Codex Prompt: Upgrade BookWords from Wireframe to Better MVP Foundation

You are working inside my existing MINERVA Next.js GitHub repo. I already have a first BookWords React/Next.js clickable wireframe at `/bookwords`. It is decent as a wireframe, but it is not yet a useful MVP. I want you to improve it into a much more polished, simpler, more functional MVP foundation while keeping the work contained and safe.

BookWords is a mobile-first app for parents who are reading physical books with their children. The parent uses the phone to quickly capture an unknown word from the printed page, then the app creates a flashcard. The core experience is: parent and child are reading a physical book together, child does not know a word, parent opens BookWords, points the camera at the word, captures it, sees a small confirmation, closes the phone, and continues reading. The phone must disappear quickly.

The app should feel like an Apple-quality mobile app, not a web form inside a phone frame. It should be simple, polished, parent-focused, and calm. It should not feel like a kid game.

Safety, GitHub, Secrets, and Setup Rules

Do not ask me for passwords or pasteable secrets in the chat.

Do not ask me to paste:

- GitHub password
- Supabase service role key
- Google Cloud credentials JSON
- OpenAI API key
- Apple Developer credentials
- Apple Pay secrets
- Stripe keys
- USPS or address provider API keys
- Any `.env` values

If you need a connection, credential, API key, Apple Developer setup, Google Cloud setup, Supabase setup, or OAuth configuration, stop and ask a clear setup question. Tell me exactly what account or service needs to be connected and what environment variable names you expect. Do not request the secret value itself inside the chat.

Use environment variables and document required setup. If a service is not configured yet, provide a mock fallback so the app still runs locally.

Work on a branch or isolated changes. Do not commit directly to main unless I explicitly ask. If possible, use or recommend a branch name like:

```
feature/bookwords-mvp-foundation
```

Before changing files:

1. Inspect the repo structure.
2. Determine whether the project uses App Router or Pages Router.
3. Use the routing pattern already present.
4. Identify existing Supabase and OpenAI patterns.
5. Keep all BookWords work contained.
6. Do not break existing MINERVA reader, library, auth, profile, or Supabase behavior.

After changes:

1. Summarize files created or changed.
2. Explain how to run and view `/bookwords`.
3. List supported click paths.
4. List what is real and what is mocked.
5. List required setup for Google Cloud Vision, Supabase sync, Apple/Google sign-in, address autocomplete, and PDF/screenshot export.
6. List assumptions and limitations.

Product Name

Working product name: **BookWords**.

BookWords is currently built inside the existing MINERVA repo, but it should look and function as a standalone product.

Core Product Principle

The parent-child reading experience is sacred.

BookWords must not become the experience. The physical book and the parent-child reading moment are the experience. The app should support that moment and then disappear.

The user should feel:

> *I can capture this unknown word in two seconds and get back to reading.*

Primary User

The parent is the primary user. The child is reading the physical book and should not be pulled into a screen-time workflow.

Do not design BookWords like a kid app. Do not use gamification. Do not add badges, streaks, coins, avatars, noisy animations, or childish visuals.

Non-Negotiable Rules

1. The app should open to the camera/capture experience.
2. No login before first capture.
3. Parent is the primary user.
4. Child stays focused on the physical book.
5. Capture must be extremely fast.
6. Camera stays live after photo.
7. Confirmation should behave like iPhone Camera: small bottom-left chip/thumbnail.
8. OCR should work most of the time. Manual entry is only a fallback.
9. High-confidence cards become Ready automatically.
10. Wrong word capture is not acceptable as a normal outcome.
11. Needs Review exists only as a backup path.
12. Definitions are fixed, not fluid.
13. Official definitions are read-only and sourced.
14. Free digital review on phone.
15. Paid physical 4x6 card flow can be polished but mocked for now.
16. Photos are temporary and deleted after card approval.
17. No notifications in MVP.
18. The UI should feel native to iPhone and Apple-quality.

