# Module Guide for Software Engineering

Team #22, TeleHealth Insights
Mitchell Weingust
Parisha Nizam
Promish Kandel
Jasmine Sun-Hu

January 15, 2025

## 1 Revision History

| Date | Version | Member | Notes |
|---|---|---|---|
| 1/13/2025 | 1.0 | Mitchell Weingust | Added 10 - Clinician Dashboard Interfaces |
| 1/14/2025 | 1.1 | Mitchell Weingust | Added 12 - Timeline |
| 1/14/2025 | 1.2 | Mitchell Weingust | Added 10 - Clinician Dashboard FSM |
| 1/14/2025 | 1.3 | Mitchell Weingust | Added 7 - Module Decomposition |

## 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 |
| Software Engineering | Explanation of program name |
| UC | Unlikely Change |
| FSM | Finite State Machine |
| [etc. —SS] | [... —SS] |

## 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 layed 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 hardware on which the software is running.

**AC2:** The format of the initial input data.

...

[Anticipated changes relate to changes that would be made in requirements, design or implementation choices. They are not related to changes that are made at run-time, like the values of parameters. —SS]

### 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.

**UC1:** Input/Output devices (Input: File and/or Keyboard, Output: File, Memory, and/or Screen).

...

## 5 Module Hierarchy

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

**M1:** Clinician GUI Module

**M2:** Parent GUI Module

**M3:** App Controller Module

**M4:** API Gateway Module

**M5:** Authentication Module

**M6:** Result Storage Module

**M7:** Media Processing Module

**M8:** Logging Module

**M9:** Question Bank Module

**M10:** Real-Time Feedback Module

**M11:** Report Generation Module

**M12:** Video Processing Module

**M13:** Audio Processing Module

**M14:** English Question Bank Module

**M15:** Mandarin Question Bank Module

**M16:** Matching Question Bank Module

**M17:** Repetition Question Bank Module

**M18:** Hardware-Hiding Module

...

| **Level 1** | **Level 2** |
|---|---|
| Hardware-Hiding Module | |
| Behaviour-Hiding Module | ? |
| | ? |
| | ? |
| | ? |
| | ? |
| | ? |
| | ? |
| | ? |
| Software Decision Module | ? |
| | ? |
| | ? |

Table 1: 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 Table 2.

[The intention of this section is to document decisions that are made "between" the requirements and the design. To satisfy some requirements, design decisions need to be made. Rather than make these decisions implicit, they are explicitly recorded here. For instance, if a program has security requirements, a specific design decision may be made to satisfy those requirements with a password. —SS]

## 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. *Software Engineering* means the module will be implemented by the Software Engineering 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 (M18)

**Secrets:** The data structure and algorithm used to implement the virtual hardware.

**Services:** Serves as a virtual hardware used by the rest of the system. This module provides the interface between the hardware and the software. So, the system can use it to display outputs or to accept inputs.

**Implemented By:** OS

### 7.2 Behaviour-Hiding Module

**Secrets:** The contents of the required behaviours.

**Services:** Includes programs that provide externally visible behaviour of the system as specified in the software requirements specification (SRS) documents. This module serves as a communication layer between the hardware-hiding module and the software decision module. The programs in this module will need to change if there are changes in the SRS.

**Implemented By:** –

### 7.2.1 Clinician GUI (M1)

**Secrets:** The interactive and visual components that allow Clinicians to interact with the system, through the App Controller (M3), to access patient data and information, and make informed decisions.

**Services:** To show application functionality to clinicians, accepting user inputs (choosing assessments to review, flagging bias questions) and displaying outputs (assessment summaries).

**Implemented By:** [Your Program Name Here]

**Type of Module:** Library

### 7.2.2 Parent GUI (M2)

**Secrets:** The interactive and visual components that allow Parents to interact with the system, through the App Controller (M3), to setup and engage in the assessment with their child.

**Services:** To show application functionality to parents, accepting user inputs (selecting answers to questions, completing setup) and displaying outputs (question visuals, button selections).

**Implemented By:** [Your Program Name Here]

**Type of Module:** Library

### 7.2.3 Etc.

## 7.3 Software Decision Module

**Secrets:** The design decision based on mathematical theorems, physical facts, or programming considerations. The secrets of this module are *not* described in the SRS.

**Services:** Includes data structure and algorithms used in the system that do not provide direct interaction with the user.

