**Sea Water Quality Awareness (SeaClear)**

**Table of Contents**

**Executive Summary ................................................................................................ i**

**1. Introduction ....................................................................................................... 1**

**1.1 Context and Problem Statement .................................................................. 1**

**1.2 Project Scope .............................................................................................. 2**

**1.3 Approach ..................................................................................................... 3**

**2. Requirements Captured ..................................................................................... 4**

**2.1 Functional Requirements ............................................................................. 4**

**2.2 Non-Functional Requirements ..................................................................... 5**

**2.3 Usability Requirements ................................................................................ 6**

**2.4 Use Case Narratives and Diagrams ............................................................. 7**

**3. Design Overview ................................................................................................ 9**

**3.1 System Architecture ..................................................................................... 9**

**3.2 Design Class Diagram ................................................................................ 11**

**3.3 Data Organization ...................................................................................... 12**

**3.4 Algorithms .................................................................................................. 13**

**4. Implementation ................................................................................................. 15**

**4.1 Data Structures .......................................................................................... 15**

**4.2 User Interface ............................................................................................ 16**

**4.3 Key Classes and Methods ......................................................................... 18**

**4.4 Special Programming Techniques or Libraries ........................................... 20**

**5. Program Validation and Verification .................................................................. 22**

**5.1 Quality Management Plan .......................................................................... 22**

**5.2 Testing Methodology .................................................................................. 24**

**5.3 Test Results and Analysis .......................................................................... 26**

**6. Conclusion ....................................................................................................... 28**

**7. References ....................................................................................................... 30**

**Appendix A: User Manual ..................................................................................... 32**

**Appendix B: Test Results ...................................................................................... 40**

**Executive Summary**

**[Your executive summary goes here - max 200 words]**

**Abstract**SeaClear is an innovative mobile-friendly website designed to address the critical issue of beach water quality awareness in Cape Town. Despite the city's stunning coastline, many beaches suffer from poor water quality, posing significant health risks to unsuspecting beachgoers. The project aims to bridge the information gap between official water quality data and public awareness by providing real-time, easily accessible information to Capetonians and tourists alike. Utilizing an agile development approach, SeaClear aggregates and presents water quality data from the City of Cape Town in an intuitive, user-friendly interface. The system features real-time displays of water quality status, an interactive map, historical data visualization, and push notifications for significant changes. Built with React.js for the frontend and Node.js for the backend, the architecture ensures responsiveness across devices and scalability for future growth. Rigorous testing, including unit, integration, and user acceptance testing, ensures the reliability and effectiveness of the system. By empowering users with timely, comprehensible water quality information, SeaClear aims to promote safer beach experiences and raise awareness about environmental issues affecting Cape Town's iconic coastline.

**1. Introduction**

**1.1 Context and Problem Statement**

Cape Town, renowned for its stunning coastline and beautiful beaches, attracts thousands of residents and tourists to its shores daily. However, a significant issue lurks beneath the surface – poor water quality at many beaches, which poses potential health risks to unsuspecting beachgoers. Despite the importance of this information, many Capetonians remain unaware of the water quality status at their favorite beaches, largely due to the lack of easily accessible, up-to-date information.

The City of Cape Town regularly collects and publishes water quality data, but this information often doesn't reach the general public in a timely or understandable manner. This gap in communication leaves beachgoers vulnerable to health hazards associated with polluted water, such as gastrointestinal illnesses, skin infections, and other waterborne diseases.

**1.2 Project Scope**

SeaClear aims to bridge this information gap by developing a mobile-friendly website that provides real-time water quality information for Cape Town's beaches. The project's primary objectives include:

1. Aggregating and presenting water quality data from the City of Cape Town in an easily understandable format.

2. Developing an intuitive, user-friendly interface accessible on both desktop and mobile devices.

3. Implementing features to help users make informed decisions about their beach activities, such as water quality ratings, trend analysis, and alerts for poor water conditions.

