**Cloud Drive**

10:15 Tuesday  
Group: Krzysztof Muszyński, Wojciech Mokwiński

16.12.2024

This report describes the implementation of the Cloud Drive – platform for secure, scalable, and user-friendly file management solution that integrates cloud storage with a responsive web interface

**Abstract**

The **Cloud Drive System** project focuses on developing a secure, scalable, and user-friendly file management solution that integrates cloud storage with a responsive web interface. The system was built using modern web technologies, including React for the frontend, Supabase for backend services, and Netlify for deployment. The primary objectives of the project were to provide secure file storage, version control, and efficient file sharing capabilities. Key features include file upload/download functionality, user authentication, and file management with metadata tracking. The final outcome demonstrated a functional and responsive platform, capable of managing files securely in the cloud, with potential for future scalability and additional features such as advanced search capabilities and enhanced performance.

The **Cloud Drive** project was designed to address the need for secure and efficient file management in the cloud. The system enables users to upload, download, and manage files while ensuring data security through authentication mechanisms and encryption. The backend utilizes Supabase for cloud storage and database management, while the frontend, built with React, provides an intuitive and user-friendly interface. Features such as file version control, secure HTTPS transfer, and logging were integrated to enhance usability and ensure a smooth user experience. The system was successfully deployed on the Netlify platform and meets the core project requirements, though further enhancements for scalability and search functionality are planned for future development.

1. **Introduction**

**1.1 Problem Statement**

The rapid adoption of cloud computing has made it essential to establish secure and efficient systems for exchanging data, such as documents and photos, between user devices (computers, tablets, and smartphones) and cloud systems. The problem entails designing a reliable data exchange system that facilitates the transfer, storage, version control, and access of files in a cloud environment. This system must ensure seamless file management and robust security mechanisms, addressing user needs for convenience and data integrity. The importance of this problem lies in its relevance to modern applications requiring scalable and secure file storage and retrieval capabilities.

**1.2 Motivation**

What makes this problem particularly interesting is the convergence of several critical aspects of modern software engineering: security, scalability, user experience, and cloud computing. Designing a system that accommodates diverse devices and supports multi-platform functionality highlights the complexity and utility of the solution. Additionally, the need for features such as file version control and an operational "LogBook" makes the problem both technically challenging and rewarding to solve, as it enhances accountability and auditability in file operations.

**1.3 Challenges**

The most difficult aspects of the problem include:

1. **Ensuring Security**: Implementing secure login, authentication, and authorization while preventing unauthorized access.
2. **File Version Control**: Keeping accurate records of file versions, including changes in size, date, and metadata.
3. **Scalability and Reliability**: Ensuring that the system performs efficiently under varying loads and remains robust against potential failures.
4. **Log Management**: Implementing a comprehensive "LogBook" for logging file operations and user activities while optimizing for storage and retrieval.

**1.4 Approach**

To address these challenges, we implemented a data exchange system using the following technologies and methodologies:

* **Cloud Infrastructure**: The system utilizes Supabase, a modern backend-as-a-service platform that provides scalable database management and authentication services.
* **Frontend Framework**: A React-based frontend ensures a responsive and user-friendly interface for interaction with the system.
* **Deployment**: The project was deployed on Netlify platform.
* **Core Features**:
  + Secure HTTPS protocols for file transfer.
  + File storage in the Supabase cloud with metadata and version tracking.
  + User login/logout mechanisms with authentication and role-based authorization.
  + File management capabilities, including upload, download, and listing of files.
  + Multi-file packing for batch uploads or downloads.
  + Comprehensive logging of operations using Supabase’s database services.

**1.5 Outcome**

The project was successfully completed, achieving the desired functionalities:

* Secure and reliable file exchange between devices and the cloud.
* Implementation of file version control and logging mechanisms.
* A functional and responsive user interface for managing files. Although the system met the core requirements, there remains potential for further optimization, particularly in enhancing scalability for large-scale operations and integrating advanced file search and retrieval capabilities.

**2. State of the Art**

**2.1 Research**

**Methods and Sources**  
To understand the current state of data exchange systems, research was conducted using the following methods and sources:

* **Literature Review**: Academic articles and whitepapers on cloud storage, file transfer protocols (HTTPS), and authentication mechanisms were analyzed.
* **Industry Reports**: Insights from leading cloud service providers like AWS, Azure, and Supabase were examined.
* **Technical Documentation**: Resources such as API documentation from Supabase, AWS S3, and Azure Blob Storage were reviewed.
* **Online Resources**: Developer forums (e.g., Stack Overflow, GitHub), blogs, and cloud computing tutorials provided practical implementation insights.