Major Design Changes from Current Wireframe

Navigation

Replace the current top-button navigation with a fixed bottom tab bar.

Use exactly three tabs:

1. **Library**
2. **Camera** - prominent center action button
3. **Account**

Do not add a fourth tab for Review, Cart, or Order in this iteration. Review and Order should live inside Library.

The bottom tab bar should be fixed across the app, including the camera screen. The center Camera button should be visually dominant, circular, and premium, similar to modern iPhone apps with a center action button.

Remove these top camera controls:

- X
- Account
- Cards
- Flash

The camera screen should not have duplicate navigation. The bottom tab bar is the navigation system.

Visual Design Direction

Move beyond black-and-white wireframe quality. Make it polished, simple, and Apple-like.

Use:

- Apple-like neutral palette: white, black, soft grays, and restrained blue accent.
- System font for UI.
- A refined literary serif only for the main vocabulary word on flashcards and physical card previews.
- Clean spacing.
- Rounded cards and native mobile patterns.
- Apple Notes-style simplicity with some Apple Wallet-style polish for card rows/previews.

Avoid:

- Kid colors
- Fantasy/Harry Potter branding
- Heavy gradients
- Decorative styling
- Browser-default form controls
- Web-looking dropdowns
- Admin-table layouts
- Noisy animations

Important: BookWords should not feel like a web form inside a phone. It should feel like a native mobile product.

Platform Direction

Keep this version inside the existing Next.js MINERVA repo at `/bookwords`. Do not start a separate React Native or Expo app yet.

However, structure the code so it can later migrate to React Native / Expo:

- modular components
- mobile-first layout
- native-feeling interactions
- clear service boundaries
- separate camera, OCR, dictionary, AI, auth, persistence, and order logic
- avoid unnecessary web-only UI patterns

At the end, include a short migration note explaining what would need to change to move BookWords from Next.js to Expo/React Native for App Store release.

Required Output from This Codex Task

Produce:

1. A polished clickable React/Next.js mobile prototype at `/bookwords`.
2. Individual PNG screenshots for each major screen/state.
3. One combined PDF containing all screens in user journey order.
4. A Figma-ready design spec.
5. A developer setup note for services that need configuration.

If fully automated PNG/PDF export is not practical, provide a script or clear command path for generating the screenshots and combined PDF.

Organize the PDF in user journey order:

1. Camera
2. Capture confirmation
3. Card Library
4. Card Detail
5. Edit / Regenerate
6. Mobile Flashcard Review
7. Order Physical Cards
8. Checkout
9. Account / Settings

The Figma-ready design spec should include:

- screen inventory

- component inventory
- typography rules
- color palette
- spacing rules
- bottom tab bar behavior
- click paths
- screen states
- reusable components

Camera Screen Requirements

The Camera tab is the heart of the app.

Implement real device camera access now.

The Camera screen should:

- Request camera permission when needed.
- Display the live camera feed.
- Allow image capture.
- Keep the live camera active after capture.
- Show the fixed bottom tab bar.
- Have no top navigation buttons.
- Have no Flash button in this iteration.
- Have no Account button.
- Have no Cards button.

If camera permission is denied or unavailable:

- Show a fallback state.
- Keep manual word entry available.
- Use a clean sample book-page background if available.
- Explain that camera access is needed to capture words from books.
- Do not make the entire app useless; manual entry should still work.

Use this camera guidance overlay:

- One large outer focus box.
- One smaller horizontal word rectangle inside it.

The inner shape must be a horizontal rectangle, not a square, because printed words appear on lines.

Manual word entry should remain available but should be a fallback, not the normal workflow. The intended workflow is Google OCR works most of the time.

Capture Flow

After the user taps capture:

1. Do not navigate away.
2. Do not show a processing page.
3. Do not show a blocking modal.
4. Keep the camera live.
5. Immediately show a small bottom-left iPhone-Camera-style chip:

```
Captured
```

Then run OCR in the background.

