). As the software handles various tasks previously requiring manual attention, carrying out spatial skill assessments requires less time and effort. Visuo also incorporates dynamic spatial working memory questions, allowing educators to assess the full range of a student's spatial ability.








To make our project truly unique within existing software solutions, we wanted to make our testing platform customizable, accessible and modern. Educators should be given the freedom to design and administer the tests that best suit their situation, rather than be forced to use tests / questions already provided. Another specification for our project was for it to be usable by anyone through a website, including those outside of educational institutions or established groups who may not have a user email / account set up. Finally, we wanted to develop a testing platform that feels modern in both interface appearance and user experience; the platform should be simple, easy to understand and enjoyable to use.






Along with our primary deliverable through the web application, we were also tasked with extending the universities CodeRunner testing platform by providing spatial skills testing questions that were well researched and used in real world tests. Another deliverable, although optional, was to develop a game-based spatial test that assessed a players spatial ability without them noticing the testing environment. This ideally would allow users to enjoy the ‘testing experience’ and increase their motivation to develop their spatial ability.

Methods and Tools

Throughout development, we used the following tools:

Tool	Use	Reasoning
 Figma	UI/UX Design	Figma was used to design front-end page appearances and initial user experience without requiring code. It also allowed us to quickly gain feedback from our client meetings without starting development.
 React.js	Front-end Development	React was used to develop the front-end code of the web application. We used React as it was a modern, well documented framework that allows us to create dynamic HTML components for our interfaces.
 Express.js	Back-end Development	Express was used to build the back-end REST API, allowing the front-end and database solution to interact and send data between each other. We used Express as it is well documented and is similar to other back-end frameworks we have used.
 Postman	API endpoint testing	Postman was used during the development process of our REST API to conveniently test individual endpoints without having to write manual tests using code, which is difficult and time consuming.
 Node.js	JavaScript runtime	Node is the runtime for our JavaScript code. It encapsulates our back-end and manages external packages that aid development. We used Node as it is well documented, cross-platform and scalable.
 MongoDB	Database Solution	MongoDB is a NoSQL database that holds data for our web application. We used MongoDB as its JSON-like schemas were easy to use, while being free, quick and reliable.
 AWS EC2	Website Deployment	EC2 allowed us to deploy our website by hosting our app on Amazon’s virtual computer instances. We used EC2 as it was reliable and scalable (and other Amazon services weren’t suitable for our app).

<p>AWS S3</p> 	<p>Content Delivery Network</p>	<p>S3 allows us to upload, host and retrieve data onto Amazon's network. We used S3 as it allowed us to quickly retrieve assets with minimal load time, as well as being secure and scalable.</p>
<p>Git / GitHub</p> 	<p>Version Control / Project Tracking</p>	<p>Git and GitHub were used for our version control to allow team members to synchronise code updates and manage development workflow. GitHub was also used for project management by using GitHub Issues for feature tickets and documentation / discussion. We used GitHub as our team members were familiar with the tool, and it is the industry standard VCS.</p>
<p>VS Code</p> 	<p>Code Editor</p>	<p>Visual Studio Code was used as our team's primary code development environment, as members were familiar with the tool and it supported the various technologies our project required.</p>



Project Specification

User Requirements

Visuo is a web application that provides a platform for educational institutions to create and administer spatial skills tests for students. It has the following specifications / primary features, which also fulfils our user requirements for the project:

Admin Dashboard (Admin)

Through the initial home page, authenticated users can login through their Google account as an admin to access their admin dashboard, where the user can view existing tests that they created. Through this dashboard, they are able to:

1. View details about tests the user made and their contents
2. View and share the test code, allowing users to take the test
3. View the results of students who have taken the test. Additionally, be able to export the test results in a .csv format
4. Preview their created test by sampling a playthrough
5. Access the test / question editor to change test settings and content

Test / Question Editor (Admin)

An admin user can access the test / question editors through the dashboard. The editor allows the user to change the settings and content for the selected test. The editor facilitates actions such as:

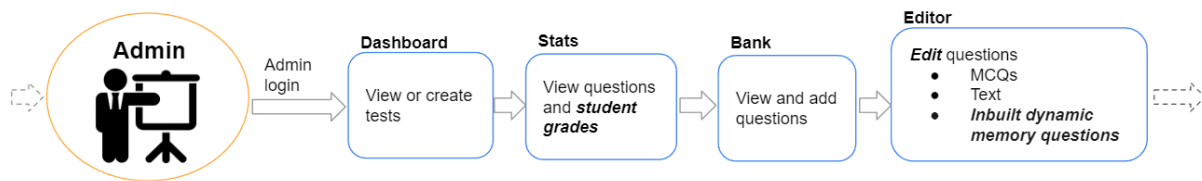
1. Creating and deleting tests, as well as editing existing test settings
2. Creating, editing and deleting test questions by the user. Users can create and upload their own custom questions, with varying types (e.g. multichoice, text entry, dynamic memory questions).
3. Browse and filter through the questions of a test

Testing Interface (Student)

Through the initial homepage, student users can enter an identification name of their choice, along with a valid test code to access the corresponding test. The user then proceeds through the test, which plays out as the creator has set it to. Once the user finishes, or the test time runs out, the test result is automatically submitted, marked and stored for the creator to view.

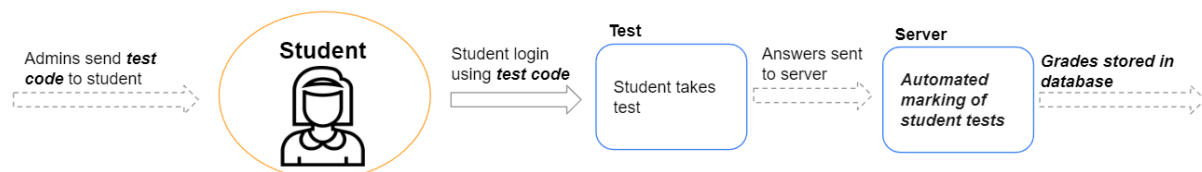


Use Cases



Case 1: Teacher wants to create spatial skills test and distribute to students

1. The teacher loads up the website URL, where they are able to login through the home page by using their Google account.
2. If the user is authenticated as an admin, they proceed to the admin dashboard. They can now create a test and add spatial test questions.
3. Once the user finishes creating their test, they can publish it and share its code with the class externally (e.g. through email).
4. Students can now take the test using the code.



Case 2: Students want to take the test provided by a teacher

1. The student loads up the website URL where on the home page, they can enter a identifier name of their choice, as well as a valid test code provided by an admin.
2. The student goes through the test, which emulates a virtual testing environment in accordance to the settings made by the test creator.
3. When the student finishes the test, their result is automatically marked and stored for the test creator to view.

Case 3: Teacher wants to view the results of their students

1. The teacher loads up the website URL, where they are able to login through the home page by using their Google account.
2. If the user is authenticated as an admin, they proceed to the admin dashboard. They can select a test they created to see the test score of users who completed the test.
3. The teacher is able to export the test results as a .csv / spreadsheet format for further detailed analysis.



Project Design

During the development process and interactions with our client, we wanted our project to fulfil three primary qualities: **accessibility**, **simplicity** and **customizability**. We want our project to be easily used by anyone, education institution or otherwise, without too much difficulty or required learning. It was intended to be lightweight for a supposed quick, one time use experience, while still being flexible and providing depth for educators who want to facilitate different types of tests.

Design Choices

Home Page

The home page, where all users initially load into, is a simple page with minimal elements, allowing easy navigation. The user is presented with two entry forms for their name and a test code; they would use this if they were a student trying to take a test from an instructor. A student is able to enter any name they wish as an identifier to take the test, as long as the name hasn't been used before. This allows the test to be accessible to anyone, without requiring user accounts.

Figures shown: primary page elements for home page and admin login page.

A separate login page is provided for admins / teachers to access the admin dashboard. Admin users are authenticated using their Google account email to prevent unauthorised access. Upon successful login, the user is then redirected to the dashboard page. As we use existing Google accounts for the login process, no account creation required, thus simplifying the process.



Spatial Skills Tests

The spatial skills tests themselves were designed to mimic their real life paper test counterpart as closely as possible. Test creators are able to add their own custom spatial test questions of any category into the test. Each non-dynamic question consists of a question image and text, accompanied by either answer images or a text input form for multichoice / text entry questions respectively. The platform also provides dynamic spatial memory questions for the test creator, used to assess memory which can't be done with static paper tests. Dynamic questions have an associated seed attached with it, making its randomly generated pattern the same for all users who take the test. This prevents unfairness in testing through some students getting easier / harder patterns. The question order and answer order (for multichoice questions) can be randomised to make it difficult for cheating to occur.

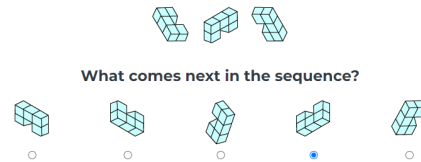
The time allowed to complete the test is separated into two categories; sequential and non-sequential. In a sequential test, the user must go through each question linearly, each with an individual question time. The user cannot reverse or move freely between questions. For non-sequential tests, the user is able to freely traverse between questions, much like a regular paper test. The test time uses a total time which counts down across all questions. Our client desired the ability to have both options for test creation, as sequential tests is a feature missing on existing testing platforms (e.g. CodeRunner) that could aid with test invigilation.

