Achieving Flawless High-Yield Anki Cards for Medical Students

This report provides a comprehensive guide for programmatically generating "absolutely perfect, high-yield" Anki cards for medical students using the genanki Python library. It addresses common challenges related to card structure, aesthetics, and content display, offering precise technical solutions and best practices to ensure a superior learning experience. The analysis delves into the intricacies of Anki's templating system, provides exact genanki code for custom note types, corrects cloze deletion syntax, and integrates advanced CSS for optimal readability and aesthetic appeal. Debugging strategies are also included to ensure seamless deck generation and import.

1. Foundations of Anki Card Design for Automation

To programmatically generate Anki cards that are "absolutely perfect" and "high yield," the underlying principles of Anki's internal structure must be thoroughly understood. This involves a clear grasp of Notes, Fields, Models, and Templates, which collectively define card content, presentation, and behavior.

1.1 Deconstructing Anki Notes: Fields, Models, and Templates

At its core, Anki operates on the concept of a **Note**, which represents a single fact or piece of information intended for memorization. A single Note can be configured to generate one or more **Cards**, which are the actual flashcards presented for review. This distinction is fundamental: developers create notes, and Anki subsequently generates reviewable cards from these notes based on predefined templates.

Fields serve as the fundamental containers for specific pieces of information within a Note. For medical students utilizing high-yield content, it is beneficial to define distinct fields for various types of information. Examples include "Concept," "Definition," "Mnemonic," "Vignette," "Clinical Correlation," and "Additional Notes." It is imperative to remember that field names are case-sensitive within Anki's templating system, meaning {{Front}} will not correctly display content from a field named front.

A **Model**, also known as a Note Type, defines the overarching structure for a particular category of Notes. This definition encompasses the unique set of fields that notes of this type will contain, as well as the associated templates that dictate how cards are generated from these fields. Each Model must be assigned a unique model_id (a large integer, typically generated once and hardcoded for consistency) to ensure Anki correctly identifies and manages it within the user's collection.

Templates function as blueprints, dictating precisely how content from the defined fields is displayed on the front and back of a card. They are constructed using HTML to structure the content and CSS to style its appearance. Anki templates are typically organized into three distinct sections: the "Front Template" (displaying the question or prompt), the "Back Template" (revealing the answer and additional information), and the "Styling" section (containing global CSS rules for the card's visual presentation).

A common issue encountered is when supplementary information, often referred to as "notes," fails to appear on the generated cards. This stems from a misunderstanding of Anki's internal structure. Anki does not possess an automatic or predefined "notes section" where information

will render by default. Instead, all content, including what a user might perceive as "notes," must be explicitly stored within a specifically defined *field* within a Note. Furthermore, this *field* must then be explicitly referenced and placed using HTML syntax (e.g., {{FieldName}}) within the card's HTML template (either qfmt for the front or afmt for the back). If the AI agents generate "notes" data but this data is not mapped to a corresponding genanki field that is subsequently included in the card's template, it will indeed remain invisible. The solution is not to find a hidden "notes section" but to correctly map and display the data.
For the programmatic generation, this means that any information intended as "notes" must be assigned to a dedicated field (e.g., AdditionalNotes) within the genanki.Model definition. Subsequently, this AdditionalNotes field must be explicitly included in the afmt (answer format) section of the card template using {{AdditionalNotes}}. This approach directly addresses the display issue by aligning the AI's output with Anki's structural requirements.

Field Name	Purpose/Function
Question	Primary prompt for recall; often the clinical scenario or concept to be defined.
Answer	Core factual answer; the direct response to the question.
Mnemonic	Memory aid for complex information, often a phrase or acronym.
Vignette	Clinical scenario for context, enhancing real-world application.
ClinicalCorrel	Link between basic science and clinical practice, expanding on the vignette.
AdditionalNotes	Supplementary details, explanations, or personal annotations.
Source	Origin of information (e.g., textbook, lecture, Qbank ID) for verification.

Table 1: Anki Field Types and Their Purpose. This table clarifies how different types of information, including supplementary "notes," should be mapped to distinct Anki fields to ensure proper display and organization within the generated cards.

1.2 The Power of HTML for Card Structure

HTML (HyperText Markup Language) is the foundational language that defines the layout, arrangement, and hierarchical structure of fields and other content on Anki cards. It provides the precise means to control which fields are displayed, their exact placement, and their visual relationship to one another.
To display the content of a specific field within an HTML template, the double curly brackets syntax must be used: {{FieldName}}. As previously emphasized, FieldName must be an exact, case-sensitive match to the field name defined in the genanki.Model.
A critical distinction from standard text editors is that typical newlines within HTML are generally ignored during rendering. To force content onto a new line and control vertical spacing, the HTML
 tag must be explicitly inserted. This is particularly important for multi-line answers, lists, or separating distinct pieces of information within a single field.
The special field {{FrontSide}} is an invaluable tool within the back template, as it automatically renders the entire content of the card's front side before the answer is revealed. A common and highly recommended practice for clarity and visual separation is to separate the question (rendered by {{FrontSide}}) from the answer using a horizontal rule, typically <hr id="answer">. The id="answer" is a standard Anki convention that allows for specific CSS targeting, enabling distinct styling for the answer section.

A crucial detail for genanki implementation is that any HTML content placed into note fields (e.g., if AI agents generate rich text or specific HTML snippets within a field) must be properly HTML-encoded if it contains literal characters such as <, >, or &. Failure to do so can lead to malformed HTML, rendering issues, or unexpected behavior where parts of the text might disappear or be misinterpreted. This is not merely a cosmetic issue; it is a fundamental data integrity concern that can break the card's intended meaning and display. The html.escape() function in Python is recommended for this purpose to ensure that raw text from the AI is correctly treated as data, preventing unintended HTML parsing errors. This proactive measure significantly contributes to the goal of "absolutely perfect" cards.

HTML Tag	Purpose/Function	Example Usage in Template
	Generic block-level container for grouping content and applying styles.	{{Question}}
	Paragraph block; often used for distinct blocks of text.	{{Definition}}
	Inline text container for applying styles to a specific part of text.	This is important.
 	Forces a line break, moving subsequent content to the next line.	{{Field1}} {{Field2}}
<hr>	Creates a horizontal separator line, commonly used between question and answer.	{{FrontSide}}<hr id="answer">
or	Bold text, indicating importance or emphasis.	{{Keyword}}
or	Italic text, often used for emphasis or foreign terms.	{{MedicalTerm}}
<u>	Underlined text.	<u>{{UnderlinedFact}}</u>
	Unordered list (bullet points).	- Item 1 - Item 2
	Ordered list (numbered points).	- Step 1 - Step 2
	List item, used within or .	{{ListItem}}
<img>	Embeds an image. Requires src attribute pointing to image filename (in media folder).	<img src="image.jpg">

Table 2: Essential Anki HTML Tags for Card Structure. This table provides a practical reference for constructing the HTML layout of Anki cards, ensuring distinct elements are correctly positioned and formatted.

