```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>COGNITO AI</title>
    <!-- Tailwind CSS -->
    <script src="https://cdn.tailwindcss.com"></script>
    <!-- Marked.js for Markdown -->
    <script src="https://cdn.jsdelivr.net/npm/marked/marked.min.js"></script>
    <!-- Google Fonts -->
    <link
href="https://fonts.googleapis.com/css2?family=Montserrat:wght@400;500;600;700&display=sw
ap" rel="stylesheet">
    <!-- Font Awesome -->
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css">
    <script>
        tailwind.config = {
            darkMode: 'class',
            theme: {
                extend: {
                    colors: {
                        primary: '#5D5CDE',
                        dark: '#222222',
                        light: '#FFFFFF',
                        darkgray: '#333333',
                        lightgray: '#f0f0f0'
                    },
                    fontFamily: {
                        montserrat: ['Montserrat', 'sans-serif']
                    },
                    animation: {
                        'pulse-slow': 'pulse 3s cubic-bezier(0.4, 0, 0.6, 1) infinite',
                        'fade-in': 'fadeIn 0.5s ease-in',
                        'slide-in': 'slideIn 0.3s ease-out',
                        'slide-up': 'slideUp 0.3s ease-out',
                        'bounce-light': 'bounceLight 2s infinite'
                    },
                    keyframes: {
                        fadeIn: {
                            '0%': { opacity: '0' },
                            '100%': { opacity: '1' }
                        },
```

```
                slideIn: {
                    '0%': { transform: 'translateX(-10px)', opacity: '0' },
                    '100%': { transform: 'translateX(0)', opacity: '1' }
                },
                slideUp: {
                    '0%': { transform: 'translateY(10px)', opacity: '0' },
                    '100%': { transform: 'translateY(0)', opacity: '1' }
                },
                bounceLight: {
                    '0%, 100%': { transform: 'translateY(0)' },
                    '50%': { transform: 'translateY(-5px)' }
                }
            }
        }
    }
}
</script>
<style>
    body {
        font-family: 'Montserrat', sans-serif;
        transition: background-color 0.3s ease, color 0.3s ease;
    }

    .dark {
        background-color: #222222;
        color: #FFFFFF;
    }

    .light {
        background-color: #FFFFFF;
        color: #222222;
    }

    .logo-container {
        clip-path: polygon(0 0, 100% 0, 85% 100%, 0 100%);
    }

    .typing-indicator::after {
        content: '';
        animation: typing 1.5s infinite;
    }

    @keyframes typing {
        0% { content: '.'; }
```

```
    25% { content: '..'; }
    50% { content: '...'; }
    75% { content: '..'; }
    100% { content: '.'; }
}

.action-btn {
    transition: all 0.2s ease;
}

.action-btn:hover {
    transform: translateY(-2px);
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
}

.scale-hover:hover {
    transform: scale(1.05);
    transition: transform 0.2s ease;
}

.page {
    transition: opacity 0.5s ease, transform 0.5s ease;
}

.page.hidden {
    opacity: 0;
    transform: translateX(20px);
    pointer-events: none;
}

.chat-message {
    transition: all 0.3s ease;
    animation: fadeIn 0.5s ease;
}

@keyframes fadeIn {
    from { opacity: 0; transform: translateY(10px); }
    to { opacity: 1; transform: translateY(0); }
}

.accordion-content {
    max-height: 0;
    overflow: hidden;
    transition: max-height 0.3s ease;
```

```
}

.accordion-content.active {
    max-height: 500px;
}

.model-card {
    transition: all 0.3s ease;
}

.model-card:hover {
    transform: translateY(-5px);
    box-shadow: 0 10px 15px rgba(0, 0, 0, 0.1);
}

.top-nav-btn {
    position: relative;
}

.top-nav-btn::after {
    content: '';
    position: absolute;
    bottom: -2px;
    left: 50%;
    width: 0;
    height: 2px;
    background-color: #5D5CDE;
    transition: all 0.3s ease;
    transform: translateX(-50%);
}

.top-nav-btn:hover::after {
    width: 100%;
}

/* Custom scrollbar */
::-webkit-scrollbar {
    width: 6px;
}

::-webkit-scrollbar-track {
    background: rgba(255, 255, 255, 0.1);
}
```

```
::-webkit-scrollbar-thumb {
    background: rgba(93, 92, 222, 0.5);
    border-radius: 3px;
}

::-webkit-scrollbar-thumb:hover {
    background: rgba(93, 92, 222, 0.8);
}

/* Loader for API calls */
.loader {
    width: 20px;
    height: 20px;
    border: 3px solid rgba(93, 92, 222, 0.3);
    border-radius: 50%;
    border-top-color: #5D5CDE;
    animation: spin 1s ease-in-out infinite;
}

@keyframes spin {
    to { transform: rotate(360deg); }
}

/* Settings panel animation */
.settings-panel {
    transition: transform 0.3s ease, opacity 0.3s ease;
}

.settings-panel.hidden {
    transform: translateX(-100%);
    opacity: 0;
    pointer-events: none;
}

/* Switch UI Animation */
.switch-ui {
    transition: all 0.5s ease;
}

.switch-ui.active {
    transform: scale(1.05);
}

/* Copy button */
```

```
        .copy-btn {
            opacity: 0;
            transition: opacity 0.3s ease;
        }

        .chat-message:hover .copy-btn {
            opacity: 1;
        }

        /* Research mode */
        .research-mode {
            transition: all 0.3s ease;
        }

        .notification {
            position: fixed;
            top: 20px;
            right: 20px;
            padding: 10px 20px;
            background-color: #333;
            color: white;
            border-left: 4px solid #5D5CDE;
            border-radius: 4px;
            box-shadow: 0 4px 6px rgba(0,0,0,0.1);
            z-index: 1000;
            animation: slideInNotification 0.3s ease-out, fadeOutNotification 0.5s ease-out 5s
forwards;
            max-width: 300px;
        }

        @keyframes slideInNotification {
            from { transform: translateX(100%); }
            to { transform: translateX(0); }
        }

        @keyframes fadeOutNotification {
            from { opacity: 1; }
            to { opacity: 0; visibility: hidden; }
        }
    </style>
</head>
<body class="dark min-h-screen flex flex-col">
    <!-- Notification Container -->
    <div id="notification-container"></div>
```

```
    <!-- First Page - Welcome Screen -->
    <div id="welcome-page" class="page flex flex-col items-center justify-center min-h-screen
p-4">
        <div class="w-full max-w-md mx-auto flex flex-col items-center animate-fade-in">
            <div class="flex items-center mb-8">
                <h1 class="text-4xl font-bold mr-4">Good day, User!</h1>
                <div class="logo-container bg-primary p-2">
                    <img
src="https://assets.onecompiler.app/42u3pr9mz/43cjuxhs6/1742640682179.jpg"
alt="COGNITO" class="h-10">
                </div>
            </div>

            <div class="w-full mb-8">
                <label for="username" class="block text-sm font-medium mb-2">Enter your
name:</label>
                <input type="text" id="username" class="w-full p-3 rounded-lg bg-darkgray text-white
border border-gray-600 focus:outline-none focus:border-primary text-base" placeholder="Your
name">
            </div>

            <div class="grid grid-cols-2 gap-4 w-full mb-8">
                <button class="action-btn bg-darkgray p-4 rounded-lg flex items-center justify-center
animate-pulse-slow">
                    <i class="fas fa-lightbulb text-yellow-400 mr-2"></i>
                    <span>How to enjoy my weekend?</span>
                </button>
                <button class="action-btn bg-darkgray p-4 rounded-lg flex items-center justify-center
animate-pulse-slow">
                    <i class="fas fa-magic text-purple-400 mr-2"></i>
                    <span>Introduce yourself!</span>
                </button>
                <button class="action-btn bg-darkgray p-4 rounded-lg flex items-center justify-center
animate-pulse-slow">
                    <i class="fas fa-atom text-blue-400 mr-2"></i>
                    <span>Explain Quantum Physics</span>
                </button>
                <button class="action-btn bg-darkgray p-4 rounded-lg flex items-center justify-center
animate-pulse-slow">
                    <i class="fas fa-code text-green-400 mr-2"></i>
                    <span>Help me with my code!</span>
