Module Guide for Bridging Gaps: AI for Diagram Accessibility

Team 22, Reading4All
Nawaal Fatima
Dhruv Sardana
Fiza Sehar
Moly Mikhail
Casey Francine Bulaclac

November 13, 2025

1 Revision History

Date	Version	Notes
November 13 2025	1.0	Rev-1 Design Document

2 Reference Material

This section records information for easy reference.

2.1 Abbreviations and Acronyms

symbol	description
AC	Anticipated Change
DAG	Directed Acyclic Graph
M	Module
MG	Module Guide
OS	Operating System
R	Requirement
SC	Scientific Computing
SRS	Software Requirements Specification
Bridging Gaps: AI for Diagram Accessibility	Explanation of program name
UC	Unlikely Change
WCAG 2.1	Web Content Accessibility Guidelines
AODA	Accessibility for Ontarians with Disabilities Act
UI	User Interface
DOM	Document Object Model
AI	Artificial Intelligence
ML	Machine Learning
ARIA	Accessible Rich Internet Applications

Contents

List of Tables

List of Figures

3   Introduction

Decomposing a system into modules is a commonly accepted approach to developing software. A module is a work assignment for a programmer or programming team (Parnas et al., 1984). We advocate a decomposition based on the principle of information hiding (Parnas, 1972). This principle supports design for change, because the "secrets" that each module hides represent likely future changes. Design for change is valuable in SC, where modifications are frequent, especially during initial development as the solution space is explored.

Our design follows the rules laid out by Parnas et al. (1984), as follows:

- System details that are likely to change independently should be the secrets of separate modules.

- Each data structure is implemented in only one module.

- Any other program that requires information stored in a module's data structures must obtain it by calling access programs belonging to that module.

After completing the first stage of the design, the Software Requirements Specification (SRS), the Module Guide (MG) is developed (Parnas et al., 1984). The MG specifies the modular structure of the system and is intended to allow both designers and maintainers to easily identify the parts of the software. The potential readers of this document are as follows:

- New project members: This document can be a guide for a new project member to easily understand the overall structure and quickly find the relevant modules they are searching for.

- Maintainers: The hierarchical structure of the module guide improves the maintainers' understanding when they need to make changes to the system. It is important for a maintainer to update the relevant sections of the document after changes have been made.

- Designers: Once the module guide has been written, it can be used to check for consistency, feasibility, and flexibility. Designers can verify the system in various ways, such as consistency among modules, feasibility of the decomposition, and flexibility of the design.

The rest of the document is organized as follows. Section 4 lists the anticipated and unlikely changes of the software requirements. Section 5 summarizes the module decomposition that was constructed according to the likely changes. Section 6 specifies the connections between the software requirements and the modules. Section 7 gives a detailed description of the modules. Section 8 includes two traceability matrices. One checks the completeness of the design against the requirements provided in the SRS. The other shows the relation between anticipated changes and the modules. Section 9 describes the use relation between modules.

4 Anticipated and Unlikely Changes

This section lists possible changes to the system. According to the likeliness of the change, the possible changes are classified into two categories. Anticipated changes are listed in Section 4.1, and unlikely changes are listed in Section 4.2.

4.1 Anticipated Changes

Anticipated changes are the source of the information that is to be hidden inside the modules. Ideally, changing one of the anticipated changes will only require changing the one module that hides the associated decision. The approach adapted here is called design for change.

AC1: The specific infrastructure on which the software is running.

AC2: The format of the initial input data accepted by the system; in the future the system might extend to support PDFs and other formats.

AC3: The machine learning model used for alt-text generation.

AC4: The language used for alt-text generation, allowing the system to generate alt-text in multiple languages beyond English.

AC5: The duration of time the user remains authenticated before being prompted to log in again.

AC6: The range of images supported for alt-text generation can be extended beyond academic STEM images.

AC7: The platform that the system operates on can be extended beyond a web application to also include desktop or mobile applications.

4.2 Unlikely Changes

The module design should be as general as possible. However, a general system is more complex. Sometimes this complexity is not necessary. Fixing some design decisions at the system architecture stage can simplify the software design. If these decision should later need to be changed, then many parts of the design will potentially need to be modified. Hence, it is not intended that these decisions will be changed.

The following are changes that the system is unlikely to incur:

UC1: The project's evaluation metrics as shown in Table 5 of the Appendix in our SRS (2025) document will not change as these metrics are concrete and will be used to evaluate the effectiveness of the generated alternative text.

