FishCast AI - Dokumentasi Lengkap Aplikasi (Bagian 3)

Daftar Isi

Troubleshooting

1. Common Issues

Port Already in Use

```
# Kill process on port 8001
sudo lsof -ti:8001 | xargs kill -9

# Atau gunakan port lain
python manage.py runserver 8002
```

Database Migration Error

```
# Reset migrations
python manage.py migrate api zero
rm api/migrations/0*.py
python manage.py makemigrations
python manage.py migrate
```

Static Files Not Loading

```
# Collect static files
python manage.py collectstatic
```

File Upload Error

- Cek permission folder `media/`
- Pastikan file tidak terlalu besar
- Validasi format CSV

2. Debug Mode

Aktifkan debug mode di `settings.py`:

```
DEBUG = True
```

3. Log Files

Cek log Django:

```
python manage.py runserver 8001 --verbosity=2
```

4. Database Inspection

```
# Masuk ke Django shell
python manage.py shell

# Inspect models
from api.models import Dataset, Prediction
Dataset.objects.all()
Prediction.objects.all()
```

5. Common Error Messages

ModuleNotFoundError: No module named 'pandas'

```
# Install missing dependencies
pip install pandas numpy
```

TemplateDoesNotExist

- Pastikan file template ada di folder `templates/`
- Cek path di `settings.py` TEMPLATES setting

CSRF Token Missing

- Pastikan form memiliki `{% csrf_token %}`
- Atau gunakan `@csrf_exempt` untuk API endpoints

Permission Denied

```
# Fix folder permissions
chmod 755 media/
chmod 755 staticfiles/
```

6. Performance Issues

Slow Database Queries

```
# Use select_related for foreign keys
predictions = Prediction.objects.select_related('dataset').all()

# Use prefetch_related for many-to-many
datasets = Dataset.objects.prefetch_related('predictions').all()
```

Large File Uploads

```
# Increase max file size in settings.py
DATA_UPLOAD_MAX_MEMORY_SIZE = 10485760  # 10MB
FILE_UPLOAD_MAX_MEMORY_SIZE = 10485760  # 10MB
```

Memory Issues

```
# Use streaming for large files
def handle_uploaded_file(f):
    with open('some/file/name.txt', 'wb+') as destination:
        for chunk in f.chunks():
            destination.write(chunk)
```

7. Security Issues

CORS Errors

```
# Update CORS settings in settings.py
CORS_ALLOWED_ORIGINS = [
    "http://localhost:3000",
    "http://127.0.0.1:3000",
    "http://localhost:8001",
]
```

File Upload Security

```
# Validate file types
ALLOWED_EXTENSIONS = ['csv']
def allowed_file(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS
```

8. Testing

Run Tests

```
python manage.py test api
```

Create Test Data

```
# In Django shell
from api.models import Dataset
from django.core.files import File

# Create test dataset
dataset = Dataset.objects.create(
    name="Test Dataset",
```

```
    description="Test data for development"
)
```

Pengembangan Selanjutnya

1. Fitur yang Direncanakan

Advanced ML Models

- **XGBoost**: Gradient boosting untuk prediksi
- **Random Forest**: Ensemble learning
- **Neural Networks**: Deep learning dengan TensorFlow
- **Time Series Models**: ARIMA, Prophet

Enhanced Optimization

- **Multi-objective**: Lebih dari 2 objectives
- **Constraint Handling**: Hard dan soft constraints
- **Interactive Optimization**: User-guided optimization
- **Real-time Optimization**: Live parameter adjustment

Data Visualization

- **Interactive Charts**: Plotly.js integration
- **3D Visualization**: Three.js untuk 3D plots
- **Geographic Maps**: Folium untuk spatial data
- **Real-time Dashboard**: WebSocket untuk live updates

User Management

- **Authentication**: User login/logout
- **Authorization**: Role-based access control
- **User Profiles**: Personal settings dan preferences
- **Project Management**: Organize datasets by projects

API Enhancements

- **GraphQL**: Alternative to REST API
- **WebSocket**: Real-time communication
- **Rate Limiting**: API usage limits
- **API Documentation**: Swagger/OpenAPI

2. Performance Improvements

Database Optimization

- **Indexing**: Add database indexes

- **Query Optimization**: Optimize database queries
- **Caching**: Redis for caching
- **Database Migration**: PostgreSQL for production

Scalability

- **Load Balancing**: Multiple server instances
- **Microservices**: Split into smaller services
- **Containerization**: Docker deployment
- **Cloud Deployment**: AWS, GCP, Azure

