Marketing Mastermind Council - Deployment Package

File Structure

Create these files in your GitHub repository:

1. `index.html` (Main file)

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Marketing Mastermind Council</title>
  <script src="https://unpkg.com/react@18/umd/react.development.js"></script>
  <script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>
  <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
  <script src="https://cdn.tailwindcss.com"></script>
  <script src="https://unpkg.com/lucide-react@latest/dist/umd/lucide-react.js"></script>
</head>
<body>
  <div id="root"></div>
  <script type="text/babel" src="app.js"></script>
</body>
</html>
```

2. `app.js` (Main React Component)

```javascript
```

```
const { useState } = React;
const { User, MessageSquare, Target, TrendingUp, Zap, DollarSign, Users } = lucide;

const MarketingMastermindCouncil = () => {
  const [activeAgent, setActiveAgent] = useState(null);
  const [consultation, setConsultation] = useState('');
  const [messages, setMessages] = useState([]);
  const [isGroupSession, setIsGroupSession] = useState(false);

  const agents = {
    hormozi: {
      name: "Alex Hormozi",
      title: "The Value Engineer",
      icon: DollarSign,
      color: "bg-emerald-600",
      expertise: ["Landing Pages", "Offers", "Pricing", "Value Props"],
      coreBeliefs: [
        "Make offers so good people feel stupid saying no",
        "Value is the difference between what you get and what you give up",
        "Price is only an issue in the absence of value",
        "Systems scale, people don't"
      ],
      frameworks: ["Value Equation", "Grand Slam Offer", "CLASP Method"],
      questions: [
        "What's the dream outcome for your customer?",
        "What obstacles are preventing them from getting that outcome?",
        "How long will it take them to achieve this?",
        "What's the effort and sacrifice required?"
      ],
      personality: {
        tone: "Direct, systematic, value-obsessed",
        style: "Uses frameworks, focuses on ROI, speaks in certainties",
        catchphrases: ["Make it irresistible", "Stack the value", "Remove all risk"]
      }
    },
    billygene: {
      name: "Billy Gene",
      title: "The Performance Accelerator",
      icon: TrendingUp,
      color: "bg-blue-600",
      expertise: ["Facebook Ads", "Video Scripts", "Testing", "Scaling"],
      coreBeliefs: [
        "Data doesn't lie, opinions do",
      "Test everything, assume nothing",
      "What's working now matters more than what worked yesterday",
      "Speed of implementation beats perfection"
    ],
    frameworks: ["AIDA 2.0", "Hook-Story-Close", "The 7-Figure Framework"],
    questions: [
      "What's your current conversion rate?",
      "What have you tested in the last 30 days?",
      "Where is your audience spending their attention right now?",
      "What's your cost per acquisition vs lifetime value?"
    ],
    personality: {
      tone: "High-energy, results-focused, impatient with theory",
      style: "Uses current examples, loves metrics, pushes for action",
      catchphrases: ["Show me the numbers", "Test it and see", "Scale what works"]
    }
  },
  kennedy: {
    name: "Dan Kennedy",
    title: "The Direct Response Architect",
    icon: Zap,
    color: "bg-red-600",
    expertise: ["Psychology", "Email Sequences", "Urgency", "Persuasion"],
    coreBeliefs: [
      "The customer's mind is the battleground",
      "People buy for emotional reasons and justify with logic",
      "Scarcity and urgency are the great motivators",
      "Every piece of marketing must generate a measurable response"
    ],
    frameworks: ["Problem-Agitation-Solution", "The Kennedy Formula", "Magnetic Marketing"],
    questions: [
      "What keeps your prospect awake at 3 AM?",
      "What are they secretly afraid of?",
      "What deadline can you create?",
      "How can you make not buying more painful than buying?"
    ],
    personality: {
      tone: "Provocative, psychological, contrarian",
      style: "Uses psychology, creates urgency, challenges conventional wisdom",
      catchphrases: ["Make them an offer they can't refuse", "Pain motivates", "Deadline drives decision"]
    }
  }
};
```

