⚙️ Technical Specification

Title	Kontinua Tech Spec
Status	COMPLETE
Authors	@mosandle @jkschies @dchadley @Ethan Handelman @lrodri99
Team	Readers
Reviewers	@BJ Klingenberg
Version	2.0
Creation date	Nov 14, 2024
Last Updated	Mar 19, 2025
Jira Issue link	Kontinua - Team Project | Backlog
Repository	GH repo

Table Of Contents

1. Introduction

a. Overview, Problem Description, Summary, or Abstract

The **Kontinua Foundation** aims to create a high-quality, self-paced STEM learning experience for students, focusing on underrepresented groups and those without access to advanced coursework. To complement its existing series of workbooks and quizzes, the foundation requires a **reader app** for iOS and Android tablets. This app will allow users to engage with digital resources and track progress interactively. The app should prioritize user experience, especially on iPads, due to their wider adoption in education, while ensuring compatibility with Android tablets.

b. Glossary or Terminology

General Terms:

- **STEM**: Acronym for Science, Technology, Engineering, and Mathematics. Refers to academic disciplines and educational focus areas.
- **Self-Paced Learning**: An educational approach allowing students to learn at their own speed, independent of a fixed schedule.
- **Prerequisite Hopping**: The ability to navigate to earlier foundational material needed to understand a current topic.
- **Pomodoro Timer**: A time management technique using 25-minute focused intervals (Pomodoros) followed by short breaks, visualized here with a color-changing indicator.

Technical Terms:

- **Cache**: A local storage mechanism used by applications to store frequently accessed data (e.g., downloaded PDFs) for quick retrieval.
- **Native Development**: Building applications using platform-specific programming languages and tools (e.g., Swift for iOS, Kotlin for Android) to achieve optimal performance and user experience.
- **AWS**: Amazon Web Services, a cloud computing platform used to host web services and backends.
- **UIKit**: A framework used in iOS development for building and managing graphical user interfaces.
- **Swift**: A programming language for iOS and macOS development.
- **Kotlin**: A programming language for Android development.

App-Specific Features:

- **Bookmarks**: A feature that tracks the last visited or frequently accessed sections of a workbook.
- **Scribble Functionality**: Allows users to annotate directly on digital pages, storing these annotations locally on the device.
- **Prerequisite Navigation**: Links embedded within the workbook allowing users to jump to prerequisite chapters or concepts.
- **Digital Resources**: Supplementary materials (e.g., videos, interactive quizzes) associated with specific workbook chapters, accessible within the app.
- **Feedback Submission**: A feature enabling users to provide suggestions or report issues for specific chapters.

User Experience (UX) Terms:

- ~~**Landscape Mode**: A horizontal orientation of the screen, offering better visibility for users with small screens or visual impairments.~~ (No longer planned)
- **Zoom Functionality**: The ability to magnify or reduce content size for improved readability.
- **Page-Based Navigation**: A user experience design where content is navigated by discrete pages, rather than continuous scrolling.

Educational Context:

- **First-Tier University**: Highly regarded institutions typically offering rigorous STEM programs.
- **Workbooks**: Educational resources combining instructional content with exercises to practice learned concepts.

c. Context or Background

- **Mission:** The foundation's mission is to prepare students for rigorous first-year STEM programs at top-tier universities by providing continuous learning materials.
- **Current state:** A quiz-authoring web app exists, but there is no mobile platform for consuming the workbook content.
- **Target audience:** Young students, particularly those lacking access to advanced coursework.
- **Technology priorities:** Native development is required to maximize performance and user experience, with Swift and Kotlin as the chosen languages for iOS and Android apps, respectively. Backend services will be hosted on AWS.
- **Timeline:** The iPad version must be completed and available on the App Store by the end of the school year.

d. Goals or Product and Technical Requirements

To deliver an **intuitive, feature-rich reading app** for iOS and Android tablets that helps students engage with STEM workbooks and supplementary resources in a seamless and productive way. The app should support learning through interactivity, progress tracking, and feedback features while maintaining high performance and reliability.

Product Requirements (User Stories)

Cache Management:

1. **As a student**, I want to download workbooks so that I can read them offline without an internet connection.
2. **As a student**, I want the app to check for updates to downloaded workbooks no more than once a week so that I can access the latest content without excessive internet usage.

Progress and Bookmarking:

3. **As a student**, I want the app to save my reading progress so that I can continue where I left off.
4. **As a student**, I want the app to track the amount of time I spend on each page so that I can monitor my study habits.

Pomodoro Timer:

5. **As a student**, I want a timer with visual feedback (e.g., a red-to-green bar) so that I can stay focused during 25-minute study sessions.

Digital Resources:

6. **As a student**, I want to browse digital resources linked to a specific chapter so that I can deepen my understanding of the material.

