ASI Market Prediction API - Frontend Developer Guide

Base URL: `http://localhost:8000`
API Version: 1.3.0

🚀 Quick Start

```javascript
// Basic API call example
const response = await fetch('http://localhost:8000/predict');
const prediction = await response.json();
console.log(`Risk Level: ${prediction.risk_level}`);
```

📍 API Endpoints

1. Root Information

```
GET /
```

Response Schema:

```json
{
  "message": "string",
  "version": "string",
  "features": ["string"],
  "status": "string",
  "database": "string",
  "data_source": "string",
  "cache_info": {
    "last_updated": "ISO datetime string",
    "cache_duration_minutes": "number",
    "data_days": "number"
  }
}
```

---

2. Get Latest Prediction

```
GET /predict
```

Response Schema:

```json
{
 "date": "YYYY-MM-DD",
 "asi_value": "number",
 "prediction": "boolean",
 "confidence": "number (0-1)",
 "explanation": "string",
 "risk_level": "HIGH|MODERATE|LOW|STABLE",
 "past_10_days": [
  {
   "date": "YYYY-MM-DD",
   "asi_value": "number",
   "prediction": "boolean",
   "confidence": "number (0-1)",
   "risk_level": "HIGH|MODERATE|LOW|STABLE",
   "actual_change_10d": "number|null",
   "actual_asi_value_10d": "number|null"
  }
 ]
}
```

Field Descriptions:

- `date`: Prediction date
- `asi_value`: Current ASI market value
- `prediction`: `true` = expects -5% drop, `false` = stable
- `confidence`: Model confidence (0.0 to 1.0)
- `risk_level`: Risk category based on prediction + confidence
- `past_10_days[].actual_change_10d`: Actual percentage change after 10 days (null if not available yet)
- `past_10_days[].actual_asi_value_10d`: Actual ASI value after 10 days (null if not available yet)

---

3. Get Prediction for Specific Date

```
GET /predict/{date}
```

Parameters:

- `date` (path): Date in YYYY-MM-DD format
- **Limits:** Max 7 days future, max 365 days past

Response Schema: Same as `/predict`

Example:

```javascript
// Get prediction for December 20th
const response = await fetch('http://localhost:8000/predict/2025-12-20');
```

---

4. Trend Analysis

```
GET /trends
```

Response Schema:

json

```
{
 "current_prediction": {
  // Same structure as /predict response
 },
 "trend_analysis": {
  "avg_confidence_5d": "number",
  "confidence_change_5d": "number",
  "prediction_count_5d": "number",
  "risk_level_distribution": {
   "HIGH": "number",
   "MODERATE": "number",
   "LOW": "number",
   "STABLE": "number"
  },
  "trend_direction": "INCREASING|DECREASING|STABLE|INSUFFICIENT_DATA",
  "accuracy_rate": "number|null",
  "completed_evaluations": "number"
 },
 "recent_predictions": [
  {
   "date": "YYYY-MM-DD",
   "asi_value": "number",
   "prediction": "boolean",
   "confidence": "number",
   "explanation": "string",
   "risk_level": "string",
   "actual_change_10d": "number|null",
   "actual_asi_value_10d": "number|null",
   "created_at": "YYYY-MM-DD HH:MM:SS"
  }
 ]
}
```

Field Descriptions:

- `avg_confidence_5d`: Average confidence over last 5 days
- `confidence_change_5d`: Change in confidence vs 5 days ago
- `prediction_count_5d`: Number of positive predictions in last 5 days
- `trend_direction`: Overall confidence trend
- `accuracy_rate`: Prediction accuracy percentage (only for completed predictions)

5. Prediction History

```
GET /history?limit={number}
```

Parameters:

- `limit` (query, optional): Number of predictions to return (default: 30)

Response Schema:

json

```
{
 "predictions": [
  {
   "date": "YYYY-MM-DD",
   "asi_value": "number",
   "prediction": "boolean",
   "confidence": "number",
   "explanation": "string",
   "risk_level": "string",
   "actual_change_10d": "number|null",
   "actual_asi_value_10d": "number|null",
   "created_at": "YYYY-MM-DD HH:MM:SS"
  }
 ],
 "summary": {
  "total_predictions": "number",
  "completed_evaluations": "number",
  "accuracy_rate": "number|null"
 }
}
```

---

6. Manual Prediction Trigger

```
POST /predict/manual
```

Response Schema:

```
json

{
  "message": "Manual prediction completed",
  "prediction": {
    // Same structure as /predict response
  }
}
```

