Phase I: Climate Anxiety Survey App - Local Storage MVP

1. Phase I Objectives

1.1 Core Goal

Build a fully functional decision tree survey application for climate anxiety assessment using local browser storage. Focus on perfecting the user experience, question flow logic, and interactive visualization before adding backend complexity.

1.2 Phase I Scope

- ✅ Complete survey functionality with all question types
- ✅ Interactive flow diagram showing decision tree
- ✅ Multi-select parallel path exploration
- ✅ Local storage for session persistence
- ✅ Reset/clear functionality to start over
- ✅ One survey per browser/user session
- ❌ No backend/database integration
- ❌ No data submission to external services

2. Technical Architecture

2.1 Tech Stack

- **Frontend**: Vite + React + TypeScript
- **Hosting**: GitHub Pages
- **Storage**: Browser localStorage only
- **Flow Visualization**: React Flow
- **Styling**: Tailwind CSS with climate-appropriate theme

2.2 Data Flow

```
Survey JSON → React State → localStorage → Flow Diagram
    ↑                                          ↓
User Interaction ← UI Components ← State Updates
```

3. Core Features

3.1 Survey Engine

- Load climate anxiety survey from JSON configuration
- Handle 4 question types: scale, binary, single_select, multi_select
- Complex branching logic based on previous answers
- Real-time flow diagram updates

3.2 Path Management

- **Single Path**: Binary/single-select continue existing path
- **Multi Path**: Multi-select creates parallel exploration paths
- **Path Navigation**: Click on flow nodes to revisit questions
- **Confirmation Dialogs**: Warn before clearing subsequent answers

3.3 Local Storage Features

- **Auto-save**: Continuous saving of responses to localStorage
- **Session Persistence**: Resume survey after browser close/refresh
- **Clear Data**: Reset button to start completely over
- **One Survey Limit**: Single survey instance per browser

4. Data Structures

4.1 localStorage Schema

```
typescript
```

```
interface LocalSurveyData {
  sessionId: string;
  surveyId: string;
  startedAt: string;
  lastUpdated: string;
  currentPaths: UserPath[];
  completedPaths: UserPath[];
  isCompleted: boolean;
  version: string; // For future migration compatibility
}

interface UserPath {
  pathId: string;
  responses: Record<string, QuestionResponse>;
  isActive: boolean;
  currentQuestionId?: string;
  createdAt: string;
}

interface QuestionResponse {
  questionId: string;
  questionText: string;
  questionType: 'scale' | 'binary' | 'single_select' | 'multi_select';
  selectedAnswers: string[];
  customAnswers?: string[];
  timestamp: string;
  nextQuestions: string[];
}
```

4.2 localStorage Keys

typescript

```
const STORAGE_KEYS = {
  SURVEY_DATA: 'climate_anxiety_survey_data',
  SESSION_ID: 'climate_anxiety_session_id',
  SURVEY_VERSION: 'climate_anxiety_survey_version'
} as const;
```

5. UI Components

5.1 Main Layout

typescript

```
const App: React.FC = () => {
  return (
    <div className="min-h-screen bg-slate-50">
      <SurveyHeader />
      <FlowDiagramSection />
      <QuestionSection />
      <NavigationControls />
    </div>
  );
};
```

5.2 Core Components

Survey Header

typescript

```
const SurveyHeader: React.FC = () => {
  const { clearSurvey, surveyData } = useSurveyContext();

  return (
    <header className="bg-white border-b border-slate-200 px-6 py-4">
      <div className="flex justify-between items-center">
        <div>
          <h1 className="text-2xl font-bold text-slate-900">
            {surveyData?.survey.title}
          </h1>
          <p className="text-slate-600 mt-1">
            {surveyData?.survey.description}
          </p>
        </div>
        <button
          onClick={clearSurvey}
          className="px-4 py-2 text-slate-600 hover:text-red-600"
        >
          Start Over
        </button>
      </div>
    </header>
  );
};
```