UC2: The primary output device for this system will remain as a screen, even if the system is later expanded to additional platforms (i.e., mobile) and not just the current web-based tool.

UC3: The system will always support keyboard input as its primary interaction method, with mouse input also supported for selecting user interface elements. This is to ensure that the functionality of keyboard navigation is always enabled for accessibility.

5   Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 2. The modules listed in Table 1 below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

AI Model Modules	Backend Modules	Frontend Modules	System Information Modules
DataPreprocess-Module	AuthenticationService Module	UserInterface-Interactions Module	Logging Module
WCAGCompliance Module	SessionManagement Module	MainScreen Module	
ModelOutput Module	ImageHandling-Validation Module	ShowHistoryScreen Module	
AIModelTraining Module	BackendController Module	LogInScreen Module	
CaptionGeneration Module		UserInterface-Accessibility Module	
InternalMetric-Compliance Module			
FeedbackMetrics Module			
FeedbackIntegration Module			
Feedback Module			

Table 1: Module Hierarchy by Subsystem

Level 1	Level 2
Hardware-Hiding Module [Section (7.1)]	N/A
Behaviour-Hiding Module [Section (7.2)]	DataPreprocess Module (M1) (see 7.2.1) WCAGCompliance Module (M2) (see 7.2.2) ModelOutput Module (M3) (see 7.2.3) AuthenticationService Module (M4)(see 7.2.4) SessionManagement Module (M5) (see 7.2.10) BackendController Module (M6) (see 7.2.11) LoggingModule (M7) (see 7.2.12) ImageValidation Module (M8) (see 7.2.13) UserInterfaceInteractions Module (M9) (see 7.2.5) MainScreen Module (M10) (see 7.2.6) ShowHistoryScreen Module (M11) (see 7.2.7) LogInScreen Module (M12) (see 7.2.8) UserInterfaceAccessibility Module (M13) (see 7.2.9)
Software Decision Module [Section (7.3)]	AIModelTraining Module (M14) (see 7.3.1) CaptionGeneration Module (M15) (see 7.3.2) InternalMetricCompliance (M16) (see 7.3.3) FeedbackMetrics Module (M17)(see 7.3.4) FeedbackLoop Module (M18) (see 7.3.5) Feedback Module (M19) (see 7.3.6)

Table 2: Module Hierarchy

6   Connection Between Requirements and Design

The design of the system is intended to satisfy the requirements developed in the SRS. In this stage, the system is decomposed into modules. The connection between requirements and modules is listed in Tables 3-17 in section 8.

The decompositon of the modules was separated by four main subsystems including: Frontend Modules, Backend Modules, AI Model Modules, and System Information Modules.

The front-end modules of the *Reading4All* system collectively ensure that the system meets all Look and Feel, Usability, Humanity, and Accessibility requirements through a responsive design and and clear interaction patterns. The `MainScreen`, `LogInScreen`, and

`ShowHistoryScreen` modules that are responsible for rendering the different screens of the user interface will use simple and clean layouts, WCAG-compliant contrast ratios, non-color-dependent cues, and fully descriptive alt text for non-text elements. These display modules present only essential information and will remain intuitive to navigate. The `UserInterfaceInteractions` module will ensure usability by providing immediate textual feedback and ensure to support efficient task completion with minimal steps. Lastly, within the frontend modules, the `UserInterfaceAccessibility` module will ensure compatibility with screen readers and keyboard navigation by managing ARIA landmarks, live-region announcements, and predictable focus order. Together, these modules will deliver an interface that is simple, memorable, inclusive, and fully aligned with WCAG 2.1 Level AA and AODA standards. It will allow all users to operate the system effectively and confidently.

In the backend subsystem, many modules play a role in ensuring that the functional, performance, privacy and operational requirements specified are fulfilled. The `BackendController Module` coordinates the flow of data between the Frontend, AI modules and other Backend modules . It ensures that user's and their inputs are validated, and alternative text is generated by invoking the appropriate modules. The `BackendController` also ensures that users are presented with the appropriate information and never exposed to technical details. The `ImageValidation` module fulfills the requirements relating to validating user inputs. This module checks that inputted images are of the correct size and types before the alt text generation process begins. The `Logging` module records any system events and errors, helping fulfill requirements related to protecting user data and having a reliable system. The `Logging` module will help ensure that user data is always deleted when any system failure occurs.