ML Pipeline Optimization

- **Parallel Processing**: Multi-threading untuk ML
- **GPU Acceleration**: CUDA support
- **Model Caching**: Cache trained models
- **Incremental Learning**: Online learning

3. Security Enhancements

Data Security

- **Encryption**: Encrypt sensitive data
- **Data Masking**: Anonymize personal data
- **Access Control**: Fine-grained permissions
- **Audit Logging**: Track all data access

API Security

- **JWT Authentication**: Token-based auth
- **OAuth2**: Third-party authentication
- **API Keys**: Secure API access
- **HTTPS**: SSL/TLS encryption

4. Monitoring & Analytics

Application Monitoring

- **Health Checks**: Automated monitoring
- **Performance Metrics**: Response times, throughput
- **Error Tracking**: Sentry integration
- **User Analytics**: Usage patterns

ML Model Monitoring

- **Model Performance**: Track model degradation
- **Data Drift**: Monitor data distribution changes
- **A/B Testing**: Compare model versions
- **Model Registry**: Version control for models

5. Specific Implementation Plans

Phase 1: Core Enhancements (1-2 months)

1. **User Authentication System**

```python
# models.py
class UserProfile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    organization = models.CharField(max_length=100)
    role = models.CharField(max_length=50)
    preferences = models.JSONField(default=dict)
```

2. **Advanced Model Support**

```python
# ml_models.py
def train_xgboost_model(X_train, y_train):
    model = XGBRegressor(
        n_estimators=100,
        learning_rate=0.1,
        max_depth=6
    )
    model.fit(X_train, y_train)
    return model
```

3. **Real-time Notifications**

```python
# views.py
from channels.layers import get_channel_layer
from asgiref.sync import async_to_sync

def notify_user(user_id, message):
    channel_layer = get_channel_layer()
    async_to_sync(channel_layer.group_send)(
        f"user_{user_id}",
        {"type": "notification", "message": message}
    )
```

Phase 2: Advanced Features (3-4 months)

1. **Interactive Dashboard**

```javascript
// dashboard.js
class RealTimeDashboard {
    constructor() {
        this.charts = {};
        this.websocket = new WebSocket('ws://localhost:8001/ws/');
        this.initializeCharts();
    }
```

```
    updateChart(chartId, data) {
        if (this.charts[chartId]) {
            this.charts[chartId].data.datasets[0].data = data;
            this.charts[chartId].update();
        }
    }
}
```

2. **Advanced Analytics**

```
# analytics.py
class AdvancedAnalytics:
    def __init__(self, dataset):
        self.dataset = dataset

    def perform_trend_analysis(self):
        # Implement trend analysis
        pass

    def detect_anomalies(self):
        # Implement anomaly detection
        pass

    def generate_insights(self):
        # Generate business insights
        pass
```

Phase 3: Enterprise Features (5-6 months)

1. **Multi-tenant Architecture**

```
# models.py
class Tenant(models.Model):
    name = models.CharField(max_length=100)
    domain = models.CharField(max_length=100, unique=True)
    settings = models.JSONField(default=dict)

class TenantAwareModel(models.Model):
    tenant = models.ForeignKey(Tenant, on_delete=models.CASCADE)

    class Meta:
        abstract = True
```

2. **Advanced Security**

```
# security.py
from cryptography.fernet import Fernet
```

```python
class DataEncryption:
    def __init__(self):
        self.key = Fernet.generate_key()
        self.cipher_suite = Fernet(self.key)

    def encrypt_data(self, data):
        return self.cipher_suite.encrypt(data.encode())

    def decrypt_data(self, encrypted_data):
        return self.cipher_suite.decrypt(encrypted_data).decode()
```

6. Technology Stack Evolution

Current Stack

- **Backend**: Django 5.2.4 + DRF
- **Database**: SQLite (dev) / PostgreSQL (prod)
- **Frontend**: HTML + CSS + JavaScript
- **ML**: Pandas + NumPy

Future Stack

- **Backend**: Django + FastAPI (hybrid)
- **Database**: PostgreSQL + Redis (caching)
- **Frontend**: React/Vue.js + TypeScript
- **ML**: TensorFlow/PyTorch + MLflow
- **Deployment**: Docker + Kubernetes
- **Monitoring**: Prometheus + Grafana

7. Development Roadmap

Q1 2024: Foundation

- ☒ Basic Django application
- ☒ Database models
- ☒ API endpoints
- ☒ Basic frontend
- ☐ User authentication
- ☐ File upload system

Q2 2024: Core Features

- ☐ Machine learning integration
- ☐ Prediction models
- ☐ Optimization algorithms
- ☐ Correlation analysis
- ☐ Data visualization
- ☐ Export functionality