If OCR succeeds:

```
Saved: craning
```

If OCR confidence is low:

```
Needs Review
```

Do not interrupt the reading flow.

OCR Requirements

Use Google Cloud Vision OCR as the intended OCR provider.

Build both:

1. The real Google Cloud Vision OCR integration path.
2. A safe mock fallback when credentials are missing.

Architecture:

```
Camera UI
-> capture image
-> send image to internal server-side API route
-> server route calls Google Cloud Vision OCR
-> server route returns detected word and confidence
-> client shows Captured / Saved: word / Needs Review
```

Rules:

- Google Cloud credentials must never be exposed client-side.
- Do not hard-code secrets.
- Do not ask me to paste credentials into Codex.
- Use environment variables and document setup steps.
- If credentials are missing or OCR fails in development, return a mock OCR result such as `craning` so the app remains usable.
- Keep OCR modular so another provider can be swapped later.

OCR should focus first on the small horizontal word rectangle, not the full page.

After capture:

1. Crop or prioritize the area inside the word-target rectangle.
2. Run OCR on that region first.
3. If confidence is high, save the detected word.
4. If confidence is low, create a Needs Review card or allow manual entry without disrupting the camera flow.
5. Keep full pointed-word/finger-detection logic modular for later.

Use an initial OCR confidence threshold of **80%**.

A captured word can become Ready only if:

- OCR confidence is at least 80%.
- The detected word is found in the dictionary source.
- The detected word has a clean word shape.

Clean word shape means letters only or normal word punctuation. No broken OCR fragments, random symbols, numbers, or garbage text.

If any check fails, mark the card **Needs Review**.

Dictionary Requirements

Use **Free Dictionary API** as the first public dictionary source for MVP definition lookup.

The dictionary service should be modular so future sources can be added later, especially:

- Noah Webster's 1828 American Dictionary of the English Language
- Merriam-Webster's Collegiate Dictionary

Official definitions are read-only.

Important line:

> *Definitions are fixed, not fluid.*

Do not let the parent manually edit the official definition.

The official definition must include a source label.

If the dictionary lookup fails:

- Do not invent an official definition and present it as sourced.
- Mark the card Needs Review.
- Show a clear fallback state.

AI Requirements

Use AI now for kid-friendly explanations.

Inspect the existing repo for any current OpenAI integration. If an OpenAI setup already exists, reuse the existing pattern rather than creating a duplicate integration.

Create or use a contained server-side API route for AI generation. Do not call OpenAI directly from client-side code. Do not expose API keys in the browser.

The AI should generate:

1. Kid-friendly explanation.
2. Book-context meaning only if a full sentence is confidently available.
3. Pronunciation fallback if dictionary source does not provide it.
4. Synonyms fallback if dictionary source does not provide them.

Do not ask me to paste OpenAI API keys into Codex. Use existing environment variables if already configured. If needed, document required environment variables without exposing secrets.

Kid-friendly explanation rules:

- Generate from the sourced dictionary definition.
- Do not invent the official meaning.
- Label it as a kid-friendly explanation, not the official definition.
- Target age: 10 / 4th grade by default.
- Use child age/grade profile if available.

Book Context Rules

The most important outcome is capturing the correct word. Book context is secondary.

Generate book-context meaning only if OCR confidently extracts the full sentence around the word.

If the full sentence is not confidently available:

- Leave book-context meaning blank, or
- Mark it as `Needs sentence context`, but
- Do not block card creation.

Do not invent sentence context. Do not fabricate a book sentence. Do not generate a context explanation if the extracted sentence is incomplete or uncertain.

If a context explanation is generated, it must preserve the correct subject and pronouns.

Example: In the Harry Potter sentence, Mrs. Dursley is craning her neck. The context meaning must say she, not he.

