CSCI 370 Final Report

The Theis Four

David Karibian
Colton Morris
Ava Moon
Iris Nemechek

Revised November 20, 2024

CSCI 370 Fall 2024

Mr. Tree

Table 1: Revision history

Revision	Date	Comments
New	August 21, 2024	Completed Sections: I. Introduction II. Functional Requirements III. Non-functional Requirements IV. Risks V. Definition of Done VI. System Architecture References Appendix A – Key Terms
Rev – 2	August 27, 2024	Updated functional and non-functional requirements to include information about end-to-end and unit testing.
Rev – 3	August 27, 2024	Added sliding functionality and 3D cone requirements.
Rev - 4	September 9, 2024	Added design documents to system architecture
Rev - 5	September 11, 2024	Wrote system architecture technical problems and overview
Rev - 6	October 8, 2024	Wrote initial drafts of Software Quality and Ethical Consideration sections VII. Software Test and Quality VIII. Project Ethical Considerations
Rev - 7	October 15, 2024	Added ethical considerations to the document.
Rev - 8	November 1, 2024	Added results to document.
Rev - 9	November 10, 2024	Added team bios, acknowledgments, and finalized references.
Rev -10	November 20, 2024	Made edits and reviewed as a team.

Table of Contents

I. Introduction

The Groundwater Project was created in 2017 to enhance prospects for the expansion of knowledge about groundwater problem-solving for the public [1]. The nonprofit advances groundwater understanding by creating learning materials that are free and easily accessible to the public. The primary goal of this field session project is to create an online tool to demonstrate the depletion of flow from a fully penetrating stream in response to a pumping well. The tool will interact with the user, taking inputs, performing calculations, and displaying data in the form of graphs and heat maps. The tool will be an addition to the currently deployed tools accessible through the Groundwater Project and demonstrate a Depletion Fully Penetrating System (DFPS) and help aid in groundwater education.

II. Functional Requirements

1. The program must be able to take in a variety of inputs from the user via an input box feature that allows values from 10 below to 10 above-provided test input including:
    a. Spatial dimensions:
        i. The perpendicular distance from the well to the stream
        ii. The factor for the size of the domain to be displayed
    b. Aquifer parameters [2]:
        i. Hydraulic conductivity
        ii. Aquifer thickness
        iii. Specific yield
    c. Stream parameters
        i. Hydraulic conductivity of stream bed
        ii. Stream flow rate
    d. Well information
        i. Pumping rate
        ii. Locations of observations for the well (to be graphed as a function of time)
    e. Time information

        i. Duration of pumping
        ii. Time step
2. From the inputs, the program shall calculate and validate calculations with unit testing:
    a. Time increments
    b. Fraction of pumped water from a stream
    c. Rate of leakage from the stream (meters cubed per day)
    d. Stream discharge (meters cubed per second)
    e. Drawdown at each grid point for each time increment
    f. Velocity vectors
3. From the calculation results, the program shall create and display:
    a. Graph simulating the aquifer as water is being pumped out.
4. The following graphs shall supplement the main visualization:
    a. Contour map of the drawdown at time t
    b. WEST-EAST cross-section at time t
    c. Fraction of pumping leaking from the stream
    d. SOUTH-NORTH section near a stream at time t
    e. Stream from rate over time
    f. Observation of well 1, well 2 over time (days)
5. The program shall implement error-catching for unaccepted inputs
    a. Error messages shall give users information on why input is unacceptable
6. The intended function of the website shall be ensured using end-to-end testing
    a. Scripts created acting as user clicking buttons, functionality (correct output) ensured

III. Non-Functional Requirements

1. Write Groundwater DFPS project in HTML and Javascript.
2. Groundwater DFPS project must use the VSCode development environment.
3. Groundwater DFPS project must be tested on a local host.
4. Groundwater DFPS project must be deployed as a web page accessible to the public and part of the already existing Groundwater Project website.
5. Pass tests of the Groundwater DFPS project with inputs designed to ‘break’ the program.
6. Design Groundwater DFPS project with the inexperienced user in mind.
7. Use Jest for unit testing [5].
8. Use Selenium for end-to-end testing [5].