---

7. Cache Refresh

```
POST /cache/refresh
```

Response Schema:

```
json

{
  "message": "Cache refreshed successfully",
  "last_updated": "ISO datetime string",
  "records_count": "number"
}
```

---

8. Health Check

```
GET /health
```

Response Schema:

```json
{
  "status": "healthy|unhealthy",
  "model": "loaded|not_loaded",
  "database": "string",
  "scheduler": "running|stopped",
  "cache": {
    "is_cached": "boolean",
    "last_updated": "ISO datetime string|null",
    "records_count": "number",
    "cache_age_minutes": "number|null"
  },
  "data_source": "string",
  "features": ["string"],
  "timestamp": "ISO datetime string"
}
```

🎯 Risk Level Values

Risk Level	Description	Confidence Range
HIGH	Very likely -5% drop	80%+
MODERATE	Likely -5% drop	60-80%
LOW	Possible -5% drop	<60%
STABLE	No significant drop expected	N/A

❌ Error Responses

Error Format:

```json
{
  "detail": "Error description"
}
```

HTTP Status Codes:

- `200`: Success
- `400`: Bad Request (invalid parameters)
- `500`: Internal Server Error

Common Errors:

```json
// Invalid date format
{
  "detail": "Invalid date format. Use YYYY-MM-DD"
}

// Date out of range
{
  "detail": "Cannot predict more than 7 days in the future"
}

// Model not loaded
{
  "detail": "Model not loaded"
}
```

🛠 Frontend Integration Examples

React Hook Example:

javascript

```javascript
import { useState, useEffect } from 'react';

function usePrediction() {
  const [prediction, setPrediction] = useState(null);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

  useEffect(() => {
    fetch('http://localhost:8000/predict')
      .then(res => {
        if (!res.ok) throw new Error('Failed to fetch');
        return res.json();
      })
      .then(data => {
        setPrediction(data);
        setLoading(false);
      })
      .catch(err => {
        setError(err.message);
        setLoading(false);
      });
  }, []);

  return { prediction, loading, error };
}

// Usage
function PredictionDisplay() {
  const { prediction, loading, error } = usePrediction();

  if (loading) return <div>Loading...</div>;
  if (error) return <div>Error: {error}</div>;

  return (
    <div>
      <h2>Market Prediction</h2>
      <div className={`risk-${prediction.risk_level.toLowerCase()}`}>
        Risk Level: {prediction.risk_level}
      </div>
      <p>Confidence: {(prediction.confidence * 100).toFixed(1)}%</p>
      <p>ASI Value: {prediction.asi_value.toLocaleString()}</p>
      <p>Explanation: {prediction.explanation}</p>
    </div>
  );
}
```

```
}
```

JavaScript Fetch Examples:

javascript

```javascript
// Get latest prediction
async function getLatestPrediction() {
  try {
    const response = await fetch('http://localhost:8000/predict');
    const data = await response.json();
    return data;
  } catch (error) {
    console.error('Error:', error);
    throw error;
  }
}

// Get specific date prediction
async function getDatePrediction(date) {
  const response = await fetch(`http://localhost:8000/predict/${date}`);
  if (!response.ok) {
    const error = await response.json();
    throw new Error(error.detail);
  }
  return response.json();
}

// Get trend analysis
async function getTrends() {
  const response = await fetch('http://localhost:8000/trends');
  return response.json();
}

// Manual prediction trigger
async function triggerManualPrediction() {
  const response = await fetch('http://localhost:8000/predict/manual', {
    method: 'POST'
  });
  return response.json();
}
```

CSS Classes for Risk Levels:

```css
.risk-high {
  background-color: #fee2e2;
  color: #dc2626;
  border: 1px solid #fecaca;
}

.risk-moderate {
  background-color: #fef3c7;
  color: #d97706;
  border: 1px solid #fed7aa;
}

.risk-low {
  background-color: #ecfdf5;
  color: #059669;
  border: 1px solid #bbf7d0;
}

.risk-stable {
  background-color: #eff6ff;
  color: #2563eb;
  border: 1px solid #bfdbfe;
}
```

🔄 Data Flow

1. **Cache**: API fetches ASI data every hour
2. **Predictions**: New predictions at 2:35 PM daily
3. **Historical Data**: Past 10 days included in every prediction
4. **Real Values**: Actual ASI values calculated automatically after 10 days

---

📱 Mobile Considerations

- All endpoints return JSON
- Response times: <500ms (cached data)
- Data usage: ~2-5KB per prediction request
- Offline support: Cache responses for better UX

Last Updated: December 23, 2025
API Version: 1.3.0