**Implemented By:** –

### 7.3.1 AppController Module (M3)

**Secrets:** The interactions between the GUIs (M1, M2) and the API Gateway (M4), acting as a means to interface with the software modules.

**Services:** Enables the user to pass information from the GUIs to the backend services.

**Implemented By:** [Your Program Name Here]

**Type of Module:** Library

### 7.3.2 API Gateway Module (M4)

**Secrets:** The interactions between the App Controller (M3) and the inter-dependencies of all other software modules, including inherited modules ( M6, M7, M8, M9, M10, M11, M13, M14, M15, M16, M17).

**Services:** Enables the user to access the system and interact with its components, consisting of the Patient, Client, and Admin views.

**Implemented By:** [Your Program Name Here]

**Type of Module:** Library

## 8 Traceability Matrix

This section shows two traceability matrices: between the modules and the requirements and between the modules and the anticipated changes.

| Req. | Modules |
|---|---|
| R1 | M18, M??, M??, M?? |
| R2 | M??, M?? |
| R3 | M?? |
| R4 | M??, M?? |
| R5 | M??, M??, M??, M??, M??, M?? |
| R6 | M??, M??, M??, M??, M??, M?? |
| R7 | M??, M??, M??, M??, M?? |
| R8 | M??, M??, M??, M??, M?? |
| R9 | M?? |
| R10 | M??, M??, M?? |
| R11 | M??, M??, M??, M?? |

Table 2: Trace Between Requirements and Modules

| **AC** | **Modules** |
|---|---|
| AC1 | M18 |
| AC2 | M?? |
| AC?? | M?? |
| AC?? | M?? |
| AC?? | M?? |
| AC?? | M?? |
| AC?? | M?? |
| AC?? | M?? |
| AC?? | M?? |
| AC?? | M?? |
| AC?? | M?? |
| AC?? | M?? |

Table 3: 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]

The interface below depicts the initial interface a clinician would see upon logging into their account in the system.

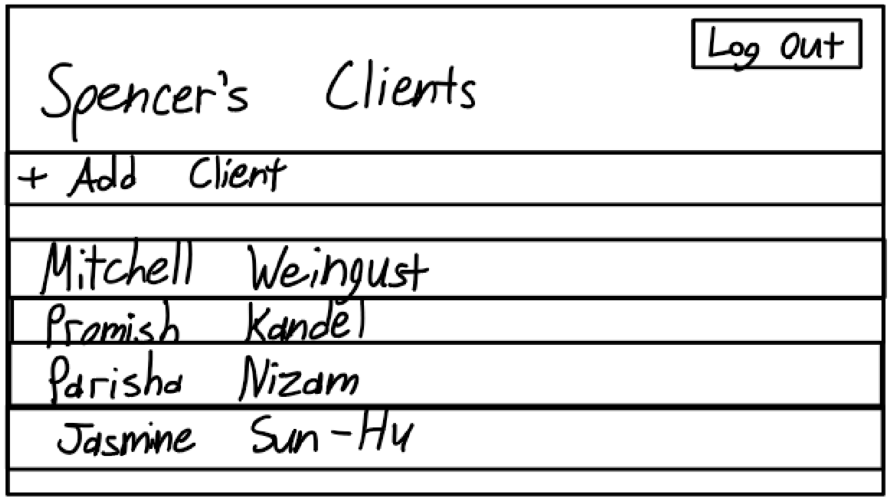

Figure 2: Clinician Dashboard

The interface below depicts the interface a clinician would see upon selecting the Add Client button on the previous Clinician Dashboard screen.

