🤝 Handover Document

1. Key Algorithms

The code involves a range of algorithms that control player strategies and game progression in a Tic-Tac-Toe game. Below are some key algorithms:

- **Random Player Algorithm**: Generates a random move from the list of available moves.

  ```
  1 python
  ```

  Copy code

  `def random_player(board, player, callback=None): moves = get_available_moves(board) return random.choice(moves), None, None, 0`

- **Minimax Algorithm**: Implements the minimax strategy for a perfect AI player. The algorithm is imported from the `tictactoe_engine` module and is invoked in the `minimax_player` function.

  ```
  1 python
  ```

  Copy code

  `def minimax_player(board, player, callback=None): _, best_move = minimax(board, player, 0) return best_move, None, None, 0`

- **LLM-Based Players**: Uses various language models (e.g., GPT-4, Gemini) to make moves. These functions call APIs asynchronously for each LLM to analyze the board and make a move.

  For example:

  ```
  1 python
  ```

  Copy code

  `async def minimal_gpt4_player_async(board, player): response = await request_minimal_gpt4_move_async(board, player) total_cost = sum(call.cost for call in response['api_calls']) return algebraic_to_index(response['selected_move']), None, response['prompt'], total_cost`

2. Importing Data from 3rd-Party Systems

The code integrates with third-party LLM APIs (GPT-4, Gemini, etc.) to make decisions based on the game state. Functions such as `request_minimal_gpt4_move_async` and `request_minimal_gemini_move_async` call these models and gather responses asynchronously. Costs incurred during API calls are also tracked in `total_cost`.

3. Key Classes and Application Layers

The game code is organized into modular functions with the following layers:

- **Player Strategy Functions**: Functions like `random_player`, `minimax_player`, and various async LLM-based player functions.
- **Game Invocation Layer**: The function `invoke_player_async` maps player names to their corresponding strategy function.
- **Game Loop Layer**: The `play_game_async` function acts as the main loop, tracking game turns, calling players, and logging game events.

4. Database Structure

This code doesn't involve a database, but if logging data were stored, a suggested schema could include tables like `Games`, `Turns`, and `Players` to store information about game sessions, moves, costs, and outcomes.

5. Deployment Guidelines

To deploy the C-LARA TicTacToe application locally, follow the steps outlined below. These instructions assume you have Python and Git installed on your machine.

1. Clone the Repository

git clone https://github.com/team061/C-LARA-TicTacToe.git Connect your Github account

2. Install Required Packages

pip install -r requirements.txt

3. Configure the Project Path

In the run.py file, ensure that the working directory is set correctly. You may need to adjust the path based on your local setup:

os.chdir('/Users/revelino/Documents/GitHub/IT-Project/C-LARA/')

4. Run the server

python manage.py runserver and run http://127.0.0.1:8000/ in your local browser

6. Licensing Agreements

This project is licensed under the MIT License.