# Arclight

A Report for an AI Powered Mobile Photo Editor Envisioned for 2030

Design and AI Report for Inter IIT Tech Meet 14.0

Submission: December 4, 2025
Team 94

This document contains the design rationale, screen breakdowns, interaction models, market scan and appendices for Arclight and the AI models used, with disclosure of datasets, nature of models, detailed analysis of performance and main workflow.

## Contents

# 1. Executive Summary

Arclight is an AI powered mobile photo editor designed with a horizon of 2030 in mind. The app presents two workspaces. Flow Space is for casual creators who prefer guided, streamlined edits. Lab Space is for users who want full control and layered editing. The design emphasis is on clarity, calm, and professional visuals. The app aims to show AI assistance without removing user control. Two AI workflows are central to the product. The first is background replacement. The second is auto enhancement. Both workflows are surfaced through layer based controls.

This report documents the design philosophy, user research findings, the information architecture, interaction model, the design system, a medium detail walkthrough for each screen, workflow documentation for AI features, accessibility considerations, and a roadmap for future improvements. Also, this report gives a detailed walkthrough of the architecture in the app-dev and AI aspects of the app.

# 2. Vision for 2030 and Design Philosophy

## 2.1. Vision for 2030

By 2030 AI will be integrated into mainstream creative tools. Users will expect seamless transitions between manual operations and automated refinements. Hardware capabilities will allow a greater portion of AI tasks to run on device while still relying on cloud compute for larger and heavier operations. The design philosophy for Arclight anticipates this environment and builds an interface that treats AI as a natural extension of user intent.

## 2.2. Core Design Principles

- **Clarity:** interfaces simplify decision making.
- **Control:** AI assists but does not dictate outcomes.
- **Calmness:** avoid visual noise to keep focus on the image.
- **Speed:** reduce friction and perceived waiting time.
- **Predictability:** results must be reversible and understandable.

## 2.3. Minimalism and Anti Skeuomorphism

Minimalism and anti skeuomorphism are core visual principles for Arclight. These principles are applied so the interface supports rapid decision making, reduces cognitive load, and keeps user attention on the image being edited. The section below explains the concepts, gives practical rules for implementation, and lists exceptions and accessibility notes.

### 2.3.1. Concepts and Goals

Minimalism means removing unnecessary visual elements and focusing only on what is needed for the user to complete their task. The goal is a clean interface that lets users act quickly and recover easily. Anti skeuomorphism means avoiding visual metaphors that mimic physical materials or devices such as faux leather, heavy shadows, or pseudo 3D buttons. Instead the interface uses clear geometry, simple fills, and semantic iconography.

**Intended effects**

- Reduced distraction so the image content is the dominant visual element.
- Faster comprehension of tool function through clear labels and simple icons.
- Predictable interactions where a control looks like what it does.

### 2.3.2. Practical Implementation Rules

The following rules translate the above principles into design tokens and component rules.

**Surfaces and depth**

- Use flat, filled surfaces with subtle contrast instead of shadows to indicate layers or sheets.
- Reserve contrast increments for hierarchy. Example: primary surface 10 percent lighter or darker than canvas background to indicate a floating panel.
- Avoid drop shadows and bevels. If depth is necessary use thin outlines or a single-step contrast change.

**Color and contrast**

- Primary dark mode background: #1A1A1A. Primary light mode background: #E8E5D8.
- Accent color #3C44A8 is used sparingly for state and calls to action only.
- Maintained WCAG AA for all text on primary surfaces. Refer to the appendix contrast table to validate each UI combination.

**Iconography and micro graphics**

- Used filled, geometric icon shapes. Stroke thickness is 4 px for a 100 px design enclosure. Scaled the stroke proportionally for smaller contexts.
- Icons must be semantic and avoid ornamental details. Prefer single pictograms with no inner gradients.

**Typography and labels**

- Kept labels short and actionable. Prefer verbs for buttons such as Create, Apply, Replace.
- Headings use the display font. Body copy uses the neutral UI font. Maintain consistent spacing and line height to aid scanning.

**Spacing and rhythm**

- Used an 8 point grid as the baseline. Multiples of 8 px are the default spacing units.
- Maintained consistent internal padding for panels and consistent external margins around the canvas.

### 2.3.3. Interaction and Motion

Motion should be purposeful and minimal to avoid distraction.

- Use short, linear transitions for simple state changes such as toggling panels or opening a modal.
- Use scale plus opacity changes for transient overlays rather than large translations.
- Avoid motion that implies physical forces. Keep motion consistent across the app so users can form a mental model.

### 2.3.4. When to break the rule

Minimalism is not an absolute. Use richer visuals only when they add measurable clarity or reduce error rates.

- Introduce a subtle highlight or illustrative help panel for first time users in a workspace.
- Temporarily surface more affordances for complex operations, then collapse them after completion.

### 2.3.5. Accessibility and Inclusivity Notes

- Ensure all control targets meet minimum touch size of 44 by 44 px.
- Provide sufficient contrast for text and interactive elements. If color is used to convey state, add an additional non color indicator such as an icon or text label.
- All gestures must have alternative on screen controls so users who do not use gestures can reach the same actions.

#### 2.3.6. Checklist for designers and engineers

1. No decorative shadows or texture layers are present on core UI components.
2. All primary actions use the accent color in no more than two places per screen.
3. Icon stroke scaling matches the 4 px at 100 px guideline and scales down proportionally.
4. Spacing follows the 8 point grid. Exceptions are documented with reasoning.
5. Motion timings are short and consistent across similar transitions.

The above rules and checks provide a usable and consistent baseline for Arclight. They keep the interface simple without sacrificing clarity and give concrete guidance to implementers so designs remain faithful to the principles.

## 3. User Research and Insights

### 3.1. Research Overview

A user survey was conducted among individuals primarily between the ages of 18 and 24. The goal of the survey was to understand how users edit photos on mobile devices, which features they value most, and what role they expect AI to play in their workflows. A total of 87 responses were recorded.

#### 3.1.1. Device Usage and Editing Frequency

- 78 percent of respondents edit photos at least once a week.
- 41 percent edit photos several times per week.
- 92 percent use their phone as their primary device for photo edits.
- 8 percent use a laptop or desktop for all edits except occasional mobile adjustments.

#### 3.1.2. Most Commonly Used Editing Apps

- Snapseed: used by 64 percent of respondents.
- PicsArt: used by 57 percent of respondents.
- Instagram filters: used by 71 percent of respondents.
- Lightroom Mobile: used by 36 percent of respondents.
- VSCO: used by 18 percent of respondents.

#### 3.1.3. AI Feature Familiarity

- Auto enhancement: 69 percent familiarity.
- Background removal or replacement: 53 percent familiarity.
- Face retouching: 41 percent familiarity.
- Style or filter generation: 38 percent familiarity.
- Sky replacement: 27 percent familiarity.

#### 3.1.4. Features Users Want AI to Perform

- Masking and subject isolation: selected by 74 percent.
- Global enhancement: selected by 62 percent.
- Background replacement: selected by 59 percent.
- Color grading suggestions: selected by 48 percent.
- Layer based retouching help: selected by 33 percent.

#### 3.1.5. Key Pain Points

- 53 percent said masking takes too long or is too difficult.
- 46 percent said it is hard to get the exact look they want.

- 39 percent mentioned slow rendering or lag in existing apps.
- 25 percent said they struggle with unclear UI layouts or tool overload.

The numerical results motivated the design direction of Arclight in favour of fast AI assistance, minimal friction, a predictable interface, and a clear transition from simple editing to expert control.

### 3.2. Personas

The personas below represent two primary user groups identified through survey data and qualitative insights. They guide decisions regarding visual design, complexity, navigation and the balance between AI automation and manual control.

#### 3.2.1. Persona 1: The Casual Creator

**Age:** 21
**Goal:** Create visually appealing images with minimal effort and share them online.

**Motivations**
- Wants fast, polished results with minimal steps.
- Enjoys experimenting but does not want complex tools.
- Prefers apps that feel fluid, light and friendly.

**Pain Points**
- Unclear tool hierarchies in many apps.
- Masking and refinements feel tedious on mobile.
- Too many tool options lead to cognitive fatigue.

**Editing Habits**
- Uses Snapseed, PicsArt and Instagram filters frequently.
- Performs quick edits several times per week.
- Comfortable with gestures and expects responsive interactions.

**Implications for Arclight**
- Flow Space must be lightweight, predictable and welcoming.
- Prompt suggestions reduce effort for common adjustments.
- AI should reduce friction while keeping users in control through layer opacity.

#### 3.2.2. Persona 2: The Project Specialist

**Age:** 30
**Goal:** Perform targeted, high quality edits for professional or semi professional work.

**Motivations**
- Values precision and consistency.
- Wants to minimise repetitive manual tasks.
- Prefers fast tools without sacrificing accuracy.

**Pain Points**
- Manual masking is time consuming and error prone.
- Many mobile apps hide advanced features or oversimplify them.
- Existing AI tools often lack predictable refinements.

**Editing Habits**
- Primarily uses desktop tools such as Photoshop or Lightroom.
- Uses mobile only for targeted quick adjustments.
- Works methodically with structured workflows.

**Implications for Arclight**
- Lab Space must provide precise, layer oriented editing.
- AI outputs should be fully editable and non destructive.
- Navigation must be predictable with minimal abstraction.

### 3.3. Key Insights

The combination of survey data and qualitative interviews produced strong themes that influenced Arclight's design.

#### 3.3.1. Users Dislike Manual Masking

Masking was the most common frustration. More than half of respondents stated that isolating regions manually is difficult or slow. This justified integrating AI based subject detection and automated background replacement.

#### 3.3.2. Users Often Have a Vision but Cannot Execute It

Users frequently described knowing what they want their final image to look like but lacking the tools or clarity to achieve it. Arclight addresses this through workflow guidance, prompt suggestions and clean entry points for adjustments.

#### 3.3.3. Speed Is Essential and Waiting Breaks Flow

Slow rendering or lag undermines trust in the interface. Arclight reduces friction by avoiding heavy blocking states, using short feedback animations and generating previews where possible.