4. Raising awareness about the importance of water quality and its impact on public health and the environment.

**1.3 Approach**

To address this critical need, we adopted an agile development approach, allowing for iterative development and continuous user feedback. Our process included:

1. Initial research and requirements gathering through surveys and interviews with potential users.

2. Rapid prototyping and user testing to refine the interface and feature set.

3. Iterative development cycles, with regular stakeholder reviews and adjustments.

4. Continuous integration and deployment to ensure timely updates and bug fixes.

5. Collaboration with local environmental groups and the City of Cape Town to ensure data accuracy and relevance.

This approach enabled us to create a solution that not only meets the technical requirements but also addresses the real needs of Cape Town's beachgoers.

**2. Requirements Captured**

**2.1 Functional Requirements**

- Real-time display of water quality data for each beach

- Search functionality for specific beaches

- Interactive map view of beach locations with water quality indicators

- Push notifications for significant changes in water quality

- Historical data visualization for trend analysis

- User-friendly rating system for overall water quality (e.g., color-coded or numerical scale)

- Integration with social media platforms for easy sharing of beach conditions

**2.2 Non-Functional Requirements**

- Mobile-responsive design optimized for various screen sizes

- Fast loading times (< 3 seconds) even on slower mobile connections

- High availability (99.9% uptime) to ensure constant access to critical information

- Secure data handling and storage, complying with relevant data protection regulations

- Scalability to accommodate increasing user base and data volume

- Cross-browser compatibility (Chrome, Firefox, Safari, Edge)

**2.3 Usability Requirements**

- Intuitive navigation with minimal learning curve

- Clear and concise presentation of water quality status

- Accessibility features for users with disabilities (e.g., screen reader compatibility, color contrast options)

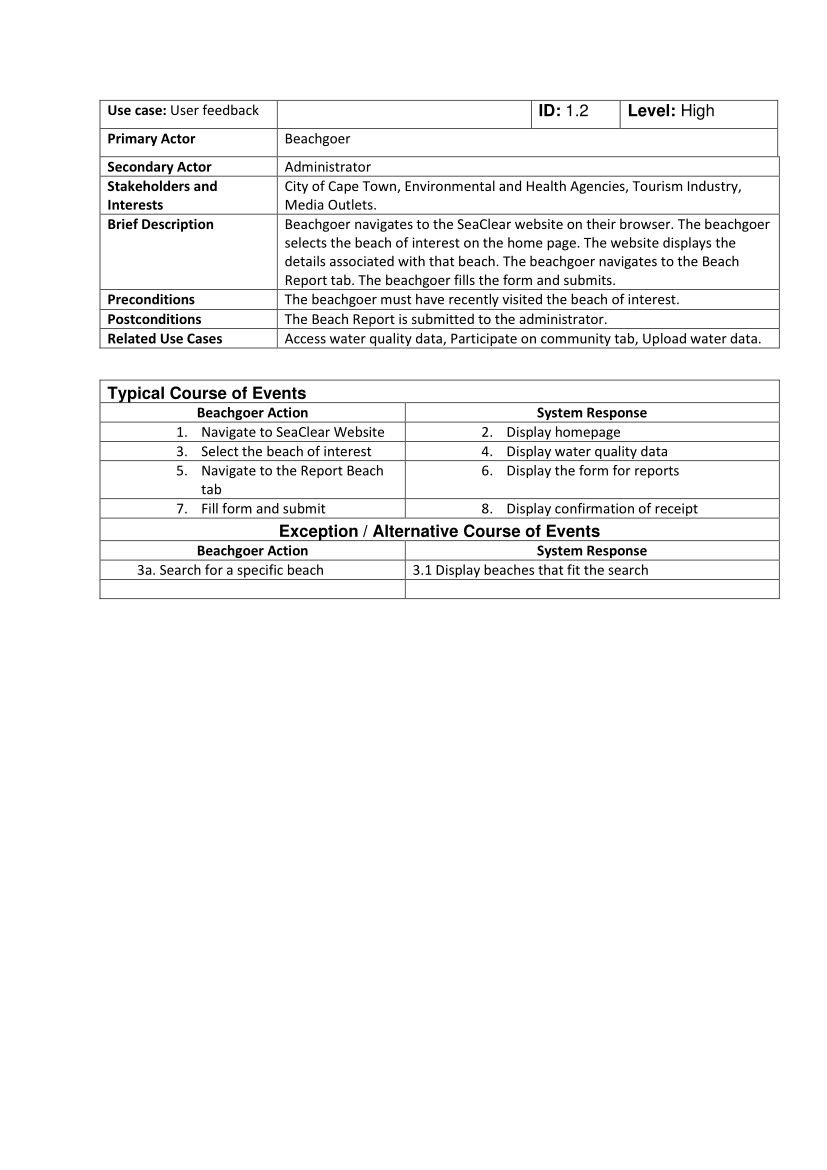
- Multi-language support (English, Afrikaans, Xhosa)

