

# Chess Editor

Devin Michael O'Brien

March 2, 2021

## Contents

<b>1</b>	<b>Project Definition</b>	<b>4</b>
1.1	Revised . . . . .	4
1.2	Original . . . . .	4
<b>2</b>	<b>Project Requirements</b>	<b>4</b>
2.1	Functional . . . . .	4
2.1.1	Primary Requirements . . . . .	4
2.1.2	Secondary Requirements . . . . .	5
2.1.3	Optional Requirements . . . . .	5
2.2	Usability . . . . .	5
2.2.1	User Interface . . . . .	5
2.2.2	Performance . . . . .	5
2.3	System . . . . .	5
2.3.1	Hardware . . . . .	5
2.3.2	Software . . . . .	6
2.3.3	Database                      VERIFY . . . . .	6
2.4	Networking . . . . .	6
2.5	Security . . . . .	6
<b>3</b>	<b>Project Specification</b>	<b>6</b>
3.1	Focus . . . . .	6
3.2	Development Environemnt . . . . .	6
3.2.1	Libraries . . . . .	6
3.2.2	Frameworks . . . . .	6
3.3	Platform . . . . .	7
3.4	Genre . . . . .	7

<b>4</b>	<b>System Design</b>	<b>7</b>
4.1	Subsystem Identification . . . . .	7
4.1.1	Chess Gameplay . . . . .	7
4.1.2	Subsystem Communication . . . . .	7
4.1.3	Sequence Diagram @DEMO . . . . .	8
4.1.4	Use-Case Diagram . . . . .	11
4.1.5	Class Diagram@DAKOTA:@TYLER:@DEMO:@PAWILLIAMSON	11
4.1.6	Entity Relationship Model (E-R Model) . . . . .	12
4.2	Design . . . . .	12
4.2.1	Mock-up Diagram . . . . .	12
4.2.2	Color Schemes . . . . .	12
4.2.3	Additional Comments . . . . .	12
4.3	Sub-System Communication . . . . .	13
4.3.1	Controls . . . . .	13
4.3.2	I/O . . . . .	13
4.3.3	Dataflow @TYLER . . . . .	13
4.4	Entity Relationship Model (E-R Model) . . . . .	13
4.5	Overall operation - System Model\ . . . . .	13
4.5.1	Account Creation Management . . . . .	13
4.5.2	Chess Gameplay Management . . . . .	13
<b>5</b>	<b>TODO System Analysis</b>	<b>13</b>
5.1	Subsystems . . . . .	13
5.2	System (Tables and Description) . . . . .	13
5.2.1	Data Dictionary . . . . .	13
5.2.2	Process Models . . . . .	14
5.3	Algorithm Analysis . . . . .	15
5.3.1	Big-O Analysis . . . . .	15
<b>6</b>	<b>TODO Project Scrum Report</b>	<b>15</b>
6.1	Overall . . . . .	15
6.2	Product Backlog . . . . .	15
6.3	Sprint Backlog . . . . .	15
6.4	Burndown Chart . . . . .	15
6.5	Sprint 1 . . . . .	15
6.5.1	Scrum . . . . .	15
<b>7</b>	<b>TODO Subsystems [/]</b>	<b>16</b>
7.1	<b>TODO</b> Chess Game [0/7] @TYLER:@DAKOTA . . . .	16
7.1.1	<b>TODO</b> Initial Design and Model . . . . .	16