IV. Risks

1. Technical risks
    a. May be too technically involved for the browser to perform computation and visualizations in an adequate amount of time.
        i. This is a low risk as we can use optimization techniques, and perhaps even a server in order to offload this computing power.
        ii. This will have a mild impact on the project because we can implement a loading screen to mitigate effects.
    b. May need help to consistently scale the graphs to show the data as it is generated via the timestamps.
        i. This will increase the complexity of the program but does not pose a threat as it is mostly a consideration that will simply be handled.
    c. May be difficult to implement complex calculations in JavaScript.
2. Skill risks
    a. Team members do not have in-depth experience with JavaScript or HTML.
        i. The team members will individually review the Groundwater HTML and JS codebase, alongside watching tutorials and self educating to understand these technologies at a development level.

b. Team members are not familiar with the geological concepts of this project.
    i. The team members were educated by Eileen Porter (Client) who provided resources and talked us through many of the technical and conceptual groundwater ideas.

V. Definition of Done

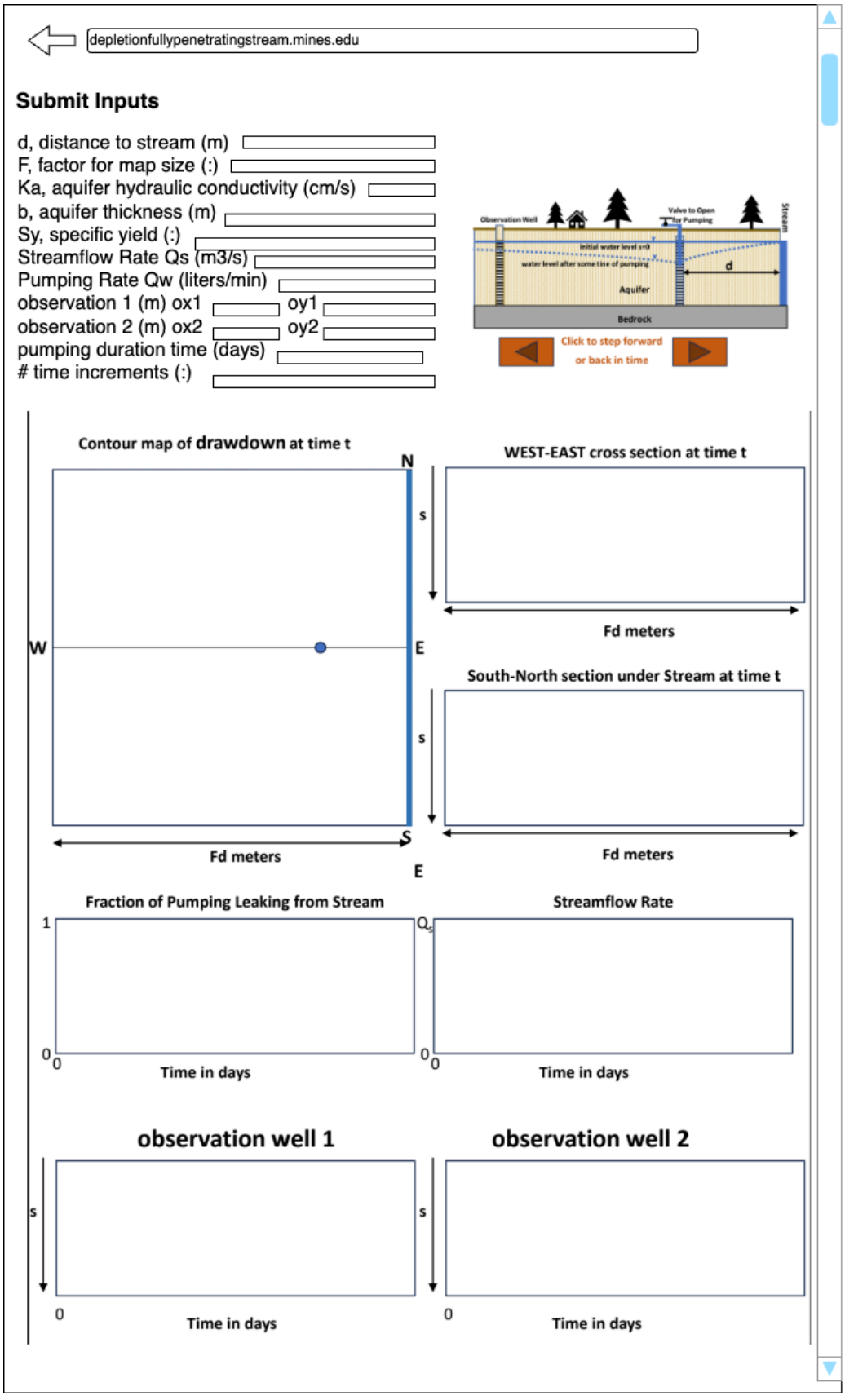
1. Minimal Useful Features
    a. Input Form:
        i. A web-based form allowing users to input
        ii. The perpendicular distance from the well to the stream
        iii. Aquifer parameters (Hydraulic conductivity, Aquifer thickness, Specific Yield)
        iv. Stream parameters (Hydraulic conductivity of stream bed, Stream flow rate)
        v. Well information (Location, Pumping rate, Observation well locations)
        vi. Time information (Duration of pumping, Number of increments)
    b. Calculation Engine:
        i. Calculate drawdown at various points within the defined spatial domain
        ii. Calculate the fraction of pumped water coming from the stream
        iii. Calculate the rate of leakage from the stream
        iv. Generate time-based contour maps and cross-sectional graphs
    c. Graphical Outputs:
        i. Visual representations of
        ii. Contour maps of drawdown over time
        iii. 3D cone representing drawdown over time
        iv. Cross-sectional graphs showing drawdown along specific axes
        v. Velocity vectors displayed on the grid
    d. Error Handling:
        i. Input validation and error messages for incorrect or nonsensical inputs
        ii. Graceful handling of out-of-bounds or undefined results
2. Client Acceptance Tests
    a. Input a variety of realistic data sets to test and verify that the outputs are accurate and consistent
    b. Test edge cases with invalid inputs to ensure error handling and feedback
    c. Evaluate user interface for ease of use and responsiveness of form
3. Product Delivery
    a. The final version will be deployed on the Groundwater Project's website
    b. The codebase will be shared with the client via GitHub
    c. The documentation will be shared along with the codebase on GitHub