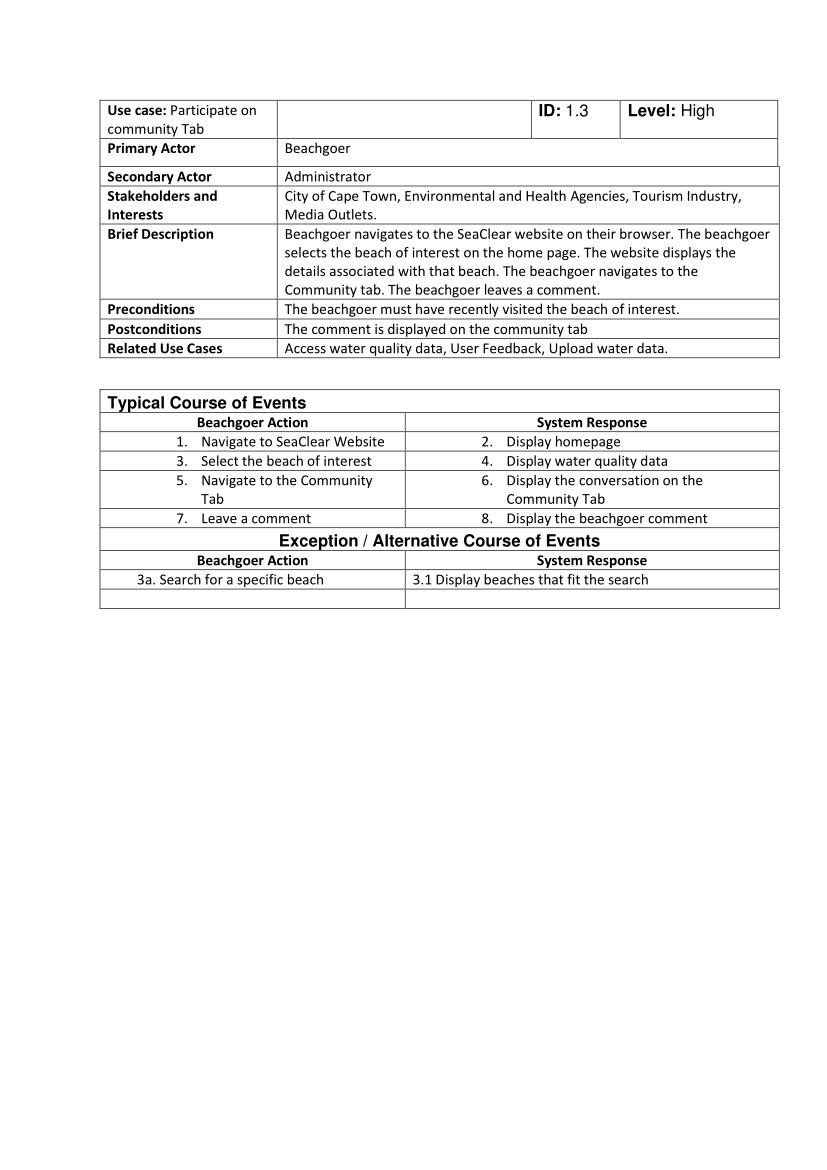
- Offline functionality for basic information access in areas with poor connectivity