                </button>
            </div>
```

```
        <button id="next-btn" class="bg-primary hover:bg-opacity-90 text-white font-bold py-3
px-6 rounded-lg transition-all duration-300 flex items-center animate-bounce-light">
            Next
            <i class="fas fa-arrow-right ml-2"></i>
        </button>
    </div>
</div>

<!-- Second Page - Chat Interface -->
<div id="chat-page" class="page hidden min-h-screen flex flex-col">
    <!-- Top Navigation -->
    <div class="flex justify-between items-center p-4 border-b border-gray-700">
        <div class="flex items-center">
            <button id="info-btn" class="mr-4 text-gray-400 hover:text-white">
                <i class="fas fa-info-circle"></i>
            </button>
            <div class="flex items-center">
                <span class="text-sm mr-2">MODEL:</span>
                <span id="current-model" class="text-primary
font-semibold">COGNITO-command-4</span>
            </div>
        </div>
        <div class="flex items-center space-x-2 md:space-x-4">
            <button id="switch-ui-btn" class="top-nav-btn text-xs font-semibold text-gray-400
hover:text-white md:block hidden">
                <i class="fas fa-exchange-alt mr-1"></i> SWITCH UI
            </button>
            <button id="research-mode-btn" class="top-nav-btn text-xs font-semibold
text-gray-400 hover:text-white">
                <i class="fas fa-search mr-1"></i> RESEARCH MODE
            </button>
            <button class="top-nav-btn text-xs font-semibold text-gray-400 hover:text-white">
                <i class="fas fa-bell mr-1"></i> NOTIFICATIONS
            </button>
            <button class="top-nav-btn text-xs font-semibold text-gray-400 hover:text-white">
                <i class="fas fa-filter mr-1"></i> FILTER
            </button>
            <button id="save-chat-btn" class="top-nav-btn text-xs font-semibold text-gray-400
hover:text-white">
                <i class="fas fa-download mr-1"></i> SAVE CHAT
            </button>
        </div>
    </div>
```

```
    <!-- Main Content Area - Split into Chat and Settings -->
    <div class="flex flex-grow overflow-hidden">
      <!-- Settings Panel (initially hidden) -->
      <div id="settings-panel" class="settings-panel hidden w-full md:w-64 lg:w-80
bg-darkgray border-r border-gray-700 overflow-y-auto p-4">
        <h2 class="text-xl font-bold mb-4">Settings</h2>

        <!-- Models Section -->
        <div class="mb-6">
          <h3 class="text-lg font-semibold mb-3">Models</h3>
          <div class="space-y-3">
            <div class="model-card bg-dark p-3 rounded-lg cursor-pointer"
data-model="COGNITO-1">
              <div class="flex justify-between items-center">
                <h4 class="font-semibold">COGNITO-1</h4>
                <span class="text-xs text-gray-400">27B</span>
              </div>
              <p class="text-xs text-gray-400 mt-1">First language model trained on billions
of data with 34% being code</p>
              <div class="mt-2 text-xs text-primary">Context: 1,000,000 tokens</div>
            </div>

            <div class="model-card bg-dark p-3 rounded-lg cursor-pointer"
data-model="COGNITO-2">
              <div class="flex justify-between items-center">
                <h4 class="font-semibold">COGNITO-2</h4>
                <span class="text-xs text-gray-400">150.2B</span>
              </div>
              <p class="text-xs text-gray-400 mt-1">Second fastest model in terms of FFTP
and first-token</p>
              <div class="mt-2 text-xs text-primary">Context: 16,000 tokens</div>
            </div>

            <div class="model-card bg-dark p-3 rounded-lg cursor-pointer"
data-model="COGNITO-FLASH">
              <div class="flex justify-between items-center">
                <h4 class="font-semibold">COGNITO-FLASH</h4>
                <span class="text-xs text-gray-400">217B</span>
              </div>
              <p class="text-xs text-gray-400 mt-1">Fastest, multimodal and versatile
model</p>
              <div class="mt-2 text-xs text-primary">Context: 16,000 tokens</div>
            </div>
```

```
                <div class="model-card bg-dark p-3 rounded-lg cursor-pointer"
data-model="COGNITO-3">
                    <div class="flex justify-between items-center">
                        <h4 class="font-semibold">COGNITO-3</h4>
                        <span class="text-xs text-gray-400">32B</span>
                    </div>
                    <p class="text-xs text-gray-400 mt-1">Capable model for coding and app/web
development</p>
                    <div class="mt-2 text-xs text-primary">Context: 8,096 tokens</div>
                </div>

                <div class="model-card bg-dark p-3 rounded-lg cursor-pointer"
data-model="COGNITO-3.1-R">
                    <div class="flex justify-between items-center">
                        <h4 class="font-semibold">COGNITO-3.1-R</h4>
                        <span class="text-xs text-gray-400">617B</span>
                    </div>
                    <p class="text-xs text-gray-400 mt-1">First-ever reasoning model based on
R1 CoT-architecture</p>
                    <div class="mt-2 text-xs text-primary">Context: 162,000 tokens</div>
                </div>

                <div class="model-card bg-dark p-3 rounded-lg cursor-pointer"
data-model="COGNITO-3.5-R">
                    <div class="flex justify-between items-center">
                        <h4 class="font-semibold">COGNITO-3.5-R</h4>
                        <span class="text-xs text-gray-400">672B</span>
                    </div>
                    <p class="text-xs text-gray-400 mt-1">Powerful model with well-rounded skills
and versatility</p>
                    <div class="mt-2 text-xs text-primary">Context: 162,000 tokens</div>
                </div>

                <div class="model-card bg-dark p-3 rounded-lg cursor-pointer"
data-model="COGNITO-3.7-commandD">
                    <div class="flex justify-between items-center">
                        <h4 class="font-semibold">COGNITO-3.7-commandD</h4>
                        <span class="text-xs text-gray-400">70B</span>
                    </div>
                    <p class="text-xs text-gray-400 mt-1">Distilled reasoning model for faster
responses</p>
                    <div class="mt-2 text-xs text-primary">Context: 162,000 tokens</div>
                </div>
```

```
                <div class="model-card bg-dark p-3 rounded-lg cursor-pointer"
data-model="COGNITO-command-4">
                    <div class="flex justify-between items-center">
                        <h4 class="font-semibold">COGNITO-command-4 (BETA)</h4>
                        <span class="text-xs text-gray-400">Latest</span>
                    </div>
                    <p class="text-xs text-gray-400 mt-1">Latest, most powerful, highly versatile
multimodal LLM</p>
                    <div class="mt-2 text-xs text-primary">Context: 16,000 tokens</div>
                </div>

                <div class="model-card bg-dark p-3 rounded-lg cursor-pointer opacity-70"
data-model="COGNITO-commandA-8B">
                    <div class="flex justify-between items-center">
                        <h4 class="font-semibold">COGNITO-commandA-8B</h4>
                        <span class="text-xs text-yellow-400">Coming Soon</span>
                    </div>
                    <p class="text-xs text-gray-400 mt-1">Specializes in Creative Writing, Story
generation</p>
                    <div class="mt-2 text-xs text-primary">Context: 16,000 tokens</div>
                </div>
            </div>
        </div>

        <!-- Subjects Section -->
        <div class="mb-6">
            <h3 class="text-lg font-semibold mb-3">Subjects</h3>
            <div class="grid grid-cols-2 gap-2">
                <div class="bg-dark p-2 rounded-lg text-center text-sm">Coding</div>
                <div class="bg-dark p-2 rounded-lg text-center text-sm">Math</div>
                <div class="bg-dark p-2 rounded-lg text-center text-sm">Physics</div>
                <div class="bg-dark p-2 rounded-lg text-center text-sm">Chemistry</div>
                <div class="bg-dark p-2 rounded-lg text-center text-sm">Biology</div>
                <div class="bg-dark p-2 rounded-lg text-center text-sm">History</div>