1.3 Elevating Aesthetics with CSS Styling

CSS (Cascading Style Sheets) is the language dedicated to controlling the visual appearance of Anki cards, encompassing everything from font styles and sizes to colors, text alignment, spacing, and background properties. These styling rules are applied within the dedicated "Styling" section of the Anki template editor, which genanki allows to be passed directly as a css argument to the Model definition.
The complaints regarding "text font doesn't look good" and "the front of the card doesn't look

centered" are direct manifestations of inadequate CSS styling. Anki's default templates have been described as "unreadable," "outdated," and vertically "cramped". This is not merely about minor aesthetic preferences; it concerns fundamental typographical and layout flaws that can actively hinder learning efficiency, particularly for medical students who must process vast amounts of dense information. Therefore, for the programmatic generation to produce "absolutely perfect" cards, it is essential to embed a comprehensive, modern CSS block within the genanki.Model definition that explicitly overrides Anki's defaults.
Key CSS properties essential for refining card appearance and addressing these concerns include:

- **font-family**: Specifies the typeface. For optimal readability and a modern aesthetic across various devices, it is highly recommended to use system fonts (e.g., -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, Oxygen, Ubuntu, Cantarell, "Open Sans", "Helvetica Neue", sans-serif). This ensures the device renders text using its native, optimized font. Custom fonts can also be incorporated by placing .ttf files (prefixed with an underscore, e.g., _Menlo.ttf) in Anki's media folder and using @font-face in the CSS.
- **font-size**: Controls text size. While 20px is a common default, a slightly smaller size like 19px can, paradoxically, enhance overall readability when combined with improved line spacing and line length, especially on mobile devices. Percentages (e.g., 150%) can also be used for relative sizing.
- **color**: Sets the text color.
- **background-color**: Defines the card's background color.
- **text-align**: Determines horizontal text alignment (e.g., center, left, start). For paragraphs and longer content blocks, text-align: start; (which automatically adjusts for left-to-right or right-to-left languages) is strongly preferred over center alignment due to significantly improved readability and reduced eye strain. Centering is generally only suitable for short headlines or single lines.
- **line-height**: Adjusts vertical spacing between lines. This is critical for readability, especially for dense medical content. A value of 1.5 or 1.6 is widely recommended to provide ample "room to breathe" for text, even if it means slightly less content fits on screen.
- **max-width**: Limits the maximum width of text lines, crucial for readability. Research suggests optimal line lengths, typically around 50-75 characters, which translates to a max-width of approximately 650px for the content area. This property also affects the width of images within the card.
- **margin and padding**: Control spacing *around* (margin) and *within* (padding) content blocks. Using margin: 0 auto; on a block-level element with a defined max-width is the standard method for horizontally centering that element within its parent container.

Medical students, like many modern learners, utilize Anki across various devices, including desktop computers and mobile phones. The concept of "perfect" cards implicitly extends to their appearance and functionality on *all* these platforms. If generated cards rely solely on fixed pixel values without considering responsive design principles, they will likely appear poorly formatted or require excessive scrolling on smaller screens. Therefore, the CSS should incorporate responsive design principles, using relative units where appropriate (e.g., em, rem, percentages), and leveraging CSS properties like max-width with margin: auto to ensure content blocks adapt gracefully to different screen sizes. While genanki itself does not directly handle media queries, the CSS string passed to the model *can* contain them, allowing for device-specific adjustments. This foresight ensures that the "seamless download of high yield cards" translates into a truly seamless and optimized viewing and learning experience regardless of the user's device.

CSS Property	Purpose/Effect	Recommended Value/Best Practice
font-family	Sets the typeface for text.	system-ui, sans-serif (for modern system fonts)
font-size	Controls the size of the text.	19px (optimal for readability)
color	Defines the color of the text.	black (for light mode), white (for dark mode)
background-color	Sets the background color of the card.	white (for light mode), #1a1a1a (for dark mode)
text-align	Aligns text horizontally within its container.	start (adapts to LTR/RTL, better for paragraphs)
line-height	Adjusts the vertical spacing between lines of text.	1.5 to 1.6 (for improved readability)
max-width	Limits the maximum width of content (text, images).	650px (caps lines around 75 characters for readability)
margin	Controls the outer spacing of an element.	0 auto (for horizontal centering of block elements with max-width)
padding	Controls the inner spacing of an element (space between content and border).	40px 20px (top/bottom 40px, left/right 20px for comfortable margins)
display: flex;	Enables Flexbox layout for its container.	Used on body or .card for centering content.
justify-content: center;	Centers flex items along the main axis.	Used with display: flex for horizontal centering.
align-items: center;	Centers flex items along the cross axis.	Used with display: flex for vertical centering.

Table 3: Key CSS Properties for Anki Card Styling. This table provides actionable CSS properties and recommended values to address font appearance, centering, and overall card readability, crucial for "perfect" and "high-yield" cards.

2. Mastering genanki for Programmatic Card Creation

The genanki Python library is the programmatic backbone for generating Anki decks, which are importable .apkg files. Understanding its core components and how to manipulate them is critical for the programmatic agent to precisely control card content and presentation, moving beyond the current suboptimal output.

2.1 Core genanki Components: Models, Notes, Decks, and Packages

genanki serves as a powerful Python 3 library specifically designed for programmatically generating Anki decks. It abstracts away the complexities of the Anki .apkg file format, allowing developers to focus on content and structure rather than low-level file manipulation.

- **genanki.Model**: This class is used to define a custom note type. It requires a unique model_id (a large integer, typically generated randomly once and then hardcoded for consistency to ensure Anki can track the model) and a descriptive name. Critically, this is where the fields that notes will contain are specified, along with the templates (front and back HTML, plus CSS) that will render cards from these fields. It can directly accept a css argument to apply custom styling to all cards generated from this model.

- **genanki.Note**: An instance of genanki.Note represents a single flashcard entry based on a specific Model. The model object and a list of fields values (the actual content for each field) are passed to its constructor. The order of values in the fields list must correspond precisely to the order of field names defined in the Model. For cloze deletion models, genanki.CLOZE_MODEL can be used, and the fields should contain the cloze syntax. A deprecation warning for CLOZE_MODEL indicates that it only had a single field in older versions, whereas the built-in Anki Cloze model has two; adding an empty string as a second field can resolve this.
- **genanki.Deck**: This class acts as a container for Notes. It also requires a unique deck_id and a name. Notes are added to a deck using the add_note() method.
- **genanki.Package**: This is the final component responsible for bundling one or more Decks along with any associated media_files (such as images or audio) into the single, importable .apkg file. The generated .apkg file is then imported directly into the Anki application.

