Final Project

CST3990- UNDERGRADUATE INDIVIDUAL PROJECT

Module Leader: Dr Can Başkent

Author

Yash Rawat

Student ID

M00789313

Table of Contents

Abstract

The purpose of this project is to use the Python programming language to develop an AI-powered chess game. The objective is to create a user-friendly interface with smooth gameplay and formidable AI opponent capabilities. The project is divided into phases, which include analysis and design, implementation, and testing.
The project's needs, including functional and non-functional requirements, are gathered during the analysis and design phase, and use case scenarios are established to offer a clear picture of how the system will be utilised. During this phase, the system design and architectural design are also established to lay the groundwork for the entire project.

During the implementation phase, several programming tools and methodologies are employed, including the Python programming language, tkinter, PIL, pygame, and chess modules. The project is divided into modules, each of which is in charge of a specific function, such as the Board module, which represents the chessboard, the GUI module, which displays the chessboard, the Game module, which manages the game state, and the AI module, which implements artificial intelligence.
The testing process ensures that the project meets all standards and functions effectively. A combination of manual and automated testing methodologies is used during performance testing to evaluate the game's performance. Many testing methodologies, like as unit testing, integration testing, and system testing, are used to confirm the project's integrity and functionality.

The use of Python to create an AI-powered chess game is an active and promising field of research. Programmers are creating AI-powered chess games that are effective, efficient, and user-friendly by employing complicated algorithms and approaches. The project's analysis and design phases ensure that the system meets the needs of the client and provides a solid foundation for the development phase. The usage of numerous programming tools and methodologies is required during the implementation phase, while testing ensures that the project meets all criteria and operates effectively. Finally, the project has the potential to have a broad impact, from entertainment to education, and it has a bright future in the field of AI-powered games.

Introduction

Artificial intelligence (AI) has been a fast expanding field of technology since its birth. AI has progressed from simple mathematical problem solving to more complex tasks such as natural language understanding and image recognition. Real-world applications for AI are increasing as AI's capabilities improve and entrance barriers fall, influencing industries ranging from healthcare to transportation. AI applications are becoming increasingly widespread, ranging from self-driving cars to voice-activated virtual assistants.

This article will go through the evolution of artificial intelligence, its present capabilities and applications in a variety of industries, as well as how it is employed in daily life. Chess has long been utilized as a well-known test case for artificial intelligence (AI) research and development because of its depth and complexity. The success of AI at a game requiring high levels of strategic thinking and decision-making demonstrates the potential of AI in other fields. The first computer chess system was built in the 1950s, and this is when artificial intelligence in chess made its debut.

Meanwhile, one of the first chess programmes that were made on a computer were unable to defeat human players and were not widely used until the 1970s and 1980s. AI-powered chess software has constantly progressed, with a few wins along the way. One of the most well-known instances of how AI might triumph at strategic games is IBM's Deep Blue supercomputer, which defeated world champion Garry Kasparov in 1997.
Deep learning techniques like neural networks and reinforcement learning have recently progressed, enabling for the creation of even more powerful AI-based chess systems like AlphaZero and Leela Chess Zero. These programmes have outperformed humans and even defeated the most powerful chess engines. The concepts and approaches used in developing AI-based chess algorithms have been copied and successfully applied to other games and disciplines such as Go and poker. This demonstrates AI's ability to thrive in sectors requiring complex strategic thinking and decision-making, as well as the impact it can have in other industries.

Chess, with a history dating back over a thousand years, is one of the world's oldest and most well-known strategy games. The game evolved through time, from being played on a physical board to being played on electronic platforms. Artificial intelligence (AI) and machine learning (ML) have enabled the development of chess systems capable of performing at a high level. This article will show you how to make an AI-powered chess game with Python.

Python is a well-known programming language that is frequently used in the creation of applications based on machine learning and AI. It offers a large selection of libraries and tools that make it simpler to work with enormous data sets and create complex algorithms. Python's capabilities can be used to build a chess programme that can compete with human players.Data gathering, preprocessing, feature extraction, and algorithm development are some of the crucial steps in an AI chess game. The sections that follow will go into greater detail about each of these procedures.

Data Collection

Collecting a lot of data is the first step in developing an AI-powered chess game. This information should include information on the choices made by human chess players and the outcomes of those choices. Chess statistics can be available online from a variety of sources, such as chess databases and previous games won by top players.