Correct direction:

> *In this sentence, "craning" means Mrs. Dursley was stretching her neck over the garden fences so she could spy on the neighbors.*

Manual Word Entry

Manual word entry is a fallback, not the primary workflow.

The normal workflow should be:

```
Parent points camera at word
-> Google Cloud Vision OCR identifies the word
-> app creates the card automatically
```

Manual entry is only for when:

- OCR fails
- OCR confidence is low
- OCR detects the wrong word
- The parent wants to quickly correct something

Manual entry should use the fastest possible flow.

When the parent manually types a word and taps Save:

1. Immediately create the card.
2. Look up the word in the dictionary.
3. If dictionary match succeeds and word shape is clean, mark Ready.
4. If dictionary match fails, mark Needs Review and show a subtle message such as:

```
Could not verify this word. Check spelling.
```

Do not add an extra spelling confirmation step. Manual entry is already intentional.

Card Status Logic

Use only these statuses:

- Needs Review
- Ready
- Ordered

A card becomes Ready automatically only when:

1. OCR confidence is at least 80%.
2. The word is found in the dictionary source.
3. The word shape is clean.

A card becomes Needs Review when:

- OCR is low confidence.
- Dictionary lookup fails.
- Word shape is suspicious.
- Sentence/context is uncertain and context is needed.
- Parent flags it during review.

Ordered is normally applied after the mocked checkout flow. Do not make Ordered a prominent manual status choice.

Image Storage and Privacy

Use a hybrid temporary image-storage model.

Captured images may be temporarily stored while OCR, dictionary lookup, AI generation, or parent review is still in progress.

Rules:

- Before sign-in, use local temporary image storage.
- After sign-in, use Supabase Storage only when needed for unresolved Needs Review cards.
- If OCR succeeds and the card becomes Ready, automatically delete the captured image after card data has been created.
- If OCR confidence is low or card is Needs Review, keep the image temporarily so the parent can inspect/correct it.
- Once the parent resolves the card and marks it Ready, automatically delete the image.
- Do not store images permanently.
- Do not show BookWords as a photo archive.

- Do not attach page photos to finished flashcards.

The image exists only to extract the word and create the card.

Data and Supabase Requirements

Use local-first behavior before sign-in.

Before sign-in:

- The parent can use the camera.
- The parent can capture words.
- The parent can run OCR.
- The parent can create temporary/local cards.
- The parent can review cards on the phone.

After Apple or Google sign-in:

- Sync local cards to Supabase.
- Use Supabase for persistent storage going forward.
- Backup/order flows require sign-in.

Do not require sign-in before first capture.

Do not create username/password login.

Use only:

- Sign in with Apple
- Sign in with Google

Do not show Amazon, Meta, email/password, or custom username/password in this version.

Before making Supabase schema changes:

1. Inspect existing Supabase integration.
2. Inspect existing migration patterns if present.
3. Add contained BookWords-specific tables only if appropriate.
4. Do not break current MINERVA schema or auth behavior.
5. If the migration path is unclear, propose the SQL first and ask me before applying it.

Minimal card data should include fields like:

- id
- user_id nullable until synced
- word

- official_definition
- definition_source
- kid_friendly_explanation
- book_context_meaning nullable
- extracted_sentence nullable
- pronunciation_text
- synonyms array
- status
- ocr_confidence
- source_image_path temporary nullable
- created_at
- updated_at
- ordered_at nullable

Account Screen Requirements

Redesign Account as a clean mobile settings screen reachable from the bottom tab bar.

Do not show a long generic list of buttons.

Use sections:

1. Sign In
2. Child Profile
3. Shipping
4. Payment / Checkout preference
5. Backup / Export - lower priority or future utility

Sign In:

- Sign in with Apple
- Sign in with Google
- No username/password
- No manual parent name/email field

Parent name/email should come from auth or checkout email.

Child Profile:

- Child name
- Child age

- Child grade

Split age and grade into separate fields.

