COMPLETE APP DEVELOPMENT PLAN

PROJECT OVERVIEW

Goal: Build a comprehensive, error-free Streamlit app for Depression Detection that combines:

- Multiple trained ML models (DistilBERT, RoBERTa, etc.)
- LLM APIs (OpenAI ChatGPT, Groq Llama, Google Gemini)
- Batch processing capabilities
- Model comparison & visualization
- Clinical insights (DSM-5/PHQ-9 alignment)

APP ARCHITECTURE

Part 1: Core Setup & Configuration (Lines 1-200)

```
├── Imports & Dependencies
├── Page Configuration
├── Custom CSS Styling
├── Session State Initialization
├── Helper Functions
│   ├── Model loading
│   ├── API configuration
│   └── Data validation
```

Part 2: Model Management (Lines 201-400)

```
├── Trained Model Loader
│   ├── Discover available models
│   ├── Load model + tokenizer
│   ├── Cache management
│   └── Error handling
├── LLM API Handlers
│   ├── OpenAI integration
│   ├── Groq integration
│   ├── Google Gemini integration
│   └── Fallback mechanisms
```

Part 3: Sidebar Controls (Lines 401-600)

```
├── Analysis Mode Selection
│   ├── Trained Models Only
│   ├── LLM APIs Only
```

```
│   └── Compare Both
├── Model Selection
│   ├── Trained model dropdown
│   ├── Model info display
│   └── Training metrics
├── LLM Configuration
│   ├── Provider selection
│   ├── API key input
│   ├── Model selection
│   └── Help links
├── Settings
│   ├── Confidence threshold
│   ├── Display options
│   └── Safety warnings
```

Part 4: Main Analysis Tab (Lines 601-1000)

```
├── Text Input Area
├── Sample Texts
├── Analysis Button
├── Results Display
│   ├── Single Model Results
│   │   ├── Prediction label
│   │   ├── Confidence score
│   │   ├── Risk level
│   │   ├── Probability chart
│   │   └── Clinical insights
│   ├── Comparison Results
│   │   ├── Side-by-side display
│   │   ├── Agreement analysis
│   │   ├── Metrics comparison
│   │   └── Export options
│   └── LLM-Only Results
│       ├── Provider info
│       ├── Prediction
│       ├── Confidence
│       └── Model details
```

Part 5: Batch Processing Tab (Lines 1001-1300)

```
├── File Upload
├── Data Preview
├── Processing Controls
├── Progress Tracking
├── Results Table
├── Statistics Summary
│   ├── Total processed
│   ├── Class distribution
```

```
│   ├── Average confidence
│   └── Processing time
├── Visualizations
│   ├── Prediction distribution
│   ├── Confidence histogram
│   └── Time series (if applicable)
├── Export Options
│   ├── CSV download
│   ├── JSON export
│   └── Report generation
```

Part 6: Model Info & Comparison Tab (Lines 1301-1600)

```
├── Training Information
│   ├── Dataset statistics
│   ├── Training duration
│   ├── Hyperparameters
│   └── Class distribution
├── Model Architecture
│   ├── Model type
│   ├── Parameters count
│   ├── Layer information
│   └── Fine-tuning details
├── Performance Metrics
│   ├── Accuracy
│   ├── Precision
│   ├── Recall
│   ├── F1 Score
│   ├── ROC-AUC
│   └── Confusion matrix
├── Model Comparison
│   ├── Metrics table
│   ├── Performance charts
│   ├── Speed comparison
│   └── Best model recommendation
├── Sample Predictions
│   ├── Test examples
│   ├── Confidence scores
│   ├── Correct/Incorrect flags
│   └── Error analysis
```

Part 7: About & Help Tab (Lines 1601-1700)

```
├── System Information
│   ├── Version
│   ├── Features list
│   ├── Model types supported
│   └── LLM providers
```

```
├── Usage Guide
│   ├── Quick start
│   ├── API key setup
│   ├── Model selection guide
│   └── Best practices
├── Clinical Information
│   ├── DSM-5 criteria
│   ├── PHQ-9 scale
│   ├── Risk assessment
│   └── Professional help resources
├── Disclaimers
│   ├── Research tool warning
│   ├── Not medical advice
│   ├── Privacy notice
│   └── Contact information
```

🔧 TECHNICAL SPECIFICATIONS

Key Features to Implement

1. ☑ Multi-model support (trained models from models/trained/)
2. ☑ Three LLM providers (OpenAI, Groq, Google)
3. ☑ Real-time API key validation
4. ☑ Comparison mode (trained vs LLM)
5. ☑ Batch CSV processing
6. ☑ Visualization charts
7. ☑ Export functionality
8. ☑ Error handling & fallbacks
9. ☑ Responsive UI design
10. ☑ Clinical context integration

