**Clario: A Web-Based Real-Time Transcription Support Platform**

**Software Requirements Specification**

*Team Members:*

Cayleigh Leishman

*Description of project:*

Clario is a web-based platform designed to provide live transcription of lectures for Deaf and Hard of Hearing students, ensuring accessibility and equity in learning environments. Transcribers provide live text transcription that students can view in real time, while admins manage users, lectures, and live sessions. The platform emphasizes real-time performance, privacy, accessibility, and historical access to transcripts.

This project is especially meaningful to me because it represents the type of platform I wish I could have used during my own time in college. Clario supports seamless notetaking with time-stamped transcripts, ensures privacy by offering one-way audio and video, and prevents accidental microphone issues by transcribers. It also provides administrators—such as accessibility departments—with tools to oversee sessions and ensure a smooth, reliable connection for students who depend on it.

**Section 1:**

Technologies Used:

| **Category** | **Tool / Technology** | **Purpose / Function** |
| --- | --- | --- |
| **Frontend** | **SvelteKit** | **Provides a reactive, component-based framework for building a fast, interactive web application.** |
|  | **HTML5, CSS3, JavaScript/TypeScript** | **Standard web technologies for layout, styling, and client-side functionality.** |
| **Backend / Server** | **Supabase** | **Handles authentication, database management (PostgreSQL), and real-time data updates.** |
|  | **Node.js (via SvelteKit endpoints)** | **Serves API routes, handles server-side logic, and connects frontend with the database.** |
| **Database & Data Management** | **PostgreSQL (via Supabase)** | **Stores user profiles, lectures, enrollment, transcriptions, session data, and notes.** |
|  | **Realtime Database / Supabase Realtime** | **Supports live updates for transcription chunks in WebRTC sessions.** |
| **Real-Time Communication** | **WebRTC** | **Enables secure, low-latency streaming of audio and video from lecture sessions to students without sharing transcriber audio/video.** |
|  | **Session Tokens & Authentication** | **Ensures secure, role-based access to live sessions for students, transcribers, and admins.** |
| **Additional Tools & Libraries** | **UUID** | **Generates unique IDs for database records and session tokens.** |
|  | **Vite** | **Optimizes the frontend development workflow.** |
|  | **Git / GitHub** | **Version control for collaborative development and deployment tracking.** |
|  | **Postman / HTTPie** | **Testing APIs and database endpoints during development.** |
| **Accessibility & UX Tools** | **Color contrast and theme selection** | **Improves visibility for visually impaired users.** |
|  | **Keyboard navigation & screen reader support** | **Ensures accessibility for users with disabilities.** |

**Problem Statement:**

While attending BYU-I, I experienced challenges with existing transcription services. Sometimes, transcribers had their audio on without realizing it, which compromised their privacy, and as a deaf student, I couldn’t detect this. Additionally, transcribers had to remember to manually send transcripts to students after class, creating extra workload and potential delays. Students, meanwhile, struggled with disorganized or incomplete notes, making it difficult to follow lectures effectively. Existing solutions did not provide a unified platform that addressed these challenges for both students and transcribers.

**Solution:**

Clario solves these problems by providing an integrated platform that combines real-time transcription, audio/video streaming, and note-taking in a single web application. For students, it delivers accurate, structured transcriptions and allows hands-on note-taking without the need for additional software. For transcribers, Clario ensures privacy by streaming audio one-way only to students, eliminating the risk of accidentally broadcasting their own audio or video. Additionally, all transcriptions are automatically saved and linked to students’ accounts, removing the need for transcribers to manually distribute notes after class. This unified platform streamlines lecture participation, protects transcriber privacy, and ensures students receive clear, organized transcripts.