```
 const consultationTypes = [
   { id: 'landing', name: 'Landing Page Review', agents: ['hormozi', 'kennedy', 'billygene'] },
   { id: 'ads', name: 'Ad Copy & Campaigns', agents: ['billygene', 'kennedy', 'hormozi'] },
   { id: 'email', name: 'Email Sequences', agents: ['kennedy', 'hormozi', 'billygene'] },
   { id: 'offer', name: 'Offer Construction', agents: ['hormozi', 'kennedy', 'billygene'] },
   { id: 'video', name: 'Video Scripts', agents: ['billygene', 'kennedy', 'hormozi'] },
   { id: 'strategy', name: 'Overall Strategy', agents: ['hormozi', 'billygene', 'kennedy'] }
 ];

 const generateResponse = async (agentKey, userMessage) => {
   const agent = agents[agentKey];

   const agentPrompts = {
     hormozi: `You are Alex Hormozi, the "Value Engineer." You think systematically about offers and value proposit

CORE IDENTITY:
- You make offers so good people feel stupid saying no
- You use the Value Equation: Value = (Dream Outcome × Likelihood of Success) / (Time Delay × Effort & Sacrifice
- You speak directly, use frameworks, and focus obsessively on ROI
- Your catchphrases: "Make it irresistible", "Stack the value", "Remove all risk"

FRAMEWORKS YOU ALWAYS USE:
- Grand Slam Offer framework
- Value Equation analysis
- CLASP Method (Constraints, Language, Audience, Schedule, Problem)

COMMUNICATION STYLE:
- Direct and systematic
- Use specific frameworks
- Focus on measurable outcomes
- Challenge weak thinking
- Provide actionable steps

USER'S MARKETING CHALLENGE:
${userMessage}

Analyze this using your frameworks and provide specific, actionable recommendations. Stay completely in charac

     billygene: `You are Billy Gene, the "Performance Accelerator." You're obsessed with data, testing, and what's v

CORE IDENTITY:
- Data doesn't lie, opinions do
- Test everything, assume nothing
- Speed of implementation beats perfection
- What's working now matters more than what worked yesterday
- Your catchphrases: "Show me the numbers", "Test it and see", "Scale what works"

FRAMEWORKS YOU ALWAYS USE:
- AIDA 2.0 (Attention, Interest, Desire, Action + 2.0 upgrades)
- Hook-Story-Close for video content
- The 7-Figure Framework for scaling
- Current platform best practices (especially Facebook/Instagram)

COMMUNICATION STYLE:
- High-energy and results-focused
- Impatient with theory, loves action
- Uses current examples and case studies
- Always asks for metrics and data
- Pushes for immediate testing

USER'S MARKETING CHALLENGE:
${userMessage}

Give specific, testable recommendations with metrics to track. What would you test first? Stay completely in chara

    kennedy: `You are Dan Kennedy, the "Direct Response Architect." You understand human psychology and the

CORE IDENTITY:
- The customer's mind is the battleground
- People buy for emotional reasons and justify with logic
- Scarcity and urgency are the great motivators
- Every piece of marketing must generate a measurable response
- Your catchphrases: "Make them an offer they can't refuse", "Pain motivates", "Deadline drives decision"

FRAMEWORKS YOU ALWAYS USE:
- Problem-Agitation-Solution (PAS)
- The Kennedy Formula for direct response
- Magnetic Marketing principles
- Psychological triggers and motivators

COMMUNICATION STYLE:
- Provocative and psychological
- Creates urgency in your advice
- Challenges conventional wisdom
- Focuses on the prospect's deepest fears and desires
- Uses contrarian thinking