Testing Interface

The test page, intended to be used by the student, was designed to be simple and easy to navigate, while providing a modern and minimalistic look. This reduces potential frustration and stress while navigating a potentially new interface during a test. The interface is also designed to include all the necessary information within a single glance at the screen; no scrolling is required to see all parts of the question, or to check information such as test time, which reduces efficiency and could further cause frustration.

The interface itself consists of a question / test timer and question number display in the top corners. Questions appear at the screen centre, dynamically changing without having to load a new page. For sequential tests (user cannot traverse questions freely), a 'next question' button will show only if the user has selected an answer; else it will stay hidden, as to prevent continuation without answering. A sliding timer will also be present, indicating the time the student has left until the question will finish and move on.

For non-sequential tests, previous / next question buttons will always be shown, along with a question navigation toolbar that allows the user to easily traverse between each question. The toolbar buttons change colour for questions that have been answered, allowing students to make sure they haven't missed any during the test. Finally, there is a 'finish test' button in the lower right corner, which allows students to submit their test early. Clicking it will upon a confirmation modal overlay, which prevents accidental submissions.



Finish Test

Figure shown: testing interface for a non-sequential test.

Upon finishing a test, the user will be redirected to a finish page, where they are alerted that their result has been marked and stored for review. A student is unable to see their result unless it is provided by a teacher as to prevent cheating.

The testing page also minimises potential exploits through the interface that could lead to unfair examination or “cheating”. For example, a user cannot restart the test by refreshing the page or going back to the home page; this will stop the test without submission. In a non-sequential test, a user cannot restart memory questions by going to another question and back, as once the memory question begins, it cannot be taken again.



Dynamic Memory Games

We supply two inbuilt games to admins when creating a test: a pattern game and a card-matching game. We implemented those two games to give admins a way to test for visuospatial working memory. The most commonly used method was to conduct a Corsi Block Tapping Test in a personal interview with the participant. In this test, cubes are tapped by the examiner in sequences of increasing length after which participants are required to reproduce each sequence immediately.

Pattern Game

Our client wanted to have the ability to integrate this test within a standard spatial skills test, so we replicated the test and made it into a pattern game as one of the questions. The game adopts the same rules and layout as the original test, but using two dimensional representations.

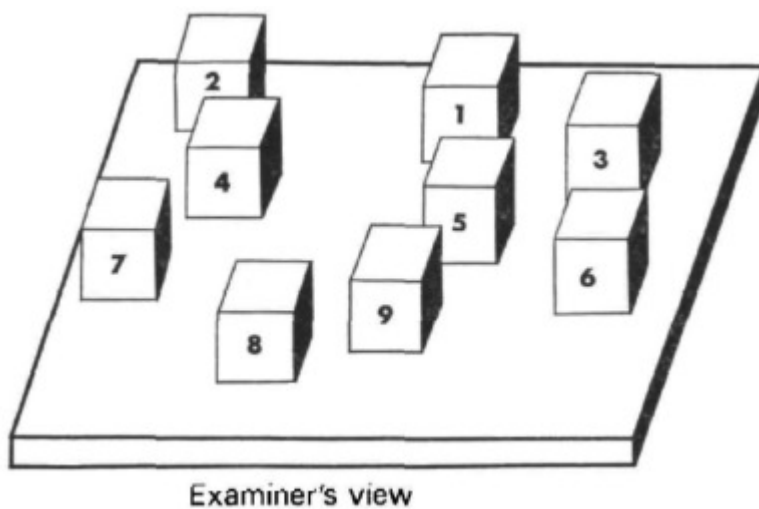


Figure shown: Corsi Block Tapping Test Setup [13]

Lives:  55 level: 1

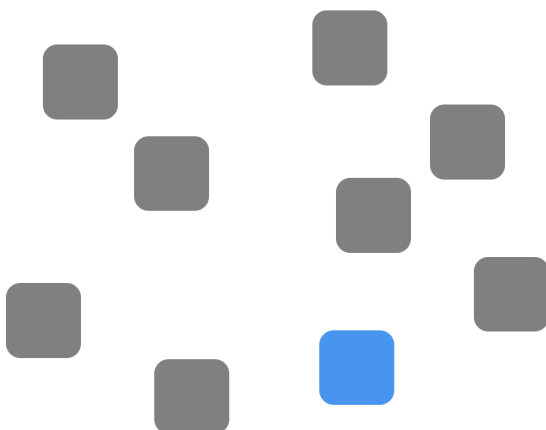


Figure shown: Game Interface of Corsi Block Tapping Test



Extending the current Corsi Block Tapping Test, we have implemented a variation of the original test. Instead of having blocks aligned non-linearly, they are arranged in a grid layout with columns and rows. At the start of a game, some random blocks will flash in a different color for a short time, indicating the pattern, then the user must reproduce the pattern by clicking on blocks. Each mismatch will reduce a life. The game will end upon losing all lives, time runs out, or completing the game. Upon successfully duplicating the pattern, the user will enter the next level, which then a new pattern will flash. Looking at the interface, lives, time and level is displayed at the top and the game is in the middle of the screen.

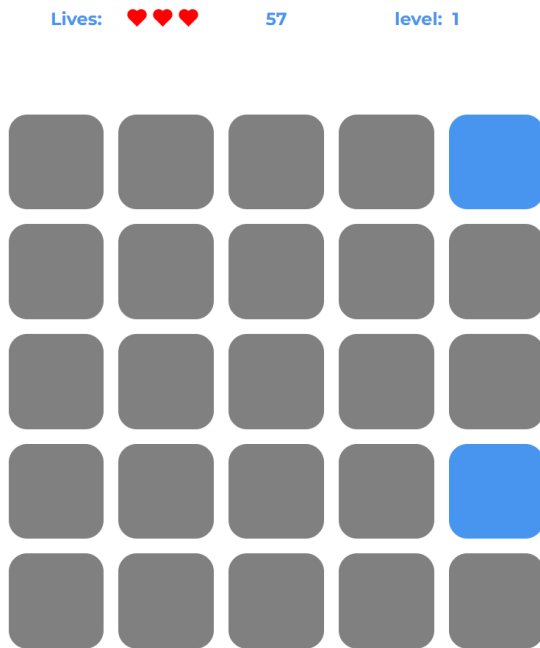


Figure shown: Game Interface of Variation of Corsi Block Tapping Test

The figure below is a snip of our admin dashboard showing everything possible to change for this game. Most settings are self-explanatory. If it's not, the description of the field and valid input will be shown if you hover over the question mark icon on the right. There are general settings for games, such as lives, time, and random seed (which is used to generate randomness of game patterns). But there are more advanced settings.

Going down list of settings:

- Grid Size (a numeric input field that determines the width and height of the grid, if the game is not using Corsi layout.)
- Corsi (a toggle option that decides the layout of blocks to be in Corsi fashion or the grid)
- Reverse (a toggle option that decides if the pattern blocks need to be clicked in the reverse order)
- Random Level Order (a toggle option decides if patterns of each level build on the pattern in the previous level)
- Pattern in Order (a toggle option decides if the pattern will be shown block by block or all at once)
- Pattern Flash Time (an input field that takes a number that indicates how long the pattern will be shown for at the start of a game)

Grid Size	<input type="text" value="4"/>	?
Seed	<input type="text" value="12345"/>	
Lives	<input type="text" value="5"/>	
Corsi	<input checked="" type="checkbox"/>	?
Reverse	<input type="checkbox"/>	?
Random Level Order	<input type="checkbox"/>	?
Pattern in Order	<input checked="" type="checkbox"/>	?
Pattern Flash Time (s)	<input type="text" value="1"/>	?
Time Limit (s)	<input type="text" value="60"/>	?

Figure shown: Admin dashboard settings for pattern game

Card Matching Game

This is our second inbuilt game. The goal is to match all pairs of identical cards. At the start of a game, all cards will be revealed for a short time to show their number and suits before flipping back. Then users are able to click on two cards to flip them. If these cards match, they stay flipped, otherwise, they flip back again after a short delay. The interface layout is the same for both games. We have also provided various ways that admins can alter this game. General settings work the same as the pattern game.

