CONTROL PROJECT - COMPLETE TECHNICAL DOCUMENTATION

Comprehensive specification for desktop application and website development

TABLE OF CONTENTS

1. Desktop Application - App
2. Desktop Application - Architecture
3. Request Management & Token Counting
4. Chat Interface Design
5. Settings Modal
6. System Prompting & Process Flows
7. Process Management
8. Features & Design
9. Platform Support
10. User Interface Descriptions
11. Notifications & Toasts
12. Firebase Integration
13. Implementation Notes
14. Website Documentation - Overview
15. Website Structure & Pages
16. Website Design & Theme
17. Functional Integrations

DESKTOP APPLICATION - APP

Entry Point

At the entry point of the app the user is prompted to log in using Google authentication. This opens the user's default browser and displays the Google auth page which will check the database for their plan and associated API key (based on plan) after authentication. The entry modal displays a "Sign in with Google" button that opens the user's default browser and initiates the Google Auth sign-in page with Firebase config.

Once authenticated, the user's profile picture, name, and email are displayed on the entry page. The AI stores user information (plan) so the user doesn't need to log in again when the app opens.

Window Properties

- Close button HIDES the window (does not close the process)
- Window is draggable by the title bar
- Minimize and maximize buttons present
- Entry modal is the only window with a solid white background and black text (NOT transparent)

API Key Configuration

If free plan: a specific Gemini API-key will be used (from .ENV file)

If Pro plan: a different Gemini API-key will be used (from .ENV file)

If Master plan: a different Gemini API-key will be used (from .ENV file)

The Gemini API-Keys (.free_key, .pro_key, .master_key) are stored in a .ENV file in the project root not a backend folder to ensure the Python backend recognizes them throughout the code.

Main Process

The entry point of the app can be closed or minimised but should not close the main process. The main process allows the user to use hotkey to access the chat window and other windows. The app should start automatically on system boot.

Floating Circular Button

Once the app is fully initialized (and as long as the main process is still running), a round circular button with the app icon appears (make this as professional as possible). When clicked, it opens the chat window. The button disappears when the chat window is hidden using hotkeys (the chat window is not to have close minimize or maximize buttons). The chat window can still be opened and hidden using hotkeys for easy accessibility. The button sticks to and is draggable around the edges of the user screen.

DESKTOP APPLICATION - ARCHITECTURE

Repositories to Include

The Electron app is a combination of features from several repositories to create one unified project:

- Opacluely: https://GitHub.com/TechyCSR/OpenCluely.git
- Free-cluely: https://GitHub.com/Prat011/free-cluely.git
- Computeruse: https://GitHub.com/Greatness0123/computeruse.git

These should be cloned for easy accessibility and code reference.

Feature Details to Imitate Code

Free-cluely Features

- Audio input (speech to text) without prompts
- Screenshot to AI (via api) function
- Chat to AI (via api) function
- Voice input which transcribes to the input (if implementation is certain)

Opencluely Features

- Windows invisibility (allows the app to be invisible to Screenshot, Screen record and screenshare)
- Hotkey code (the code that allows the window to open and hide via hotkey)

- Toggle interaction function (the code that allows the user to interact with windows beyond the app once a hotkey is clicked or activated)
- MARKDOWN Function

Computeruse Features

- Computer use code (the codes that allow the Gemini AI via API to make use of the user keyboard, mouse, terminal and see the user screen using screenshot)
- Python-based backend that runs concurrently when npm start is executed
- The codes will be optimized and properly connected to the main application
- Uses Python libraries and necessary promptings

REQUEST MANAGEMENT & TOKEN COUNTING

Request counters and token counters should be added to track user activity:

Request Limits (per day)

- Free users: 5 requests per day
- Pro users: 50 requests per day
- Master plan users: 100 requests per day

Token Counting

- Each user request should be stored in the Firebase database alongside the amount of tokens used
- Context memory is included so the AI remembers past conversations for each user
- All this information should be uploaded to the user's document constantly as long as the user is online

Disabling Input

Once the user exceeds their daily limit (based on their plan), the chat input should be disabled. The user must wait 24 hours before making another set of requests based on their plan.