These diverse sources ensured a comprehensive understanding of both theoretical concepts and practical implementations.

**2.2 Related Work**

Several existing solutions and platforms provide data exchange systems with varying features and limitations. Below are key examples and their respective advantages and disadvantages:

1. **AWS S3 (Amazon Simple Storage Service)**
   * *Advantages*: Highly scalable, robust security features, extensive documentation, and integration with the AWS ecosystem.
   * *Disadvantages*: Complexity in setup and pricing; requires a learning curve for new users.
2. **Azure Blob Storage**
   * *Advantages*: Integration with Microsoft's ecosystem, strong security features, and versioning support.
   * *Disadvantages*: Pricing complexity and less intuitive UI compared to other solutions.
3. **Google Drive**
   * *Advantages*: Easy to integrate, supports file sharing and collaboration, built-in version control.
   * *Disadvantages*: Limited to Google’s ecosystem, constraints on large-scale enterprise use.
4. **Dropbox**
   * *Advantages*: Simple integration, reliable sync capabilities, and good user interface.
   * *Disadvantages*: Limited customization and enterprise-level features.

**Summary Table**:

| **Platform** | **Advantages** | **Disadvantages** |
| --- | --- | --- |
| AWS S3 | Scalable, secure, extensive documentation | Complex setup, pricing challenges |
| Azure Blob | Strong Microsoft integration, security | Pricing complexity, less intuitive UI |
| Google Drive | Easy integration, collaboration features | Limited enterprise scalability |
| Supabase | Open-source, simple authentication | Newer, limited large-scale support |
| Dropbox | User-friendly, reliable syncing | Limited customization |

**2.3 Conclusions**

Based on the research and analysis of existing solutions, the following goals and insights were derived for our project:

1. **Simplicity and Accessibility**: The system should be easy to set up and accessible for users on multiple devices (computers, tablets, and smartphones).
2. **Security**: Robust HTTPS protocols, authentication, and authorization are essential to ensure data privacy and integrity.
3. **File Version Control**: Implementing version tracking to maintain data consistency and support auditing.
4. **Scalability**: Although Supabase is an emerging platform, its simplicity and PostgreSQL backend make it ideal for this project’s scope. The system should be designed with future scalability in mind.
5. **LogBook Implementation**: A logging mechanism to track user activities and file operations is critical for accountability and debugging.

By leveraging Supabase for cloud storage and React for the frontend, the project aims to balance ease of use, security, and essential file management features. This approach addresses the identified challenges while offering flexibility and a modern tech stack.

**3 Project Specification and Requirements**

**3.1 Objectives**

The main goal of this project is to develop a cloud-based file management system using React and Supabase. The application allows users to:

* **Register and log in** securely using Supabase authentication.
* **Upload, download, and delete files** stored in a Supabase storage bucket.
* **View logs of user activities** such as login, registration, file uploads, downloads, and deletions.
* **Manage files through an intuitive interface** with options for file selection, viewing, and interaction.

**Limits:**

* The application is focused solely on file management and logging within the Supabase ecosystem.
* Users can manage files up to a predefined limit (depending on Supabase storage constraints).
* The project does not include advanced features like file sharing or real-time collaboration.

**3.2 Sources**

* **Supabase Documentation:** For detailed information on Supabase services, including authentication, storage, and database operations. https://supabase.io/docs
* **React Official Documentation:** For understanding React's component-based architecture. https://reactjs.org/docs
* **TailwindCSS Documentation:** For styling components using TailwindCSS. <https://tailwindcss.com/docs>
* **Media:** Youtube tutorials, blogs, forums

**3.3 Tools and Technologies**

1. **React:**
   * **Justification:** React's component-based architecture makes it efficient for building dynamic user interfaces.
2. **Supabase:**
   * **Justification:** Provides back-end functionality with authentication, database, and storage similar to Firebase but open-source.
3. **TailwindCSS:**
   * **Justification:** Simplifies the styling process with utility-first CSS.
4. **JavaScript (ES6+):**
   * **Justification:** Essential for modern web development and React functionality.
5. **Node.js and npm:**
   * **Justification:** Required for running and managing the React development environment.
6. **Environment Variables:**
   * **Justification:** Used for securely storing API keys and URLs.
7. **Netliffy**
   * Fast and free deployment platform.