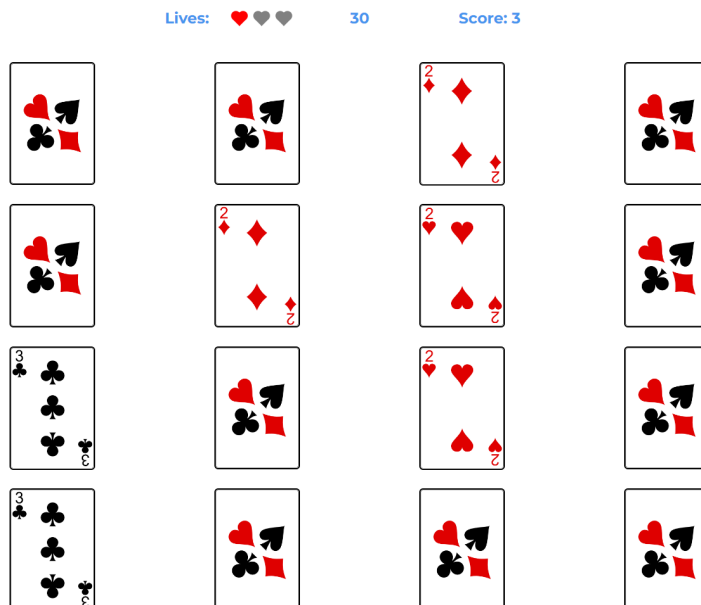


Figure shown: Game interface of Card Matching Game

There are three advanced settings to note:



- Number of Pairs (a numeric input field that decides how many pairs of cards will be shown and to be matched)
- Start Delay (a numeric input field that indicates how long cards will be revealed at the start of a game in seconds)
- Selection Delay (a numeric input field that indicates the delay before flipping cards back after a mismatch in seconds)

The image shows a settings interface for a card matching game. It consists of a light blue background with several rows of settings. Each row has a label on the left, a numeric input field in the middle, and a question mark icon on the right. The settings are: 'Number of Pairs' with value 8, 'Seed' with value 12345, 'Lives' with value 3, 'Start Delay (s)' with value 1, 'Selection Delay (s)' with value 1, and 'Time Limit (s)' with value 60.

Setting	Value	Help Icon
Number of Pairs	8	?
Seed	12345	
Lives	3	
Start Delay (s)	1	?
Selection Delay (s)	1	?
Time Limit (s)	60	?

Figure shown: Admin dashboard settings for card matching game

Before users can start a memory game, there will be general instructions that will help users understand what they are required to do. We also give test creators a section to display any messages they want to show. Memory games will only begin after users click the start button.

Memory Test: Card Matching

Good Luck!

Instructions

When the game starts, 7 card pairs will be temporarily shown.

Match those cards in pairs before time runs out!

You will lose a life for each mismatch.

Note: once you start, you cannot redo the question!

Start

Figure shown: Instructions before a game starts



Admin Dashboard

When an administrator logs into the app, they will be greeted with the admin dashboard. This is where the admin can view tests that they have created, as well as the option to create a new test.

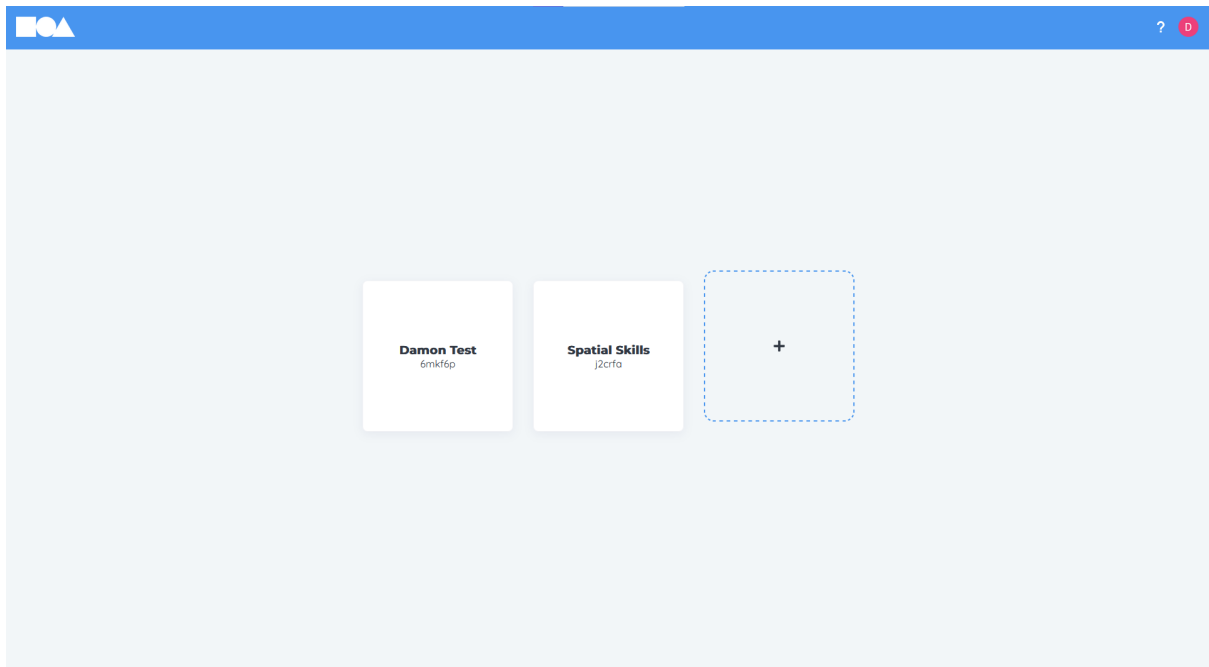


Figure shown: Test selection page

We designed this page with simplicity in mind, with tests centred and focused in the middle of the page. Hovering over any of the items will play a minor animation indicating its interactivity. Clicking on a test will bring you to the statistics page.

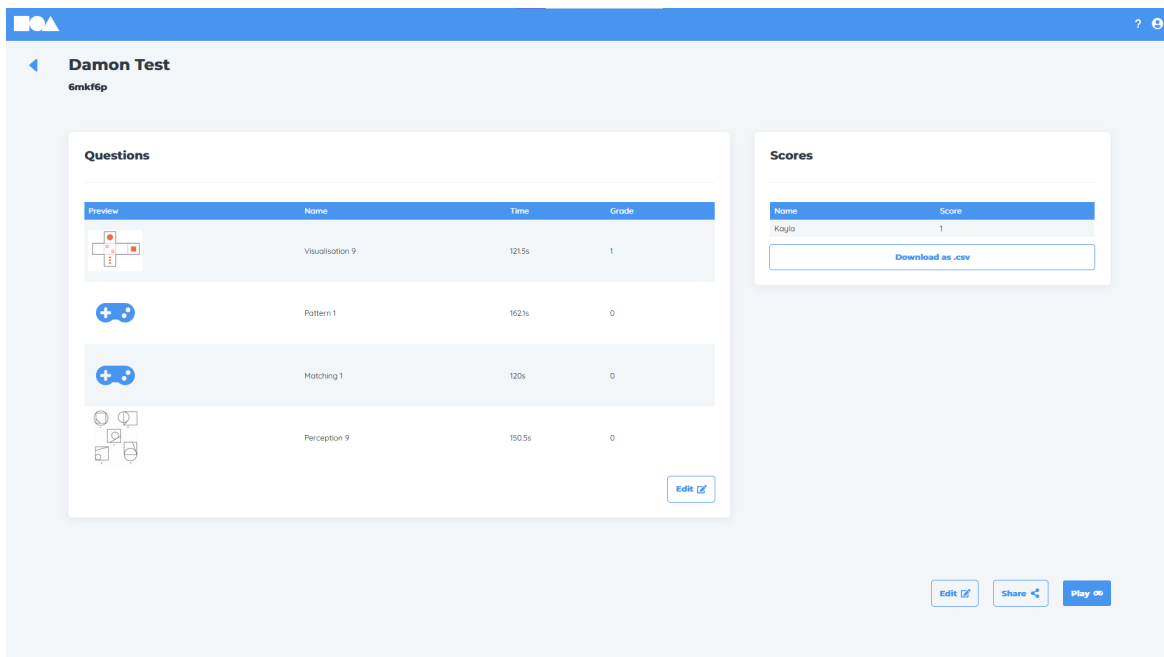


Figure shown: Statistics page for a test.



Here admins get an overview of the test that they have created. On the left we have previews for questions within the test, including its name, time limit, and associated grade. On the right we have a minor overview of students and their scores. Admins have the option to download more detailed student answer information at the bottom of the scores panel. Finally, the bottom right contains action items which allow admins to edit test settings, share the test code, and preview the current test. We use our accent colour to highlight key areas to draw the user's attention. Notice that we use it for all buttons, notably the 'Play' button uses a higher concentration to differentiate it from the others, as it is of greater importance. Clicking edit within the question panel sends the user to the question bank page.

