Technical Documentation - RainfallFussion

Table of Contents

1. Architecture Overview
2. Code Structure
3. Core Functions
4. Data Flow
5. Algorithm Details
6. Variable Reference
7. Extending the Code

Architecture Overview

Design Pattern

RainfallFussion menggunakan **Monolithic Web Application** dengan pattern:

- **Presentational Layer:** HTML + CSS (UI)
- **Business Logic Layer:** Vanilla JavaScript (Processing)
- **Data Layer:** In-memory state (dataFiles, imputedData)

Technology Stack

```
┌─────────────────────────────────────────┐
│        Browser (Client-Side)            │
├─────────────────────────────────────────┤
│  HTML5         │  CSS3        │  JS ES6 │
├─────────────────────────────────────────┤
│  External Libraries (CDN)               │
│  - XLSX.js (2.8.5)                      │
│  - Chart.js (3.9.1)                     │
│  - JSZip (3.10.1)                       │
└─────────────────────────────────────────┘
```

Data Flow Diagram

```
User Input (Coordinates)
        ↓
Upload Files (XLSX)
        ↓
Parse Excel → dataFiles {target, ref1, ref2}
        ↓
Calculate Correlations & Distances
        ↓
```

```
Imputation Ensemble Processing
  - Normal Ratio Method
  - Inverse Distance Weighting
  - Linear Regression
  - Correlation Weighting
        ↓
Voting Ensemble → imputedData
        ↓
Validation Metrics Calculation
        ↓
Visualization & Export
        ↓
Download Excel Results
```

Code Structure

File Organization

```
index.html (1239 lines)
├── <head> (lines 1-10)
│   ├── Meta tags
│   ├── Title
│   └── External libraries
├── <style> (lines 11-307)
│   ├── Global styles
│   ├── Component styles
│   └── Responsive design
├── <body> (lines 308-350)
│   └── HTML structure
└── <script> (lines 351-1239)
    ├── Global variables
    ├── Initialization
    ├── File handling
    ├── Data processing
    ├── Imputation algorithms
    ├── Validation metrics
    ├── Visualization
    └── Export functions
```

HTML Structure (lines 308-350)

```
<div class="container">
  <h1>Judul</h1>

  <div class="coordinate-section">
    <!-- Input koordinat 3 stasiun -->
  </div>
```

```
    <div class="upload-grid">
      <!-- Upload file 3 stasiun -->
    </div>

    <div class="button-center">
      <!-- Tombol Process & Download -->
    </div>

    <div id="progress" class="progress-hidden">
      <!-- Progress bar -->
    </div>

    <div id="metrics" class="metrics-hidden">
      <!-- Metric cards -->
    </div>

    <div id="distanceInfo">
      <!-- Distance information -->
    </div>

    <table id="foTable" class="fo-table">
      <!-- Fill-out locations table -->
    </table>

    <div class="chart-container">
      <!-- Chart canvas -->
    </div>
  </div>
```

Core Functions

1. File Input & Parsing

`readExcelFile(file, station)`

Purpose: Membaca file Excel dan mengubah ke format JSON

Parameters:

- `file` (File): File object dari input
- `station` (string): Tipe stasiun ("target", "ref1", "ref2")

Logic:

```
1. Create FileReader
2. Read file as ArrayBuffer
3. Parse dengan XLSX.read()
4. Convert sheet to JSON array
5. Store di dataFiles[station]
6. Check apakah semua file loaded
```

Return: void (update global dataFiles)

Example:

```
const inputElement = document.getElementById('targetFile');
inputElement.addEventListener('change', (e) => {
    readExcelFile(e.target.files[0], 'target');
});
```

`parseData(jsonData)`

Purpose: Transform raw Excel array menjadi struktur data standar

Parameters:

- `jsonData` (array): Output dari XLSX.utils.sheet_to_json

Structure Output:

```
[
  {
    year: 2020,
    day: 1,
    months: [12.5, 15.3, 20.1, ..., 14.2]  // 12 elemen
  },
  ...
]
```

Logic:

```
1. Loop dari row 1 (skip header)
2. Extract year (col 0) dan day (col 1)
3. Convert 12 bulan (col 2-13) ke numbers
4. Ganti null/empty dengan null value
5. Return array of structured data
```

---

2. Coordinate & Distance

`saveCoordinate(station)`

Purpose: Simpan koordinat stasiun

Parameters:

- `station` (string): "target", "ref1", atau "ref2"

Logic:

```
1. Get latitude & longitude dari input
2. Validate dalam rentang yang tepat
3. Store di global coordinates object
4. Calculate distances (if all coords saved)
5. Display distance info
```