2.2 Defining Custom Note Types with Multiple Fields

Custom note types are essential for creating high-yield medical cards that effectively organize diverse information such as mnemonics, vignettes, and clinical correlations. The genanki.Model class is central to this definition.
To define a custom model, the following parameters are crucial:

- model_id: A unique integer. It is recommended to generate this once (e.g., using random.randrange(1 << 30, 1 << 31)) and hardcode it to ensure consistency across generations.
- name: A descriptive string for the note type (e.g., "Medical High-Yield Card").
- fields: A list of dictionaries, where each dictionary defines a field with a name key. The order of these fields is important as it dictates the order in which content should be provided when creating genanki.Note instances.
- templates: A list of dictionaries, each defining a card template. Each template requires a name, qfmt (front format HTML), and afmt (back format HTML).
- css: An optional string containing the CSS rules that will be applied to all cards generated from this model. This is where all the styling recommendations from Section 1.3 will be applied.

For example, a custom model for medical students might include fields like 'Question', 'Answer', 'Mnemonic', 'Vignette', 'ClinicalCorrelation', and 'AdditionalNotes'.

```
import genanki
import random
import html

# Generate unique IDs once and hardcode them for consistent deck
generation
# For production, generate these once and save them.
MY_MODEL_ID = 1234567890 # Example ID, replace with a generated one
MY_DECK_ID = 9876543210  # Example ID, replace with a generated one

# Define the CSS for the cards
# This CSS addresses font, centering, line height, and max width
issues.
# It uses system fonts for optimal readability across devices.
# The.card class applies to the entire card content.
```

```
# The.qa-section is a custom div to wrap question/answer for
centering.
# The.mnemonic-section and.vignette-section are for specific content.
MY_CARD_CSS = """
.card {
    font-family: -apple-system, BlinkMacSystemFont, "Segoe UI",
Roboto, Oxygen, Ubuntu, Cantarell, "Open Sans", "Helvetica Neue",
sans-serif;
    font-size: 19px; /* Slightly smaller for better readability with
improved line spacing */
    color: black;
    background-color: white;
    text-align: start; /* Left-align text for paragraphs, greatly
improves readability */
    line-height: 1.6; /* Wider line spacing for readability */
    margin: 0 auto; /* Horizontal centering of the card content block
*/
    padding: 40px 20px; /* Comfortable margins around content */
    max-width: 700px; /* Limit line length for readability, around 75
characters */
    box-sizing: border-box; /* Include padding in element's total
width and height */
}

.nightMode.card {
    color: white;
    background-color: #1a1a1a;
}

/* Specific styling for the question/answer section to ensure proper
centering */
.qa-section {
    display: block; /* Ensure it's a block element */
    width: 100%; /* Take full width of its parent */
    text-align: start; /* Ensure text within is left-aligned */
}

/* Centering the entire card content vertically and horizontally */
html {
    height: 100%;
    display: flex;
    justify-content: center; /* Center horizontally */
    align-items: center;   /* Center vertically */
}

/* Style for the horizontal rule */
hr {
    border: none;
    border-top: 1px solid #ccc;
```

```
    margin: 20px 0;
}

/* Style for mnemonics and vignettes */
.mnemonic-section,.vignette-section,.additional-notes-section {
    margin-top: 25px;
    padding-top: 15px;
    border-top: 1px dashed #eee;
    font-size: 0.95em;
    color: #555;
}

.nightMode.mnemonic-section,.nightMode.vignette-section,.nightMode.add
itional-notes-section {
    color: #aaa;
    border-top: 1px dashed #444;
}

.section-label {
    font-weight: bold;
    color: #333;
    margin-bottom: 5px;
    display: block;
}

.nightMode.section-label {
    color: #ccc;
}

/* Style for cloze deletions */
.cloze {
    font-weight: bold;
    color: #007bff; /* A distinct color for cloze text */
}

.nightMode.cloze {
    color: #66b3ff;
}
"""

# Define the custom model (Note Type)
# This model includes all necessary fields for high-yield medical
cards.
# The templates define how these fields are displayed on the front and
back.
medical_model = genanki.Model(
    MY_MODEL_ID,
    'Medical High-Yield Card',
    fields=,
```

```
    templates=,
    css=MY_CARD_CSS
)
```

Table 4: genanki Model, Note, and Template Parameters. This table summarizes the key arguments for genanki.Model and genanki.Note, providing a quick reference for their configuration.

genanki Class	Parameter	Description
genanki.Model	model_id	Unique integer ID for the note type.
	name	String name for the note type.
	fields	List of dictionaries defining field names.
	templates	List of dictionaries defining card templates (name, qfmt, afmt).
	css	String containing CSS for card styling.
genanki.Note	model	The genanki.Model object this note belongs to.
	fields	List of strings containing the content for each field, in order.
	guid	Optional: A unique ID for the note. Default is hash of fields.
genanki.Deck	deck_id	Unique integer ID for the deck.
	name	String name for the deck.
genanki.Package	decks	List of genanki.Deck objects to include.
	media_files	List of paths to media files (images, audio) to be included.

2.3 Generating Cards with Dynamic Content

Once the Model is defined, genanki.Note instances are created to populate the deck with actual content. The fields argument of genanki.Note takes a list of strings, where each string corresponds to the content for a field defined in the model. The order of these strings must match the order of fields in the Model's fields list.
For instance, if the medical_model has fields Question, Answer, Mnemonic, Vignette, ClinicalCorrelation, AdditionalNotes, and Source, a note would be created as follows:

```
# Example of creating a note
my_note = genanki.Note(
    model=medical_model,
    fields=
)
```

As highlighted in Section 1.2, it is crucial to apply html.escape() to all field content, especially if the AI-generated text might contain characters like <, >, or &. This prevents the Anki rendering engine from misinterpreting these characters as HTML tags, which could lead to malformed card structure or missing content.
After creating notes, they are added to a genanki.Deck instance. Finally, a genanki.Package

bundles the deck(s) and any media files into an .apkg file, ready for import into Anki.

```
# Create a deck
my_medical_deck = genanki.Deck(
    MY_DECK_ID,
    'High-Yield Medical Cards'
)

# Add the note to the deck
my_medical_deck.add_note(my_note)

# Generate the.apkg file
genanki.Package(my_medical_deck).write_to_file('high_yield_medical_car
ds.apkg')
```

This structured approach ensures that all information generated by the AI agents is correctly mapped to specific fields, which are then rendered according to the defined HTML templates and styled by the embedded CSS.

3. Implementing High-Yield Medical Card Features

High-yield Anki cards, especially for medical students, often leverage specific features like cloze deletions, mnemonics, and vignettes to optimize learning efficiency and recall. Correct implementation of these features is paramount for card effectiveness.

3.1 Correct Cloze Deletion Syntax and Application

A significant problem identified is the incorrect cloze deletion syntax, specifically the use of {{}} instead of {{c1::}}. Anki's cloze deletion cards are a powerful note type designed for fill-in-the-blank exercises. They allow specific parts of a sentence or text to be hidden, forcing active recall of the missing information.