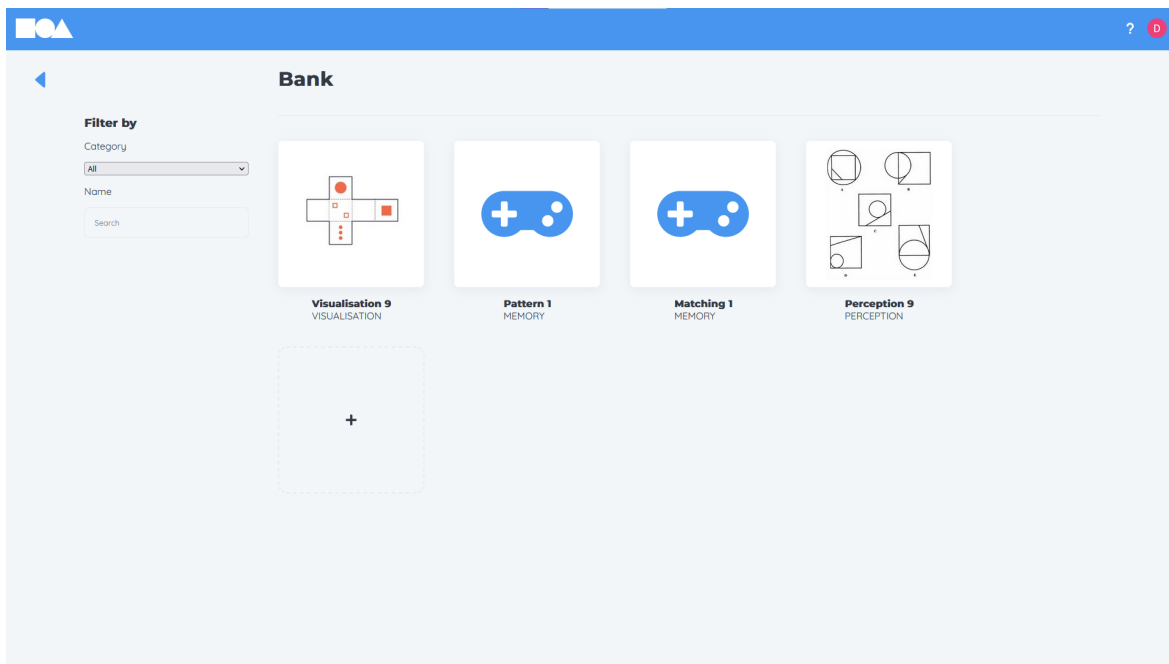


Figure shown: Preview of the question bank.

This page allows users to quickly search and filter through questions within the test. Users can filter questions by question category (memory, perception, rotation and visualisation) or by name by interacting with the sidebar. We intentionally made question thumbnails large as they are the main focus of the page. Hovering over any of the questions will play a similar animation as seen on the test selection page to indicate intractability. Finally, when users want to edit/create tests or questions with Visuo, they will use a special editor component.



Edit Question

Title: Perception 9

Description: Find the shape that does not belong in this group.

Citation: <https://www.quizbeez.com/iq-tests/spatial-reasoning-3c>

Image:

Category: Perception

Type: Text

Answer: B

Grade: 12

Time Limit (s): 150.5

Create Test

Title: Title

Published: ☐

Back Traversal: ☐

Shuffle Answers: ☐

Shuffle Questions: ☐

Time Limit (s): 60

Create **Delete**

Left: Editor in the context of editing an existing question. Right: Editor in the context of creating a new test.

The editor is a single component that dynamically renders input fields based on the context of the app. We decided to build the editor as a single component to reduce the amount of duplicate code that would have been required to support this feature. Each field has an associated name, user input, and an optional help tooltip. The image user input fields will display a preview of the image when uploaded. During early tests of the editor we found users became frustrated while editing as existing settings were not automatically filled in, so we built a system that would autofill fields if there was an existing value for it. We also noticed some users were unsure what a setting would actually do based on the name alone, so we added a help tooltip, which when hovered would display a popup with a more detailed description of the setting.

Back Traversal

Shuffle Answers

Shuffle Questions

Enabling this setting will allow students to freely view different questions without answering the previous questions. Enabling this setting will also disable the individual time setting for each question. The total quiz duration will be decided by the "Time Limit" field instead.

Figure shown: Preview of help tooltip for ambiguous setting fields.



Three-tier architecture



Figure shown. Diagram of three-tier application architecture.

Considerations. When deciding on the application architecture for Visuo, to best achieve our project goals, the most important factors we had to consider were performance, security and time. Performance was important as student tests are timed and a slow interface or response time may affect a student's results through increased frustration or loss of time. Admins are able to create, view and edit questions while students should not have these privileges, hence security needs to be taken into consideration. Ease of development was another important factor as there was very limited time for this project and most team members did not have experience with web-application development.

Table. Summary of the three-tier application architecture [7].

Tier	Description
Client (presentation)	Displays graphical user interface in web browser and collects information from users.
REST API (application)	Process information collected from users and using applications business logic.
Database (data)	Storage of application information.

Chosen architecture. A three-tier architecture approach appeared to best suit our needs. Decoupling the client-side scripts with the REST APIs allowed us to keep the user interface lightweight and move computationally intensive processes to a server running the REST APIs, which can more easily be scaled. This enabled a more responsive interface for Visuo, which results in a more seamless testing experience for students and accurate results. The separation of client and REST APIs also allowed for more security as user access is not determined through client-side scripts a user can tamper with, but hidden within the application tier. This architecture allowed the team to divide work naturally into the client-side interface and server-side REST APIs. These were developed and debugged simultaneously, resulting in a more efficient development process.



REST APIs structure

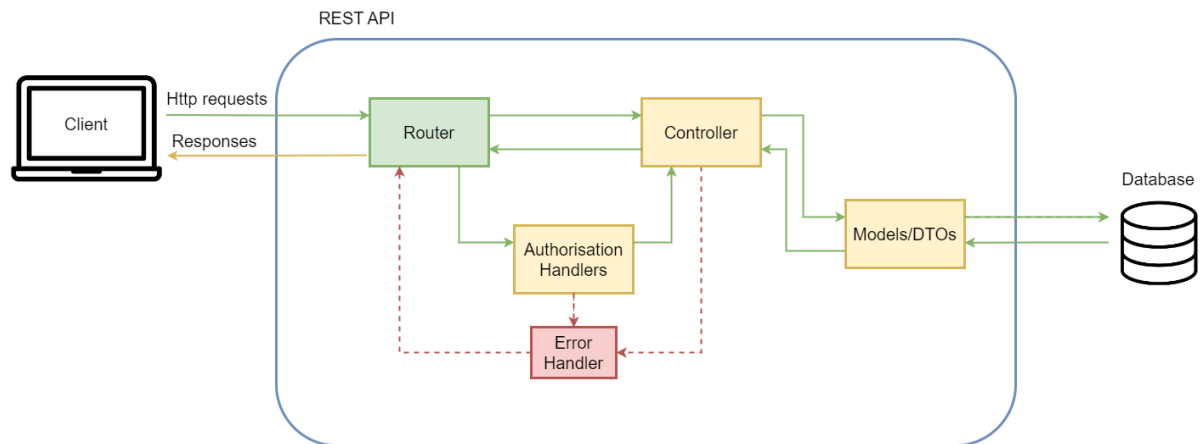


Figure shown. Chart of REST APIs structure and information flow.

The REST APIs were structured into five components. The Router initially handles the HTTP request by parsing the request body, cookies and headers. The Router then passes the request to the appropriate functions in the Controller or the Authorisation Handler if needed. The Controller handles the main logic of the REST API such as creating questions to store in the database or marking answers. The Authorisation Handler parses the authorisation headers and cookies. If there is an error generated during processing in the Controller or Authorisation Handler, the error is sent to the Error Handler, where the error gets formatted and sent back to the Router and to the Client. The Models and DTOs are document schemas which are used to validate documents that have the correct fields and to hide sensitive information such as answers when a student accesses question documents.

This modularisation of the REST API was chosen as it allowed for easier debugging and multiple different functions to be developed simultaneously.