Question Display Components

typescript

```
// Scale Question (1-5 ratings)
const ScaleQuestion: React.FC<ScaleQuestionProps> = ({
  question,
```

```
  currentAnswer,
  onAnswer
}) => {
  return (
    <div className="space-y-6">
      <h2 className="text-xl font-semibold text-slate-900">
        {question.question}
      </h2>
      <div className="flex space-x-2">
        {question.scale?.labels.map((label, index) => (
          <button
            key={index}
            onClick={() => onAnswer([(index + 1).toString()])}
            className={`px-4 py-3 rounded-lg border-2 transition-all ${
              currentAnswer?.[0] === (index + 1).toString()
                ? 'border-teal-500 bg-teal-50 text-teal-900'
                : 'border-slate-200 hover:border-slate-300'
            }`}
          >
            <div className="text-sm font-medium">{index + 1}</div>
            <div className="text-xs text-slate-600">{label}</div>

          </button>
        ))}
      </div>
    </div>
  );
};

// Multi-Select Question with Custom Input
const MultiSelectQuestion: React.FC<MultiSelectQuestionProps> = ({
  question,
  currentAnswers,
  onAnswer
}) => {
  const [customInput, setCustomInput] = useState('');

  return (
    <div className="space-y-6">
      <h2 className="text-xl font-semibold text-slate-900">
```

```
<h2 className="text-xl font-semibold text-slate-900">
  {question.question}
</h2>
<div className="grid gap-3">
  {question.options?.map((option) => (
    <label key={option.value} className="flex items-center space-x-3">
      <input
        type="checkbox"
        checked={currentAnswers?.includes(option.value)}
        onChange={(e) => handleOptionToggle(option.value, e.target.checked)}
        className="w-4 h-4 text-teal-600"
      />
      <span className="text-slate-700">{option.label}</span>
    </label>
  ))}

  {question.allowCustom && (
    <div className="mt-4 space-y-2">
      <label className="flex items-center space-x-3">
        <input
          type="checkbox"

          checked={customInput.length > 0}
          readOnly
          className="w-4 h-4 text-teal-600"
        />
        <span className="text-slate-700">Other (please specify):</span>
      </label>
      <input
        type="text"
        value={customInput}
        onChange={(e) => setCustomInput(e.target.value)}
        maxLength={question.customLimit || 100}
        placeholder="Enter your custom response..."
        className="w-full px-3 py-2 border border-slate-300 rounded-md"
      />
      <div className="text-xs text-slate-500">
        {customInput.length}/{question.customLimit || 100} characters
      </div>
    </div>
```

```
        </div>
      )}
    </div>
  </div>
 );
};
```

6. State Management

6.1 Survey Context

typescript

```
interface SurveyContextType {
 surveyData: Survey | null;
 currentPaths: UserPath[];
 flowData: { nodes: Node[], edges: Edge[] };
 currentQuestionId: string | null;
 isLoading: boolean;
 error: string | null;

 // Actions
 answerQuestion: (questionId: string, answers: string[], customAnswers?: string[]) => void;
 navigateToQuestion: (questionId: string, pathId: string) => void;
 clearSurvey: () => void;
 loadSurvey: () => void;
}

const SurveyContext = createContext<SurveyContextType | null>(null);
```

6.2 Local Storage Manager

typescript

```
class LocalStorageManager {
  private static STORAGE_KEY = 'climate_anxiety_survey_data';

  static saveSurveyData(data: LocalSurveyData): void {
    try {
      localStorage.setItem(this.STORAGE_KEY, JSON.stringify(data));
```

```
    } catch (error) {
      console.error('Failed to save survey data:', error);
    }
  }

  static loadSurveyData(): LocalSurveyData | null {
    try {
      const data = localStorage.getItem(this.STORAGE_KEY);
      return data ? JSON.parse(data) : null;
    } catch (error) {
      console.error('Failed to load survey data:', error);
      return null;
    }
  }

  static clearSurveyData(): void {
    localStorage.removeItem(this.STORAGE_KEY);
  }

  static generateSessionId(): string {
    return `session_${Date.now()}_${Math.random().toString(36).substr(2, 9)}`;
  }
}
```

7. Flow Diagram Implementation

7.1 React Flow Integration

typescript

```
const SurveyFlowDiagram: React.FC = () => {
  const { flowData, navigateToQuestion } = useSurveyContext();

  const onNodeClick = useCallback((event: React.MouseEvent, node: Node) => {
    const questionId = node.id;
    const pathId = node.data?.pathId;

    if (questionId && pathId) {
      navigateToQuestion(questionId, pathId);
```

```
    }
  }, [navigateToQuestion]);

  return (
    <div className="h-64 bg-white border-b border-slate-200">
      <ReactFlow
        nodes={flowData.nodes}
        edges={flowData.edges}
        onNodeClick={onNodeClick}
        nodeTypes={customNodeTypes}
        edgeTypes={customEdgeTypes}
        fitView
        attributionPosition="bottom-left"
      >
        <Background />
        <Controls />
      </ReactFlow>
    </div>
  );
};
```

7.2 Custom Node Types

typescript

```
const QuestionNode: React.FC<NodeProps> = ({ data }) => {
  const isCurrentQuestion = data.isCurrent;
  const questionType = data.questionType;

  return (
    <div className={`px-3 py-2 rounded-lg border-2 cursor-pointer transition-all ${
      isCurrentQuestion
        ? 'border-teal-500 bg-teal-50 shadow-lg'
        : 'border-slate-300 bg-white hover:border-slate-400'
    }`}>
      <div className="text-sm font-medium text-slate-900">
        {data.questionId}
      </div>
      <div className="text-xs text-slate-600">
```

```
        {questionType}
      </div>
      {data.selectedAnswer && (
        <div className="text-xs text-teal-600 mt-1">
          {data.selectedAnswer}
        </div>
      )}
    </div>
  );
};

const customNodeTypes = {
  question: QuestionNode,
};
```