Furthermore, within the backend subsystem, the `AuthenticationService` and `SessionManagement` modules together ensure secure and reliable user interaction with the Reading4All platform. The `AuthenticationService` module is responsible for verifying user credentials, issuing secure tokens, and validating user identity before granting access to system functionalities. It guarantees that only authenticated users can upload images or retrieve generated alt text, fulfilling the security and privacy requirements defined in the SRS. Supporting this, the `SessionManagement` module manages each user's active session lifecycle by creating, validating, refreshing, and terminating session tokens. It also maintains a short-term record of user interactions during the active session, such as image uploads and model outputs, ensuring that users can view their recent activity while preserving system integrity and confidentiality. Together, these two backend modules ensure that all authentication, authorization, and session tracking processes are handled transparently and securely, thereby supporting reliability, data protection, and user trust across the Reading4All system.

7 Module Decomposition

Modules are decomposed according to the principle of "information hiding" proposed by Parnas et al. (1984). The *Secrets* field in a module decomposition is a brief statement of the design decision hidden by the module. The *Services* field specifies *what* the module will do without documenting *how* to do it. For each module, a suggestion for the implementing software is given under the *Implemented By* title. If the entry is *OS*, this means that the module is provided by the operating system or by standard programming language libraries. *Bridging Gaps: AI for Diagram Accessibility* means the module will be implemented by the Bridging Gaps: AI for Diagram Accessibility software.

Only the leaf modules in the hierarchy have to be implemented. If a dash (–) is shown, this means that the module is not a leaf and will not have to be implemented.

7.1 Hardware Hiding Modules

This subsection is N/A as the Reading4All system has no hardware components.

7.2 Behaviour-Hiding Modules

This subsection includes modules that pertain to the required behavior of the Reading4All system. The modules are defined as their own subsection and follow the following format:

Secrets: This part describes the essential behavioral components that define a module's intended functionality.

Services: This component defines the module's programs that implement the externally visible behaviors of the system, as outlined in the Software Requirements Specification (SRS). Any modifications to the SRS will likely require corresponding updates to the programs within this module.

Implemented By: This part mentions if the Reading4All system or external systems will implement the specific module.

Type of Module: This section defines the module type into a Record, Library, Abstract Object, or Abstract Data Type. This section also includes information for leaf modules in the decomposition by secrets tree.

7.2.1 DataPreprocess Module (M1)

Secrets: Includes the steps used to prepare input images for further processing.

Services: Normalizes, sorts, and standardizes the image input so it can be correctly used by the training and caption generation modules.

Implemented By: Reading4All

Type of Module: Library

7.2.2 WCAGCompliance Module (M2)

Secrets: The rules used to decide whether a caption meets AODA (Ontario (2005)) and WCAG 2.1 AA accessibility standards (W3C (2018)).

Services: Checks the generated text against WCAG 2.1 AA (W3C (2018)) criteria and flags or adjusts any generated alt text that does not meet these standards.

Implemented By: Reading4All

Type of Module: Library

7.2.3 ModelOutput Module (M3)

Secrets: The structure used to store and present the final text to the user via the Backend-Controller as mentioned in Section 7.2.11.

Services: Produces and formats the final version of the accessibility-compliant text for output or download.

Implemented By: Reading4All

Type of Module: Record

7.2.4 AuthenticationService Module (M4)

Secrets: The internal logic for credential verification, token structure and signing, token refresh rules, lockout thresholds, and interactions with external authentication providers. Includes error classification for authentication failures and audit-safe handling of sensitive data.

Services: Verifies user credentials, issues and refreshes authentication tokens, validates tokens on incoming requests, resolves current user identity (userID/role) for access control, and coordinates with Session Management to create/terminate sessions.

Implemented By: Reading4All

Type of Module: Library

7.2.5 UserInterfaceInterfactions Module (M9)

Secrets: The internal logic and mapping of user-triggered actions such as clicks and keyboard presses on the user interface into the backend and display screens modules.

Services: Captures user actions from the UI and connects UI controls to application behavior by attaching event handlers that calls on other modules' access programs with validated parameters.

Implemented By: Reading4All Team

Type of Module: Library

7.2.6   MainScreen Module (M10)

Secrets: The internal layout composition of the main screen. This includes how the upload panel and supporting UI elements are arranged and rendered into the `DOM`.