                <div class="bg-dark p-2 rounded-lg text-center text-sm">English</div>
                <div class="bg-dark p-2 rounded-lg text-center text-sm">Arts</div>
                <div class="bg-dark p-2 rounded-lg text-center text-sm">Law</div>
                <div class="bg-dark p-2 rounded-lg text-center text-sm">Business</div>
            </div>
        </div>

        <!-- Articles Section -->
        <div class="mb-6">
```

```
            <h3 class="text-lg font-semibold mb-3">Articles</h3>
            <div class="space-y-2">
              <div class="bg-dark p-3 rounded-lg">
                <h4 class="font-medium text-sm">Coming Soon...</h4>
              </div>
              <div class="bg-dark p-3 rounded-lg">
                <h4 class="font-medium text-sm">Coming Soon...</h4>
              </div>
              <div class="bg-dark p-3 rounded-lg">
                <h4 class="font-medium text-sm">Coming Soon...</h4>
              </div>
            </div>
          </div>
        </div>

        <!-- Chat Area - Standard UI -->
        <div id="standard-ui" class="flex-grow flex flex-col overflow-hidden">
          <!-- Chat Messages Container -->
          <div id="chat-container" class="flex-grow overflow-y-auto p-4 space-y-6">
            <!-- Messages will be added here dynamically -->
          </div>

          <!-- Chat Input Area -->
          <div class="p-4 border-t border-gray-700">
            <div class="relative">
              <textarea id="message-input" class="w-full p-3 pr-16 rounded-lg bg-darkgray
text-white border border-gray-600 focus:outline-none focus:border-primary min-h-[50px]
max-h-[150px] text-base" placeholder="Message COGNITO..."></textarea>
              <button id="send-btn" class="absolute right-3 bottom-3 text-primary
hover:text-white">
                <i class="fas fa-paper-plane"></i>
              </button>
            </div>

            <!-- Bottom Action Bar -->
            <div class="flex justify-between items-center mt-4">
              <div class="flex flex-wrap gap-2">
                <button id="extended-mode-btn" class="bg-darkgray hover:bg-dark px-3 py-2
rounded-full text-xs font-semibold flex items-center">
                  <i class="fas fa-expand-alt mr-1 text-primary"></i>
                  EXTENDED MODE
                </button>
                <button id="delete-chats-btn" class="bg-darkgray hover:bg-dark px-3 py-2
rounded-full text-xs font-semibold flex items-center">
```

```
                        <i class="fas fa-trash-alt mr-1 text-red-500"></i>
                        DELETE CHATS
                    </button>
                    <button id="stop-btn" class="bg-darkgray hover:bg-dark px-3 py-2 rounded-full
text-xs font-semibold flex items-center">
                        <i class="fas fa-stop-circle mr-1 text-yellow-500"></i>
                        STOP
                    </button>
                    <button id="incognito-btn" class="bg-darkgray hover:bg-dark px-3 py-2
rounded-full text-xs font-semibold flex items-center">
                        <i class="fas fa-user-secret mr-1 text-blue-400"></i>
                        InCOGNITO MODE
                    </button>
                </div>
                <button id="chat-cognito-btn" class="bg-primary hover:bg-opacity-90 px-4 py-2
rounded-lg text-white text-sm font-semibold flex items-center">
                    CHAT COGNITO
                    <i class="fas fa-arrow-right ml-2"></i>
                </button>
            </div>
        </div>
    </div>

    <!-- Chat Area - Modern UI (initially hidden) -->
    <div id="modern-ui" class="hidden flex-grow flex flex-col overflow-hidden">
        <div class="flex items-center justify-center p-4 mb-4">
            <img
src="https://assets.onecompiler.app/42u3pr9mz/43cnbgq3z/EXTENDED%20MODE_20250323_
164008_0000.png" alt="COGNITO Extended" class="h-12">
        </div>

        <!-- Chat Messages Container for Modern UI -->
        <div id="modern-chat-container" class="flex-grow overflow-y-auto p-4 space-y-6">
            <!-- Messages will be added here dynamically -->
        </div>

        <!-- Chat Input Area for Modern UI -->
        <div class="p-4 border-t border-gray-700">
            <div class="relative">
                <textarea id="modern-message-input" class="w-full p-3 pr-16 rounded-lg
bg-darkgray text-white border border-gray-600 focus:outline-none focus:border-primary
min-h-[50px] max-h-[150px] text-base" placeholder="Ask anything..."></textarea>
                <button id="modern-send-btn" class="absolute right-3 bottom-3 text-primary
hover:text-white">
```

```
                    <i class="fas fa-paper-plane"></i>
                </button>
            </div>

            <!-- Bottom Action Bar for Modern UI -->
            <div class="flex justify-between items-center mt-4">
                <div class="flex space-x-2">
                    <button id="modern-extended-mode-btn" class="bg-primary bg-opacity-10
hover:bg-opacity-20 px-3 py-2 rounded-lg text-xs font-semibold flex items-center">
                        <i class="fas fa-expand-alt mr-1 text-primary"></i>
                        EXTENDED
                    </button>
                    <button id="modern-stop-btn" class="bg-primary bg-opacity-10
hover:bg-opacity-20 px-3 py-2 rounded-lg text-xs font-semibold flex items-center">
                        <i class="fas fa-stop-circle mr-1 text-yellow-500"></i>
                        STOP
                    </button>
                    <button id="modern-incognito-btn" class="bg-primary bg-opacity-10
hover:bg-opacity-20 px-3 py-2 rounded-lg text-xs font-semibold flex items-center">
                        <i class="fas fa-user-secret mr-1 text-blue-400"></i>
                        InCOGNITO
                    </button>
                </div>
                <div class="flex space-x-2">
                    <button id="modern-delete-chats-btn" class="bg-primary bg-opacity-10
hover:bg-opacity-20 p-2 rounded-lg text-xs">
                        <i class="fas fa-trash-alt text-red-500"></i>
                    </button>
                    <button id="modern-save-chat-btn" class="bg-primary bg-opacity-10
hover:bg-opacity-20 p-2 rounded-lg text-xs">
                        <i class="fas fa-download text-primary"></i>
                    </button>
                </div>
            </div>
        </div>
    </div>

    <!-- Research Mode Interface (initially hidden) -->
    <div id="research-ui" class="hidden flex-grow flex flex-col overflow-hidden">
        <div class="flex items-center justify-between p-4 border-b border-gray-700">
            <div class="flex items-center">
                <button id="back-from-research-btn" class="mr-4 text-gray-400
hover:text-white">
                    <i class="fas fa-arrow-left"></i>
```

```
                    </button>
                    <h2 class="text-xl font-bold">Research Mode</h2>
                </div>
                <div class="flex items-center space-x-4">
                    <div class="flex items-center">
                        <span class="text-sm mr-2">MODEL:</span>
                        <span id="current-research-model" class="text-primary
font-semibold">COGNITO-commandRS</span>
                        <button id="toggle-research-model-btn" class="ml-2 text-gray-400
hover:text-white">
                            <i class="fas fa-exchange-alt"></i>
                        </button>
                    </div>
                </div>
            </div>

            <!-- Research Chat Messages Container -->
            <div id="research-chat-container" class="flex-grow overflow-y-auto p-4 space-y-6">
                <div class="bg-darkgray p-4 rounded-lg">
                    <h3 class="font-semibold mb-2">Research Assistant</h3>
                    <p class="text-sm text-gray-300">I'm your research assistant. I can help you find
accurate and relevant information on any topic. Ask me a question to get started.</p>
                </div>
            </div>

            <!-- Research Chat Input Area -->
            <div class="p-4 border-t border-gray-700">
                <div class="relative">
                    <textarea id="research-message-input" class="w-full p-3 pr-16 rounded-lg
bg-darkgray text-white border border-gray-600 focus:outline-none focus:border-primary