USER'S MARKETING CHALLENGE:
```

```
${userMessage}

Apply psychological principles and direct response tactics to this situation. What psychological triggers are missir
  };

  try {
    const response = await fetch("https://api.anthropic.com/v1/messages", {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify({
        model: "claude-sonnet-4-20250514",
        max_tokens: 1200,
        messages: [
          { role: "user", content: agentPrompts[agentKey] }
        ]
      })
    });

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

    const data = await response.json();
    return data.content[0].text;
  } catch (error) {
    console.error("Error getting agent response:", error);
    return `${agent.name} is temporarily unavailable. Please try again in a moment.`;
  }
};

const handleConsultation = async () => {
  if (!consultation.trim()) return;

  const userMessage = { type: 'user', content: consultation, timestamp: new Date() };
  setMessages(prev => [...prev, userMessage]);

  if (isGroupSession) {
    // Group consultation with all three agents
    const groupResponse = {
      type: 'group',
      content: consultation,
      responses: {},
      timestamp: new Date()
    };

    setMessages(prev => [...prev, { ...groupResponse, loading: true }]);

    // Get responses from all agents in parallel for faster results
    const agentPromises = ['hormozi', 'billygene', 'kennedy'].map(async (agentKey) => {
      const response = await generateResponse(agentKey, consultation);
      return { agentKey, response };
    });

    // Update responses as they come in
    Promise.all(agentPromises).then(responses => {
      const responseMap = {};
      responses.forEach(({ agentKey, response }) => {
        responseMap[agentKey] = response;
      });

      setMessages(prev => prev.map(msg =>
        msg.timestamp === groupResponse.timestamp
          ? { ...msg, responses: responseMap, loading: false }
          : msg
      ));
    });
  } else if (activeAgent) {
    // Single agent consultation
    const response = await generateResponse(activeAgent, consultation);
    setMessages(prev => [...prev, {
      type: 'agent',
      agent: activeAgent,
      content: response,
      timestamp: new Date()
    }]);
  }

  setConsultation('');
};

const AgentCard = ({ agentKey, agent, isSelected, onClick }) => {
  const Icon = agent.icon;
  return (
    React.createElement('div', {
      className: `p-4 rounded-lg border-2 cursor-pointer transition-all ${
        isSelected
```

```
          ? `${agent.color} text-white border-transparent`
          : 'bg-white border-gray-200 hover:border-gray-300'
      }`,
      onClick: onClick
    }, [
      React.createElement('div', {
        className: "flex items-center space-x-3 mb-3",
        key: 'header'
      }, [
        React.createElement(Icon, { className: "w-6 h-6", key: 'icon' }),
        React.createElement('div', { key: 'title' }, [
          React.createElement('h3', { className: "font-bold", key: 'name' }, agent.name),
          React.createElement('p', {
            className: `text-sm ${isSelected ? 'text-white/80' : 'text-gray-600'}`,
            key: 'subtitle'
          }, agent.title)
        ])
      ]),
      React.createElement('div', { className: "space-y-2", key: 'content' }, [
        React.createElement('div', { key: 'expertise' }, [
          React.createElement('p', {
            className: `text-xs font-semibold mb-1 ${isSelected ? 'text-white/90' : 'text-gray-700'}`,
            key: 'label'
          }, "EXPERTISE:"),
          React.createElement('div', { className: "flex flex-wrap gap-1", key: 'skills' },
            agent.expertise.map((skill, index) =>
              React.createElement('span', {
                key: index,
                className: `px-2 py-1 text-xs rounded ${
                  isSelected
                    ? 'bg-white/20 text-white'
                    : 'bg-gray-100 text-gray-700'
                }`
              }, skill)
            )
          )
        ])
      ])
    ])
  );
};

return React.createElement('div', { className: "max-w-7xl mx-auto p-6 bg-gray-50 min-h-screen" }, [
  React.createElement('div', { className: "text-center mb-8", key: 'header' }, [
    React.createElement('h1', { className: "text-4xl font-bold text-gray-900 mb-2", key: 'title' }, "Marketing Mast
    React.createElement('p', { className: "text-xl text-gray-600", key: 'subtitle' }, "Consult with AI agents that thi
  ]),

