FishCast AI - Dokumentasi Lengkap Aplikasi (Bagian 2)

Daftar Isi

Komponen Frontend

1. Base Template (`base.html`)

Fitur: - Responsive navigation dengan sidebar - Bootstrap 5 styling dengan custom CSS - Font Awesome icons - Chart.js integration - Global JavaScript functions

Komponen: - **Navbar**: Logo, menu utama, user info - **Sidebar**: Navigation menu dengan active states - **Main Content**: Area untuk konten halaman - **Alerts**: System untuk menampilkan messages - **Modals**: Reusable modal components

CSS Variables:

```css
:root {
    --primary-color: #2c3e50;
    --secondary-color: #3498db;
    --accent-color: #e74c3c;
    --success-color: #27ae60;
    --warning-color: #f39c12;
    --light-bg: #ecf0f1;
    --dark-bg: #2c3e50;
}
```

JavaScript Functions:

```javascript
// Global functions
function showLoading(element) {
    element.querySelector('.loading').style.display = 'inline-block';
    element.querySelector('.btn-text').style.display = 'none';
}

function hideLoading(element) {
    element.querySelector('.loading').style.display = 'none';
    element.querySelector('.btn-text').style.display = 'inline-block';
}
```

```
function showAlert(message, type = 'success') {
    const alertDiv = document.createElement('div');
    alertDiv.className = `alert alert-${type} alert-dismissible fade show`;
    alertDiv.innerHTML = `
        ${message}
        <button type="button" class="btn-close" data-bs-dismiss="alert"></button>
    `;
    document.querySelector('.main-content').insertBefore(alertDiv, document.querySelector('.
}
```

2. Dashboard (`dashboard.html`)

Komponen: - **Statistics Cards**: Total datasets, predictions, optimizations, correlations - **Recent Activities**: Tabel aktivitas terbaru - **Quick Actions**: Button untuk aksi cepat

Statistics Cards:

```
<div class="stats-card">
    <div class="d-flex justify-content-between">
        <div>
            <h3>{{ total_datasets }}</h3>
            <p>Total Datasets</p>
        </div>
        <div class="align-self-center">
            <i class="fas fa-database fa-3x opacity-75"></i>
        </div>
    </div>
</div>
```

JavaScript Functions:

```
function viewDataset(datasetId) {
    window.location.href = `/api/datasets/${datasetId}/`;
}

function viewPrediction(predictionId) {
    window.location.href = `/api/predictions/${predictionId}/`;
}

// Auto-refresh dashboard every 30 seconds
setInterval(function() {
    // You can add AJAX call here to refresh data without full page reload
}, 30000);
```

3. Datasets Page (`datasets.html`)

Fitur: - **Upload Modal**: Form untuk upload dataset - **Dataset Table**: Daftar semua dataset dengan actions - **Action Buttons**: View, Predict, Optimize, Correlate, Delete

Upload Form:

```
<form id="uploadForm" enctype="multipart/form-data">
    <div class="mb-3">
        <label for="datasetName" class="form-label">Dataset Name</label>
        <input type="text" class="form-control" id="datasetName" name="name" required>
    </div>
    <div class="mb-3">
        <label for="datasetFile" class="form-label">CSV File</label>
        <input type="file" class="form-control" id="datasetFile" name="file" accept=".csv" r
    </div>
    <div class="mb-3">
        <label for="datasetDescription" class="form-label">Description (Optional)</label>
        <textarea class="form-control" id="datasetDescription" name="description" rows="3"><
    </div>
</form>
```

