# WYSIWYG Web Design Tool — MVP Implementation Plan & Acceptance Criteria (v1)

This document turns the MVP requirements into an actionable plan with architecture, state design, acceptance criteria, test cases, and milestones. Scope is strictly MVP, with clearly marked out-of-scope items for later phases.

## 1) Executive summary

**Goal:** Ship a visual, code‑free layout tool that outputs valid, production‑ready HTML/CSS using web‑native primitives (flow, Flexbox, optional CSS Grid, absolute/fixed when toggled).

**Guiding constraints:** - Rectangles/boxes only; no overlapping except when explicitly absolute/fixed. - What‑you‑see‑is‑what‑you‑get at each breakpoint. - WCAG 2.1 AA guidance integrated (basic checks) and zero “impossible” CSS states through the UI.

**Primary outputs:** HTML + CSS export, on‑canvas preview, code preview pane.

## 2) High‑level architecture

* **Frontend:** React 18 + Redux Toolkit + dnd‑kit.
* **Rendering:** Canvas is a DOM-based designer (not <canvas>), leveraging CSS for layout so preview fidelity is native.
* **Styling:** CSS Modules (default) with build‑time Autoprefixer. Option to switch to Styled Components post‑MVP.
* **Code Editor:** Monaco Editor (read‑only with guarded “Refine” mode).
* **Validation:** Lightweight rules engine for CSS/HTML checks; color contrast calc; keyboard traps; missing alt.
* **Undo/Redo:** Command system over Redux actions with history compaction.

Note: No backend required for MVP beyond local storage/project file import/export.

## 3) Core domain model (Redux state)

Use normalized entities and per‑breakpoint overrides. Types below are indicative.

// Breakpoint identifiers used as keys  
export type BreakpointId = 'mobile' | 'desktop' | 'large';  
  
export interface ProjectState {  
 meta: { id: string; name: string; createdAt: string; updatedAt: string };  
 ui: UIState;  
 canvas: CanvasState;  
 rectangles: EntityState<RectangleNode>; // normalized graph of boxes  
 styles: StylesState;  
 components: ComponentsState; // built‑ins + styled variants  
 breakpoints: BreakpointsState; // widths + active bp  
 history: HistoryState; // undo/redo stacks  
 export: ExportState; // generated code snapshots + validation  
}  
  
export interface RectangleNode {  
 id: string;  
 parentId: string | null;  
 children: string[];  
 role: 'container' | 'text' | 'image' | 'component';  
 // layout props (flow by default)  
 layout: {  
 engine: 'flex' | 'grid';  
 direction?: 'row' | 'column';  
 gap?: number; // px  
 gridTemplate?: string; // e.g., '1fr 1fr' (MVP: simple templates)  
 };  
 position: {  
 mode: 'flow' | 'absolute' | 'fixed';  
 top?: Unit; left?: Unit; width?: Unit; height?: Unit; zIndex?: number;  
 };  
 spacing: { margin?: Spacing; padding?: Spacing };  
 minSize?: { width?: number; height?: number }; // enforced at runtime  
 content?: TextContent | ImageContent | ComponentRef;  
 classNames: string[]; // applied CSS classes (from StylesState)  
 perBreakpoint?: Partial<Record<BreakpointId, Partial<RectangleNode>>>; // overrides  
}  
  
type Unit = { value: number; unit: 'px' | '%' | 'vw' | 'rem' };  
  
interface TextContent { kind: 'text'; text: string; typographic?: TypoProps }  
interface ImageContent { kind: 'image'; ratio?: '16:9'|'4:3'|'1:1'|'3:2'|'custom'; alt?: string }  
interface ComponentRef { kind: 'component'; key: string; stateVariant?: ComponentState }  
  
interface StylesState {  
 classes: Record<string, StyleClass>; // key = className  
 order: string[]; // draggable order to control cascade/specificity  
 tokens: { colors: Record<string,string>; spacing: number[]; fonts: string[]; };  
}  
  
interface StyleClass {  
 name: string; // semantic suggestion e.g., 'btnPrimary', 'textMuted'  
 selectors: string[]; // base + pseudo e.g., ['.btnPrimary', '.btnPrimary:hover']  
 rules: CSSRule[]; // limited guarded set in Basic/Advanced modes  
 scope?: 'character' | 'component';  
 breakpoints?: Partial<Record<BreakpointId, CSSRule[]>>; // overrides  
}