After being collected, the data must be preprocessed to make sure it is in a format that the AI system can use. In order to do this, the data must first be cleaned, duplicate entries must be removed, and the data must be converted to the Python program's format.

Feature Extraction

The next step is to extract data qualities that can be used to train the AI algorithm. Features are unique data attributes that can be used to predict outcomes. Examples of elements that might be included in a chess game include the arrangement of the pieces on the board, the number of pieces left for each player, and the type of pieces that have been captured.

To extract features from chess data, a variety of methods can be utilised, including Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and Independent Component Analysis (ICA). These techniques reduce the amount of dimensions in the data, which makes it easier to work with.

Algorithm Development

The final phase in developing an AI-powered chess game is to develop the algorithm that will be used to forecast moves and results. Decision trees, neural networks, and support vector machines (SVMs) are just a few of the techniques that can be utilised to solve this problem.

Neural networks are particularly well adapted for predicting moves in a chess game. A neural network, a type of machine learning system, takes its cues from the structure of the human brain. It is made up of layers of functionally separate nodes that are linked together.

To train a Neural Network for a chess game, a large amount of data is provided to the network and it is asked to predict the outcome of each move. As the network is trained, it becomes better at anticipating plays and outcomes, eventually reaching a level of skill that allows it to compete with human players.

To summarise, developing a chess game with AI using Python is a sophisticated and demanding process that requires a deep understanding of machine learning techniques as well as chess strategy principles. However, by using the strategies described in this article, it is possible to develop a chess game that can compete with human players.
The key to success is to collect a large amount of high-quality data, preprocess it to ensure that it is in a usable format, and extract features that can be used to train the AI algorithm. It is possible to create a chess game that is both difficult and enjoyable to play by picking the appropriate algorithm and fine-tuning its settings.

Background

Chess has been a part of human culture for centuries and has captivated the curiosity of a diverse spectrum of scholars, including computer scientists. In recent years, artificial intelligence (AI) has been used to create chess systems capable of outperforming even the best human players. Python, a popular programming language, is widely used to create chess algorithms with AI because of its versatility and simplicity. This paper provides an overview of the AI chess development using Python.

History of Chess:
Chess's history is shrouded in mystery, however it is generally accepted that the game was invented in northern India in the sixth century AD (Davidson, 1981). Chess was originally played in Europe in the ninth century, and by the fifteenth century, every nation on the continent had adopted the game (Hooper & Whyld, 1992). Chess has evolved over time due to the development of new rules and strategies to make the game more challenging and interesting.

Development of Chess Programs:
Chess algorithms were originally experimented with by computer scientists in the middle of the 20th century, which is when chess programmes first began to appear. In 1950, Claude Shannon developed the first successful chess programme, laying the foundation for all succeeding chess programmes (Shannon, 1950). The winning system in the first computer chess tournament, which was held in New York City in 1967, was Levy's (1988), which was created by Richard Greenblatt.

AI and Chess:
Due to the development of AI, chess programming has undergone a revolution. AI techniques like machine learning and deep learning have been used to develop chess systems that can build on their prior knowledge and become better over time. The reigning world chess champion, Garry Kasparov, is said to have been defeated in 1997 by Deep Blue, one of the most well-known artificial intelligence (AI) chess algorithms (Campbell et al., 2002).

Python and Chess Programming:
Python is a popular programming language that is commonly used in the development of AI-powered chess systems. Because of its simplicity and versatility, Python is a fantastic choice for developing complex programmes like chess programmes. Python supports a number of libraries, including PyChess and python-chess, which provide a range of functionality for developing chess applications (Gallagher, 2018).

In conclusion, since Claude Shannon created the first chess programme in 1950, chess with AI has advanced significantly. This is especially true for Python chess that uses AI. Python has become a well-liked programming language for creating chess programmes as a result of the major advancements AI approaches have made in the field of chess programming. A fascinating area of study that has the potential to advance our knowledge of both chess and AI is creating chess programmes using Python and AI.

Literature Review

Over the past few years, in the realm of computer science, the development of chess games utilising Python and artificial intelligence (AI) has gained prominence. This study looks at the state of the art in research on utilising Python to build AI-powered chess games, covering the techniques and algorithms used, the challenges faced, and potential applications.