7.1.2	<b>TODO</b>	Data Dictionary . . . . .	16
7.1.3	<b>TODO</b>	Revisions (Refinement) . . . . .	16
7.1.4	<b>TODO</b>	Scrum Backlog . . . . .	16
7.1.5	<b>TODO</b>	Coding . . . . .	16
7.1.6	<b>TODO</b>	User Training . . . . .	16
7.1.7	<b>TODO</b>	Testing . . . . .	16
7.2	<b>TODO</b>	User Authentication [0/7] @PAWILLIAMSON . . .	16
7.2.1	<b>TODO</b>	Initial Design and Model . . . . .	16
7.2.2	<b>TODO</b>	Data Model . . . . .	16
7.2.3	<b>TODO</b>	Refinement . . . . .	16
7.2.4	<b>TODO</b>	Scrum Backlog . . . . .	16
7.2.5	<b>TODO</b>	Coding . . . . .	17
7.2.6	<b>TODO</b>	User Training . . . . .	17
7.2.7	<b>TODO</b>	Testing . . . . .	17
7.3	<b>TODO</b>	Server - Client [0/7] @TYLER . . . . .	17
7.3.1	<b>TODO</b>	Initial Design and Model . . . . .	17
7.3.2	<b>TODO</b>	Data Dictionary . . . . .	17
7.3.3	<b>TODO</b>	Refinement . . . . .	17
7.3.4	<b>TODO</b>	Scrum Backlog . . . . .	17
7.3.5	<b>TODO</b>	Coding . . . . .	17
7.3.6	<b>TODO</b>	User Training . . . . .	17
7.3.7	<b>TODO</b>	Testing . . . . .	17
7.4	<b>TODO</b>	Computer Opponent [0/7] @DEMO . . . . .	17
7.4.1	<b>TODO</b>	Initial Design and Model . . . . .	17
7.4.2	<b>TODO</b>	Data Dictionary . . . . .	17
7.4.3	<b>TODO</b>	Refinement . . . . .	17
7.4.4	<b>TODO</b>	Scrum Backlog . . . . .	17
7.4.5	<b>TODO</b>	Coding . . . . .	18
7.4.6	<b>TODO</b>	User Training . . . . .	18
7.4.7	<b>TODO</b>	Testing . . . . .	18
8	<b>TODO</b>	<b>Complete System</b>	<b>18</b>
8.1	<b>TODO</b>	Final Product . . . . .	18
8.2	<b>TODO</b>	Source code and user manual + Technical Report . .	18
8.2.1	<b>TODO</b>	GitHub . . . . .	18
8.3	<b>TODO</b>	Evaluation by client and instructor . . . . .	18
8.4	<b>TODO</b>	Team Member Description . . . . .	18
8.4.1		Dakota Simpkins . . . . .	18
8.4.2		Tyler Wallshleger . . . . .	18
8.4.3		Devin O'Brien . . . . .	18

8.4.4	Preston Williamson . . . . .	18
8.4.5	Brandon Kyle . . . . .	18

## 1 Project Definition

### 1.1 Revised

Devin's: Chess is a game that has been played by millions since its conception. It is a game that has relatively simple rules that make it which make it a game that has been sort of a standard for testing the computation ability of automata. Although they have become advanced enough where chess is no longer sufficient, we feel it is still a good start for ones' ability. The goal of this project is to develop a chess application using the Angular framework and test some concepts that we are interested in, such as User Authentication with Google Identity and Implementation of an Open-Source Standard like the Unified Chess Interface; with an added bonus of trying to wrangle with an External Web Server with Multiplayer.

### 1.2 Original

People like Chess and need a way to quickly develop and practice new strategies. I love Chess!!! Chess Editor will let people play Chess on their PC and set up the board the way they want to test different strategies, or just play in a different way. It will support playing against an AI, or playing against another person locally. It will include an account system to track wins and losses. For the purposes of testing, there will be an undo and redo button, as well as tools to set up specific board scenarios. It will be programmed in Typescript and we will use Stockfish for the AI.

## 2 Project Requirements

### 2.1 Functional

#### 2.1.1 Primary Requirements

The primary requirements of functionality for **Chess Editor** include:

- Ability for the user to play an online game of chess with another user, or alternatively against **AI** that will be provided.
- Ability for the user to track their **statistics** of play over the life of their account; such as:

- Wins
- Losses
- Draws
- Ability for the user to modify the starting layout of the chess pieces to match the configuration of either a previously ongoing game or a chess scenario (i.e., a game that started offline).

