India Demographic Heatmap Visualization Tool

Product Requirements Document (PRD)

Version: 1.0
Date: January 10, 2026
Project: UIDAI Demographic Data Heatmap Visualization Platform
Status: Final
Prepared for: Hackathon Challenge

Executive Summary

The India Demographic Heatmap Visualization Tool is an interactive web-based platform designed to transform raw UIDAI demographic CSV data into intuitive geographic visualizations. This tool enables government officials, policy makers, researchers, and data analysts to upload demographic datasets and instantly visualize population distribution patterns across India at multiple geographic levels (State, District, Pincode) with powerful filtering and export capabilities.

Key Value Propositions

- **Instant Visualization:** Convert CSV demographic data to interactive heatmaps in seconds
- **Multi-level Analysis:** Drill-down from national view to district-level and pincode-level granularity
- **Smart Filtering:** Filter by date, geography, age groups, and population density
- **Actionable Insights:** Pre-built charts, statistics, and comparison tools
- **Easy Export:** Generate PDF reports, CSV files, and map images for presentations
- **Zero Server Dependency:** All processing happens in-browser for maximum privacy

---

1. Problem Analysis

Current State Challenges

1.1 Data Analysis Inefficiency

- Manual analysis of large CSV files (up to 2M records) is time-consuming
- No standardized visualization method exists
- Analysts spend hours manipulating spreadsheets instead of deriving insights

1.2 Lack of Geographic Context

- Raw demographic numbers don't reveal geographic patterns
- Decision makers cannot quickly identify high-density regions
- Policy targeting is difficult without location-based insights

1.3 Data Integration Complexity

- Multiple CSV files (split by ranges: 0-500K, 500K-1M, 1M-1.5M, 1.5M-2M records)
- Manual merging is error-prone
- No automatic validation or quality checks

1.4 Information Accessibility Gap

- Non-technical users struggle with data analysis tools
- Government officials lack proper visualization infrastructure
- Knowledge workers spend time learning tools instead of using data

1.5 Reporting Challenges

- Cannot quickly generate reports for stakeholder meetings
- Exporting filtered data to stakeholders is cumbersome
- Comparison analysis between time periods is difficult

Target Problem Statement

"Create a user-friendly platform where UIDAI demographic data can be uploaded, validated, and visualized across India's geography to enable quick insights for policy making, planning, and research."

---

2. Product Vision & Objectives

Vision Statement

"Democratize access to India's demographic insights by providing an intuitive, interactive visualization platform that transforms raw data into actionable geographic intelligence for decision makers and researchers."

Primary Objectives

Objective	Success Metric
Enable rapid data exploration	Load and visualize CSV in <3 seconds
Support multi-level analysis	Support drill-down: State → District → Pincode
Ensure data accuracy	Validate 100% of input records
Provide actionable insights	Generate 3+ types of visualizations per dataset
Maximize usability	95%+ user task completion rate in testing
Ensure accessibility	WCAG 2.1 AA compliance

---

3. Target Users & Personas

Primary User Personas

Persona 1: Government Data Analyst

- **Name:** Raj Kumar
- **Role:** Senior Data Analyst, Department of Health
- **Tech Level:** Intermediate
- **Goal:** Quickly identify population distribution patterns for resource allocation
- **Pain Points:** Spends hours in Excel, cannot drill-down geographically
- **Success:** Can visualize data in <2 minutes and generate insights

Persona 2: Policy Maker

- **Name:** Priya Singh
- **Role:** District Magistrate
- **Tech Level:** Basic
- **Goal:** Understand demographic trends for planning meetings
- **Pain Points:** Overwhelmed by raw numbers, needs simple visualizations
- **Success:** Can understand key trends from heatmap at a glance

Persona 3: Research Institution

- **Name:** Dr. Arun Desai
- **Role:** Demographer, Research Institute
- **Tech Level:** Advanced
- **Goal:** Analyze demographic trends, conduct comparative analysis
- **Pain Points:** Cannot easily aggregate/filter large datasets
- **Success:** Can export customized datasets for statistical analysis

Persona 4: Urban Planner

- **Name:** Neha Patel
- **Role:** City Development Authority
- **Tech Level:** Intermediate
- **Goal:** Identify areas for infrastructure development
- **Pain Points:** Cannot see geographic distribution of specific age groups
- **Success:** Can filter by age group and density to identify hotspots

User Statistics

- **Primary Users:** 50-100 government analysts and officials
- **Secondary Users:** 200-300 researchers, planners, NGOs
- **Geographic Reach:** All 28 states + 8 union territories
- **Use Frequency:** Daily to weekly for active users

---

4. Core Product Features

4.1 Data Import & Validation

Feature Overview

Users upload one or multiple CSV files containing demographic data. The system automatically validates data against schema requirements and provides detailed feedback.