#### 3.3.4. Users Want Real Time Previews of Adjustments

Previews help users understand changes without committing to them. Arclight treats every AI enhancement as a separate layer with instant toggling and opacity control.

#### 3.3.5. Complexity Should Scale With Expertise

Casual users benefit from simple, guided tools. Professional users require precise layer based workflows. This motivated the dual workspace model, where Flow Space offers simplicity and Lab Space provides full control.

#### 3.3.6. Navigation Must Be Predictable

Respondents disliked inconsistent tool placement in competitor apps. Arclight uses a stable top bar and sidebar, plus mode locked canvases, to maintain clear expectations and reduce accidental context switches.

These insights form the foundation of Arclight's visual language, navigation patterns and feature prioritisation.

## 4. Market Scan

The market scan investigates how Arclight fits within the current ecosystem of mobile photo editors. It identifies strengths and weaknesses in competing applications, highlights user frustrations, and validates the need for a dual workspace system that balances simplicity and professional control. This section covers the competitive set, a structured comparison, and a detailed analysis of workflow friction and gaps.

### 4.1. Selected Competitors

The applications chosen for comparison represent a spectrum from casual consumer tools to advanced professional editors. They were selected based on survey responses, app store popularity and relevance to the workflows Arclight aims to support.

- **Lightroom Mobile** A professional grade editing app with granular controls but a high learning curve.

- **Photoshop for iPad** A powerful but complex application. It offers strong precision tools but limited speed for casual edits.
- **Snapseed** A lightweight editor with a good set of tools but minimal AI assistance.
- **PicsArt** Feature rich with heavy emphasis on creative effects, but often cluttered and inconsistent.
- **VSCO** Known for its preset oriented workflow and strong colour science, less focused on complex edits.

The selection ensures coverage across multiple axes: professional versus casual, guided versus manual workflows, and AI rich versus AI minimal applications.

### 4.2. Feature Comparison Matrix

To examine how Arclight compares with competitors, specific editing features and workflow characteristics were analysed. Table 1 highlights major features, focusing on those most relevant to Arclight's purpose. The matrix emphasises distinctions in layer functionality, AI assistance and preview responsiveness.

| Feature | Arclight | Lightroom | Photoshop | Snapseed | PicsArt | VSCO |
|---|---|---|---|---|---|---|
| Layer based editing | Yes | Partial | Yes | No | Partial | No |
| AI background replace | Yes | No | Partial | No | Partial | No |
| Auto enhancement AI | Yes | Yes | Yes | Partial | Partial | Partial |
| Real time previews | Partial | Partial | Partial | No | No | No |
| Precision masking tools | AI based | Manual | Manual | Limited | Limited | No |
| Workspace specialisation | Yes | No | No | No | No | No |

Table 1: Feature comparison matrix for Arclight and selected competitors.

#### Interpretation of the matrix

- **Layer based control** is rare among mobile editors. Professional tools offer it but are not optimised for mobile ergonomics. Arclight fills this gap.
- **AI workflows** are inconsistent in competitor apps. Many lack a predictable refinement process after AI operations.
- **Real time previews** remain limited across applications due to computational constraints. Arclight focuses on fast, layer based previews which feel immediate and understandable.
- **Workspace specialisation** is unique to Arclight. No competitor provides separate editing environments for casual and expert users.

### 4.3. Workflow Friction and Opportunity Analysis

The qualitative and quantitative data from the survey were combined with hands on competitor testing to identify areas where current apps fall short. These findings directly informed the design of Arclight.

#### 4.3.1. Friction in existing mobile editors

- **Tool overload** Many apps present numerous tools at once which overwhelms casual users. Professionals often struggle to reach advanced tools quickly because they are placed behind multiple layers of menus.
- **Unclear AI behaviour** Some apps apply AI effects without showing how results were generated. This increases user distrust and makes corrections difficult.
- **Manual masking fatigue** A significant pain point is the difficulty of making clean masks using mobile touch input. Users often need several attempts to get precise selections.

- **Slow rendering and preview lag** Heavy filters or high resolution images often cause noticeable delays which disrupt flow and reduce experimentation.
- **Inconsistent navigation patterns** Competitors frequently mix editing and organisational features in the same screens, creating mental model confusion.

#### 4.3.2. Opportunities for Arclight

Arclight was designed to take advantage of gaps revealed by the friction analysis.

**Opportunity 1: A dual workspace model** A major opportunity is the clear separation between **Flow Space** and **Lab Space**.

- Flow Space solves the overload problem by giving casual users a pathway of simple actions and AI powered shortcuts.
- Lab Space offers precision, predictability and full control, attracting professionals.

**Opportunity 2: Layer based AI refinement** Many competitor apps flatten AI results directly into the image. Arclight instead generates results as new layers, allowing:

- opacity control,
- toggling visibility,
- stacking multiple AI effects,
- avoiding destructive workflows.

**Opportunity 3: Predictable navigation** Arclight adopts a top bar plus sidebar model that remains stable across contexts. Unlike competing editors, there are no hidden or shifting tool clusters.

**Opportunity 4: Real time feedback loops** While full real time previews are computationally expensive, Arclight focuses on:

- near instant opacity previews,
- cached previews for repeated operations,
- short and informative loading states.

**Opportunity 5: Reduced cognitive load** The minimalist and anti skeuomorphic design reduces visual noise so that users can focus on the image. This is supported by a small core palette and consistent component shapes.

#### 4.3.3. Summary of strategic insights

Arclight's design direction aligns with observed market gaps:

- Competitors either favour simplicity or complexity but not both. Arclight embraces both by dividing workflows.
- AI features are often opaque. Arclight treats them as transparent, editable steps.
- Mobile ergonomics are under optimised in many apps. Arclight prioritises gesture efficiency and predictable behaviours.

The market analysis validates that Arclight is well positioned to serve both casual and professional users with a thoughtful combination of speed, clarity and control.

## 5. Information Architecture

Information architecture defines how users move through Arclight and how editing contexts are managed. It ensures that every screen, tool and flow is placed in a predictable and logical structure. This section explains the navigation model and provides a high level description of the sitemap that will be illustrated later through diagrams.

### 5.1. Navigation Model

The navigation system in Arclight is designed to support clarity, predictability and speed. It is built around three core principles: stable navigation patterns, clear entry points into editing workflows, and protection against accidental context switching during editing.

#### 5.1.1. Top Navigation Bar

The top navigation bar is visible across most screens. It contains:

- the current workspace label,
- the project level controls such as back, home and export shortcuts,
- context sensitive actions that relate directly to the active editing mode.

The top bar remains consistent so users always have a stable point of reference. It avoids deep nesting of controls and communicates the active context explicitly.

#### 5.1.2. Sidebar as Primary Navigation

The sidebar is the main global navigation element. It appears when the user is outside an active editing session or when explicitly invoked. The sidebar contains:

- access to the project gallery,
- new project creation,
- tutorials and help,
- account settings and preferences.

The sidebar is intentionally kept separate from the editing UI to prevent users from leaving the canvas accidentally. It provides a clear separation between global app functions and editing operations.

#### 5.1.3. Mode Selection Before Editing

Before entering a project, users select either Flow Space or Lab Space. This choice locks the workspace for the duration of the editing session. The decision to lock the mode is intentional because:

- it preserves the mental model for each workspace,
- it prevents accidental switches that could disrupt the workflow,
- it allows the system to optimise the layout and tools more effectively for each mode.

Flow Space relies on light, guided controls. Lab Space maintains a more complex structure. A locked mode ensures each workspace can present a coherent toolset.

#### 5.1.4. Canvas as a Dedicated Editing Environment

Once inside an editing workspace, the canvas becomes the dominant element. Only tools relevant to the chosen workspace appear. The information architecture ensures:

- minimal navigation during active editing,
- quick tool access with predictable placements,
- clear separation between image content and UI chrome.

Users are not required to navigate back and forth between screens during editing. Every action is triggered from the immediate UI surrounding the canvas.

#### 5.1.5. Consistency Across Screens

To avoid confusion, the following rules are applied throughout Arclight:

- icons, labels and tool positions are consistent across workspaces,
- the back action always leads to the previous stable context,
- important actions such as export or project switching are not hidden behind deeply nested menus.

Consistency supports learnability and reduces cognitive load, especially for casual users who may only use a subset of the app's capabilities.

### 5.2. Sitemap Overview

The sitemap describes how all screens in Arclight relate to each other. The actual figure is attached in the appendix, but here follows the structure explained verbally:

#### High level structure

Arclight's pages are organised into four main clusters:

- **Entry and account screens** such as the Auth Page, Login and OTP verification.
- **Home and project organisation screens** including Home, Project Gallery and New Project.
- **Workspace selection and editing environments** which include Flow Space and Lab Space.
- **Auxiliary screens** such as Export, History, Layers Panel, Tools Panel and the Sidebar.

#### Connectivity and flow

- The user enters through Auth, then proceeds to Home.
- From Home, the user may browse projects or create a new one.
- Creating a project leads to the Mode Selector.
- Selecting a mode opens the appropriate editing workspace.
- Within a workspace, tools, layers, history and export are accessed locally, without leaving the editing context.
- The Sidebar and top navigation bar provide global and contextual pathways back to non editing areas.

#### Diagram placeholder

A full sitemap diagram will be inserted here. It will be represented as a directed graph showing:

- nodes for each screen,
- arrows showing valid transitions,
- clusters showing workspace boundaries,
- clear highlighting of locked pathways inside editing environments.

This diagram will visually communicate the principles explained above and will support both engineering handoff and design documentation.

## 6. Interaction Model

The interaction model defines how users perform actions, manipulate elements on the canvas and access tools across both Flow Space and Lab Space. It focuses on reducing friction, promoting intuitive exploration and making AI powered actions feel natural and predictable. Interaction decisions were informed by user research, mobile ergonomics and the need for consistent refinement workflows.

### 6.1. Core Gestures and Controls

Gestures play a central role in Arclight because they allow quick, low friction actions that reduce dependency on on screen buttons. Each gesture was chosen to align with common mobile paradigms while avoiding conflicts with editing operations.