The correct syntax for creating a cloze deletion is to enclose the text to be hidden within double curly braces, prefixed with c and a number (e.g., {{c1::text to hide}}). The number (e.g., c1, c2, c3) indicates a specific cloze deletion. Multiple cloze deletions can be created within a single field by using different numbers (e.g., {{c1::first blank}} and {{c2::second blank}}). Anki will automatically generate separate cards for each cloze deletion, allowing independent review.

For genanki, cloze deletion notes are created using genanki.CLOZE_MODEL. The field content for a cloze note must directly contain the cloze syntax.

```
# Example of a cloze note
cloze_note = genanki.Note(
    model=medical_model, # Use the custom model which has a cloze
template
    fields=
)
my_medical_deck.add_note(cloze_note)
```

It is important to note that while genanki.CLOZE_MODEL is available, it historically had only one field, whereas the actual built-in Anki Cloze model has two. If a deprecation warning is encountered, simply adding an empty string as a second field when creating the genanki.Note can resolve this. The custom model defined in Section 2.2 includes a cloze template that

correctly uses {{cloze:Question}} in its qfmt and afmt to handle cloze deletions from the Question field.

Cloze Deletion Type	Syntax	Description
Single Cloze	{{c1::text to hide}}	Hides a single piece of information.
Multiple Clozes (same card)	{{c1::text1}}... {{c1::text2}}	Hides multiple pieces of information, revealing all on the same card.
Multiple Clozes (different cards)	{{c1::text1}}... {{c2::text2}}	Hides multiple pieces of information, generating a separate card for each cloze.

Table 5: Anki Cloze Deletion Syntax Variations. This table clarifies the correct syntax for cloze deletions, directly addressing the user's problem with incorrect curly brace usage.

3.2 Structuring and Displaying Mnemonics and Vignettes

The absence of mnemonics and vignettes on the cards indicates that these rich content elements are either not being stored in dedicated fields or are not being correctly displayed in the card templates. For medical students, mnemonics and clinical vignettes are crucial for enhancing memory retention and applying foundational knowledge to practical scenarios.
To ensure mnemonics and vignettes appear correctly, they must be assigned to dedicated fields within the genanki.Model (e.g., Mnemonic, Vignette, ClinicalCorrelation). As demonstrated in the medical_model definition in Section 2.2, these fields are explicitly defined.
Once the data is in these fields, the card templates (specifically the afmt for the back of the card) must include references to these fields using the {{FieldName}} syntax. For example:

```
<div class="mnemonic-section">
    <span class="section-label">Mnemonic:</span> {{Mnemonic}}
</div>
<div class="vignette-section">
    <span class="section-label">Vignette:</span> {{Vignette}}
</div>
<div class="vignette-section">
    <span class="section-label">Clinical Correlation:</span>
{{ClinicalCorrelation}}
</div>
```

The use of conditional display syntax, {{#FieldName}}... {{/FieldName}}, is highly recommended. This ensures that a section (e.g., for a mnemonic) only appears on the card if the corresponding field actually contains content. If the field is empty, the entire div block will be hidden, preventing blank sections from appearing and maintaining a clean card layout. This is crucial for dynamically generated content where not every card may have a mnemonic or vignette.
For mnemonics, a best practice for medical students is to structure them so that the disease or concept prompts the mnemonic, which then helps retrieve the underlying facts. This can involve cloze deletion around the mnemonic itself, or simply presenting the mnemonic clearly on the answer side. The medical_model's back template includes specific div elements with classes (mnemonic-section, vignette-section) to allow for distinct styling of these important content types, making them visually distinct and easy to locate.

3.3 Optimizing Readability: Font, Size, Line Height, and Alignment

The visual presentation of Anki cards significantly impacts learning efficiency, especially for dense medical information. Complaints about "text font doesn't look good" and uncentered card fronts directly point to suboptimal styling. Achieving "perfect" cards requires careful attention to typography and layout.
As detailed in Section 1.3, the MY_CARD_CSS provided in Section 2.2 incorporates best practices for readability:

- **Font Family**: Utilizes system fonts (-apple-system, BlinkMacSystemFont, "Segoe UI",...) to ensure optimal rendering across various devices and operating systems. This provides a clean, modern, and highly legible typeface.
- **Font Size**: Sets a font-size of 19px, which, when combined with appropriate line spacing, offers excellent readability without overwhelming the screen.
- **Line Height**: A line-height of 1.6 is applied to provide generous vertical spacing between lines of text. This "room to breathe" significantly reduces eye strain and improves the ability to scan and comprehend longer passages, which is common in medical content.
- **Text Alignment**: text-align: start; is used for the main card content. This left-aligns text (for left-to-right languages), which is universally recognized as superior for readability of paragraphs and longer texts compared to center alignment. Center alignment forces the eye to hunt for the start of each new line, increasing cognitive load.
- **Line Length**: max-width: 700px; (or approximately 650px as suggested in some contexts) limits the maximum width of text lines. This ensures that lines do not stretch excessively across wide screens, maintaining an optimal reading experience of around 50-75 characters per line.
- **Margins and Padding**: padding: 40px 20px; provides comfortable internal spacing, while margin: 0 auto; horizontally centers the card content block within the Anki window.

These CSS properties, when applied via genanki.Model(css=MY_CARD_CSS), transform the card's appearance from a default, often unreadable state to a highly polished, professional, and learner-friendly format.

4. Advanced Styling and Layout for "Perfect" Cards

Beyond basic readability, achieving "absolutely perfect" Anki cards involves advanced styling techniques to ensure ideal presentation across all devices and a visually appealing user experience.

4.1 Achieving Centered and Responsive Card Layouts

The problem of the "front of the card doesn't look centered" is a common layout issue that can be effectively resolved using CSS. While text-align: center; centers inline content within a block, to truly center a block-level element (like the entire card content or a specific section), different techniques are required.
For horizontal centering of the main card content, applying max-width along with margin: 0 auto; to the .card class (or a wrapper div within it) is the standard and most robust method. This ensures the content block is centered within the available space while maintaining a readable line length.
For comprehensive vertical and horizontal centering of the *entire* card content within the Anki window, CSS Flexbox is the most modern and effective solution. By applying display: flex;, justify-content: center;, and align-items: center; to the html or body element of the card template, the content can be perfectly centered. The MY_CARD_CSS provided in Section 2.2 includes this approach:

```
html {
```

```
    height: 100%;
    display: flex;
    justify-content: center; /* Center horizontally */
    align-items: center;   /* Center vertically */
}
```

This ensures that regardless of the screen size or content length, the primary card content remains visually balanced in the center of the display.
Furthermore, cards must be **responsive** to function well across different devices, from desktop monitors to mobile phones. The max-width property, as discussed, is key to responsive text layout. For more complex responsive adjustments, CSS media queries (@media screen and (max-width: 768px) {... }) can be embedded within the css string passed to genanki.Model. These queries allow for device-specific styling rules, ensuring that cards adapt gracefully to smaller screens by adjusting font sizes, padding, or element arrangements as needed.