Add proper error handling with professional, well-agreed text to inform users of errors such as:

- "You have no more requests for today"
- "Invalid authentication"
- "No internet connection"
- "Session expired"

CHAT INTERFACE DESIGN

Design

The chat-window is to be full screen with an invisible background with two sections (the top (message display) and the bottom part of the same width as the top the input section). The parts without sections should be clickable and accessible by the user. In contradiction to the previous design the sections are to have a white background which should turn invisible when alt+a is pressed and back to white when it is pressed back so technically it becomes transparent so the users can sorta see what they are doing in the application behind.

Message Display

- User messages appear in bubbles on the right side
- AI responses appear without bubbles on the left side
- AI should display proper action logging showing what each action is performing
- Include a loader to show the AI is running an action
- Display a checkmark when a task is completed
- Display an X when a task fails

Chat Display and Settings Interaction

When the settings button is pressed:

- The chat display should shift a bit to the left to make room for the settings panel
- The settings and chat display should be the same height and displayed side by side above the chat input
- When settings are closed, the chat display should shift back to its original position

SETTINGS MODAL

The settings modal should include the following features:

Hotkey Configuration

- A section to change the hotkey used to toggle chat window visibility and interaction mode
- Recording functionality: pressing an input field records the keys users press to set custom hotkeys
- Interaction indicator: RED when users can access the apps behind Control, GREEN when they can't (non-interactive)

Theme Toggle

- Light mode
- Dark mode
- System default
- Super mode (galaxy purple gradient themed with color accents)

Reset Settings

Puts all settings back to default values

Quit App

Closes the entire app process (main process) and all it entails till it is opened again

Dismissal

The settings modal should be closable if the user taps anywhere outside the modal

Start up on Power

A toggle that allows the user to choose whether the main process should start up on the device powering on or not

Default Hotkeys

- Open entry modal: Click on the app icon in desktop
- Open/close chat window: Cmd/Ctrl + Space (by default)
- Toggle interaction: Alt + A (by default)

SYSTEM PROMPTING & PROCESS FLOWS

The AI should receive proper and extensive system prompting (if Gemini allows) so it knows what format to send its messages in. The program receiving the AI message will know how to interpret it. The AI should send its responses with the following parts:

- Action Log
- Terminal code
- Keyboard & mouse (PyAutoGUI) code
- Timing (interval between running each code)
- Screenshot

The AI should be able to use the user device properly using the user OS terminal and keyboard and mouse using screenshots for visuals and also perfect prompting

The AI should differentiate when the user is asking a simple question versus when they're asking it to perform a task on their device.

For Normal Questions - The Process

Step 1: User Input

User types or speaks their question into the chat interface

Step 2: Thinks

AI processes the question and formulates a response (AI thinks)

Step 3: End Message

AI sends its reply as the end message to the user

For Action Processes - The Complete Workflow

Step 1: User Input

User requests an action to be performed on their device

Step 2: Planning

AI breaks the tasks into simpler processes and plans the execution sequence

Step 3: Acting (ACTIONS)

AI sends Action Log, terminal codes/scripts, and timing instructions. The program automatically executes these actions with the AI running in the background. The AI should always request screenshots at the end of every action process sent.

Step 4: Verification

Using screenshots sent (from the screenshot request at the end of the action process command) to verify if the process is completed.

Step 5a: ACTION COMPLETED

If action is successful: Send an end message without screenshot request

Step 5b: ACTION NOT COMPLETED

If action fails: Go back to planning to use an alternative method that might work

Note: This should be done for every simpler process as planned by the AI until the major task is completed. Speed and accuracy are paramount.

PROCESS MANAGEMENT

AI processes should not stop even when the windows or modals are hidden.

FEATURES & DESIGN

Application Concept

The entire application is called "Control" - a Computer use agent which can also function as a regular AI to answer questions.

Visual Style

- Color pallet layout is strictly black white and grey except super theme is toggled
- Transparent windows and modal divs with white backgrounds which turn transparent when interaction is toggled via hotkey
- Audio input available
- Chat input functionality
- Professional & modern look with production standard icons not emojis
- Hotkeys for quick access
- App theme is mainly white, black and grey (except Super mode which includes galaxy purple gradient with color accents)

Application Features

- Auto-start on system boot (by default but can be put off in settings)
- Floating circular button for easy access to chat window
- Draggable around screen edges
- Request and token counting
- Firebase real-time database sync
- Conversation history and context memory
- Multi-user support with separate data storage

PLATFORM SUPPORT

The entire app should be compilable to .exe and .dmg and should use Electron for better UI compatibility on both platforms.