Error Handling Strategy

```
- Model loading: Try/catch with fallback to default
- API calls: Timeout handling + retry logic
- File uploads: Validation + error messages
- Data processing: Progress tracking + cancellation
- Network errors: Graceful degradation
```

Performance Optimizations

```
- @st.cache_resource for model loading
- @st.cache_data for data processing
- Lazy loading of heavy components
```

```
- Progress indicators for long operations
- Background processing for batch jobs
```

CODE STRUCTURE BREAKDOWN

Section 1: Imports & Setup (~100 lines)

- All library imports
- Environment setup
- Constants definition
- Utility functions

Section 2: Model Functions (~200 lines)

- get_available_models()
- load_trained_model()
- predict_with_trained_model()
- predict_with_openai()
- predict_with_groq()
- predict_with_google()

Section 3: UI Components (~300 lines)

- Sidebar creation
- Tab structure
- Input forms
- Button handlers

Section 4: Analysis Logic (~400 lines)

- Single text analysis
- Batch processing
- Comparison mode
- Results formatting

Section 5: Visualization (~200 lines)

- Charts creation
- Metrics display
- Data tables
- Export functions

Section 6: Information Pages (~200 lines)

- Model info tab
- About section
- Help documentation

- Clinical guidelines

🎨 UI/UX DESIGN

Color Scheme

- Primary: Gradient purple (#667eea → #764ba2)
- Success: Green (#44ff44)
- Warning: Yellow (#ffaa00)
- Error: Red (#ff4444)
- Background: White/Light gray

Layout

- Wide mode (1400px)
- Sidebar: 300px
- Main area: Flexible
- Footer: Full width

Components

- Metric cards with gradients
- Colored prediction badges
- Interactive charts
- Progress bars
- Data tables with styling

🚀 IMPLEMENTATION PHASES

Phase 1: Foundation (Part 1 - 500 lines)

✓ Setup imports, config, CSS
✓ Model loading functions
✓ LLM API handlers
✓ Basic UI structure

Phase 2: Core Features (Part 2 - 600 lines)

✓ Sidebar controls
✓ Single text analysis
✓ Results display
✓ Error handling

Phase 3: Advanced Features (Part 3 - 500 lines)

✓ Batch processing
✓ Comparison mode

✓ Visualizations
✓ Export functionality

Phase 4: Polish (Part 4 - 200 lines)

✓ Model info tab
✓ About section
✓ Help documentation
✓ Final testing

Total: ~1800 lines of clean, organized code

☑ QUALITY CHECKLIST

Code Quality

- ☐ No syntax errors
- ☐ All imports valid
- ☐ Functions documented
- ☐ Error handling complete
- ☐ Type hints where appropriate

Functionality

- ☐ All trained models load
- ☐ Each LLM provider works
- ☐ Comparison mode functional
- ☐ Batch processing works
- ☐ Charts render correctly
- ☐ Export files generate

User Experience

- ☐ Intuitive navigation
- ☐ Clear error messages
- ☐ Fast loading times
- ☐ Responsive design
- ☐ Help text available

Safety

- ☐ API keys not logged
- ☐ Disclaimers visible
- ☐ Privacy respected
- ☐ Rate limiting considered

📊 TESTING STRATEGY

Unit Tests

- Model loading
- Prediction functions
- Data processing
- Chart generation

Integration Tests

- End-to-end workflows
- API interactions
- File uploads
- Export functions

User Acceptance

- Load app successfully
- Make single prediction
- Process batch file
- Compare models
- Export results

SUCCESS CRITERIA

1. ☑ App launches without errors
2. ☑ All tabs functional
3. ☑ Models load correctly
4. ☑ LLM APIs work with keys
5. ☑ Batch processing completes
6. ☑ Visualizations render
7. ☑ Export files download
8. ☑ No UI glitches
9. ☑ Performance acceptable
10. ☑ User-friendly interface

DELIVERABLES

1. **app.py** - Single complete file
2. **README_APP.md** - Usage instructions
3. **requirements.txt** - Dependencies (if missing)
4. **test_app.py** - Basic tests

IMPLEMENTATION ORDER

I will code the app in 4 parts:

Part 1 (Lines 1-500): Setup, imports, model loaders, LLM handlers
Part 2 (Lines 501-1000): Sidebar, single analysis, basic results
Part 3 (Lines 1001-1500): Batch processing, comparison, charts
Part 4 (Lines 1501-1800): Model info, about, final polish

Each part will be complete, tested, and error-free before proceeding.

🚀 READY TO START

Next Step: Begin coding Part 1 (Foundation & Model Management)