Prerequisite Hopping:

7. **As a student**, I want to jump to prerequisite chapters so that I can review foundational concepts.

Search:

8. **As a student**, I want to search the entire series, including undownloaded chapters, so that I can quickly find relevant content.

Feedback:

9. **As a student**, I want to submit feedback or suggestions for specific chapters so that I can contribute to improving the content.
10. **As the app owner**, I want feedback submissions to include a verified email address so that I can identify and respond to legitimate suggestions.

Scribbling:

11. **As a student**, I want to annotate or scribble directly on the workbook pages so that I can highlight important content or jot down notes.

Navigation and Viewing:

12. **As a student**, I want to navigate the workbooks by pages instead of scrolling so that I can read content in a book-like format.
13. **As a student**, I want to zoom in and out of pages so that I can focus on small details or fit an entire page into view.
14. ~~**As a student**, I want to use landscape mode to view half-pages when I need better readability.~~

Technical Requirements

Compatibility:

1. The system shall run on **iOS tablets (iPad)** and **Android tablets** only, prioritizing iPads.
2. The system shall use **Swift (UIKit)** for iOS development and **Kotlin** for Android development.
3. The app shall not support phones, desktop devices, or older tablet OS versions that are no longer updated.

Hosting:

4. The backend shall be hosted on **Amazon Web Services (AWS)** to ensure scalability and reliability.
    a. Current AWS EC2 instance future changes possible
5. The system shall require no installation on customer-owned servers.
6. Simple SMTP service for the Feedback on the Backend web service

Performance:

6. The app shall **cache workbook content locally** to reduce reliance on the internet.
7. The app shall query the backend no more than once a week to validate or update cached content.

User Interface:

8. The app shall provide a **responsive and intuitive interface** for both iOS and Android platforms.

Accessibility:

10. The app shall provide **zoom functionality** and ~~**landscape mode**~~ to accommodate users with small screens or vision impairments.

Security:

11. The system shall securely handle **user feedback** submissions, ensuring that email addresses are validated and stored in compliance with data protection regulations.
12. Scribbles and annotations shall be stored **locally on the user's device** to protect user privacy.

Open Source:

13. All code for the app and backend shall be publicly released on GitHub under an open-source license.

e. Non-Goals or Out of Scope

- Adding the web quizzes to the reader app
- ~~The app shall support **landscape modes**, with smooth transitions between viewing modes.~~

f. Future Goals

- Integration with other devices like iPhones or MacBooks
- Integration with Apple Pencils.
- integrate WebApp Quizzes directly into the readers

g. Assumptions

- Students will have access to iPads and Android tablets
- Students will want to use these workbooks
- Students will take advantage of the features offered by this platform.
- Teachers will show their students this platform.

🏁 2. Solutions

a. Current or Existing Solution / Design

The current solution for the Kontinua Sequence are printed out versions of the workbook with a Webapp that allows students and teachers to create quizzes for each chapter.

Pros	Cons
• **Physical Tangibility**: Printed workbooks provide a tangible, easy-to-use resource that doesn't rely on digital access, which can be ideal for younger students or those in areas with limited technology. • **Ease of Use**: Students and teachers can directly annotate printed pages, making it simple to highlight and take notes. • **No Tech Learning Curve**: Teachers and students who prefer traditional methods can engage with the material without needing to learn new technologies. • **Webapp Customization**: The ability to create quizzes for each chapter provides flexibility and adaptability to different classroom needs.	1. **Frequent Reprints**: Each update to the workbook requires a new print run, which can be costly, environmentally unfriendly, and logistically cumbersome. 2. **Limited Interactivity**: Printed versions lack the interactive potential of digital resources, such as videos, dynamic problem-solving tools, or immediate feedback. 3. **Access Issues**: Students and teachers without access to a printer may face difficulties replacing lost or damaged materials. 4. **Scalability**: Adapting the workbook to new pedagogical methods or standards may require significant effort, especially if it involves constant reprints and reformatting.

