Team A0: Tactile Chess

Edison Aviles, Mukundh Balajee, Juan Mejia

Problem Statement/Use Case

- Online chess inaccessible to the visually-impaired
  - Develop smart chess board
- Using a physical chess board to understand online gameplay
  - Useful for beginner and novice players
- Provide tactile and vocal cues
  - Quick transition between online platform and physical board
- Smooth gameplay experience between the online and physical board

Case Requirements - Revisited

Case	Requirements
User Experience	60 sec setup time - Minimum 4 hours battery life - Maximum latency of 1 second
Piece Detection & Board Integrity	- Differentiate between colours and pieces - Aim to achieve 100% accuracy - Use sensors to verify board state and move legality
Accuracy & Latency	- Full system latency max 1 second - Accuracy of piece detection: 100%

Solution Approach - Accessibility

Barrier	Solution
Identification of Pieces	● Pieces will have different textures and braille using 3D printing techniques
Identification of Opponent Moves	● Piece location clear vocalization through speakers on the chosen device (based on standard chess coordinates)
Set up convenience	● Integrated buttons on the board to start/end/reset games

Solution Approach- Board & Piece Design

- Custom design pieces on 3D Modeling Softwares
  - Add Braille notations on the stem of pieces
- Custom design board with etched and raised tiles
  - Add Braille notations for the coordinates
- Have a lock and key mechanism between piece and board
  - Pieces will have pegs and board will have holes
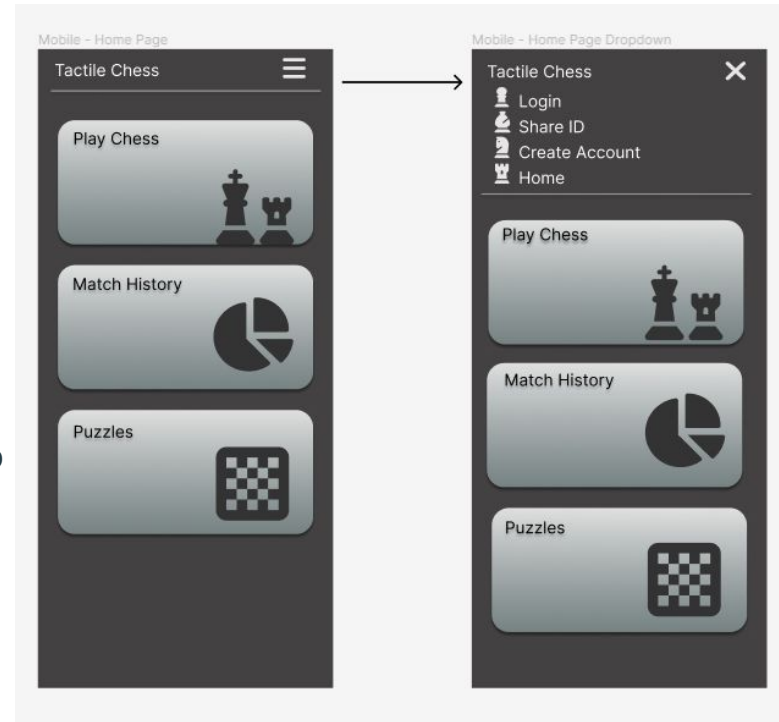
Solution Approach - Piece Detection, Board Integrity

- Use Hall Effect sensors to detect pieces
  - Switch polarity of magnets for opposing teams
  - Control strength of magnet for each unique piece
- Validate moves with the help of the Stockfish API (done on RPi)
- Maintaining a state of the board and recording moves
  - Store moves list for training purposes
- Ensuring physical board and online game are identical
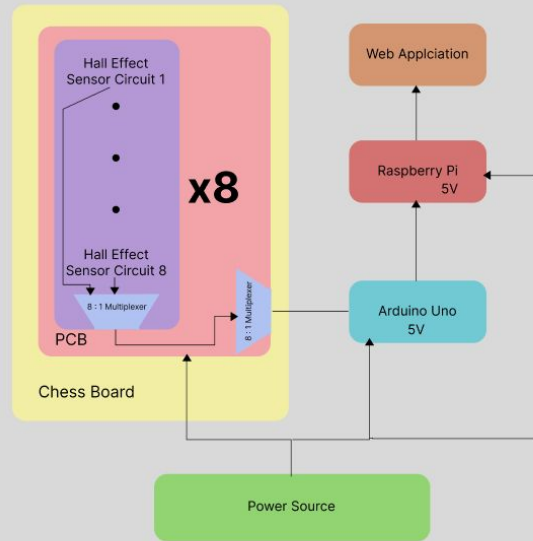
Solution Approach - Website