Data Schema Requirements

Required Columns:

- `date` (DD-MM-YYYY format) - Reference date for demographic data
- `state` (String) - State or Union Territory name
- `district` (String) - District name within state
- `pincode` (6-digit numeric) - Indian postal code
- `demo_age_5_17` (Integer) - Population count, ages 5-17 years
- `demo_age_17_` (Integer) - Population count, age 17+ years

Example Record:
date,state,district,pincode,demo_age_5_17,demo_age_17_
01-03-2025,Uttar Pradesh,Gorakhpur,273213,49,529
01-03-2025,Karnataka,Bengaluru,560072,92,438

Validation Rules

Schema Validation:

- ✓ Column headers match exactly (case-sensitive)
- ✓ No required columns missing
- ✓ No unexpected additional columns (ignored if present)

Data Type Validation:

- ✓ `date`: Matches DD-MM-YYYY format exactly
- ✓ `pincode`: Exactly 6 numeric digits (000000-999999)
- ✓ `demo_age_5_17`, `demo_age_17_`: Non-negative integers (0-9,999,999)

Referential Integrity:

- ✓ State names match official Indian state list
- ✓ District names correspond to parent state
- ✓ Pincodes geographically match their district
- ✓ No null values in required fields

Data Quality Checks:

- ✓ Duplicate detection (same state, district, pincode, date combination)
- ✓ Outlier flagging (unusually high or low values)
- ✓ Date range consistency (no future dates beyond current date)

File Upload Specifications

Parameter	Specification
Supported Format	CSV (Comma-Separated Values)
Single File Limit	25 MB
Total Upload Limit	100 MB (for multiple files)
Records Per File	Up to 500,000 records
Accepted Encodings	UTF-8, UTF-16, ISO-8859-1
Line Endings	LF (Unix), CRLF (Windows), CR (Mac)
CSV Dialect	Standard (comma-delimited, quote-escaped)

Upload Workflow

1. User clicks "Upload CSV"
   ↓
2. File picker opens (multi-select enabled)
   ↓
3. User selects one or multiple CSV files
   ↓
4. Preview first 5 rows before upload
   ↓
5. User confirms upload
   ↓
6. Browser validates locally in real-time
   ↓
7. Show validation results:
   ├─ Success: "100 records validated ✓"
   ├─ Warnings: "Schema OK, 5 duplicate records detected"
   └─ Errors: "Date format invalid in row 42"
   ↓
8. If all valid → Store in IndexedDB
   ↓
9. Trigger automatic data aggregation
   ↓
10. Render heatmap with new data

Error Handling & Reporting

Validation Report Features:

- Downloadable error report (CSV or PDF)
- Line-by-line error details with row numbers
- Suggested fixes for common issues
- Validation summary statistics

Error Recovery:

- Option to skip invalid rows and proceed with valid data
- Partial import mode for majority-valid files
- Re-upload option to fix and resubmit

4.2 Interactive Geographic Heatmap

Heatmap Specifications

Map Library: Leaflet.js 1.9.x with OpenStreetMap base layer

Visualization Levels:

Level	Granularity	Use Cases
India (National)	Entire country view	Overview, national trends
State Level	28 states + 8 UTs	Regional comparisons
District Level	750+ districts	District-level planning
Pincode Level	1.5M+ postal zones	Neighborhood precision analysis

Color Intensity Scheme:

- **Minimum (Green):** 0 population
- **Low (Light Yellow):** 25th percentile
- **Medium (Orange):** 50th percentile (median)
- **High (Red-Orange):** 75th percentile
- **Maximum (Dark Red):** 100th percentile

Color Palettes (Selectable):

1. **Viridis** (Default) - Green to Yellow to Red (colorblind-friendly)
2. **Plasma** - Purple to Yellow (good contrast)
3. **Cool** - Blue gradient (calming)
4. **Warm** - Orange to Red (intensity-focused)
5. **Greyscale** - Gray gradient (print-friendly)

Interactive Features

Hover Tooltips:
Shows detailed information when hovering over any region:

```
┌──────────────────────────────────┐
│ Bengaluru, Karnataka             │
│ (Pincode: 560072)                │
│ ──────────────────────────────── │
│ Ages 5-17: 1,245                 │
│ Ages 17+: 8,923                  │
```

```
│ Total: 10,168 │
│ Date: 01-Mar-2025 │
│ Density: 45.2/km² │
└──────────────────────────────────┘
```

Click Interactions:

- Click on any region to drill-down to next level
- Automatic zoom to selected region
- Smooth animation (300ms transition)
- Breadcrumb trail shows navigation path
- "Back" button or breadcrumb to return to parent level

Drill-Down Navigation:

India (National) → Click State
↓
State View → Click District
↓
District View → Click Pincode
↓
Pincode-level Detail View

Breadcrumb Navigation:
Home > Uttar Pradesh > Gorakhpur > 273213
Users can click any breadcrumb level to jump to that view.

Legend & Scale:

- Color gradient legend showing value ranges
- Numeric scale (min to max population values)
- Indicate data collection date
- "More Info" link to data source attribution

Performance Targets:

- Map initial load: < 2 seconds
- Drill-down animation: < 300ms
- Hover tooltip display: < 100ms
- Pan/zoom smoothness: 60 FPS minimum

4.3 Advanced Filtering System

Filter Panel Components

4.3.1 Date Range Selector

- Calendar widget (start date → end date)
- Display only available dates from uploaded data
- Preset buttons:
  - "Today" - Current date only
  - "Last 7 Days" - Past week
  - "Last 30 Days" - Past month

    - "Last 90 Days" - Past quarter
    - "Year-to-Date" - Current year
    - "All Data" - Complete date range
- Visual indicator of date availability