VI. System Architecture

1. Technical Issues
    a. Our team is currently not experiencing any technical issues because of our simplified tech stack of only creating a front end.
2. Technical Stack
    a. We plan on only creating a front end to accommodate our client's wishes. This means we will primarily be using JavaScript, HTML, and CSS. We also plan on using Jest for JavaScript unit testing and Selenium for end-to-end testing. We chose Jest because it's easy to use with JavaScript, and Selenium because it helps test user interactions. We are also using EsLint as a linter to help us adhere to better coding standards [9]. We used Plotly to graph all of the plots for our visualizations.
3. Design Documents
    a. LaTex Math Equations

i. These pictures of our overleaf document contain the important input parameters, math equations, and an overview of output graphs pertinent to our minimum viable product. The main purpose of this document is to help us break down these equations while developing so that we can understand them at a base level and build them from the ground up. It will also assist us in initially coding the inputs so we can create groupings for certain equations. For the output graphs, we just put a brief overview because most of the work that will be going into those will be handled by a graphing library of our discretion.

b. Wire Frame



c.

i. The wireframe, also known as the user storyboard, contains three important elements, (1) the input section, (2) the water simulation, and (3) the graphs. These are the core graphical components of the application, which in conjunction provide a highly educational experience for

the user. The user flow will work as such. The user first will input a specific parameter. They will be able to step through the simulation and see what happens to the well given their input data. The graphs will help them to understand the math and equations behind this simulation.