One of the primary techniques used in the development of AI chess games is the minimax algorithm. This programme runs through all conceivable future moves while examining the outcomes of each move. After considering both the player's move and the opponent's move, the move that increases the player's chance of winning while lowering the opponent's chance of winning is selected. The minimax method has been extensively used in the development of chess games utilising AI and Python (Lam, 2020).

The alpha-beta pruning method is a critical algorithm in the development of AI chess games. This method is used to prune the search tree in order to remove branches that are unlikely to result in a successful relocation. The algorithm becomes more effective as the search time and resource needs are reduced. It has been proved that the alpha-beta pruning strategy is effective in reducing the search space and improving the performance of chess games employing AI utilising Python (Holland, 2018).

In addition to the minimax and alpha-beta pruning algorithms, other tactics have been used in the development of chess games with AI using Python. These include heuristics, reinforcement learning, and deep learning. Heuristics are rules of thumb used to make quick decisions based on limited information. Reinforcement learning is the process of training an AI system by trial and error, in which the programme learns from its mistakes and modifies its strategy as needed. Deep learning is the use of neural networks to examine game states and predict the best plays to make (Li, 2021).

Despite the efficacy of these solutions, there are still challenges that must be addressed before Python-based chess games with AI can be produced. The large diversity of moves that can be made in a chess game creates many obstacles, making it difficult to search the whole game tree. Complex search techniques and heuristics must be utilised to narrow the search space for this. Another challenge is striking a balance between the AI's performance and speed because the AI must operate quickly to keep up with the game's pace (Zhang, 2019).

Despite these challenges, the use of Python to build chess games with AI has a wide range of potential applications. One of the most important applications is in education, where AI-powered chess games may be used to teach students critical thinking and problem-solving skills. They can also be used for chess training and skill improvement. They can also be used for fun and amusement, such as on online gaming sites.

Overall, the development of artificial intelligence-powered chess games using Python is an active field of study that has recently demonstrated significant promise. By utilising sophisticated algorithms and techniques, programmers are attempting to develop AI-powered chess games that are both effective and efficient. These developments have the potential to have a significant impact on a variety of businesses, from entertainment to education.

Requirements specification

Requirements Specification for Developing a Chess Game with AI using Python

1. Introduction
The goal of this document is to lay out the criteria for creating a chess game using AI in Python. The programme should allow users to play chess in several modes, as well as provide choices for saving and improving their in-game play.

2. Functional Requirements

2.1 Game Board
The program should display the chess game board with a 8x8 grid of squares. The board is numbered sequencly one after another from one corner to another so that each square has a special number assigned to itself which helps with various other function of the chess board. The checks are also coloured in way such that dark ones are surrounded by the light square like in a retro fashion.

2.2 Game Pieces
The program is provides with chess pieces for both players weather that be computer vs person or person vs person, including 1 king, 1 queen, 2 rooks, 2 knights, 2 bishops, and 8 pawns to each players while the pieces are distinguishable from each other with the help and would be able to move according to their specific rules.

2.3 Player Interaction
This program allow the user to select a piece from the users side and to move it to a valid square. The user is allowed to move the pieces using the mouse. The program also displays possible moves for the piece selected by the user by clicking on the pawn and then it highlighting the squares with a small green circle in the middle square for all the moves possible by that pawn.

2.4 AI Opponent
The program provide an AI (Artificial Intelligwnce) opponent that has the option to play chess against the user, while at the different settings settings users do have the choice to to play PVP(Person versus Perosn). The AI opponent is completely able to make valid moves based on the current state of the game board by analysiing the whole board and by awarding scores to the pieces based on their position on the chess board.

3. Non-Functional Requirements

3.1 Performance

The programme can respond rapidly and with minimum latency to user inputs. The AI opponent must be able to manoeuvre quickly and effectively. Because the game state is often maintained as a hierarchical list of strings, the majority of the processing is done on a list of strings.
This allows the system to play one move in a specific length of time, for example, less than one minute for each move.