### **2.1.2 Secondary Requirements**

The secondary scope of functionality for Chess Editor include:

- Ability for the end user to create an account to access features
- Ability for the end user to login via their Google Account credentials.

### **2.1.3 Optional Requirements**

## **2.2 Usability**

### **2.2.1 User Interface**

The user interface of the application must not be obtrusive and needs to be adapted for mobile devices.

### **2.2.2 Performance**

The performance of the application must not interfere with other processes on the users system.

## **2.3 System**

### **2.3.1 Hardware**

1. Severside All server-side requirments will be provided by the hosting site: ecowebhosting.co.uk
2. Clientside The application will require hardware capable of running a modern web-brower without difficulty.

### **2.3.2 Software**

1. Clientside The application will require a modern web-browser.
2. Serverside The server will require the database and networking software required of a web server.

### **2.3.3 Database**

**VERIFY**

This project will have one database which is used to keep track of on-going games. This is specifically MySQL version 10.4.14.

## **2.4 Networking**

All project artifacts and interconnectivities are provided by the hosting server.

## **2.5 Security**

For Google Authentication, we will be leveraging the Google OAuth 2.0 API. The other security requirements are inherently provided by the hosting server.

# **3 Project Specification**

## **3.1 Focus**

This project has a focus on developing experience with Angular and with a client-server application. This project will attempt to utilize open-source projects when it is reasonably possible.

## **3.2 Development Environemnt**

### **3.2.1 Libraries**

This project will be using the Google Identity API for user Authentication.

### **3.2.2 Frameworks**

This project will be developed using the Angular framework (11.1.2) with TypeScript.

### **3.3 Platform**

This project is going to be web-based application,

### **3.4 Genre**

This project is considered as an online board game.

## **4 System Design**

### **4.1 Subsystem Identification**

We will be leveraging our MySQL database engine to create, update, read, and delete user information. Google Identity will also be used to facilitate account creation with a user's existing Google account.

#### **4.1.1 Chess Gameplay**

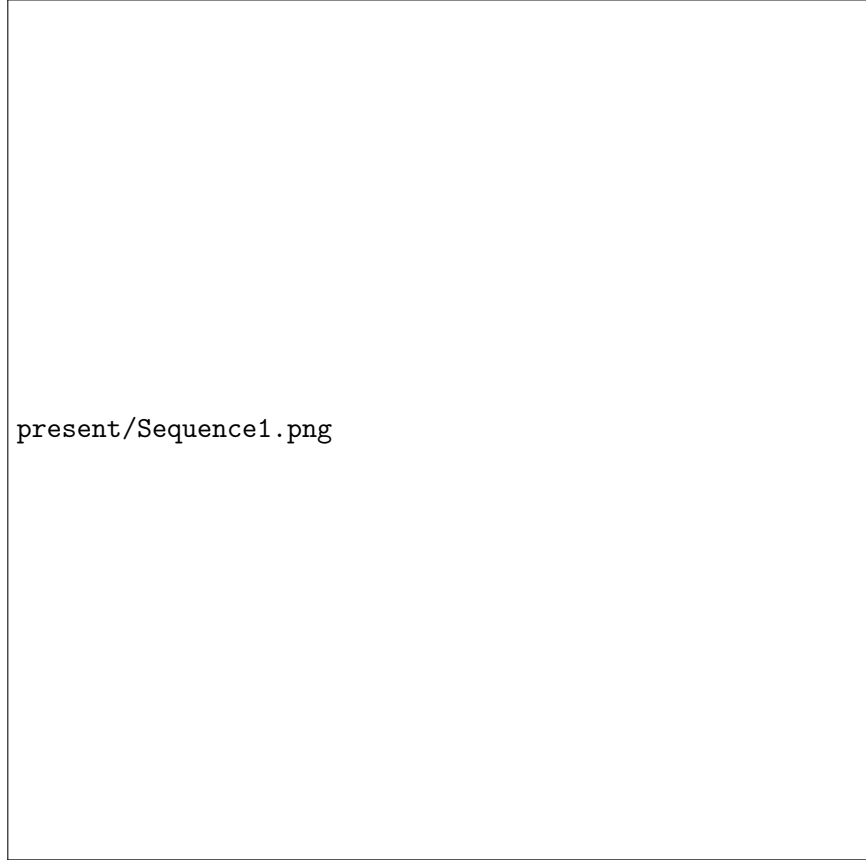
The MySQL database engine will be used for tracking gameplay metrics. The Angular framework will be implemented with TypeScript to manage the gameplay logic and for validation of chess scenarios. The Stockfish AI framework will be used to respond to the end player's movement when playing against the "computer".