Visual Design

- Stylish icon and animations throughout the app
- Interaction topic affects all modals and the entire app
- Glassmorphism effects with subtle blur and shadow
- Smooth transitions and hover effects

USER INTERFACE DESCRIPTIONS

1. Desktop View - Entry Modal (Initial Launch)

Context

This is what the user sees when they first launch the application from their desktop.

Layout Description

- Background: The user's regular desktop with all their other application icons visible on the left side
- Central Modal Window: A centered, non-transparent modal window with a white background (this is the ONLY window that is not transparent)
- Modal Header: At the top of the modal, there is text reading "Welcome" followed by "to Control" on the next line, both centered
- User Info Display (After Authentication): Once signed in, displays user profile picture, name, and email
- Google Sign-In Button: A prominent button for Google authentication
- Modal Appearance: This modal should have clean, professional styling with a white background and black text, contrasting with all other windows in the app which will be transparent with 40% opacity
- Window Controls: Minimize, maximize, and hide (close) buttons on the title bar. Title bar is draggable.
- Purpose: This is the authentication/entry point where users log in with Google before accessing the main application features

2. Active State - Main Chat Interface (When Hotkey is Activated)

Context

This view appears when the user presses Cmd/Ctrl + Space while the main process is running.

Layout Description

- Background: Full screen with transparent background allowing the user to see their desktop through it and also interact through it
- Chat Display Area (adjustable for settings):
  - White background
  - User messages appear in bubbles on the right side
  - AI responses appear without bubbles on the left side
  - AI action logs are displayed showing what the AI is performing
  - Shifts left when settings panel is open to accommodate side-by-side display
  - This entire chat area has a "half blur" (subtle blur) effect at the bottom creating a frosted glass appearance
- Top Right Corner of chat display area:
  - Settings button for accessing configuration options
  - Interaction indicator dot (green when interactive/can't access desktop, red when non-interactive/can access desktop)
- Bottom Section - Chat Input Area:
  - Text input field for typing messages
  - Send/Stop button to submit text or stop actions (should be turned to stop while the ai is performing an action)
  - Audio input button for voice commands (this should have an indicator on the same button (microphone icon turns to box) to show that it is listening and then real time transcription to the input field and then when clicked again stops the live transcription ... I think there's a javascript library that can handle this.... just make it work)
  - "New conversation" button to start a fresh chat
  - This area also has to be transparent when the interaction is toggled
- Other Desktop App Icons: Still visible on the left side of the screen through the transparent interface

3. Floating Circular Button

Context

Appears once the app is fully initialized and chat window is hidden.

Features

- Round circular button with app icon
- Draggable around the edges of the user screen
- Sticks to screen edges for easy access
- Clicking opens the main chat interface
- Disappears when chat window is active
- Should reappear when chat window is hidden

4. Settings Modal View

Context

This view appears when the user clicks the settings button in the top right corner of the chat interface.

Layout Description

- Central - Chat Display: Remains fully visible and functional, shifted to the left to make room
- Right Side - Settings Panel:
  - A white background modal panel appears on the right side of the screen (no sliding animation needed, can be instant)
  - The panel header displays "Settings" text at the top
  - Below are multiple rectangular sections/buttons for different settings options
  - The panel should NOT overlap or interfere with the chat message display area
  - The panel appears to the right and is separate from the main chat area
  - Same height as the chat display area
  - Closable by clicking outside the modal
- Settings Options Include:
  - Hotkey Customization - Recording inputs for toggle interaction and chat open/hide hotkeys
  - Interaction Indicator Display - Shows current status
  - Theme Toggle - Light, Dark, System, or Super mode
  - Auto start toggle - whether the main process starts up on device power on

    - Reset Settings - Returns all settings to default
    - Quit App - Closes the entire application process
    - Update Checker - Check for app updates
- Behavior: The settings modal should be easily dismissible and should not interfere with ongoing AI processes or chat functionality. Its background should also turn transparent on interaction toggle via hotkey

NOTIFICATIONS & TOASTS

Customized toasts should be displayed for:

- Setting updates confirmation
- No internet connection (disables chat input)
- Unauthenticated user attempts
- Request limit exceeded
- Successful operations

FIREBASE INTEGRATION

Database Structure

Each user data is stored under their user ID for easier fetching:

```
users/
  {userId}/
    userInfo/
      - email
      - name
      - profilePicture
      - plan (free/pro/master)
      - createdAt
    conversations/
      - {conversationId}
        - messages[]
        - createdAt
        - updatedAt
    usage/
      - totalTokensUsed
      - requestsToday
```