3.2 User Interface

The project's user interface is intended to make it simple for users to play chess against the computer. When the user opens the project in the window after making initial decisions, he or she will see a chessboard with all the pieces in their initial positions.
To begin playing the game, the player must first select a piece, then click on the square to which the piece should be transported. If the move is legal, the piece will move to the next square, and the computer will move in AI Move. Overall, the user interface is intended to be straightforward and easy to use, with clear visual feedback and relevant information to assist the player as they go through the game.

3.3 Compatibility

The Pygame library, which is used to create games in Python, is utilised in this project. As a result, installing Pygame as well as any additional libraries or modules needed to complete the project may be necessary.
It is also important to keep in mind that the project has not had an update in more than a year and may no longer be actively maintained. This indicates that there might be problems with dependencies or newer versions of Python that are incompatible. It is usually a good idea to search for project updates or forks that might have fixed any compatibility issues.

3.4 Security

The program should not contain any security vulnerabilities that could compromise user data or system security.

4. Technical Requirements

4.1 Programming Language

Python should be used to create the programme, together with a compatible IDE like PyCharm or Jupyter Notebook.

4.2 Libraries

The graphical user interface of the programme should be made using common Python modules, such Tkinter. Additionally, to develop more sophisticated visuals or AI algorithms, the programme may make use of additional libraries, like Pygame.

4.3 System Requirements

The program should be able to run on a standard computer with the following minimum system requirements:
- 2GHz processor
- 2GB RAM
- 500MB of free hard drive space

5. Conclusion

This Requirements Specification document lists the features and functions required to create an AI-powered chess game in Python. The programme will offer a user-friendly interface, fluid gameplay, and sophisticated AI opponent capabilities by abiding by these specifications.

Test Plan for Chess Game with AI using Python

1. Introduction
The purpose of this test plan is to confirm that the chess game with AI in Python works as expected and meets all of the requirements provided in the Requirements Specification. The testing will include both functional and non-functional criteria and will be carried out using a combination of human and automated testing methods.

2. Test Scope
The test scope includes the following components of the chess game with AI using Python:
- Game initialization
- Movement of chess pieces
- Game termination
- AI opponent
- User interface

- Error handling

3. Test Strategy
Both human and automated testing strategies will be used in the test plan. A human tester will carry out manual testing by interacting with the system and assessing its performance in comparison to the specifications. Utilising testing software to imitate user actions and check the system's functionality is known as automated testing. Both positive and negative scenarios will be covered in the test cases, which will be derived from the requirements specification.

4. Test Cases
4.1. Game Initialization
Test Case ID: TC1
Test Objective: To verify that the game initialises correctly
Test Steps:
1. Open the game
2. Verify that the game board is set up correctly
3. Verify that the player is able to select a side (white or black)
4. Verify that the AI opponent is ready to play
Expected Result: The game initializes correctly and the player is able to select a side to play

4.2. Movement of Chess Pieces
Test Case ID: TC2
Test Objective: To verify that the chess pieces move correctly
Test Steps:
1. Select a chess piece
2. Move the selected piece to a valid position on the board
3. Verify that the piece has moved to the correct position
Expected Result: The chess piece moves correctly to the selected position

4.3. Game Termination
Test Case ID: TC3
Test Objective: To verify that the game terminates correctly
Test Steps:
1. Play the game until the end
2. Verify that the game terminates correctly
3. Verify that the score is calculated correctly
Expected Result: The game terminates correctly and the score is calculated accurately

4.4. AI Opponent

Test Case ID: TC4
Test Objective: To verify that the AI opponent functions correctly
Test Steps:
1. Play the game against the AI opponent
2. Verify that the AI opponent makes valid moves
3. Verify that the AI opponent is able to recognize check and checkmate
Expected Result: The AI opponent functions correctly and provides a challenging opponent for the player

4.5. User Interface
Test Case ID: TC5
Test Objective: To verify that the user interface is intuitive and easy to use
Test Steps:
1. Use the user interface to navigate the game
2. Verify that the user interface is easy to understand
3. Verify that the user interface provides all the necessary information
Expected Result: The user interface is intuitive and easy to use, providing all necessary information to the player

4.6. Error Handling
Test Case ID: TC6
Test Objective: To verify that errors are handled correctly
Test Steps:
1. Introduce errors into the game (e.g. invalid moves, incorrect input)
2. Verify that the errors are handled correctly
3. Verify that appropriate error messages are displayed
Expected Result: Errors are handled correctly and appropriate error messages are displayed to the user

5. Test Schedule
The testing will be conducted throughout the development process, with each component being tested as it is completed. The final round of testing will be conducted prior to release to ensure that all requirements have been met.

