**SIFS Streamlit Frontend Setup Guide**

**Quick Start**

**1. Install Dependencies**

# Navigate to your project root

*cd SIFS\_Ultimate*

# Install Streamlit frontend requirements

*pip install -r streamlit\_requirements.txt*

**2. Start the Backend API**

# Make sure your FastAPI backend is running

# From your project root:

*python -m uvicorn backend.api.app:app --reload --host 0.0.0.0 --port 8000*

**3. Start the Streamlit Frontend**

# Run the Streamlit app

*streamlit run streamlit\_app.py*

The application will be available at: http://localhost:8501

**Default Login Credentials**

Based on your backend auth system, you'll need to create users first. Here are some test accounts you can create:

**Admin User**

* Username: admin
* Password: admin123
* Role: admin

**Features Included**

**✅ Authentication System**

* JWT token-based authentication
* Role-based access control (Admin, Data Entry, Viewer)
* Secure login/logout functionality

**✅ Dashboard**

* Key metrics overview
* Sales trend visualization
* Inventory status charts
* Recent activity feed

**✅ Forecasting Module**

* SKU selection and forecast generation
* Interactive forecast charts
* Accuracy metrics display
* Historical vs predicted data visualization

**✅ Inventory Management**

* Reorder recommendations from your backend API
* Inventory alerts and warnings
* Supplier filtering
* Stock level monitoring

**✅ Reports**

* Sales summary reports
* Inventory reports
* Forecast accuracy analysis
* Supplier performance metrics

**✅ Data Upload**

* CSV/Excel file upload
* Data preview and validation
* Integration with your backend data processing pipeline
* Upload progress tracking

**✅ User Management (Admin Only)**

* Create, edit, delete users
* Role assignment
* User activity monitoring

**✅ System Settings (Admin Only)**

* Forecasting parameters configuration
* Notification settings
* System maintenance tools
* Backup and cache management

**API Integration**

The frontend is fully integrated with your existing FastAPI backend:

* **Authentication**: /api/auth/login
* **Dashboard**: /api/dashboard
* **Users**: /api/users
* **Data Upload**: /api/data/upload-sales
* **Inventory**: /api/inventory/reorder-recommendations
* **System Config**: /api/system/configurations

**Configuration**

**Backend API URL**

The default API base URL is set to http://localhost:8000. If your backend runs on a different host/port, update the API\_BASE\_URL variable in streamlit\_app.py:

API\_BASE\_URL = "http://your-backend-host:port"

**Environment Variables**

The frontend will work with your existing .env file for email configurations.

**Troubleshooting**

**Backend Connection Issues**

1. Ensure your FastAPI backend is running on port 8000
2. Check that CORS is properly configured in your backend
3. Verify the API\_BASE\_URL in the Streamlit app

**Authentication Problems**

1. Make sure you have users created in your database
2. Check that the JWT secret key is consistent
3. Verify user roles are correctly set

**Data Upload Issues**

1. Ensure your backend data processing pipeline is working
2. Check file format requirements (CSV/Excel)
3. Verify upload file size limits

**Next Steps**

1. **Test the Authentication**: Create test users and verify login works
2. **Upload Sample Data**: Test the data upload functionality with sample sales data
3. **Configure Settings**: Use the admin settings page to configure forecasting parameters
4. **Generate Forecasts**: Test the forecasting functionality with your data
5. **Monitor Inventory**: Check that reorder recommendations are working

**Customization**

The frontend is designed to be easily customizable:

* **Colors/Themes**: Modify the Streamlit theme in .streamlit/config.toml
* **Layout**: Adjust column layouts and component arrangements
* **Charts**: Customize Plotly visualizations
* **Metrics**: Add or modify KPI calculations
* **Pages**: Add new pages or modify existing ones

**Support**

If you encounter any issues:

1. Check the browser console for JavaScript errors
2. Check the Streamlit terminal output for Python errors
3. Verify your backend API is responding correctly
4. Ensure all dependencies are properly installed