#### 6.1.1. Primary gestures

- **Drag and drop** Used for placing assets, repositioning elements and organising layers in Lab Space. This gesture reduces the need for modal dialogs and makes layer reordering more tactile.
- **Long press** Invokes contextual options such as layer renaming, duplication or quick actions. Long press avoids clutter and keeps secondary functions discoverable without overwhelming the interface.
- **Swipe from left or right to go back** A familiar mobile interaction pattern that preserves continuity of flow. The gesture takes the user to the previous stable context without invoking unnecessary confirmation dialogs.
- **Tap outside a modal to dismiss** Ensures fast closure of overlays such as tool popovers or reference panels. This promotes fluid navigation and reduces the number of explicit cancel buttons.
- **Tap and hold on sliders for precision control** Pausing on an adjustment slider activates precision mode. This improves fine tuning and supports professional editing habits in Lab Space.

#### 6.1.2. Ergonomic considerations

- All interactive elements meet the minimum touch target size to prevent accidental activation.
- Controls are clustered near the bottom or sides to reduce thumb travel distance.
- Gestures do not overlap with canvas manipulation controls to avoid conflicts between tool selection and image movement.

### 6.2. Prompt and Dictation Interaction

Text prompts provide an alternative interaction mode for users who prefer high level instructions rather than manual adjustments. The prompt interface is placed adjacent to the toolbar so it is always reachable without shifting hand position.

#### 6.2.1. Prompt input field

The text box supports:

- typed prompts,
- voice dictation,
- prompt history suggestions,
- AI generated auto suggestions.

Voice input is triggered by tapping the microphone icon. Auto suggestions appear contextually below the text box. Users can tap a suggestion to insert it directly into the prompt field. This reduces typing effort and guides less experienced users toward effective AI instructions.

#### 6.2.2. Prompt to layer workflow

Prompt based actions generate results as new layers rather than directly modifying the image. This choice:

- maintains non destructive editing,
- allows opacity refinement,
- provides instant comparison using visibility toggles,
- fits naturally within the Lab Space editing paradigm.

#### 6.2.3. Accessibility considerations

Prompt suggestions use high contrast text. Dictation feedback is shown in real time to reassure the user that input is correctly captured. All suggestion chips meet touch target guidelines to

support a wide range of users.

### 6.3. AI Result Handling and Layer Based Refinement

AI driven interactions are integrated carefully to ensure transparency, predictability and user control. Arclight avoids opaque, one click transformations. Instead, every AI operation becomes an editable, reversible step.

#### 6.3.1. AI actions as new layers

Whenever an AI workflow is executed, whether it is background replacement or auto enhancement, the output appears as a separate layer. This approach:

- supports immediate toggling of the effect,
- enables fine grained refinement through opacity,
- allows stacking of multiple AI transformations,
- greatly improves error recovery.

The user never feels locked in to an AI result. This builds trust and invites experimentation.

#### 6.3.2. No intermediate state exposure

Intermediate previews for AI tasks are not shown. This reduces visual noise and prevents confusion when partial or low quality steps appear during processing. Instead:

- A brief loading indicator appears,
- The final output is placed onto a new layer,
- The user can immediately refine or discard it.

This approach is particularly helpful for mobile hardware where intermediate results may be low fidelity or inconsistent.

#### 6.3.3. Refinement through layers

Layer refinement is central to Arclight's editing model. Users can:

- adjust opacity for subtle blending,
- reorder layers for composite effects,
- mask or partially reveal AI outputs,
- combine AI and manual effects seamlessly.

This system addresses the findings from the user research which emphasised that users want AI assistance without losing manual control.

#### 6.3.4. Interaction predictability

Every AI interaction has clear input and output states. Users understand:

- what action they triggered,
- where the result appears,
- how to modify or undo it,
- how it fits into their current workflow.

This predictable pattern supports confidence and reduces cognitive load across both Flow Space and Lab Space.

## 7. Design System

The design system defines the visual language of Arclight. It ensures clarity, consistency and scalability across all screens and interactions. A well defined system is essential for maintaining coherence between Flow Space and Lab Space, while also giving room for growth as features evolve. The following subsections describe the typography, color palette, iconography and spacing rules that form the foundation of Arclight's visual identity.

### 7.1. Typography

Typography shapes the tone of the interface and establishes hierarchy. Arclight uses two complementary typefaces from the Google Fonts family. These were chosen for their clarity, geometric structure and suitability for mobile displays.

#### 7.1.1. Heading font: Dela Gothic One

Dela Gothic One is used for headers, section labels and key visual anchors. Its large geometric forms give structure and presence without adding visual weight. It communicates confidence and aligns with Arclight's focus on control and precision.

**Usage guidelines**

- Use for major screen titles such as Home, Flow Space and Lab Space.
- Avoid using at small sizes because the display oriented forms can reduce legibility.
- Apply generous spacing to maintain a clean, calm visual tone.

#### 7.1.2. Body font: DM Sans

DM Sans is used for all body text, labels, captions, button text and tool descriptions. It is clean, neutral and highly readable on small screens.

**Usage guidelines**

- Use for all paragraph level text, tooltips and secondary labels.
- Maintain a comfortable line height to support readability.
- Avoid mixing weights excessively. Regular and bold weights are sufficient.

#### 7.1.3. Hierarchy and readability considerations

Although Arclight does not enforce a strict typographic hierarchy, consistent sizing improves predictability. The design tokens in the appendix define recommended minimum and maximum sizes for key text categories. Typography aims to:

- keep labels short,
- ensure no text overlaps with core editing regions,
- communicate importance through size and weight rather than decoration.

### 7.2. Color System

The color palette reflects Arclight's minimalist and professional aesthetic. It balances contrast with neutrality and provides a stable foundation for images to stand out. Both light and dark modes use a small, controlled palette for predictability and accessibility.

#### 7.2.1. Dark mode palette

Dark mode is the primary experience for Arclight because it reduces eye strain and emphasizes the image content.

- **Primary background:** #1A1A1A Used for canvas surroundings and high level interface surfaces.
- **Secondary surface:** #E4E4E4 Used for text, high contrast elements and contextual accents.

#### 7.2.2. Light mode palette

Light mode provides a bright alternative for users in high ambient light environments.

- **Primary background:** #E8E5D8 A soft, neutral tone that avoids harsh brightness.
- **Secondary surface:** #222222 Used for text, icons and strong contrast elements.

#### 7.2.3. Accent color

Arclight uses a single global accent to highlight key actions and states. It is used deliberately and sparingly. It is changed on whether the mode is dark or light (in terms of brightness and saturation only), to harmonize better with the other colors, tone-wise.

- **Accent:** #3C44A8

#### 7.2.4. Contrast and accessibility

All text color combinations are validated to meet WCAG AA contrast guidelines. Color is never used alone to indicate state. The appendix includes a complete contrast table for both light and dark themes.

### 7.3. Iconography

Icons in Arclight follow a filled, geometric style that supports quick recognition without visual complexity. They are designed to be clear at small sizes and to work well in both light and dark backgrounds.

#### 7.3.1. Design characteristics

- Filled minimal silhouettes.
- Consistent geometric curvature and corner rounding.
- Stroke thickness equivalent to 4 px at a 100 px enclosure size.
- Scaled proportionally for smaller icon sizes.

#### 7.3.2. Usage guidelines

- Use icons only when they provide visual clarity. Avoid decorative icons.
- Pair icons with labels when actions may not be instantly recognisable.
- Maintain consistent spacing around icons to prevent visual imbalance.

#### 7.3.3. Avoiding skeuomorphism

Icons avoid gradients, faux shadows and physical metaphors. This aligns with Arclight's anti skeuomorphic philosophy and ensures a clean, computational aesthetic.

### 7.4. Spacing and Layout Rhythm

Spacing is one of the most important contributors to perceived clarity. Arclight uses a consistent spacing system derived from an 8 point grid. This creates visual rhythm across screens and reduces cognitive load.

#### 7.4.1. Grid and spacing units

- Core spacing increments: 8, 16, 24, 32 pixels.
- Smaller spacing units (4 px) are used only for fine alignment.
- Large spacing values are used in Flow Space to maintain a calm presentation.

#### 7.4.2. Component padding

Internal component padding uses multiples of the grid. Panels, buttons and input fields maintain consistent padding to support scanning and readability.

### 7.4.3. Canvas centric design

The canvas is always the visual anchor. Spacing around the canvas ensures:

- tools do not overpower the image,
- gestures are not obstructed by UI elements,
- both Flow Space and Lab Space feel balanced even with different tool densities.

### 7.4.4. Breakpoints and responsiveness

Although Arclight focuses on mobile, the spacing system adapts naturally to differing screen widths. Elements scale while maintaining consistent proportions and rhythm.

The spacing and layout system contributes significantly to Arclight's aesthetic coherence by ensuring every element sits comfortably within a predictable structure.

## 8. Screen by Screen Breakdown

This section documents every major screen in Arclight with medium level detail. Each screen description includes its purpose, the core components, key interactions and the design rationale.

### 8.1. Auth Page

**Purpose.** Introduces the user to Arclight and provides entry into the authentication flow. Sets a calm, focused first impression.

**Components.**

- App identity mark and minimal title.
- Primary login button.
- Optional help or secondary links.

**Interactions.** Simple tap based initiation of authentication.


Figure 1: Auth Page

### 8.2. Email Login

**Purpose.** Allows users to enter their email to begin the login process.

**Components.**

- Email input field.
- Continue button.
- Keyboard aware layout.

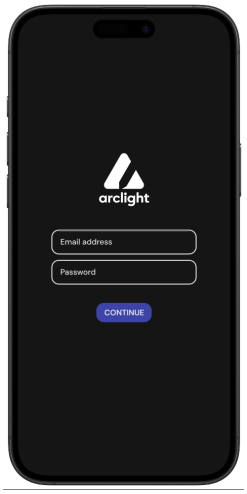


Figure 2: Email Login

### 8.3. OTP Verification

**Purpose.** Verifies identity using a simple four digit OTP entry.

**Components.**

- Four digit OTP cells.
- Resend option.
- Minimal explanation text.