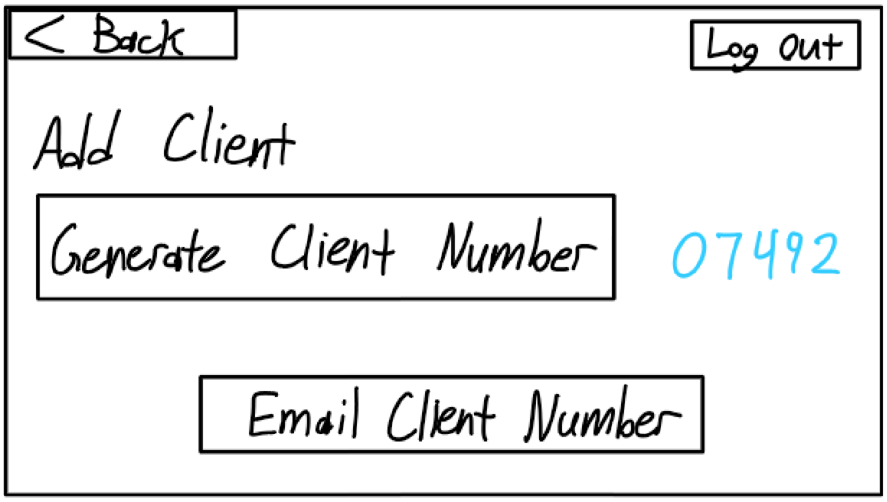

Figure 3: Add Client

The interface below depicts the patient overview, which can be reached from the Clinician Dashboard by selecting a name from the client list.

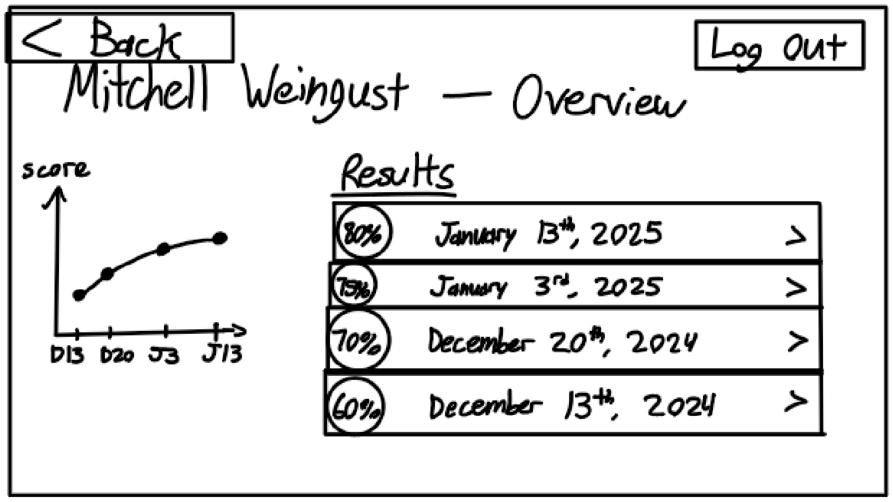

Figure 4: Patient Overview

The interface below depicts the patient assessment results analysis, which can be reached from the Patient Overview by selecting an assessment date from the list of assessments.

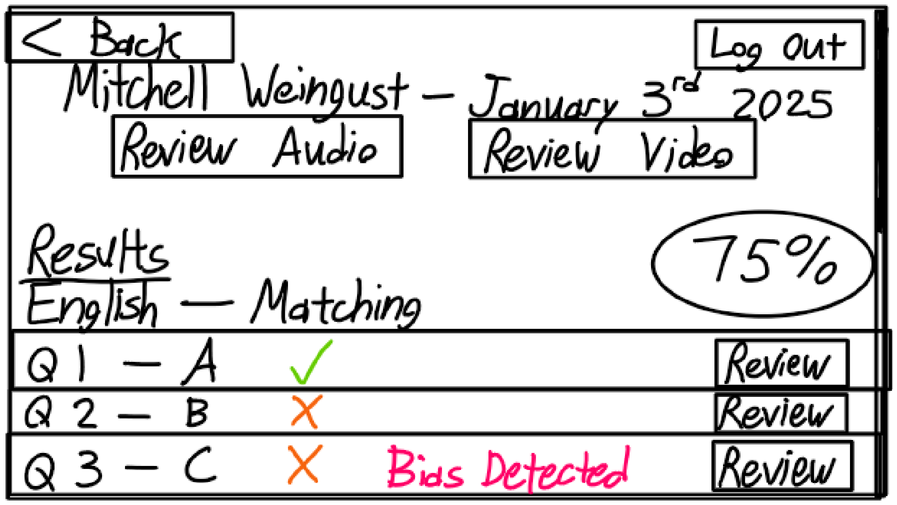

Figure 5: Patient Assessment Results Analysis (1)

The interface below depicts a continuation of the patient assessment results analysis, which can be reached from the previous figure, by scrolling the scrollbar on the right edge of the screen.