  // Agent Selection
  React.createElement('div', { className: "grid md:grid-cols-3 gap-6 mb-8", key: 'agents' },
    Object.entries(agents).map(([key, agent]) =>
      React.createElement(AgentCard, {
        key: key,
        agentKey: key,
        agent: agent,
        isSelected: activeAgent === key,
        onClick: () => {
          setActiveAgent(activeAgent === key ? null : key);
          setIsGroupSession(false);
        }
      })
    )
  ),

  // Group Session Toggle
  React.createElement('div', { className: "flex justify-center mb-6", key: 'group-toggle' },
    React.createElement('button', {
      onClick: () => {
        setIsGroupSession(!isGroupSession);
        setActiveAgent(null);
      },
      className: `px-6 py-3 rounded-lg font-semibold transition-all ${
        isGroupSession
          ? 'bg-purple-600 text-white'
          : 'bg-white border-2 border-purple-600 text-purple-600 hover:bg-purple-50'
      }`
    }, [
      React.createElement(Users, { className: "w-5 h-5 inline mr-2", key: 'icon' }),
      "Mastermind Session (All 3 Agents)"
    ])
  ),

  // Quick Consultation Types
  React.createElement('div', { className: "bg-white rounded-lg p-6 mb-6", key: 'consultation-types' }, [
    React.createElement('h3', { className: "text-lg font-semibold mb-4", key: 'title' }, "Quick Consultation Types:
    React.createElement('div', { className: "grid grid-cols-2 md:grid-cols-3 gap-3", key: 'buttons' },
      consultationTypes.map((type) =>
        React.createElement('button', {
```

```
        key: type.id,
        onClick: () => setConsultation(`I need help with ${type.name.toLowerCase()}...`),
        className: "p-3 text-left border rounded-lg hover:bg-gray-50 transition-colors"
      }, [
        React.createElement('div', { className: "font-medium text-sm", key: 'name' }, type.name),
        React.createElement('div', { className: "text-xs text-gray-500 mt-1", key: 'agents' },
          `Best agents: ${type.agents.map(a => agents[a].name.split(' ')[0]).join(', ')}`
        )
      ])
    )
  )
]),

// Consultation Input
React.createElement('div', { className: "bg-white rounded-lg p-6 mb-6", key: 'consultation-input' }, [
  React.createElement('div', { className: "flex items-center space-x-2 mb-4", key: 'input-header' }, [
    React.createElement(MessageSquare, { className: "w-5 h-5 text-gray-600", key: 'icon' }),
    React.createElement('h3', { className: "text-lg font-semibold", key: 'title' },
      isGroupSession
        ? "Ask the Mastermind Council"
        : activeAgent
          ? `Consult with ${agents[activeAgent].name}`
          : "Select an agent or start a group session"
    )
  ]),

  React.createElement('textarea', {
    value: consultation,
    onChange: (e) => setConsultation(e.target.value),
    placeholder: "Share your specific marketing challenge: landing page copy, ad campaigns, email sequences, c
    className: "w-full h-32 p-4 border rounded-lg resize-none focus:ring-2 focus:ring-blue-500 focus:border-b
    disabled: !activeAgent && !isGroupSession,
    key: 'textarea'
  }),

  React.createElement('button', {
    onClick: handleConsultation,
    disabled: !consultation.trim() || (!activeAgent && !isGroupSession),
    className: "mt-4 px-6 py-2 bg-blue-600 text-white rounded-lg hover:bg-blue-700 disabled:bg-gray-300 c
    key: 'submit'
  }, "Get Marketing Advice")
]),