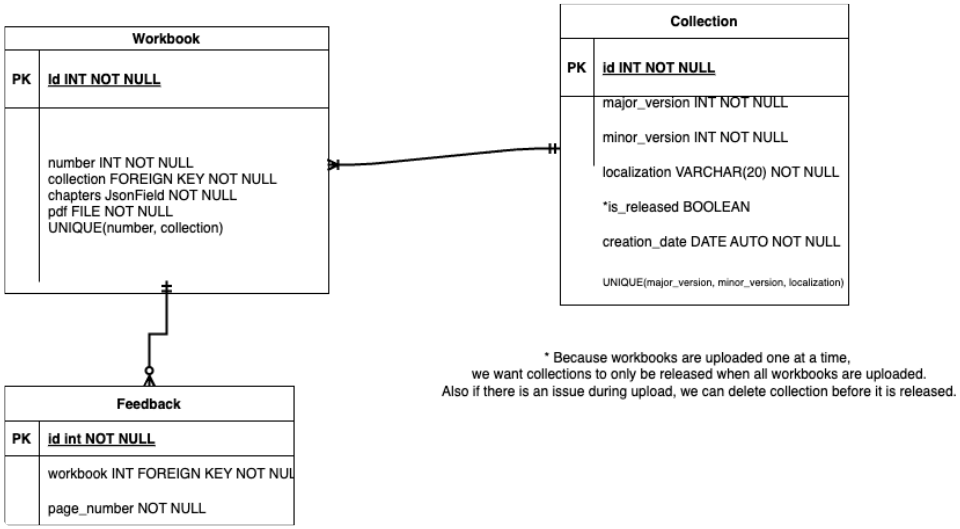
	5. **Storage and Portability**: Physical books can be bulky and require storage space, whereas digital solutions are easier to carry and organize.

b. Suggested or Proposed Solution / Design

	Option 1	Option 2
Description	Existing PDF viewer	Custom PDF viewer
Pros and cons	➕ Quick solution ➖ Many unnecessary features and not focused on Kontinua Sequence	➕ Fully customizable to stakeholder and user needs ➖ In-house development
Estimated cost	SMALL	MEDIUM

The custom pdf viewer will depend on the Mentoris Django app, to process the feedback to the same database

- **Data Model / Schema Changes *(or the new one if it's a new product)***



-
- **Business Logic**
  - Workbook Management
    - i. Download Management
      - Validate local cache against server version (maximum once per week)
      - Download new workbook content when updates are available
      - Store workbooks locally for offline access
      - Handle cleanup of unused cached content
    - ii. Progress Tracking
      - Record time spent on each page
      - Save last viewed position in each workbook
      - Track completion status of chapters
      - Persist reading progress across sessions
    - iii. Pomodoro Timer Logic
      - Initialize 25-minute study sessions

- Track active/break states
- Calculate and update visual indicators (red-to-green progress)
- Handle timer interruptions and resets

iv. Navigation System

- Handle prerequisite chapter linking
- Manage page-based navigation
- Process zoom levels and landscape mode transitions
- Track and restore navigation history

v. Annotation System

- Store scribbles and annotations locally
- Associate annotations with specific page coordinates
- Handle different input methods (touch, stylus)
- Manage undo/redo operations

vi. Search Functionality

- Index downloaded content for local search
- Query server for global search across all workbooks
- Handle search result ranking and relevance
- Cache frequent search results

vii. Feedback System

- Validate email addresses
- Send email to foundation email account
- Queue feedback submissions
- Handle offline feedback storage
- Sync with server when connection available

viii. Resource Management

- Track digital resource availability
- Handle resource downloads and caching
- Manage resource updates
- Control resource access permissions

◦ State Management

i. Application States

- Active reading
- Download in progress
- Search active
- Timer running
- Annotation mode
- Offline mode

ii. Data Persistence

- Local storage management
- Cache invalidation rules
- Progress synchronization
- Annotation backup

iii. Error Handling

- Network connectivity issues
- Download failures

        - Storage capacity limits
        - Invalid content format
        - Synchronization conflicts
    iv. Performance Optimization
        - Lazy loading of content
        - Resource preloading
        - Cache management
        - Memory usage optimization
        - Battery usage considerations
  - Integration Points
    i. Backend Services
        - Content delivery system
        - Feedback submission
        - Search service
        - Update checking
        - Analytics collection
    ii. Device Integration
        - Screen orientation handling
        - Storage management
        - Network state monitoring
        - Battery state awareness
- **Presentation Layer**
- Lo-Fi
  - lofi kontinua
  - Kontinua UI Mock
- Hi-Fi:
  - Kontinua Mock
- Changes:
  - Went to a more simplistic design, with majority being text instead of icons
- Mobile Concerns:
  - Don't overlap with native navigation, swipe bar on bottom
- Web Concerns:
  - N/A
- **Other questions to answer**
  - How will the solution scale?
    - The solution should scale well - it is almost all executed locally, as it is just displaying workbooks. All other features require no heavy lifting by the backend, so they will scale 100% fine. As we continue to develop and work out the backend more robustly, this portion may change.
  - What are the limitations of the solution?
    - The only limitations to the solution are whichever limits exist on the current platform it being developed on, and the limitations of the workbooks themselves.
  - How will it recover in the event of a failure?
    - In the event of a failure, the solution can be re-optimized in order to address any problems that occurred, due to how the code functions together. We are looking into an option to "Revert" your textbook if you get an updated version that breaks anything.

- How will it cope with future requirements?
  - Most features work together seamlessly, so the solution is able to handle most if not all future integrations/requirements. Most current features do not rely heavily on each other and can be updated and changed without major issue.

c. Test Plan

In our current state we have no minimum test coverage bar, other than ensuring that the code compiles properly on main and all pull requests. We will have more information on all of our testing plans as we continue development and figure out what we need to test fully.