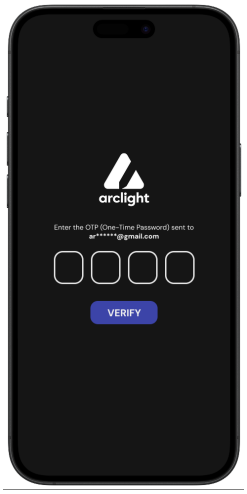


Figure 3: OTP Verification Screen

### 8.4. Home Screen

**Purpose.** Acts as the hub for creating new projects or revisiting existing ones.

**Components.**

- New Project tile.
- Projects Gallery tile.
- Top bar navigation.


Figure 4: Home Screen

### 8.5. New Project Screen and Workspace Selector

**Purpose.** Allows the user to begin a new project and choose the canvas width depending on the workspace. Also allows for choosing which workspace.

**Components.**

- Project creation panel.
- Image selection options.

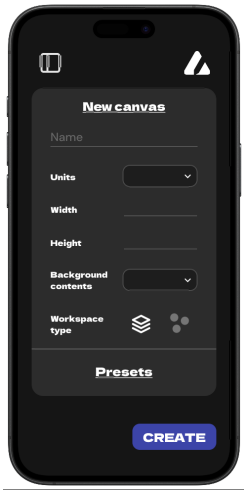


Figure 5: New Project Screen

### 8.6. Flow Space Canvas

**Purpose.** Provides a simple, guided editing environment for casual users.

**Components.**

- Central canvas.
- Minimal toolbar.
- Prompt box with voice dictation.
- Auto suggestion chips.

**Interactions.** Tap suggestions, type or dictate prompts, simple gestures.

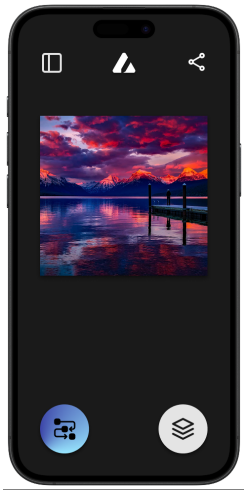

Figure 6: Flow Space Canvas

### 8.7. Lab Space Canvas

**Purpose.** Advanced environment for precise, layer based editing.

**Components.**

- Full layers panel.
- Advanced tools panel.
- Gesture rich canvas.
- Sidebar integration.


Figure 7: Lab Space Canvas

### 8.8. Tools Panel

**Purpose.** Central catalogue of tools accessed within both workspaces.

**Components.**

- Manual tools such as crop or adjust.
- AI tools including background replace and auto enhance.


Figure 8: Tools Panel

### 8.9. Projects Gallery

**Purpose.** Allows viewing, opening and managing past projects.

**Components.**

- Grid of saved projects.
- Thumbnails and timestamps.


Figure 9: Projects Gallery

### 8.10. History Screen

**Purpose.** Provides a chronological list of editing actions and allows users to navigate backwards in the workflow.

**Components.**

- List of operations.
- Layer and tool indicators.

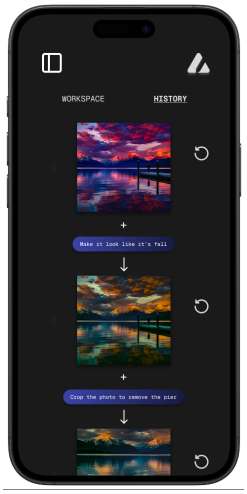


Figure 10: History Screen

### 8.11. Layers Panel

**Purpose.** Displays all layers created during editing, including AI generated effects.

**Components.**

- Thumbnail for each layer.
- Visibility toggle.
- Opacity slider.
- Reordering handle.

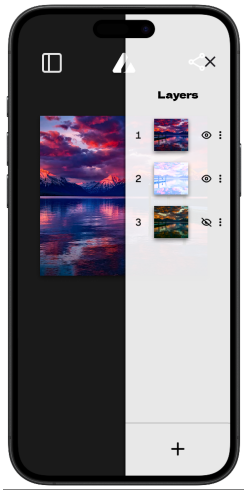


Figure 11: Layers Panel

### 8.12. New Layer Screen

**Purpose.** Allows users to insert a new layer within Lab Space. This includes both image layers and adjustment layers, supporting non destructive editing and giving users explicit control over the structure of their composition.

**Components.**

- Large, easily identifiable buttons for creating an **Image Layer** or an **Adjustment Layer**.
- Center aligned circular icons that follow Arclight's geometric iconography style.
- A transient overlay that appears above the canvas and dismisses upon completion or cancellation.
- A small cancel button to exit the layer creation flow without changes.

**Interactions.**

- Tap "Image Layer" to add a new image based element to the composition.
- Tap "Adjustment Layer" to apply non destructive transformations such as curves or color adjustments.
- Tap the cancel icon or outside the modal to dismiss.

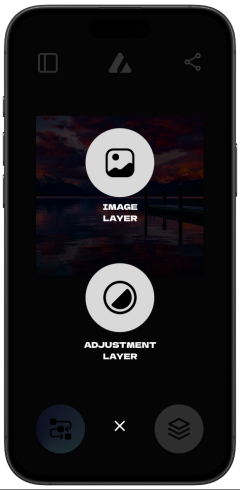


Figure 12: New Layer Screen

### 8.13. Sidebar

**Purpose.** Provides global, out of canvas navigation to non editing areas.

**Components.**

- Links to Gallery, Help and Settings.
- Minimal iconography.

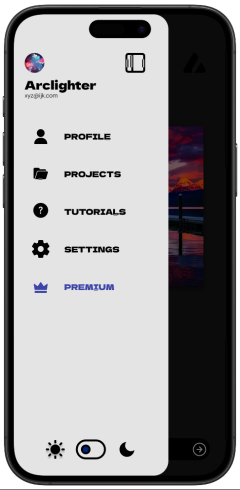


Figure 13: Sidebar

### 8.14. Loading Screen

**Purpose.** Communicates ongoing operations that require computation (and adds a little fun fact to keep the user occupied).

**Components.**

- Spinner or progress bar.
- Short explanatory text.

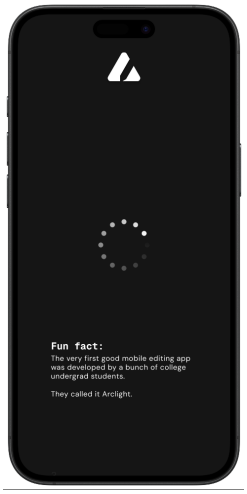


Figure 14: Loading Screen

### 8.15. Export Handling

Arclight does not implement a custom export screen. Instead, it invokes the default export sheet provided by the operating system. This ensures:

- familiar behaviour for users on both iOS and Android,
- system level performance optimisations,
- access to device specific export destinations and share sheets.

### 8.16. Removed Screens

The Settings screen was removed to reduce cognitive load and focus on core user workflows. Most settings provide marginal value during editing and were therefore excluded.

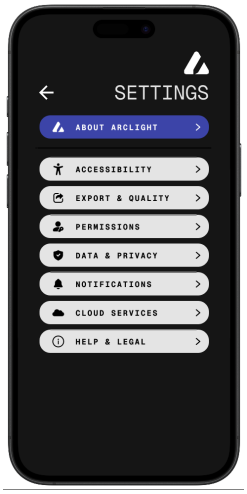


Figure 15: Removed Screen Example: App Settings Page

## 9. Conclusion (for Design Section)

Arclight bridges the gap between casual and professional mobile photo editing through intuitive controls, assistive AI, and a dual workspace model.

## 10. Technical Execution: Workspace 1 (FlowSpace)

This section details the technical implementation of FlowSpace, Arclight's layer-based editing workspace designed for casual creators who prefer guided, streamlined edits.

### 10.1. Core Architecture: Dual-Layer Image Decomposition

The system employs a **dual-layer decomposition approach** as its foundational image processing pipeline. Every uploaded image undergoes automatic segmentation to separate the foreground subject from the background, enabling a variety of downstream editing features.

#### 10.1.1. Image Classification

Upon image upload, the system sends the image to the **Gemini API** for subject classification. The API determines whether the primary subject is a *human* or an *object*. This classification is critical because human segmentation requires specialized handling for complex boundaries such as hair, skin edges, and fine details, whereas general objects can be processed with standard segmentation models.

#### 10.1.2. Subject Segmentation using U2Net

Based on the classification result, the image is routed to the appropriate segmentation container:

- **Human subjects**: Processed by `u2net-human-seg`, a variant optimized for human contours and fine edge detection.
- **Object subjects**: Processed by the standard `u2net` model for general salient object detection.

U2Net generates a *saliency map* that highlights the subject region. Since our next model in the pipeline needs a binary mask of the subject, post-processing is needed. It converts the subject layer (foreground) into a clean binary mask where background pixels are assigned a value of 0 (black) and subject pixels are assigned a value of 255 (white).

#### 10.1.3. Mask Dilation for Accurate Inpainting

The **LaMa-Cleaner** model used for background inpainting requires a binary mask indicating the region to be filled. However, directly using the segmentation mask produced by models such as SAM (Segment Anything Model) leads to suboptimal results.

**The Problem: Object Recreation** When the raw segmentation mask is provided to LaMa, the model often attempts to *reconstruct* the subject rather than remove it. This occurs because segmentation masks typically represent a tight boundary around the detected subject, leaving residual pixels—such as edge artifacts, shadows, or fine details like hair strands—outside the masked region. LaMa interprets these residual pixels as contextual information and uses them as references to recreate the subject within the masked area.

**The Solution: Mask Dilation** To ensure proper background inpainting, we apply **morphological dilation** to the segmentation mask before passing it to LaMa. Dilation expands the mask boundary outward by a specified kernel size, effectively:

- Encompassing all residual pixels and edge artifacts within the mask
- Eliminating reference points that could trigger object reconstruction
- Forcing LaMa to rely solely on the surrounding background for inpainting

This preprocessing step ensures that the entire subject region, including peripheral details, is covered by the mask. Consequently, LaMa performs true background reconstruction by extrapolating texture and color information exclusively from the unmasked background regions, producing a clean background plate free of subject remnants.