Services: Renders the "Main" screen of the application. This screen displays the image-upload entry point for users. It calls on the UserInterfaceAccessibility module to assign the main landmark, set initial focus, and announce the screen to screen readers.

Implemented By: Reading4All Team

Type of Module: Library

7.2.7   ShowHistoryScreen Module (M11)

Secrets: The internal layout composition of the show history screen when an authenticated user wants to see their previously generated alt-text history. This includes how the history items are formatted and displayed.

Services: Renders the user's current session history to the screen. It relies on the UserInterfaceAccessibility module to set the main landmark, focus the first history item (if any), and announce the screen.

Implemented By: Reading4All Team

Type of Module: Library

7.2.8   LogInScreen Module (M12)

Secrets: The internal layout composition of the log in screen including username/email and password fields, and a log in button.

Services: Renders the system's Log In screen. It calls on the UserInterfaceAccessibility module to set focus on the fields on the log in form and announce the screen.

Implemented By: Reading4All Team

Type of Module: Library

7.2.9 UserInterfaceAccessibility Module (M13)

Secrets: The internal mechanism for providing screen reader and keyboard navigation compatibility in the browser. This includes how ARIA (W3C (2024)) live regions are implemented, how focus is manipulated, and how keyboard navigation is enabled internally.

Services: Provides accessibility capabilities to other user interface modules by: assigning and maintaining the correct main landmark on each screen; managing which element receives initial keyboard focus; sending announcements to screen readers using ARIA (W3C (2024)) live regions; and enabling keyboard navigation patterns.

Implemented By: Reading4All Team

Type of Module: Library

7.2.10 SessionManagement Module

Secrets: The representation and storage of active sessions, expiry/refresh policies, session-history data structures (interaction entries and Reading4All data references), and eviction/cleanup strategies. Includes indexing strategies for fast token lookups and privacy rules for session-scoped logs.

Services: Creates, validates, refreshes, and deletes sessions; appends interaction entries; links session activity to Reading4All data records; retrieves filtered session history and recent data references for the current session.

Implemented By: Reading4All

Type of Module: Abstract Data Type

7.2.11 BackendController Module (M6)

Secrets: The internal application flow of data and user inputs, including how backend requests are processed and routed between services.

Services: Coordinates backend responses by receiving requests from the frontend and directing it to the appropriate service such as the User Authentication or AI Model, and returning the results.

Implemented By: Reading4All

Type of Module: Abstract Objects

7.2.12   Logging Module (M7)

Secrets: The internal details of which events and errors are logged and the specific format used for log entries.

Services: Records system events and errors received from the BackendController Module.

Implemented By: Reading4All

Type of Module: Library

7.2.13   ImageValidation Module (M8)

Secrets: The internal details of how uploaded images are verified, including how file type, size are checked and whether a file is reachable.

Services: Validates uploaded images to ensure that they meet system requirements before the alt text generation process.

Implemented By: Reading4All

Type of Module: Library

7.3   Software Decision Modules

This subsection includes the modules' internal operations, connections with other modules and decision making components. Similar to Section 7.2 the modules are defined as their own subsection and follow the following format:

Secrets: Internal logic, data flow, and processing steps that define how the system produces and improves alternative text.

Services: Implements the internal operations of the system, such as model training, caption generation, quality validation, feedback collection, and output production.

Implemented By: This part mentions if the Reading4All system or external systems will implement the specific module.

Type of Module: This section defines the module type into a Record, Library, Abstract Object, or Abstract Data Type. This section also includes information for leaf modules in the decomposition by secrets tree.

7.3.1 AIModelTraining Module (M14)

Secrets: The internal design of how the model is built and improved. This includes how image and text data are linked, how training is divided into sets, and what hidden patterns the model learns. The specific model structure and training process are not exposed to other modules.

Services: Trains the system to understand relationships between images and text so it can later describe new images accurately. Produces a trained model and supporting data (such as validation results) that are used by the CaptionGeneration module.

Implemented By: Reading4All

Type of Module: Abstract Object

7.3.2 CaptionGeneration Module (M15)

Secrets: The process the system uses to generate descriptive text from an image. This includes how image features are interpreted and how words are selected and arranged into sentences.

Services: Uses the trained model from the AIModelTraining module to produce alternative text for a given image. Passes the generated caption to the WCAG (W3C (2018)) and InternalMetricCompliance modules to ensure it meets accessibility and clarity standards.