JavaScript Functions:

```
document.getElementById('uploadForm').addEventListener('submit', function(e) {
    e.preventDefault();

    const formData = new FormData(this);
    const submitBtn = this.querySelector('button[type="submit"]');

    showLoading(submitBtn);

    fetch('/api/datasets/', {
        method: 'POST',
        body: formData
    })
    .then(response => response.json())
    .then(data => {
        hideLoading(submitBtn);
        if (data.id) {
            showAlert('Dataset uploaded successfully!', 'success');
            setTimeout(() => location.reload(), 1500);
        } else {
            showAlert('Error uploading dataset: ' + (data.error || 'Unknown error'), 'dange
        }
    })
    .catch(error => {
        hideLoading(submitBtn);
        showAlert('Error uploading dataset: ' + error.message, 'danger');
    });
});

function runPrediction(datasetId) {
    if (confirm('Run prediction on this dataset?')) {
        fetch('/api/predict/', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify({
                dataset_id: datasetId,
                models: ['Linear', 'LSTM', 'GRU']
            })
        })
        .then(response => response.json())
        .then(data => {
            if (data.message) {
                showAlert('Prediction started successfully!', 'success');
                setTimeout(() => window.location.href = '/predictions/', 1500);
            } else {
                showAlert('Error running prediction: ' + (data.error || 'Unknown error'), 'c
            }
        })
        .catch(error => {
            showAlert('Error running prediction: ' + error.message, 'danger');
        });
    }
}
```

4. Predictions Page (predictions.html)

Fitur: - **Prediction Modal**: Form untuk run prediction - **Results Table**: Daftar hasil prediksi - **Chart Modal**: Visualisasi hasil prediksi

Prediction Form:

```
<form id="predictionForm">
    <div class="mb-3">
        <label for="predictionDataset" class="form-label">Select Dataset</label>
        <select class="form-select" id="predictionDataset" name="dataset_id" required>
            <option value="">Choose a dataset...</option>
            {% for dataset in datasets %}
                <option value="{{ dataset.id }}">{{ dataset.name }}</option>
            {% endfor %}
        </select>
    </div>
    <div class="mb-3">
        <label class="form-label">Select Models</label>
        <div class="form-check">
            <input class="form-check-input" type="checkbox" name="models" value="Linear" id=
            <label class="form-check-label" for="modelLinear">Linear Regression</label>
        </div>
        <!-- More model options -->
    </div>
</form>
```

Chart Visualization:

```
function showChart(predictionId) {
    fetch(`/api/predictions/${predictionId}/`)
    .then(response => response.json())
    .then(data => {
        const ctx = document.getElementById('predictionChart').getContext('2d');

        // Destroy existing chart if it exists
        if (window.predictionChart) {
            window.predictionChart.destroy();
        }

        const actualValues = data.actual_values;
        const predictedValues = data.predictions;

        const labels = Array.from({length: actualValues.length}, (_, i) => i + 1);

        window.predictionChart = new Chart(ctx, {
            type: 'line',
            data: {
                labels: labels,
                datasets: [
                    {
                        label: 'Actual Values',
                        data: actualValues,
                        borderColor: 'rgb(75, 192, 192)',
                        backgroundColor: 'rgba(75, 192, 192, 0.2)',
                        tension: 0.1
                    },
                    {
                        label: 'Predicted Values',
                        data: predictedValues,
                        borderColor: 'rgb(255, 99, 132)',
                        backgroundColor: 'rgba(255, 99, 132, 0.2)',
                        tension: 0.1
                    }
                ]
            },
            options: {
                responsive: true,
                plugins: {
                    title: {
                        display: true,
                        text: `${data.model_type} Prediction Results`
                    }
                },
                scales: {
                    y: {
                        beginAtZero: true
                    }
                }
            }
        });

        $('#chartModal').modal('show');
    })
    .catch(error => {
        showAlert('Error loading chart: ' + error.message, 'danger');
    });
}
```

5. Optimization Page (`optimization.html`)

Fitur: - **Optimization Modal**: Form dengan parameter NSGA-III - **Results Table**: Daftar hasil optimisasi - **Pareto Front Chart**: Scatter plot solutions

Optimization Form:

```
<form id="optimizationForm">
    <div class="mb-3">
        <label for="optimizationDataset" class="form-label">Select Dataset</label>
        <select class="form-select" id="optimizationDataset" name="dataset_id" required>
            <option value="">Choose a dataset...</option>
            {% for dataset in datasets %}
                <option value="{{ dataset.id }}">{{ dataset.name }}</option>
            {% endfor %}
        </select>
    </div>
    <div class="mb-3">
        <label for="populationSize" class="form-label">Population Size</label>
        <input type="number" class="form-control" id="populationSize" name="population_size"
        <div class="form-text">Number of individuals in the population (10-200)</div>
    </div>
    <div class="mb-3">
        <label for="generations" class="form-label">Number of Generations</label>
        <input type="number" class="form-control" id="generations" name="generations" value=
        <div class="form-text">Number of generations to evolve (10-500)</div>
    </div>
</form>
```