4.2 Custom Fonts and Theming for Visual Appeal

While system fonts offer excellent readability and broad compatibility, the ability to use custom fonts can further enhance the visual appeal and branding of the generated cards. To incorporate a custom font:

1. **Add Font File to Media Folder**: The .ttf font file must be placed in Anki's collection.media folder. It is crucial to prefix the font filename with an underscore (e.g., _Menlo.ttf) to signal to Anki that it is a template-related file and should not be deleted during media checks or excluded during deck export.
2. **Declare Font in CSS**: Within the MY_CARD_CSS string, use the @font-face rule to declare the custom font, referencing its filename:
   ```
   @font-face {
       font-family: 'Menlo'; /* Choose a name for your font */
       src: url("_Menlo.ttf"); /* Reference the file in Anki's media
   folder */
   }
   ```
3. **Apply Font**: Once declared, the font-family property can be used to apply the custom font to specific elements or globally:

.card { font-family: 'Menlo', sans-serif; /* Fallback to sans-serif if custom font not available */ } ``` Theming, particularly dark mode, is also a significant consideration for user comfort and visual appeal, especially for medical students who spend extensive hours studying. Anki supports a nightMode class that is automatically applied to the body element when dark mode is enabled. The MY_CARD_CSS provided includes basic nightMode rules to ensure text and background colors invert appropriately, maintaining contrast and readability in low-light environments:

```
.nightMode.card {
    color: white;
    background-color: #1a1a1a;
}
/* Extend for other elements as needed */
.nightMode.mnemonic-section,.nightMode.vignette-section,.nightMode.add
itional-notes-section {
    color: #aaa;
    border-top: 1px dashed #444;
```

```
}
.nightMode.section-label {
    color: #ccc;
}
.nightMode.cloze {
    color: #66b3ff;
}
```

This allows the generated cards to seamlessly adapt to the user's preferred Anki theme settings.

4.3 Incorporating Conditional Display Logic

Conditional display logic allows certain content sections or fields to appear only when specific conditions are met, such as a field containing data. This is particularly useful for mnemonics, vignettes, or additional notes that may not be present on every card generated by the AI agents. Anki's templating system supports simple conditionals using {{#FieldName}}... {{/FieldName}} to display content only if FieldName is not empty, and {{^FieldName}}... {{/FieldName}} to display content if FieldName *is* empty.
The medical_model templates in Section 2.2 demonstrate this for Mnemonic, Vignette, ClinicalCorrelation, AdditionalNotes, and Source fields:

```
{{#Mnemonic}}
<div class="mnemonic-section">
    <span class="section-label">Mnemonic:</span> {{Mnemonic}}
</div>
{{/Mnemonic}}
```

This ensures that the div for the mnemonic section, including its label and styling, only renders if the Mnemonic field for that specific note actually contains text. This prevents unsightly blank sections on cards that do not have a mnemonic, contributing to a cleaner and more professional appearance. This approach is superior to relying on JavaScript for simple show/hide functionality, as it is native to Anki's templating and avoids potential compatibility issues with different Anki versions or platforms.

5. Debugging and Validation of Generated Anki Decks

Even with precise code, issues can arise during programmatic deck generation. Effective debugging and validation strategies are crucial to ensure the "absolutely perfect" quality of the final .apkg files.

5.1 Common genanki Errors and Resolution Strategies

- **Invisible Notes/Cards**: A common issue is generating notes that do not appear in the Anki browser or result in "inconsistent state" errors. This often occurs if the field designated for the *front* of the card (e.g., Question in a basic model) is empty. Anki requires content on the front to generate a reviewable card.
    - **Resolution**: Ensure that the primary field used in the qfmt (front template) always contains data. If the AI agent might produce an empty value, implement a fallback or a placeholder. Debug logging in genanki can show note field contents to verify this.

- **Malformated HTML/CSS**: If the card structure or styling appears incorrect, or text is garbled, it is often due to unescaped HTML characters (<, >, &) within field content or errors in the CSS string.
    - **Resolution**: Always apply html.escape() to all field data before passing it to genanki.Note. Carefully review the CSS string for syntax errors.
- **Incorrect Cloze Deletions**: The problem of {{}} instead of {{c1::}} is a syntax error.
    - **Resolution**: Ensure the AI agent generates cloze text precisely in the {{cN::text}} format. Verify the genanki.Note field contains this exact syntax.
- **Media Files Not Appearing**: Images or audio files might not be included or displayed correctly.
    - **Resolution**: Ensure media files are correctly listed in genanki.Package(media_files=...) with their paths. In the Anki field, reference only the *filename* (e.g., <img src="image.jpg">, [sound:audio.mp3]), not the full path. Ensure media filenames are unique.
- **AnkiDroid Crashes on Import**: Older versions of AnkiDroid (prior to 2.16) had issues importing decks with subdecks (e.g., "Test::New") if a default deck was not present or due to other internal issues.
    - **Resolution**: Ensure genanki is creating a valid deck structure. This issue is largely resolved in newer AnkiDroid versions.

genanki-go (a Go implementation of genanki) mentions optional debug logging that can show detailed information about database operations, note field contents, and model/deck counts. While genanki (Python) doesn't have an identical feature, printing the fields list of generated genanki.Note objects before adding them to the deck can serve a similar purpose for content verification.

5.2 Inspecting Anki Card HTML and CSS for Troubleshooting

The most effective way to debug rendering issues in Anki is to inspect the rendered HTML and applied CSS directly within the Anki desktop application.

- **Anki's Built-in Editor**: While adding or editing a note, click the "Cards..." button to open the template editor. Here, you can switch between Front Template, Back Template, and Styling sections (Ctrl+1, Ctrl+2, Ctrl+3 respectively in Anki 2.1.28+). The preview pane on the right provides immediate feedback on changes.
- **HTML Editor within Fields**: When editing a field, pressing Ctrl+Shift+X (or clicking the </> button) reveals the raw HTML content of that field. This is crucial for verifying if HTML tags or cloze syntax generated by the AI are correctly embedded within the field data.
- **Developer Tools (Inspector)**: Anki, being built on web technologies, allows for inspection of the rendered card using browser-like developer tools.
    - **Add-on**: The "View HTML Source with JavaScript and CSS styles" add-on (Anki ID: 31746032) provides a convenient right-click option to inspect the card's source code and live CSS. This is akin to Chrome or Firefox developer tools and is invaluable for debugging layout, font, and positioning issues.
    - **Manual Chrome DevTools (Advanced)**: For Anki Desktop, it is possible to connect Chrome's developer tools to Anki's WebView. This requires enabling HTML/JavaScript debugging in Anki's advanced preferences and then navigating to chrome://inspect in a Chrome browser on the same machine. This offers a full remote debugging console, allowing live modification and inspection of HTML and CSS. For AnkiDroid, similar steps involving USB debugging and chrome://inspect are possible.