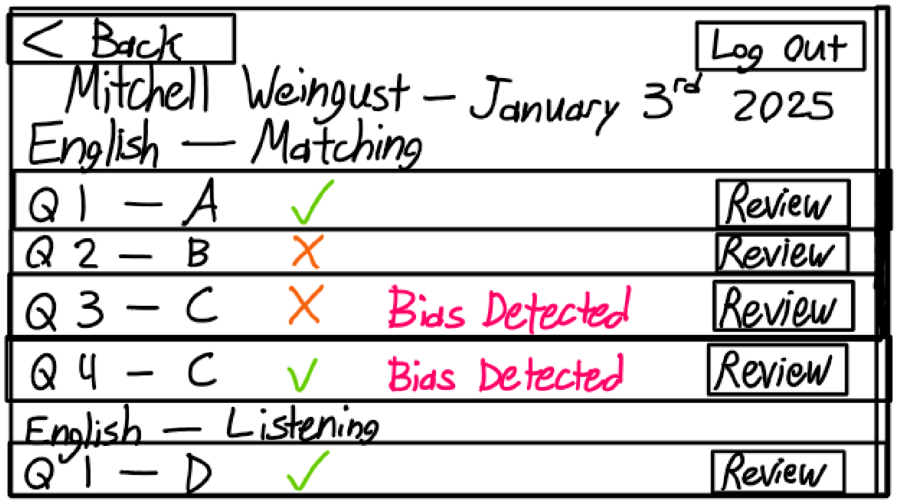

Figure 6: Patient Assessment Results Analysis (2)

The interface below depicts the bias review, which can be reached from the Patient Assessment Results Analysis by selecting Review on any of the questions on an assessment.

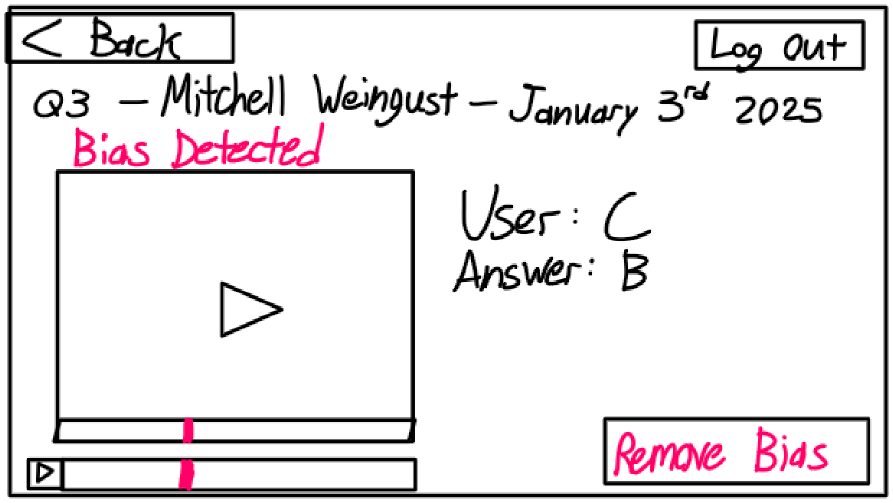

Figure 7: Bias Review

The interface below depicts a question review page, where no bias has been detected. The ability to Flag Bias is present in the bottom right corner, to give the Clinician the ability to manually reflect bias in a question.

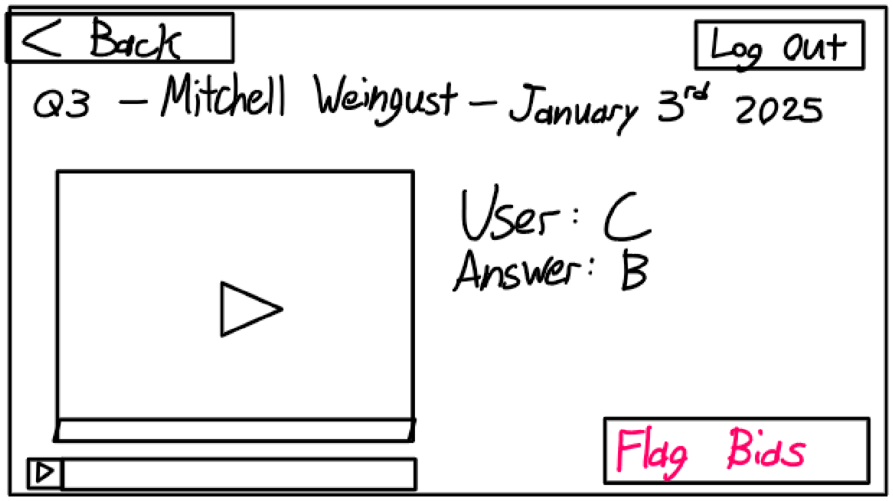

Figure 8: Flag Bias

The below finite state machine depicts how the clinician can interface with the dashboard, as well as which interactions lead to changes in states in the system.

