LearningEDU

Version <1.0>

Table of Contents

1. Introduction

1.1 Product Vision

FOR enthusiastic players of chess all the way from brand new to intermediate levels WHO wish to learn the game at their own pace incrementally, THE ChessLingo App is an educational service THAT offers small, accessible, and interactive lessons that teach users the basics and the very heart of the game. UNLIKE other chess products, services, or software applications, such as Chess.com, the focus of ChessLingo is bite-sized interactive lessons, rather than long and complex modules paired with practice, OUR PRODUCT provides an easy way to learn this intimidating game through not just theory, but practice using lessons that give the user strategy piece by piece and present opportunities to apply what they have learned immediately.

1.2 Profiles

Name: Grant Jones
Phone Number: (913)645-4050
Email: g641j712@ku.edu
Available times for team meetings: MWF 8am-1pm, TR 8am-2:30pm
Major: Interdisciplinary Computing with an Emphasis in Biology
Year: Junior
Relevant Courses: EECS 168, 268, 368
Proficient Programming Languages: C++, JavaScript, Java, and HTML
Hobbies: Watching and playing sports

Name: Joe Murray
Phone Number: 913-269-0760
Email: j604m256@ku.edu
Available times for team meetings: MW 11-1, TuesThur 9:30-10:30, 12-2
Major: Computer Science
Year: Senior
Relevant Courses: EECS 168,268,368,
Proficient Programming Languages: C++, javascript, python
Hobbies: Basketball, bowling, martial arts

Name: Adair Torres
Phone Number: (620) 640-7414
Email: adair.tor24@ku.edu
Available times for team meetings: M: 10am-10pm, TuTh: 4pm-10pm, W: 4pm-10pm, F: 6pm-10pm
Major: Computer Science
Year: Junior
Relevant Courses: EECS 168, EECS 268, EECS 368
Proficient Programming Languages: C++, JavaScript, C#, Python, HTML
Hobbies: Tabletop & digital games, digital art, weightlifting, racquetball

Name: Jack Reynolds
Phone Number: (913)-634-0412
Email: jackreynolds@ku.edu
Available times for team meetings: M: 8pm - 10 pm, TWRF: 6pm - 10pm, flexible Saturday/Sunday
Major: Computer Science
Year: Junior
Relevant Courses: EECS 168, EECS 268, EECS 368
Proficient Programming Languages: C++, C, JavaScript, HTML
Hobbies: Playing saxophone, tabletop and video games, cooking

Name: Rylan DeGarmo
Phone Number: (316) 796-3719
Email: r031d544@ku.edu
Available times for team meetings: M/F 8am-9am, 4pm-10pm | Tu/Th 8am-10am, 4pm-10pm | W 8am-9am, 12pm-10pm | Sa/Su 8am-10pm
Major: Computer Science
Year: Junior
Relevant Courses: EECS 168, EECS 268, EECS 140, EECS 210
Proficient Programming Languages: C++, HTML
Hobbies: Youtube, video games

Name: Chinh Nguyen
Phone Number: (620) 277-6337
Email: nguyenchinh@ku.edu
Available times for team meetings: M-W-F 5PM-10PM, Flexible Saturday and Sunday
Major: Computer Science
Year: Junior
Relevant Courses: EECS 168, EECS 268, EECS 368
Proficient Programming Languages: Python, JavaScript, C++, HTML
Hobbies: Video games, weight lifting, art, music

1.3 Roles and Responsibilities

Name: Grant Jones
Role: Project Manager
Responsibilities: Responsible for providing up-to-date status of team progress, managing the team meetings, and maintaining a record (minutes or log) of each meeting (e.g., when, purpose, who attended, etc.).

Name: Chinh Nguyen
Role: Quality Assurance Engineer
Responsibilities: Responsible for the final quality of each artifact, e.g., technical accuracy, but also uniformity
in typesetting (consistency of font sizes, margins, colors), correct spelling and grammatically correct sentences, adhering to the templates, checking for consistency among deliverables, etc.

Name: Rylan DeGarmo
Role: Project Leader
Responsibilities: Responsible for compiling "original project deliverables" which has been accomplished by all team members, directing the project and leading project portion meetings, and reporting to the professor project technical issues not resolvable within the team.

Name: Joe Murray
Role: Lesson Developer
Responsibilities: Responsible for the creation of lesson modules, components, and teaching methodology. Must present information in a user-friendly format. Works closely with the Data Administrator to meet storage constraints.

Name: Adair Torres
Role: Data Administrator
Responsibilities: Responsible for development of file server system backend, which may track user lessons, store login credentials, and organize and deliver files required by individual lessons.

Name: Jack Reynolds
Role: UI / Accessibility Developer
Responsibilities: Responsible for the development of frontend software, user interfaces and displays. Emphasis on creating user assets that can be used without significant difficulty.