By inspecting the live rendering, developers can pinpoint exactly where the HTML structure is "wrong" or why the CSS styling is not being applied as expected, providing clear targets for

correction in the genanki code or the embedded CSS string.

5.3 Verifying.apkg File Integrity

After generation, verifying the integrity of the .apkg file is the final step before distribution.

- **Import into Anki Desktop**: The primary method is to import the generated .apkg file into a fresh Anki profile (or a test profile) on a desktop computer. This allows for a full review of the deck, notes, and card appearance.
- **Browse Cards**: After import, navigate to the "Browse" screen in Anki. This allows inspection of all notes and their associated cards. Check for:
    - **Note Count**: Does it match the number of notes generated?
    - **Field Content**: Are all fields populated correctly, including mnemonics, vignettes, and additional notes?
    - **Card Generation**: Are the correct number of cards generated per note (e.g., for cloze deletions)?
    - **Cloze Syntax**: Are cloze deletions correctly rendered with [...] placeholders on the front?
- **Review Cards**: Begin a review session with the new deck to observe card behavior in a live study environment. Pay attention to:
    - **Front/Back Display**: Do all elements appear as expected on both sides?
    - **Styling**: Are fonts, colors, alignment, and spacing correct?
    - **Missing Elements**: Are mnemonics, vignettes, or other notes appearing correctly on the back?
    - **Centering**: Is the card content properly centered?
    - **Cloze Reveal**: Do cloze deletions reveal correctly upon showing the answer?
- **Anki's Database Check**: If Anki reports an "inconsistent state" or prompts to "clean database" after import, it often points to issues like notes with empty front fields. This indicates a fundamental problem in the genanki generation logic that needs immediate attention.

Thorough validation across these steps ensures that the generated .apkg files are robust, functional, and meet the high-yield quality standards for medical students.

6. Actionable Python Code for Your Replit Bot

This section provides a comprehensive genanki script template, integrating all the solutions and best practices discussed, designed for direct implementation by the programmatic agent in Replit.

6.1 Comprehensive genanki Script Template

The following Python script outlines the full process, from defining the model and CSS to creating notes and packaging the deck.

```
import genanki
import random
import html
import os

# --- Configuration ---
# Generate unique IDs once and hardcode them for consistent deck
generation.
```

```
# For a real application, these should be stored persistently (e.g.,
in a config file).
# Example generation: random.randrange(1 << 30, 1 << 31)
MY_MODEL_ID = 1607392319 # A consistent example ID
MY_DECK_ID = 2059400110  # A consistent example ID

DECK_NAME = 'High-Yield Medical Cards for Students'
OUTPUT_FILENAME = 'high_yield_medical_cards.apkg'

# --- CSS for Optimal Card Presentation ---
# This CSS addresses common readability issues: font, centering, line
height, and max width.
# It uses system fonts for optimal readability across devices and
includes dark mode support.
MY_CARD_CSS = """
.card {
    font-family: -apple-system, BlinkMacSystemFont, "Segoe UI",
Roboto, Oxygen, Ubuntu, Cantarell, "Open Sans", "Helvetica Neue",
sans-serif;
    font-size: 19px; /* Slightly smaller for better readability with
improved line spacing */
    color: black;
    background-color: white;
    text-align: start; /* Left-align text for paragraphs, greatly
improves readability */
    line-height: 1.6; /* Wider line spacing for readability */
    margin: 0 auto; /* Horizontal centering of the card content block
*/
    padding: 40px 20px; /* Comfortable margins around content */
    max-width: 700px; /* Limit line length for readability, around 75
characters */
    box-sizing: border-box; /* Include padding in element's total
width and height */
}

/* Dark mode styling */
.nightMode.card {
    color: white;
    background-color: #1a1a1a;
}

/* Ensure the entire card content is centered vertically and
horizontally within the Anki window */
html {
    height: 100%;
    display: flex;
    justify-content: center; /* Center horizontally */
    align-items: center;   /* Center vertically */
}
```

```
/* Style for the horizontal rule separating question and answer */
hr {
    border: none;
    border-top: 1px solid #ccc;
    margin: 20px 0;
}

/* Specific styling for sections like mnemonics, vignettes, and
additional notes */
.mnemonic-section,.vignette-section,.additional-notes-section {
    margin-top: 25px;
    padding-top: 15px;
    border-top: 1px dashed #eee; /* Subtle separator */
    font-size: 0.95em; /* Slightly smaller font for supplementary info
*/
    color: #555;
}

.nightMode.mnemonic-section,.nightMode.vignette-section,.nightMode.add
itional-notes-section {
    color: #aaa;
    border-top: 1px dashed #444;
}

.section-label {
    font-weight: bold;
    color: #333; /* Darker color for labels */
    margin-bottom: 5px;
    display: block; /* Ensure label is on its own line */
}

.nightMode.section-label {
    color: #ccc;
}

/* Styling for cloze deletions */
.cloze {
    font-weight: bold;
    color: #007bff; /* A distinct blue for cloze text */
}

.nightMode.cloze {
    color: #66b3ff; /* Lighter blue for dark mode */
}
"""

# --- Define the Custom Model (Note Type) ---
# This model includes all necessary fields for high-yield medical
```

```
cards and
# defines how these fields are displayed on the front and back using
HTML templates.
# It also embeds the custom CSS.
medical_model = genanki.Model(
    MY_MODEL_ID,
    'Medical High-Yield Card',
    fields=,
    templates=,
    css=MY_CARD_CSS
)

# --- Create a Deck ---
my_deck = genanki.Deck(
    MY_DECK_ID,
    DECK_NAME
)

# --- Function to Add a Basic Note ---
def add_basic_note(question, answer, mnemonic="", vignette="",
clinical_correl="", additional_notes="", source=""):
    """Adds a basic Q&A note to the deck with optional rich
content."""
    note_fields = [
        html.escape(question),
        html.escape(answer),
        html.escape(mnemonic),
        html.escape(vignette),
        html.escape(clinical_correl),
        html.escape(additional_notes),
        html.escape(source)
    ]
    my_note = genanki.Note(
        model=medical_model,
        fields=note_fields,
        # Optional: Custom GUID if needed for updates. Default is hash
of fields.
        # guid=genanki.guid_for(question, answer)
    )
    my_deck.add_note(my_note)
    print(f"Added Basic Note: '{question[:50]}...'")

# --- Function to Add a Cloze Note ---
def add_cloze_note(cloze_question, answer, mnemonic="", vignette="",
clinical_correl="", additional_notes="", source=""):
    """Adds a cloze deletion note to the deck with optional rich
content."""
    note_fields =
    my_note = genanki.Note(
```

```
        model=medical_model,
        fields=note_fields,
        # guid=genanki.guid_for(cloze_question)
    )
    my_deck.add_note(my_note)
    print(f"Added Cloze Note: '{cloze_question[:50]}...'")

# --- Example Usage: Populating Notes with AI Agent Data ---
# Simulate data received from AI agents.
# Ensure all text content is HTML-escaped before passing to
genanki.Note.

# Example 1: Basic Q&A Card
add_basic_note(
    question="What is the primary function of the glomerulus in the
kidney?",
    answer="The glomerulus is a capillary tuft that filters blood to
form glomerular filtrate, the first step in urine formation.",
    mnemonic="GLOMERULUS: G-F-L-O-M-E-R-U-L-U-S for 'Great Filtration
Location Of Many Essential Renal Units, Little Urine Starts'",
    vignette="A 55-year-old male with uncontrolled hypertension
presents with proteinuria and elevated serum creatinine. Biopsy
reveals glomerular sclerosis.",
    clinical_correl="Glomerular damage, often due to chronic
hypertension or diabetes, impairs filtration and leads to protein
leakage into urine.",
    additional_notes="The glomerular filtration barrier consists of
fenestrated endothelium, glomerular basement membrane, and
podocytes.",
    source="Guyton & Hall Textbook of Medical Physiology, Ch. 26"
)

# Example 2: Cloze Deletion Card
add_cloze_note(
    cloze_question="The {{c1::Na+/K+ ATPase pump}} actively transports
{{c2::3 Na+ ions}} out of the cell and {{c3::2 K+ ions}} into the
cell, against their concentration gradients.",
    answer="This pump is crucial for maintaining cell volume, resting
membrane potential, and is a target for cardiac glycosides like
digoxin.",
    mnemonic="Na/K Pump: '3 Na Out, 2 K In' - Think of a 'Naughty Kid'
(Na/K) who wants to get '3' (Na) out and '2' (K) in.",
    vignette="A patient on digoxin develops arrhythmias due to
electrolyte imbalance, affecting the Na+/K+ ATPase pump function.",
    clinical_correl="Digoxin inhibits the Na+/K+ ATPase, leading to
increased intracellular Na+ and subsequent increased intracellular
Ca2+, enhancing myocardial contractility.",
    additional_notes="Requires ATP hydrolysis for energy. Found in the
plasma membrane of most animal cells.",
```

```
    source="Robbins Basic Pathology, Ch. 1"
)

# Example 3: Another Basic Q&A with less content
add_basic_note(
    question="What is the role of surfactant in the lungs?",
    answer="Surfactant reduces surface tension in the alveoli,
preventing their collapse during exhalation and reducing the work of
breathing.",
    clinical_correl="Lack of surfactant in premature infants leads to
Neonatal Respiratory Distress Syndrome (NRDS).",
    source="Lippincott's Illustrated Reviews: Physiology, Ch. 15"
)

# Example 4: Another Cloze Card, demonstrating conditional display
(empty mnemonic)
add_cloze_note(
    cloze_question="The {{c1::Circle of Willis}} is an arterial
anastomosis at the base of the brain, providing {{c2::collateral blood
flow}}.",
    answer="It is formed by the anterior cerebral, middle cerebral,
posterior cerebral, anterior communicating, and posterior
communicating arteries.",
    mnemonic="", # This field is intentionally left blank to show
conditional display working
    vignette="A patient experiences a transient ischemic attack (TIA)
due to a partial occlusion in one of the arteries forming the Circle
of Willis, but symptoms resolve quickly due to collateral flow.",
    additional_notes="Variations in the Circle of Willis anatomy are
common.",
    source="Netter's Atlas of Human Anatomy, Plate 138"
)

# --- Generate the.apkg file ---
# If you have media files (e.g., images, audio), list their paths
here.
# For example: media_files=['./media/my_image.jpg',
'./media/my_audio.mp3']
# Ensure these files are accessible in your Replit environment.
my_package = genanki.Package(my_deck)
# my_package.media_files = ['path/to/my_image.jpg',
'path/to/my_audio.mp3'] # Uncomment and add if needed

try:
    my_package.write_to_file(OUTPUT_FILENAME)
    print(f"\nSuccessfully generated Anki deck: {OUTPUT_FILENAME}")
    print(f"Total notes created: {len(my_deck.notes)}")
except Exception as e:
    print(f"\nError generating Anki deck: {e}")
```