Implemented By: Reading4All

Type of Module: Library

7.3.3 InternalMetricCompliance Module (M16)

Secrets: The internal measures chosen by Team 22 to judge how clear, concise or relevant the generated text is.

Services: Reviews each caption using internal quality checks that complement the AODA (Ontario (2005)) and WCAG 2.1 AA (W3C (2018)) validation standards.

Implemented By: Reading4All

Type of Module: Library

7.3.4 FeedbackMetrics Module (M17)

Secrets: How user feedback is collected, interpreted, and organized for later use in model improvement. The exact method for analyzing or weighting different types of feedback is hidden.

Services: Receives direct input from users in the form of comments, ratings, or corrections about the generated alternative text. Summarizes this information into a structured format that can be used to guide model retraining and future system updates.

Implemented By: Reading4All

Type of Module: Library

7.3.5 FeedbackLoop Module (M18)

Secrets: The rules and internal logic that determine how user feedback is used to update and improve the system's model. The specific approach to retraining or fine-tuning is hidden.

Services: Takes processed feedback data from the FeedbackMetrics module and uses it to retrain or adjust the model over time. This allows the system to gradually improve its accuracy and produce captions that better reflect user expectations.

Implemented By: Reading4All

Type of Module: Abstract Data Type

7.3.6 Feedback Module (M19)

Secrets: How feedback information is stored and organized.

Services: Stores the user's feedback entries and makes them available to the FeedbackMetrics and FeedbackLoop modules for review and improvement purposes.

Implemented By: Reading4All

Type of Module: Record

8 Traceability Matrix

This section shows below how different requirements connect to our modules. This can be seen in Table 3, Table 4, Table 4, Table 5, Table 6, Table 7, Table 8, Table 9, Table 10, Table 11, Table 12, Table 13, Table 14, Table 15, Table 16. It's important to note some requirements do not map to a module because they describe coding practices or infrastructure details.

Finally, Table 17 shows how our modules connect to the anticipated changes two traceability matrices: between the modules and the requirements and between the modules and the anticipated changes.

Req.	Modules
FR1	M1, M6, M8
FR2	M6, M15
FR3	M2, M3, M6, M13
FR4	M6
FR5	M5, M6, M11
FR6	M4, M6

Table 3: Trace Between Functional Requirements and Modules

Req.	Modules
LFR-AR1	M9, M10, M11, M12, M13
LFR-AR2	M10, M9, M11, M12, M13
LFR-AR3	M10, M9, M11, M12, M13
LFR-AR4	M16, M10, M9, M11, M12. M13

Table 4: Trace Between Look and Feel Requirements and Modules

Req.	Modules
LFR-SR1	M9, M10, M11, M12, M13
LFR-SR2	M9, M10, M11, M12, M13
LFR-SR3	M9, M10, M11, M12, M13

Table 5: Trace Between Style Requirements and Modules

Req.	Modules
UHR-EUR1	M2, M3, M9, M10, M11, M12. M13
UHR-EUR2	M9, M10, M11, M12, M13
UHR-EUR3	M9, M10, M11, M12, M13
UHR-EUR4	M9, M10, M11, M12, M13

Table 6: Trace Between Usability and Humanity Requirements and Modules

Req.	Modules
UHR-PIR1	M9, M10, M13

Table 7: Trace Between Personalization and Internationalization Requirements and Modules

Req.	Modules
UHR-LR1	M9, M10, M11, M12, M13

Table 8: Trace Between Learning Requirements and Modules

Req.	Modules
UHR-UPR1	M9, M10, M11, M12, M13

Table 9: Trace Between Understandability and Politeness Requirements and Modules

Req.	Modules
UHR-AR1	M9, M10, M11, M12, M13
UHR-AR2	M9, M10, M11, M12, M13

Table 10: Trace Between Accessibility Requirements and Modules

Req.	Modules
PR-SL1	M15
PR-SL2	M13
PR-SCR1	M5
PR-SCR2	M16
PR-SCR3	M9, M10, M11, M12, M13
PR-SR-HA1	M6, M7, M5, M4
PR-SR-HA2	M6, M7
PR-SR-HA3	M6, M8
PR-PAR1	M15
PR-PAR2	M16
PR-PAR3	M17
PR-RFT1	M8, M6, M7
PR-RFT2	M6, M7
PR-CR1	M5, M4, M6, M7
PR-CR2	No Module
PR-SER1	M14
PR-LR1	M14, M2
PR-LR2	No Module