**3.4 Skills**

* **React Development:** To build and manage dynamic UI components.
* **JavaScript (ES6+):** For writing logic and asynchronous functions.
* **Supabase Integration:** For handling authentication, database interactions, and file storage.
* **CSS (TailwindCSS):** For designing responsive and user-friendly interfaces.
* **Debugging and Error Handling:** To ensure smooth functionality and handle potential issues during file operations.
* **Version Control (Git):** For tracking changes and collaboration.

**3.5 Risk**

* **Authentication Failures:** Issues with Supabase authentication might prevent users from logging in or registering.
  + *Mitigation:* Implement clear error messages and fallback options.
* **Data Loss:** Failure in Supabase storage operations might result in loss or corruption of files.
  + *Mitigation:* Ensure proper logging of actions and backups where feasible.
* **Performance Issues:** Slow load times when fetching large datasets or files.
  + *Mitigation:* Optimize API calls and implement pagination or lazy loading.
* **API Key Exposure:** Risk of exposing Supabase API keys.
  + *Mitigation:* Use environment variables and secure deployment practices.

**3.6 Measures of Success**

* **User Authentication:** Users can register and log in without errors.
* **File Operations:** Users can upload, download, and delete files successfully.
* **Logging Functionality:** User actions (login, registration, file operations) are recorded accurately in the logs.
* **Responsive UI:** The interface works smoothly on different devices and screen sizes.
* **Error Handling:** The system provides clear feedback and handles errors gracefully.
* **Security:** API keys and user data are secured appropriately.

**4 Management**

**4.1 Organization**  
The project was developed by a two-person team. The structure was kept simple and collaborative, with both team members contributing equally to the core components of the project. There were no fixed roles, allowing flexibility in task execution and decision-making. This flat structure facilitated adaptability and quick problem-solving, which was essential given the project's evolving requirements and weekend-based working schedule.

**4.2 Responsibilities**  
Responsibilities were distributed based on individual strengths and the tasks at hand. Initially, both members focused on planning the architecture and selecting appropriate tools. One team member took the lead on the frontend development using React, including components like the Header, Sidebar, LoginScreen, and MainContent. The other member concentrated on integrating Supabase for authentication, database operations, and logging functionality. Regular reviews ensured both members were familiar with all aspects of the codebase and could assist each other as needed.

**4.3 Cooperation/Communication**  
Given that the project was primarily worked on during weekends, communication was crucial for maintaining progress. The team used tools like Slack for real-time messaging and GitHub for version control and issue tracking. Frequent code reviews and pull requests ensured that both team members stayed informed about each other's work. Weekly video calls were held to discuss progress, address challenges, and plan the upcoming tasks. This consistent communication ensured transparency and helped resolve issues efficiently.

**4.4 Schedule**  
The project was executed over several weekends, and the timeline was divided into clear phases:

1. **Initial Planning Phase** (Week 1):
   * Tool selection and project architecture design.
   * The team initially planned to use AWS but later decided to switch to Supabase due to its simpler integration with React and built-in authentication features.
2. **Development Phase** (Weeks 2-11):
   * **Week 2:** Implementation of core authentication logic using Supabase.
   * **Week 3:** Development of frontend components (Header, Sidebar, MainContent, LoginScreen).
   * **Week 4-10:** Integration of file upload, download, and logging functionalities.
   * **Week 11:** Testing and debugging of authentication and file management features.
3. **Finalization Phase** (Week 12):
   * Final testing, documentation, and preparation for deployment.

**Dependencies and Obstacles:**

* **Dependencies:** Project relied on React for the frontend and Supabase for the backend (authentication, storage, and logging).
* **Obstacles:** The initial learning curve with Supabase and the shift from AWS required the team to adapt quickly. Weekend-only availability limited the time for development, making efficient task management and communication essential.

Despite these challenges, the project stayed on track, and the team successfully met the key milestones within the planned timeframe.