- requestsMadeAllTime
- lastResetDate
- dailyStats[]

Real-time Updates

All user information should be uploaded to Firebase constantly as long as the user is online, including:

- Token usage
- Request counts
- Conversation history
- User activity

IMPLEMENTATION NOTES - DESKTOP APPLICATION

Critical Technical Details

Main.js Development

- Pay close attention to main.js configuration as this gave issues previously
- Proper window management for minimize, maximize, and hide functionality (in the entry modal and via hotkey not button for the chat window)
- Correct hotkey registration and handling
- Proper process management to keep main process running when entry modal is closed

Python Backend

- Python script runs concurrently when npm start is executed
- Uses .ENV file for API keys (three separate keys for free/pro/master plans)
- Handles computer use actions (keyboard, mouse, terminal, screenshots)
- Properly isolated from Electron UI process
- Test with provided Gemini API key: AIzaSyD82tjQB9v7QIOR4xo-CZCIOBNPwxoERVQ to ensure correct response format

Node Modules & Dependencies

- Install all required node modules
- Install all Python dependencies
- Ensure no errors during installation; fix any issues before proceeding
- Test application thoroughly before compilation

Compilation

- Compile to .exe for Windows
- Compile to .dmg for macOS
- Both should compile without errors
- Fix any compilation issues before finalizing

Cleanup

After successful compilation and testing:

- Delete all cloned repositories
- Delete compiled .exe and .dmg files
- Keep only clean source code

Documentation

Create comprehensive documentation including:

- Every file and folder in the project
- What each component contributes to the application
- Where to go to make future UI changes
- How to troubleshoot common issues
- Detailed UI modification guide

Important Implementation Notes

- Hotkeys must be fully customizable through the settings interface
- The interaction indicator (green/red dot) should be visible at all times when the app is active
- Request counting and Firebase integration is critical for user plan management (5 requests for free, 50 for pro, 100 for master)
- AI should be able to clearly distinguish between simple queries and action requests (prompt well)
- The workflow must include proper verification steps with screenshots after each action
- Speed and accuracy are paramount in AI action execution
- AI processes must continue running even when windows are hidden
- Use Markdown for proper formatting of user inputs and AI responses
- Authentication is exclusively through Google Sign-In (no manual key input)
- Entry modal hides window instead of closing it
- Window is draggable by title bar with minimize/maximize controls
- Floating button sticks to screen edges and is draggable
- App auto-starts on system boot (adjustable via settings)

WEBSITE DOCUMENTATION

Website Overview

You are to build a fully functional, modern website for the Control desktop app using React + Tailwind CSS + Firebase + Flutterwave.

Purpose

The website is where users learn about the app, collect their license key, view their dashboard and usage stats, manage subscriptions, and download the desktop app.

Design Aesthetic

Futuristic, trustworthy, and professional with slight glassmorphism effects, motion, animations, and clean dark styling. Reference cluely.com for design inspiration.

WEBSITE STRUCTURE & PAGES

1. MAIN PAGE (PUBLIC)

Purpose

Attract new users and explain the product before login.

Hero Section

- Left Side:
  - Headline: "Control — Run any task on any device. Fast. Invisible. Accurate."
  - Include psychological and customer-attracting tagline with wordplay using "control"
  - Secondary CTA: "Download CONTROL"
- Right Side:
  - Interactive mock animation showing Control performing tasks on a user desktop

    - Examples: Control using Blender, VS Code, or other well-known applications
    - Can loop through several application usage scenarios
- Background:
    - Faint animated constellation-like particle background
    - Subtle glassmorphism effects with high contrast
- Visual Effects: Smooth animations and transitions

Feature Section (Interactive Cards)

Using Framer Motion for animations, include:

- Execute any task with one prompt
- Stealth background execution
- High speed and precision
- Sleek, intuitive interface
- Secure and encrypted
- (Think of and add other compelling features)

Each feature card should have:

- Perfect design and visual hierarchy
- Smooth hover animations
- Icon or visual representation
- Clear description

Why Use Control Section

Persuasive 3-4 paragraph explanation covering:

- Saves time by automating repetitive work
- Works on all devices with natural language commands
- Designed for both individuals and power users
- (Add other compelling reasons)