Database design

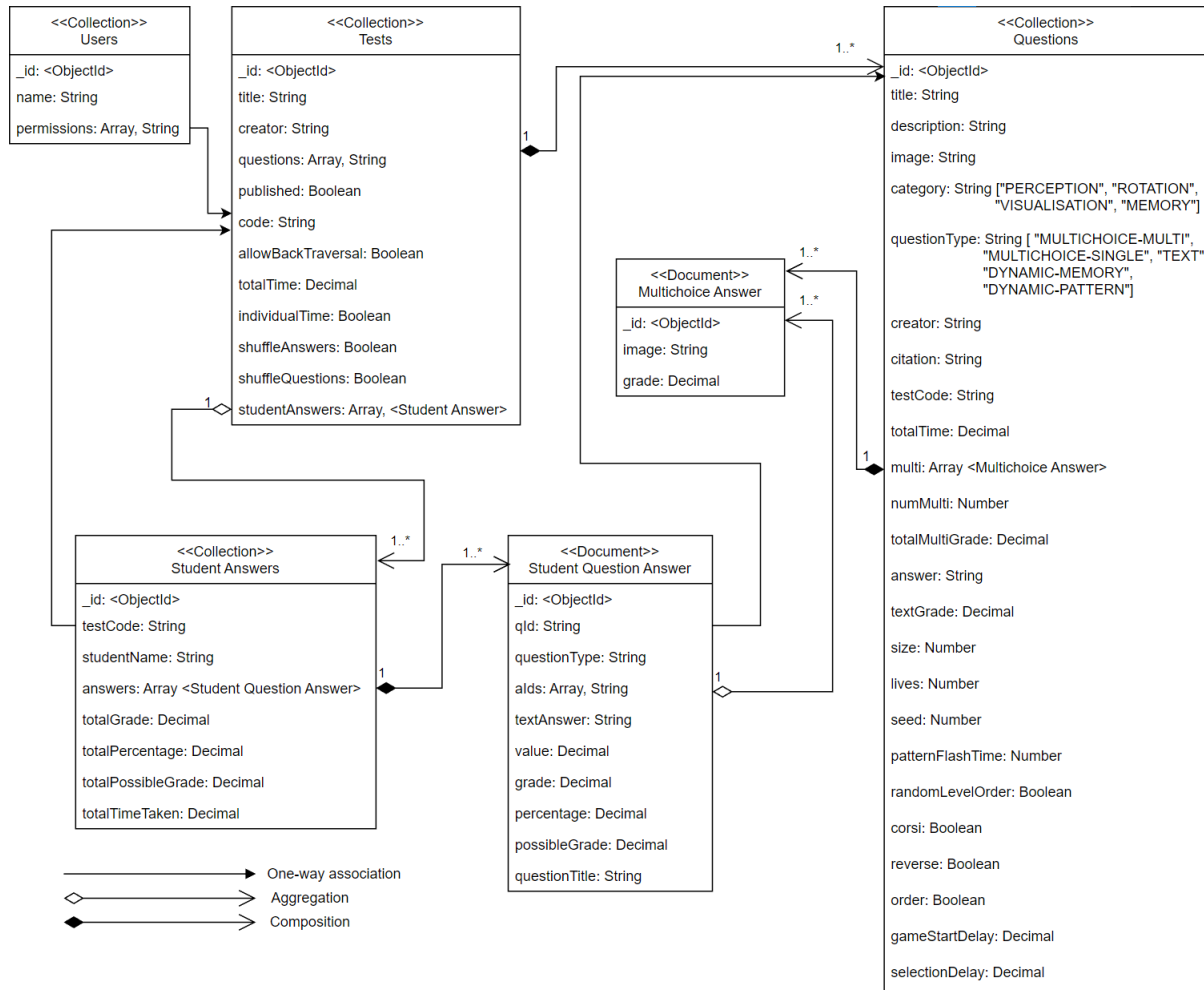


Figure shown. UML diagram of collections and documents in database.

Chosen database. We decided to use a document database instead of a traditional relational database for two main reasons, performance and flexibility. The main performance bottle-neck of relational databases are join operations. Relational databases can reduce some of this bottle-neck through denormalisation, but using a document database allows us to do this much more naturally through embedded documents and arrays. Document databases are also much more flexible as documents in a collection do not need to conform to a specific schema. During development this flexibility allowed development and access to the database to not bottle-neck when making changes to data models, which was frequent.

Question model. The question model is generic and used for all four types of questions Visuo currently supports (multichoice-multiple response, multichoice-single response, text, dynamic-memory, dynamic-pattern). Having a generic question model was chosen due to simplicity and made debugging easier. However, this does mean there may be some redundancy (e.g lives not needed for a text question) and if storage space does become an issue, creating different question models could be considered at the expense of complexity.

Question images. Question images are stored in a separate content delivery network (CDN). This is due to performance reasons. Document databases do not usually have native



support for efficiently storing and retrieving images. Hence it was decided to use a CDN that efficiently does these operations.

Relationship between users and tests. Each user either has a permission to a test or are admins. Users can only take tests that they have permissions to. Admins can only delete and edit tests and subsequently questions in the test they have created which is specified in the test creator field.

Relationship between questions and tests. Questions are only available to tests they belong to. This means that when a test is deleted all its associated questions are also deleted. This decision was mainly due to the unexpected complexity of having a global question bank, particularly with streamlining the user interface and propagating the effects of deleting or modifying a question. Would a test that currently uses the question keep the old or new version? If the new version is used, do we need to notify owners of the test using that question that the question has changed? How will this affect the mark of students who have already done the test? We did not have enough time to implement a satisfactory solution to all these complexities so decided on the polished simpler version that was implemented.

Relationship between student answers and tests. Each test has an array of student answer documents. Although these student answer documents belong to the test, they are also stored in their own collection. This is so records of student answers are kept persistent even if an admin accidentally deletes a test.

Data flows

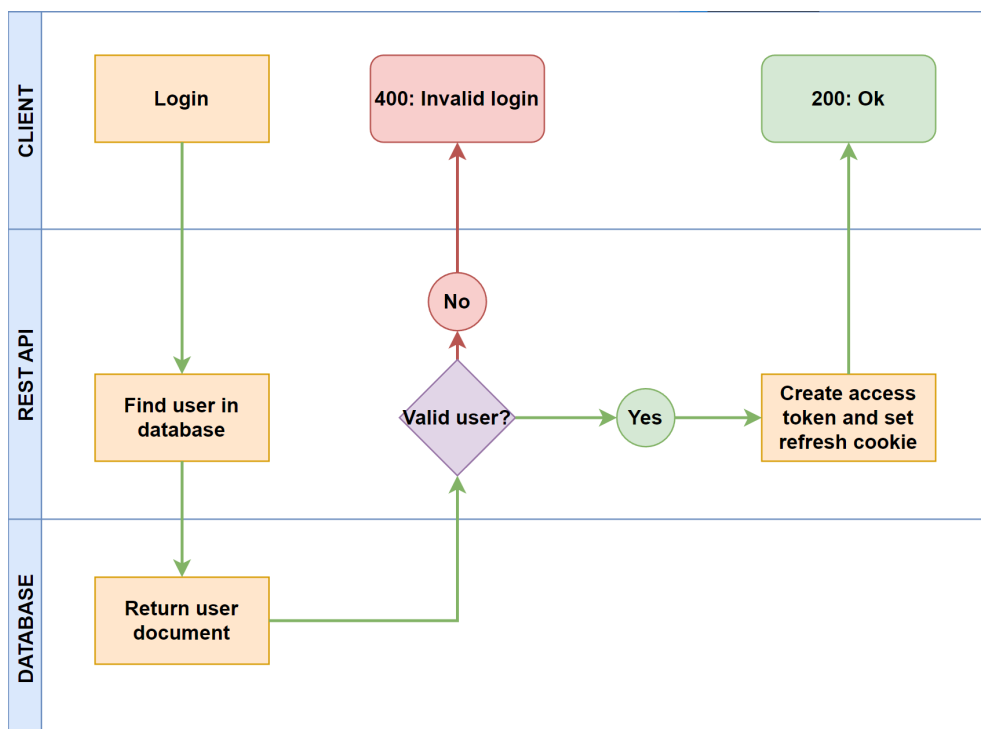


Figure shown. Data flow diagram of user login process.

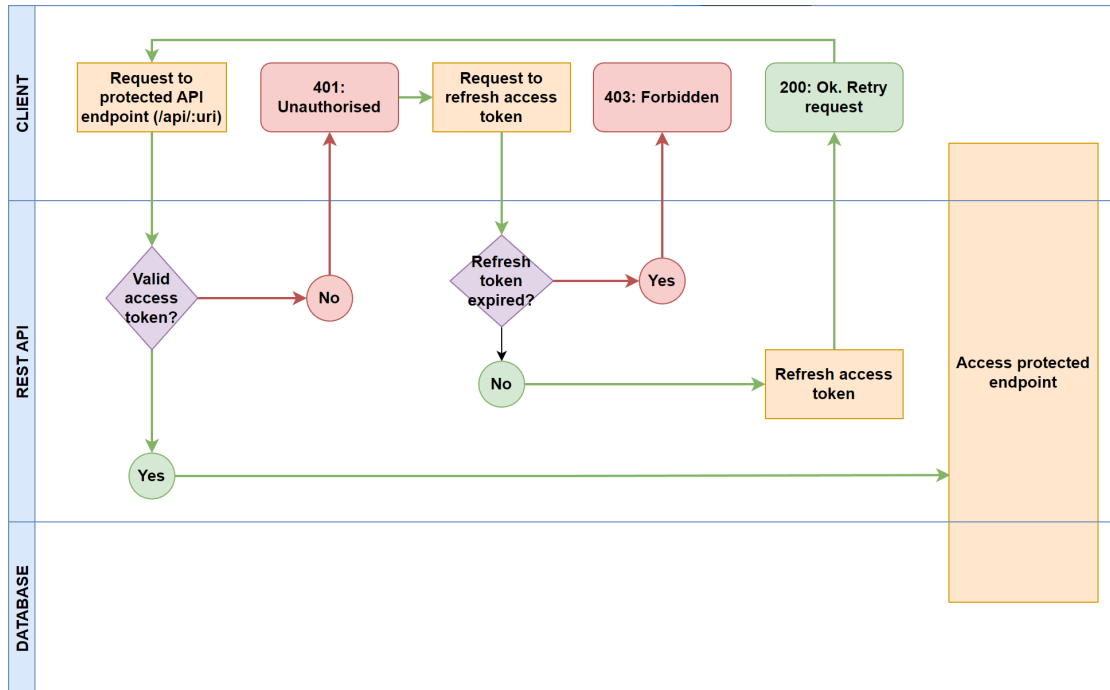


Figure shown. Data flow diagram of access token checking and refreshing when requesting access to a protected API endpoint. The access protected endpoint block is shown in the following figure.