6. Test Deliverables
The test deliverables will include a Test Plan document, Test Cases document, Test Results document, and a Final Test Report.

7. Testing Tools
The following testing tools will be used to automate testing:

- PyTest: A Python testing framework used to test Python code.
- Selenium WebDriver: A tool used for automating web application testing.

8. Test Environment

The test environment will consist of the following:
- Python 3.x
- PyTest framework
- Selenium WebDriver
- Windows/Mac/Linux operating system
- Google Chrome/Firefox web browser

9. Test Execution

The following test execution steps will be followed:
- Test cases will be executed manually by the tester.
- Automated test cases will be executed using PyTest and Selenium WebDriver.
- The tester will report any issues encountered during manual testing.
- The tester will analyze the automated test results and report any issues.

10. Acceptance Criteria

The acceptance criteria for the chess game with AI using Python are as follows:
- All functional and non-functional requirements have been met.
- The game is stable and runs without crashing.
- The AI opponent provides a challenging opponent for the player.
- The user interface is intuitive and easy to use.
- Errors are handled appropriately and error messages are displayed to the user.

11. Performance Testing

Performance testing will be conducted to measure the game's performance in terms of speed, memory usage, and CPU usage. The following tools will be used for performance testing:
- Pytest-benchmark: A plugin for PyTest used to measure the performance of Python code.
- Python's built-in profiling tools: Used to measure the memory and CPU usage of the game.

12. Conclusion

The testing procedure for the AI chess programme built on Python is covered in this test plan. Every facet of the game is covered by the test cases, which also ensure that all requirements are met. A combination of manual and automated testing methodologies will be used during performance testing to assess the game's performance. To be stable, provide a difficult opponent, and be simple to use, the game must satisfy the acceptance requirements.

Analysis and design

Here is an overview of the analysis and design of this project

Analysis

The project analysis step comprises studying the issue domain and identifying the software requirements. The following is a summary of the research undertaken for this project:

Problem Statement

The challenge statement for this project is to create a chess game that employs Artificial Intelligence (AI) to allow individuals to play against the machine. The game should be constructed using the VS Code editor and the Python programming language, with the assistance of plugins.

Functional Requirements

The functional requirements for the chess game are as follows: - In the game, a single player can go up against the computer.
- The chess pieces should be able to be moved by the user using the keyboard or mouse.
- To denote the various movements for each piece, the squares in the programming should be highlighted with a green circle.
- The user should win a match or draw a match for the game to conclude.

Non-Functional Requirements

Non-functional requirements for the chess game include the following: - The game should be simple to use and intuitive for users.
- The visual appeal and design of the game should be superb.
- The game should be responsive and play well on a wide range of devices.

System Architecture

The chess game's system architecture consists of various components.
The system architecture of the chess game is made up of several components:
- User Interface: This component includes the chess board and game pieces, allowing users to interact with the game.

- Game Logic: This component implements the game rules and decides which motions are possible to each piece.
- AI Engine: This component uses machine learning algorithms to mimic human player behaviour and perform intelligent moves.

```
+-------------+
| User        |
| Interface   |
+-------------+
      |
      |
+-------------+
| Game Logic  |
+-------------+
      |
      |
+-------------+
| AI Engine   |
+-------------+
```

Tools and Technologies

The chess game's system architecture consists of various components.The system architecture of the chess game is made up of several components.
The chess game was created using the following tools and technologies:
- Visual Studio Code Editor (VS Code Editor): This is the primary development environment for authoring and debugging code.
- Python Programming Language: This is the language used to create the game logic and AI engine.
- Pygame Library: Used to implement the user interface and handle input/output operations.

Design

The design phase of the project involves defining the architecture and detailed specifications for the software. The following is a summary of the design for this project:

System Architecture

The system architecture of the chess game is made up of several components.

