Shipmnts SDE Task Round

This task is designed to test your web development skills. You can either work on a backend task or a frontend task. There are multiple options in each. **Choose either one** and try to finish the same to the best of your ability.

Timeline: 3 hours

Examination Guidelines:

1. You are allowed to access the internet to search for code syntax.
2. ***Please avoid plagiarism at all costs. If found, it will result in the immediate disqualification.***

Submission Guidelines:

1. Create a github repo and push everything there before the deadline. Any commits after the deadline will not be considered.
2. Please add a readme in the repo which explains how to setup and run the project
3. Please make progressive commits to the repo rather than pushing all at once.
4. If possible, deploy your code in a cloud environment of your choice. For Backend, we recommend using vercel.com along with hosted postgres database or mongodb atlas for NoSQL. For frontend, you can deploy the same on Netlify or any other platform.
5. Once done, please fill up the form hosted here, and update the github and app links here.

Task 1: Implement Data Import Functionality for Book and Auther

Objective:

Implement a feature that allows users to upload an Excel sheet containing company and contact data. Upon upload, display the data in a table for review and confirmation before storing it in the database.

Requirements:

1. **Upload Feature**: Allow users to upload an Excel (.xls or .xlsx) file containing data for companies and contacts:
2. **Author Has Many Book (sample Excel Link)**

   Book Model.

   - Name
   - ISBN Code
   - Author Id

   Author

   - Name
   - Email
   - Date of Birth
3. **Display Table**: Display the contents of the uploaded Excel sheet in a table format for user review.
4. **Database Storage**: Store the validated data into the database after user confirmation.
5. **Data Validation**: Validate the uploaded data to ensure it meets the required format and constraints (e.g., data types, mandatory fields).
6. **Error Handling**: Handle errors gracefully and provide feedback to users in case of invalid data or other issues during the upload process.

Steps to Implement:

1. **Frontend Implementation**:
   - Create a web page with a file upload form.
2. **Backend Implementation**:
   - Set up a backend server using a framework.
3. **Data Parsing and Validation**:
   - Parse the uploaded Excel file to extract company and contact data.
   - Validate each row of data to ensure it matches expected formats (e.g., string for name, numeric for phone number).
   - Handle missing or invalid data appropriately.
4. **Database Integration**:
   - Prepare database schema and tables to store company and contact data.
5. **User Interface**:
   - After parsing and validating the data, display it in a table format on the web page for user review.
   - Provide options to confirm or cancel the upload.
   - Implement functionality to store the data in the database upon user confirmation.
6. **Error Handling and Feedback**:

    - Implement error handling for file upload failures, data validation errors, and database storage issues.
    - Provide appropriate error messages and feedback to users to guide them through the process.

Deliverables:

- Fully functional web application with file upload functionality.
- Clear documentation on how to use the application, including instructions for uploading Excel files, reviewing data, and confirming uploads.

Bonus:

- Ensure compatibility with different Excel versions (.xls and .xlsx).
- Consider security aspects such as file size limits, file type validation, and sanitization of user inputs.
- Provide a user-friendly interface with clear instructions and feedback messages (validations).

Task 2:Frontend Chess Task Requirements

1. Task Overview:

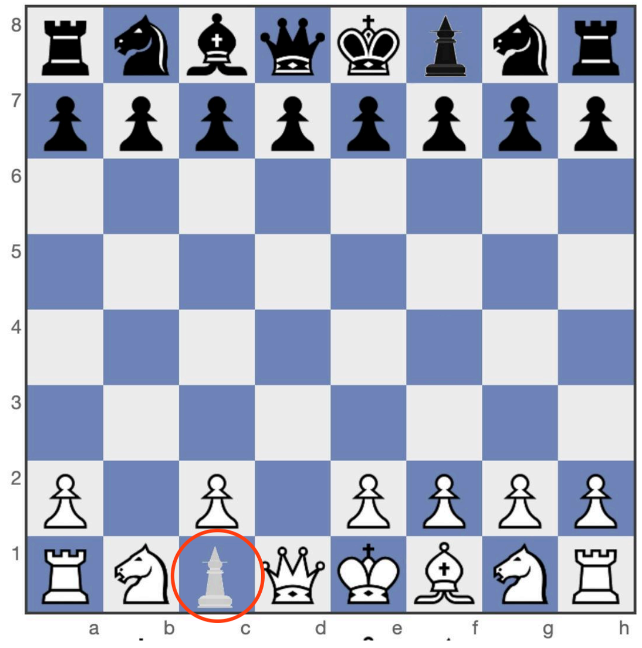
Candidates are required to build a web-based chessboard interface using a frontend framework or vanilla JavaScript. The interface should allow users to:

- Visualize and simulate the movement of standard chess pieces (pawn, rook, knight, bishop, queen, king).
- Work with a pre-defined custom chess piece called the "Archer."