min-h-[50px] max-h-[150px] text-base" placeholder="Research anything..."></textarea>
                    <button id="research-send-btn" class="absolute right-3 bottom-3 text-primary
hover:text-white">
                        <i class="fas fa-paper-plane"></i>
                    </button>
                </div>

                <!-- Research Bottom Action Bar -->
                <div class="flex justify-between items-center mt-4">
                    <div class="flex space-x-2">
                        <button id="research-extended-mode-btn" class="bg-darkgray hover:bg-dark
px-3 py-2 rounded-full text-xs font-semibold flex items-center">
                            <i class="fas fa-expand-alt mr-1 text-primary"></i>
                            EXTENDED MODE
```

```
                    </button>
                    <button id="research-stop-btn" class="bg-darkgray hover:bg-dark px-3 py-2
rounded-full text-xs font-semibold flex items-center">
                        <i class="fas fa-stop-circle mr-1 text-yellow-500"></i>
                        STOP
                    </button>
                </div>
                <div class="flex space-x-2">
                    <button id="research-clear-btn" class="bg-darkgray hover:bg-dark px-3 py-2
rounded-full text-xs font-semibold flex items-center">
                        <i class="fas fa-broom mr-1 text-red-500"></i>
                        CLEAR
                    </button>
                    <button id="research-save-btn" class="bg-darkgray hover:bg-dark px-3 py-2
rounded-full text-xs font-semibold flex items-center">
                        <i class="fas fa-download mr-1 text-blue-400"></i>
                        SAVE
                    </button>
                </div>
            </div>
        </div>
    </div>
</div>

<script>
    // Dark mode detection
    if (window.matchMedia && window.matchMedia('(prefers-color-scheme: dark)').matches) {
        document.documentElement.classList.add('dark');
    }
    window.matchMedia('(prefers-color-scheme: dark)').addEventListener('change', event => {
        if (event.matches) {
            document.documentElement.classList.add('dark');
        } else {
            document.documentElement.classList.remove('dark');
        }
    });

    // DOM Elements - Welcome Page
    const welcomePage = document.getElementById('welcome-page');
    const chatPage = document.getElementById('chat-page');
    const nextBtn = document.getElementById('next-btn');
    const usernameInput = document.getElementById('username');
```

```
        // DOM Elements - Standard UI
        const currentModelEl = document.getElementById('current-model');
        const messageInput = document.getElementById('message-input');
        const sendBtn = document.getElementById('send-btn');
        const chatContainer = document.getElementById('chat-container');
        const settingsPanel = document.getElementById('settings-panel');
        const infoBtn = document.getElementById('info-btn');
        const extendedModeBtn = document.getElementById('extended-mode-btn');
        const deleteChatsBtn = document.getElementById('delete-chats-btn');
        const stopBtn = document.getElementById('stop-btn');
        const incognitoBtn = document.getElementById('incognito-btn');
        const chatCognitoBtn = document.getElementById('chat-cognito-btn');
        const saveChatBtn = document.getElementById('save-chat-btn');

        // DOM Elements - Modern UI
        const standardUI = document.getElementById('standard-ui');
        const modernUI = document.getElementById('modern-ui');
        const switchUIBtn = document.getElementById('switch-ui-btn');
        const modernChatContainer = document.getElementById('modern-chat-container');
        const modernMessageInput = document.getElementById('modern-message-input');
        const modernSendBtn = document.getElementById('modern-send-btn');
        const modernExtendedModeBtn =
document.getElementById('modern-extended-mode-btn');
        const modernStopBtn = document.getElementById('modern-stop-btn');
        const modernIncognitoBtn = document.getElementById('modern-incognito-btn');
        const modernDeleteChatsBtn = document.getElementById('modern-delete-chats-btn');
        const modernSaveChatBtn = document.getElementById('modern-save-chat-btn');

        // DOM Elements - Research Mode
        const researchUI = document.getElementById('research-ui');
        const researchModeBtn = document.getElementById('research-mode-btn');
        const backFromResearchBtn = document.getElementById('back-from-research-btn');
        const currentResearchModelEl = document.getElementById('current-research-model');
        const toggleResearchModelBtn = document.getElementById('toggle-research-model-btn');
        const researchChatContainer = document.getElementById('research-chat-container');
        const researchMessageInput = document.getElementById('research-message-input');
        const researchSendBtn = document.getElementById('research-send-btn');
        const researchExtendedModeBtn =
document.getElementById('research-extended-mode-btn');
        const researchStopBtn = document.getElementById('research-stop-btn');
        const researchClearBtn = document.getElementById('research-clear-btn');
        const researchSaveBtn = document.getElementById('research-save-btn');

        // State management
```

```
let state = {
    username: 'User',
    currentMode: 'standard', // 'standard', 'modern', or 'research'
    currentModel: 'COGNITO-command-4',
    currentResearchModel: 'COGNITO-commandRS',
    modelMap: {
        'COGNITO-1': 'google/gemma-3-27b-it:free',
        'COGNITO-2': 'google/gemini-2.0-flash-lite-preview-02-05:free',
        'COGNITO-3': 'deepseek/deepseek-chat:free',
        'COGNITO-3.1-R': 'open-r1/olympiccoder-32b:free',
        'COGNITO-3.5-R': 'deepseek/deepseek-r1:free',
        'COGNITO-3.7-commandD': 'deepseek/deepseek-r1-distill-llama-70b:free',
        'COGNITO-FLASH': 'google/gemini-2.0-pro-exp-02-05:free',
        'COGNITO-command-4': 'mistralai/mistral-small-3.1-24b-instruct:free',
        'COGNITO-commandRS': 'cognitivecomputations/dolphin3.0-mistral-24b:free',
        'COGNITO-commandO+': 'google/gemini-2.0-flash-thinking-exp:free'
    },
    modelInstructions: {
        'COGNITO-1': 'You are COGNITO-1, COGNITO\'s first language model trained on billions of data with 34% being code. You understand NLPs well and can do a variety of tasks.',
        'COGNITO-2': 'You are COGNITO-2, COGNITO\'s second fastest model in terms of FFTP and first-token. You ensure accuracy within a faster time frame in responding.',
        'COGNITO-3': 'You are COGNITO-3, COGNITO\'s most capable and versatile model for coding, programming and app/web development—a compact yet powerful model prioritizing bug-free outputs.',
        'COGNITO-3.1-R': 'You are COGNITO-3.1-R, COGNITO\'s first-ever reasoning model based on the R1 CoT-architecture. This is a coding model with a whopping 617B parameters.',
        'COGNITO-3.5-R': 'You are COGNITO-3.5-R, a powerful model from COGNITO with well-rounded skills and versatility—excellent in coding, math, creative writing, and complex task & problem-solving.',
        'COGNITO-3.7-commandD': 'You are COGNITO-3.7-commandD, a 70B-distilled reasoning model from COGNITO made to provide responses a little more faster than COGNITO-3.5 & 3.1-R while reasoning is still present.',
        'COGNITO-FLASH': 'You are COGNITO-FLASH, the fastest, multimodal and one of the most versatile models amongst the COGNITO models behind COGNITO-3.7.',
        'COGNITO-command-4': 'You are COGNITO-command-4, the latest, most powerful, highly versatile, and a multimodal large language model from COGNITO that\'s 6x smaller, more compact yet capable of outperforming much larger language models.',
        'COGNITO-commandRS': 'You are COGNITO-commandRS, a researching model from COGNITO that focuses on providing thorough, accurate and relevant information all while having the ability to reason and respond fast — with internet via Google access.',
        'COGNITO-commandO+': 'You are COGNITO-commandO+, a researching model from COGNITO with O+ which stands for Optimized+. You opt for concise, direct yet detailed responses/outputs.'