Pareto Front Visualization:

```
function showOptimizationChart(optimizationId) {
    fetch(`/api/optimization-results/${optimizationId}/`)
    .then(response => response.json())
    .then(data => {
        const ctx = document.getElementById('optimizationChart').getContext('2d');

        // Destroy existing chart if it exists
        if (window.optimizationChart) {
            window.optimizationChart.destroy();
        }

        // Extract data for Pareto front visualization
        const solutions = data.solutions;

        window.optimizationChart = new Chart(ctx, {
            type: 'scatter',
            data: {
                datasets: [
                    {
                        label: 'All Solutions',
                        data: solutions.map((s, i) => ({
                            x: s.total_stok,
                            y: s.mse
                        })),
                        backgroundColor: 'rgba(75, 192, 192, 0.6)',
                        borderColor: 'rgb(75, 192, 192)',
                        pointRadius: 4
                    },
                    {
                        label: 'Best Solution',
                        data: [{
                            x: data.best_solution.total_stok,
                            y: data.best_solution.mse
                        }],
                        backgroundColor: 'rgba(255, 99, 132, 0.8)',
                        borderColor: 'rgb(255, 99, 132)',
                        pointRadius: 8,
                        pointStyle: 'star'
                    }
                ]
            },
            options: {
                responsive: true,
                plugins: {
                    title: {
                        display: true,
                        text: 'NSGA-III Optimization Results - Pareto Front'
                    }
                },
                scales: {
                    x: {
                        title: {
                            display: true,
                            text: 'Total Stock'
                        }
                    },
                    y: {
                        title: {
                            display: true,
                            text: 'MSE'
                        }
                    }
                }
            }
        });

        $('#optimizationChartModal').modal('show');
    })
    .catch(error => {
        showAlert('Error loading optimization chart: ' + error.message, 'danger');
    });
}
```

6. Correlation Page (`correlation.html`)

Fitur: - **Correlation Modal**: Form untuk run analysis - **Results Table**: Daftar hasil analisis - **Heatmap Modal**: Visualisasi correlation matrix

Correlation Analysis:

```
function showCorrelationMatrix(correlationId) {
    fetch(`/api/correlation-results/${correlationId}/`)
    .then(response => response.json())
```

```
.then(data => {
    const ctx = document.getElementById('correlationMatrixChart').getContext('2d');

    // Destroy existing chart if it exists
    if (window.correlationChart) {
        window.correlationChart.destroy();
    }

    const correlationMatrix = data.correlation_matrix;
    const variables = Object.keys(correlationMatrix);

    // Prepare data for heatmap
    const chartData = {
        labels: variables,
        datasets: variables.map((var1, i) => ({
            label: var1,
            data: variables.map(var2 => correlationMatrix[var1][var2]),
            backgroundColor: variables.map((var2, j) => {
                const value = correlationMatrix[var1][var2];
                const alpha = Math.abs(value);
                return value >= 0
                    ? `rgba(75, 192, 192, ${alpha})`
                    : `rgba(255, 99, 132, ${alpha})`;
            }),
            borderColor: variables.map(var2 => correlationMatrix[var1][var2] >= 0 ? 'rgb
            borderWidth: 1
        }))
    };

    window.correlationChart = new Chart(ctx, {
        type: 'bar',
        data: chartData,
        options: {
            responsive: true,
            maintainAspectRatio: false,
            plugins: {
                title: {
                    display: true,
                    text: 'Correlation Matrix Heatmap'
                },
                legend: {
                    display: false
                }
            },
            scales: {
                x: {
                        title: {
                            display: true,
                            text: 'Variables'
                        }
                    },
                    y: {
                        title: {
                            display: true,
                            text: 'Correlation Coefficient'
                        },
                        min: -1,
                        max: 1
                    }
                }
            }
        });

        // Populate correlation table
        const tableBody = document.getElementById('correlationTable').querySelector('tbody')
        tableBody.innerHTML = '';

        variables.forEach(var1 => {
            const row = tableBody.insertRow();
            const cell1 = row.insertCell(0);
            const cell2 = row.insertCell(1);

            cell1.textContent = var1;
            cell2.innerHTML = variables.map(var2 => {
                const value = correlationMatrix[var1][var2];
                const color = value >= 0 ? 'text-success' : 'text-danger';
                return `<span class="${color}">${value.toFixed(3)}</span>`;
            }).join(' | ');
        });

        $('#correlationMatrixModal').modal('show');
    })
    .catch(error => {
        showAlert('Error loading correlation matrix: ' + error.message, 'danger');
    });
}
```