**Section 2a: Must Have Requirements**

| Requirement | Description | Acceptance Criteria | Hours Expected | Due Date |
| --- | --- | --- | --- | --- |
| 1. User Authentication & Role Dashboards | Users can log in as Student, Transcriber, or Admin. Role-based dashboard displayed. | Successful login shows correct dashboard. | 16 | Oct 18 |
| 2. Student Join Lecture via Join Code | Students can join lecture sessions using a code. | Enter code → system confirms → student joins session. | 12 | Oct 25 |
| 3. Real-Time Transcription Streaming | Transcriber input streams live to students. | Typed text instantly visible to all students. | 20 | Nov 1 |
| 4. Final Transcript Storage | Session transcripts saved post-lecture. | Transcript viewable in student dashboard. | 16 | Nov 15 |
| 5. Student Note-Taking | Students can add personal notes during session. | Notes saved & retrievable later. | 16 | Nov 29 |
| 6. Accessibility Features | Support keyboard navigation, text resizing, high contrast. | Students can navigate without mouse and adjust text size/colors. | 12 | Dec 6 |

**Section 2b: Stretch Requirements**

| **Goal** | **Description** | **Acceptance Criteria** | **Hours Expected** |
| --- | --- | --- | --- |
| 1. Multi-Admin Support | Multiple admins manage users & sessions. | Two admins perform actions without conflict. | 8 |
| 2. Admin Session Management | Admins assign transcribers, view logs. | Admin dashboard shows correct session setup. | 10 |
| 3. Multi-Transcriber Support | More than one transcriber in same session. | Students see combined text. | 8 |
| 4. Real-Time Notifications | Students receive reminders or alerts. | Students see notification without page refresh. | 8 |

**Section 2c:** **Schedule**

| **Requirement** | **Description** | **Acceptance Criteria** | **Hours** | **Due Date** |
| --- | --- | --- | --- | --- |
| User Authentication & Role-Based Dashboard | Allow users to sign up and log in as Student, Transcriber, or Admin. Display correct dashboard based on role. | Users log in and are directed to the appropriate dashboard without errors. | 16 | Oct 18 |
| Student Join Lecture via Join Code | Students enter a code to join a live session. | Student enters code → system confirms → student joins session successfully. | 12 | Oct 25 |
| Real-Time Transcription Streaming | Transcriber input appears live to all connected students. | Text typed by transcriber appears immediately for all students in session. | 20 | Nov 1 |
| Final Transcript Storage | Merge all real-time chunks into a final transcript after session ends. | Transcript stored in database and accessible in student dashboard. | 16 | Nov 15 |
| Student Note-Taking | Students can add personal notes alongside live transcription. | Notes are saved and retrievable after the session. | 16 | Nov 29 |
| Accessibility Features | Keyboard navigation, text resizing, color contrast adjustments. | Students can navigate without mouse, resize text, and apply high-contrast theme. | 12 | Dec 6 |

**Grand Total (Core Features):** 126 hrs

**Stretch Goal Hours (Optional):** 34 hrs

A diagram of a system

AI-generated content may be incorrect.**Section 3:** Overview of the Product.

**A diagram of a process

AI-generated content may be incorrect.*Workflow****:*   
● **Scheduling (Admin):** The admin sets up lectures. They choose a transcriber for each lecture and make a join code. The transcriber sees the schedule on their dashboard. The student gets the join code to join the lecture.

● **Starting the Session (Transcriber):** The transcriber clicks on their scheduled lecture to start it. This opens the session and a secure channel for live transcription.

● **Joining the Session (Student):** The student enters the join code or clicks “Join Lecture.” If they are new, the system signs them up automatically. They are then connected to the live session.

● **Live Transcription:** The transcriber types what is said in the lecture. Students see it in real time. More than one transcriber can work together if needed.

● **Taking Notes (Student):** Students can write their own notes while reading the live transcript.

● **Privacy:** Only students see the transcriber’s words. Transcribers audio and video will not stream to the student.

● **Ending the Session:** When the lecture is finished, all typed text is combined into a final transcript. Students can read it right away with their personal notes.

● **Accessibility:** Students can make text bigger, change colors, or switch between light and dark mode. They can also use only the keyboard to move around the site.

● **Admin Control:** The admin manages schedules, making it easy for transcribers and safe for students. This keeps everything organized and private.