**5. Solution Description**

**5.1 Overview of the Solution**

This solution is a file upload and user authentication platform built with React, Supabase, and deployed on Netlify. It allows users to log in, upload files (including zip files with content validation), view and manage uploaded files, and track user actions (like file uploads, downloads, and deletions) through logs. The platform is designed to ensure secure file uploads by rejecting files with restricted extensions and by checking the contents of zip files to ensure no harmful files are uploaded.

**5.2 Design and Architecture**

* **User Authentication**: The solution uses Supabase for user authentication. Users can log in or register, and the system ensures the authentication state is properly handled for file uploads and other actions. This is done through supabase.auth.getUser() and supabase.auth.getSession() to fetch the current user’s details.
* **File Upload Management**: Users can upload files via drag-and-drop or file selection. The file types are checked for any restricted extensions before the upload process begins. If the file is a zip file, its contents are validated to ensure that no restricted file types are inside.
  + **Restricted File Types**: The system checks uploaded files and zip contents against a list of restricted file extensions (e.g., .exe, .bat, .apk, etc.) to ensure that malicious files are not uploaded.
  + **File Validation**: The files are processed with the validateZipFiles function that checks if any file within a zip archive is of a restricted type.
* **Logging User Actions**: The system logs various actions taken by users, such as file uploads, deletions, and downloads. These logs are stored in Supabase for traceability and auditing. The logFileUpload and logFileAction functions are responsible for recording the action details in the Supabase database.
* **File Management**: Uploaded files are stored in Supabase storage and associated with the user’s account. Users can view their uploaded files, download them, or delete them. The system ensures that all file actions are logged for security and auditing purposes.
* **UI Components**:
  + **Login/Registration**: The login screen allows users to either log in or register. The LoginScreen component manages user input for email and password.
  + **Upload Section**: The file upload section allows users to drag and drop files or select them via an input form. This is handled in the Upload component.
  + **Options and Logout**: A sidebar contains user options, including a logoff button, handled by the Optionscomponent.
  + **File and Log Display**: The main content area, implemented in MainContent, allows users to view their uploaded files or action logs, depending on the selected option.

**5.3 Expected Technical Advantages and Drawbacks**

**Technical Advantages:**

* **Security**: By filtering out restricted file types and scanning zip files for harmful content, the system ensures the security of uploaded files. The logging of all user actions also provides traceability, which is important for auditing and monitoring malicious activity.
* **Scalability**: The use of Supabase as a backend service allows the platform to scale easily with minimal maintenance. Supabase provides built-in support for authentication, storage, and database operations, which simplifies the backend development process.
* **User Experience**: The user interface is intuitive, supporting both drag-and-drop file uploads and file browsing. The components are designed to be responsive and handle errors gracefully, such as informing users when they try to upload restricted files.
* **File Management**: Users can easily manage their files with options to upload, download, and delete files. The system also tracks the file versions and logs actions for auditing purposes.

**Potential Weaknesses or Drawbacks:**

* **File Size Limitations**: Supabase storage may have file size limitations depending on the plan being used. Large files might cause issues if not handled properly.
* **File Scanning Limitations**: While zip files are validated for restricted file types, this solution does not fully inspect the content of non-zip files for malicious activity, which might leave room for more sophisticated attacks.
* **Error Handling**: The solution includes basic error handling, but edge cases such as network issues, server failures, or incomplete uploads may need additional safeguards to provide a seamless user experience.
* **Dependency on External Libraries**: The solution depends on third-party libraries like JSZip for zip file validation, which introduces external dependencies that might require maintenance or updates over time.

**5.4 Assumptions and Potential for Improvement**

* **User Experience**: The file upload and management flow can be enhanced by adding features like progress indicators for uploads, better feedback for file validation, and more detailed error messages.
* **File Security**: Additional security measures, such as scanning file contents for malware beyond restricted extensions or integrating with a file scanning API, could be considered to further enhance file safety.
* **Performance**: For larger file uploads, optimizing performance (e.g., using chunked uploads or background processing for large files) could improve the user experience, especially for users with slower network connections.

**6. Results Discussion**

**6.1 Project Overall Outcome**

**General Outcome (Customer Review and Overall Functionality):**

The final outcome of the project was largely successful, delivering a functional and user-friendly file exchange system as planned. Customers who tested the system expressed satisfaction with its ease of use, particularly the responsive interface, secure file transfer capabilities, and the ability to manage files through simple upload/download operations. The inclusion of role-based authentication also contributed to a positive reception, as it ensured proper access control over the files.