1 Input Parameters

The following input parameters are required for the tool:

1.1 Spatial Dimensions

- Perpendicular distance from the well to the stream, d (in meters)
- Size factor for the display domain, F (dimensionless)

1.2 Aquifer Parameters

- Hydraulic conductivity, K_a (in cm/s, converted to m/day)
- Aquifer thickness, b (in meters)
- Specific yield, S_y (dimensionless)

Note: Transmissivity T is used in calculations and is given by:

$$T = K_a \cdot b$$

1.3 Stream Parameters

- Hydraulic conductivity of stream bed, K_{sb} (in cm/s, converted to m/day)
- Stream flow rate, Q_s (in m^3/s)

1.4 Well Information

- Well location defined as $x = 0, y = 0$
- Pumping rate, Q_w (in liters per minute, converted to m^3/s)
- Locations of observation wells for graphing drawdown, (ox_1, oy_1) and (ox_2, oy_2) (in meters relative to the well)

1.5 Time Information

- Duration of pumping, t (in days)
- Number of time increments for calculations

2 Mathematical Equations

The following equations govern the behavior of the aquifer system and stream depletion:

2.1 Stream Depletion Fraction Over Time

The fraction of the pumped water coming from the stream (i.e., stream depletion) at time t is given by:

$$Q_{\text{fraction}} = \text{erfc}\left(\sqrt{\frac{S_y \cdot d^2}{4T \cdot t}}\right)$$

Where:

- Q_{fraction} = Fraction of pumping rate coming from the stream (dimensionless)
- S_y = Specific yield (dimensionless)
- d = Distance from well to stream (in meters)
- T = Transmissivity, $T = K_a \cdot b$ (in m^2/day)
- t = Time since pumping began (in days)
- erfc = Complementary error function

2.2 Stream Leakage Rate at Time t

The rate of leakage from the stream is calculated as:

$$Q_{\text{stream leakage at } t} = Q_{\text{fraction}} \cdot Q_w$$

Where:

- Q_w = Pumping rate (converted to m^3/s)
- Q_{fraction} = Fraction of pumping rate coming from the stream

2.3 Stream Discharge at Time t

The stream discharge at time t is given by:

$$Q_{s@t} = Q_s - Q_{\text{stream leakage at } t}$$

Where:

- Q_s = Initial volumetric discharge of the stream (in m^3/s)
- $Q_{\text{stream leakage at } t}$ = Stream leakage rate at time t

2.4 Drawdown Calculation

For each time increment and at each grid point, determine the straight-line distance r to the well:

$$r = \sqrt{(x_{\text{grid}} - x_{\text{well}})^2 + (y_{\text{grid}} - y_{\text{well}})^2}$$

Calculate the drawdown at each grid point using:

$$s(x, y, t) = \frac{Q_w}{4\pi T} \left(W(u) - W(u')\right)$$

Where:

- $s(x, y, t)$ = Drawdown at a location (x, y) and time t
- Q_w = Constant well discharge rate (in m^3/s)
- T = Transmissivity (in m^2/day)
- $W(u)$ = Well function for u
- $u = \frac{r^2 S_y}{4Tt}$, $u' = \frac{(2d-r)^2 S_y}{4Tt}$
- r = Distance from the well

2.5 Grid and Velocity Calculations

For each time increment:

- Divide the map into a grid with 21 x 21 points.
- At each point, calculate the distance r to the well and drawdown using the above formula.
- Velocity vectors are calculated using Darcy's Law:

$$v = \frac{(h_{\max} - h_{\min}) \cdot K_a}{0.02 \cdot L}$$

  Where L is the grid cell side length.

3 Graphical Output

For each time increment:

- A contour map of drawdown within the specified domain will be displayed.
- Locations of the well and observation wells will be marked.
- Streamflow and leakage graphs will be updated dynamically.