***Resources*:**

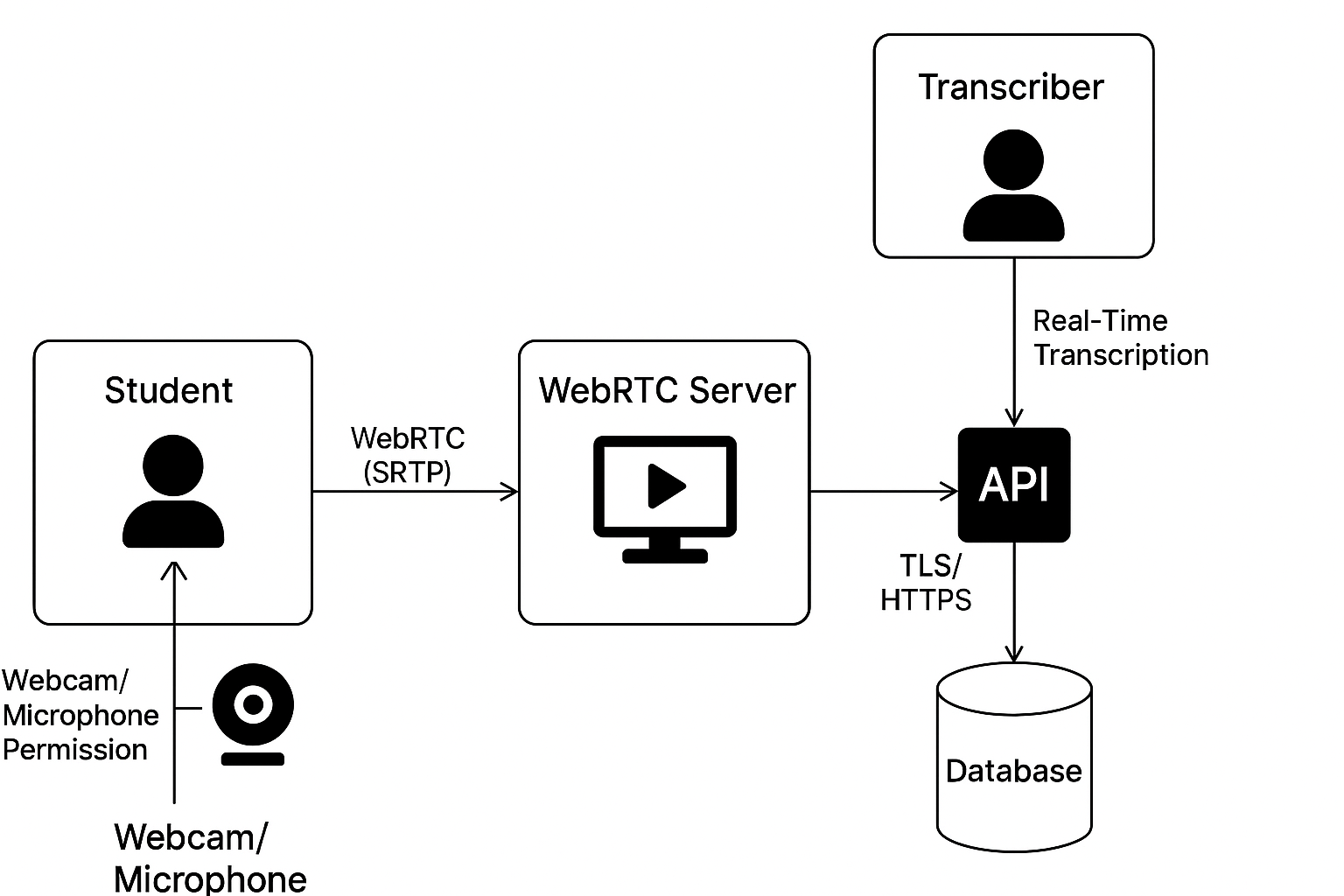
| **Component** | **Tool / Technology** | **What It Does (Simple)** |
| --- | --- | --- |
| **User Interface** | SvelteKit, HTML, CSS, JavaScript | Loads pages fast and shows updates in real time |
|  | Tailwind CSS | Keeps design consistent and nice on all devices |
| **Server & Logic** | SvelteKit + Node.js | Connects frontend to database and handles live sessions |
|  | Vite | Helps build and update website faster |
| **Development & Testing** | Git / GitHub | Tracks changes and helps teamwork |
|  | Postman / HTTPie | Tests system to make sure it works |
| **Live Session** | WebRTC | Sends lecture audio and video to students quickly |
|  | Session Tokens | Only authorized users can join |
|  | Privacy | Transcriber audio/video is private |
| **Database & Storage** | Supabase / PostgreSQL | Stores users, lectures, enrollments, live & final transcripts, notes |
|  | UUIDs | Creates unique, hard-to-guess IDs |
|  | Encryption | Keeps passwords and sensitive info safe |
| **Accessibility** | Frontend Features | Change text size, color, contrast; keyboard navigation; screen reader support |