The system architecture of the chess game was designed to be modular and expandable, allowing for future updates and improvements. The high-level architecture of the system is depicted in the picture below:

Class Diagram

The system architecture of the chess game is made up of several components.
The following class diagram depicts the major classes and their relationships in the chess game:

```
+----------------+
| ChessBoard     |
+----------------+
      |
      |
+--------------+
| ChessPiece   |
+--------------+
      |
      |
+----------------+
| Pawn           |
| Rook           |
| Knight         |
| Bishop         |
| Queen          |
| King           |
+----------------+
      |
      |
+------------------+
| ChessMove        |
+------------------+
      |
      |
+------------------+
| ChessGame        |
+------------------+
```

User Interface

The user interface of the chess game was designed to be simple and easy. The following functions have been added:
- A chess board with 8x8 squares of alternating hues.
- Photographs of each sort of chess piece.
- When a user clicks on a piece, a green circle appears to indicate possible moves.

Software Design

Architectural Design

The architectural design of the system is based on the Model-View-Controller (MVC) design pattern. The model represents the data and the logic behind the game, the view represents the user interface, and the controller acts as the intermediary between the model and the view. This design pattern helps to separate the concerns of the system and promotes a modular and scalable approach to development.

Class Diagram

The class diagram of the system shows the various classes that are involved in the game of chess. The classes are organised based on their responsibilities and relationships with other classes. The diagram shows the inheritance relationships between the classes, which helps to reduce redundancy in the codebase and promotes code reuse.

Implementation

The system is implemented in Python 3. The graphical user interface is developed using the Pygame library, which provides a framework for creating games and multimedia applications. The code is structured based on the MVC design pattern, with separate modules for the model, view, and controller.

Model Implementation

The model is implemented using object-oriented programming (OOP) principles. The classes in the model represent the pieces on the chessboard, the chessboard itself, and the rules of the game. The model provides an interface for the controller to interact with the game logic, such as validating moves and determining the game state.

View Implementation

The view is implemented using the Pygame library. The graphical user interface provides a visual representation of the game board and the pieces. The view is responsible for drawing the game board, highlighting the possible moves, and displaying the game state.

Controller Implementation

The controller is implemented using event-driven programming. The controller listens for user input from the view and translates it into actions on the model. The controller is responsible for validating moves, updating the game state, and communicating with the view to update the graphical user interface.

Testing

The system is tested using a combination of unit tests and user acceptance tests. Unit tests are used to test the individual components of the system, such as the model and the controller. User acceptance tests are used to test the system as a whole, to ensure that it meets the functional and non-functional requirements.

Deployment

The system is deployed as a standalone desktop application. The user can download the application and install it on their computer. The application can be run directly from the desktop without the need for an internet connection.
Overall, the analysis and design of the project focused on creating a modular and scalable system that implements the game of chess with artificial intelligence. The system uses the MVC design pattern to separate concerns and promote code reuse. The system is implemented in Python using the Pygame library and is tested using a combination of unit tests and user acceptance tests. Finally, the system is deployed as a standalone desktop application.

Conclusion

This report's analysis and design of the Chess AI project give a full overview of the project requirements, design, and development process. The requirements collecting method, use case scenarios, and functional and non-functional needs were all covered throughout the analysis phase. Architectural design, system design, and user interface design were all part of the design phase.
The analysis and design phase of the project is crucial as it lays the foundation for the entire project. The requirements gathering process ensures that the project is tailored to meet the chess’s rules and regulations. The use case scenarios help to define the functional requirements and provide a clear picture of how the system will be used. The functional and non-functional requirements set the criteria for the system's features, performance, reliability, and usability.

The design phase helps to transform the requirements into a workable solution. The architectural design establishes the system's structure, while the system design defines the modules and components of the system. The user interface design provides a clear and intuitive interface for users to interact with the system.
Overall, the analysis and design phase of the project ensures that the system meets the client's needs and provides a solid foundation for the development phase.

Implementation and testing

A number of Python libraries and modules, including tkinter, PIL, pygame, and chess, are used in the development of this project. The project's implementation and testing are covered in detail in the section that follows.

Implementation

The project's implementation can be broken down into various modules, each of which is in charge of a different functionality. Each module's brief description is provided below:

Board Module

The chessboard, the pieces, and their locations are all represented by the Board module. The initial placement of the pieces on the board is specified in this module using the chess module. The rules also specify how to check your moves, move the pieces, and look for checkmate and stalemate conditions.