API Endpoints

Base URL: `http://localhost:8001/api/`

1. Health Check

```
GET /api/health/
Response: {"status": "healthy"}
```

2. Dataset Endpoints

List Datasets

```
GET /api/datasets/
Response: [{"id": 1, "name": "dataset1", "uploaded_at": "2024-01-01T00:00:00Z", ...}]
```

Upload Dataset

```
POST /api/datasets/
Content-Type: multipart/form-data
Body: {
    "name": "Dataset Name",
    "file": <CSV file>,
    "description": "Optional description"
}
Response: {"id": 1, "name": "Dataset Name", ...}
```

Get Dataset Detail

```
GET /api/datasets/{id}/
Response: {"id": 1, "name": "dataset1", "file": "path/to/file.csv", ...}
```

Delete Dataset

```
DELETE /api/datasets/{id}/
Response: 204 No Content
```

3. Prediction Endpoints

Run Prediction

```
POST /api/predict/
Content-Type: application/json
Body: {
    "dataset_id": 1,
    "models": ["Linear", "LSTM", "GRU"]
}
Response: {"message": "Prediction started successfully"}
```

List Predictions

```
GET /api/predictions/
Response: [{"id": 1, "model_type": "Linear", "mse": 0.1234, ...}]
```

Get Prediction Detail

```
GET /api/predictions/{id}/
Response: {
    "id": 1,
    "model_type": "Linear",
    "predictions": [1.2, 1.3, 1.4, ...],
    "actual_values": [1.1, 1.2, 1.3, ...],
    "mse": 0.1234,
    "mae": 0.0987
}
```

4. Optimization Endpoints

Run Optimization

```
POST /api/optimize/
Content-Type: application/json
Body: {
    "dataset_id": 1,
    "population_size": 40,
    "generations": 100
}
Response: {"message": "Optimization started successfully"}
```

List Optimization Results

```
GET /api/optimization-results/
Response: [{"id": 1, "best_total_stok": 1000.5, "best_mse": 0.1234, ...}]
```

Get Optimization Detail

```
GET /api/optimization-results/{id}/
Response: {
    "id": 1,
    "solutions": [{"total_stok": 1000.5, "mse": 0.1234}, ...],
    "best_solution": {"total_stok": 1000.5, "mse": 0.1234},
    "best_total_stok": 1000.5,
    "best_mse": 0.1234
}
```

5. Correlation Endpoints

Run Correlation Analysis

```
POST /api/correlation/
Content-Type: application/json
Body: {
    "dataset_id": 1
}
Response: {"message": "Correlation analysis started successfully"}
```

List Correlation Results

```
GET /api/correlation-results/
Response: [{"id": 1, "dataset": {"name": "dataset1"}, ...}]
```

Get Correlation Detail

```
GET /api/correlation-results/{id}/
Response: {
    "id": 1,
    "correlation_matrix": {
        "var1": {"var1": 1.0, "var2": 0.5, "var3": -0.3},
        "var2": {"var1": 0.5, "var2": 1.0, "var3": 0.7},
        "var3": {"var1": -0.3, "var2": 0.7, "var3": 1.0}
    }
}
```

6. Export Endpoint

Export Prediction Results

```
GET /api/export/{prediction_id}/
Response: CSV file with actual vs predicted values
```

Machine Learning Pipeline

1. Data Preprocessing

```python
def preprocess_data(df):
    # Handle missing values
    df = df.fillna(method='ffill')

    # Feature scaling
    scaler = StandardScaler()
    numerical_cols = df.select_dtypes(include=[np.number]).columns
    df[numerical_cols] = scaler.fit_transform(df[numerical_cols])
    # Train-test split
    train_size = int(len(df) * 0.8)
    train_data = df[:train_size]
    test_data = df[train_size:]

    return train_data, test_data, scaler
```

