Solar Mirror Optimizer - Technical Specification for AI Agent

Project Overview

Project Name: Solar Mirror Optimizer with Real-time Sun Position Visualization

Primary Goal: Build a web-based visualization tool that displays the sun's position throughout the day with a 2D animation showing a house at the center and the sun moving along its arc path. The system will calculate precise solar angles to later optimize mirror positioning for a solar panel booster system.

Target User: Electrical engineering student working on solar energy optimization project in Ho Chi Minh City, Vietnam.

Platform: Desktop application running on macOS (M3 MacBook), web-based interface accessed via browser.

System Architecture

Technology Stack

Backend:

- Language: Python 3.9+
- Framework: Flask (lightweight web server)
- Solar calculations: pvlib-python library
- Time handling: pandas, pytz
- Math operations: numpy

Frontend:

- HTML5 Canvas for 2D graphics
- Vanilla JavaScript (no frameworks)
- CSS3 for styling
- No external JavaScript libraries initially (pure implementation)

Development Environment:

- macOS with Python 3.9+
- Modern web browser (Chrome/Safari)
- Text editor or IDE (VS Code recommended)

Project Structure

```
solar-mirror-optimizer/
│
├── backend/
│   ├── app.py                  # Flask application entry point
│   ├── solar_calculator.py     # Core solar position calculations
│   ├── config.py               # Configuration (location, timezone)
│   └── requirements.txt        # Python dependencies
│
├── frontend/
│   ├── index.html              # Main HTML page
│   ├── css/
│   │   └── style.css           # All styling
│   ├── js/
│   │   ├── canvas.js           # Canvas drawing functions
│   │   ├── animation.js        # Animation loop and controls
│   │   └── api.js              # Backend API communication
│   └── assets/
│       └── (icons will be simple shapes, no images initially)
│
├── tests/
│   └── test_solar_calculator.py  # Unit tests for accuracy
│
└── README.md                   # Setup instructions
```

Functional Requirements

Core Features (MVP - Minimum Viable Product)

1. **Solar Position Calculation**

   - Calculate sun elevation angle (degrees above horizon)
   - Calculate sun azimuth angle (degrees from North, clockwise)
   - Use pvlib-python SPA algorithm for ±0.01° accuracy
   - Location: Ho Chi Minh City (10.8231°N, 106.6297°E, altitude 19m)
   - Timezone: Asia/Ho_Chi_Minh (UTC+7)

2. **2D Canvas Visualization**
   - Canvas size: 800x600 pixels (responsive later)
   - Center point: House icon at (400, 400)
   - Sun path: Semi-circular arc from East (90°) to West (270°)
   - Arc radius: 250 pixels from center
   - Background: Light blue for day, dark blue for night
3. **Visual Elements**
   - House: Simple rectangle or triangle at center (50x50px)
   - Sun: Yellow circle (diameter 30px) moving along arc
   - Sun path: Dashed arc line showing full trajectory
   - Cardinal directions: N, E, S, W labels at edges
   - Grid lines: Optional compass lines every 45°
4. **Information Display**
   - Current time: HH:MM:SS format
   - Solar elevation: "Elevation: XX.XX°"
   - Solar azimuth: "Azimuth: XXX.XX°"
   - Sunrise time: "Sunrise: HH:MM"
   - Sunset time: "Sunset: HH:MM"
   - Current date: YYYY-MM-DD
5. **Animation Controls**
   - Play/Pause button
   - Time slider (00:00 to 23:59)

  - Speed control: 1x, 10x, 60x, 300x (real-time to 5 min/sec)
  - Date picker (to simulate different days)
  - Reset button (go to current time)

Backend API Endpoints

Endpoint 1: Get Current Sun Position

```
GET /api/sun-position
Response: {
  "timestamp": "2025-11-11T10:30:00+07:00",
  "elevation": 45.23,
  "azimuth": 135.67,
  "zenith": 44.77,
  "is_daytime": true
}
```

Endpoint 2: Get Sun Position at Specific Time

```
GET /api/sun-position?datetime=2025-11-11T14:00:00
Response: (same format as above)
```

