Agile Breakdown for Markdown-to-PDF Converter

Epics, User Stories, and Story Points

Epic 1: Core Conversion Module

Goal: Develop the core functionality to convert Markdown to PDF with robust formatting and table handling.
Total Story Points: 31

User Story	Story Points	Complexity Considerations
FR1: As a user, I want to input Markdown via file upload or text paste.	5	Basic input handling but requires validation for different formats.
FR2: As a user, I want the PDF to preserve headings, lists, and code blocks.	8	Requires integration with Markdown parser and CSS styling.
FR3: As a user, I want wide tables to render without overflow (via CSS auto-scaling and font adjustments).	8	Complex due to dynamic table resizing and cross-browser PDF compatibility.
FR4: As a user, I want PDFs auto-adjusted to A4 size by default.	5	Requires page size configuration and content scaling logic.

User Story	Story Points	Complexity Considerations
FR5: As a user, I want to add configurable text watermarks (position/ rotation).	5	Moderate complexity (backend logic for watermark placement).

Epic 2: User Interface Development

Goal: Build an intuitive GUI for input, preview, and configuration.
Total Story Points: 18

User Story	Story Points	Complexity Considerations
UI1: As a user, I want a basic GUI with a text area and file upload button.	5	Simple UI setup but requires integration with backend.
UI2: As a user, I want a real-time preview panel for Markdown rendering.	8	High complexity due to live rendering synchronization.
UI3: As a user, I want a section to configure watermark settings (e.g., text, position).	5	Requires state management and interaction with core module.

Epic 3: Deployment & Non-Functional Requirements

Goal: Ensure cross-platform compatibility, performance, and packaging.
Total Story Points: 29

User Story	Story Points	Complexity Considerations
NFR1: As a user, I want PDF generation under 5s for ≤50-page documents.	8	Requires optimization of HTML-to-PDF conversion pipeline.
NFR2: As a user, I want the tool to work on Windows, macOS, and Linux.	8	High complexity due to dependency management (e.g., wkhtmltopdf bundling).
NFR3: As a user, I want the tool to be usable within 30 seconds (intuitive design).	5	Moderate (UI/UX refinement and documentation).
Deliverable: As a developer, I want executables packaged with PyInstaller and bundled dependencies.	8	Complex due to platform-specific builds and testing.

Epic 4: Risk Mitigation & Additional Features

Goal: Address risks and implement optional features.
Total Story Points: 23

User Story	Story Points	Complexity Considerations
	8	

User Story	**Story Points**	**Complexity Considerations**
Risk1: As a user, I want horizontal scroll for tables exceeding page width.		Requires CSS/PDF rendering adjustments for overflow.
Risk2: As a developer, I want to use wkhtmltopdf to ensure cross-platform consistency.	5	Moderate (integration and testing).
Risk3: As a user, I want preset watermark positions (e.g., center, diagonal) to avoid alignment issues.	5	Backend logic for preset configurations.
Optional: As a user, I want a CLI for batch conversions.	5	Moderate (additional interface layer).

Notes:
1. **Story Points** follow the Fibonacci scale (1, 2, 3, 5, 8).
2. Epics are balanced to stay under 35 points each.
3. Optional features (e.g., CLI) are included in Epic 4 for flexibility.

Next Steps: Prioritize Epics 1 and 3 first to establish core functionality and deployment readiness.