## 4) Project file format (JSON)

Single JSON file serializing ProjectState plus a CSS bundle string for export previews. Example (abridged):

{  
 "meta": {"id":"p1","name":"Landing","createdAt":"2025-08-19"},  
 "breakpoints": {"active":"desktop","defs":{"mobile":[320,768],"desktop":[769,1200],"large":[1201,9999]}},  
 "rectangles": {  
 "entities": {  
 "root": {"id":"root","parentId":null,"children":["r1"],"role":"container","layout":{"engine":"flex","direction":"column"},"classNames":["container"]},  
 "r1": {"id":"r1","parentId":"root","children":["r2","r3"],"role":"container","layout":{"engine":"flex","direction":"row","gap":24},"classNames":["row"]}  
 },  
 "ids": ["root","r1"]  
 },  
 "styles": {"classes": {"container": {"name":"container","selectors":[".container"],"rules":[{"prop":"max-width","value":"1200px"}]}}, "order": ["container"]}  
}

## 5) Interaction design & algorithms

### 5.1 Split

* **Gesture:** Shift + drag from a rectangle edge; or context menu → Split H/V.
* **Algorithm:**
  1. Determine split axis based on edge or command.
  2. Insert two child rectangles (siblings) within the parent; copy styling from source.
  3. Distribute space via flex-basis (50/50 initially) or grid template if parent uses grid.
  4. Enforce min sizes: interactive 32px, text containers 16px. Apply sticky resistance near limits.
* **Acceptance:** New rectangles inherit styles; layout remains valid at all breakpoints; history logs one composite action.

### 5.2 Merge

* **Gesture:** Right‑click on shared border → Merge; or drag border to zero → auto‑merge.
* **Algorithm:**
  1. Detect adjacency (same parent, contiguous in axis).
  2. Remove one node, expand sibling flex-basis to fill; merge classNames (de‑dupe) and preserve common content only if single child had content.
  3. Update selection to merged node; maintain history.
* **Acceptance:** No orphaned references; min sizes respected; undo restores prior state.

### 5.3 Absolute/Fixed toggle

* Change position.mode; reveal drag handles; constrain moves within parent unless explicitly unlocked.
* Units picker (%, vw, rem, px) with live conversion.

### 5.4 Visual feedback

* Hover/selection outlines; guides for spacing; breadcrumb/hierarchy pill showing parent → child path.

## 6) Content system

### 6.1 Text

* Inline editing with presets: Short Headline / Medium Paragraph / Long Article / Lorem Ipsum (S/M/L).
* Properties: font family/size/weight, color, line height, letter spacing, text align.
* Right‑click → Replace content → pick preset or paste custom.

### 6.2 Images

* Placeholders drawn as responsive boxes with ratio badges; object-fit controls; alt text prompt (enforced before export).

**Acceptance:** Typing or replacing content never breaks layout; text reflows across breakpoints; images keep ratio.

## 7) Styling & class management

* **Live class generation** with semantic suggestions (e.g., from role + intent: btnPrimary, headingXL).
* **Specificity control** via draggable class order; computed cascade preview.
* **Granularity modes:**
  + *Basic:* spacing, color, typography, borders, background.
  + *Advanced:* flex/grid knobs, position, z‑index, transforms.
  + *Expert:* guarded freeform CSS editor with validation and lint (disallow layout‑breaking props).
* **Style palette:** character vs component styles; apply as additive (merge rules) or replacement (swap class list).
* **Conflict warning:** if target has conflicting declarations for same property group.

**Acceptance:** Applying or reordering classes updates canvas within 300ms; no duplicate rules in exported CSS; warnings appear on conflicts.

## 8) Component system (built‑ins)

* **Components:** Button, Link, Input (text/email/submit), Image, Text container.
* **States:** base, :hover, :focus, :active, [disabled] via visual State Editor tabs.
* **Preview mode:** simulate hover/focus/active with keyboard toggles.

**Acceptance:** State styles map to valid selectors; keyboard can cycle states; disabled state blocks focus.

## 9) Responsive design

* Default breakpoints: Mobile (320–768), Desktop (769–1200), Large (1201+).
* **Per‑breakpoint editing:** property panels scoped to active bp; side‑by‑side compare view.
* **Cascade:** base → bp override; preview shows effective value per bp.