**A diagram of a course

AI-generated content may be incorrect.Data at Rest**The database is structured to manage users, courses, sessions, and transcription data efficiently. It uses auth.users (from Supabase) to handle authentication, while the profile table stores user details and roles. Student\_enrollment links students to lectures in course\_lecture, and student\_note stores notes taken by students. Live sessions are tracked in active\_sessions, with transcription data recorded both in realtime\_chunks for immediate viewing and in final\_transcription for post-session reference. This structure ensures clear relationships between users, lectures, and transcription data, supporting both real-time updates and secure data storage.

**Data on the Wire**

During live sessions, students join and grant permission for their webcams and microphones, allowing transcribers to view them. WebRTC streams one-way audio and video from the lecture, while transcription chunks are sent to the database in real time. All data is secured with TLS/HTTPS and SRTP, and session tokens ensure only authorized students can access the live stream and transcriptions. This setup provides low-latency communication, real-time updates, and secure data delivery.



**Data State**Clario data progresses through four states: **Pre-Session**, where lectures, enrollments, and profiles exist; **Session Start**, when an active session is created and a token is issued; **Live Session**, with real-time transcription and WebRTC streaming; and **Post-Session,** when final transcriptions are stored and the session is archived. This ensures efficient management, real-time updates, and data integrity throughout the session lifecycle.

**HMI/HCI/GUI**

Clario’s interface is designed to be **intuitive, accessible, and responsive**, ensuring that both students and transcribers can interact with the platform effectively.

**Admin Interface (Stretch Goal):**

* Assign transcribers to lectures.
* Monitor active sessions and student attendance.
* Access logs of completed sessions and transcripts.

**Transcriber Interface:**

* **Session Dashboard:** Shows lectures they are assigned to, along with active session status and student participation.
* **Live Transcription Pane:** Allows real-time transcription input without risk of broadcasting their personal audio or video.
* **Session Management Tools:** Start/end sessions, monitor student connections, and manage multiple transcribers in collaborative sessions.
* **Automated Transcript Handling:** All transcriptions are automatically stored in students’ accounts after the session.

**Student Interface:**

* **Lecture Dashboard:** Students see a list of upcoming lectures with join codes and status (live, upcoming, completed).
* **Live Lecture View:** A single-page layout where students can:
  + View **real-time transcription** in a scrolling pane.
  + Take **personal notes** in a side panel.
  + Watch **lecture video/audio** (one-way from the transcriber).
  + Adjust **accessibility settings** such as text size, color contrast, and theme (light/dark mode).
* **Final Transcripts:** After a session ends, students can access merged transcripts with search and download options.

**Accessibility Features:**

* Keyboard navigation for all major actions.
* Screen reader support for visually impaired users.
* Adjustable text size and color themes for readability.
* High contrast mode for low-vision users.

**Pictures**

A diagram of a flowchart

AI-generated content may be incorrect.

A diagram of a website

AI-generated content may be incorrect.

**Mockup Ideas:***wireframes:  
For All Roles:*

Header and Footer:

A whiteboard with writing on it

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.Transcript View (Created using Figma Make):

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer screen

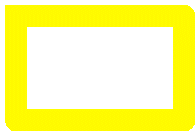
AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.



A screenshot of a whiteboard

AI-generated content may be incorrect.