#### **4.1.2 Subsystem Communication**

The player will interact with the Chess UI with their keyboard and mouse. The player will notice a response from the UI with their monitor. The system should have proper UI responses indicating a success/failure of the attempted action.

#### 4.1.3 Sequence Diagram

@DEMO



present/Sequence1.png



present/Sequence2.png

present/Sequence3.png

#### 4.1.4 Use-Case Diagram

#### 4.1.5 Class Diagram@DAKOTA:@TYLER:@DEMO:@PAWILLIAMSON



#### 4.1.6 Entity Relationship Model (E-R Model)



[https://lucid.app/lucidchart/b4d167c8-e8fa-475e-9e63-86c4e3aed414/view?page=0\\_0#](https://lucid.app/lucidchart/b4d167c8-e8fa-475e-9e63-86c4e3aed414/view?page=0_0#)

## 4.2 Design

### 4.2.1 Mock-up Diagram

### 4.2.2 Color Schemes

### 4.2.3 Additional Comments

We will be using Bootstrap framework.

### **4.3 Sub-System Communication**

#### **4.3.1 Controls**

#### **4.3.2 I/O**

#### **4.3.3 Dataflow**

@TYLER

### **4.4 Entity Relationship Model (E-R Model)**

### **4.5 Overall operation - System Model\**

#### **4.5.1 Account Creation Management**

The user will modify their account information via the Chess Game UI that will be fed into their respective records in the database.

#### **4.5.2 Chess Gameplay Management**

Gameplay metrics will be automatically tracked by the Chess Game UI in tandem with pre-existing records stored in the MYSQL database. Gameplay involving the "computer" will be handled by the Stockfish AI program to determine the next best move. <https://lucid.app/lucidchart/invitations/accept/18b1143a-20e9-441c-ac34-2c95f7a2d031>

## **5 TODO System Analysis**

### **5.1 Subsystems**

- Account Creation/Management
- Chess Gameplay Management

### **5.2 System (Tables and Description)**

#### **5.2.1 Data Dictionary**

Table	Column	Data Type	Description
User	userID (Pky)	int	UUID of user
	firstName	varchar	User's first name
	lastName	varchar	User's last name
	wins	int	Number of user's wins
	losses	int	Number of user's losses
	activeGameID (fky)	int	UUID of currently active game
Game	gameID (Pky)	int	UUID of current game session
	turnList	List<Peice, Integer, Integer>	UUID of recorded turn
	playerOne (fky)	int	UUID of player one (white)
	playerTwo (fky)	int	UUID of player two (black)

### 5.2.2 Process Models



[https://lucid.app/lucidchart/7d69e18e-721a-43ad-a77d-83df4e8d1f3a/view?page=0\\_0#](https://lucid.app/lucidchart/7d69e18e-721a-43ad-a77d-83df4e8d1f3a/view?page=0_0#)

## 5.3 Algorithm Analysis

### 5.3.1 Big-O Analysis

We expect that the game when not playing with an AI will run in constant time as there is nothing in our current algorithms that will execute a variable number of times. This will probably change when the AI is implemented.