**Acceptance:** Switching breakpoints updates property inspectors and canvas instantly; exports group overrides under @media queries.

## 10) Code generation & export

### 10.1 HTML

* Deterministic structure from rectangle tree; semantic tags where role permits (e.g., text container → <section>/<article> optional post‑MVP; MVP uses <div>/<p>/<img> responsibly).
* ARIA labels added only when necessary (no redundant roles).

### 10.2 CSS

* One class = one concern bias; dedupe identical rule sets; order from StylesState.order; breakpoint overrides grouped per class.
* Use CSS variables for palette tokens.

### 10.3 Validation & preview

* W3C CSS/HTML rule subset checks; surface errors inline; code preview is syntax‑highlighted and read‑only by default.

### 10.4 Export formats

* Zipped HTML/CSS package; PNG/JPG snapshot via DOM‑to‑image; PDF via print CSS.

**Acceptance:** Exported package renders identical in a fresh tab at each breakpoint; validator shows no errors; images include alt text.

## 11) Accessibility (WCAG 2.1 AA — MVP checks)

* **Color contrast:** check text/background pairs (4.5:1 normal, 3:1 large text). Offer remediation suggestions.
* **Semantics:** ensure interactive elements are focusable and have visible focus; no keyboard traps; form inputs have labels.
* **Images:** require alt or mark as decorative.
* **Structure:** warn if heading levels skip or page lacks a landmark container.

**Acceptance:** Running “Preview‑time checks” lists issues with element references and fix suggestions.

## 12) Performance & complexity

* **Debounced CSS build:** 300ms after last change; incremental stylesheet patching.
* **DOM complexity meter:** warn when depth > N or children per node > M (MVP: N=10, M=12).
* **Frame budget:** UI interactions under 100ms; record long tasks and display banner if sustained.

**Acceptance:** On a 2020 mid‑tier laptop, initial load < 3s; canvas interactions remain <100ms average.

## 13) Undo/redo system

* **Model:** Command pattern wrapping Redux actions; group micro‑edits into a single command (e.g., continuous drag → one entry).
* **Scopes:** Global history + scoped history when editing a component in isolation.
* **Visualization:** Timeline list with labels (“Split r3 vertically”, “Apply class .btnPrimary”).

**Acceptance:** Ctrl/Cmd+Z/Y works globally; scoped undo only affects the focused component editor.

## 14) Keyboard & usability

* **Shortcuts (MVP):**
  + Select parent: Esc or Alt+↑; select first child: Alt+↓.
  + Split H/V: Ctrl/Cmd+Shift+H / Ctrl/Cmd+Shift+V.
  + Toggle absolute: Ctrl/Cmd+Shift+A.
  + Nudge position/size: arrows (1px); with Shift (10px); with Alt (0.1rem).
  + Undo/Redo: Ctrl/Cmd+Z / Ctrl/Cmd+Shift+Z.

**Acceptance:** All actions have accessible equivalents (not mouse‑only) and are discoverable via Command Palette.

## 15) UI layout (workspaces)

* **Primary:** Canvas with left toolbar (select/hand/split/merge), right properties panel, bottom status.
* **Secondary panes:** Responsive preview (split view), Interaction testing (state toggles), Skeleton view (outline mode).
* **Progressive disclosure:** Mode switcher: Basic / Advanced / Expert. Tooltips + inline “?” help.

**Acceptance:** Switching modes never loses information; Expert reveals additional controls for the same selected node.

## 16) Testing strategy

* **Unit:** reducers/selectors, layout math (split/merge), CSS generation.
* **Component:** property panels, class order drag, state editor.
* **E2E:** Playwright flows — create layout, style, set breakpoints, export; assert pixel equivalence thresholds per bp.
* **Accessibility tests:** axe‑core in preview; color contrast unit.

**Acceptance:** CI runs < 8 minutes; core flows green on Chrome+Firefox; a11y checks < 0.5s per run.

## 17) Milestones & deliverables (16 weeks)

**Phase 1 (Weeks 1–4): Core Layout** - Rectangle graph + renderer; split/merge; selection/hover; basic text/image; min‑size constraints; basic undo/redo; skeleton view.

