

# Software Requirements Specification (SRS)

## Project: Real-Time Chess AI Analyzer

---

### 1. Introduction

#### 1.1 Purpose

The purpose of this system is to build a **real-time chess analysis platform** that evaluates chess positions, suggests optimal moves, and detects blunders using an AI-powered chess engine written in C++. The system will provide an interactive web interface using the MERN stack.

#### 1.2 Scope

The system will:

- Allow users to play chess on a web interface.
- Analyze moves in real-time.
- Suggest best possible moves.
- Display evaluation scores.
- Store games and analysis history.
- Provide a dashboard for reviewing past games.

The core AI logic will be implemented in **C++**, while the web application will be built using **MongoDB, Express, React, and Node.js (MERN)**.

#### 1.3 Definitions

Term	Meaning
------	---------

FEN	Forsyth–Edwards Notation, used to represent board state
-----	---

Evaluation Score	showing advantage for white or black
------------------	--------------------------------------

Blunder	A significantly bad move
---------	--------------------------

Engine	The C++ AI chess analyzer
--------	---------------------------

---

## 2. Overall Description

### 2.1 Product Perspective

The system consists of two major parts:

#### 1. C++ Chess Engine

- Handles move generation and AI evaluation.

#### 2. MERN Web Application

- Provides the user interface.
- Communicates with the C++ engine.
- Stores user and game data.

### 2.2 System Architecture

React Frontend

↓

Node.js API (Express)

↓

C++ Chess Engine

↓

MongoDB Database

### 2.3 User Classes

User Type	Description
Guest	Can analyze a single game without saving
Registered User	Can save games and view history
Admin (optional)	Can manage users and data

---

## 3. Functional Requirements

### 3.1 User Module

- User registration.
- User login/logout.
- Profile management.
- View saved games.

### **3.2 Chessboard Module**

- Interactive chessboard.
- Legal move validation.
- Move history display.
- Reset or start new game.

### **3.3 Real-Time Analysis Module**

- Send board position to backend.
- Receive evaluation from engine.
- Display:
  - Best move
  - Evaluation score
  - Top move suggestions
- Update analysis after each move.

### **3.4 Game Storage Module**

- Save completed games.
- Store:
  - Move history
  - Evaluation data
  - Timestamps
- Load previous games.

### **3.5 Analysis Dashboard**

- Graph of evaluation over time.

- Highlight blunders.
  - Show best moves for each turn.
- 

## **4. C++ Engine Functional Requirements**

### **4.1 Board Engine**

- Parse FEN strings.
- Represent board state.
- Generate legal moves.

### **4.2 AI Evaluation**

- Material balance.
- Piece activity.
- King safety.
- Pawn structure.

### **4.3 Search Algorithm**

- Minimax search.
- Alpha-beta pruning.
- Depth-limited search.

### **4.4 Engine Interface**

- Accept FEN as input.
- Return analysis in JSON format:

Example:

```
{  
  "bestMove": "e2e4",  
  "evaluation": "+0.85",  
  "depth": 18  
}
```

---

## **5. Non-Functional Requirements**

### **5.1 Performance**

- Engine response time:  $\leq 2$  seconds per move.
- Real-time updates after each move.

### **5.2 Scalability**

- Support multiple users simultaneously.
- Modular engine and backend design.

### **5.3 Security**

- JWT-based authentication.
- Password hashing (bcrypt).
- Secure API endpoints.

### **5.4 Usability**

- Simple and intuitive chessboard interface.
- Clear evaluation indicators.

### **5.5 Reliability**

- System uptime  $\geq 95\%$ .
- Proper error handling between Node and C++ engine.

---

## **6. External Interface Requirements**

### **6.1 User Interface**

- Web-based UI.
- Chessboard with drag-and-drop moves.
- Evaluation bar.
- Move suggestions panel.

### **6.2 Software Interfaces**

Component	Technology
Frontend	React
Backend	Node.js + Express
Database	MongoDB
AI Engine	C++
Real-time Communication Socket.io	

---

## 7. Data Requirements

### 7.1 User Collection

- User ID
- Name
- Email
- Password (hashed)
- Game history

### 7.2 Game Collection

- Game ID
  - User ID
  - Move list
  - Evaluation per move
  - Date/time
- 

## 8. System Modules

### C++ Modules

1. Board Representation
2. Move Generator

3. Evaluation Engine
4. Search Algorithm
5. Engine Interface (JSON output)

### **MERN Modules**

1. Authentication Module
  2. Chessboard Module
  3. Real-Time Analysis Module
  4. Game Storage Module
  5. Dashboard Module
- 

## **9. Assumptions and Constraints**

### **Assumptions**

- Users have internet access.
- System runs on modern browsers.

### **Constraints**

- Engine depth limited by server performance.
  - Real-time analysis dependent on network latency.
- 

## **10. Future Enhancements**

- Multiplayer games with analysis.
- Opening book integration.
- Cloud-based deep analysis.
- Mobile application.
- Puzzle generation system.