### 10.1.4. Background Reconstruction using LaMa Inpainting

This binary mask and original image are passed to the **LaMa** (Large Mask Inpainting) model. LaMa employs a deep learning-based inpainting algorithm that intelligently reconstructs the background region occluded by the subject. By analyzing surrounding textures, patterns, and contextual information, inpaints the background with the subject region seamlessly filled in.

### 10.1.5. Layer Generation

The pipeline produces two distinct layers that serve as the foundation for all editing features:

1. **Foreground Layer**: The extracted subject with an alpha channel preserving transparency information.
2. **Background Layer**: The LaMa-inpainted image representing the reconstructed background without the subject.

This dual-layer representation enables non-destructive editing, allowing users to manipulate the foreground and background independently.

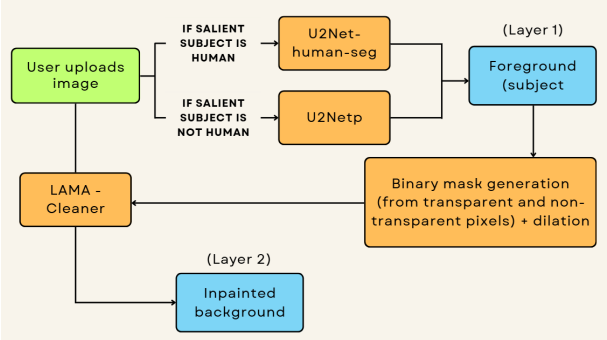


Figure 16: Layer-based editing pipeline

## 10.2. Feature 1: Background Replacement

This feature allows users to replace the original background with a custom image while maintaining photorealistic visual coherence.

### 10.2.1. New Background Compositing

When the user uploads a new background image, the system performs alpha compositing by overlaying the extracted foreground subject onto the new background. At this stage, the composite may exhibit visual inconsistencies due to differences in lighting conditions, color temperature, and ambient tones between the original capture environment and the new background.

### 10.2.2. Image Harmonization using PCT-Net

To address these visual inconsistencies, the composite undergoes a harmonization process:

1. The subject is re-segmented using U2Net to generate a fresh binary mask aligned with the composite.
2. The mask and composite image are passed to **PCT-Net** (Pixel-wise Color Transform Network).
3. PCT-Net analyzes the new background's lighting and color characteristics, then applies pixel-wise color transformations to the foreground subject.

This process adjusts the subject's luminance, color temperatures, saturation with the new background, producing a photorealistic composite.

### 10.2.3. Output Generation

The harmonized image undergoes a final segmentation pass to extract updated layer metadata. The system returns the harmonized composite along with separated layers and metadata for continued interactive editing.

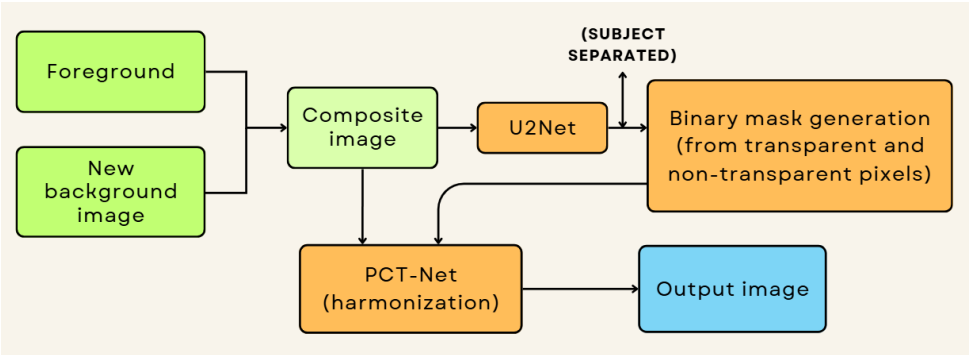


Figure 17: Feature 1: Background Replacement Pipeline

## 10.3. Feature 2: Auto Enhancement

This feature provides automated image quality improvement by intelligently analyzing the input image and applying only the necessary enhancement operations in an optimized sequence.

### 10.3.1. Overview

Traditional image enhancement pipelines apply a fixed sequence of operations regardless of the input image's characteristics, often resulting in over-processing or inefficient computation. Our system employs an **AI-driven adaptive enhancement pipeline** that dynamically selects and orders enhancement operations based on the specific degradations present in each image.

### 10.3.2. Enhancement Models

The system integrates four specialized deep learning models, each targeting a different feature:

1. **Low-Light Enhancement**: Improves visibility and color accuracy in underexposed images by adjusting luminance and correcting color cast introduced by insufficient lighting.
2. **Face Restoration**: Enhances facial features in degraded images, recovering fine details such as skin texture, eyes, and facial structure using generative restoration techniques.
3. **Denoising**: Removes sensor noise, grain, and other stochastic artifacts commonly present in images captured at high ISO settings or in low-light conditions.
4. **Deblurring**: Corrects motion blur and out-of-focus blur by estimating and reversing the blur kernel, restoring image sharpness and edge definition.

### 10.3.3. Pipeline Architecture

**Layer Composition** When the user initiates auto enhancement, all layers currently on the canvas are merged into a single composite image. This ensures that any prior edits are preserved and the enhancement is applied to the complete visual output.

**Image Analysis** The composite image is sent to the **Gemini API** for comprehensive quality analysis. Gemini evaluates the image and suggests the order in which we should apply corrections to enhance the image:

- A list of detected features that should be applied for enhancement
- The optimal sequence in which enhancements should be applied

The sequencing is critical—applying operations in an incorrect order can amplify artifacts or reduce effectiveness. For instance, denoising before low-light enhancement prevents noise amplification during brightness adjustment, while face restoration typically yields better results when applied after global corrections.

**Selective Model Invocation** Based on Gemini's analysis, the backend constructs a dynamic processing pipeline, invoking *only* the models required for the specific image. This selective approach offers two key advantages:

- **Computational Efficiency**: Unnecessary model inference is avoided, reducing processing time and server load.
- **Quality Preservation**: Images are not subjected to redundant processing that could introduce artifacts or degrade quality.

#### 10.3.4. Processing Flow

The complete auto enhancement pipeline can be summarized as follows:

1. Canvas layers are merged into a single composite image.
2. The composite is sent to Gemini for analysis.
3. Gemini returns the required enhancements and their optimal execution order.
4. The backend sequentially invokes only the necessary enhancement models.
5. The enhanced image is returned to the frontend and decomposed back into editable layers.

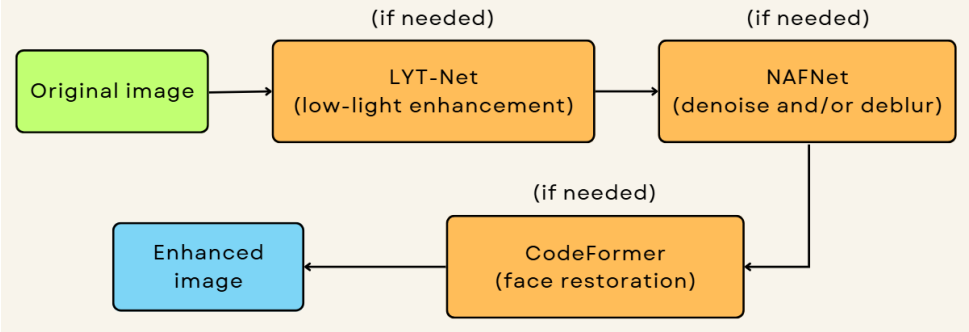


Figure 18: Feature 2: Auto Enhancement Pipeline

## 11. Technical Execution: Workspace 2 (LabSpace)

**LabSpace** is a secondary workspace providing direct access to individual AI models for isolated testing and simplified single-operation edits.

### 11.1. Available Models

The workspace exposes seven models: Object Removal, Background Removal, Style Transfer, Deblurring, Denoising, Face Restoration, and Low-Light Enhancement.

### 11.2. Prompt-Driven Model Selection

LabSpace employs a natural language interface powered by **Gemini API**. Users upload an image and describe their prompt (e.g., "remove the blur", "enhance lighting"). Gemini analyzes the prompt to determine the appropriate model and any required parameters.

### 11.3. Model-Specific Behaviors

#### 11.3.1. Object Removal

When object removal is requested, Gemini instructs the frontend to enter **point selection mode**. The user clicks on the target object, and the coordinates are sent to SAM for mask generation, followed by LaMa for inpainting.

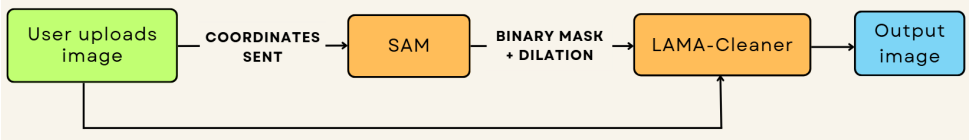


Figure 19: Feature 1: Object Removal

#### 11.3.2. Background Removal

Gemini classifies the subject as *human* or *object* and routes accordingly:

- **Human**: `u2net-human-seg` (optimized for human contours)
- **Object**: Standard `u2net` model

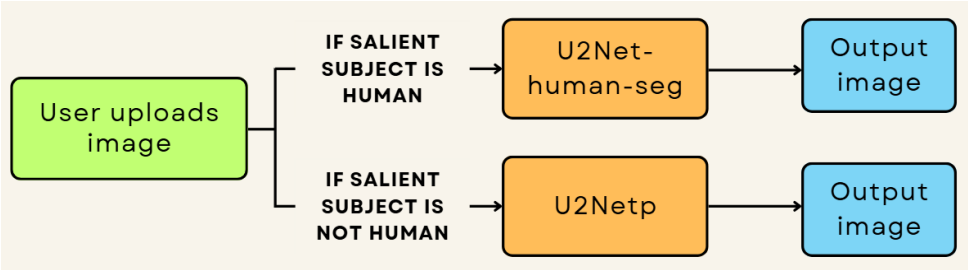


Figure 20: Feature 2: Background Removal

#### 11.3.3. Style Transfer

Style Transfer (PCA-based knowledge distillation (MobileNet)) takes the input image and reference image whose style is to transferred on the original input image.