2. Model Training

Linear Regression

```
def train_linear_model(X_train, y_train):
    model = LinearRegression()
    model.fit(X_train, y_train)
    return model
```

LSTM Model

```
def create_lstm_model(input_shape):
    model = Sequential([
        LSTM(50, return_sequences=True, input_shape=input_shape),
        LSTM(50, return_sequences=False),
        Dense(25),
        Dense(1)
    ])
    model.compile(optimizer='adam', loss='mean_squared_error')
    return model
```

GRU Model

```
def create_gru_model(input_shape):
    model = Sequential([
        GRU(50, return_sequences=True, input_shape=input_shape),
        GRU(50, return_sequences=False),
        Dense(25),
        Dense(1)
    ])
    model.compile(optimizer='adam', loss='mean_squared_error')
    return model
```

3. NSGA-III Optimization

```
def nsga3_optimization(dataset, population_size=40, generations=100):
    # Define objectives
    def objective1(solution):  # Maximize total_stok
        return calculate_total_stok(solution)
```

```python
    def objective2(solution):  # Minimize MSE
        return calculate_mse(solution)

    # Initialize population
    population = initialize_population(population_size)

    for generation in range(generations):
        # Evaluate objectives
        objectives = [objective1(population), objective2(population)]

        # Non-dominated sorting
        fronts = non_dominated_sort(objectives)

        # Selection, crossover, mutation
        offspring = genetic_operators(population)

        # Environmental selection
        population = environmental_selection(population, offspring)

    return extract_pareto_front(population)
```

4. Correlation Analysis

```python
def calculate_correlation_matrix(df):
    # Select numerical columns only
    numerical_df = df.select_dtypes(include=[np.number])

    # Calculate correlation matrix
    correlation_matrix = numerical_df.corr()

    # Convert to dictionary format
    correlation_dict = correlation_matrix.to_dict()

    return correlation_dict
```

Cara Penggunaan

1. Setup Development Environment

```bash
# Clone repository
git clone <repository-url>
cd backend

# Create virtual environment
```

```
python -m venv venv
source venv/bin/activate  # Linux/Mac
# atau
venv\Scripts\activate  # Windows

# Install dependencies
pip install -r requirements.txt

# Run migrations
python manage.py makemigrations
python manage.py migrate

# Create superuser (optional)
python manage.py createsuperuser

# Run development server
python manage.py runserver 8001
```

2. Akses Aplikasi

- **Dashboard**: http://localhost:8001/
- **Admin Panel**: http://localhost:8001/admin/
- **API Base**: http://localhost:8001/api/

3. Workflow Penggunaan

Step 1: Upload Dataset

1. Buka halaman Datasets
2. Klik "Upload Dataset"
3. Isi nama dataset
4. Pilih file CSV
5. Tambah deskripsi (opsional)
6. Klik "Upload"

Step 2: Run Prediction

1. Buka halaman Predictions
2. Klik "Run New Prediction"
3. Pilih dataset
4. Pilih model(s) yang diinginkan
5. Klik "Run Prediction"
6. Tunggu proses selesai
7. Lihat hasil di tabel

Step 3: Run Optimization

1. Buka halaman Optimization
2. Klik “Run New Optimization”
3. Pilih dataset
4. Set parameter (population_size, generations)
5. Klik “Run Optimization”
6. Tunggu proses selesai
7. Lihat Pareto front chart

Step 4: Run Correlation Analysis

1. Buka halaman Correlation Analysis
2. Klik “Run New Analysis”
3. Pilih dataset
4. Klik “Run Analysis”
5. Lihat correlation matrix heatmap

4. Format Dataset

Dataset harus dalam format CSV dengan struktur:

```
date,temperature,salinity,ph,dissolved_oxygen,fish_count
2024-01-01,25.5,35.2,7.8,6.2,150
2024-01-02,26.1,34.8,7.9,6.5,165
...
```

Requirements: - Header row dengan nama kolom - Data numerik untuk analisis - Minimal 10 baris data - Maksimal 10MB file size

---

Lanjut ke bagian 3 untuk troubleshooting dan pengembangan selanjutnya…