Contact + Quick Links (Footer)

- Short contact form (name, email, message)
- On submission: send message to developer's Gmail (grucookorie@gmail.com) via Firebase Cloud Function

- Quick links to other pages
- Social media links (if applicable)

2. PRICING PAGE

Plans & Pricing

- Free: $0 (For test use or simple tasks like writing essays)
- Pro: $25/month or $250/year (For bulk use, programming, and varied usage patterns)
- Master: $75/month or $750/year (For video editors, graphic designers, animators, and heavy users)

UI & Behavior

- Horizontal pricing cards layout
- Toggle for Monthly/Annual pricing that updates displayed prices dynamically
- Highlight Pro plan as "Most Popular"
- Interactive comparison table at the bottom showing which features each plan includes
- When "Get Plan" is clicked: direct to Flutterwave Checkout
- Both One-time and Subscription payment options (subscription is default/focused)

Pricing Comparison Table

Show which features are included/excluded for each plan:

- Number of daily requests
- Priority support
- Advanced features
- API access
- Custom integrations (for higher tiers)

3. USER DASHBOARD (AUTHENTICATED)

Tech Stack

Firebase Auth + Firestore

User Features

- Auto sign-in persistence using Firebase Auth session
- Free users can make 3 requests per day
- Pro users can make up to 100 requests per day
- Master users can make unlimited requests
- Display current request count and remaining requests for the day

Main Dashboard View

- Overview of key metrics at the top
- Quick access buttons to different sections

Analytics Section

- Fetch user usage data from Firestore dynamically
- Display in a line or area chart using Recharts
- No static mock data: always fetch real data
- Filter buttons for Day / Week / Month
- Show metrics like:
  - Requests made
  - Tokens consumed
  - Average response time
  - Most used features

Billing Tab

- Show current plan
- Next billing date
- Link to manage Flutterwave subscription
- Option to upgrade or downgrade plan

Settings Tab

- Basic profile info (email, profile photo from Google, current plan)
- Option to update profile information
- Manage connected devices

Onboarding (First Login Only)

- Show onboarding cards explaining dashboard components
- Cards and explained components should be in focus
- Rest of dashboard remains in background with dimmed effect
- Option to skip or close onboarding
- Appears only on first login

4. DOWNLOAD PAGE

Layout

- Three OS cards: macOS, Windows, Linux
- Each card displays:
  - OS logo
  - Download button
  - File size and version number

Installation Guide (Below Cards)

1. Download installer for your OS
2. Open the app
3. Get authenticated
4. Get absolute control

FAQ & Troubleshooting Section

- Common installation issues and solutions
- System requirements
- Supported operating systems and versions

Download Tracking

- Counter that tracks download button clicks for each OS
- Data displayed in Admin Dashboard (not visible to regular users)
- Used for analytics and marketing insights

5. LOGIN / SIGN-UP PAGE

Layout

Single page with toggle/slider for switching between Login and Sign-up

Sign-up Form

- Email input
- Password input
- Confirm password input
- Google Sign-in button
- "Already have an account?" link to switch to login

Login Form

- Email input
- Password input
- "Remember me" checkbox
- "Forgot password?" link
- Google Sign-in button
- "Don't have an account?" link to switch to sign-up

Design

- Minimal, dark, professional styling
- Clean form layout
- Clear error messages
- Firebase Auth persistence for seamless re-login

Password Reset

- "Forgot password?" sends password reset email via Firebase
- User can set new password via email link

6. ADMIN DASHBOARD (PIN PROTECTED)

Access

- URL: /admin
- Before viewing: PIN popup appears requesting "CTRL-1234" (default)
- After 3 incorrect PIN attempts: Redirect to homepage
- No Firebase authentication required: PIN security only

PIN Management

- Section to edit the access PIN
- Button to revert to default PIN
- Confirmation before changing PIN

Dashboard Overview

Chart at Top

- Cumulative average of all user usage stats
- Shows overall platform usage trends
- Line chart or area chart visualization

User Management

- List of all users from Firebase
- Each user displayed in a div/card
- Click on user: Navigate to detailed user info page
- Detailed user page displays:
  - All user information as stored in Firebase
  - Option to update user information
  - Option to delete user
  - Option to block/unblock user access
  - Save changes to Firebase

User Blocking

- Admin can block/unblock user access to the application
- Blocked users cannot authenticate or use the app