```

```
        },
        isExtendedMode: false,
        isInCognitoMode: false,
        isStopped: false,
        isGenerating: false,
        messages: [],
        researchMessages: [],
        apiKey:
'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'
    };

    // Push notification function
    function showNotification(title, message) {
        if (!("Notification" in window)) {
            console.log("This browser does not support notifications");
            return;
        }

        // If we cannot use the Notification API directly, we'll show a UI notification
        const notificationContainer = document.getElementById('notification-container');
        const notificationEl = document.createElement('div');
        notificationEl.className = 'notification';
        notificationEl.innerHTML = `
            <div class="font-semibold">${title}</div>
            <div class="text-sm mt-1">${message}</div>
        `;

        notificationContainer.appendChild(notificationEl);

        // Remove the notification after 6 seconds
        setTimeout(() => {
            notificationEl.remove();
        }, 6000);

        // Try to use the Notification API if available and allowed
        if (Notification.permission === "granted") {
            const notification = new Notification(title, {
                body: message,
                icon: "https://assets.onecompiler.app/42u3pr9mz/43cjuxhs6/1742640682179.jpg"
            });
        }
        else if (Notification.permission !== "denied") {
            Notification.requestPermission().then(permission => {
                if (permission === "granted") {
```

```
                    const notification = new Notification(title, {
                        body: message,
                        icon:
"https://assets.onecompiler.app/42u3pr9mz/43cjuxhs6/1742640682179.jpg"
                    });
                }
            });
        }
    }

    // Page Navigation
    nextBtn.addEventListener('click', () => {
        if (usernameInput.value.trim()) {
            state.username = usernameInput.value.trim();
        }
        welcomePage.classList.add('hidden');
        chatPage.classList.remove('hidden');

        // Add welcome message
        addBotMessage(`Hello, ${state.username}! I'm COGNITO, your AI assistant. How can I
help you today?`);
    });

    // Toggle Settings Panel
    infoBtn.addEventListener('click', () => {
        settingsPanel.classList.toggle('hidden');
    });

    // Switch UI Mode
    switchUIBtn.addEventListener('click', () => {
        if (state.currentMode === 'standard') {
            standardUI.classList.add('hidden');
            modernUI.classList.remove('hidden');
            researchUI.classList.add('hidden');
            state.currentMode = 'modern';
            // Copy all messages from standard to modern UI
            modernChatContainer.innerHTML = chatContainer.innerHTML;
        } else {
            standardUI.classList.remove('hidden');
            modernUI.classList.add('hidden');
            researchUI.classList.add('hidden');
            state.currentMode = 'standard';
            // Copy all messages from modern to standard UI
            chatContainer.innerHTML = modernChatContainer.innerHTML;
```

```
    }
});

// Toggle Research Mode
researchModeBtn.addEventListener('click', () => {
    if (state.currentMode !== 'research') {
        standardUI.classList.add('hidden');
        modernUI.classList.add('hidden');
        researchUI.classList.remove('hidden');
        state.currentMode = 'research';
    }
});

// Back from Research Mode
backFromResearchBtn.addEventListener('click', () => {
    if (state.currentMode === 'research') {
        researchUI.classList.add('hidden');
        if (standardUI.classList.contains('hidden')) {
            modernUI.classList.remove('hidden');
            state.currentMode = 'modern';
        } else {
            standardUI.classList.remove('hidden');
            state.currentMode = 'standard';
        }
    }
});

// Toggle Research Model
toggleResearchModelBtn.addEventListener('click', () => {
    if (state.currentResearchModel === 'COGNITO-commandRS') {
        state.currentResearchModel = 'COGNITO-commandO+';
    } else {
        state.currentResearchModel = 'COGNITO-commandRS';
    }
    currentResearchModelEl.textContent = state.currentResearchModel;

    // Add model change notification
    const notification = document.createElement('div');
    notification.className = 'text-center text-xs text-gray-400 my-2 animate-fade-in';
    notification.textContent = `Switched to ${state.currentResearchModel}`;
    researchChatContainer.appendChild(notification);
    researchChatContainer.scrollTop = researchChatContainer.scrollHeight;
});
```

```
// Model Selection
document.querySelectorAll('.model-card').forEach(card => {
    card.addEventListener('click', () => {
        const model = card.dataset.model;
        if (state.modelMap[model] || model === 'COGNITO-commandA-8B') {
            state.currentModel = model;
            currentModelEl.textContent = model;
            settingsPanel.classList.add('hidden');

            // Add model change notification to both UIs
            const notification = document.createElement('div');
            notification.className = 'text-center text-xs text-gray-400 my-2 animate-fade-in';
            notification.textContent = `Switched to ${model}`;
            chatContainer.appendChild(notification.cloneNode(true));
            modernChatContainer.appendChild(notification);
            chatContainer.scrollTop = chatContainer.scrollHeight;
            modernChatContainer.scrollTop = modernChatContainer.scrollHeight;
        }
    });
});

// Extended Mode Toggle (Standard UI)
extendedModeBtn.addEventListener('click', () => {
    toggleExtendedMode();
});

// Extended Mode Toggle (Modern UI)
modernExtendedModeBtn.addEventListener('click', () => {
    toggleExtendedMode();
});

// Extended Mode Toggle (Research UI)
researchExtendedModeBtn.addEventListener('click', () => {
    toggleExtendedMode();
});

function toggleExtendedMode() {
    state.isExtendedMode = !state.isExtendedMode;

    // Update UI for all interfaces
    extendedModeBtn.classList.toggle('bg-primary', state.isExtendedMode);
    extendedModeBtn.classList.toggle('bg-darkgray', !state.isExtendedMode);

    modernExtendedModeBtn.classList.toggle('bg-primary', state.isExtendedMode);
```

```
    modernExtendedModeBtn.classList.toggle('bg-opacity-10', !state.isExtendedMode);
    modernExtendedModeBtn.classList.toggle('bg-opacity-80', state.isExtendedMode);

    researchExtendedModeBtn.classList.toggle('bg-primary', state.isExtendedMode);
    researchExtendedModeBtn.classList.toggle('bg-darkgray', !state.isExtendedMode);

    // Add notification to appropriate container
    const notification = document.createElement('div');
    notification.className = 'text-center text-xs text-gray-400 my-2 animate-fade-in';
    notification.textContent = state.isExtendedMode ?
        'Extended Mode enabled. Responses up to 16,000 tokens.' :
        'Extended Mode disabled.';

    if (state.currentMode === 'standard') {
        chatContainer.appendChild(notification.cloneNode(true));
        chatContainer.scrollTop = chatContainer.scrollHeight;
    } else if (state.currentMode === 'modern') {
        modernChatContainer.appendChild(notification.cloneNode(true));
        modernChatContainer.scrollTop = modernChatContainer.scrollHeight;
    } else if (state.currentMode === 'research') {
        researchChatContainer.appendChild(notification.cloneNode(true));
        researchChatContainer.scrollTop = researchChatContainer.scrollHeight;
    }
}

// InCOGNITO Mode Toggle (Standard UI)
incognitoBtn.addEventListener('click', () => {
    toggleInCognitoMode();
});

// InCOGNITO Mode Toggle (Modern UI)
modernIncognitoBtn.addEventListener('click', () => {
    toggleInCognitoMode();
});

function toggleInCognitoMode() {
    state.isInCognitoMode = !state.isInCognitoMode;

    // Update UI for all interfaces
    incognitoBtn.classList.toggle('bg-primary', state.isInCognitoMode);
    incognitoBtn.classList.toggle('bg-darkgray', !state.isInCognitoMode);

    modernIncognitoBtn.classList.toggle('bg-primary', state.isInCognitoMode);
    modernIncognitoBtn.classList.toggle('bg-opacity-10', !state.isInCognitoMode);
```

```
    modernIncognitoBtn.classList.toggle('bg-opacity-80', state.isInCognitoMode);

    // Add notification to appropriate container
    const notification = document.createElement('div');
    notification.className = 'text-center text-xs text-gray-400 my-2 animate-fade-in';
    notification.textContent = state.isInCognitoMode ?
        'InCOGNITO Mode enabled. All messages will be encrypted.' :
        'InCOGNITO Mode disabled.';

    if (state.currentMode === 'standard') {
        chatContainer.appendChild(notification);
        chatContainer.scrollTop = chatContainer.scrollHeight;
    } else if (state.currentMode === 'modern') {
        modernChatContainer.appendChild(notification);
        modernChatContainer.scrollTop = modernChatContainer.scrollHeight;
    }
}

// Delete Chats (Standard UI)
deleteChatsBtn.addEventListener('click', () => {
    deleteChats('standard');
});

// Delete Chats (Modern UI)
modernDeleteChatsBtn.addEventListener('click', () => {
    deleteChats('modern');
});

// Delete Chats (Research UI)
researchClearBtn.addEventListener('click', () => {
    deleteChats('research');
});

function deleteChats(mode) {
    if (confirm('Are you sure you want to delete all chat messages?')) {
        if (mode === 'standard' || mode === 'modern') {
            chatContainer.innerHTML = '';
            modernChatContainer.innerHTML = '';
            state.messages = [];

            // Add deletion notification
            const notification = document.createElement('div');
            notification.className = 'text-center text-xs text-gray-400 my-2 animate-fade-in';
            notification.textContent = 'Chat history cleared';
```

```
                chatContainer.appendChild(notification.cloneNode(true));
                modernChatContainer.appendChild(notification);
            } else if (mode === 'research') {
                researchChatContainer.innerHTML = '';
                state.researchMessages = [];