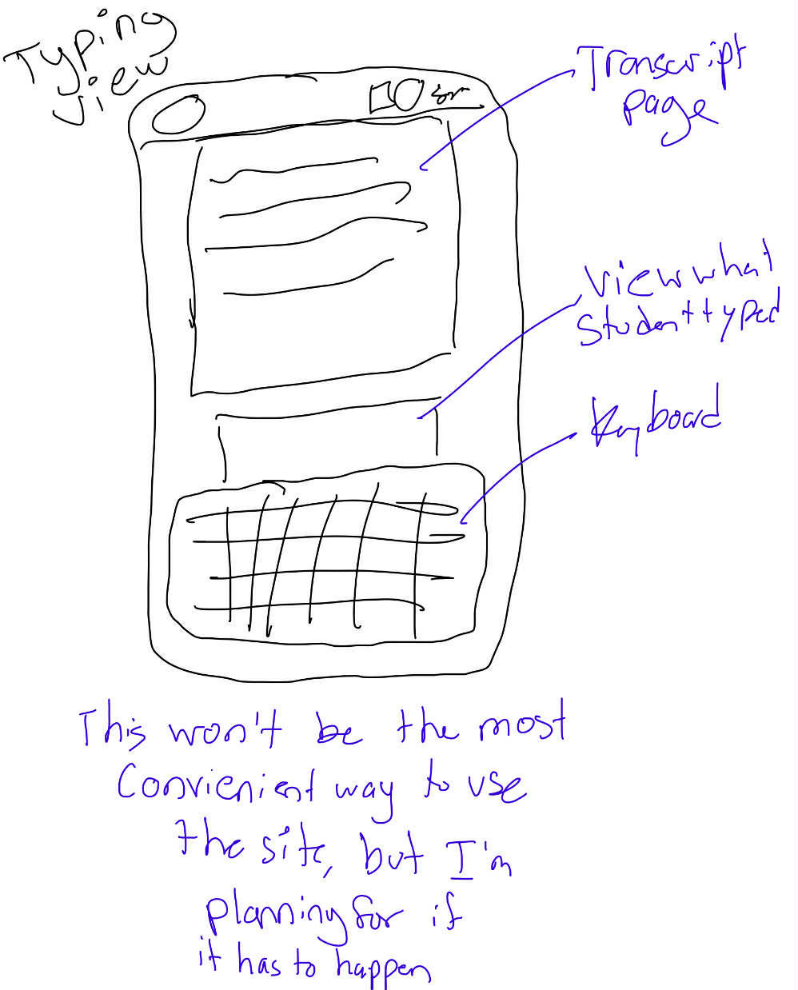
A sketch of a computer screen

AI-generated content may be incorrect.

A white paper with black text and a list of tasks

AI-generated content may be incorrect.

A drawing of a computer

AI-generated content may be incorrect.

**Section 4:** Verification:

**Demo / Functional Validation**

Conduct live sessions with a transcriber and students to ensure:

* Students can join sessions using a join code.
* Real-time transcription streams correctly to all connected students.
* Students’ personal notes are saved and retrievable.
* Final transcripts are stored and accessible after the session.

**Testing**

| **Testing Type** | **Focus / What is Tested** | **Expected Outcome / Validation** |
| --- | --- | --- |
| **Unit Testing** | Authentication and role-based dashboards | Users see correct dashboard based on role |
|  | Lecture join functionality | Students can join a lecture session using the code |
|  | Transcription input/output handling | Typed text appears correctly for students |
|  | Note-saving and retrieval functions | Notes are saved and retrievable after session |
| **Integration Testing** | End-to-end module interaction | Session creation → student joining → real-time transcription → note saving → transcript storage works seamlessly |
|  | Multi-transcriber streaming (stretch goal/future) | Combined input visible to students (future enhancement) |
| **Accessibility Testing** | Keyboard navigation | Users can navigate the platform without a mouse |
|  | Screen reader compatibility | Transcription and interface elements readable via screen readers |
|  | Adjustable text size, color contrast, high-contrast mode | Users can adjust settings to improve readability |
| **Security Testing** | Session tokens | Unauthorized users cannot join sessions |
|  | Encryption (TLS/HTTPS, SRTP) | Data in transit is secure |
|  | Privacy of transcriber audio/video | Students cannot access transcriber’s personal audio/video |
| **Demonstration / Peer Review** | Functional testing with instructor/peers | Students can join sessions, see live transcription, save notes, access final transcripts; feedback collected on usability and clarity |
| **Performance Testing** | Real-time streaming | Updates occur with low latency under different network conditions; multiple students supported |