Statistics Cards (Bottom of Page)

- Number of downloads (counted from Download Page)
- Total number of users
- Number of blocked users
- Number of subscribed users per plan:
  - Free users count
  - Pro users count
  - Master users count

Additional Admin Features

- Search functionality to find users

- Filter users by plan type or status
- Export user data (optional)
- View system logs and activity

WEBSITE DESIGN & THEME

Framework

React + TailwindCSS + Framer Motion

UI Components

shadcn/ui or custom Tailwind cards & buttons

Color Palette

- Primary: Black and White
- Accent: Grey
- Frosted surfaces: rgba(255, 255, 255, 0.1) with backdrop-filter: blur(12px)

Typography

Inter or Poppins

Style Guidelines

- Clean, futuristic, slightly glassy aesthetic
- Responsive design for all screen sizes
- Extensive animations and smooth transitions
- High contrast for readability
- Consistent spacing and alignment
- Modern UI patterns and micro-interactions

FUNCTIONAL INTEGRATIONS

Firebase

- Authentication (email/password + Google)
- Firestore (user data, usage stats, plans, subscriptions)
- Cloud Functions (for sending contact messages to Gmail)
- Real-time database updates

Flutterwave

- Checkout integration for one-time payments
- Subscription plans API for recurring billing
- Both one-time and subscription options in payment flow
- Subscription default/focused in checkout
- Webhook handling for payment confirmations

Charts & Data Visualization

- Recharts for usage statistics and analytics
- Line charts, area charts, bar charts as needed
- Real-time data updates

Routing

- React Router for page navigation
- Nested routes for dashboard sections
- Protected routes for authenticated pages
- Admin route with PIN protection

ADDITIONAL PAGES

Terms of Service

Placeholder page with standard T&Cs

Privacy Policy

Placeholder page with standard privacy policy

About Page

- Information about Control app
- Team/creator information
- Contact information

IMPLEMENTATION NOTES - WEBSITE

General Guidelines

- Keyboard shortcuts (Ctrl + Space, etc.) apply only to the desktop app, not the website
- Focus on smooth UX, fast load times, and trustworthy design
- All data fetching should use real Firebase data, never static mock data for production
- Ensure responsive design works on mobile, tablet, and desktop
- Test all Firebase integrations thoroughly
- Test Flutterwave payment flow in sandbox mode before production
- Implement proper error handling and user feedback
- Optimize performance and load times
- Ensure accessibility standards are met

FINAL OUTPUT EXPECTATION

Complete responsive React + Tailwind website featuring:

- All required pages and functionality
- Working Firebase Auth/Firestore integration
- Dynamic data fetching for dashboard with real user data
- Functional Flutterwave payment flow

- Contact form that sends messages to Gmail via Firebase Cloud Functions
- Admin dashboard with PIN protection
- Download tracking and analytics
- Comprehensive onboarding experience
- Smooth animations and glassmorphism design
- Full responsiveness across all devices

PROJECT SUMMARY

Desktop Application - Control

A sophisticated Electron-based desktop application that combines computer use automation with AI assistance. The app uses Gemini API for AI capabilities and provides a seamless interface for users to control their devices through natural language commands. The application maintains a non-intrusive presence on the desktop with a floating button, supports custom hotkeys, manages requests based on subscription plans, and stores conversation history and usage statistics in Firebase.

Website

A modern, feature-rich React website built with Tailwind CSS that serves as the hub for the Control application. The website provides user authentication, a comprehensive dashboard for viewing usage statistics, pricing information, app downloads, subscription management through Flutterwave, and an admin dashboard for system management. All user data is stored and synchronized with Firebase for real-time updates.

Integration Points

- Google OAuth for seamless authentication across web and desktop
- Firebase for real-time database synchronization and authentication
- Flutterwave for secure payment processing and subscription management
- Gemini API for AI-powered computer automation and query response
- Cloud Functions for backend operations like sending contact messages

Key Features Summary

- Multi-platform desktop application (.exe and .dmg)
- Plan-based request limiting (5 free, 50 pro, 100 master)
- Real-time conversation history and context memory
- Computer automation via keyboard, mouse, and terminal commands
- Customizable hotkeys and themes
- Glassmorphism UI with professional animations
- User dashboard with analytics and usage statistics
- Admin panel with user management and download tracking
- Responsive design across all devices
- Secure authentication and encrypted data storage