8. Navigation & Controls

8.1 Navigation Component

typescript

```
const NavigationControls: React.FC = () => {
  const {
    currentPaths,
    isCompleted,
    canGoBack,
    goBack,
    clearSurvey
  } = useSurveyContext();

  return (
    <div className="fixed bottom-0 left-0 right-0 bg-white border-t border-slate-200 px-6 py-4">
      <div className="flex justify-between items-center">
        <button
          onClick={goBack}
          disabled={!canGoBack}
          className="px-4 py-2 text-slate-600 hover:text-slate-900 disabled:opacity-50"
        >
          ⇐ Back
```

```
      </button>

      <div className="text-sm text-slate-600">
        {currentPaths.length} active path{currentPaths.length !== 1 ? 's' : ''}
      </div>

      {isCompleted ? (
        <div className="space-x-3">
          <span className="text-green-600 font-medium">Survey Complete!</span>
          <button
            onClick={clearSurvey}
            className="px-4 py-2 bg-teal-600 text-white rounded-md hover:bg-teal-700"
          >
            Start New Survey
          </button>
        </div>
      ) : (
        <button
          onClick={clearSurvey}
          className="px-4 py-2 text-red-600 hover:text-red-700"

        >
          Clear & Restart
        </button>
      )}
    </div>
  </div>
  );
};
```

9. Confirmation Dialogs

9.1 Answer Change Confirmation

typescript

```
const ConfirmationDialog: React.FC<ConfirmationDialogProps> = ({
  isOpen,
  title = "Confirm Answer Change",
  message = "Changing this answer will clear all subsequent responses. Continue?",
  onConfirm,
  onCancel
}) => {
  if (!isOpen) return null;

  return (
    <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center justify-center z-50">
      <div className="bg-white rounded-lg p-6 max-w-md mx-4">
        <h3 className="text-lg font-semibold text-slate-900 mb-3">
          {title}
        </h3>
        <p className="text-slate-600 mb-6">
          {message}
```

```
        </p>
        <div className="flex space-x-3 justify-end">
          <button
            onClick={onCancel}
            className="px-4 py-2 text-slate-600 hover:text-slate-900"
          >
            Cancel
          </button>
          <button
            onClick={onConfirm}
            className="px-4 py-2 bg-amber-600 text-white rounded-md hover:bg-amber-700"
          >
            Continue
          </button>
        </div>
      </div>
    </div>
  );
};
```

10. Error Handling & Edge Cases

10.1 localStorage Limitations

typescript

```
const useLocalStorageWithFallback = () => {
  const [isAvailable, setIsAvailable] = useState(true);

  useEffect(() => {
    try {
      const testKey = '__localStorage_test__';
      localStorage.setItem(testKey, 'test');
      localStorage.removeItem(testKey);
    } catch (error) {
      setIsAvailable(false);
      console.warn('localStorage not available, using in-memory storage');
    }
  }, []);
```

```
  return { isAvailable };
};
```

10.2 Data Migration

typescript

```
const migrateLegacyData = (data: any): LocalSurveyData => {
  // Handle future changes to data structure
  if (!data.version) {
    return {
      ...data,
      version: '1.0.0',
      sessionId: data.sessionId || LocalStorageManager.generateSessionId()
    };
  }
  return data;
};
```

11. Development Phases

Phase I.1: Basic Survey Engine (Week 1)

- ✅ Load survey JSON
- ✅ Display single questions
- ✅ Handle all 4 question types
- ✅ Basic localStorage save/load

Phase I.2: Flow Logic (Week 2)

- ✅ Implement branching logic
- ✅ Path management for single selections
- ✅ Basic flow diagram with React Flow

Phase I.3: Multi-Select Paths (Week 3)

- ✅ Parallel path creation
- ✅ Multi-path visualization
- ✅ Path synchronization

Phase I.4: Interactive Features (Week 4)

- ✅ Click navigation in flow diagram
- ✅ Confirmation dialogs
- ✅ Reset functionality
- ✅ Polish and testing

12. Testing Strategy

12.1 Core Functionality Tests

- Survey loading and parsing
- Question type rendering
- Answer validation
- localStorage persistence
- Path branching logic

12.2 User Experience Tests

- Browser refresh persistence
- Clear/reset functionality
- Flow diagram navigation
- Mobile responsiveness
- Accessibility compliance

13. Success Criteria for Phase I

13. Success Criteria for Phase I

- ✅ Complete climate anxiety survey functionality
- ✅ Smooth multi-select parallel path exploration
- ✅ Persistent storage across browser sessions
- ✅ Interactive flow diagram with click navigation
- ✅ Clean, responsive UI following design system

- ✅ Zero data loss during normal usage
- ✅ Ready for Phase II backend integration

14. Deployment

14.1 GitHub Pages Setup

yaml

```yaml
# .github/workflows/deploy.yml
name: Deploy to GitHub Pages
on:
  push:
    branches: [ main ]
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - uses: actions/setup-node@v2
      - run: npm ci
      - run: npm run build
      - uses: peaceiris/actions-gh-pages@v3
        with:
          github_token: ${{ secrets.GITHUB_TOKEN }}
          publish_dir: ./dist
```

14.2 Build Configuration