                // Add welcome message back
                const welcomeMsg = document.createElement('div');
                welcomeMsg.className = 'bg-darkgray p-4 rounded-lg';
                welcomeMsg.innerHTML = `
                    <h3 class="font-semibold mb-2">Research Assistant</h3>
                    <p class="text-sm text-gray-300">I'm your research assistant. I can help you find accurate and relevant information on any topic. Ask me a question to get started.</p>
                `;
                researchChatContainer.appendChild(welcomeMsg);
            }
        }
    }

    // Save Chat (Standard UI)
    saveChatBtn.addEventListener('click', () => {
        saveChat();
    });

    // Save Chat (Modern UI)
    modernSaveChatBtn.addEventListener('click', () => {
        saveChat();
    });

    // Save Chat (Research UI)
    researchSaveBtn.addEventListener('click', () => {
        saveResearchChat();
    });

    function saveChat() {
        let chatText = '';

        // Iterate through messages and format them
        state.messages.forEach(msg => {
            if (msg.role === 'user') {
                chatText += `${state.username}: ${msg.content}\n\n`;
            } else {
                chatText += `COGNITO: ${msg.content}\n\n`;
```

```
        }
    });

    downloadText(chatText, 'cognito-chat.txt');
}

function saveResearchChat() {
    let chatText = '';

    // Iterate through research messages and format them
    state.researchMessages.forEach(msg => {
        if (msg.role === 'user') {
            chatText += `${state.username}: ${msg.content}\n\n`;
        } else {
            chatText += `${state.currentResearchModel}: ${msg.content}\n\n`;
        }
    });

    downloadText(chatText, 'cognito-research.txt');
}

function downloadText(text, filename) {
    // Create a blob from the text
    const blob = new Blob([text], { type: 'text/plain' });

    // Create a URL for the blob
    const url = URL.createObjectURL(blob);

    // Create a temporary anchor element and click it
    const a = document.createElement('a');
    a.href = url;
    a.download = filename;
    document.body.appendChild(a);
    a.click();

    // Clean up
    document.body.removeChild(a);
    URL.revokeObjectURL(url);

    // Show notification
    showNotification('Chat Saved', `Chat history has been saved as ${filename}`);
}

// Stop Generation (Standard UI)
```

```
stopBtn.addEventListener('click', () => {
    stopGeneration();
});

// Stop Generation (Modern UI)
modernStopBtn.addEventListener('click', () => {
    stopGeneration();
});

// Stop Generation (Research UI)
researchStopBtn.addEventListener('click', () => {
    stopGeneration();
});

function stopGeneration() {
    if (state.isGenerating) {
        state.isStopped = true;

        // Add stop notification to appropriate container
        const notification = document.createElement('div');
        notification.className = 'text-center text-xs text-gray-400 my-2 animate-fade-in';
        notification.textContent = 'Response generation stopped';

        if (state.currentMode === 'standard') {
            chatContainer.appendChild(notification);
            chatContainer.scrollTop = chatContainer.scrollHeight;
        } else if (state.currentMode === 'modern') {
            modernChatContainer.appendChild(notification);
            modernChatContainer.scrollTop = modernChatContainer.scrollHeight;
        } else if (state.currentMode === 'research') {
            researchChatContainer.appendChild(notification);
            researchChatContainer.scrollTop = researchChatContainer.scrollHeight;
        }
    }
}

// Send Message (Standard UI)
sendBtn.addEventListener('click', () => {
    sendStandardMessage();
});
messageInput.addEventListener('keypress', (e) => {
    if (e.key === 'Enter' && !e.shiftKey) {
        e.preventDefault();
        sendStandardMessage();
```

```
    }
});

// Send Message (Modern UI)
modernSendBtn.addEventListener('click', () => {
    sendModernMessage();
});
modernMessageInput.addEventListener('keypress', (e) => {
    if (e.key === 'Enter' && !e.shiftKey) {
        e.preventDefault();
        sendModernMessage();
    }
});

// Send Message (Research UI)
researchSendBtn.addEventListener('click', () => {
    sendResearchMessage();
});
researchMessageInput.addEventListener('keypress', (e) => {
    if (e.key === 'Enter' && !e.shiftKey) {
        e.preventDefault();
        sendResearchMessage();
    }
});

function sendStandardMessage() {
    const message = messageInput.value.trim();
    if (!message || state.isGenerating) return;

    addUserMessage(message, 'standard');
    messageInput.value = '';
    messageInput.style.height = 'auto';

    // Process in InCOGNITO mode if enabled
    const processedMessage = state.isInCognitoMode ?
        encryptMessage(message) : message;

    generateResponse(processedMessage, 'standard');
}

function sendModernMessage() {
    const message = modernMessageInput.value.trim();
    if (!message || state.isGenerating) return;
```

```
            addUserMessage(message, 'modern');
            modernMessageInput.value = '';
            modernMessageInput.style.height = 'auto';

            // Process in InCOGNITO mode if enabled
            const processedMessage = state.isInCognitoMode ?
                encryptMessage(message) : message;

            generateResponse(processedMessage, 'modern');
        }

        function sendResearchMessage() {
            const message = researchMessageInput.value.trim();
            if (!message || state.isGenerating) return;

            addUserMessage(message, 'research');
            researchMessageInput.value = '';
            researchMessageInput.style.height = 'auto';

            generateResearchResponse(message);
        }

        function addUserMessage(message, uiMode) {
            const messageEl = document.createElement('div');
            messageEl.className = 'chat-message flex items-start';

            messageEl.innerHTML = `
                <div class="rounded-full bg-primary w-8 h-8 flex items-center justify-center text-white
font-bold mr-3 flex-shrink-0">
                    ${state.username.charAt(0).toUpperCase()}
                </div>
                <div class="flex-grow">
                    <div class="font-semibold">${state.username}</div>
                    <div class="mt-1 text-sm">${message}</div>
                </div>
            `;

            if (uiMode === 'standard') {
                chatContainer.appendChild(messageEl);
                chatContainer.scrollTop = chatContainer.scrollHeight;

                // Add to message history
                state.messages.push({
                    role: 'user',
```

```
            content: message
        });

        // Clone to modern UI
        modernChatContainer.appendChild(messageEl.cloneNode(true));
        modernChatContainer.scrollTop = modernChatContainer.scrollHeight;
    } else if (uiMode === 'modern') {
        modernChatContainer.appendChild(messageEl);
        modernChatContainer.scrollTop = modernChatContainer.scrollHeight;

        // Add to message history
        state.messages.push({
            role: 'user',
            content: message
        });

        // Clone to standard UI
        chatContainer.appendChild(messageEl.cloneNode(true));
        chatContainer.scrollTop = chatContainer.scrollHeight;
    } else if (uiMode === 'research') {
        researchChatContainer.appendChild(messageEl);
        researchChatContainer.scrollTop = researchChatContainer.scrollHeight;

        // Add to research message history
        state.researchMessages.push({
            role: 'user',
            content: message
        });
    }
}

function addBotMessage(message, isComplete = true, uiMode = 'standard') {
    let container, existingBotMessage;

    // Determine which container to use
    if (uiMode === 'standard' || uiMode === 'modern') {
        container = uiMode === 'standard' ? chatContainer : modernChatContainer;
        existingBotMessage = container.querySelector('.bot-typing');
    } else if (uiMode === 'research') {
        container = researchChatContainer;
        existingBotMessage = container.querySelector('.research-bot-typing');
    }

    if (existingBotMessage && !isComplete) {
```

```
            // Update the existing message
            const contentEl = existingBotMessage.querySelector('.bot-content');
            contentEl.innerHTML = marked.parse(message);
            return;
        } else if (existingBotMessage && isComplete) {
            // Complete the existing message
            existingBotMessage.classList.remove(uiMode === 'research' ? 'research-bot-typing' :
'bot-typing');
            const contentEl = existingBotMessage.querySelector('.bot-content');
            contentEl.innerHTML = marked.parse(message);

            // Add copy button
            const copyBtn = document.createElement('button');
            copyBtn.className = 'copy-btn text-gray-400 hover:text-white p-1 ml-2';
            copyBtn.innerHTML = '<i class="fas fa-copy"></i>';
            copyBtn.addEventListener('click', () => {
                navigator.clipboard.writeText(message).then(() => {
                    // Show temporary copy feedback
                    copyBtn.innerHTML = '<i class="fas fa-check"></i>';
                    setTimeout(() => {
                        copyBtn.innerHTML = '<i class="fas fa-copy"></i>';
                    }, 2000);
                });
            });
            existingBotMessage.querySelector('.flex-grow .font-semibold').appendChild(copyBtn);

            // Add continue button if in extended mode
            if (state.isExtendedMode) {
                const continueBtn = document.createElement('button');
                continueBtn.className = 'text-primary text-xs mt-2 hover:underline';
                continueBtn.textContent = 'Continue generating...';
                continueBtn.addEventListener('click', () => {
                    if (uiMode === 'standard' || uiMode === 'modern') {
                        const lastMessage = state.messages[state.messages.length - 1];
                        if (lastMessage.role === 'assistant') {
                            generateContinuation(lastMessage.content, uiMode);
                        }
                    } else if (uiMode === 'research') {