2. Core Requirements:

A. Chessboard Setup:

- Implement an 8x8 chessboard with standard grid coordinates (e.g., A1 to H8).
- The board should be responsive and visually appealing, with clear differentiation between white and black squares.
- Intial Board(**You can use alphabates if you unbale to find piece svg**)

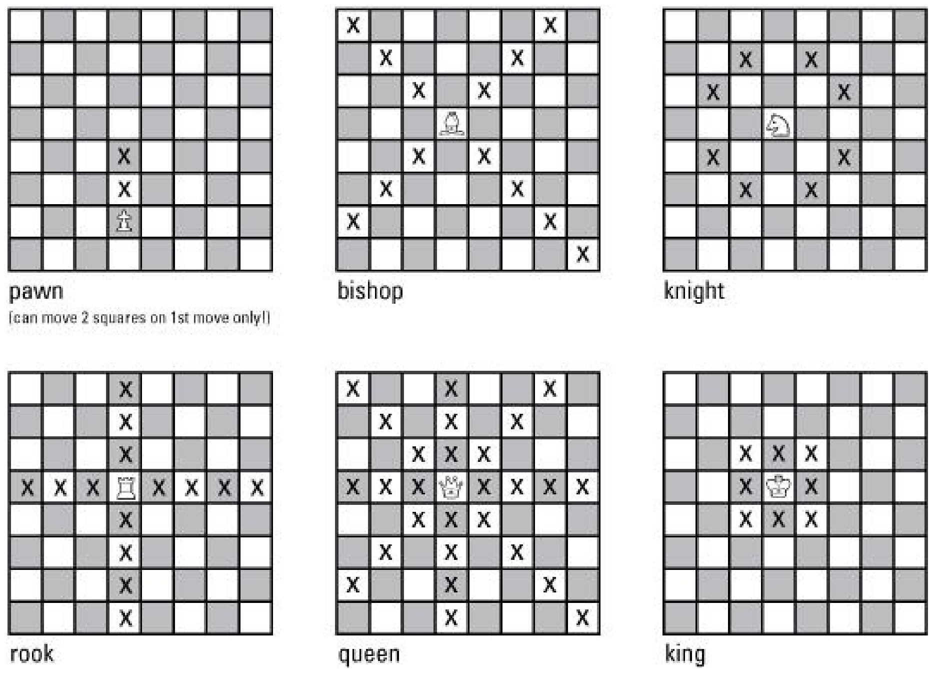


-

-

B. Piece Movements:

Implement the movement rules for all standard chess pieces:

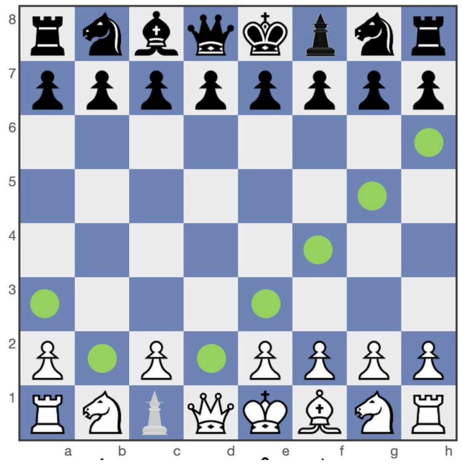


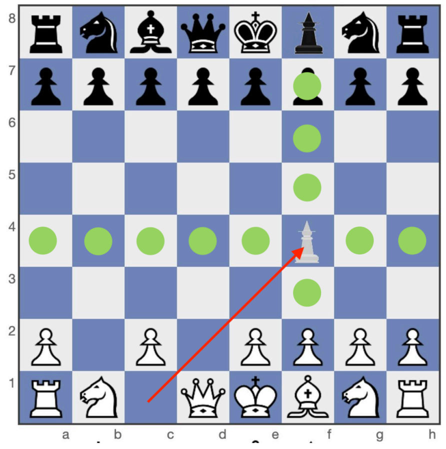
-
    - **Pawn:** Moves one square forward, captures diagonally.
    - **Rook:** Moves any number of squares horizontally or vertically.
    - **Knight:** Moves in an "L" shape.

- **Queen:** Combines the movement of rook and bishop.
- **King:** Moves one square in any direction.

Replace one of the Bishops with the custom piece:

Archer: This piece alternates its movement pattern with each turn:





-
    - On the first turn, it moves like a Bishop (any number of squares diagonally).
    - On the next turn, it moves like a Rook (any number of squares horizontally or vertically).
    - It continues alternating between these two movement styles on subsequent turns.

C. Turn-Based Logic:

- Implement logic to track turns and ensure that the Archer's movement alternates correctly between Bishop and Rook styles with each move.

3. Evaluation Criteria:

- **Code Quality:** Clean, maintainable, and well-documented code.
- **User Experience:** Intuitive design, responsive layout, and clear movement visualization.
- **Functionality:** Accurate implementation of piece movements and turn-based logic.
- **Creativity:** Effective implementation of the Archer's alternating movement pattern.

4. Submission Guidelines:

- The candidate should submit the completed project as a GitHub repository.
- Include a README.md file with instructions on how to set up and run the project.
- Provide a brief description of the approach and any challenges faced.