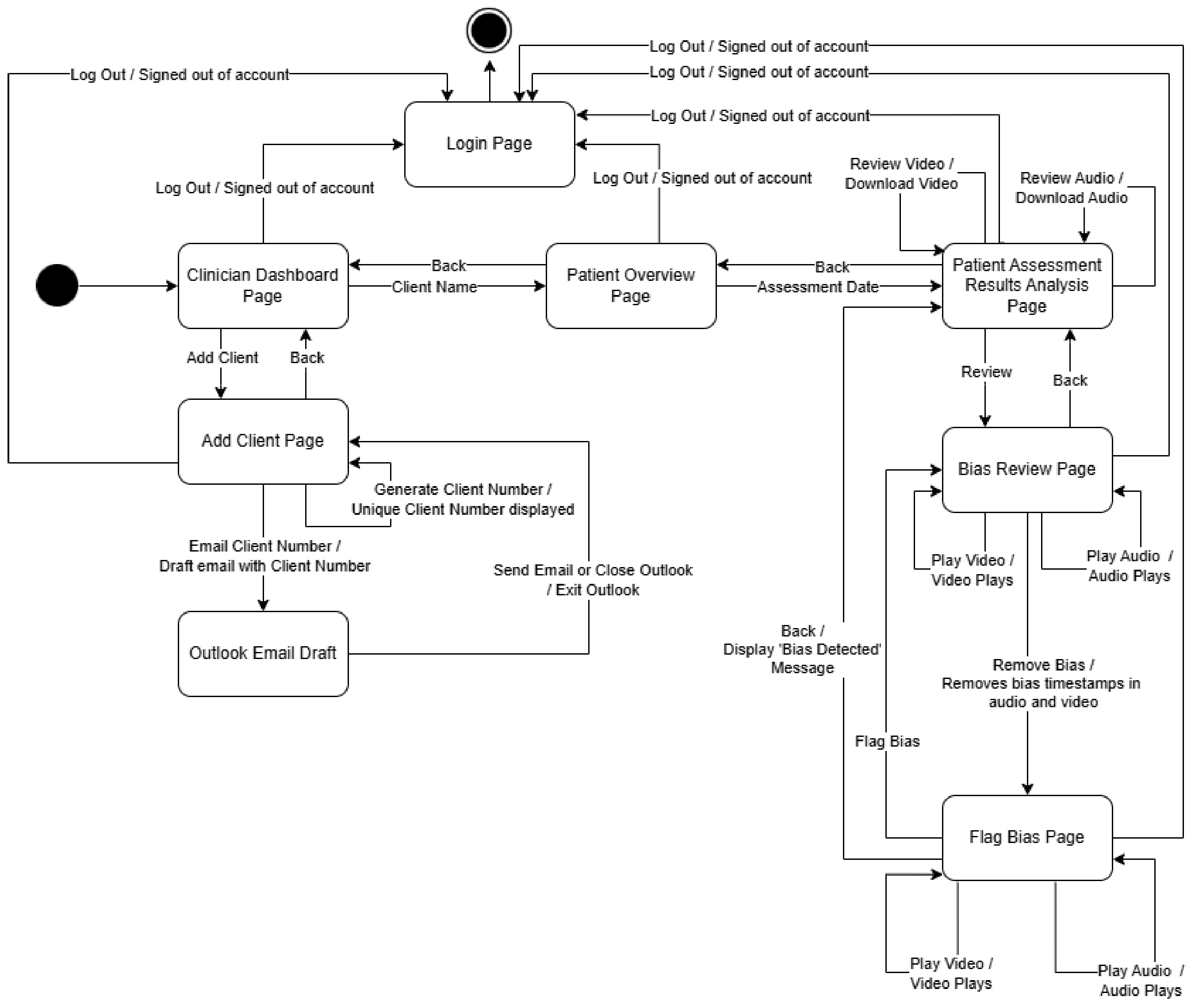
Login Page
Clinician Dashboard Page
Patient Overview Page
Patient Assessment Results Analysis Page
Add Client Page
Outlook Email Draft
Bias Review Page
Flag Bias Page
Log Out / Signed out of account
Back
Client Name
Assessment Date
Review Video / Download Video
Review Audio / Download Audio
Add Client
Generate Client Number / Unique Client Number displayed
Email Client Number / Draft email with Client Number
Send Email or Close Outlook / Exit Outlook
Review
Play Video / Video Plays
Play Audio / Audio Plays
Back / Display 'Bias Detected' Message
Remove Bias / Removes bias timestamps in audio and video
Flag Bias