2. Specifications and Requirements

2.1 Software Requirements Specifications

2.1.1. Goals

- 2.1.1.1. Users will be able to learn what is widely considered the 'basics' of chess such as individual piece movement, attacking patterns, checks, etc.
- 2.1.1.2. Users will be able to learn what is widely considered the more 'technical' and 'advanced' aspects of chess, such as defending, pins and skewers, openings, en passant, etc.
- 2.1.1.3. User will learn the main line of two openings for both white and black

2.1.2. Lessons

- 2.1.2.1. Each lesson will be interactive
  - 2.1.2.1.1. At a minimum the lesson will end with a way for the user to apply or demonstrate acquired knowledge.
- 2.1.2.2. Each module ends with a cumulative review

2.1.3. Settings

- 2.1.3.1. User can toggle the following settings:
  - 2.1.3.1.1. Highlighting possible moves after selecting a piece
  - 2.1.3.1.2. Highlighting of hanging pieces.
  - 2.1.3.1.3. Highlighting of pinned pieces.
  - 2.1.3.1.4. Highlighting of pieces in or entering capture spaces.
  - 2.1.3.1.5. User will be notified when a player is in check
  - 2.1.3.1.6. Confirm each move
  - 2.1.3.1.7. Auto-Queen when pawn promotes
  - 2.1.3.1.8. Highlighting last move
  - 2.1.3.1.9. Sounds
  - 2.1.3.1.10. Change theme

2.1.4. Playing the game

- 2.1.4.1. A pair of users can play a game on a single device.
  - 2.1.4.1.1. During the game, the user has the option to resign, offer draw, see previous moves, accept/decline draw
    - 2.1.4.1.1.1. If a player offers a draw, the other player will have the option to accept or decline the draw
      - 2.1.4.1.1.1.1. If the player accepts the draw, then a popup will state “Draw” with option to play again or go back to home screen
      - 2.1.4.1.1.1.2. If the player declines the draw, then the other player will be notified that the player decline the draw, and the game will resume
  - 2.1.4.1.2. After checkmate, resignation, or stalemate, a popup will state “Black

wins", "White wins", or "Draw" with option to play again or go back to home screen

2.1.5. Home screen

2.1.5.1. Home screen will have option to play a game or go to lessons

2.1.5.1.1. If clicked play a game, then the user will be provided a board to play with a friend locally

2.1.5.1.2. If clicked lessons, then the next recommended lesson will pop up with the remaining lessons underneath

2.2 Use Case Specifications

2.2.1. Account Creation

- 2.2.1.1. Actor is an unregistered user
- 2.2.1.2. Server needs to request email, username, password
- 2.2.1.3. A user account with this information is created in the database afterwards
- 2.2.1.4. Alternative Flow: Login is already taken
  - 2.2.1.4.1. The process needs to loop to prevent the actor from progressing…
  - 2.2.1.4.2. …until unique account information is entered
- 2.2.1.5. Alternative Flow: Actor is signed in
  - 2.2.1.5.1. This specific actor should not be allowed to enter this system
  - 2.2.1.5.2. Prompt them if they want to sign out instead

2.2.2. Signing In

- 2.2.2.1. Actor is a registered user that is not logged in
- 2.2.2.2. The actor needs to be prompted with a log-in page
- 2.2.2.3. Sign-in page needs to contact the database to validate the user
- 2.2.2.4. Actor should be redirected to a menu with their account specific features
- 2.2.2.5. Alternative Flow: Actor is signed in
  - 2.2.2.5.1. This specific actor should not be allowed to enter this system
  - 2.2.2.5.2. Prompt them if they want to sign out instead
- 2.2.2.6. Alternative Flow: Actor enters invalid information
  - 2.2.2.6.1. Interface needs to prompt them about *what* is invalid (ex: wrong password)
  - 2.2.2.6.2. The interface needs to prevent the actor from progressing

2.2.3. Signing Out

- 2.2.3.1. Actor is a registered user that is logged in
  - 2.2.3.1.1. Any other user should not be able to see this option
- 2.2.3.2. Server is contacted that the user wants to log out
- 2.2.3.3. Actor is redirected to the view of the page a logged out user can see

2.2.4. Selecting a Lesson

- 2.2.4.1. Actor is any user (unregistered/registered)
- 2.2.4.2. Actor needs access to a catalog of available lessons
  - 2.2.4.2.1. Server needs to gather lessons from the database that contains them…
  - 2.2.4.2.2. …handling any connection issues in the process
- 2.2.4.3. Actor can choose to enter a lesson, which hands control to a new display
- 2.2.4.4. As a lesson progresses, the interface includes different options displayed to the user:
  - 2.2.4.4.1. A user can flip from one page to the next/previous page
  - 2.2.4.4.2. Interface accesses pages from the server database

- 2.2.4.4.3. Pages need to display text to the user in an easily readable manner
- 2.2.4.4.4. Interactive elements (such as small chess boards):
  - 2.2.4.4.4.1. User can move pieces in real time
  - 2.2.4.4.4.2. Element provides feedback (ex: “good move!”) after an action
  - 2.2.4.4.4.3. Element accurately updates the position of pieces
  - 2.2.4.4.4.4. A piece can be “locked” so the user cannot move it
    - 2.2.4.4.4.4.1. Ex: the piece belongs to the opponent
    - 2.2.4.4.4.4.2. Ex: the lesson is teaching you a strategy with one of the pieces