Endpoint 3: Get Full Day Sun Path

```
GET /api/sun-path?date=2025-11-11&interval=60
Parameters:
  - date: YYYY-MM-DD
  - interval: minutes between data points (default 60)
Response: {
  "date": "2025-11-11",
  "sunrise": "05:48",
  "sunset": "17:29",
  "solar_noon": "11:38",
  "path": [
    {"time": "06:00", "elevation": 5.2, "azimuth": 88.5},
    {"time": "07:00", "elevation": 15.8, "azimuth": 95.2},
    ... (24 data points for interval=60)
  ]
}
```

Endpoint 4: Serve Static Files

```
GET / → index.html
GET /css/* → CSS files
GET /js/* → JavaScript files
```

Technical Specifications

Canvas Coordinate System

Coordinate Mapping:

- Canvas origin (0,0) = top-left
- House center = (400, 400) in canvas coordinates
- Y-axis inverted: higher Y = lower on screen

Sun Position Calculation:

```
Given: elevation (α), azimuth (β)

Canvas coordinates:
- x = centerX + radius * sin(azimuth_rad) * cos(elevation_rad)
- y = centerY - radius * cos(azimuth_rad) * cos(elevation_rad)

Where:
- centerX = 400
- centerY = 400
- radius = 250
- azimuth_rad = (azimuth - 90) * π/180  (adjust for East=0)
- elevation_rad = elevation * π/180
```

Arc Path Drawing:

- Start angle: 0° (East, right side)
- End angle: 180° (West, left side)
- Arc is only drawn above horizon (elevation > 0)

- Use dashed line style

Animation Implementation

Animation Loop:

```
let currentTime = new Date();
let isPlaying = false;
let speed = 1; // real-time multiplier

function animate() {
  if (isPlaying) {
    currentTime += (1000 * speed); // advance time
    fetchSunPosition(currentTime);
    drawVisualization();
  }
  requestAnimationFrame(animate);
}
```

Frame Rate:

- Target: 30 FPS minimum
- Update sun position: every frame if animating, or on user input
- API calls: throttled to max 1 per second during animation

Data Flow

1. **Initial Load:**
    - Frontend loads HTML/CSS/JS
    - JavaScript calls `/api/sun-position` for current time
    - JavaScript calls `/api/sun-path` for full day data
    - Draw initial visualization
2. **Animation Loop:**
    - Increment internal time variable
    - Interpolate sun position from cached path data

   - Redraw canvas with new position
   - Update text displays

3. **User Interaction:**
   - Slider drag: update time, pause animation, fetch new position
   - Play button: start animation loop
   - Date picker: fetch new day data, reset animation

Calculation Details

Solar Position Algorithm:

- Use pvlib.location.Location class
- Method: spa_python() for highest accuracy
- Atmospheric refraction: included
- Equation of time: handled by pvlib

Input Parameters:

```
latitude = 10.8231    # Ho Chi Minh City
longitude = 106.6297
altitude = 19         # meters above sea level
timezone = 'Asia/Ho_Chi_Minh'
```

Edge Cases to Handle:

- Sun below horizon (elevation < 0): don't draw sun, show night mode
- Midnight: handle date transition
- Leap years: handled by datetime
- DST: Vietnam doesn't have DST, but timezone library handles it

Visual Design Specifications

Color Scheme

Day Mode (sun elevation > 0):

- Background: #87CEEB (sky blue)
- Ground: #90EE90 (light green)
- House: #8B4513 (brown)
- Sun: #FFD700 (gold) with #FFA500 (orange) glow
- Path line: #FFFFFF (white) dashed, 50% opacity
- Text: #000000 (black)

Night Mode (sun elevation ≤ 0):

- Background: #191970 (midnight blue)
- Ground: #2F4F4F (dark slate gray)
- House: #696969 (dim gray)
- Moon: #F0E68C (khaki) - optional
- Text: #FFFFFF (white)

Layout

```
+------------------------------------------+
|  Solar Position Visualizer               |
|  [Date Picker] [Time: 10:30:45]         |
+------------------------------------------+
|                                          |
|              N                           |
|              |                           |
|         W ---🏠--- E                     |
|              |                           |
|              S       ☀️                  |
|          (Canvas 800x600)                |
|                                          |
+------------------------------------------+
|  Elevation: 45.23°  |  Azimuth: 135.67° |
|  Sunrise: 05:48     |  Sunset: 17:29    |
+------------------------------------------+
|  [◀️] [⏸️] [▶️]    Speed: [1x] [10x] [60x] |
|  Time: [===============|==============]   |
|        00:00                       23:59  |
+------------------------------------------+
```