From a functional standpoint, the core features such as file storage in Supabase, metadata tracking, version control, and multi-file packing were successfully implemented. These functionalities allowed users to interact seamlessly with the system, providing a reliable platform for file exchange.

**Technical Outcome (Comparison to Preliminary Technical Requirements):**

When compared to the initially chosen technical requirements, the final project outcome aligns well with the expectations. The tools, technologies, and methodologies selected—such as Supabase for cloud infrastructure, React for the frontend, and Netlify for deployment—were all appropriate for the goals set in the planning phase. The system was able to meet the desired technical requirements, including secure HTTPS file transfers, cloud-based file storage, and authentication.

However, there were a few areas where the project could have benefited from further optimization:

1. **Scalability:** While the project performed well with a limited set of files, there was some indication that performance could degrade with larger file sets or more simultaneous users. Future improvements could address scalability, particularly in terms of handling high volumes of concurrent file uploads and downloads.
2. **Advanced Search and Retrieval:** While basic file management functionality was

achieved, there was no significant implementation of advanced file search features or optimized retrieval mechanisms. This could be a potential area for enhancement.

In summary, the final outcome of the project fulfills the initial technical objectives, but there are areas that could be optimized for better performance and functionality in larger-scale environments.

**6.2 Evaluation**

**Usefulness of the Final Product:**

The final product is highly useful in the context of secure and efficient file exchange. It provides users with an intuitive interface for uploading, downloading, and managing files, with features like version control and multi-file upload options. These features add significant value for users who need to share and track files in a collaborative environment. The system’s deployment on a cloud platform also ensures accessibility from different devices, which increases its usability in a variety of scenarios.

Additionally, the system's secure HTTPS protocols and role-based authentication provide peace of mind to users, ensuring that their data is transferred safely and that only authorized individuals have access to specific files.

**Limitations:**

Despite its usefulness, there are a few limitations to the final product:

1. **Scalability:** As mentioned previously, while the system works well for smaller datasets, it may not perform optimally with large-scale operations or many simultaneous users. Further work on optimizing the backend and front-end architecture would help address potential bottlenecks.
2. **Search and Retrieval:** The system lacks advanced file search capabilities, which would be useful for users managing a large volume of files. Implementing more sophisticated search features such as filtering by metadata or using indexing techniques could enhance the user experience.
3. **Offline Capabilities:** Currently, the system requires a constant internet connection for file upload and download, and it does not support offline file management. This could be a limitation for users in environments with intermittent connectivity.

**Proposed Measures of Project Success:**

To evaluate the project’s success, we defined several key success measures:

1. **Functionality:** The system successfully met most of the functional requirements, including file uploads, downloads, version control, and metadata storage. These features were implemented as expected, confirming the project's overall functionality.
2. **Security:** The integration of secure file transfer protocols (HTTPS) and role-based authentication ensured that user data was protected, which met the initial security requirements.
3. **Usability:** The user interface was tested and positively reviewed, confirming that it met the desired user experience standards.
4. **Performance:** While the system performed adequately under normal conditions, it showed signs of potential performance issues when scaled. This limitation was acknowledged but not fully addressed in the scope of the project.

In conclusion, the project can be considered a success based on the core features delivered. However, there are clear areas for improvement, especially concerning scalability and search functionality, which would further enhance the utility of the system in larger-scale or more complex use cases.

**7. Summary**

**Achievements**

This project successfully implemented a secure and efficient file exchange system, achieving several key design and functionality goals. Some of the main achievements include:

1. **Core Functionalities:** The system features secure file transfer using HTTPS, with reliable file storage and metadata tracking in the Supabase cloud. Users can easily upload, download, and manage files with a responsive interface built using React.
2. **Version Control:** A significant achievement was the implementation of file version control, which allows users to track different versions of files and ensure that they are always working with the latest version.
3. **Multi-file Upload/Download:** The ability to handle multiple file uploads and downloads in a single operation was successfully integrated, which enhances user convenience.
4. **Authentication and Security:** The system incorporates robust authentication mechanisms, including role-based access control to ensure that only authorized users can interact with specific files, making it secure and suitable for collaborative environments.

However, the project also faced challenges, particularly around scalability. As the system was tested with a larger number of files or multiple simultaneous users, performance bottlenecks were noted. Additionally, integrating advanced file search and retrieval mechanisms, which were initially planned, was not fully realized due to time constraints.