                        const lastMessage = state.researchMessages[state.researchMessages.length
- 1];
                        if (lastMessage.role === 'assistant') {
                            generateResearchContinuation(lastMessage.content);
                        }
                    }
```

```
                });
                existingBotMessage.querySelector('.flex-grow').appendChild(continueBtn);
            }

            // Show notification if document is not visible
            if (!document.hidden && isComplete) {
                showNotification('Response Complete', 'COGNITO has finished responding to your
message.');
            }

            return;
        }

        // Create a new bot message
        const messageEl = document.createElement('div');
        messageEl.className = `chat-message flex items-start ${!isComplete ? (uiMode ===
'research' ? 'research-bot-typing' : 'bot-typing') : ''}`;

        messageEl.innerHTML = `
            <div class="logo-container bg-primary p-1 w-10 h-10 flex items-center justify-center
mr-3 flex-shrink-0">
                <img
src="https://assets.onecompiler.app/42u3pr9mz/43cjuxhs6/1742640682179.jpg"
alt="COGNITO" class="h-6">
            </div>
            <div class="flex-grow">
                <div class="font-semibold">${uiMode === 'research' ? state.currentResearchModel
: 'COGNITO'}</div>
                <div class="mt-1 text-sm bot-content">${marked.parse(message)}</div>
                ${!isComplete ? `<div class="typing-indicator text-xs text-gray-400
mt-2">Generating</div>` : ''}
            </div>
        `;

        // Add copy button if message is complete
        if (isComplete) {
            const copyBtn = document.createElement('button');
            copyBtn.className = 'copy-btn text-gray-400 hover:text-white p-1 ml-2';
            copyBtn.innerHTML = '<i class="fas fa-copy"></i>';
            copyBtn.addEventListener('click', () => {
                navigator.clipboard.writeText(message).then(() => {
                    // Show temporary copy feedback
                    copyBtn.innerHTML = '<i class="fas fa-check"></i>';
                    setTimeout(() => {
```

```
                copyBtn.innerHTML = '<i class="fas fa-copy"></i>';
            }, 2000);
        });
    });
    messageEl.querySelector('.flex-grow .font-semibold').appendChild(copyBtn);
}

if (uiMode === 'standard') {
    chatContainer.appendChild(messageEl);
    chatContainer.scrollTop = chatContainer.scrollHeight;

    if (isComplete) {
        // Add to message history
        state.messages.push({
            role: 'assistant',
            content: message
        });

        // Clone to modern UI if needed
        const modernClone = messageEl.cloneNode(true);
        modernChatContainer.appendChild(modernClone);
        modernChatContainer.scrollTop = modernChatContainer.scrollHeight;
    }
} else if (uiMode === 'modern') {
    modernChatContainer.appendChild(messageEl);
    modernChatContainer.scrollTop = modernChatContainer.scrollHeight;

    if (isComplete) {
        // Add to message history
        state.messages.push({
            role: 'assistant',
            content: message
        });

        // Clone to standard UI if needed
        const standardClone = messageEl.cloneNode(true);
        chatContainer.appendChild(standardClone);
        chatContainer.scrollTop = chatContainer.scrollHeight;
    }
} else if (uiMode === 'research') {
    researchChatContainer.appendChild(messageEl);
    researchChatContainer.scrollTop = researchChatContainer.scrollHeight;

    if (isComplete) {
```

```
                // Add to research message history
                state.researchMessages.push({
                    role: 'assistant',
                    content: message
                });
            }
        }
    }

    async function generateResponse(message, uiMode) {
        if (state.currentModel === 'COGNITO-commandA-8B') {
            addBotMessage('Sorry, COGNITO-commandA-8B is coming soon and not yet
available.', true, uiMode);
            return;
        }

        state.isGenerating = true;
        state.isStopped = false;

        // Start with loading message
        addBotMessage('', false, uiMode);

        try {
            // Build message history for context
            const messageHistory = state.messages.slice(-10).map(msg => ({
                role: msg.role,
                content: msg.content
            }));

            // Add current message
            messageHistory.push({
                role: 'user',
                content: message
            });

            // Add system message with model instructions
            if (state.modelInstructions[state.currentModel]) {
                messageHistory.unshift({
                    role: 'system',
                    content: state.modelInstructions[state.currentModel]
                });
            }

            // Prepare API request
```

```
const modelEndpoint = state.modelMap[state.currentModel];
const requestOptions = {
    model: modelEndpoint,
    messages: messageHistory,
    max_tokens: state.isExtendedMode ? 16000 : 4000,
    temperature: 0.7,
    stream: true
};

// Create stream for chunked response
const response = await fetch('https://openrouter.ai/api/v1/chat/completions', {
    method: 'POST',
    headers: {
        'Authorization': `Bearer ${state.apiKey}`,
        'Content-Type': 'application/json'
    },
    body: JSON.stringify(requestOptions)
});

if (!response.ok) {
    throw new Error(`API responded with status: ${response.status}`);
}

const reader = response.body.getReader();
const decoder = new TextDecoder();
let responseText = '';

while (true) {
    // Check if stopped
    if (state.isStopped) {
        addBotMessage(responseText || 'Response generation stopped.', true, uiMode);
        break;
    }

    const { done, value } = await reader.read();

    if (done) {
        // Final complete message
        addBotMessage(responseText, true, uiMode);
        break;
    }

    // Decode the chunk and handle it
    const chunk = decoder.decode(value);
```

```
                // Parse SSE data
                const lines = chunk.split('\n');
                let newContent = '';

                for (const line of lines) {
                    if (line.startsWith('data:')) {
                        const data = line.slice(5).trim();
                        if (data === '[DONE]') continue;

                        try {
                            const parsed = JSON.parse(data);
                            if (parsed.choices && parsed.choices[0] && parsed.choices[0].delta &&
parsed.choices[0].delta.content) {
                                newContent += parsed.choices[0].delta.content;
                            }
                        } catch (e) {
                            console.error('Error parsing SSE chunk:', e);
                        }
                    }
                }

                if (newContent) {
                    responseText += newContent;
                    addBotMessage(responseText, false, uiMode);
                }
            }
        } catch (error) {
            console.error('API Error:', error);
            addBotMessage(`I'm sorry, I encountered an error: ${error.message}`, true, uiMode);
        } finally {
            state.isGenerating = false;
        }
    }

    async function generateResearchResponse(message) {
        state.isGenerating = true;
        state.isStopped = false;

        // Start with loading message
        addBotMessage('', false, 'research');

        try {
            // Build message history for context
```

```
const messageHistory = state.researchMessages.slice(-10).map(msg => ({
    role: msg.role,
    content: msg.content
}));

// Add current message
messageHistory.push({
    role: 'user',
    content: message
});

// Add system message with model instructions
if (state.modelInstructions[state.currentResearchModel]) {
    messageHistory.unshift({
        role: 'system',
        content: state.modelInstructions[state.currentResearchModel]
    });
}

// Prepare API request
const modelEndpoint = state.modelMap[state.currentResearchModel];
const requestOptions = {
    model: modelEndpoint,
    messages: messageHistory,
    max_tokens: state.isExtendedMode ? 16000 : 4000,
    temperature: 0.7,
    stream: true
};

// Create stream for chunked response
const response = await fetch('https://openrouter.ai/api/v1/chat/completions', {
    method: 'POST',
    headers: {
        'Authorization': `Bearer ${state.apiKey}`,
        'Content-Type': 'application/json'
    },
    body: JSON.stringify(requestOptions)
});

if (!response.ok) {
    throw new Error(`API responded with status: ${response.status}`);
}

const reader = response.body.getReader();
```

```
            const decoder = new TextDecoder();
            let responseText = '';

            while (true) {
                // Check if stopped
                if (state.isStopped) {
                    addBotMessage(responseText || 'Response generation stopped.', true,
'research');
                    break;
                }

                const { done, value } = await reader.read();

                if (done) {
                    // Final complete message
                    addBotMessage(responseText, true, 'research');
                    break;
                }

                // Decode the chunk and handle it
                const chunk = decoder.decode(value);

                // Parse SSE data
                const lines = chunk.split('\n');
                let newContent = '';

                for (const line of lines) {
                    if (line.startsWith('data:')) {
                        const data = line.slice(5).trim();
                        if (data === '[DONE]') continue;

                        try {
                            const parsed = JSON.parse(data);
                            if (parsed.choices && parsed.choices[0] && parsed.choices[0].delta &&
parsed.choices[0].delta.content) {
                                newContent += parsed.choices[0].delta.content;
                            }
                        } catch (e) {
                            console.error('Error parsing SSE chunk:', e);
                        }