`haversineDistance(lat1, lon1, lat2, lon2)`

Purpose: Hitung jarak geografis antar stasiun

Parameters:

- `lat1, lon1` (number): Koordinat stasiun 1
- `lat2, lon2` (number): Koordinat stasiun 2

Formula:

```
R = 6371 (Earth radius in km)
Δlat = lat2 - lat1 (converted to radians)
Δlon = lon2 - lon1 (converted to radians)

a = sin²(Δlat/2) + cos(lat1) × cos(lat2) × sin²(Δlon/2)
c = 2 × atan2(√a, √(1-a))
distance = R × c
```

Return: number (distance in km)

Example:

```
const dist = haversineDistance(-7.5515, 110.8243, -7.5600, 110.8100);
// Returns: 8.547 (km)
```

---

3. Data Processing & Imputation

`calculateCorrelation()`

Purpose: Hitung korelasi Pearson antara target dan referensi

Parameters: None (menggunakan global dataFiles)

Output:

```
{
    ref1: 0.876,  // Pearson coefficient
    ref2: 0.824
}
```

Algorithm - Pearson Correlation:

```
r = (n×ΣXY - ΣX×ΣY) / √[(n×ΣX² - (ΣX)²) × (n×ΣY² - (ΣY)²)]

Dimana:
- n = jumlah data points
- X = nilai target
- Y = nilai referensi
```

`multiStationVotingEnsemble(targetData, refData1, refData2, distances, correlation)`

Purpose: Algoritma utama untuk imputasi menggunakan voting ensemble

Parameters:

- `targetData` (array): Data stasiun target
- `refData1, refData2` (array): Data referensi
- `distances` (object): {ref1: km, ref2: km}
- `correlation` (object): {ref1: r, ref2: r}

Main Logic:

```
for each data point i {
    for each month m {
        if targetData[i].months[m] is null {
            predictions = []

            // Method 1: Normal Ratio
            nr = normalRatioMethod(...)
            if (nr >= 0) predictions.push(nr)

            // Method 2: IDW
            idw = inverseDistanceWeighting(...)
            if (idw >= 0) predictions.push(idw)

            // Method 3: Linear Regression
            lr = linearRegressionMethod(...)
            if (lr >= 0) predictions.push(lr)

            // Method 4: Correlation Weighting
            cw = correlationWeighting(...)
            if (cw >= 0) predictions.push(cw)
```

```
            // Voting Ensemble
            if (predictions.length > 0) {
                result[i].months[m] = average(predictions)
            } else {
                result[i].months[m] = 0
            }
        }
    }
}
```

Return: array (imputed data dengan struktur sama seperti input)

`normalRatioMethod(target, ref1, ref2, index, month)`

Purpose: Normal Ratio Method imputation

Formula:

```
Jika Ada Ref1: NRM1 = (E[Target,m] / E[Ref1,m]) × Ref1[i,m]
Jika Ada Ref2: NRM2 = (E[Target,m] / E[Ref2,m]) × Ref2[i,m]

NRM_Final = average(available NRM values)

Dimana E[X,m] = rata-rata annual untuk bulan m
```

Logic:

```
1. Calculate annual mean untuk target dan referensi
2. Hitung ratio target terhadap referensi
3. Kalikan ratio dengan nilai observasi ref
4. Average dari kedua method (jika available)
```

`inverseDistanceWeighting(refData1, refData2, index, month, distances)`

Purpose: IDW imputation method

Formula:

```
IDW = Σ(w_i × V_i) / Σ(w_i)

dimana:
w_i = 1 / d_i²  (weight is inverse of distance squared)
V_i = observed value at station i
d_i = distance to station i
```

Logic:

```
1. Get observations dari ref1 dan ref2
2. Calculate weights: w1 = 1/d1², w2 = 1/d2²
3. Gunakan only available observations
4. Weighted average: (w1×v1 + w2×v2) / (w1+w2)
```

`linearRegressionMethod(target, ref1, ref2, index, month)`

Purpose: Linear regression imputation

Model: Y = a + b×X

Algorithm:

```
1. Collect paired observations (target vs ref)
2. Calculate mean: X̄ = ΣX/n, Ȳ = ΣY/n
3. Calculate slope: b = Σ(X-X̄)(Y-Ȳ) / Σ(X-X̄)²
4. Calculate intercept: a = Ȳ - b×X̄
5. Predict: Ŷ = a + b×observed_ref_value
6. Average prediction dari ref1 dan ref2
```

Note: Hanya gunakan data points di bulan yang sama

`correlationWeighting(refData1, refData2, index, month, correlation)`

Purpose: Imputation dengan bobot korelasi

Formula:

```
CW = (|r1| × V1 + |r2| × V2) / (|r1| + |r2|)

dimana:
r1, r2 = Pearson correlation koefficients
V1, V2 = observed values
```