## 6 TODO Project Scrum Report

### 6.1 Overall

### 6.2 Product Backlog

### 6.3 Sprint Backlog

### 6.4 Burndown Chart

### 6.5 Sprint 1

Sprint 1 began on January 22, 2021 and continued to February 6, 2021. The period lasted one day longer than the allocated duration.

#### 6.5.1 Scrum

During sprint, two scrum meetings took place

- January 28, 2021: Discussed the framework of the project and decided to use Angular. Discussed the scope of the project and decided to be a web application. Discussed authentication services for the server.
- February 4, 2021: Discussed some work that was done since the previous scrum; includes diagrams and investigations of Google Authentication viability for the server.

Item	Created BY	Date	Status
Project Definition	dobrienUNCG	01/21/21	Completed by pizzaza
Project requirements	dobrienUNCG	01/21/21	Completed during Scrums 1 and 2 by group
Identify subsystems	dobrienUNCG	01/21/21	Moved to Sprint 2 backlog
Project Specification	dobrienUNCG	01/21/21	Moved to sprint 2 backlog

## 7 TODO Subsystems [/]

### 7.1 TODO Chess Game [0/7] @TYLER:@DAKOTA

#### 7.1.1 TODO Initial Design and Model

#### 7.1.2 TODO Data Dictionary

#### 7.1.3 TODO Revisions (Refinement)

#### 7.1.4 TODO Scrum Backlog

Task	On	Assigned To	Completed On
Generate Chessboard		Tyler, Dakota	
Chess Pieces			
Movement			
Movement and player interfaces			
Display Board			
Drag and move piece			
Validate Moves			
Detect Check			
Detect Win			

1. **TODO** User Story Categories @DEMO

#### 7.1.5 TODO Coding

1. Language

#### 7.1.6 TODO User Training

#### 7.1.7 TODO Testing

### 7.2 TODO User Authentication [0/7] @PAWILLIAMSON

#### 7.2.1 TODO Initial Design and Model

#### 7.2.2 TODO Data Model

#### 7.2.3 TODO Refinement

#### 7.2.4 TODO Scrum Backlog

1. **TODO** User Story Categories @DEMO



- 7.2.5 TODO Coding
- 7.2.6 TODO User Training
- 7.2.7 TODO Testing
- 7.3 TODO Server - Client [0/7] @TYLER
  - 7.3.1 TODO Initial Design and Model
  - 7.3.2 TODO Data Dictionary
  - 7.3.3 TODO Refinement
  - 7.3.4 TODO Scrum Backlog
  - 7.3.5 TODO Coding
  - 7.3.6 TODO User Training
  - 7.3.7 TODO Testing
- 7.4 TODO Computer Opponent [0/7] @DEMO
  - 7.4.1 TODO Initial Design and Model
  - 7.4.2 TODO Data Dictionary
  - 7.4.3 TODO Refinement
  - 7.4.4 TODO Scrum Backlog
- 1. TODO User Story Categories

#### **7.4.5 TODO Coding**

#### **7.4.6 TODO User Training**

#### **7.4.7 TODO Testing**

### **8 TODO Complete System**

#### **8.1 TODO Final Product**

#### **8.2 TODO Source code and user manual + Technical Report**

##### **8.2.1 TODO GitHub**

#### **8.3 TODO Evaluation by client and instructor**

#### **8.4 TODO Team Member Description**

Our team consists of five members: Dakota Simpkins, Tyler Wallshleger, Devin O'Brien, Preston Williamson, and Brandon Kyle.

##### **8.4.1 Dakota Simpkins**

##### **8.4.2 Tyler Wallshleger**

##### **8.4.3 Devin O'Brien**

##### **8.4.4 Preston Williamson**

##### **8.4.5 Brandon Kyle**