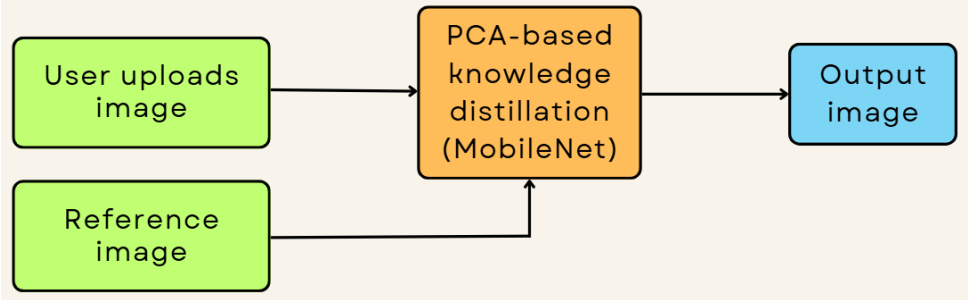


Figure 21: Feature 3: Style Transfer

#### 11.3.4. Face Restoration

Face restoration addresses degraded facial images suffering from blur, noise, compression artifacts, or low resolution. The system employs **CodeFormer**, a transformer-based blind face restoration model that leverages learned discrete codebook priors.

**Controllable Fidelity** CodeFormer provides a **fidelity weight parameter** ($w \in [0, 1]$) that controls the balance between quality enhancement and identity preservation. We have kept w=0.5 as default.


Figure 22: Feature 5: Face Restoration

### 11.3.5. Denoising and Deblurring

Image denoising and deblurring are handled by **NAFNet** (Nonlinear Activation Free Network), a state-of-the-art image restoration model that achieves competitive performance with a simplified architecture.

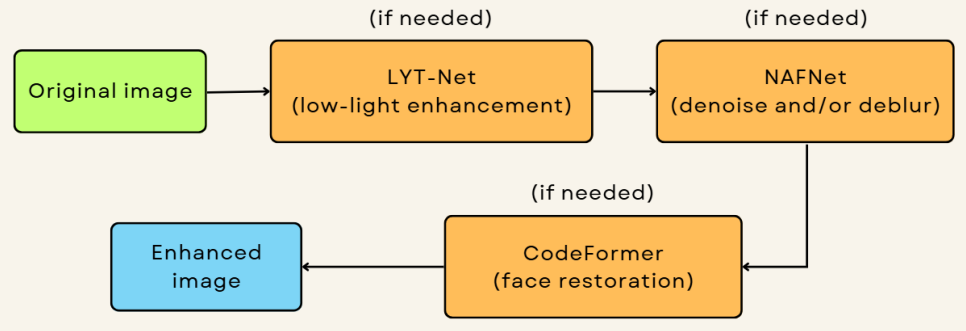


Figure 23: Feature 4: Denoise/Deblur

### 11.3.6. Low-Light Enhancement

Low-light enhancement addresses images captured in insufficient lighting conditions, suffering from underexposure, color distortion, and amplified noise. The system employs **LYT-Net** (Lightweight YUV Transformer Network), an efficient model designed specifically for low-light image enhancement.


Figure 24: Feature 6: Low-light Enhancement

## 12. Why Dilation?

This section demonstrates the importance of mask dilation in the object removal pipeline through visual examples.

Figure 25: 1. Original input image

Figure 26: 2. Binary Mask Generation without dilation

Figure 27: 3. Object Removal output (without dilation)

Figure 28: 4. Binary Mask Generation with dilation

Figure 29: 5. Object Removal output (with dilation)

## 13. Docker Containerization

In our system architecture, every deep learning model: U²-Net, U²-Net Human, SAM, NAFNet, CodeFormer, LYT-Net, LaMa-Cleaner, PCT-Net, and Style Transfer, is deployed in an independent Docker container. This design choice was crucial for ensuring stability, reproducibility, and scalability throughout development and deployment.

- **Dependency Isolation:** Each model in our project depends on a different set of libraries, PyTorch versions, CUDA toolkits, and system packages, like CodeFormer requires a specific PyTorch version to work properly, LaMa uses a distinct version of PyTorch Lightning, NAFNet requires a different CUDA setup.
  Running all of these in a single shared environment would cause library conflicts, version mismatches, and unstable behaviour.
  Docker solves this by providing isolated environments for each model; each container has its own python, pytorch version; its own CUDA and cuDNN dependencies; ensuring no library conflicts between models.
- **Independent Scaling per Model:** Different models in the system have different computational requirements and usage frequencies. With Docker, each model runs as a separate service.
- **Easy Deployment and Updates**
- **Fault Tolerance and Robustness:** Since each model runs in its own container as an isolated service. If one model crashes, the entire system does not crash.

## 14. System Infrastructure

The AI models employed in this system are computationally intensive, requiring dedicated GPU acceleration for real-time inference. The deployment infrastructure consists of a primary server for production workloads and a backup server for redundancy and load balancing.

### 14.1. Hardware Configuration

Table 2: Server Hardware Specifications

| Component | Primary Server | Backup Server |
|---|---|---|
| System | ASUS ROG Strix G16 | Acer Predator Helios Neo 16 |
| Processor | Intel Core i9-13980HX (24 Cores, 32 Threads) | Intel Core i7-14700HX (20 Cores, 28 Threads) |
| GPU | NVIDIA GeForce RTX 4060 (8 GB VRAM) | NVIDIA GeForce RTX 4050 (6 GB VRAM) |

## 15. Model Optimization

To reduce memory footprint and improve inference speed, we explored various optimization techniques across the deployed models. This section details the quantization attempts for CodeFormer and the successful ONNX deployment for segmentation models.

### 15.1. ONNX Deployment: U2Net Segmentation Models

The segmentation models (U2Net and U2Net-human-seg) are deployed using **ONNX** (Open Neural Network Exchange) format, enabling framework-agnostic inference with optimized runtime performance.

#### 15.1.1. Benefits of ONNX Conversion

Converting the PyTorch-based U2Net models to ONNX provides several advantages:

- **Runtime Optimization**: ONNX Runtime applies graph-level optimizations including operator fusion, constant folding, and memory planning.
- **Framework Independence**: Models can be deployed without PyTorch dependencies, reducing container size and startup time.
- **Cross-Platform Compatibility**: ONNX models can execute on various hardware backends (CPU, GPU, edge devices) with consistent behavior.

#### 15.1.2. Deployed Models

- **u2net.onnx**: General-purpose salient object detection for non-human subjects.
- **u2net-human-seg.onnx**: Human segmentation variant optimized for human contours, hair boundaries, and clothing edges.

Both models are served via ONNX Runtime with GPU execution provider, achieving real-time segmentation performance suitable for interactive editing workflows.

### 15.2. Quantization Analysis: CodeFormer

We investigated model quantization techniques for the CodeFormer face restoration model to further reduce memory footprint and improve inference speed. Two quantization approaches were evaluated: FP16 (half-precision floating point) and INT8 (8-bit integer quantization).

#### 15.2.1. FP16 Quantization

FP16 quantization reduces model weights from 32-bit to 16-bit floating point representation, theoretically halving memory usage while maintaining reasonable numerical precision. However, the CodeFormer model failed to convert to FP16 format.

CodeFormer employs a **Codebook Lookup Transformer** architecture that relies on discrete codebook indices and vector quantization operations. These components are sensitive to precision

reduction—the integer-indexed codebook retrieval and transformer attention layers produce large intermediate values incompatible with FP16's limited dynamic range. When FP16 conversion was attempted, the model produced numerical instabilities (NaN/Inf values) during inference, rendering the output unusable.

### 15.2.2. INT8 Quantization

INT8 dynamic quantization was successfully applied using ONNX Runtime's quantization toolkit. This approach quantizes model weights to 8-bit integers while computing activations in higher precision during inference.

### 15.2.3. Results

Table 3: INT8 Quantization Results for CodeFormer

| **Metric** | **Original (FP32)** | **INT8** | **Change** |
|---|---|---|---|
| Model Size | 359.5 MB | 91.2 MB | 74.6% reduction |
| Inference Time (CPU) | 12,925 ms | 16,863 ms | 30.5% slower |


Figure 30: Comparison of CodeFormer outputs: (a) Input image, (b) Original FP32 model output, (c) INT8 quantized model output, (d) Pixel-wise difference amplified 20×.

#### 15.2.4. Quality Analysis

Despite achieving 74.6% model size reduction, INT8 quantization introduced critical issues: (1) inference time increased by approximately 30% due to dynamic quantization overhead on CPU, (2) visible quality degradation in facial details, color accuracy, and edge sharpness (Figure 30d), and (3) quantization errors propagating through CodeFormer's precision-sensitive codebook lookup mechanism, resulting in incorrect codebook matches and degraded outputs.
Hence we retained the original FP32 model. The computational overhead is mitigated through GPU acceleration on our deployment infrastructure.

## 16. Compute Profile Analysis

This section provides a comprehensive comparison of all AI models deployed in the system, analyzing their computational requirements, inference performance, and efficiency characteristics.