4.3.2 Geographic Filter

- **State Selector:** Multi-select dropdown with 28 states + 8 UTs
- **District Selector:** Dynamically filtered based on selected states
- **Pincode Search:** Type to find specific pincode
- **Search:** Full-text search for state/district names
- **Clear All:** Reset all geographic filters

4.3.3 Age Group Filter

- Toggle buttons:
    - "Ages 5-17"
    - "Ages 17+"
    - "Both / Total"
    - "Ratio" (5-17 : 17+)

4.3.4 Population Density Filter

- Slider: Min to Max population range
- Input fields for precise values
- Dynamic range based on current data
- Show count of regions matching filter

4.3.5 Quick Filter Presets

- **Urban Hotspots:** Top 20% high-density areas
- **Rural Areas:** Bottom 20% low-density areas
- **Major Metro Zones:** Pre-defined metro boundaries
- **High Growth Areas:** Regions with significant date-to-date growth
- **Custom Saved:** User-saved filter combinations

Filter Application & Results

Real-Time Updates:

- Map updates instantly (< 500ms) as filters change
- Show "X regions match filters" counter
- Highlight/dim non-matching regions
- Smooth fade transitions

Filter Combinations:

- Multiple filters work in AND logic (all must match)
- Example: "State=Delhi AND Ages=5-17 AND Date Range=Last 30 Days"
- Visual indication of active filters (badges/chips)
- "Clear Filters" button to reset all

Filter Persistence:

- Save filter configurations as "Views"

- Load previous filter states
- Share filter URLs with colleagues
- Bookmark specific filtered views

4.4 Analytics Dashboard

Dashboard Widgets

Summary Statistics Card:

```
┌─────────────────────────────────────────┐
│ Demographic Summary                     │
├─────────────────────────────────────────┤
│ Total Population: 24,567,890            │
│ Regions Analyzed: 1,245                 │
│ Average per Region: 19,745              │
│ Data Last Updated: 01-Mar-2025          │
│ Data Quality: ✓ Validated               │
└─────────────────────────────────────────┘
```

Chart 1: Top 10 Districts by Population

- Horizontal bar chart
- Sortable, filterable
- Click to filter map to that district
- Show both age groups stacked

Chart 2: Age Group Distribution

- Pie chart showing 5-17 vs 17+ split
- Percentage breakdown
- Interactive legend (click to filter)

Chart 3: Population Trends Over Time

- Line chart if multiple dates in dataset
- Show trend lines for each age group
- Highlight significant changes
- Compare year-over-year if applicable

Chart 4: Geographic Distribution

- Box plot showing distribution across regions
- Outlier detection
- Quartile breakdown

Data Table:

- Sortable by any column
- Filterable search
- Pagination: 10/25/50/100 rows per page
- CSV export of visible table
- Columns: Date, State, District, Pincode, 5-17 Pop, 17+ Pop, Total, Density

Responsive Dashboard

- Desktop: 2-column layout with map on left, dashboard on right
- Tablet: Stacked layout with dashboard below map
- Mobile: Full-width, collapsible dashboard sections

4.5 Export & Reporting

Export Format Options

4.5.1 CSV Export

- Export filtered/visible data as CSV
- All records or selected records only
- Include metadata rows (date range, filters applied)
- Downloaded filename: `demographic_data_[state]_[date].csv`

4.5.2 PDF Report

- Professional multi-page PDF
- Page 1: Title page with metadata
  - Report title
  - Date range
  - Geographic scope
  - Filters applied
  - Generated timestamp
- Page 2-3: Heatmap visualization
  - Embedded map image
  - Legend and scale
  - Geographic boundaries marked
- Page 4: Summary Statistics
  - Key metrics in table format
  - Data completeness info
  - Quality assessment
- Page 5-6: Charts & Visualizations
  - Top regions chart
  - Age distribution pie chart
  - Trends (if available)
- Final Page: Data Table
  - Complete data in tabular format
  - Sorted by population (descending)
  - Data source attribution

4.5.3 Excel Export

- Multi-sheet workbook:
  - Sheet 1: Raw Data (with all records)
  - Sheet 2: Summary Statistics
  - Sheet 3: Aggregated by District
  - Sheet 4: Data Dictionary
- Features:
  - Column filtering enabled
  - Conditional formatting (color by value)

- Charts embedded as images
- Professional styling with company branding

4.5.4 Map Export (PNG/SVG)

- High-resolution map image (300 DPI for print)
- Include legend, scale, north arrow
- Option: Include/exclude region labels
- Option: Include/exclude data values
- Format: PNG for easy sharing, SVG for editing

4.5.5 Custom Report Builder

- Drag-and-drop sections to include
- Choose charts to include
- Add custom text/annotations
- Select color scheme
- Preview before export

Export Specifications

Format	File Size	Use Case	Settings
CSV	<5MB	Data analysis in Excel/R	Delimited, UTF-8
PDF	5-15MB	Printing, email, presentations	300 DPI, A4
XLSX	10-20MB	Data sharing with formulas	Multiple sheets
PNG	2-8MB	Sharing online, quick reference	300 DPI
SVG	<2MB	Web embedding, editing	Scalable

4.6 Comparison & Trend Analysis

Multi-Period Comparison

Side-by-Side View:

- Select two date ranges from uploaded data
- Display heatmaps side by side
- Synchronized zoom/pan
- Identical color scales for comparison

Difference Map:

- Third visualization showing change between periods
- Color gradient: Blue (decrease) → Gray (no change) → Red (increase)

- Magnitude proportional to percentage change
- Show % change in tooltips

Statistics Comparison:

- Table showing metrics for both periods:
  - Total population
  - Per-region average
  - Top regions in each period
  - Biggest changes

Trend Analysis

Requirements: Multiple dates in dataset

Time Series Features:

- Line chart showing population trends per district/state
- Multiple series (one per region) overlaid
- Toggle regions on/off
- Zoom into date range
- Show growth rate, CAGR, trend line

---

5. Technical Architecture

5.1 Tech Stack Recommendations

Frontend Framework

React 18.x

- Modern component architecture
- Hooks for state management
- Performance optimizations built-in
- Large ecosystem and community support

TypeScript

- Type safety reducing bugs
- Better IDE support
- Self-documenting code

Mapping & Geospatial

Leaflet.js 1.9.x

- Lightweight, widely-used map library
- Excellent mobile support
- Plugin ecosystem for heatmaps

Leaflet-Heat Plugin

- Native heatmap layer support
- Performance optimized

- Easy integration with Leaflet

TopoJSON Format

- Compressed geospatial data format
- India boundaries pre-converted
- ~200KB for complete India map data

Mapbox GL (Optional Enhancement)

- 3D visualization capabilities
- Advanced styling options
- Real-time data updates

Data Visualization

D3.js 7.x

- Advanced visualizations
- Full control over rendering
- Interactive features
- Steep learning curve but powerful

Recharts (Recommended for charts)

- React-native chart library
- Built on D3 fundamentals
- Easier API than D3
- Good performance

Plotly.js

- Alternative: Rich visualization library
- Interactive features built-in
- Good for statistical plots

Data Management & Storage

IndexedDB (Browser Storage)

- Client-side database
- Supports large datasets (>1GB)
- No server required
- Persistent across sessions
- Asynchronous API

Dexie.js

- Wrapper around IndexedDB
- Cleaner, promise-based API
- Query support
- Recommended approach

Apache Arrow / Parquet (Optional)

- Memory-efficient columnar storage
- Fast data access
- Compression support
- For very large datasets (1M+ records)

CSV Processing

PapaParse 5.x

- Fast CSV parsing
- Worker thread support for large files
- Error handling
- Streaming support

Export Functionality

jsPDF

- PDF generation in browser
- Add text, images, tables
- Supports multiple pages
- Good for simple to complex reports

html2canvas

- Convert DOM elements to images
- Captures map screenshots
- For PDF/PNG export

ExcelJS

- Create Excel workbooks programmatically
- Format, styling, formulas support
- Multiple sheet support

FileSaver.js

- Handle file downloads
- Browser compatibility wrapper

State Management

Zustand (Recommended)

- Simple, lightweight
- Minimal boilerplate
- Easy learning curve
- Good performance

Redux Toolkit (Alternative)

- More structured
- Better for large applications
- Mature ecosystem
- More boilerplate

UI Component Library

Radix UI

- Unstyled, accessible components
- Full customization control
- Keyboard navigation built-in
- ARIA attributes

shadcn/ui

- Copy-paste component library
- Built on Radix UI
- Tailwind CSS styling
- Full control over components

Styling

Tailwind CSS 3.x

- Utility-first approach
- Fast development
- Small CSS bundle
- Responsive design utilities

CSS Modules

- Component-scoped styles
- No naming conflicts
- Good for complex components

Build & Development

Vite 4.x

- Ultra-fast development server
- Fast build times
- ES modules support
- Better than Create React App

ESLint

- Code quality rules
- Catch errors early

Prettier

- Code formatting
- Team consistency

Vitest

- Fast unit testing
- Vite native
- Jest-compatible API

Cypress

- End-to-end testing
- Real browser automation
- Great debugging experience

Performance Optimization

React Query

- Data fetching and caching
- Stale-while-revalidate pattern
- Reduces unnecessary re-renders

Lazy Loading

- Code splitting by route
- Dynamic imports

Web Workers

- Offload CSV parsing to background
- Keep UI responsive

5.2 Application Architecture

Component Hierarchy

```
App (Root)
├── Layout
│   ├── Header
│   │   ├── Logo & Title
│   │   ├── Navigation Menu
│   │   └── Help/Docs Button
│   ├── Main Container (flex: 1)
│   │   ├── Sidebar (width: 320px)
│   │   │   ├── DataUploadPanel
│   │   │   │   ├── FileInput
│   │   │   │   ├── UploadButton
│   │   │   │   └── ValidationReport
│   │   │   ├── FilterPanel
│   │   │   │   ├── DateRangeFilter
│   │   │   │   ├── GeographicFilter
│   │   │   │   ├── AgeGroupFilter
│   │   │   │   ├── PopulationFilter
│   │   │   │   ├── QuickFilters
│   │   │   │   ├── SavedViews
│   │   │   │   └── ApplyFiltersButton
│   │   │   └── ExportPanel
│   │   │   ├── CSVExportBtn
│   │   │   ├── PDFExportBtn
│   │   │   ├── ExcelExportBtn
│   │   │   └── MapImageBtn
│   │   └── Main Content (flex: 1)
```

```
│ │ ├── MapContainer
│ │ │ ├── MapViewer (Leaflet)
│ │ │ ├── HeatmapLayer
│ │ │ ├── Legend
│ │ │ ├── Tooltip
│ │ │ ├── Breadcrumb
│ │ │ └── LoadingSpinner
│ │ └── DashboardArea
│ │ ├── SummaryStatsCard
│ │ ├── ChartGrid
│ │ │ ├── BarChart (Top Regions)
│ │ │ ├── PieChart (Age Distribution)
│ │ │ ├── LineChart (Trends)
│ │ │ └── BoxPlot (Distribution)
│ │ └── DataTable
│ │ ├── TableHeader
│ │ ├── TableBody (virtualized)
│ │ ├── Pagination
│ │ └── SearchBar
│ └── Footer
│ ├── DataSource Attribution
│ ├── Version Info
│ └── Contact/Support Links
└── Modal Overlays
├── FileValidationReport
├── ExportProgress
├── ComparisonView
├── SettingsModal
└── HelpModal
```