2.2.4.5. Actor can leave the lesson at any point

- 2.2.4.5.1. And save progress if logged in

2.2.5. Playing On Your Own

2.2.5.1. Actor is any user (unregistered/registered)

2.2.5.2. Actor can leave this page at any time

2.2.5.3. For Local Games:

- 2.2.5.3.1. Control alternates between a “white” and a “black” player (white goes first)
  - 2.2.5.3.1.1. The other player’s pieces are locked during one player’s turn
- 2.2.5.3.2. Player is prompted that it is their move
- 2.2.5.3.3. Player can move a piece to a position
  - 2.2.5.3.3.1. If the movement is invalid for that piece:
    - 2.2.5.3.3.1.1. Do not update the board
    - 2.2.5.3.3.1.2. Prompt the player that the move is invalid
    - 2.2.5.3.3.1.3. Give the player the ability to move again
  - 2.2.5.3.3.2. If the movement is valid for that piece:
    - 2.2.5.3.3.2.1. Update the board
- 2.2.5.3.4. If the movement captures a piece:
  - 2.2.5.3.4.1. The piece is removed from the board
- 2.2.5.3.5. Control shifts to the opposing player when the current player makes a valid move
- 2.2.5.3.6. If the player is in “check”:
  - 2.2.5.3.6.1. A different set of movement rules should be applied…
  - 2.2.5.3.6.2. …so that the user has to address the threat
- 2.2.5.3.7. The interface should check if the board state after a move produces either:
  - 2.2.5.3.7.1. A checkmate
    - 2.2.5.3.7.1.1. In other words: There is no way to prevent the King from being taken
    - 2.2.5.3.7.1.2. The player who last moved is the “winner” of the game
  - 2.2.5.3.7.2. A stalemate
    - 2.2.5.3.7.2.1. In other words: The player up to move has no choices for movements
    - 2.2.5.3.7.2.2. The interface should prompt the players that there is a draw

2.2.6. Editing User Settings

- 2.2.6.1. Actor is a registered user
  - 2.2.6.1.1. All other actors should not have access to this menu
- 2.2.6.2. The actor should be prompted with a list of settings
- 2.2.6.3. When the actor updates a setting, the server should receive the request…
- 2.2.6.4. …and save the change to the database
- 2.2.6.5. If a setting is sensitive (“change password”, “update email”):
  - 2.2.6.5.1. The actor should be prompted to validate their information
    - 2.2.6.5.1.1. After a set number of attempts, they should be signed out
  - 2.2.6.5.2. Email confirmation or specific menus may be necessary/helpful

2.3 Supplementary Specifications

2.3.1. Client / Server

- 2.3.1.1. The user interacts with the system through a mobile or a web browser
- 2.3.1.2. Either will execute HTML and Java queries to the web engine using JSP
- 2.3.1.3. The engine communicates with the database using SQL
- 2.3.1.4. The database returns the data to the engine which, in turn, returns the resulting package to the browser or application, displaying the results afterwards.

2.3.2. Security

- 2.3.2.1. Individual users have the option to create accounts within the system for tracking personal progress and data.
- 2.3.2.2. Account credentials will need to be stored within the database, made publicly inaccessible, and encrypted to guarantee security.
- 2.3.2.3. No information beyond bare minimum identification and module progress will be stored to minimize the impact of a data leakage should security measures fail.

2.3.3. Accessibility

- 2.3.3.1. System should include accessibility the certain groups of users depend on in order to utilize any software
  - 2.3.3.1.1. This may include and is not limited to:
    - 2.3.3.1.1.1. Text narration
    - 2.3.3.1.1.2. Screen readers
    - 2.3.3.1.1.3. Variable font size
    - 2.3.3.1.1.4. Audio captions
    - 2.3.3.1.1.5. Color correction

3. Notes

3.1 Meeting Log

Date	Time	Description	Attendance
9/10/22	75 minutes	Fill out profiles, discuss roles, and brainstorm project ideas	Adair, Jack, Rylan, Chinh, and Grant
9/17/22	40 minutes	Decide what project we're doing and create a product vision statement	Jack, Rylan, Chinh, Joe, Adair, and Grant
9/28/22	85 minutes	Discussing outline for use case requirements	Jack, Rylan (small group)
10/1/22	35 minutes	Finalizing first iteration of requirements, clarification on individual work	Adair, Jack, Rylan, Chinh

3.2 Glossary

3.2.1. Lesson: Short interactive experience for the user to learn something new about chess
3.2.2. Module: Group of lessons
3.2.3. Unregistered User: A user who doesn’t have an account
3.2.4. Registered User: A user who does have an account
3.2.5. Check: The King is threatened by an opposing piece