6.2 Example: Custom Model with Medical Fields and Templates

The medical_model defined above is a robust example of a custom note type tailored for high-yield medical content. It includes specific fields for Question, Answer, Mnemonic, Vignette, ClinicalCorrelation, AdditionalNotes, and Source.
The templates section defines two card types:

- **Card 1 (Basic Q&A)**: Displays the Question on the front. On the back, it shows the Question again ({{FrontSide}}), followed by a horizontal rule, the Answer, and then conditionally displays Mnemonic, Vignette, ClinicalCorrelation, AdditionalNotes, and Source if those fields contain data.
- **Card 2 (Cloze Deletion)**: Uses {{cloze:Question}} on both the front and back to enable cloze deletions directly on the Question field. The back also includes the Answer and other rich content fields, similar to the basic card.

The embedded MY_CARD_CSS ensures consistent and high-quality styling across all cards generated from this model, addressing font, alignment, spacing, and responsiveness.

6.3 Example: Populating Notes with Cloze, Mnemonics, and Vignettes

The add_basic_note and add_cloze_note functions demonstrate how to populate the medical_model fields with data from AI agents.

- **html.escape()**: This function is applied to *all* field content before it is passed to genanki.Note. This is a critical step to prevent malformed HTML or unexpected rendering issues if the AI-generated text contains special characters like <, >, or &.
- **Cloze Syntax**: For cloze cards, the cloze_question parameter directly accepts the text with {{cN::text}} syntax. genanki handles the parsing and card generation based on this syntax and the {{cloze:Question}} directive in the template.
- **Conditional Fields**: By passing empty strings ("") for fields like mnemonic, vignette, or additional_notes when they are not available from the AI agents, the conditional display logic ({{#FieldName}}... {{/FieldName}}) in the template ensures that these sections simply do not appear on the card, maintaining a clean layout.

6.4 Example: Integrated CSS for Optimal Presentation

The MY_CARD_CSS block is a comprehensive solution for the styling issues identified.

- **Readability**: It prioritizes readability by setting a modern font-family, an optimal font-size, generous line-height, and text-align: start for paragraphs.
- **Centering**: The html element is styled with display: flex; justify-content: center; align-items: center; to achieve perfect vertical and horizontal centering of the entire card content. The .card class uses margin: 0 auto; and max-width for horizontal centering and line length control within the card's content area.
- **Visual Structure**: Custom classes like .mnemonic-section, .vignette-section, and .additional-notes-section are defined to provide clear visual separation and consistent styling for rich content elements. Labels are bolded and distinct, improving scanability.
- **Dark Mode**: Basic nightMode support is included to ensure the cards look good when Anki's dark theme is activated.
- **Cloze Styling**: A distinct color is applied to cloze deletions (.cloze) to make them easily identifiable.