Table 4: Model Tasks and Architecture Overview

| **Model** | **Task** | **Architecture** |
|---|---|---|
| CodeFormer | Face Restoration | Transformer + VQGAN |
| LaMa | Image Inpainting | FFT-based CNN |
| U²-Net | Salient Object Segmentation | Nested U-Net |
| SAM (ViT-B) | Promptable Segmentation | Vision Transformer |
| PCA-KD (PhotoWCT2) | Style Transfer | Distilled CNN |
| LYT-Net | Low-Light Enhancement | YUV Transformer |
| NAFNet-width64 | Image Denoise/Deblur | UNet + NAFBlocks |
| PCT-Net ViT | Image Harmonization | Vision Transformer |

Table 5: Model Size Comparison (Smallest to Largest)

| **Rank** | **Model** | **Parameters** | **Checkpoint Size** |
|---|---|---|---|
| 1 | LYT-Net | 44.92K | 0.20 MB |
| 2 | PCA-KD | 72.84K | 0.29 MB |
| 3 | U²-Net-lite | 1.13M | 4.7 MB |
| 4 | PCT-Net ViT | 4.81M | 18.39 MB |
| 5 | U²-Net | 44.0M | 167.8 MB |
| 6 | NAFNet | 67.89M | 258.98 MB |
| 7 | SAM (ViT-B) | 93.74M | 357 MB |
| 8 | CodeFormer | 94.11M | 376.64 MB |
| 9 | LaMa | 51.06M | 410.05 MB |

Table 6: Model Input/Output Specifications

| **Model** | **Input Resolution** | **Output Resolution** |
|---|---|---|
| CodeFormer | 512×512 (Fixed) | 512×512×3 |
| LaMa | Variable + Mask | Same as input |
| U²-Net | Variable (optimal 320×320) | Same as input (1 channel) |
| SAM (ViT-B) | Variable → 1024 | 3 masks @ input resolution |
| PCA-KD | Variable (divisible by 32) | Same as input |
| LYT-Net | Variable | Same as input |
| NAFNet | Variable | Same as input |
| PCT-Net | Variable (composite + mask) | Same as input |

Table 7: CPU Inference Performance Comparison

| **Model** | **256×256** | **512×512** | **Native Resolution** | **FPS** |
|---|---|---|---|---|
| PCA-KD | 65.67 ms | 95.58 ms | 65.67 ms @256 | **15.23** |
| PCT-Net | 126 ms | 356 ms | 659 ms @512 | 2.4 |
| U²-Net | ∼200 ms | ∼600 ms | ∼600 ms @320 | ∼1.5 |
| LYT-Net | 245.06 ms | 1,006.31 ms | 245.06 ms @256 | 4.08 |
| NAFNet | 2,013.88 ms | 8,975.93 ms | 2,013.88 ms @256 | 0.50 |
| LaMa | – | 2,437 ms | 2,437 ms @512 | 0.41 |
| CodeFormer | – | 4,761 ms | 4,761 ms @512 | 0.21 |
| SAM (ViT-B) | 9,009 ms | 9,009 ms | 9,009 ms @1024 | 0.11 |

Table 8: GPU Inference Latency

| **Model** | **Mean Latency** | **Hardware** |
|---|---|---|
| LYT-Net | 2.3 ms | – |
| U²-Net | 19.73 ms | – |
| PCT-Net | 25.68 ms | – |
| PCA-KD | 40 ms | GTX 1080 |
| SAM (ViT-B) | ∼50 ms | A100 |
| LaMa | ∼50–100 ms | – |
| CodeFormer | 70 ms | V100 |

Table 9: Computational Efficiency Comparison

| **Speed Rank** | **Model** | **FLOPs/GMACs** | **Notes** |
|---|---|---|---|
| 1 (Fastest) | PCA-KD | 0.72% of WCT2 | Highly distilled |
| 2 | LYT-Net | 3.49 GFLOPs | Lightweight transformer |
| 3 | U²-Net | ∼30 GFLOPs | – |
| 4 | NAFNet | 1.1–65 GMACs | Variable by resolution |
| 5 | LaMa | ∼50 GFLOPs | – |
| 5 | PCT-Net | ∼10 GFLOPs | – |
| 7 | CodeFormer | ∼100 GFLOPs | – |
| 8 (Slowest) | SAM (ViT-B) | ∼180 GFLOPs | – |

Table 10: Training Datasets and Model Variants

| Model | Training Dataset | Available Variants | Deployed Variant |
|---|---|---|---|
| CodeFormer | FFHQ | CodeFormer | Base |
| LaMa | Places2 | LaMa | Big-LaMa |
| U²-Net | DUTS-TR | U²-Net, U²-Net-lite | Full (44.0M) |
| SAM | SA-1B (11M images) | ViT-B, ViT-L, ViT-H | ViT-B (93.74M) |
| PCA-KD | COCO | MobileNet, VGG | MobileNet (72.84K) |
| LYT-Net | LOL | LYT | Base (44.92K) |
| NAFNet | GoPro/SIDD | width32, width64, SIDD | width64 (67.89M) |
| PCT-Net | iHarmony4 | ViT_pct, CNN_pct, sym, polynomial | ViT_pct (4.81M) |

Table 11: CPU Speed Rankings at 512×512 Resolution

| Rank | Model | Latency | FPS |
|---|---|---|---|
| 1 | PCA-KD | 95.58 ms | 10.46 |
| 2 | PCT-Net | 356 ms | 2.81 |
| 3 | U²-Net | ∼600 ms | ∼1.67 |
| 4 | LYT-Net | 1,006.31 ms | 0.99 |
| 5 | LaMa | 2,437 ms | 0.41 |
| 6 | CodeFormer | 4,761 ms | 0.21 |
| 7 | NAFNet | 8,975.93 ms | 0.11 |
| 8 | SAM (ViT-B) | 9,009 ms | 0.11 |

## 17. Models Evaluated and Discarded

Several models were evaluated during development but ultimately not selected due to technical constraints and time limitations.

- **FLUX.1-dev** (inpainting): Gated on HuggingFace; model weights inaccessible without approved access token.
- **MiDaS** (depth perception): Dependency conflicts with `opencv-python-headless` prevented stable installation.
- **ZoeDepth** (depth perception): Package conflicts between `easyocr` and OpenCV distributions caused installation failures.
- **GFPGAN** (face restoration): Excessive memory requirements for CPU execution; GPU-only dependencies incompatible with Windows; Colab runtime errors with output directories.
- **MonoDepth2** (depth perception): OpenCV version conflicts similar to other depth models.
- **Stefann** (scene text editing): Built on TensorFlow 1.x, incompatible with Python 3.12 and CUDA 12 environments. Would require legacy environment setup or model conversion.
- **Fast Reflection Removal** (reflection removal): Output quality insufficient for project requirements.

### 17.1. Common Issues

Recurring challenges included gated HuggingFace models, OpenCV dependency conflicts, GPU-only packages failing on CPU/Windows, and legacy TensorFlow incompatibility with modern runtimes.

## 18. Future Enhancements

While our current implementation demonstrates the viability of containerized AI models for image processing, several avenues remain for future development.

### 18.1. Model Optimization

- **Advanced Quantization:** Due to time constraints and limited access to high-end GPUs, we were unable to perform Quantization-Aware Training (QAT) across all models. Future work includes INT8/INT4 quantization with proper calibration, potentially reducing model sizes by 4-8x while maintaining accuracy.
- **Model Pruning:** Structured pruning techniques could further reduce computational costs, particularly for transformer-based models like SAM and CodeFormer.
- **Knowledge Distillation:** Training smaller student models from our larger pipelines could yield faster inference without significant quality degradation.

### 18.2. Deployment and Scalability

- **Cloud Deployment:** Deploying containerized models on AWS (EC2, Lambda, or SageMaker) with auto-scaling capabilities for production workloads.

### 18.3. Model Expansion

- **Additional Tasks:** Integrating models for super-resolution (Real-ESRGAN), video processing, and 3D-aware generation.
- **Fine-tuning with LoRA:** Applying Low-Rank Adaptation (LoRA) to customize models for domain-specific tasks with minimal compute overhead.
- **Multi-model Pipelines:** Chaining models (e.g., SAM $\rightarrow$ LaMa $\rightarrow$ CodeFormer) for complex editing workflows.

### 18.4. Mobile and On-Device Processing

By 2030, mobile devices are projected to have significantly enhanced NPU capabilities. Future iterations could leverage:

- On-device inference using CoreML (iOS) or NNAPI (Android)
- Lightweight model variants (U2-Net-lite, PCA-KD) already viable for mobile
- Progressive enhancement: cloud fallback for heavy models, on-device for lightweight ones

### 18.5. Flow Lab: Layer-Based Editing

Our vision for Flow Lab was to replicate Photoshop-style layer functionality using AI:

- **Current State:** Automatic foreground/background separation with operations (auto-enhancement, background replacement).
- **Future Goals:**
  - Multiple layer support with independent transformations
  - Non-destructive editing with layer history
  - Real-time layer compositing with harmonization (PCT-Net)
  - Mask refinement using SAM for precise layer boundaries

## 19. Conclusion

Arclight bridges the gap between casual and professional mobile photo editing through intuitive controls, assistive AI, and a dual workspace model.

## Appendix

### A. Figma Navigation Graph

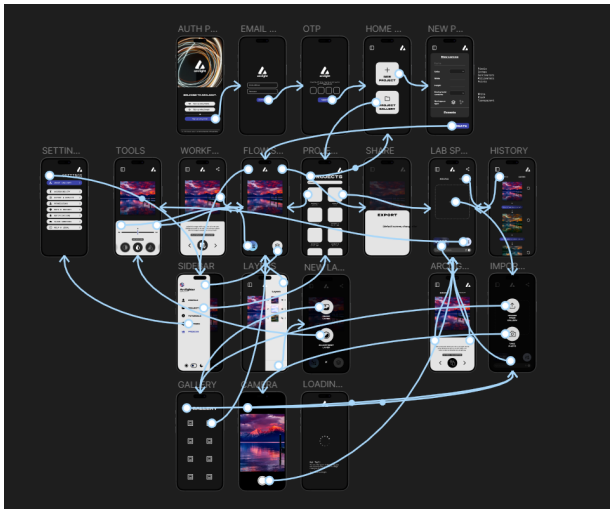


Figure 31: Navigation graph for buttons (using mid-fi screens)

### B. Survey Graphs (from 52 respondents)

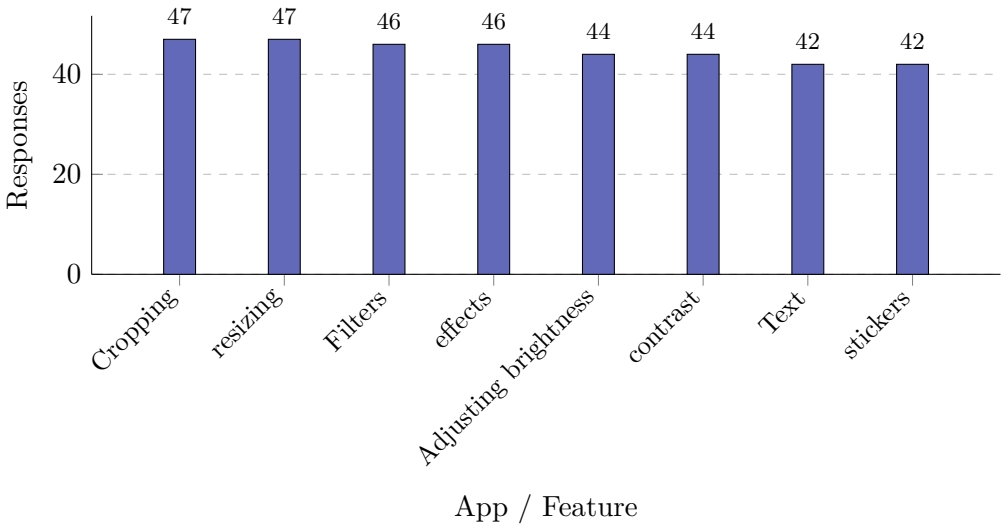


Figure 32: Apps and editing operations reported by respondents. Values are counts from the survey.