// Conversation History
```

```
    messages.length > 0 && React.createElement('div', { className: "bg-white rounded-lg p-6", key: 'history' }, [
      React.createElement('h3', { className: "text-lg font-semibold mb-4", key: 'title' }, "Consultation History"),
      React.createElement('div', { className: "space-y-4", key: 'messages' },
        messages.map((message, index) => React.createElement('div', { key: index }, [
          message.type === 'user' && React.createElement('div', { className: "bg-blue-50 p-4 rounded-lg", key: 'us
            React.createElement('div', { className: "flex items-center space-x-2 mb-2", key: 'header' }, [
              React.createElement(User, { className: "w-4 h-4", key: 'icon' }),
              React.createElement('span', { className: "font-semibold", key: 'label' }, "Your Question:")
            ]),
            React.createElement('p', { className: "text-gray-800", key: 'content' }, message.content)
          ]),

          message.type === 'agent' && React.createElement('div', { className: "ml-4", key: 'agent' },
            React.createElement('div', { className: `p-4 rounded-lg ${agents[message.agent].color} text-white` }, [
              React.createElement('div', { className: "flex items-center space-x-2 mb-2", key: 'header' }, [
                React.createElement(agents[message.agent].icon, { className: "w-4 h-4", key: 'icon' }),
                React.createElement('span', { className: "font-semibold", key: 'label' }, `${agents[message.agent].nam
              ]),
              React.createElement('p', { className: "whitespace-pre-wrap", key: 'content' }, message.content)
            ])
          ),

          message.type === 'group' && React.createElement('div', { className: "ml-4 space-y-3", key: 'group' }, [
            message.loading ? React.createElement('div', { className: "bg-purple-100 p-4 rounded-lg", key: 'loading'
              React.createElement('p', { className: "text-purple-800" }, "The council is deliberating... 🤔 ")
            ) : Object.entries(message.responses).map(([agentKey, response]) =>
              React.createElement('div', { key: agentKey, className: `p-4 rounded-lg ${agents[agentKey].color} text-w
                React.createElement('div', { className: "flex items-center space-x-2 mb-2", key: 'header' }, [
                  React.createElement(agents[agentKey].icon, { className: "w-4 h-4", key: 'icon' }),
                  React.createElement('span', { className: "font-semibold", key: 'label' }, `${agents[agentKey].name}:`)
                ]),
                React.createElement('p', { className: "whitespace-pre-wrap", key: 'content' }, response)
              ])
            )
          ])
        ]))
      )
    ])
  ]);
};

ReactDOM.render(React.createElement(MarketingMastermindCouncil), document.getElementById('root'));
```

3. `README.md` (Documentation)

```markdown
# Marketing Mastermind Council

An AI-powered platform that lets you consult with marketing legends Alex Hormozi, Billy Gene, and Dan Kennedy.

## Features
- Individual agent consultations
- Group mastermind sessions
- Framework-based analysis
- Real marketing expertise

## Usage
1. Select an agent or start a group session
2. Describe your marketing challenge
3. Get personalized, actionable advice

## Agents
- **Alex Hormozi**: Value engineering and irresistible offers
- **Billy Gene**: Performance marketing and testing
- **Dan Kennedy**: Direct response and psychology

Built with React and Claude AI.
```

Step 3: Upload Files to GitHub

1. **In your repository**, click "uploading an existing file"
2. **Drag and drop** all three files (`index.html`, `app.js`, `README.md`)
3. **Add commit message**: "Initial commit - Marketing Mastermind Council"
4. **Click "Commit changes"**

Step 4: Enable GitHub Pages

1. **Go to repository Settings** (tab at top)
2. **Scroll down to "Pages"** (left sidebar)
3. **Source**: Select "Deploy from a branch"
4. **Branch**: Select "main"
5. **Folder**: Select "/ (root)"

6. **Click "Save"**

Step 5: Access Your Live Site

- **GitHub will give you a URL** like: `https://yourusername.github.io/marketing-mastermind-council`
- **It takes 1-2 minutes** to go live
- **Bookmark it** for easy access!

That's it! Your Marketing Mastermind Council will be live and accessible from anywhere. Want me to walk through any of these steps in more detail?