This integrated approach ensures that the generated Anki cards are not only functionally correct but also aesthetically pleasing, highly readable, and optimized for the demanding study routines of medical students.

Conclusion: Delivering a Superior Anki Learning Experience

The programmatic generation of high-yield Anki cards for medical students, while powerful, requires meticulous attention to Anki's underlying structure and the precise application of genanki, HTML, and CSS. The challenges encountered, such as incorrect card structure, poor font appearance, missing notes, and improper cloze syntax, are addressable through a systematic approach that prioritizes both functional correctness and optimal user experience. The analysis demonstrates that issues with content not appearing (e.g., "notes section") stem from a need to explicitly define dedicated fields for all information and correctly place these fields within the HTML templates. Similarly, the "structure is all wrong" and "font doesn't look good" problems are fundamentally styling challenges, requiring a comprehensive and modern CSS block to override Anki's outdated defaults. Implementing html.escape() for all field content is critical for data integrity, preventing misinterpretation of special characters as HTML. The precise {{cN::text}} syntax is indispensable for functional cloze deletions, and dedicated fields coupled with conditional HTML display ensure mnemonics and vignettes are presented effectively when available.
By leveraging genanki to define custom models with appropriate fields and templates, embedding robust CSS for readability and centering, and applying careful HTML encoding, the programmatic agent can produce Anki cards that are not merely functional but truly "perfect." These cards will be visually appealing, highly readable, and structured to maximize learning efficiency for medical students, ultimately leading to a superior and seamless learning experience that users would be compelled to continue using. The detailed Python code and styling recommendations provided offer a clear pathway to achieving this high standard.

Works cited

1. kerrickstaley/genanki: A Python 3 library for generating Anki decks - GitHub, https://github.com/kerrickstaley/genanki 2. genanki/README.md at main - GitHub, https://github.com/kerrickstaley/genanki/blob/master/README.md 3. genanki-rs: A Rust Crate for Generating Anki Decks - Crates.io, https://crates.io/crates/genanki-rs 4. genanki_rs - Rust - Docs.rs, https://docs.rs/genanki-rs 5. How to Create and Maximise Anki Card Templates - Memo Cards, https://www.memo.cards/blog/how-to-create-and-use-anki-templates 6. A Step-by-Step Guide to Customizing Anki Card Templates - Memo Cards, https://www.memo.cards/blog/how-to-customize-card-templates-in-anki 7. Field Replacements - Anki Manual, https://docs.ankiweb.net/templates/fields.html 8. Using AI to turn textbooks into cards : r/Anki - Reddit, https://www.reddit.com/r/Anki/comments/1d3m99l/using_ai_to_turn_textbooks_into_cards/ 9. Styling & HTML - Anki Manual - AnkiWeb, https://docs.ankiweb.net/templates/styling.html 10. Modernize default card template for readability and simplicity - Suggestions - Anki Forums, https://forums.ankiweb.net/t/modernize-default-card-template-for-readability-and-simplicity/53356 11. Modernize default card templates for readability: revised proposal and technical details, https://forums.ankiweb.net/t/modernize-default-card-templates-for-readability-revised-proposal-and-technical-details/54669 12. Better Designed Card Templates - Anki Forums, https://forums.ankiweb.net/t/better-designed-card-templates/53270 13. How can I make list elements (`<li>`) more readable, while still being subtle? - Anki Forums,

https://forums.ankiweb.net/t/how-can-i-make-list-elements-li-more-readable-while-still-being-subtle/60144 14. Formatting in Ankiweb - Card Design - Anki Forums, https://forums.ankiweb.net/t/formatting-in-ankiweb/39624 15. Center-left alignment for Cloze deletion - Card Design - Anki Forums, https://forums.ankiweb.net/t/center-left-alignment-for-cloze-deletion/16799 16. Top Anki Decks and Strategies for Medical Students in 2025 - Physeo Blog, https://blog.physeo.com/top-anki-decks-and-strategies-for-medical-students-in-2025/ 17. A Step-by-Step Guide to Using Anki for Medical Studies - Memo Cards, https://www.memo.cards/blog/how-to-use-anki-for-medical-studies 18. mp3 files are noto being packaged/python : r/Anki - Reddit, https://www.reddit.com/r/Anki/comments/1e8l4s6/mp3_files_are_noto_being_packagedpython/ 19. Building Recallr: How I Turned PDFs into Anki Flashcards with AI - DEV Community, https://dev.to/synthpy/building-recallr-how-i-turned-pdfs-into-anki-flashcards-with-ai-133i 20. Go library for generating Anki notes, decks, packages (genanki port) - GitHub, https://github.com/npcnixel/genanki-go 21. workbookpdf.com, https://workbookpdf.com/learn/anki-cloze#:~:text=The%20cloze%20deletion%20card%20is,%3A%3Atext%20to%20hide%7D%7D%20. 22. Anki Cloze Deletion Cards: The Ultimate How To Guide - WorkbookPDF, https://workbookpdf.com/learn/anki-cloze 23. High-Yield Embryology Fact Sheets & Anki Deck - MedSchoolCoach, https://info.medschoolcoach.com/embryology-fact-sheets-anki-deck 24. How would you make a card to memorize a list of things? : r/Anki - Reddit, https://www.reddit.com/r/Anki/comments/11x65g7/how_would_you_make_a_card_to_memorize_a_list_of/ 25. How to hide a card field in anki? - Stack Overflow, https://stackoverflow.com/questions/29183304/how-to-hide-a-card-field-in-anki 26. AnKing/AnkiHub one by one mnemonic formatting : r/medicalschoolanki - Reddit, https://www.reddit.com/r/medicalschoolanki/comments/14fgwxv/ankingankihub_one_by_one_mnemonic_formatting/ 27. Centering Your Templates in #Anki - YouTube, https://m.youtube.com/watch?v=yU2wi9GIbN8&pp=ygUNI3N1YW5raW9nbmljcw%3D%3D 28. Does anyone here know if there is an add-on that you could hide and show information in card? - Anki Forums, https://forums.ankiweb.net/t/does-anyone-here-know-if-there-is-an-add-on-that-you-could-hide-and-show-information-in-card/28759 29. Python: newly made anki deck notes from genanki are invisible - Stack Overflow, https://stackoverflow.com/questions/57774213/python-newly-made-anki-deck-notes-from-genanki-are-invisible 30. [Bug] Importing deck with subdeck generated using genanki-js crash the app. Deck imported successfully on desktop · Issue #9518 · ankidroid/Anki-Android - GitHub, https://github.com/ankidroid/Anki-Android/issues/9518 31. Sharing CSS across multiple cards in Anki - ChrisK91.me, https://chrisk91.me/post/2017-12-20-Sharing-CSS-across-multiple-cards-in-Anki/ 32. AnkiWebView Inspector - AnkiWeb, https://ankiweb.net/shared/info/31746032 33. Mauville/AnkiDroid-Card-Preview - GitHub, https://github.com/Mauville/AnkiDroid-Card-Preview