Table 11: Trace Between Performance Requirements and Modules

Req.	Modules
OER-EP1	No Module
OER-EP2	No Module
OER-WE1	M2
OER-WE2	No Module
OER-IAS1	M1
OER-IAS2	M8
OER-IAS3	M16
OER-PR1	No Module
OER-RL1	M1, M14, M15, M2, M16
OER-RL2	No Module

Table 12: Trace Between Operational and Environmental Requirements and Modules

Req.	Modules
MS-MNT1	M1, M15, M2, M9, M10, M11, M12, M13, M6
MS-MNT2	No Module
MS-MNT3	No Module
MS-SUP1	M7
MS-AD1	M14
MS-AD2	M5,M6, M7
MS-AD3	No Module

Table 13: Trace Between Maintainability and Support Requirements and Modules

Req.	Modules
SR-AR1	M6, M4
SR-AR2	M6, M5
SR-IR1	M6
SR-IR2	M6, M1
SR-PR1	M6
SR-PR2	M16
SR-AU1	M7
SR-AU2	M7
SR-IM1	M8
SR-IM2	M4

Table 14: Trace Between all Security Requirements and Modules

Req.	Modules
CR1	M15
CR2	M16
CR3	M16

Table 15: Trace Between all Cultural Requirements and Modules

Req.	Modules
CR-LR1	M2
CR-SCR1	M6
CR-SCR2	M7

Table 16: Trace Between all Compliance Requirements and Modules

AC	Modules
AC1	M4, M5, M6, M7
AC2	M1, M8,
AC3	M1, M14, M16, M17, M18,
AC4	M2, M3, M15
AC5	M4, M5, M6
AC6	M1, M15
AC7	M9, M10, M11, M12, M13

Table 17: Trace Between Anticipated Changes and Modules

9 Use Hierarchy Between Modules

In this section, the uses hierarchy between modules is provided. Parnas (1978) said of two programs A and B that A *uses* B if correct execution of B may be necessary for A to complete the task described in its specification. That is, A *uses* B if there exist situations in which the correct functioning of A depends upon the availability of a correct implementation of B. Figure 1 illustrates the use relation between the modules. It can be seen that the graph is a directed acyclic graph (DAG). Each level of the hierarchy offers a testable and usable subset of the system, and modules in the higher level of the hierarchy are essentially simpler because they use modules from the lower levels.

[The uses relation is not a data flow diagram. In the code there will often be an import statement in module A when it directly uses module B. Module B provides the services that module A needs. The code for module A needs to be able to see these services (hence the import statement). Since the uses relation is transitive, there is a use relation without an import, but the arrows in the diagram typically correspond to the presence of import statement. —SS]

[If module A uses module B, the arrow is directed from A to B. —SS]

Figure 1: Use hierarchy among modules

10   User Interfaces

[Design of user interface for software and hardware. Attach an appendix if needed. Drawings, Sketches, Figma —SS]

11   Design of Communication Protocols

[If appropriate —SS]

12   Timeline

[Schedule of tasks and who is responsible —SS]
   [You can point to GitHub if this information is included there —SS]

References

Software requirements specification for bridging gaps: Ai for diagram accessibility, 2025. URL https://github.com/4G06-CAPSTONE-2025/Reading4All/blob/main/docs/SRS-Volere/SRS.pdf.

Ontario. Accessibility for ontarians with disabilities act, 2005. https://www.ontario.ca/laws/statute/05a11, 2005. Accessed: 2025-11-10.

David L. Parnas. On the criteria to be used in decomposing systems into modules. *Comm. ACM*, 15(2):1053–1058, December 1972.

David L. Parnas. Designing software for ease of extension and contraction. In *ICSE '78: Proceedings of the 3rd international conference on Software engineering*, pages 264–277, Piscataway, NJ, USA, 1978. IEEE Press. ISBN none.

D.L. Parnas, P.C. Clement, and D. M. Weiss. The modular structure of complex systems. In *International Conference on Software Engineering*, pages 408–419, 1984.

W3C. Web content accessibility guidelines (wcag) 2.1. https://www.w3.org/TR/WCAG21/, 2018. Accessed: 2025-11-10.

W3C. Wai-aria authoring practices 1.2. https://www.w3.org/TR/html-aria/, 2024. Accessed: 2025-11-10.