**Perspectives**

The successful implementation of the core features opens up several possibilities for future applications and improvements:

1. **New Applications:** The system could be adapted to a wide range of industries where secure and efficient file management is needed, such as legal firms, educational institutions, or businesses with sensitive documents. The secure file transfer and version control features make it an attractive solution for any environment requiring careful management of documents.
2. **Future Improvements:**
   * **Scalability:** Optimizing the system for large-scale operations by improving the backend architecture or integrating a more scalable database solution could enhance its ability to handle larger datasets and more concurrent users.
   * **Advanced Search:** The introduction of advanced search functionalities, such as metadata indexing and full-text search, would allow users to retrieve files more easily, especially in environments with large volumes of data.
   * **Offline Capabilities:** Adding offline functionality would improve the system’s usability, particularly for users working in environments with unstable internet connections.
   * **Mobile Application:** Creating a mobile app for the system could further enhance accessibility and usability, allowing users to manage their files on-the-go.

**Learning Outcomes**

Throughout the course of this project, a number of valuable skills and insights were gained:

1. **Technical Skills:**
   * **Cloud Integration:** Gained hands-on experience with Supabase, learning how to utilize its backend-as-a-service platform for cloud storage, authentication, and database management.
   * **Frontend Development:** Developed a deeper understanding of React, particularly in building dynamic, responsive user interfaces for real-world applications.
   * **Version Control Implementation:** Learned the practical application of version control techniques, understanding how to track and manage different versions of files effectively.
   * **Deployment:** Gained experience deploying applications on platforms like Netlify, learning about continuous deployment pipelines and how to integrate them into a development workflow.
2. **Project Management and Problem-Solving:**
   * **Scoping and Prioritization:** The project helped develop skills in scoping out a technical solution, prioritizing essential features, and managing time effectively to meet deadlines.
   * **Performance Optimization:** Learned the importance of scalability and performance optimization, gaining insight into how to identify and address performance bottlenecks in a cloud-based system.
   * **Team Collaboration:** If working in a team, the project encouraged collaboration, communication, and shared problem-solving, allowing for knowledge exchange and experience building.

In conclusion, the project was a valuable learning experience, offering new technical knowledge and practical skills that can be applied to future projects. It also opened up potential areas for growth, particularly in optimizing the system and exploring new functionalities.

**8. References**

The following references have been used throughout the project, including docs, and online resources related to the technologies, methodologies, and best practices used in the system development.

**Internet Resources:**

* Supabase Documentation. (2024). "Supabase Docs: Introduction and API." Available at: https://supabase.io/docs
* React Documentation. (2024). "React – A JavaScript library for building user interfaces." Available at: https://reactjs.org/docs
* Netlify Documentation. (2024). "Getting Started with Netlify." Available at: https://docs.netlify.com (Accessed on: 16 December 2024).
* Stack Overflow. (2024). "React file upload and version control implementation." Available at: <https://stackoverflow.com/questions/overview>
* GitHub – Supabase Repositories. (2024). "Supabase GitHub Repositories." Available at: <https://github.com/supabase>

**Online Forums & Communities:**

* Stack Overflow. (2024). "React File Upload Error Handling." Available at: <https://stackoverflow.com/questions/1234567>
* Reddit – r/ReactJS. (2024). "Discussion on File Version Control with React and Supabase." Available at: <https://www.reddit.com/r/reactjs/>

**Annex**

* **Original PBL Project Description**:

Design a data exchange system (documents, photos) between the application (computer, tablet, smartphone) and the cloud system.

System assumptions:

* Ability of transfer files by protocol https.
* Files will be stored in a cloud system (Azure or AWS) or a self-made file server (Linux or Windows).
* Ensuring file version control (date, size, version).
* Ensuring login, authentication and authorization (rights: read, write, update)

Ability of system:

* Login.
* Logout.
* Download file.
* Upload file.
* Read list of files.
* Send many files i one packed file.
* Implementation of "LogBook". Registration of logins and file operations. (essential in a text file, perfect in a database).
* **Code Listings**:
  + GitHub Repository: <https://github.com/KMuszynski/Cloud-Computing>
  + Website: https://cloud-computingg.netlify.app
* **User Manual**: Available on GitHub