VII. Software Test and Quality

1. End-To-End Testing
    a. For this project, we will be utilizing end-to-end testing (E2E) to ensure the reliability and performance of the code. End-to-end testing will allow us to verify that all aspects of the user experience are working as intended, which is crucial given the UI-focused nature of the project. End-to-end testing provides holistic validation by testing the entire application as a whole, not just isolated components. Using Selenium, we can connect to a browser and run scripts that simulate real user interactions, such as clicking buttons or entering inputs. This ensures that when a user performs certain actions, the program responds accurately, including proper navigation, correct data handling, and appropriate feedback to the user. By complementing end-to-end testing, we cover user experience, which ensures that the program as a whole operates smoothly in real-world scenarios.
2. Unit Testing
    a. We will use Jest testing for JavaScript unit tests to ensure the correctness of calculations and various function behaviors. The testing strategy focuses on validating both normal and edge-case scenarios. JUnit testing is beneficial because it can catch bugs early, ensure that code behaves as expected, and provide a safety net when making changes, reducing the likelihood of regressions.
3. Code Reviews
    a. The team has established a set of requirements for merging branches into the project to maintain high code quality and adherence to standards. Before a pull request can be published on GitHub, the following conditions must be met:
        i. The code must pass a linting check using ESLint.
        ii. All tests must pass successfully.
        iii. The code must meet the acceptance criteria specified in the corresponding Jira ticket.
    b. Once these prerequisites are satisfied, the pull request can be submitted for review. To continue with the merge, approval from at least two out of three other team members is required. This process not only ensures that the code aligns with the quality standards set by both the client and the team but also fosters collaboration and accountability among team members.
4. Client Acceptance Testing
    a. The team has started with a preparation phase.
        i. The client will become equipped with a functional testing environment that mirrors our own, following instructions detailed in our README.
        ii. We will provide a brief training session, if necessary, to ensure our client is comfortable using our testing software.
    b. Once the client is set up with the software, we will begin the execution phase.
        i. The client will perform the scripted E2E tests on their own device.
        ii. The results of these tests will be logged in a spreadsheet for accountability.
        iii. The client will also be able to use the website as a user to see if the UI meets their standards.

c. Once tests have been executed, we will enter the review phase.
    i. If any issues are identified, the team will address them immediately, followed by retesting to ensure the problem is resolved. The client may repeat tests as necessary to confirm that the software now behaves as expected.
    ii. Once the client is satisfied that all tests have passed successfully and the software is functioning correctly, they will provide formal feedback, which includes any final thoughts or requests.
d. Finally, when the client is satisfied, we will enter the sign-off and deployment phase.
    i. After all tests pass and the client confirms that the software meets the required standards, they will provide formal approval via a sign-off document. This document will indicate that the client accepts the system as ready for deployment.
    ii. Upon receiving the client's sign-off, the project will be ready for deployment to the official Groundwater website.

VIII. Project Ethical Considerations

1. Relevant ACM/IEEE Principle
    a. Principle 1: Public Interest
        i. This principle emphasizes that computing professionals should contribute to society and human well-being. The Groundwater Project aims to enhance public understanding of groundwater issues through accessible educational tools, aligning with this principle [1].
    b. Principle 3: Professional Responsibility
        i. Developers must ensure that their work is of high quality and meets the expectations of users. Given the project's educational focus, maintaining high-quality standards is essential to provide reliable information and tools to users [7].
    c. Principle 4: Quality Products
        i. This principle highlights the need for software products to be reliable and defects-free. The software quality plan, including unit testing and end-to-end testing, aims to ensure that the final product meets these quality standards.
2. Principles Most at Risk of Violation
    a. Principle 2: Avoiding Harm
        i. If the software contains errors or unreliable information, it could misinform users about groundwater management, potentially leading to environmental harm or poor decision-making.
    b. Principle 5: Respect for Privacy
        i. We need to ensure that any user input processed is handled safely. If user data is collected without proper safeguards, there is a risk of privacy violations. This could result in unauthorized access to sensitive information.
3. Application of Michael Davis's Tests
    a. The Test of Harm:
        i. Will the use of the software potentially cause harm to users or the environment? The Groundwater Project must ensure that the calculations and visualizations it provides do not misrepresent data or lead to poor groundwater management practices. If miscalculations occur, users could make harmful decisions regarding water usage or conservation.
    b. The Test of Professional Standards:
        i. Does the product meet the standards of the profession? Given that the project aims to provide educational resources, it must adhere to the highest standards of accuracy and reliability in its calculations and outputs. If the software fails to meet professional standards, it undermines the educational purpose of the project.
4. Ethical Considerations of Inadequate Software Quality Plan
    a. If the software quality plan is not implemented properly or is insufficient, several ethical concerns may arise:
        i. Deficient Product Quality: Users may receive inaccurate information, which could lead to harmful environmental or public health decisions.

ii. Erosion of Trust: If users encounter frequent errors or unclear outputs, they may lose trust in the Groundwater Project and its mission, undermining its goal of education and awareness.
iii. Legal and Regulatory Risks: Inaccurate representations of groundwater data could lead to compliance issues with environmental regulations, resulting in legal action against the organization.
iv. User Exploitation: Without proper validation and error handling, inexperienced users may be exploited due to misinformation, leading to financial or resource-related consequences.