Q3 2024: Advanced Features

- ☐ Real-time dashboard
- ☐ Advanced ML models
- ☐ User management
- ☐ Project organization
- ☐ API documentation
- ☐ Testing suite

Q4 2024: Enterprise Ready

- ☐ Multi-tenant support
- ☐ Advanced security
- ☐ Performance optimization
- ☐ Monitoring & analytics
- ☐ Cloud deployment
- ☐ Documentation

8. Success Metrics

Technical Metrics

- **Response Time**: < 200ms for API calls
- **Uptime**: $> 99.9\%$
- **Error Rate**: $< 0.1\%$
- **Test Coverage**: $> 90\%$

User Metrics

- **User Adoption**: 100+ active users
- **Feature Usage**: 80% of users use ML features
- **User Satisfaction**: $> 4.5/5$ rating
- **Retention Rate**: $> 80\%$ monthly retention

Business Metrics

- **Data Processing**: Handle 1GB+ datasets
- **Model Accuracy**: $> 85\%$ prediction accuracy
- **Processing Speed**: < 5 minutes for ML tasks
- **Scalability**: Support 1000+ concurrent users

Kesimpulan

FishCast AI adalah aplikasi yang menggabungkan kemudahan penggunaan interface web dengan kekuatan machine learning untuk analisis data perikanan. Aplikasi ini menyediakan:

1. Dashboard Interaktif: Interface yang user-friendly untuk non-technical users

- Modern UI dengan Bootstrap 5
- Responsive design untuk semua device
- Real-time statistics dan charts
- Intuitive navigation

2. API RESTful: Integrasi dengan aplikasi lain

- Comprehensive API endpoints
- JSON-based communication
- CORS support untuk cross-origin requests
- Well-documented API structure

3. Multi-Model ML: Berbagai algoritma untuk prediksi

- Linear Regression untuk baseline
- LSTM untuk time series
- GRU untuk sequence modeling
- BiLSTM untuk bidirectional analysis
- RNN untuk recurrent patterns

4. Optimisasi Multi-Objective: NSGA-III untuk optimisasi

- Pareto front visualization
- Multi-objective optimization
- Interactive parameter tuning
- Solution comparison tools

5. Analisis Korelasi: Visualisasi hubungan antar variabel

- Correlation matrix heatmap
- Interactive charts
- Statistical insights
- Data exploration tools

6. Extensible Architecture: Mudah untuk menambah fitur baru

- Modular design
- Plugin-based architecture
- API-first approach
- Scalable database design

Key Benefits:

For Researchers:

- **Easy Data Management**: Simple upload and organization
- **Multiple Analysis Tools**: Comprehensive ML pipeline
- **Visualization**: Rich charts and graphs
- **Export Capabilities**: Multiple format support

For Developers:

- **Clean Codebase**: Well-structured Django application
- **API-First**: RESTful API for integration
- **Extensible**: Easy to add new features
- **Documented**: Comprehensive documentation

For Organizations:

- **Cost Effective**: Open-source solution
- **Scalable**: Can handle growing data needs
- **Secure**: Built-in security features
- **Maintainable**: Clear code structure

Future Vision:

FishCast AI dirancang untuk menjadi platform terdepan dalam analisis data perikanan dengan:

1. **Advanced AI/ML**: Integration dengan state-of-the-art models
2. **Real-time Analytics**: Live data processing dan visualization
3. **Collaborative Features**: Multi-user collaboration tools
4. **Industry Integration**: Connectivity dengan sistem perikanan existing
5. **Global Scale**: Support untuk berbagai jenis data perikanan worldwide

Technical Excellence:

- **Performance**: Optimized untuk large datasets
- **Security**: Enterprise-grade security features
- **Reliability**: Robust error handling dan recovery
- **Usability**: Intuitive interface untuk semua skill levels
- **Maintainability**: Clean code dan comprehensive documentation

Aplikasi ini tidak hanya menyediakan tools untuk analisis data perikanan, tetapi juga membuka jalan untuk inovasi dalam bidang aquaculture dan fisheries management melalui teknologi AI/ML yang advanced.

Kontak & Support

Untuk pertanyaan atau dukungan teknis: - **Email**: support@fishcast.ai - **Documentation**: https://docs.fishcast.ai - **GitHub**: https://github.com/fishcast-ai - **Issues**: https://github.com/fishcast-ai/issues

---

Dokumentasi ini dibuat pada: 2024-01-15 Versi Aplikasi: 1.0.0 Django Version: 5.2.4 Status: Development Complete - Ready for Production