Development Milestones

Milestone 1: Backend Foundation (Day 1, 4 hours)

Deliverables:

- Flask app runs on localhost:5000
- `/api/sun-position` returns accurate data
- `/api/sun-path` returns array of positions
- Validated against NOAA calculator (< 0.1° error)

Acceptance Criteria:

- API accessible via curl or browser
- JSON response well-formatted
- Calculations match reference within tolerance

Milestone 2: Static Visualization (Day 1, 4 hours)

Deliverables:

- Canvas draws house at center
- Canvas draws sun at current position
- Canvas draws full sun path arc
- Text displays current angles
- Cardinal directions labeled

Acceptance Criteria:

- Visual matches design spec
- Sun position visually correct (East at sunrise, West at sunset)
- All text readable and accurate

Milestone 3: Animation (Day 2, 4 hours)

Deliverables:

- Play/pause button functional
- Time slider updates visualization
- Sun moves smoothly along path
- Time display updates in sync

Acceptance Criteria:

- Animation smooth (≥30 FPS)
- No jitter or jumps
- Controls responsive

Milestone 4: Polish & Testing (Day 2, 4 hours)

Deliverables:

- Speed controls working
- Date picker functional
- Responsive design (basic)
- Error handling (no internet, invalid input)
- Documentation updated

Acceptance Criteria:

- All features work as specified
- No console errors
- Clean code, commented

Testing Requirements

Unit Tests (Backend)

```
def test_solar_position_accuracy():
    # Compare with NOAA calculator reference values
    # Test cases: sunrise, noon, sunset, midnight
    # Assert error < 0.1 degrees
```

```
def test_edge_cases():
    # Test midnight boundary
    # Test southern hemisphere (future)
    # Test extreme latitudes (future)
```

Manual Tests (Frontend)

- [ ] Sun rises in East (~90° azimuth)
- [ ] Sun sets in West (~270° azimuth)
- [ ] Sun highest at solar noon
- [ ] Night mode triggers correctly
- [ ] All buttons respond within 100ms
- [ ] Animation smooth at all speeds
- [ ] Date picker changes visualization

Integration Tests

- [ ] API responds within 500ms
- [ ] Frontend handles API errors gracefully
- [ ] Time sync correct between backend and frontend

Dependencies

Python (requirements.txt):

```
flask==3.0.0
flask-cors==4.0.0
pvlib==0.10.3
pandas==2.1.3
numpy==1.26.2
pytz==2023.3
```

JavaScript:

- None (vanilla JS only)

System:

- Python 3.9+
- Modern browser with Canvas support

Configuration

config.py:

```
# Location settings
LATITUDE = 10.8231
LONGITUDE = 106.6297
ALTITUDE = 19
TIMEZONE = 'Asia/Ho_Chi_Minh'

# Visualization settings
CANVAS_WIDTH = 800
CANVAS_HEIGHT = 600
SUN_PATH_RADIUS = 250
UPDATE_INTERVAL = 60  # seconds for data points

# Animation settings
DEFAULT_SPEED = 1
MAX_SPEED = 300
FPS_TARGET = 30
```

Success Criteria

The MVP is complete when:

1. User can open browser and see live sun position
2. Animation shows sun moving across sky smoothly
3. All displayed angles match NOAA calculator within 0.1°
4. User can scrub through any time of day
5. System runs stable for 1+ hour without issues

Future Enhancements (Not in MVP)

- Mirror position calculator
- 3D visualization option
- Multiple location support
- Historical data playback
- Export animation as video
- Mobile responsive design
- Dark mode toggle
- Multiple language support
- Cloud deployment

Notes for AI Agent

- Prioritize accuracy of calculations over UI polish
- Keep code simple and readable
- Comment all coordinate transformations
- Use meaningful variable names
- Handle edge cases gracefully
- Log errors to console for debugging
- Make constants configurable
- Write modular functions (single responsibility)