Data Flow Architecture

```
User Input Events (Upload, Filter)
↓
Action Handlers
↓
Zustand Store (State Management)
├── dataStore
│ └── { records, metadata, validationStatus }
├── filterStore
│ └── { dateRange, states, districts, ageGroup, populate }
└── uiStore
└── { currentLevel, breadcrumb, loading }
↓
Selectors & Derived State
├── Filtered records
├── Aggregated by geography
├── Color scale calculation
└── Chart data transformation
↓
Component Re-render
```

├── MapViewer (re-render heatmap)
├── Charts (re-render with new data)
├── DataTable (update visible rows)
└── SummaryStats (update metrics)
↓
Browser Rendering (React Reconciliation)
↓
Leaflet/D3 Visualization Updates

Data Storage Layer

IndexedDB Schema:

```
-- Store: demographics
{
keyPath: "id",
indexes: [
{ name: "date" },
{ name: "state" },
{ name: "district" },
{ name: "pincode" },
{ compound: ["state", "district"] },
{ compound: ["date", "state"] }
]
}

-- Store: metadata
{
keyPath: "id",
data: {
uploadedFiles: [],
totalRecords: number,
dateRange: { min, max },
states: string[],
validationStats: {}
}
}

-- Store: geodata
{
keyPath: "level",
data: {
features: GeoJSON FeatureCollection
}
}
```

5.3 Performance Optimization Strategies

1. Data Processing

CSV Parsing:

- Use Web Worker for parsing (offload to background thread)
- Chunk processing for large files
- Progress feedback to user

Data Aggregation:

- Pre-calculate aggregations at upload time
- Store at multiple levels (pincode, district, state)
- Use memoization to avoid recalculation

2. Rendering Optimization

React Optimization:

- Use React.memo for expensive components
- useMemo for derived state
- useCallback for event handlers
- Lazy load components with React.lazy()

Virtual Scrolling:

- For data table with 10K+ rows
- Use react-window or react-virtualized
- Only render visible rows

Heatmap Optimization:

- Canvas rendering for heatmap layer
- Debounce zoom/pan events
- Limit tooltip updates to 60fps

3. Storage Optimization

IndexedDB:

- Create indexes on frequently queried fields
- Compress large datasets
- Implement data pagination queries
- Use generators for streaming large results

Browser Cache:

- Cache geodata (rarely changes)
- Cache computed aggregations
- Use localStorage for user preferences

4. Network (if backend added later)

- Gzip compression
- Image optimization
- Code splitting
- Lazy loading of features