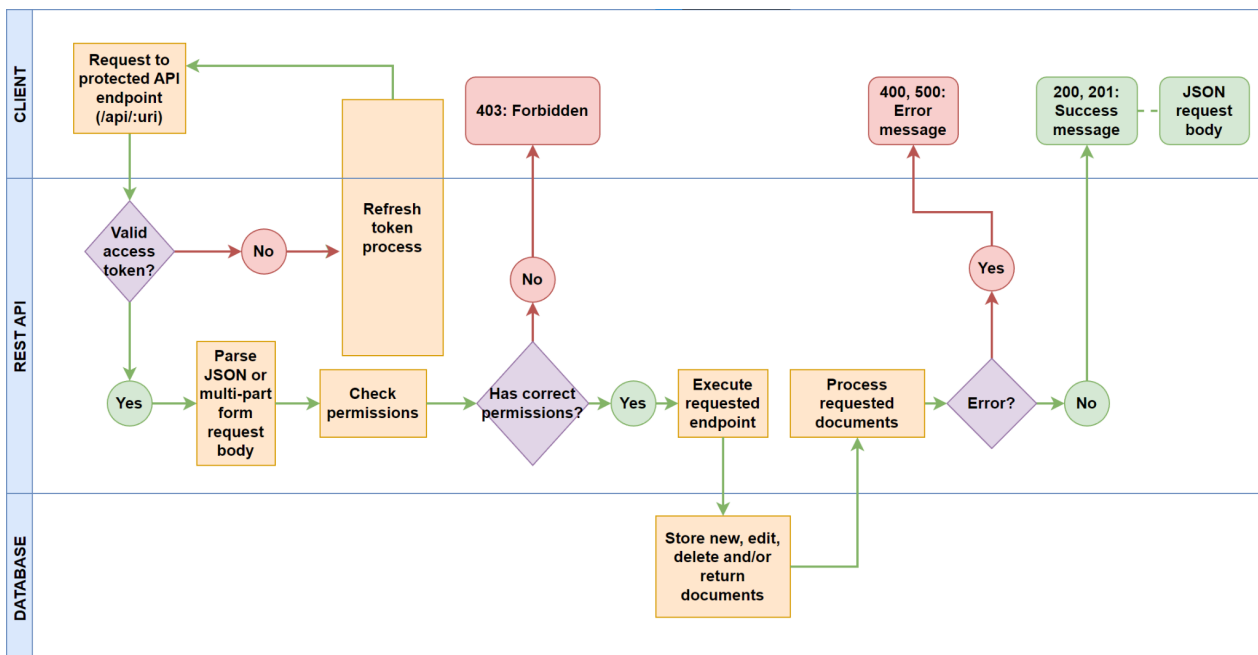


Figure shown. Data flow diagram of access to a protected API endpoint. Refresh token process block shown in the previous figure.

The three figures above show the three types of generic data flow processes for Visuo. Both admins and student logins follow the same data flow structure, however the admin login will search for a valid admin in the database whereas the student login will search for a valid



student. All API endpoints were designed to follow a similar data flow structure with minor modifications when needed. This allowed for easier debugging as we can backtrack through the data flow diagram to find the sources of errors or unintended outputs.

Project Implementation

Git Workflow

For our version control workflow, we decided on a feature branching approach where each feature would be relegated to its own dedicated branch. This strategy had many advantages which include,

- Features/issues were focused and decoupled from each other. Each member could focus on their individual tasks and did not have to worry about massive merge conflicts.
- Detecting the cause of bugs was much easier as we could track the branch where it originated from.
- We used Github issues to track feature implementations; by using this method we enable future developers to have the ability to track our development process.

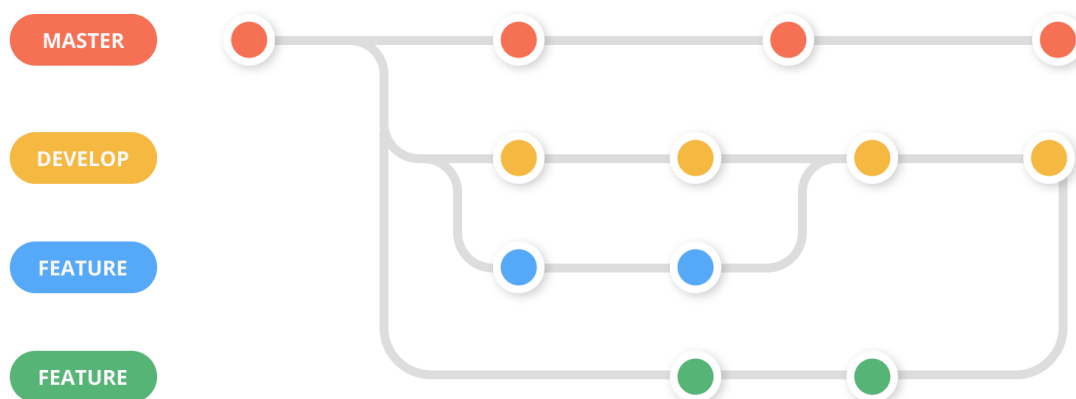


Figure shown: Representation of our git workflow during development.

We utilised issue and pull request templates as a means to keep our development documented and consistent.

Project Management

We followed an agile methodology for this project. Each sprint was one week long, with each sprint starting on Monday and ending on the following Sunday. During sprints we would have tri-weekly standup meetings where each member would talk about what they had worked on since the previous meeting, what they will work on till the next meeting and if they had any challenges/blockers. We documented these meetings in a minutes format here. We used Github Projects along with Github Issues to track development progress.

Interaction between client-side and REST API

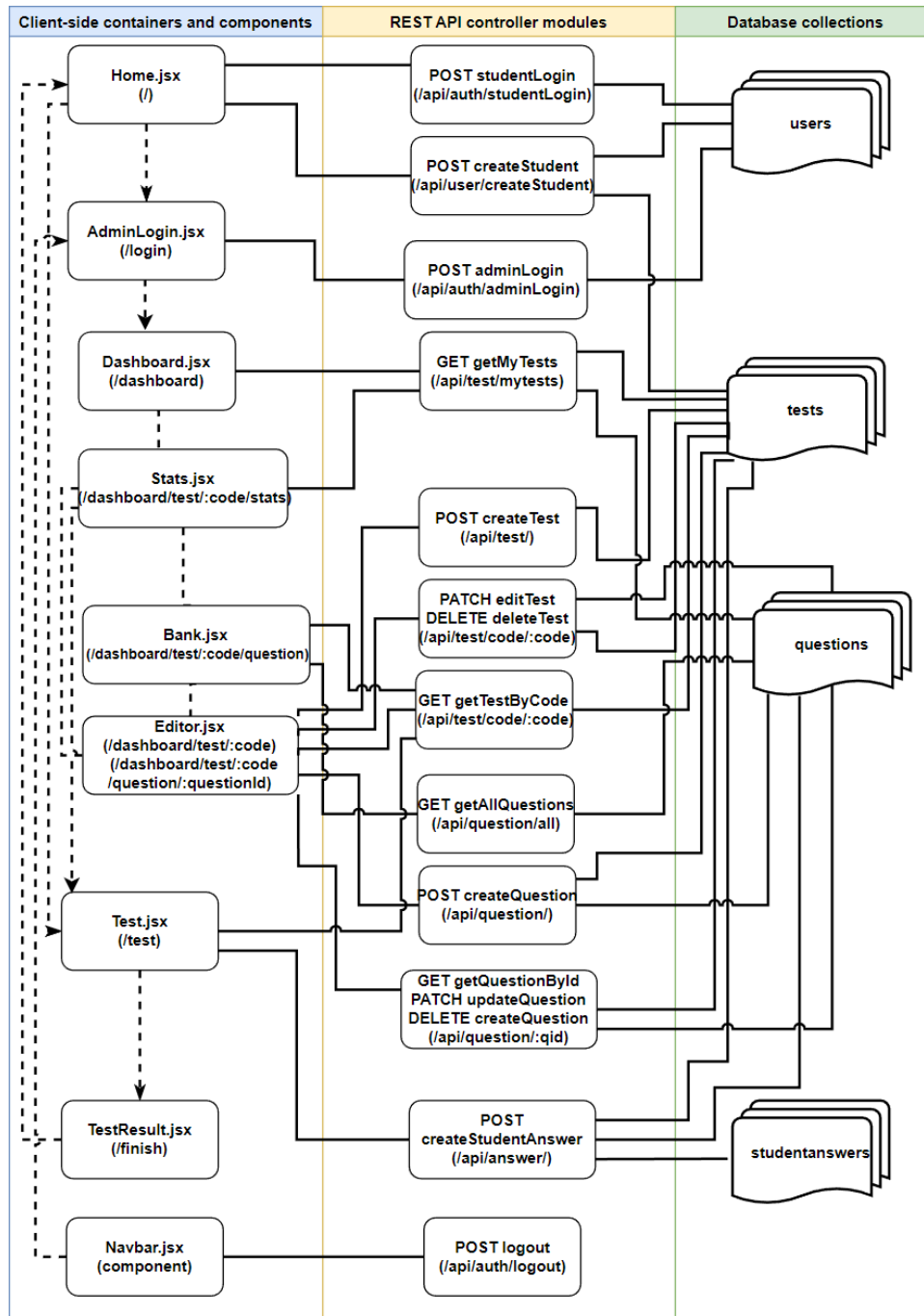


Figure shown. Interactions between client-side containers (pages) and components with REST API endpoints and database collections.