- ~~Unit tests (not yet implemented)~~
- Integrations tests on Backend(Automated through GitHub Actions)
- Linter/Formatter reviews:
  - iOS: (Automated with SwiftLint and SwiftFormat using GitHub Actions)
  - Kotlin: (Automated with ktlint integrated with Spotless for linting and Spotless for formatting using GitHub Actions)
  - ~~Kotlin Jetpackcompose: (Not implemented yet)~~
- ~~Other types of tests (e.g., End-to-end tests or Acceptance tests, Exploratory tests, Penetration tests, etc.)~~ (not yet implemented)
- Github Actions has integrated a "build" for iOS and Android so that we know all code can compile.
- There will be a lot of personal user testing by clicking buttons and stress testing specific features as human beings. This will give us a good idea of what our user will do, and how they will do it.
- After determining pain points/most commonly used features, simulated UI tests will be created using XCTest's built in UI recorder (similar to Selenium/Cypress for JavaScript webapps)

d. Alternate Solutions / Designs

- Other alternative solutions were fairly similar save for a couple UI differences which included different menu designs, different tool bar functions/looks, as well as different menus.
- Since both the solution and the alternatives would work relatively similarly, none of their functionality are that different.
- The alternative solutions were disliked by the customer, so we tried to cater to their vision instead
- Most of the UI designs came across as too much and were seen as being distracting for the potential users, so they were toned down in the proposed solution.
- Most of the code would be still useful in any alternative and we would just have to work on any new UI designs as well as adding or removing any features that affected the proposed solution.

Note: This section refers to alternate solutions to be implemented by the team and not alternate solutions already provided either in-house, in the market, or opensource.

✅ 3. Further Considerations

a. Security considerations

- Honestly this is extremely low security risk, which is great. It does not involve passwords, user data, or anything else sensitive. It does involve working with children who are generally insecure, but because there is so little information to share it should be okay.

b. Privacy considerations

- Our app collects some data from the user, like how long they spend on each page, but does not warn them it is collecting this data in a clear way. There will be a small warning button, but the user may not read it thoroughly. This is not a huge concern because the data is so low level that there is not much risk associated with the collection or lack of informing, because it contains no personal identifying information.
- Collecting OS version and type of device for the feedback is a privacy concern we are curretnly looking into with the stakeholder.

c. Regional considerations

- Internationalization and localization require the app to support multiple languages (through different textbook versions) . This will require efficient region/language file handling on AWS. Latency issues can occur when users are far from hosting servers, making CDNs and region-specific data centers essential, although this should be through AWS and offline caching. Legal concerns include compliance with data regulations like GDPR, obtaining user consent, and adhering to copyright laws. Reliable service availability, especially in areas with intermittent internet access, may require offline content caching (which will be implemented in the app).

d. Accessibility considerations

- Our tool is designed to be used by children that are underprivileged and do not have access to powerful STEM resources in their classrooms. It is deployed on iPads, which means at a minimum these children need to have tablets available to them. Due to the target audience of this app, they may not all have this tablet access we require. This will certainly limit the accessibility of our app, but there is not much way to get around this.

e. Operational considerations

- There should not be many adverse aftereffects caused by the Kontinua reader app, as it is mainly just displaying textbooks that have already been written. The only data that would need to be recovered in case of a failure would be the user's annotations saved in the app. This could be accomplished by saving backups, or allowing for export of the annotations. Operational costs will probably already be exceedingly low as serving PDFs is generally a lightweight server task and there is a unlikely ammount of feedback traffic.

f. Risks

- There are not many risks with this solution, besides investing our time and energy into creating the app. Perhaps one risk is limiting the devices to only iPads and tablets, but this can easily be changed by creating apps for other platforms as well.

🤔 4. Deliberation

a. Discussion

- How will the workbook information be given to us through the json?
- Do we set up a new backend or integrate it with the old code from last year?
- How will we handle workbooks as they get updated and page ID's change?

b. Open Questions

- How soon is the Android app expected to be completed?
- How do we want the data collection warning to be written, and where should it be displayed?
- Will feedback be integrated into the existed Django quiz app, or will a new, more streamlined backend be created to host the workbooks and accept feedback?

📚 5. References and Acknowledgements

- Kontinua: https://www.kontinua.org
- XCode Testing: https://www.hotovo.com/blog/testing-with-xcode

📄 6. Documentation

- Backend Overview: https://github.com/KontinuaFoundation/readers/blob/main/docs/backend/overview.md Connect your G ithub account
- API reference: https://github.com/KontinuaFoundation/readers/blob/main/docs/backend/api-reference.md Connect your G ithub account