IX. Project Completion Status

Our project aimed to develop an educational tool to visualize the depletion of fully penetrating streams for the Groundwater Organization. The project is nearly complete, with all primary components developed, tested, and documented. Key achievements include designing and implementing input handling, calculation engines, and comprehensive graphical outputs for simulating groundwater depletion. The codebase has undergone rigorous unit and end-to-end testing using Jest and Selenium to ensure both functionality and reliability, addressing user interface interactions and calculation accuracy. Additionally, the team has completed thorough documentation, covering functional requirements, testing processes, and ethical considerations, and has conducted client acceptance tests to validate the system's performance and user experience. The only remaining step is to deploy the final version of the tool to the official Groundwater Project website, marking the final phase of project completion. Once deployed, the tool will be accessible to the public, supporting the project's mission to enhance understanding of groundwater management.

Selenium End-to-end Tests:

Test Name	Description	Goal	Result
testMenuNavigation	Tests menu navigation	Navigate through GUI	Pass
testGraphsDisplay	Tests graph display	Check if the graphs load	Pass
testGraphLayout	Tests layout of graphs	Ensure the layout is sufficient for ~8 graphs	Pass

Jest Unit Tests:

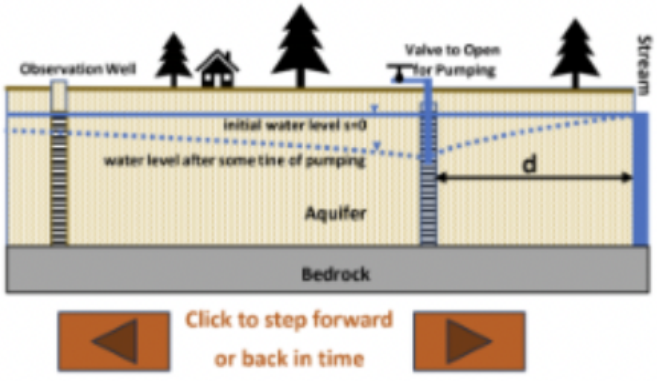
Test Name	Description	Goal	Result
calculateLogarithmicTimeSteps - Test 1	Logarithmic time steps	Validate calculation accuracy	Pass
calculateLogarithmicTimeSteps - Test 2	Logarithmic time steps	Validate calculation accuracy	Pass
calculateLogarithmicTimeSteps - Test 3	Logarithmic time steps	Validate calculation accuracy	Pass
erfc Function - Test 1	Complementary error function	Check erfc at 0	Pass

erfc Function - Test 2	Complementary error function	Check erfc at 1	Pass
erfc Function - Test 3	Complementary error function	Check erfc at -1	Pass
erfc Function - Test 4	Complementary error function	Check erfc at 2	Pass
calculateQFraction - Test 1	Q fraction calculation	Check Q fraction value	Pass
calculateQFraction - Test 2	Q fraction calculation	Qfraction with d=0	Pass
calculateQFraction - Test 3	Q fraction calculation	Qfraction with large d	Pass
calculateQFraction - Test 4	Q fraction calculation	Throw error on zero time	Pass
calculateQFraction - Test 5	Q fraction calculation	Throw error on negative time	Pass
calculateDistance - Test 1	Distance calculation	Check distance (0,0)-(3,4)	Pass
calculateDistance - Test 2	Distance calculation	Check distance (1,1)-(4,5)	Pass
calculateDistance - Test 3	Distance calculation	Check distance (-2,-3)-(4,1)	Pass
calculateDistance - Test 4	Distance calculation	Grid and well same point	Pass
calculateDrawdown - Test 1	Drawdown calculation	W(u)-W(u')=0 returns 0	Pass
calculateDrawdown - Test 2	Drawdown calculation	Different u and u' values	Pass
calculateDrawdown - Test 3	Drawdown calculation	r=0 returns correct value	Pass
calculateDrawdown - Test 4	Drawdown calculation	Large u and u' approach 0	Pass
calculateDrawdown - Test 5	Drawdown calculation	Throw error on zero time	Pass
calculateDrawdown - Test 6	Drawdown calculation	Throw error on negative time	Pass
calculateStreamLeakage - Test 1	Stream leakage calc	Valid input calculation	Pass
calculateStreamLeakage - Test 2	Stream leakage calc	Zero pumping rate	Pass
calculateStreamDischarge - Test 1	Stream discharge calc	Correct discharge calculation	Pass