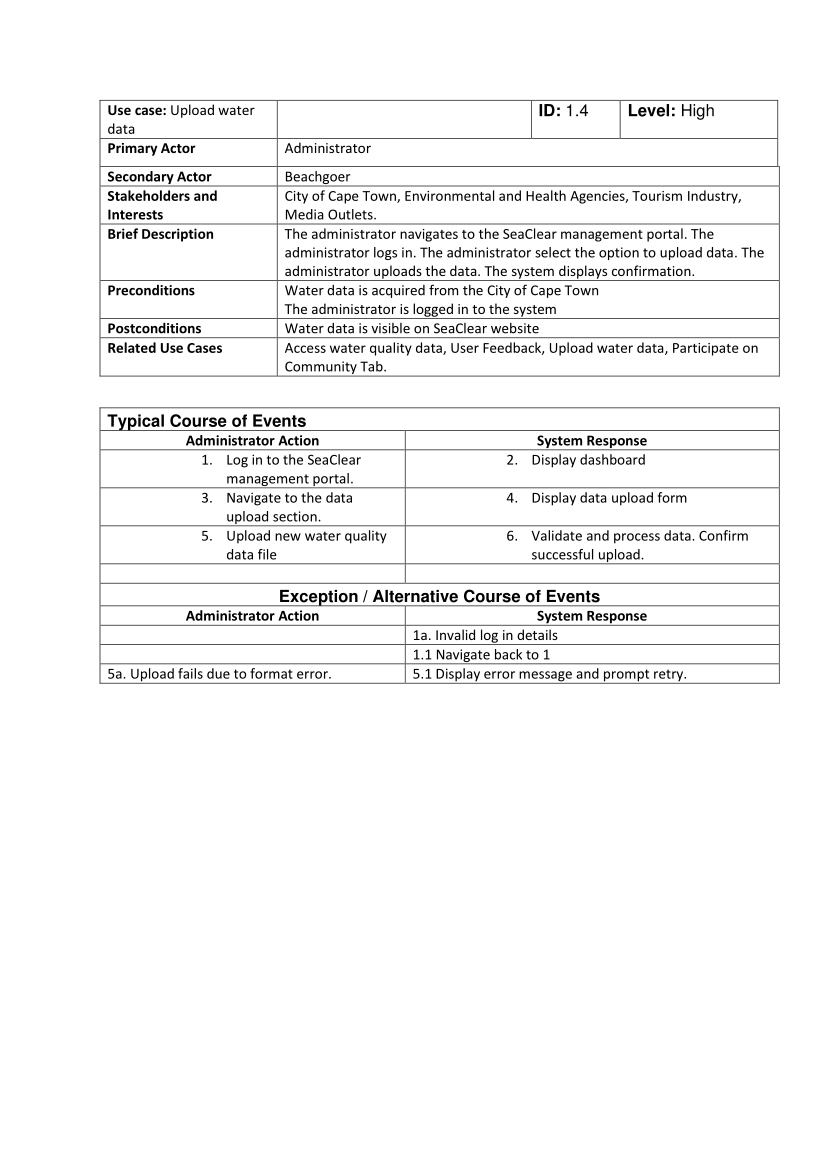
**2.4 Use Case Narratives and Diagrams**