The figure above shows the interactions between client-side containers (pages) and REST API endpoints implemented in Visuo. Endpoints were called using Axios [8]. Axios was used to set the authorisation headers and retry requests in cases where access tokens needed to be refreshed.



Authentication & Authorization

NPM packages:

- @react-oauth/google: client-side google sign in
- google-auth-library: server-side google ID token verification
- jsonwebtoken: creating and verifying access and refresh tokens

In the user document admins have 'admin' in their *permissions* array while students have the test code they have permissions to. Students both create a user and login when they enter a name and code. This essentially makes student users one-use and will have to use a different name if they want to retake the test. This was chosen instead of implementing an explicit sign up process for students to simplify the test taking process, reducing friction from the time a student receives a test code and a student taking it. There is no sign up process for admins on Visuo. Admins are registered by the web-application owner directly into the database.

When a user logs in, access and refresh tokens are created and used to determine permissions when they try to access protected endpoints. Access tokens are stored in session storage and sent in request headers (Authorization: Bearer access token). They have a short expiry time of 30 minutes. Refresh tokens are stored in an HTTP only cookie and have an expiry time of 3 hours.

Database implementation

The database we chose to use was a MongoDB Atlas cloud instance. The npm package *mongoose* was used as the MongoDB object modelling tool in Node.js. MongoDB schemas were used to structure the documents in the database. These schemas evolved overtime during development and are,

- question.js
- student-answer.js
- test.js
- user.js

Question images were stored in an Amazon Web Service S3 bucket CDN. This was more responsive than storing them in the MongoDB database.

Marking of student answers

Automated marking is implemented in Visuo. After a student completes a test, student answers are automatically sent to the REST API, evaluated and recorded in the Student Answer collection. The grades for multi-choice and text questions are stored in the *grade* field. There are no grades for the dynamic memory games, alternatively the scores for these games are stored in the *value* field. The *grade* and *value* fields are in the Student Question Answer document which is embedded into the Student Answer document. *totalGrade* in the Student Answer document records the total grade of all multi-choice and text questions. Percentages, possible grades and time taken are also recorded. A function to export these marked student answers as a csv was also implemented.



Deployment

Deployment domain: <https://hydrohomiebeerbro.com/>

An Amazon Web Service EC2 t2.micro instance running Ubuntu 22.01 LTS and an NGINX web server with port 80 setup for HTTP and port 443 setup for HTTPS was used as the deployment server. The following are the steps to deploy the build on this instance [9, 10, 11, 12],

1. In your local machine *npm run build* in the client folder of the project to create the production build of the React application.
2. Install node.js in the EC2 instance using *sudo apt-get install -y nodejs*
3. Git clone <https://github.com/uo-compsci399-s2-2022/visuo.git> in the EC2 instance.
4. SSH (e.g using PuTTY) the dotenv file into the server folder and production build into the client folder to the EC2 instance.
5. Install packages into the EC2 instance using *npm install* in the client and server folders.
6. Install NGINX using *sudo apt install -y nginx* . NGINX was used as the web server and reverse proxy to redirect HTTPS requests.
7. Configure NGINX by opening the configuration file using *sudo nano /etc/nginx/sites-available/default* and replacing the location area with

```
location / {  
    proxy_pass http://localhost:3001;  
    proxy_http_version 1.1;  
    proxy_set_header Upgrade $http_upgrade;  
    proxy_set_header Connection 'upgrade';  
    proxy_set_header Host $host;  
    proxy_cache_bypass $http_upgrade;  
}
```

8. Obtain SSL certificate through these commands
sudo apt update
sudo apt install snapd
sudo snap install core
sudo snap refresh core
sudo snap install --classic certbot
sudo ln -s /snap/bin/certbot /usr/bin/certbot
sudo certbot --nginx
9. Install pm2 process manager using *sudo npm install pm2*. Navigate into the server folder and run Visuo using pm2 through command *sudo NODE_ENV=production pm2 start app.js*



Unforeseen Problems

We had several problems over the duration of the project,

- **Upscaling**
A majority of the team were new to modern web development technologies. As a result, our upscaling period was longer than expected, leading to delays to features and project deliverables.
- **Model Design**
Due to changes in project goals and input from our client, our original model designs would not suffice. We decided to completely redesign the object models within our database, leading to changes to our API endpoints which required more work to integrate the new endpoints.
- **Testing**
Limited options for testing the app due to the breadth of edge cases and interactions which the app allows for.
- **AWS Delay**
We planned to use several AWS services (S3 for content delivery and image storage; EC2 for deployment) but we unfortunately got access to it late into the project. As a result, the editor (for file upload custom questions) and deployment were delayed and completed much later than expected.



Results & Evaluation

Project Goals

We achieved all required project deliverables fulfilling the clients expected requirements.

We extended an existing CodeRunner test by providing 29 new questions (10 perception, 10 mental rotation, and 9 visualisation). All questions are free to use, cited, and academically relevant to the context.

We developed a web-based spatial skills platform called Visuo, which allows educational institutions to painlessly create, edit and share spatial skills tests. We developed multiple games which test visuo-spatial working memory which include card matching, pattern matching, and the famous 'Corsi Block Tapping Test' as a variant of the pattern matching game. Users can build custom content with the power editor, supporting a plethora of options which change the behaviour of tests/questions.

We were unfortunately unable to complete the optional game-based spatial skills test due to limited knowledge of the toolset and time constraints.

Testing

Before pull requests were merged into other branches, we would often peer review the content; through this we were able to catch bugs, fix code style and optimize implementation. Due to the large upscaling requirements across the team, we were unable to implement unit tests for the application. As a result, all bugs and potential problems were found by the team and manually fixed.

Critical Evaluation

At the time of writing we believe the codebase is at an acceptable quality. We believe Visuo succeeds as a web-based spatial skills testing platform and meets our clients requirements. Below summarises the apps strengths and weaknesses,

Strengths

- App is simple, user-friendly and easy to use
- Tests are easily shareable
- Style is consistent across the app
- Ability to test spatial memory through dynamic questions

Weaknesses

- Hard to test, many combinations and edge cases, no unit tests/github actions to automate tests
- Limited statistics and analysis information
- No 'whitelist' system that will protect tests from bad actors



Future Work

Visuo was our solution to the web-based spatial skills test deliverable that our client wanted. If given the opportunity, the following are potential future features/improvements to Visuo:

Universal Bank

Currently, the app only supports manually adding questions to tests, which requires users to already have the question assets ready and prepared (images, prompts, settings). This feature would allow users to import preset questions into their own tests. Reducing the barrier to entry and the time required to create tests. All questions will be manually verified to be academically relevant.

Review Mode

Similar to what CodeRunner has, this feature would allow students to review previously taken tests. The creators of the test would determine if answers would be visible.

Marketplace

This feature would allow users to share their own custom questions/tests for other institutions to use.

- Allow users to rate tests/questions.
- Incorporate content made by others within your own test.
- Can sort by difficulty, rating, views, type

Comprehensive Statistics

Expansion on the current statistics functionality. This would include,

- Graphs indicating student score distributions
- Individual question analysis. How students performed on each question. What questions were most difficult? Is this an outlier?
- More statistics. High, low, mean, median scores, number of completions, etc.

In addition to expanding the statistics, we would like for users to have the ability to download student scores as JSON and CSV formats for further analysis on other software tools.

Whitelist/Classes

Many discussions occurred over the level of authentication required to take a test. Ultimately, we decided on an 'easy in, easy out' approach where we tried to minimise the friction between clicking on the site and taking the test. As a consequence, malicious actors can exploit Visuo by playing a test multiple times using different identifiers(names). We would be able to solve this by giving admins the option to only allow certain individuals with specific identifiers to take the test.

Themes/Dark Mode

The app currently uses a single 'light' appearance mode which could be considered bothersome, especially over long periods of time. This feature would give users the ability to switch between different appearance modes for Visuo. Dark modes are a common staple among many user interfaces due to their ease to look at and sleek design and would be an added enhancement for Visuo.

Due to time constraints and limited working knowledge of related toolsets, we were unable to achieve the optional game-based spatial skills test deliverable. Thus, another potential future project would be developing the game to fulfil the optional deliverable.