- Use reactjs frontend and nodejs backend
  - Nodejs backend makes calls to lichess.org API and authenticate users
- Authenticate users through lichess.org or other OAuth APIs
- Seek game and display game once connection is established
- The game/board status streamed through pipeline to RPi
  - Based on updates user receive vocalized cues
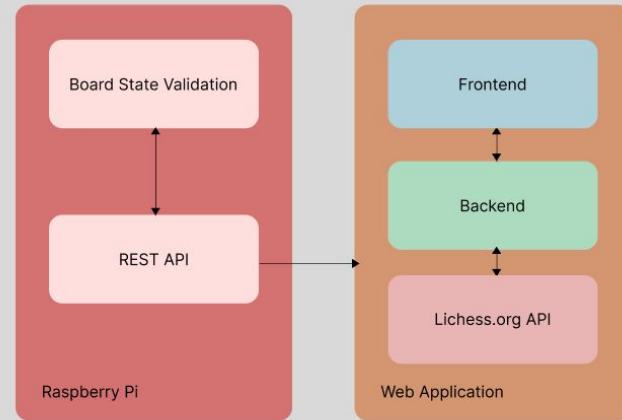 - RPi will be a central data hub for information from board and online game

Block Diagram

Implementation Plan

	Materials	System Design
Hardware	● Hall effect sensors ● Magnets (varying sizes and shapes) ● Resistors, wires, … (for PCB testing) ● Arduino ● Raspberry Pi ● Multiplexers	● PCB ● Arduino (Analog to Digital) ● RPi (integrity checks & web app communication)
	Software Stack	**System Design**
Software	● ReactJS, NodeJS ● Python, Stockfish ● Lichess.org	● Web application ● Board integrity check ● Chess API

Test, Verification, and Validation

Requirement	Test	Mitigation of Failure
Accessibility	● Compare ease of navigation of blindfolded vs not blindfolded games	● Potentially increase the number of tactile as well as vocal cues
Board Integrity	● Compare live board state to expected state	● Identify what is causing inaccuracies - vary magnet placement, etc.
Latency	● Test the latency time of each individual sub-system and the entire system cycle	● Identify latency bottlenecks and mitigate based on what is identified
Battery Life	● Measure power usage while system is in idle state and while the system is in use	● Visit different battery options, determine if power usage can be lowered

Project Management/Schedule
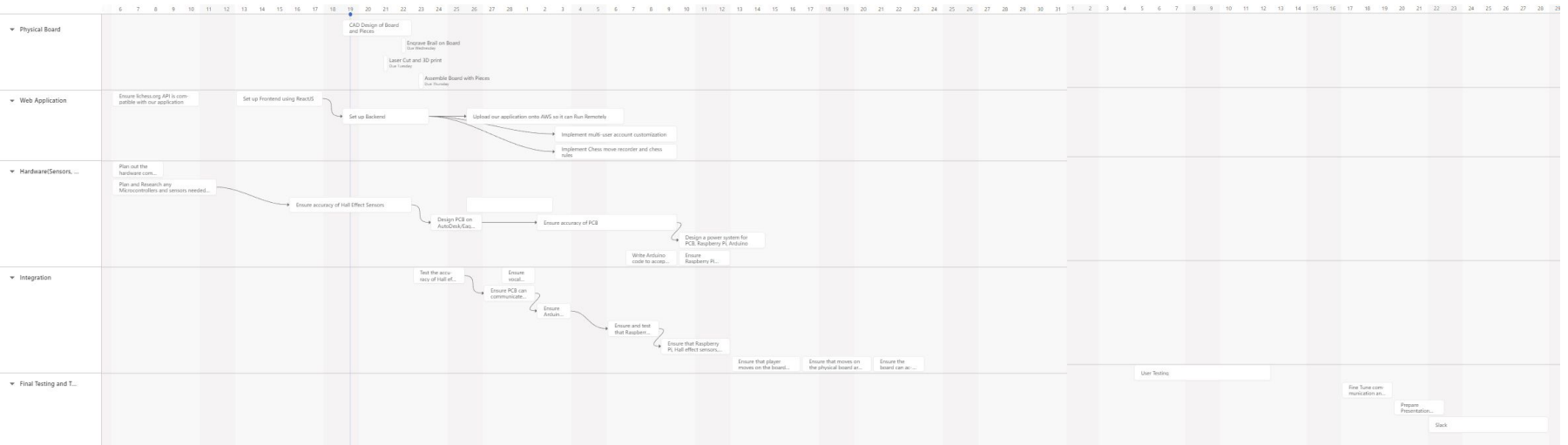
Post-MVP Plan

- Add LED system on board to train beginners
- Access and setup puzzles for training purposes
- Access past games details