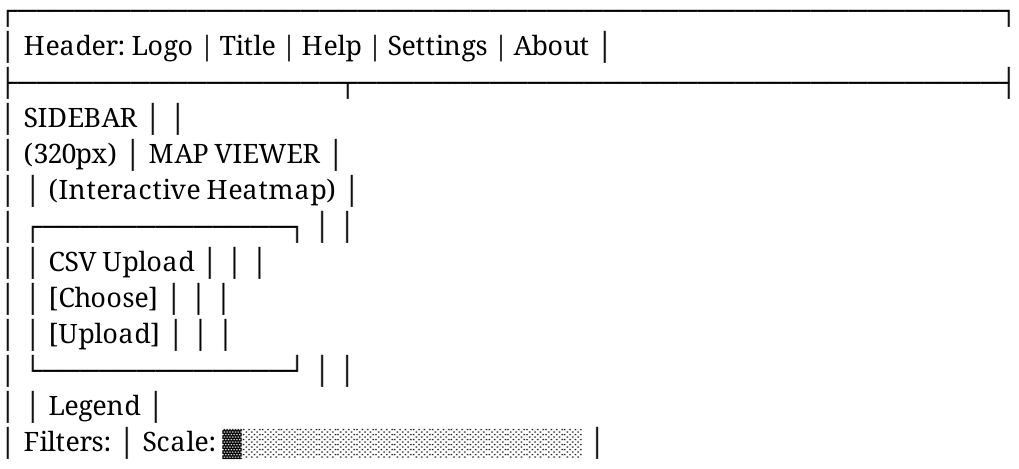
---

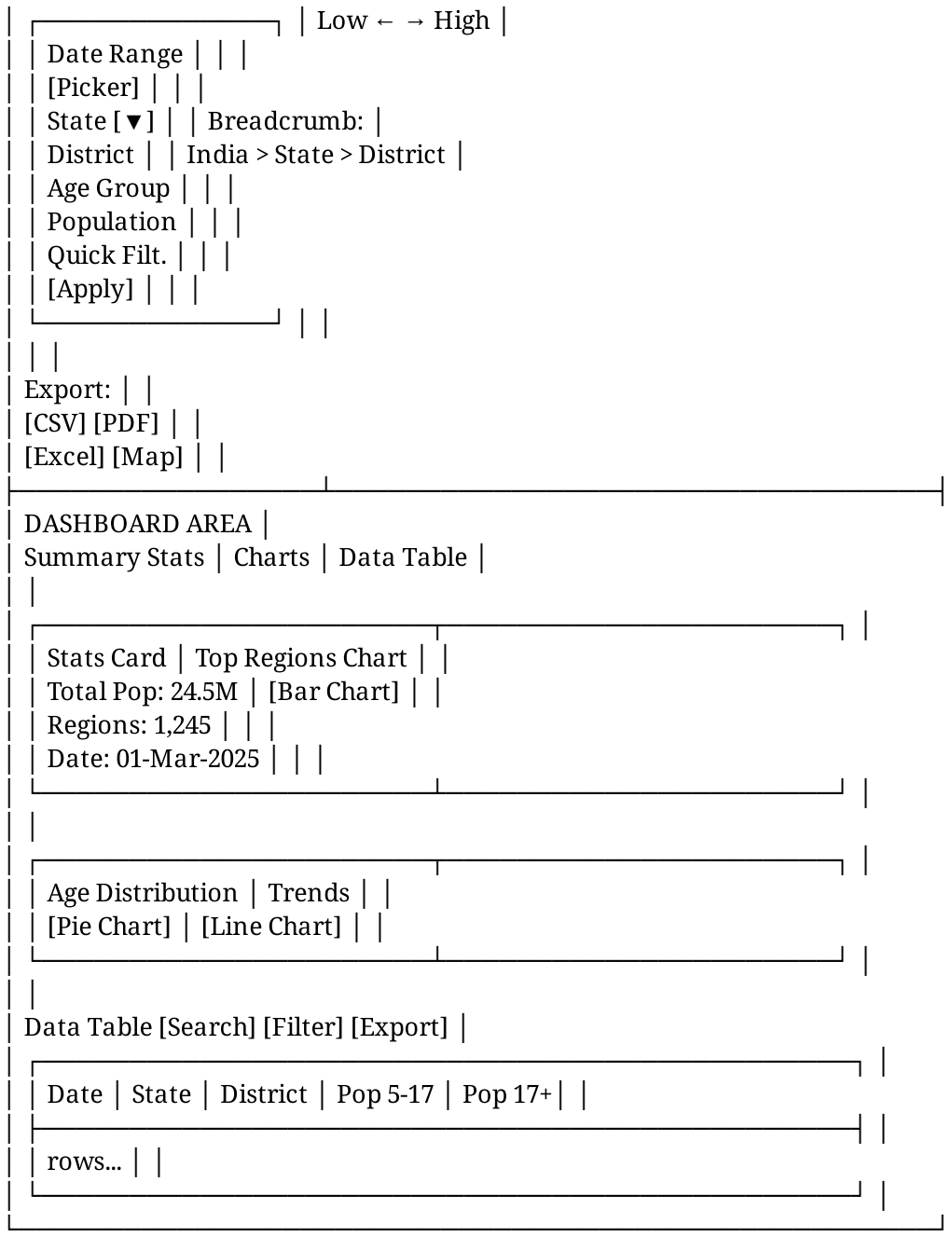
6. User Experience Design

6.1 Design Principles

1. **Clarity Over Complexity**
   - Show only essential information
   - Progressive disclosure (details on demand)
   - Consistent visual language
2. **Responsiveness**
   - Real-time feedback for actions
   - Loading states and progress indicators
   - Error messages that suggest fixes
3. **Accessibility**
   - WCAG 2.1 AA compliance minimum
   - Keyboard navigation throughout
   - Screen reader friendly
   - High contrast (4.5:1 ratio)
   - Colorblind-friendly palettes
4. **Data Literacy**
   - Tooltips explaining metrics
   - Educational onboarding
   - "What's This?" help icons
   - Legends and explanations
5. **Efficiency**
   - Keyboard shortcuts for power users
   - Quick access to common operations
   - Batch operations support
   - Smart defaults

6.2 Wireframe Layout





6.3 Color Palette

Brand Colors:

- Primary: #1F7A8C (Teal) - Actions, highlights
- Secondary: #F18F01 (Orange) - Warnings, alerts
- Success: #06A77D (Green) - Positive feedback
- Error: #D62828 (Red) - Errors, warnings
- Neutral: #CCCCCC to #333333 - Text, borders, backgrounds

Heatmap Colors (Viridis - Colorblind Friendly):

- Low: #440154 (Dark Purple)
- → #31688E (Blue)

- → #35B779 (Green)
- → #FDE724 (Yellow)
- High: #FFD000 (Bright Yellow)

Semantic Colors:

- Highlight: #FFE135
- Muted: #999999
- Disabled: #CCCCCC

6.4 Typography

- **Font Family:** Inter or System UI fonts
- **Headings:** Bold, 24-32px (H1), 20-24px (H2), 16-20px (H3)
- **Body:** Regular, 14-16px
- **Monospace:** Courier New / Monaco (data values, pincodes)

6.5 Spacing & Layout

- **Grid:** 8px baseline grid
- **Card Padding:** 16px
- **Section Margins:** 24px
- **Mobile Viewport:** 360px minimum width
- **Tablet Breakpoint:** 768px
- **Desktop Breakpoint:** 1024px

---

7. Use Cases & User Workflows

Use Case 1: Government Analyst Analyzing Regional Demographics

Scenario: A public health analyst needs to identify regions with high child population (ages 5-17) for immunization program planning.