If parent enters age for the first time, auto-suggest grade once:

- age 8 -> 2nd grade
- age 10 -> 4th grade
- age 12 -> 6th grade

After the parent manually changes grade, do not override it again.

Remove from MVP Account screen:

- Order history
- Privacy row
- Prominent non-working Back up my cards button

Backup/export can exist lower on the screen only if it has a clear mocked or working path. Future idea: export cards to CSV/Excel and let parent email or download them.

Shipping Address

Implement real U.S.-only shipping address autocomplete/validation for the order flow.

Use a provider abstraction so the address service can be swapped later.

Do not hard-code secrets. Do not ask me to paste API keys into Codex. Use environment variables and document setup.

Address UX should:

- Limit shipping to the United States.
- Autocomplete as the parent types.
- Validate street, city, state, and ZIP.
- Save the selected shipping address to the order flow.

If provider credentials are missing, show a mocked autocomplete fallback so the app remains usable locally.

Payment

For this iteration, do not implement real Apple Pay or real payment processing.

Use a polished Apple Pay-style button as a mocked UI element only.

The button should look native and premium, but it must not charge cards or connect to a real payment processor.

When tapped, it should proceed to a mocked order confirmation.

Clearly mark payment as mocked in code comments or developer notes, but do not make the customer-facing UI look fake or unfinished.

Card Library Requirements

Design Card Library as a clean Apple Notes-style list with Apple Wallet-like polish.

The Library should show:

- A simple empty state when there are no cards.
- Main actions when cards exist.
- Sections grouped by status.

Empty state:

```
No cards yet.
When your child finds a word they don't know, take a quick photo. BookWords will turn it into a flashcard.
Open Camera
```

If a status group is empty, hide that section entirely.

Do not show `Needs Review` if there are zero Needs Review cards.

Do not repeat status labels inside individual cards when the heading already communicates status.

Card rows should show:

- word
- short definition preview
- subtle chevron or affordance

Do not show Review/Edit buttons on every card.

Tapping a card opens Card Detail.

At the top of Library, when cards exist, show:

- Review Ready Cards
- Order 10-card pack - $9.99 only when 10+ Ready cards exist

When the parent taps Review Ready Cards, review all Ready cards.

For MVP, review order is newest first. Structure it so smarter review order can be added later.

Allow iOS-style swipe-to-delete in Library. Once parent swipes and taps Delete, delete immediately. Do not show another confirmation after the Delete tap.

Card Detail Requirements

Card Detail should be a polished flashcard detail page, not a generic form.

Use:

- Large vocabulary word at top in literary serif.
- Subtle status display.
- Apple Settings-style grouped sections.
- Clean read-only sections.
- Editing as a secondary action.

Sections:

- Official Definition
- Kid-Friendly Explanation
- Book Context
- Pronunciation
- Synonyms

Do not show every value as a raw editable input.

Official definition:

- read-only
- sourced
- not user-editable

The parent can directly edit only:

- captured word
- extracted sentence/context if present

After either changes, show:

```
Regenerate this card?
Updating the word or sentence will refresh the definition, kid-friendly explanation, pronunciation, synonyms, and context.
Cancel / Yes, regenerate
```

Use iOS-style action sheet or native-feeling bottom sheet for important actions. Do not use browser dropdowns.

Status change should use an iOS-style bottom action sheet, not a normal HTML dropdown.

Status action sheet should include:

- Needs Review
- Ready
- Cancel

Ordered should usually be applied automatically after checkout, not manually selected.

Mobile Flashcard Review

Review uses all Ready cards, newest first.

Flashcard review should be parent-led and low pressure.

Do not include:

- Know it
- Still learning
- Scores
- Streaks
- Badges
- Mastery tracking
- Correct/incorrect pressure

Use:

- Tap card to flip.
- Back button.
- Next button.
- Quiet secondary action to mark card Needs Review.