                    }
                }

                if (newContent) {
```

```
                    responseText += newContent;
                    addBotMessage(responseText, false, 'research');
                }
            }
        } catch (error) {
            console.error('API Error:', error);
            addBotMessage(`I'm sorry, I encountered an error: ${error.message}`, true,
'research');
        } finally {
            state.isGenerating = false;
        }
    }

    async function generateContinuation(previousContent, uiMode) {
        if (state.isGenerating) return;

        state.isGenerating = true;
        state.isStopped = false;

        // Start with loading message
        addBotMessage(`${previousContent}\n\n_Continuing..._`, false, uiMode);

        try {
            // Prepare continuation prompt
            const continuationPrompt = `Continue exactly where you left off, without repeating
anything. Pick up from:\n\n${previousContent.slice(-200)}`;

            // Build message history for context
            const messageHistory = state.messages.slice(-10).map(msg => ({
                role: msg.role,
                content: msg.content
            }));

            // Add continuation prompt
            messageHistory.push({
                role: 'user',
                content: continuationPrompt
            });

            // Add system message with model instructions
            if (state.modelInstructions[state.currentModel]) {
                messageHistory.unshift({
                    role: 'system',
                    content: state.modelInstructions[state.currentModel]
```

```
    });
}

// Prepare API request
const modelEndpoint = state.modelMap[state.currentModel];
const requestOptions = {
    model: modelEndpoint,
    messages: messageHistory,
    max_tokens: state.isExtendedMode ? 16000 : 4000,
    temperature: 0.7,
    stream: true
};

// Create stream for chunked response
const response = await fetch('https://openrouter.ai/api/v1/chat/completions', {
    method: 'POST',
    headers: {
        'Authorization': `Bearer ${state.apiKey}`,
        'Content-Type': 'application/json'
    },
    body: JSON.stringify(requestOptions)
});

if (!response.ok) {
    throw new Error(`API responded with status: ${response.status}`);
}

const reader = response.body.getReader();
const decoder = new TextDecoder();
let responseText = previousContent + '\n\n';

while (true) {
    // Check if stopped
    if (state.isStopped) {
        addBotMessage(responseText, true, uiMode);
        break;
    }

    const { done, value } = await reader.read();

    if (done) {
        // Final complete message
        addBotMessage(responseText, true, uiMode);
        break;
```

```
                }

                // Decode the chunk and handle it
                const chunk = decoder.decode(value);

                // Parse SSE data
                const lines = chunk.split('\n');
                let newContent = '';

                for (const line of lines) {
                    if (line.startsWith('data:')) {
                        const data = line.slice(5).trim();
                        if (data === '[DONE]') continue;

                        try {
                            const parsed = JSON.parse(data);
                            if (parsed.choices && parsed.choices[0] && parsed.choices[0].delta &&
parsed.choices[0].delta.content) {
                                newContent += parsed.choices[0].delta.content;
                            }
                        } catch (e) {
                            console.error('Error parsing SSE chunk:', e);
                        }
                    }
                }

                if (newContent) {
                    responseText += newContent;
                    addBotMessage(responseText, false, uiMode);
                }
            }
        } catch (error) {
            console.error('API Error:', error);
            addBotMessage(`${previousContent}\n\nI'm sorry, I encountered an error continuing:
${error.message}`, true, uiMode);
        } finally {
            state.isGenerating = false;
        }
    }

    async function generateResearchContinuation(previousContent) {
        if (state.isGenerating) return;

        state.isGenerating = true;
```

```
        state.isStopped = false;

        // Start with loading message
        addBotMessage(`${previousContent}\n\n_Continuing..._`, false, 'research');

        try {
            // Prepare continuation prompt
            const continuationPrompt = `Continue exactly where you left off, without repeating anything. Pick up from:\n\n${previousContent.slice(-200)}`;

            // Build message history for context
            const messageHistory = state.researchMessages.slice(-10).map(msg => ({
                role: msg.role,
                content: msg.content
            }));

            // Add continuation prompt
            messageHistory.push({
                role: 'user',
                content: continuationPrompt
            });

            // Add system message with model instructions
            if (state.modelInstructions[state.currentResearchModel]) {
                messageHistory.unshift({
                    role: 'system',
                    content: state.modelInstructions[state.currentResearchModel]
                });
            }

            // Prepare API request
            const modelEndpoint = state.modelMap[state.currentResearchModel];
            const requestOptions = {
                model: modelEndpoint,
                messages: messageHistory,
                max_tokens: state.isExtendedMode ? 16000 : 4000,
                temperature: 0.7,
                stream: true
            };

            // Create stream for chunked response
            const response = await fetch('https://openrouter.ai/api/v1/chat/completions', {
                method: 'POST',
                headers: {
```

```
        'Authorization': `Bearer ${state.apiKey}`,
        'Content-Type': 'application/json'
    },
    body: JSON.stringify(requestOptions)
});

if (!response.ok) {
    throw new Error(`API responded with status: ${response.status}`);
}

const reader = response.body.getReader();
const decoder = new TextDecoder();
let responseText = previousContent + '\n\n';

while (true) {
    // Check if stopped
    if (state.isStopped) {
        addBotMessage(responseText, true, 'research');
        break;
    }

    const { done, value } = await reader.read();

    if (done) {
        // Final complete message
        addBotMessage(responseText, true, 'research');
        break;
    }

    // Decode the chunk and handle it
    const chunk = decoder.decode(value);

    // Parse SSE data
    const lines = chunk.split('\n');
    let newContent = '';

    for (const line of lines) {
        if (line.startsWith('data:')) {
            const data = line.slice(5).trim();
            if (data === '[DONE]') continue;

            try {
                const parsed = JSON.parse(data);
```

```
                        if (parsed.choices && parsed.choices[0] && parsed.choices[0].delta &&
parsed.choices[0].delta.content) {
                            newContent += parsed.choices[0].delta.content;
                        }
                    } catch (e) {
                        console.error('Error parsing SSE chunk:', e);
                    }
                }
            }

            if (newContent) {
                responseText += newContent;
                addBotMessage(responseText, false, 'research');
            }
        }
    } catch (error) {
        console.error('API Error:', error);
        addBotMessage(`${previousContent}\n\nI'm sorry, I encountered an error continuing:
${error.message}`, true, 'research');
    } finally {
        state.isGenerating = false;
    }
}

// Chat Cognito button demo examples
chatCognitoBtn.addEventListener('click', () => {
    if (state.isGenerating) return;

    const demoQuestions = [
        "Can you explain how quantum computing works?",
        "Write a short story about a robot discovering emotions.",
        "What are the key features of JavaScript ES6?",
        "How can I improve my productivity while working from home?",
        "Explain the concept of climate change and its impacts."
    ];

    const randomQuestion = demoQuestions[Math.floor(Math.random() *
demoQuestions.length)];

    if (state.currentMode === 'standard') {
        messageInput.value = randomQuestion;
        sendStandardMessage();
    } else if (state.currentMode === 'modern') {
        modernMessageInput.value = randomQuestion;
```

```
            sendModernMessage();
        }
    });

    // Simple InCOGNITO mode encryption (for demo purposes)
    function encryptMessage(message) {
        // Basic Caesar cipher for demo
        return message.split('').map(char => {
            const code = char.charCodeAt(0);
            // Shift letters by 5 places
            if (code >= 65 && code <= 90) {
                return String.fromCharCode(((code - 65 + 5) % 26) + 65);
            } else if (code >= 97 && code <= 122) {
                return String.fromCharCode(((code - 97 + 5) % 26) + 97);
            }
            return char;
        }).join('');
    }

    // Auto-resize textarea for all inputs
    function setupTextareaResize(textarea) {
        textarea.addEventListener('input', function() {
            this.style.height = 'auto';
            this.style.height = (this.scrollHeight) + 'px';
        });
    }

    setupTextareaResize(messageInput);
    setupTextareaResize(modernMessageInput);
    setupTextareaResize(researchMessageInput);

    // Request notification permission
    if (Notification.permission !== "granted" && Notification.permission !== "denied") {
        Notification.requestPermission();
    }

    // Page visibility change detection for notifications
    document.addEventListener('visibilitychange', function() {
        if (!document.hidden && state.isGenerating) {
            // User returned to the page while generation is happening
            showNotification('Generation in Progress', 'COGNITO is still generating a response for
you.');
        }
    });
```

```
        // Initialize with welcome screen
        usernameInput.focus();
    </script>
</body>
</html>
```