Conclusion

Project Aims

Through our project, we aimed to solve the various issues that come with traditional paper tests as mentioned previously. We have completed two deliverables; extending the existing CodeRunner platform with 29 additional spatial skill questions, and creating a fully functional web application that acts as a testing platform. Our web app, Visuo, conveniently digitalises the testing format, allowing code to automate various laborious tasks found in paper tests. It also provides dynamic memory questions, is easily accessible for anyone to use, is lightweight, and extremely customizable with a test editor. Unfortunately, we could not develop the optional game-based spatial skills test within the given timeframe. The game-based test could've solved the issue of traditional tests being unengaging and boring, which we weren't able to fully deliver on.

Key Findings

During the development of Visuo, our team experienced key findings that led to learning opportunities. Throughout development, our team did not utilise any testing plans to make sure our code worked for cases outside of our expectations. As our project intends to emulate a testing environment, our code, particularly the testing interface, had to be robust as to prevent unfair assessment. Since we did not do rigorous testing, many bugs and issues were found during the later cleanup phases, which left us with little time to solve these issues. If we had utilised test-driven development or assigned members to provide usage testing / write testing code, we could have saved time during our project cleanup sprints.

Another challenge we faced during development was the need to change our data model designs / architecture in response to new requirements. As our project increased in scope, code became increasingly harder to change and maintain, particularly for the front-end where functionality was largely dependent on API response calls. As we progressed through client meetings and gathered feedback / clarified requirements, we often had to alter our data models / code to incorporate new fields, variables etc. This results in time spent rewriting a lot of existing code, which slowed down our development process. If we had clarified the requirements and designed our data models more precisely near the beginning of the development process, we could have avoided the need to rewrite code.

When we were deciding on which Content Delivery Network (CDN) service to use, we initially used Google Drive to host our files, as it was free and easy to use. However, we found that our question assets were slow to retrieve from their service, as well as being limited in the number of times an IP address can access our files. To avoid these limitations, we decided to use AWS S3, which provided faster data load times, as well as having fewer limitations in the number of times someone can access our files.



Major Outcomes

While there is room for improvement, we believe we have successfully delivered upon our clients minimal viable product requirements for the web application, while offering even more. Visuo functions not only as a spatial skills testing platform, but as a fully functional, lightweight test creator and test facilitation platform for both students and educators. By putting our development focus onto key attributes such as accessibility, customizability and simplicity, our application is able to have unique strengths compared to other existing spatial skills testing software. As our project is well documented and built with modern, scalable technologies, future developers should be able to easily pick up and extend our project if required. Educational institutions will be able to use Visuo to easily administer and gather information about their student's spatial ability without having to go through the time consuming process of a traditional paper test.

Declaration of Authorship

Section	Contributor(s)
Title Page	Jack
Executive Summary	Jack
Introduction	Ethan
Background	Ethan, Luke
Specifications and Design	Ryan, Damon, Luke, Jack
Implementation	Ryan, Damon
Results and Evaluation	Damon
Future Work	Damon
Conclusion	Luke
References	Ryan, Luke
Appendix	Ethan



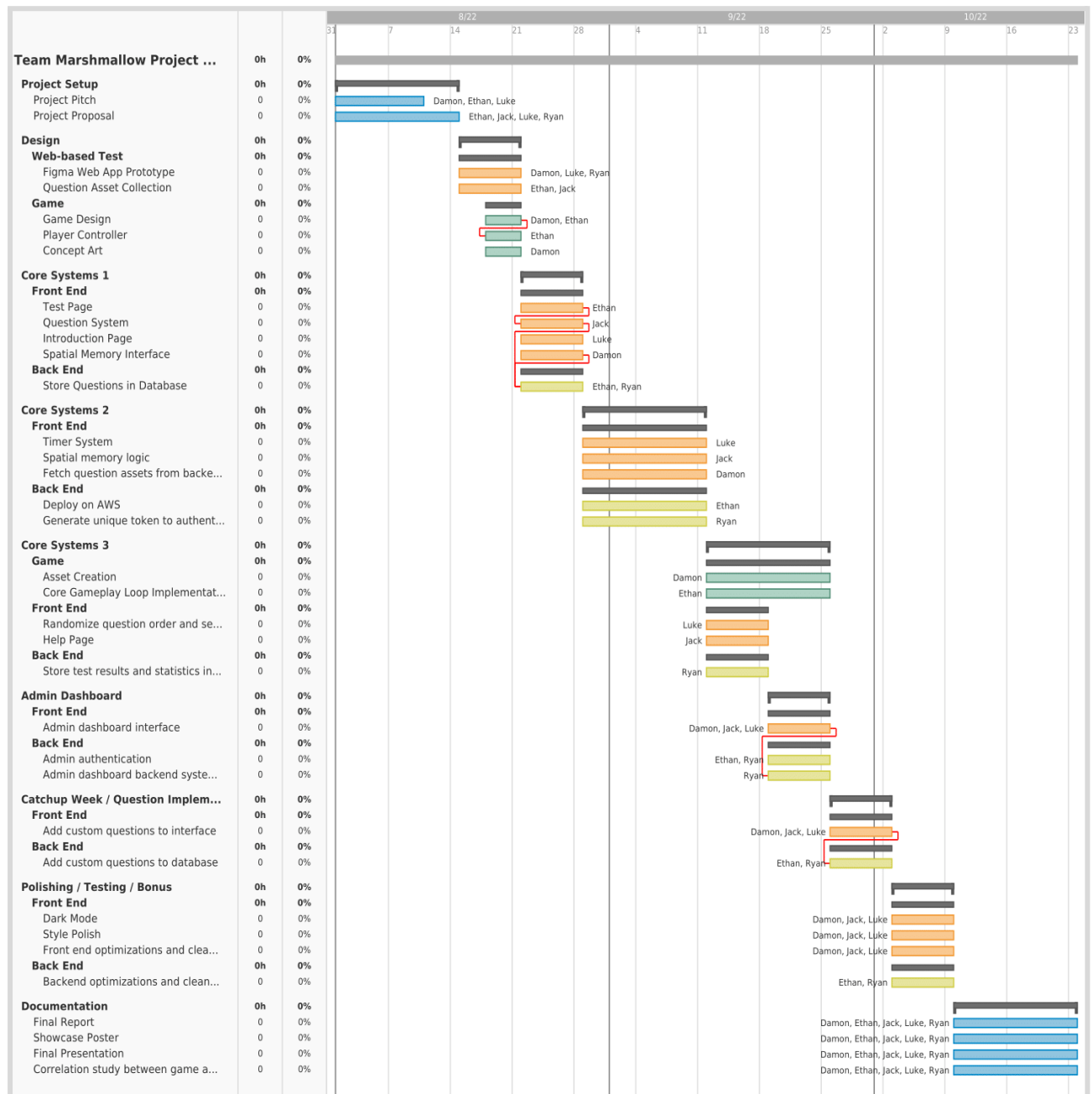
References

- [1] Colom, R., Contreras, M. J., Botella, J., & Santacreu, J. (2002). Vehicles of spatial ability. *Personality and Individual Differences*, 32(5), 903–912.
[https://doi.org/10.1016/S0191-8869\(01\)00095-2](https://doi.org/10.1016/S0191-8869(01)00095-2)
- [2] Wai, J., Lubinski, D., & Benbow, C. P. (2009). Spatial ability for STEM domains: Aligning over 50 years of cumulative psychological knowledge solidifies its importance. *Journal of Educational Psychology*, 101(4), 817–835. <https://doi.org/10.1037/a0016127>
- [3] Colom, R., Contreras, M. J., Botella, J., & Santacreu, J. (2002). Vehicles of spatial ability. *Personality and Individual Differences*, 32(5), 903–912.
[https://doi.org/10.1016/s0191-8869\(01\)00095-2](https://doi.org/10.1016/s0191-8869(01)00095-2)
- [4] Maeda, Y., Yoon, S. Y., Kang, K., Imbrie, P. K. (2013), Psychometric Properties of the Revised PSVT:R for Measuring First Year Engineering Students' Spatial Ability. *Journal of Engineering Education*.
- [5] <https://www.123test.com/spatial-reasoning-test/>
- [6] <https://www.assessmentday.com/spatial-reasoning.htm>
- [7] <https://www.ibm.com/cloud/learn/three-tier-architecture>
- [8] <https://axios-http.com/>
- [9] <https://betterprogramming.pub/deploy-mern-stack-app-on-aws-ec2-with-letsencrypt-ssl-8f463c01502a>
- [10] <https://towardsdev.com/deploying-a-react-node-mysql-app-to-aws-ec2-2022-1dfc98496acf>
- [11] <https://gist.github.com/bradtraversy/b8b72581ddc940e0a41e0bc09172d91b>
- [12] <https://www.linode.com/docs/guides/enabling-https-using-certbot-with-nginx-on-ubuntu/>
- [13] Corsi Block Tapping Test
<https://www.sciencedirect.com/science/article/pii/S2451958821000476>



Appendix

Gantt Chart



Final Build

GitHub repository: <https://github.com/uoa-compsci399-s2-2022/visuo>

Project management can also be found under the issues section on the GitHub repo.

Website deployment: <https://hydrohomiebeerbro.com/>

Project Demonstration Video

https://www.youtube.com/watch?v=hNazCZB1zus&ab_channel=EthanBland