calculateStreamDischarge - Test 2	Stream discharge calc	Zero leakage case	Pass
Stream Flow Integration - Test	Stream depletion over time	Validate stream flow model	Pass
calculateLogarithmicTimeSteps - Negative Time	Negative total time	Handle negative input	Pass
calculateLogarithmicTimeSteps - Zero Increments	Zero increments	Handle zero increments	Pass
calculateLogarithmicTimeSteps - Large Increments	Large increments	Validate performance on large input	Pass
calculateQFraction - Negative Sy	Negative specific yield	Check for error on negative Sy	Pass
calculateQFraction - High Specific Yield	High specific yield	Handle high Sy values	Pass
calculateDistance - Non-numeric Input	Invalid non-numeric input	Handle non-numeric values	Pass
calculateDistance - Large Coordinates	Large coordinate values	Manage high-value inputs	Pass
calculateStreamLeakage - Negative Qw	Negative pumping rate	Ensure error on negative Qw	Pass
calculateStreamDischarge - Zero Initial Qs	Zero initial discharge	Handle zero discharge gracefully	Pass

X. Future Work

Future development will focus on creating a dynamic visualization tool to represent key hydrological parameters in real time. This tool will depict fluctuating water levels, the baseline (initial) water level, and the spatial relationships between critical features such as the stream, valve, and pumping opening, along with the aquifer's depth profile [10]. This feature will respond interactively as users input various parameters and adjust time steps, allowing for real-time updates to the graphical representation. Graphs will be synchronized with interactive control elements (such as the orange buttons) to facilitate the display of gradient changes over time. This synchronization is essential for ensuring a seamless user experience, where the gradient graph reflects the evolving dynamics accurately.



XI. Lessons Learned

1. **The Importance of Simplified Technology Stacks**: Our choice to use only a front-end stack (JavaScript, HTML, and CSS) streamlined development, allowing us to focus on the UI and calculations without the complexity of back-end integration. This decision minimized technical issues and made debugging more manageable.
2. **Value of Testing and Validation**: Implementing unit tests with Jest and end-to-end tests with Selenium was essential for ensuring that our application met functional requirements. Early testing allowed us to identify and resolve issues quickly, ultimately improving the reliability of the tool. It saved us a great amount of time by implementing these tests before developing because we knew that everything was functional as we developed more, so no parts of the code were built on faulty functionality.
3. **Effective Use of Project Management and Collaboration Tools**: Utilizing code review processes on GitHub and maintaining coding standards with ESLint helped us maintain quality across the codebase and avoid merge conflicts. This setup fostered collaboration and kept our code organized, allowing for smooth teamwork and version control.
4. **Challenges of Working in Unfamiliar Domains**: Developing an educational groundwater tool required learning domain-specific knowledge about aquifer behavior and water flow. Researching and consulting resources to understand these concepts was necessary to ensure the accuracy of our calculations and visualizations.
5. **Adaptability of Agile Development**: Applying an iterative approach helped us build up functionality progressively, from basic input handling to complex visualizations.
6. **Deploying to Production Environments**: Preparing the final product for deployment on the official Groundwater Project website posed unique challenges, especially in configuring files and ensuring seamless integration. This phase showed the importance of deployment testing and comprehensive documentation to support smooth delivery.