Workflow:

1. Open application
2. Upload February 2025 demographic CSV
3. Validation: "50,000 records validated ✓"
4. View India heatmap (national overview)
5. Filter: Age Group = "5-17"
6. Observe: Northern regions show higher child population
7. Click on Uttar Pradesh state
8. Drill-down: View district-level distribution
9. Identify top 5 districts by child population
10. Filter by: "High Density" + "Ages 5-17"
11. Export as: PDF report for ministry meeting
12. Presentation: Show heatmap + summary statistics
13. Decision: Allocate resources to identified districts

Success Criteria:

- ✓ CSV processed in <3 seconds

- ✓ Heatmap renders without lag
- ✓ Filter updates instant
- ✓ Can drill-down to district level
- ✓ PDF export includes key charts and data
- ✓ All data is accurate

Use Case 2: Urban Planner Comparing Growth Patterns

Scenario: A city development authority wants to compare population growth from January to February 2025 in metro areas.

Workflow:

1. Upload January 2025 demographic data
2. Upload February 2025 demographic data
3. Activate "Comparison Mode"
4. Select Period 1: Jan 1-31, 2025
5. Select Period 2: Feb 1-28, 2025
6. Display: Side-by-side heatmaps + difference map
7. Observe: Red regions = growth, Blue = decline
8. Identify: Bangalore, Mumbai, Delhi showing high growth
9. View statistics: Growth rates per city
10. Export: Compare report with growth analysis
11. Zoom: Focus on specific metro areas
12. Planning: Identify high-growth neighborhoods for investment

Success Criteria:

- ✓ Side-by-side comparison is clear
- ✓ Difference calculation is accurate
- ✓ Growth metrics properly calculated
- ✓ Report highlights significant changes
- ✓ Can zoom into specific regions

Use Case 3: Researcher Analyzing Demographic Trends

Scenario: A demographer at research institute studies population distribution patterns across quarters.

Workflow:

1. Upload Q1 2025 data (3 months)
2. Select: "View Trends Over Time"
3. Display: Line chart showing population trajectory
4. Select: Multiple districts for comparison
5. Analyze: CAGR, growth rates, trends
6. Filter: By age group and geography
7. Export: Excel with calculated metrics
8. Statistical analysis: Use exported data in R/Python
9. Publication: Data insights for research paper
10. Visualization: Include charts in academic presentation

Success Criteria:

- ✓ Trends display with multiple date points
- ✓ CAGR correctly calculated
- ✓ Multi-series comparison works
- ✓ Excel export is analysis-ready
- ✓ Time series charts are publication-quality

---

8. Non-Functional Requirements

8.1 Performance Requirements

Metric	Target	Measurement
Initial Load Time	< 2s	Time to interactive
CSV Parse (500K records)	< 5s	Browser time
Filter Application	< 500ms	Heatmap update
Drill-Down Animation	< 300ms	Smooth 60fps
Hover Tooltip	< 100ms	Display time
PDF Export (10K rows)	< 10s	Generation time
Export File Size	< 50MB	Reasonable download

8.2 Scalability

- **Data Volume:** Support up to 2M demographic records
- **Concurrent Features:** 50 UI elements without lag
- **Storage:** Browser IndexedDB limit (typically 50GB+)
- **Responsive:** Mobile to 4K displays

8.3 Browser Support

Browser	Version	Support
Chrome	90+	Full support
Firefox	88+	Full support
Safari	14+	Full support
Edge	90+	Full support
Mobile Chrome	90+	Full support
Mobile Safari	14+	Full support

8.4 Accessibility (WCAG 2.1)

- **Level AA Compliance** (minimum)
- Color contrast: 4.5:1 for normal text, 3:1 for large text
- Keyboard navigation: All features accessible via keyboard
- Screen reader: Compatible with NVDA, JAWS, VoiceOver
- Focus indicators: Always visible
- Alternative text: For all images

8.5 Security

- **Data Privacy:** All processing client-side (no server)
- **Input Validation:** All CSV data validated before use
- **XSS Prevention:** Sanitize all user inputs
- **Content Security Policy:** Strict CSP headers
- **HTTPS Only:** If served over network

8.6 Reliability

- **Uptime:** N/A for client-side app
- **Data Integrity:** Validation ensures data quality
- **Error Recovery:** Graceful error handling
- **Offline Support:** Fully functional offline (after first load)

---

9. MVP vs. Phase 2+ Features

MVP Features (Release 1.0)

✅ **Included in MVP:**

- Single/multiple CSV upload with validation
- Data schema validation (100% accuracy)
- Basic heatmap visualization (national + state level)
- Date range filtering
- Geographic filtering (state/district)
- Age group toggling
- Summary statistics (total, average, count)
- CSV export functionality
- Responsive design (mobile/tablet/desktop)
- WCAG AA accessibility
- Data quality reporting