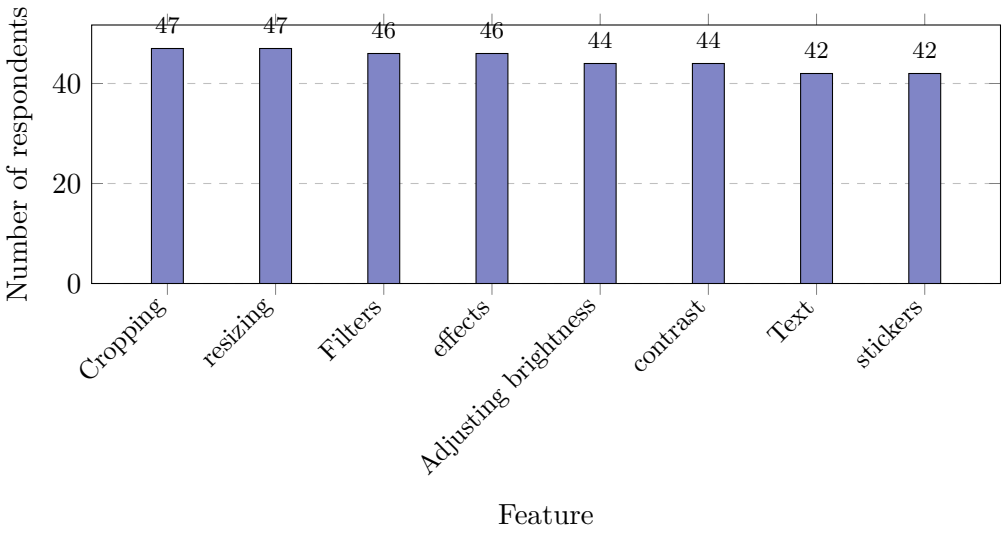


Figure 33: Features respondents most want AI to perform. Counts are taken directly from the survey responses.

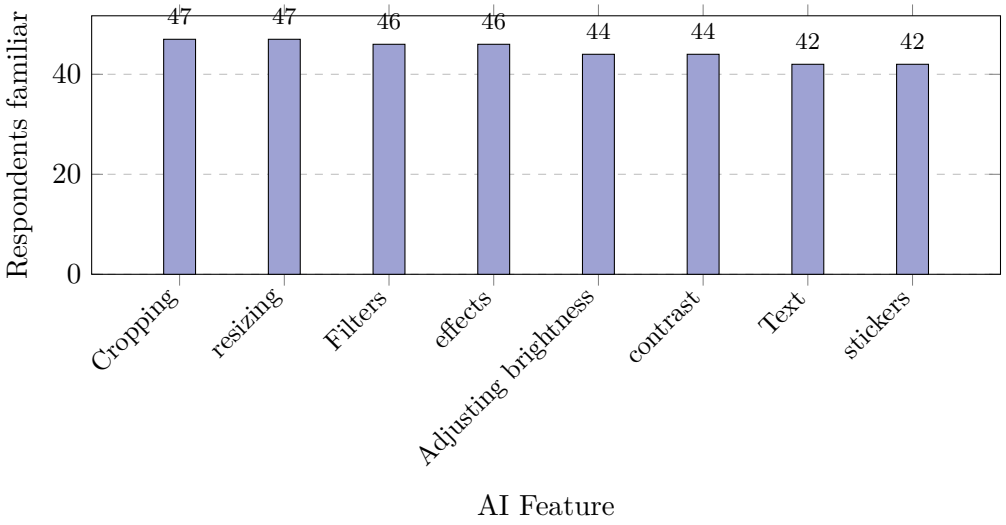


Figure 34: Familiarity with various editing features and AI assisted operations. Values correspond to counts from the survey.

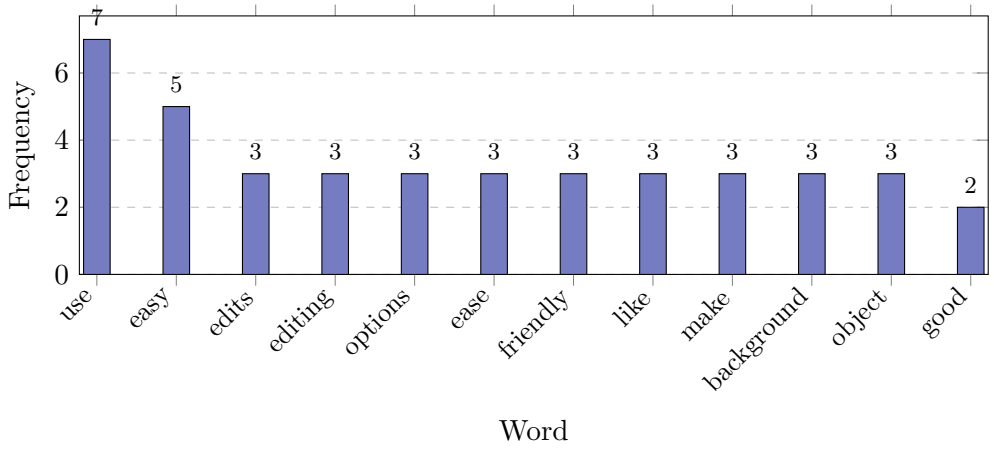


Figure 35: Top words from open text responses. This simple frequency chart highlights common themes such as ease and usability.

### C. Contrast Table

**Contrast Table: Dark Mode**

| Element | Foreground | Background | Ratio | WCAG |
|---|---|---|---|---|
| Primary text | #E4E4E4 | #1A1A1A | 13.45:1 | AA / AAA Pass |
| Secondary text | #BFBFBF | #1A1A1A | 8.21:1 | AA / AAA Pass |
| Icon default state | #E4E4E4 | #1A1A1A | 13.45:1 | AA / AAA Pass |
| Inactive icon state | #777777 | #1A1A1A | 4.6:1 | AA Pass |

Table 12: Contrast ratios for dark mode foreground and background combinations.

**Contrast Table: Light Mode**

| Element | Foreground | Background | Ratio | WCAG |
|---|---|---|---|---|
| Primary text | #222222 | #E8E5D8 | 12.1:1 | AA / AAA Pass |
| Secondary text | #555555 | #E8E5D8 | 7.2:1 | AA / AAA Pass |
| Icon default state | #222222 | #E8E5D8 | 12.1:1 | AA / AAA Pass |
| Inactive icon state | #888888 | #E8E5D8 | 4.3:1 | AA Pass |

Table 13: Contrast ratios for light mode foreground and background combinations.

**Contrast Table: Accent Color**

| Usage | Accent | Background | Ratio | WCAG |
|---|---|---|---|---|
| Accent on dark | #3C44A8 | #1A1A1A | 4.55:1 | AA Pass |
| Accent on light | #3C44A8 | #E8E5D8 | 6.8:1 | AA / AAA Pass |
| Accent for icon state | #3C44A8 | #FFFFFF | 5.1:1 | AA / AAA Pass |

Table 14: Contrast ratios involving Arclight's accent color in different contexts.

## References


[1] L. Chen, X. Chu, X. Zhang, and J. Sun, "Simple Baselines for Image Restoration," *arXiv preprint arXiv:2204.04676*, 2022.

[2] A. Brateanu, R. Balmez, A. Avram, C. Orhei, and C. Ancuti, "LYT-Net: Lightweight YUV Transformer-based Network for Low-light Image Enhancement," *IEEE Signal Processing Letters*, pp. 1–5, 2025, doi: 10.1109/LSP.2025.3563125.

[3] X. Qin, Z. Zhang, C. Huang, M. Dehghan, O. Zaiane, and M. Jagersand, "U2-Net: Going Deeper with Nested U-Structure for Salient Object Detection," *Pattern Recognition*, vol. 106, p. 107404, 2020.

[4] S. Zhou, K. C.K. Chan, C. Li, and C. C. Loy, "Towards Robust Blind Face Restoration with Codebook Lookup Transformer," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[5] X. Wang, L. Xie, K. Yu, K. C.K. Chan, C. C. Loy, and C. Dong, "BasicSR: Open Source Image and Video Restoration Toolbox," 2022. [Online]. Available: https://github.com/XPixelGroup/BasicSR

[6] R. Suvorov, E. Logacheva, A. Mashikhin, A. Remizova, A. Ashukha, A. Silvestrov, N. Kong, H. Goka, K. Park, and V. Lempitsky, "Resolution-robust Large Mask Inpainting with Fourier Convolutions," *arXiv preprint arXiv:2109.07161*, 2021.

[7] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment Anything," *arXiv preprint arXiv:2304.02643*, 2023.

[8] T.-Y. Chiu and D. Gurari, "PCA-Based Knowledge Distillation Towards Lightweight and Content-Style Balanced Photorealistic Style Transfer Models," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2022, pp. 7844–7853.

[9] J. J. A. Guerreiro, M. Nakazawa, and B. Stenger, "PCT-Net: Full Resolution Image Harmonization Using Pixel-Wise Color Transformations," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2023, pp. 5917–5926.