XII. Acknowledgments

We want to extend our sincere gratitude to our advisor, Professor Tree, and to our client, Eileen Poeter, for their invaluable contributions throughout this project. To our advisor, Professor Tree, thank you for your detailed review of our document and your thoughtful feedback, which greatly improved the clarity and focus of our work. We also deeply appreciate your careful consideration of our questions and your flexibility in working with us as we navigated the complexities of this project. To our client, Eileen Poeter, we are incredibly grateful for your commitment to ensuring our design documentation was thorough and precise. Your clear communication and willingness to explain technical details—often more than once—helped us understand and address our challenges. Your guidance and support have been instrumental in shaping this design, and we truly appreciate the time and effort you both dedicated to its success.

XIII. Team Profile

Colton Morris

Colton is a senior studying computer science. He is interested in machine learning and software development and is the group's client liaison.

David Karibian

David is a junior studying computer science with an interest in quantum finance and business. He is the advisor liaison of the group.

Ava Moon

Ava is a junior majoring in computer science with an interest in cybersecurity and consulting. She is the scrum master of the group.

Iris Nemechek

Iris is a junior majoring in computer science with an interest in cybersecurity and software engineering.

References

1. The Groundwater Project. [Online]. Available: https://www.gw-project.org. [Accessed: Oct. 8, 2024].
2. U.S. Geological Survey, "Aquifers and groundwater," [Online]. Available: https://www.usgs.gov/special-topic/water-science-school/science/aquifers-and-groundwater. [Accessed: Oct. 8, 2024].

3. Jest Documentation. "JavaScript testing framework." [Online]. Available: https://jestjs.io/docs/getting-started. [Accessed: Oct. 8, 2024].
4. Selenium Documentation. [Online]. Available: https://www.selenium.dev/documentation/. [Accessed: Oct. 8, 2024].
5. Guru99, "Software testing basics," [Online]. Available: https://www.guru99.com/software-testing-introduction.html. [Accessed: Oct. 8, 2024].
6. National Groundwater Association, "Understanding aquifer depletion," [Online]. Available: https://www.ngwa.org/aquifer-depletion. [Accessed: Oct. 8, 2024].
7. International Association of Hydrogeologists, "Stream-aquifer interactions," [Online]. Available: https://www.iah.org/stream-aquifer. [Accessed: Oct. 8, 2024].
8. Stanford University, "Hydraulic conductivity Logging Glacial Aquifers," [Online]. Available: https://gemcenter.stanford.edu/estimating-hydraulic-conductivity-nmr-logging-glacial-aquifers. [Accessed: Oct. 8, 2024].
9. GitHub, "Using ESLint for coding standards," [Online]. Available: https://eslint.org/docs/latest/user-guide. [Accessed: Oct. 8, 2024].
10. Environmental Protection Agency, "Groundwater Modeling Researching," [Online]. Available: https://www.epa.gov/land-research/ground-water-modeling-research. [Accessed: Oct. 8, 2024].

Appendix A – Key Terms

Term	Definition
Depletion Fully Penetrating System (DFPS)	*A system that models the impact of water pumping from an aquifer on fully penetrating streams, showing the resulting drawdown and flow changes.*
Hydraulic Conductivity	*A measure of a material's ability to transmit water, typically expressed in meters per day.*
Aquifer	*A body of permeable rock or sediment that holds and transmits groundwater.*
Streambed Conductivity	*A parameter representing the ease with which water flows through the streambed material into or out of the stream.*
Jest	*A JavaScript testing framework for unit testing to validate the correctness of individual functions and modules.*
Selenium	*A software testing framework used for end-to-end testing of web applications by simulating user interactions.*
E2E Testing	*A testing methodology that validates the functionality and performance of an entire application by simulating real-world user interactions from start to finish. This ensures that all integrated components of the system work together as expected.*
ESLint	*A static code analysis tool for identifying and fixing problems in JavaScript code. It enforces coding standards and improves code quality by identifying errors, style issues, and potential bugs before runtime.*
Drawdown	*The lowering of the water table or piezometric surface caused by groundwater pumping.*
Contour Map	*A two-dimensional graphical representation showing lines of constant drawdown values across the modeled aquifer.*