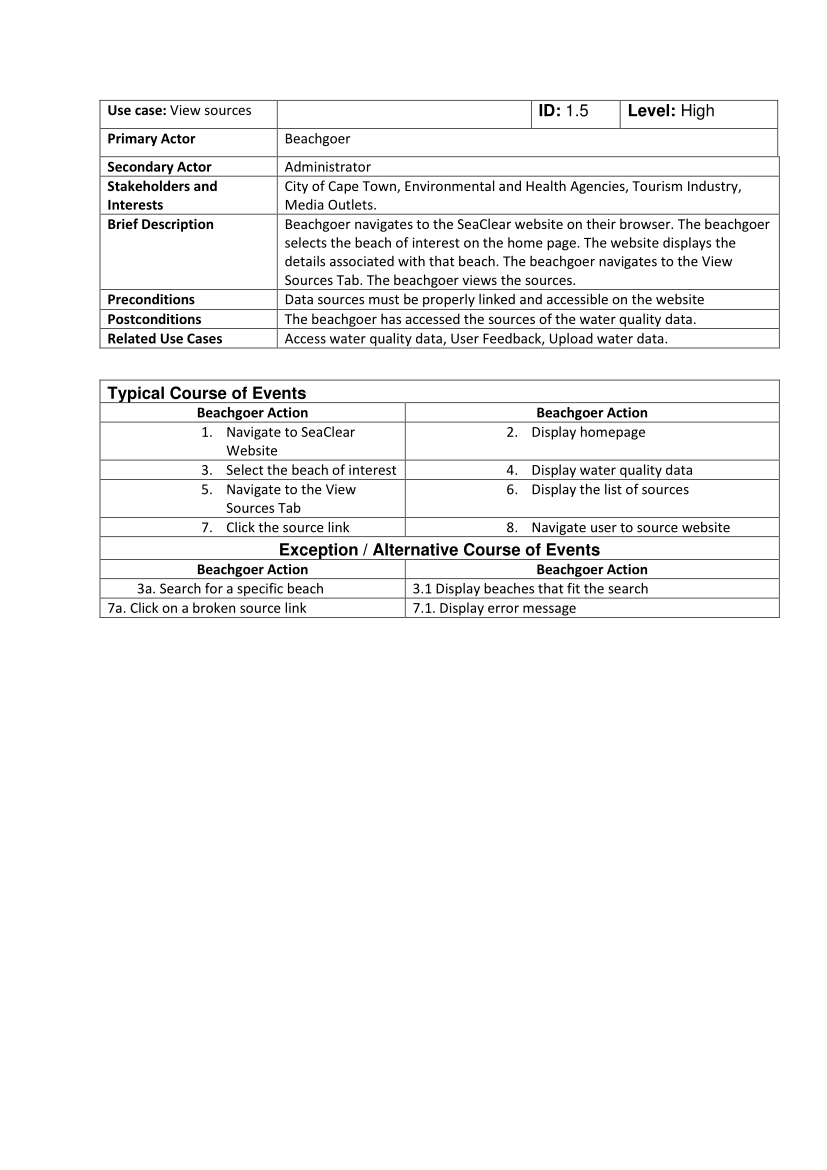
**[Your use case narratives and diagrams go here]**

**2.4.1 Use Cases**

****

****

****

****

**3. Design Overview**

**3.1 System Architecture**

[Your layered architecture diagram goes here]

This section is crucial as it gives readers an understanding of how different components of your system interact and work together. Here's what you should include:

1. Layered Architecture Diagram: This should be a visual representation of your system's architecture. The diagram should clearly show each layer and how they interact with each other. Use boxes or layers to represent each component, and arrows to show the flow of data or requests between layers.
2. Explanation of Each Layer: For each layer mentioned (Presentation, Application, Data Access, Database, and External Services), provide a more detailed description:
   * Presentation Layer: Explain how React is used to create the user interface. Mention any key libraries or frameworks used for UI components, state management (e.g., Redux), or routing.
   * Application Layer: Describe how Node.js is used to handle the backend logic. Explain any key modules or frameworks used (e.g., Express.js). Mention how this layer processes requests from the frontend and manages the application's business logic.
   * Data Access Layer: Detail how this layer interacts with the database and external APIs. Explain any ORM (Object-Relational Mapping) tools used, or how you've structured your API calls.
   * Database Layer: Elaborate on why MongoDB was chosen. Explain the database schema, how data is organized, and any indexing strategies used for efficient querying.
   * External Services Layer: Describe how you integrate with external data sources. Explain any challenges in data synchronization or rate limiting, and how you've addressed them.
3. Data Flow: Explain how data flows through your system. For example, describe the journey of a user request from the frontend, through the backend, to the database, and back to the user.
4. Scalability and Maintainability: Elaborate on how this architecture supports scalability. Discuss how you can add new features or scale specific components without affecting the entire system.
5. Security Considerations: Mention any security measures implemented at different layers, such as data encryption, API authentication, or input validation.
6. Justification of Architectural Choices: Explain why you chose this particular architecture. Discuss any alternatives you considered and why this approach was deemed most suitable for SeaClear.
7. Future Extensibility: Briefly touch on how this architecture allows for future enhancements or integrations, such as adding new data sources or implementing advanced features.

