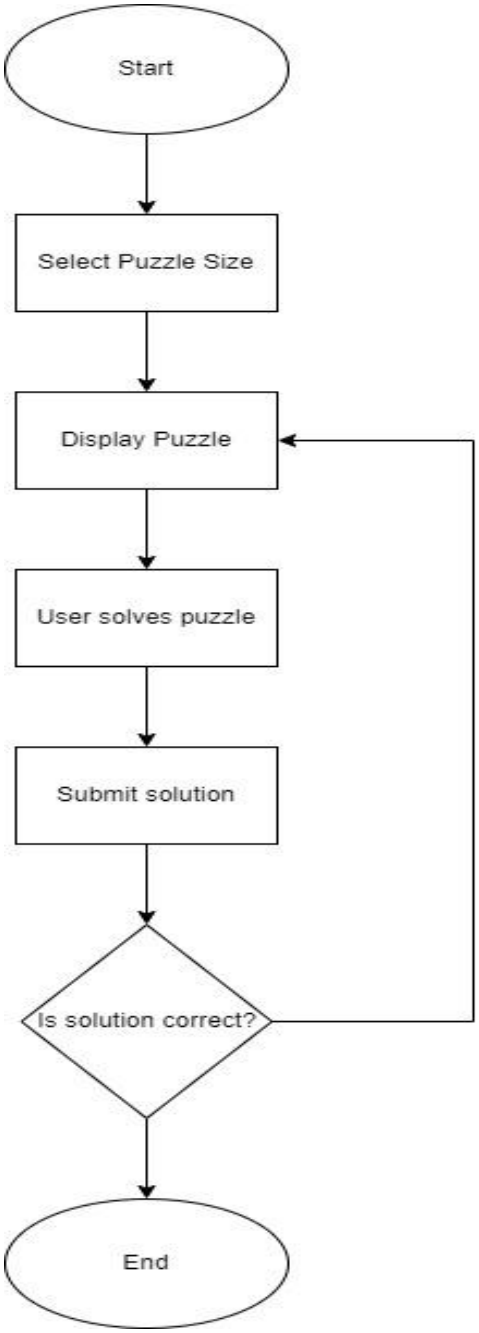
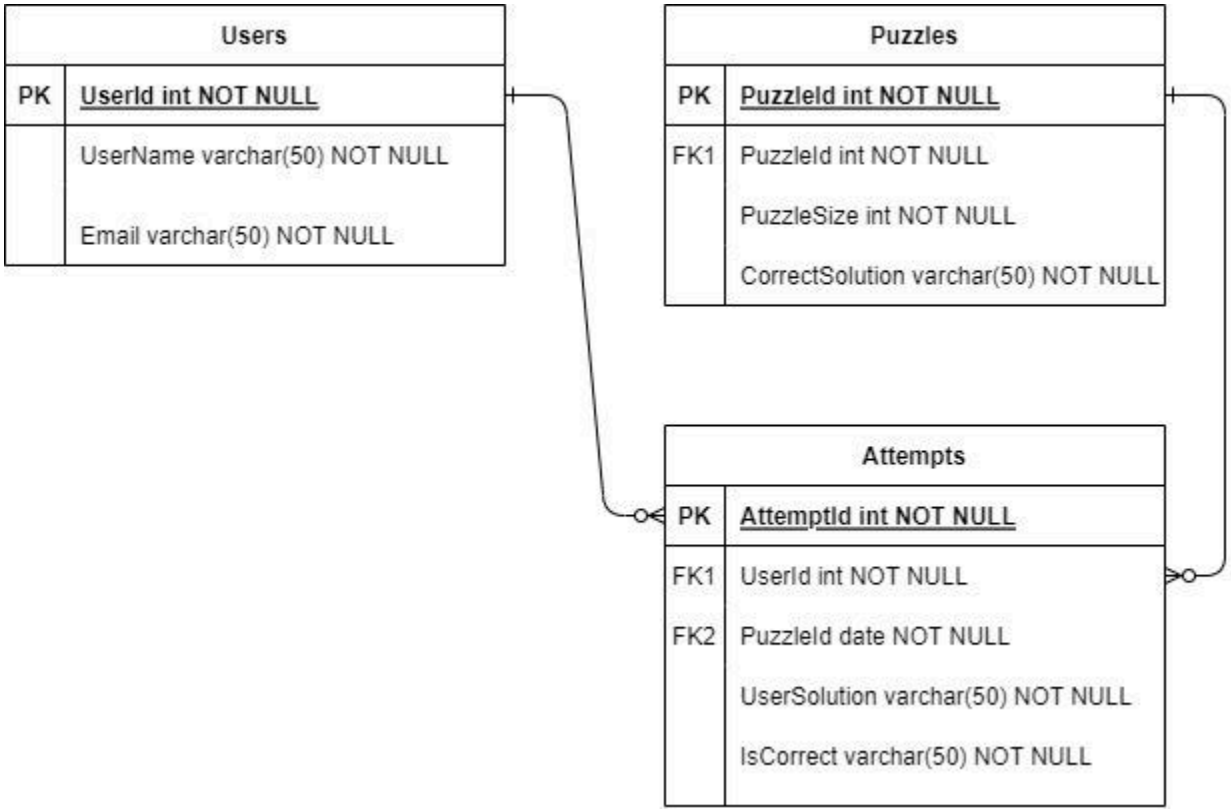
Project Description