Figure 9: FSM - Clinician Dashboard

## 11 Design of Communication Protocols

N/A

## 12 Timeline

| Milestone | Module/Pages | Objective | Mitchell | Parisha | Promish | Jasmine | Date |
|---|---|---|---|---|---|---|---|
| Assessment | Question Bank Module | | X | X | | | 1/19/25 |
| Assessment | English Question Bank Module | | X | X | | | 1/19/25 |
| Assessment | Matching Question Bank Module | | X | X | | | 1/19/25 |
| Assessment | Repetition Question Bank Module | | X | X | | | 1/19/25 |
| Assessment | | Verification and Validation Testing | X | X | X | X | 1/19/25 |
| Controllers | API Gateway | | | | X | | 1/19/25 |
| Controllers | App Controller | | | | X | | 1/19/25 |
| Controllers | | Verification and Validation Testing | X | X | X | X | 1/19/25 |
| Assessment GUI | Assessment Selection Page | | | | | X | 1/19/25 |
| Assessment GUI | Parent Checklist Page | | | X | | | 1/22/25 |
| Assessment GUI | Input Check Page | | | | | X | 1/22/25 |
| Assessment GUI | Assessment Questions Page | | X | | | | 1/22/25 |
| Clinician Dashboard | Result Storage Module | | | | X | | 1/22/25 |
| Assessment GUI | Assessment Instructions Page | | | X | | | 1/25/25 |
| Assessment GUI | Tutorial Page | | X | | | | 1/25/25 |
| Assessment GUI | Assessment Completion Page | | | | | X | 1/25/25 |
| Assessment GUI | | Verification and Validation Testing | X | X | X | X | 1/25/25 |
| Clinician Dashboard | Report Generation Module | | | | X | | 1/25/25 |
| Clinician Dashboard | | Verification and Validation Testing | X | X | X | X | 1/25/25 |
| Clinician Dashboard GUI | Clinician Dashboard Overview Page | | X | | | | 1/28/25 |
| Clinician Dashboard GUI | Patient Overview Page | | | X | | | 1/28/25 |
| Clinician Dashboard GUI | Patient Assessment Results Analysis Page | | | | | X | 1/28/25 |
| Media Processing | Media Processing Module | | | | X | | 1/28/25 |
| Clinician Dashboard GUI | Bias Review Page | | | | | X | 1/31/25 |
| Clinician Dashboard GUI | Add New Client Page | | X | | | | 1/31/25 |
| Clinician Dashboard GUI | | Verification and Validation Testing | X | X | X | X | 1/31/25 |
| Homepage | Authentication Module | | | | X | | 1/31/25 |
| Homepage GUI | Select Account Type Page | | | X | | | 1/31/25 |
| Homepage GUI | Login Page (Parent) Page | | | X | | | 2/3/25 |
| Homepage GUI | Login Page (Clinician) Page | | | X | | | 2/3/25 |
| Homepage GUI | Create Account Page | | X | | | | 2/3/25 |
| Homepage GUI | Homepage (Parent) Page | | | | | X | 2/3/25 |
| Homepage GUI | | Verification and Validation Testing | X | X | X | X | 2/3/25 |
| Media Processing | Video Processing Module | | | | X | | 2/3/25 |
| Media Processing | Audio Processing Module | | X | | | | 2/6/25 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Media Processing | | Verification and Validation Testing | X | X | X | X | 2/6/25 |
| Miscellaneous | Logging Module | | | X | | | 2/6/25 |
| Miscellaneous | Real-Time Feedback Module | | | | X | | 2/6/25 |
| Miscellaneous | | Verification and Validation Testing | X | X | X | X | 2/6/25 |
| Admin | Add Clinician Page | | | | | X | 2/6/25 |
| Admin | | Verification and Validation Testing | X | X | X | X | 2/6/25 |
| Rev0 | | Full System Testing | X | X | X | X | 2/8/25 |
| Rev0 | | Rev0 Practice | X | X | X | X | 2/9/25 |
| Rev0 | | Rev0 Presentation | X | X | X | X | 2/10/25 |

## References

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.