The SeaClear system employs a modern, scalable architecture consisting of the following layers:

1. Presentation Layer: React-based frontend for responsive user interface

2. Application Layer: Node.js backend handling business logic and data processing

3. Data Access Layer: API services for retrieving and updating water quality data

4. Database Layer: MongoDB for storing beach information and historical water quality data

5. External Services Layer: Integration with City of Cape Town data sources and weather APIs

[Detailed explanation of each layer, data flow, scalability, security considerations, justification of architectural choices, and future extensibility goes here]

3.2 Design Class Diagram

[Your design class diagram goes here]

3.3 Data Organization

[Your explanation of data organization goes here]

3.4 Algorithms

[Your explanation of key algorithms goes here]

**4. Implementation**

**4.1 Data Structures**

Key data structures used in SeaClear include:

1. Beach object: Stores information about each beach, including name, location, and current water quality status.

2. WaterQualityMeasurement object: Represents a single water quality measurement, including parameters like E. coli levels, pH, and turbidity.

3. User object: Stores user preferences and notification settings for registered users.

4. Alert object: Represents water quality alerts or warnings for specific beaches.

[Further details on data structures go here]

**4.2 User Interface**

This section is describing the User Interface (UI) of the SeaClear system. Here's a detailed explanation of what should be included in this part of your report:

Technology Choice:  
  
 React was chosen for the frontend due to several key advantages:

- Component-based architecture: The code is organized into reusable components (e.g., HomePage, CliftonBeach), which promotes code reusability and easier maintenance.  
- Rich ecosystem: The project leverages popular React libraries like React Router (evident from the use of useNavigate and Link components) for seamless navigation.

- State management: React's useState hook is used extensively for managing local component state, such as searchInput and searchResults.

1. Key Features:

a. Dashboard:

* Layout: The dashboard (HomePage component) features a header with a search bar, an interactive map, and beach tiles displaying key information.
* Information displayed: The dashboard shows a map of beaches in Cape Town and individual beach tiles with status (safe/unsafe), description, and basic weather information.
* Data visualization: The code uses color-coding (safe/unsafe status) and image representation for different beaches.

b. Detailed Beach Pages:

* Layout: The CliftonBeach component represents a detailed beach page, with a header, status section, weather info, map, and additional details.
* Current water quality data: Presented as a status (safe/unsafe) with a color-coded indicator.
* Historical data: Not explicitly shown in the provided code, but the structure allows for easy integration of historical data visualizations.
* Interactive elements: The map allows users to explore nearby beaches, and there's a search function for finding specific beaches.

c. Interactive Map:

* Maplibre API integration: The code uses Maplibre GL JS, likely chosen for its open-source nature and compatibility with OpenStreetMap data.
* Map features: The map includes custom markers for each beach, popup information on hover, and click functionality to navigate to detailed beach pages.
* Visual representation: Beaches are represented by markers. The code suggests color-coding (e.g., red for unsafe, blue for safe beaches).
* Clustering: The current code doesn't implement clustering, but the map initialization provides a foundation for adding this feature.

d. Search Function:

* Implementation: The search bar is part of the header in both HomePage and CliftonBeach components.
* Autocomplete feature: As users type, the handleSearchInput function filters the beaches array based on the input, updating searchResults state.
* Efficiency: The current implementation uses a simple array filter method. For larger datasets, more efficient algorithms like trie data structures or backend-powered search could be implemented.