Logic:

```
1. Get correlation coefficient r1 dan r2
2. Use absolute value |r| sebagai weight
3. Weighted average dari available observations
4. Stasiun dengan korelasi tinggi lebih berpengaruh
```

---

4. Validation Metrics

`calculateValidationMetrics()`

Purpose: Hitung semua metrik validasi

Parameters: None (menggunakan global dataFiles dan imputedData)

Metrics Calculated:

```
{
    rmse: number,
    mae: number,
    mape: number,
    nse: number,
    correlation: number,
    imputedCount: number
}
```

Individual Metric Functions:

RMSE (Root Mean Squared Error)

```
function calculateRMSE(original, imputed) {
    let sumSquaredErrors = 0;
    let count = 0;

    for each point {
        if (imputed !== null) {
            sumSquaredErrors += Math.pow(original - imputed, 2);
            count++;
        }
    }

    return Math.sqrt(sumSquaredErrors / count);
}
```

MAE (Mean Absolute Error)

```
function calculateMAE(original, imputed) {
    let sumAbsErrors = 0;
    let count = 0;

    for each point {
        if (imputed !== null) {
```

```
            sumAbsErrors += Math.abs(original - imputed);
            count++;
        }
    }

    return sumAbsErrors / count;
}
```

MAPE (Mean Absolute Percentage Error)

```
function calculateMAPE(original, imputed) {
    let sumPercentErrors = 0;
    let count = 0;

    for each point {
        if (original > 0) {
            sumPercentErrors += Math.abs((original - imputed) / original);
            count++;
        }
    }

    return (sumPercentErrors / count) × 100;  // in percentage
}
```

NSE (Nash-Sutcliffe Efficiency)

```
NSE = 1 - (Σ(Y_obs - Y_sim)² / Σ(Y_obs - Ȳ)²)

Dimana:
- Y_obs = observed values
- Y_sim = simulated/imputed values
- Ȳ = mean dari observations
```

---

5. Visualization & Export

`createCharts(imputedData)`

Purpose: Buat grafik visualisasi

Library: Chart.js 3.9.1

Chart Type: Line chart

Data:

```
{
    labels: months (Jan-Dec),
    datasets: [
        {label: 'Original', data: [...], borderColor: 'blue'},
        {label: 'Imputed', data: [...], borderColor: 'red'}
    ]
}
```

`downloadExcel()`

Purpose: Export hasil ke format Excel

Output Format:

- File name: `RainfallFussion_Results_[timestamp].xlsx`
- Sheet 1: Imputed Data
- Sheet 2: Validation Metrics
- Sheet 3: Method Comparison

Implementation:

```
1. Create workbook dengan XLSX library
2. Prepare data untuk setiap sheet
3. Add formulas dan formatting
4. Generate Excel file
5. Trigger download via blob
```

Data Flow

1. Input Phase

```
User Input Coordinates
  ↓
Change Event Listener
  ↓
saveCoordinate()
  ↓
haversineDistance()
  ↓
displayDistanceInfo()
```

2. Upload Phase

```
User Select Files (3x)
  ↓
readExcelFile() (for each station)
  ↓
parseData()
  ↓
dataFiles{target, ref1, ref2} updated
  ↓
checkAllFilesLoaded()
  ↓
Enable processBtn
```

3. Processing Phase

```
User Click PROCESS DATA
  ↓
processData() async
  ↓
calculateCorrelation()
  ↓
multiStationVotingEnsemble()
  ├─ normalRatioMethod()
  ├─ inverseDistanceWeighting()
  ├─ linearRegressionMethod()
  └─ correlationWeighting()
  ↓
calculateValidationMetrics()
  ├─ calculateRMSE()
  ├─ calculateMAE()
  ├─ calculateMAPE()
  ├─ calculateNSE()
  └─ calculatePearson()
  ↓
displayMetrics()
displayFillOutTable()
createCharts()
  ↓
Enable downloadBtn
```

4. Export Phase

```
User Click DOWNLOAD
  ↓
downloadExcel()
  ↓
Create XLSX workbook
  ↓
Add sheets with data
```

```
  ↓
Generate file blob
  ↓
Browser download
```

Algorithm Details

Ensemble Voting Strategy

Problem: Single method tidak selalu optimal untuk semua kondisi

Solution: Voting Ensemble

```
1. Run 4 different imputation methods
2. Validate each result >= 0 (logical value)
3. Average valid results
4. If no valid result, use 0
```

Why This Works:

- **Diversity:** 4 metode dengan mathematical principles berbeda
- **Robustness:** Mitigasi bias individual method
- **Flexibility:** Automatically uses available methods
- **Balance:** Equal weight untuk semua methods