❌ **Not in MVP:**

- Pincode-level visualization
- Comparison mode
- Trend analysis / time series
- Advanced charts (D3.js complex viz)
- PDF export with styling
- Excel export with formatting
- Map image export
- Saved filter presets

- Custom color schemes
- Advanced analytics (anomaly detection)

Phase 2 Features (Release 1.1-2.0)

Q2 2026:

- Pincode-level drill-down
- Comparison mode (multi-period analysis)
- Advanced time-series charts
- PDF report generation
- Excel export with embedded charts
- Map image export (PNG/SVG)
- Saved view configurations

Q3 2026:

- Custom color scheme selection
- Advanced filtering (ranges, regex)
- Statistical analysis tools
- Anomaly detection
- Growth rate calculations
- Demographic projections

Q4 2026+:

- Backend API (database storage)
- Real-time data ingestion
- Multi-user collaboration
- Comments/annotations
- Advanced sharing features
- Mobile app (React Native)
- API for 3rd-party integration

---

10. Success Metrics & KPIs

User Adoption Metrics

- Users uploading data: >70% of active users
- CSV validation success rate: >95%
- Average session duration: >15 minutes
- Returning users (weekly): >60%
- Feature completion rate: >80% of workflows

Data Quality Metrics

- Data validation pass rate: >95%
- Duplicate detection accuracy: 100%
- Field validation error rate: <1%
- Data integrity issues: <0.5%

Performance Metrics

- Page load time: <2 seconds
- Filter response time: <500ms
- Export generation: <10 seconds
- Crash rate: <0.1%

User Satisfaction

- NPS Score (Net Promoter Score): >50
- User satisfaction (1-5): >4.2/5
- Task completion rate: >85%
- Support tickets: <2% of users

---

11. Risk Management

Risk	Probability	Impact	Mitigation
Browser storage exceeds limit	Low	High	Implement data chunking, compression
Large CSV crashes browser	Medium	High	Web Worker processing, progress updates
Map rendering performance	Medium	Medium	Canvas optimization, data aggregation
Geospatial data inaccuracy	Low	Medium	Validate against official boundaries
User confusion with navigation	Medium	Medium	Clear breadcrumbs, tooltips, help docs
Data privacy concerns	Low	High	Client-side processing, transparent policy
Mobile usability issues	Medium	Medium	Comprehensive mobile testing
Accessibility compliance gaps	Low	Medium	WCAG audit, screen reader testing

---

12. Timeline & Milestones

Phase 1: Design & Setup (Week 1-2)

- Finalize design system
- Set up development environment
- Create component library
- Establish coding standards

Phase 2: Core Development (Week 3-6)

- Implement CSV upload/validation
- Build map visualization
- Create filter system
- Develop dashboard

Phase 3: Export & Polish (Week 7-9)

- Implement export functionality
- Responsive design refinement
- Performance optimization
- Accessibility audit

Phase 4: Testing & Launch (Week 10-13)

- Comprehensive testing (unit, integration, E2E)
- User acceptance testing
- Bug fixes and polish
- Production deployment

---

13. Appendix: Technical Resources

Libraries & Documentation

Mapping:

- Leaflet.js: https://leafletjs.com
- Leaflet-Heatmap: https://github.com/Leaflet/Leaflet.heat
- TopoJSON: https://github.com/topojson/topojson

Data Processing:

- PapaParse: https://www.papaparse.com
- Dexie.js: https://dexie.org

Visualization:

- D3.js: https://d3js.org
- Recharts: https://recharts.org

UI:

- Radix UI: https://www.radix-ui.com

- shadcn/ui: https://ui.shadcn.com
- Tailwind CSS: https://tailwindcss.com

Export:

- jsPDF: https://github.com/parallax/jsPDF
- ExcelJS: https://github.com/exceljs/exceljs

Sample Data Resources

- UIDAI Open Data: https://uidai.gov.in/
- Government Data Portal: https://data.gov.in/
- Census Data: https://censusindia.gov.in/

Code Repository Structure

```
india-demographic-heatmap/
├── public/
│   ├── index.html
│   ├── favicon.ico
│   └── geodata/
│   ├── india-states.json
│   ├── india-districts.json
│   └── india-boundaries.topojson
├── src/
│   ├── components/
│   │   ├── Map/
│   │   ├── Filters/
│   │   ├── Dashboard/
│   │   └── Export/
│   ├── hooks/
│   │   ├── useMapData.ts
│   │   ├── useFilters.ts
│   │   └── useExport.ts
│   ├── store/
│   │   ├── dataStore.ts
│   │   ├── filterStore.ts
│   │   └── uiStore.ts
│   ├── utils/
│   │   ├── csvParser.ts
│   │   ├── geoProcessor.ts
│   │   └── colorScale.ts
│   ├── types/
│   │   ├── data.ts
│   │   ├── geo.ts
│   │   └── ui.ts
│   └── App.tsx
├── tests/
│   ├── unit/
│   ├── integration/
│   └── e2e/
├── vite.config.ts
```

```
├── tsconfig.json
├── tailwind.config.js
└── package.json
```

Document Version: 1.0
Last Updated: January 10, 2026
Status: Ready for Development
Contact: [Project Lead Email]