If parent chooses to flag a card during review, show an iOS-style bottom action sheet:

```
Mark this card Needs Review?
This will move the card back to Needs Review so it can be corrected later.
Mark Needs Review
Cancel
```

Physical Card Ordering

The physical order flow should be polished but mocked in this iteration.

BookWords is free for digital use. Parents only pay when ordering physical 4x6 cards.

Show:

```
Order 10-card pack - $9.99
```

Order option appears only when the parent has at least 10 Ready cards.

The app suggests a 10-card pack, but the parent can adjust selected cards.

Checkout flow:

1. Select Cards
2. Preview Physical Cards
3. Shipping Address
4. Mock Apple Pay-style payment
5. Order Confirmation

Physical card layout:

Front:

- vocabulary word only

Back:

- vocabulary word
- pronunciation
- short official definition
- kid-friendly explanation
- book-context meaning if available
- synonyms

Do not include book title/page.

Do not include page photo.

Preview screen on mobile:

- show a list of selected words
- show one sample 4x6 front/back preview
- do not try to show every physical card full-size on mobile

After mocked order confirmation, selected cards can move to Ordered status in the local/mock state.

Native iOS Interaction Patterns

Avoid default browser controls for important interactions.

Use native-feeling iOS-style patterns:

- fixed bottom tab bar
- prominent center camera action
- bottom action sheets
- dimmed background for action sheets
- rounded sheets
- large tap targets
- separated Cancel button
- Apple Pay-style button
- clean swipe-to-delete

Do not use ugly HTML selects/dropdowns for status changes.

Export Package

Create an exportable review package:

1. Individual PNG screenshots for each major screen/state.
2. One combined PDF containing all screens in user journey order.
3. Figma-ready screen/design spec.

If automated export is not practical, provide clear scripts or commands for capturing screenshots and compiling the PDF.

Required Click Paths

Support these click paths:

1. Open `/bookwords` -> Camera screen.
2. Grant camera permission -> live camera feed appears.
3. Deny camera permission -> fallback with manual entry.
4. Tap capture -> show `Captured` chip.
5. OCR succeeds -> chip updates to `Saved: [word]`.
6. OCR low confidence -> chip updates to `Needs Review`.
7. Manual word entry -> create card immediately.
8. Bottom tab Library -> Card Library.
9. Library empty -> Open Camera.
10. Library with cards -> grouped sections.
11. Tap card -> Card Detail.

12. Edit word or sentence -> Regenerate card prompt.
13. Status change -> iOS-style action sheet.
14. Review Ready Cards -> flashcard stack.
15. Tap flashcard -> flip.
16. Back/Next in review.
17. Mark Needs Review -> iOS action sheet.
18. Swipe-to-delete in Library.
19. 10+ Ready cards -> Order 10-card pack - $9.99.
20. Order flow -> Select, Preview, Shipping, Mock Apple Pay, Confirmation.
21. Account tab -> sign in, child profile, shipping/payment basics.
22. Sign in with Apple/Google -> sync local cards to Supabase if configured.

What to Ask Me Before Proceeding

Ask me a clarifying/setup question before doing anything destructive or blocked by an external dependency.

Ask me before:

- Applying Supabase schema migrations if the repo pattern is unclear.
- Changing existing MINERVA auth behavior.
- Adding new required environment variables that are not documented.
- Removing existing app files.
- Connecting real Google Cloud Vision credentials.
- Connecting real Supabase auth providers.
- Connecting real address autocomplete credentials.
- Making real payment integration.

Do not ask for secret values. Ask me to configure the relevant service or environment variable and tell me the expected variable names.

Final Developer Summary Required

When finished, provide:

1. Files changed.
2. How to run the app locally.
3. How to view `/bookwords`.
4. How to generate screenshots/PDF.

5. Which features are real.
6. Which features are mocked.
7. Required environment variables and external setup.
8. Supabase schema changes, if any.
9. Known limitations.
10. Expo/React Native migration note.