**Phase 2 (Weeks 5–8): Styling & Components** - Styles state + palette; class order control; built‑in components + State Editor; semantics for buttons/links/inputs.

**Phase 3 (Weeks 9–12): Responsive & Export** - Breakpoints + per‑bp overrides + side‑by‑side; code preview; HTML/CSS export; PNG/JPG/PDF; basic validators.

**Phase 4 (Weeks 13–16): Polish & Perf** - Performance monitors; DOM complexity warnings; UX refinements; documentation + onboarding walkthrough.

## 18) Detailed acceptance criteria by epic

### 18.1 Core Layout System

* Split & merge maintain valid tree; sticky drag near 32px/16px; absolute/fixed mode exposes handles and unit picker; guides show margins/padding.

### 18.2 Content

* Inline text editing supports presets and custom; images enforce ratio; alt required before export.

### 18.3 Styling

* Creating a class updates canvas within 300ms; reordering classes changes cascade; conflict dialog shows property groups in conflict.

### 18.4 Components

* Buttons/links/inputs expose state tabs; preview toggles state without affecting saved rules.

### 18.5 Responsive

* Editing at “mobile” creates overrides only for that bp; switching to “desktop” shows base unless overridden; reflow preview animates.

### 18.6 Export

* Exported HTML/CSS opens in a blank project and matches canvas within 1px tolerance for spacing/size; W3C lint passes; no duplicate CSS rules.

### 18.7 Accessibility

* Running checks lists issues with element IDs; contrast failures suggest nearest token pair that passes.

### 18.8 Undo/Redo

* Continuous drags compress to one history entry; per‑component editing has isolated history.

## 19) Non‑functional requirements

* **Performance:** as stated; guardrails for complex layouts.
* **Reliability:** autosave to local storage every 10s + on blur; project recovery dialog.
* **Internationalization:** UI copy in English for MVP; structure prepared for i18n.
* **Security/Privacy:** entirely client‑side; no external calls in MVP.

## 20) Telemetry & success metrics (opt‑in)

* Event counts: time‑to‑first‑layout, successful export, errors surfaced.
* Funnel: created layout → styled → responsive → exported.

## 21) Open decisions (recommended defaults)

1. **Class naming:** semantic, camelCase (btnPrimary, headingXL).
2. **CSS ordering:** tokens first, then character styles, then components.
3. **Absolute positioning bounds:** constrained to parent by default; unlock toggle allowed.
4. **Grid support:** MVP offers simple 2–4 column templates; advanced templates post‑MVP.
5. **Validation scope:** lint only the subset the UI can generate; warn (not block) on expert freeform rules.

## 22) Risks & mitigations

* **CSS generation complexity:** start with whitelist of properties per mode; incremental growth.
* **Performance on deep trees:** virtualization of selection overlays; batch DOM writes via React concurrent mode.
* **Learning curve:** contextual tips + Command Palette + skeleton view to teach structure.

## 23) Definition of Done (per epic)

* Feature demo recorded; acceptance criteria met; a11y smoke passes; export parity validated; docs updated.

## 24) Backlog seeds (initial tickets)

1. Rectangle graph: CRUD + render
2. Selection model + hover outlines
3. Split/merge algorithms with min‑size constraints
4. Absolute/fixed toggle + handles + unit converter
5. Text tool with presets + inline editor
6. Image placeholder tool with ratios + alt prompt
7. Styles state + class creator + order control UI
8. Style palette application (additive vs replace)
9. Built‑in components (Button/Link/Input/Image/Text container)
10. State Editor for pseudo states
11. Breakpoint manager + overrides
12. Code preview pane (Monaco)
13. Export: HTML/CSS + PNG/JPG + PDF
14. Validators (HTML/CSS subset + contrast)
15. Undo/redo command system + history panel
16. Perf monitors + complexity warnings
17. Onboarding walkthrough + tooltips

## 25) Appendix — CSS export rules (MVP)

* No IDs for styling; classes only.
* No inline styles in export (editor may use inline while editing, but export strips them).
* Use box-sizing: border-box; globally.
* Media queries per breakpoint:

@media (min-width:769px) and (max-width:1200px) { /\* desktop \*/ }  
@media (min-width:1201px) { /\* large \*/ }

* Palette tokens via :root { --color-primary: ... } etc.; classes reference tokens.

**End of document**