**Sources/Citation/Resources**:

**Sources / References**The following resources were consulted during the Design and implementation of this project

**Tutorials Followed**

* **Accessibility for Teams: Front-End Development** – [digital.gov](https://digital.gov/guides/accessibility-for-teams/front-end-development)
* **SvelteKit Authentication with Supabase** – [YouTube Tutorial](https://www.youtube.com/watch?v=lSm0GNnh-0I&t=9s.96)
* **ChatGPT** – [OpenAI](https://chat.openai.com/)
* **Gemini** – [Gemini](https://gemini.google.com/)
* **Supabase Documentation** – [Supabase Docs](https://supabase.com/docs)
* **Coding Tutorials** – [W3Schools](https://www.w3schools.com/)
* **Microsoft Edge Accessibility Tips** – [Microsoft Learn](https://learn.microsoft.com/en-us/microsoft-edge/accessibility/design)
* **Front-End Frameworks: Evaluating Accessibility** – [University of Michigan](https://accessibility.umich.edu/how-to/design-development/front-end-frameworks)
* **ADA Web Guidance** – [ADA.gov](https://www.ada.gov/resources/web-guidance/)

**Readings / Books Utilized**

**Accessibility**

* **Introduction to Web Accessibility** – [W3.org](https://www.w3.org/WAI/fundamentals/accessibility-intro/)
* **Different Types of Adaptive Switches for Individuals With Disabilities** – [Enabling Devices](https://enablingdevices.com/blog/different-types-of-adaptive-switches-for-individuals-with-disabilities/?srsltid=AfmBOor0bOoZn6GwoxEsx7uM7GWnH0X744uKdV2Uo7soqo6LuFXHs421)
* **Ways to Improve Web Accessibility** – [Acquia](https://www.acquia.com/blog/ways-to-improve-web-accessibility)
* **Accessibility for Teams: Front-End Development** – [digital.gov](https://digital.gov/guides/accessibility-for-teams/front-end-development/)
* **Video Introduction to Web Accessibility and W3C Standards** – [W3.org](https://www.w3.org/WAI/videos/standards-and-benefits/)
* **WebAIM – Web Accessibility in Mind** – [WebAIM](https://webaim.org/projects/million/)
* **MDN ARIA Reference** – [MDN](https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA/Reference/Attributes/aria-label)
* **Accessibility Canada –** Accessibility Handbook on Accessible Web Design – [PDF](https://accessibilitycanada.ca/wp-content/uploads/2018/10/AccessAbility-Handbook-on-Acc.-Web-Design.pdf)
* **ARIA Patterns: Read This First** – [W3.org](https://www.w3.org/WAI/ARIA/apg/patterns/)
* **ARIA Practices with Code** – [W3.org](https://www.w3.org/WAI/ARIA/apg/practices/read-me-first/)
* **Prefers Contrast Media Query** – [MDN](https://developer.mozilla.org/en-US/docs/Web/CSS/@media/prefers-contrast)

**Database**

* **What Are Relational Databases** – [YouTube](https://www.youtube.com/watch?v=OqjJjpjDRLc)
* **How to Normalize a Database** – [YouTube](https://www.youtube.com/watch?v=siiYInWniFs)
* **The Manga Guide to Databases** – Mana Takahashi Shoko Azuma TREND-PRO Co., Ltd [PDF](https://oberstar.eu.org/share/Documents/The-Manga-guide-to-databases.pdf)

**Design**

* **The Design of Everyday Things (Revised & Expanded Edition)** – Don Norman – [JND.org](https://jnd.org/books/the-design-of-everyday-things-revised-and-expanded-edition/)

**Security**

* **Penetration Testing** – Georgia Weidman – [PDF](https://www.kea.nu/files/textbooks/humblesec/penetrationtesting.pdf)
* **Web Security for Developers** – Malcolm McDonald – [PDF](https://www.kea.nu/files/textbooks/humblesec/websecurityfordevelopers.pdf)

Additional Resources will be documented as the Project Progresses

You can see updates by visiting <https://github.com/CayleighLeishman/Clario/blob/main/developerNotes/DeveloperNotes.md>