Example:

```
Missing value di Target[100].months[0] (January)

NRM:     22.3 mm (based on ratio)
IDW:     23.8 mm (distance-weighted)
LR:      21.9 mm (regression-based)
CW:      22.6 mm (correlation-weighted)

Ensemble Average = (22.3 + 23.8 + 21.9 + 22.6) / 4 = 22.65 mm
```

Variable Reference

Global Variables

```
// Data storage
const dataFiles = {
    target: [{year, day, months[]}],
    ref1: [{year, day, months[]}],
```

```
    ref2: [{year, day, months[]}]
};

// Coordinates (decimal degrees)
const coordinates = {
    target: {lat, lon},
    ref1: {lat, lon},
    ref2: {lat, lon}
};

// Distances (kilometers)
const distances = {
    ref1: number,
    ref2: number
};

// Results
let imputedData = [{year, day, months[]}];
let validationMetrics = {
    rmse, mae, mape, nse, correlation, imputedCount
};
```

DOM Element References

```
// Coordinate inputs
#targetLat, #targetLon
#ref1Lat, #ref1Lon
#ref2Lat, #ref2Lon

// File uploads
#targetFile, #ref1File, #ref2File

// Buttons
#processBtn, #downloadBtn

// Display areas
#progress, #metrics, #distanceInfo
#foTable, chartCanvas

// Status messages
.status (div elements)
```

Extending the Code

How to Add a New Imputation Method

1. **Create function:**

```
function myNewMethod(targetData, refData1, refData2, index, month) {
    // Your algorithm here
    let result = /* calculation */;

    if (result >= 0) {
        return result;  // Valid value
    } else {
        return null;    // Invalid value
    }
}
```

1. **Add to ensemble:**

```
async function multiStationVotingEnsemble(...) {
    const predictions = [];

    // ... existing methods ...

    const newMethod = myNewMethod(...);
    if (newMethod !== null && newMethod >= 0) {
        predictions.push(newMethod);
    }

    // ... voting logic ...
}
```

1. **Update validation:**

```
async function calculateValidationMetrics() {
    // ... existing code ...

    // Add comparison untuk method baru
    methodComparison.newMethod = [...];
}
```

How to Add New Validation Metric

```
function calculateNewMetric(original, imputed) {
    let result = 0;
    let count = 0;

    for (let i = 0; i < original.length; i++) {
        if (imputed[i] !== null) {
            // Your metric calculation
            result += /* calculation */;
```

```
            count++;
        }
    }

    return result / count;
}

// Update displayMetrics() to show new metric
validationMetrics.newMetric = calculateNewMetric(...);
```

How to Modify Export Format

Edit `downloadExcel()` function:

```
function downloadExcel() {
    const wb = XLSX.utils.book_new();

    // Modify existing sheets
    const sheet1 = XLSX.utils.json_to_sheet(imputedData);
    XLSX.utils.book_append_sheet(wb, sheet1, "Modified Sheet Name");

    // Add new sheet
    const customData = [/* your data */];
    const customSheet = XLSX.utils.json_to_sheet(customData);
    XLSX.utils.book_append_sheet(wb, customSheet, "New Sheet");

    // Download
    XLSX.writeFile(wb, `Custom_${new Date().getTime()}.xlsx`);
}
```

Performance Considerations

Data Size Impact

```
100 records ×  12 months:  ~50ms
500 records ×  12 months:  ~200ms
2000 records × 12 months:  ~800ms
```

Browser Compatibility

- **Modern browsers:** Full support
- **Old browsers:** May not support BigInt or newer JS features
- **Mobile:** Testing needed for large datasets

Memory Usage

- **Typical case:** < 10 MB for 2000 records
- **Large case:** Monitor for browser memory limits

Optimization Tips

```
// Use lazy loading untuk large data
// Implement worker threads untuk processing
// Cache calculations untuk repeated operations
// Use requestAnimationFrame untuk UI updates
```

Debugging Tips

Enable Console Logging

```
// Add untuk trace execution
console.log('Variable state:', dataFiles);
console.log('Metric calculated:', validationMetrics);
```

Browser DevTools

```
F12 → Console
- Check untuk errors
- Monitor variable values
- Trace function calls

F12 → Performance
- Profile untuk bottlenecks
- Analyze memory usage
```

Common Issues & Solutions

```
Issue: "Cannot read property of undefined"
→ Check Array initialization

Issue: "File format error"
→ Validate Excel struktur

Issue: "NaN di hasil"
→ Check division by zero, null values
```

Documentation Complete

Dokumentasi teknis selesai. Untuk pertanyaan lebih lanjut, rujuk ke README.md atau QUICK_START.md