1. Material Design Principles: Elaborate on how material design is incorporated:
   * Explain key material design concepts used (e.g., elevation, responsive grids, meaningful transitions)
   * Describe how these principles contribute to the modern and intuitive feel of the UI
   * Mention any specific material design components or patterns used
2. Accessibility Considerations: Discuss how the UI design ensures accessibility:
   * Mention any WCAG guidelines followed
   * Describe features that make the app usable for people with disabilities (e.g., screen reader compatibility, keyboard navigation)
3. Performance Optimization: Explain any techniques used to ensure the UI remains responsive:
   * Discuss lazy loading of components or images
   * Mention any performance monitoring tools used
4. User Testing and Iterations: Briefly mention how user feedback influenced the UI design:
   * Describe any major changes made based on user testing
   * Highlight how the current design addresses user needs and preferences
5. Future UI Enhancements: Conclude with potential future improvements or features planned for the UI

expand on these points, you'll provide a comprehensive overview of the user interface, demonstrating how it effectively communicates water quality information to users in an accessible and intuitive manner

**4.3 Key Classes and Methods**

**[Your explanation of key classes and methods goes here]**

**4.4 Special Programming Techniques or Libraries**

**[Your explanation of special programming techniques or libraries goes here]**

1. **Program Validation and Verification**

**This section is crucial as it demonstrates how you've ensured that the SeaClear system meets its requirements and functions correctly.**

**5.1 Quality Management Plan**

This subsection outlines the overall strategy for testing the system. It breaks down the testing approach into several key areas:

1. Unit Testing: This involves testing individual components or functions in isolation. For example, you might test a function that calculates water quality ratings to ensure it produces correct outputs for various inputs.
2. Integration Testing: This checks how different parts of the system work together. For instance, you might test if the data retrieved from the database is correctly displayed in the user interface.
3. User Acceptance Testing: This involves real users testing the system to ensure it meets their needs and expectations. It's crucial for validating the usability and effectiveness of the interface.
4. Performance Testing: This evaluates how the system performs under various conditions, especially under high load. It ensures the system remains responsive even when many users are accessing it simultaneously.
5. Security Testing: This identifies potential vulnerabilities in the system. For a system dealing with public data, it's crucial to ensure that the data cannot be tampered with and that user privacy is protected.

The table that should be included here would typically detail each type of testing, specifying what's being tested, how it's being tested, and the expected outcomes.

**5.2 Testing Methodology**

This subsection goes into more detail about the specific tools and techniques used for testing:

1. Jest for automated unit tests: Jest is a popular JavaScript testing framework. It's used here to automate the testing of individual components, ensuring they work correctly in isolation.
2. Cypress for integration tests: Cypress is an end-to-end testing framework. It's used to automate tests that simulate user interactions with the application, ensuring different parts of the system work together correctly.
3. Apache JMeter for load testing: JMeter is used to simulate high traffic scenarios. This helps ensure the system can handle many users accessing it simultaneously without slowing down or crashing.
4. Manual exploratory testing: This involves testers manually using the system to find unexpected issues or edge cases that automated tests might miss. It's particularly useful for identifying usability issues.
5. Accessibility testing: This ensures the system is usable by people with disabilities. WAVE is a tool that can automatically detect many accessibility issues, while manual screen reader testing involves actually using the system with a screen reader to ensure it's navigable and understandable.

*The final statement emphasizes that this combination of different testing approaches (the "multi-faceted approach") ensures that the system is thoroughly tested from various angles. This comprehensive testing strategy helps maintain high quality throughout the development process by catching and addressing issues early and consistently.*

*In your report, you might want to expand on each of these points, providing specific examples of tests you've run, issues you've uncovered and resolved, and how the testing process has improved the overall quality of the SeaClear system. You could also discuss how this testing strategy aligns with industry best practices and how it specifically addresses the unique challenges of a water quality information system.*

**5.3 Test Results and Analysis [Your test results and analysis go here]  
  
6. Conclusion**

**[Your conclusion goes here]**

**7. References**

**[Your references go here]**

**Appendix A: User Manual**

**[Your user manual goes here]**

**Appendix B: Test Results**

**[Your detailed test results go here]**