The Magic Square Puzzle Solver is a web application designed to offer users an engaging platform to solve magic square puzzles. A magic square is a grid of numbers where the sum of every row, column, and diagonal is equal. This application provides an interactive environment where users can test their problem-solving skills by attempting to complete magic square puzzles of varying sizes and difficulties.

Flowchart:



ER Diagram:



Methods Used in the Puzzle Solver

- **Magic Square Generation:** Utilizes mathematical techniques and algorithms, including the Siamese method for odd-sized squares and a specialized method for even-sized squares, to generate solvable puzzles.
- **Solution Validation:** A real-time feedback mechanism that checks the sums of rows, columns, and diagonals against the magic constant to validate user solutions.
- **Dynamic Puzzle Generation:** Adjusts the difficulty by varying the number of pre-filled cells based on the user's experience or choice, using randomization techniques to ensure each puzzle is unique.

Market Space and Selling Points

Market Space:

- Targets the educational games market, appealing to puzzle enthusiasts, students, and educators looking for an interactive way to explore mathematical concepts.

Selling Points:

- Educational and Interactive: Enhances mathematical and logical skills.

- User-Friendly: Accessible to users of all tech levels.
- Adaptive Difficulty: Suits beginners to advanced solvers.
- Instant Feedback: Encourages learning through immediate solution validation.
- Community Engagement: Supports challenging others and sharing puzzles.

Functional Specifications

Product Features:

- Puzzle Generation: Users can generate puzzles of varying sizes and difficulties.
- Interactive Solving Interface: A drag-and-drop or click-based interface for inputting numbers.
- Solution Checker: Instant feedback on the correctness of the user's solution.
- User Accounts: Optional registration for tracking progress and achievements.
- Leaderboards and Challenges: For community engagement and competition.

Multiplayer Features (Bonus):

- Live Puzzle Races: Users can compete in real-time to solve puzzles faster than opponents.
- Puzzle Sharing: Ability to create and share puzzles with friends or the public.
- Collaborative Solving: Multiple users can work together on the same puzzle.

Deployment

Prepare the Application:

- Ensure all dependencies are listed in requirements.txt.
- Include a Procfile with the command to run the Flask app (e.g., web: gunicorn app:app).

Use Heroku for Deployment:

- Create a Heroku account and install the Heroku CLI.
- Log in to Heroku through the CLI and create a new app.
- Connect GitHub repository to Heroku or use the Heroku Git repository for deployment.
- Deploy the application by pushing the code to Heroku (either through GitHub integration or using git push heroku master).
- Configure any environment variables or add-ons like databases through the Heroku dashboard.

Verify Deployment:

After deployment, access the application URL provided by Heroku to ensure it's running correctly.

Milestone Features and Timelines

M1 (Week 1): Setup repository, initialize Flask application, begin backend development for magic square generation.
M2 (Week 2): Implement dynamic puzzle generation and API endpoints.
M3 (Week 3): Develop the React frontend for puzzle selection and solving.
M4 (Week 4): Integrate frontend and backend, begin testing and debugging.
M5 (Week 5): Finalize UI enhancements, complete documentation, and deploy the application.