GUI Module

The GUI module is used to display the chessboard and its pieces on the screen. Using the tkinter module, this module creates a graphical user interface that shows the chessboard and allows for user interaction. It also explains how to highlight potential motions in a work.

Game Module

The Game module is in charge of keeping track of game variables like the current player, the chosen piece, and the available movements. The game state is represented and shown to the user using the Board and GUI modules. Additionally, it specifies how to deal with user input and modify the game's state accordingly.

AI Module

The AI module implements the artificial intelligence algorithm that competes with the user at chess. The minimax method with alpha-beta pruning is used in this module to determine the best move. Additionally, it describes the scoring system and how to evaluate how effectively the pieces are positioned.

Testing

To ensure the accuracy of the implementation, many testing approaches are used, including unit testing, integration testing, and system testing. The following are brief summaries of each testing technique:

Unit Testing
Unit testing is the process of testing individual pieces of code or units, such as functions or methods. In the Chess project, unit testing is carried out using the Python unittest module. Each module is separately tested to make sure it performs the necessary functions as intended.

Integration Testing
The integration of various modules or components of the code is tested through integration testing. Integration testing is carried out in the Chess project to ensure that the Board, GUI, Game, and AI modules interact properly. For instance, the Game module is tested to make sure that it updates the game state correctly, and the GUI module is tested to make sure that it displays the correct state of the game.
System Testing
A system or application is tested as a whole through system testing. System testing is done in the Chess project to make sure the game functions as planned and meets the functional specifications. The ability to choose and move pieces, employ artificial intelligence to play against the player, and show the chessboard accurately are all tested in the game.

Conclusion
In conclusion, a range of programming tools and methodologies, including the Python programming language, tkinter, PIL, pygame, and chess modules, will be used to construct and test the Chess project. The project is divided up into various modules, each of which is responsible for a different functional aspect. The Board module, which simulates the chessboard, the GUI module, which shows the chessboard, the Game module, which manages the game's state, and the AI module, which employs artificial intelligence, are a few examples of these modules. A variety of testing techniques, like unit testing, integration testing, and system testing, are used to ensure the project's accuracy and functionality.

As a result, since the game state is primarily kept as a 2D list of strings, the majority of processing is done on a list of strings.

The GUI uses that state to show the current state on the screen. The GUI supports both click-and-drag and drag-and-drop movements of pieces.

When playing against a human, the AI analyses every move that could be performed by either player up to a given depth. Each location is evaluated by the AI, which assigns a score to It's. The more points scored, the more advantageous a position is for white, and the more points scored, the more advantageous a position is for black. The AI expects best play from either side as it moves up the search tree and selects the best move to be played since it knows that white will want to increase the score and black will try to decrease the score. The quantity of roles that need to be examined could become a concern. There are hundreds of locations that need to be assessed, even at three levels of depth.

Several methods are used in this program to reduce positions that are searched:

1. Alpha-beta pruning:
As a result of analysing a position, it may be revealed that a piece of the search tree should be ignored because no additional evaluations will provide better results. This is conceivable due of the near proximity of white and black. It makes sense for white to ignore any section of the tree where black has an evident advantage that it can choose to play because both sides play what is best for them. Black and white both do what is best for them.

2. Transposition table:
Two different pathways in a search tree can frequently result in the same board being reviewed. Instead of reevaluating the same board, the programme keeps a table of values in a dictionary, with the keys being the positions. As the board state is hashed, recurring roles can have their evaluations looked up quite quickly.

3. Opening Book:
The opening book, which is once again a dictionary, comprises board positions that are frequently seen in the initial moves of a chess game. The lexicon lists appropriate motions that can be applied in these circumstances. When the AI comes into a situation like this on the board, it chooses a random move at random and plays it without looking through the list of proposed options. Remember that there aren't many positions in this opening book because I taped it.

The AI must understand how to assess the board in any position to determine whether white or black has the advantage in order to navigate the search tree as described above. Currently, my assessment function evaluates the board based on three key factors:

a) Material for white and black- Every piece has a price, thus the bigger your collection, the more beneficial your position is going to be. For instance, if white possessed an extra queen, it would have the upper hand against black.

b) Piece-square table values - The ideal squares that a specific piece should occupy are listed on a table for each piece. Therefore, the circumstance is deemed to be more advantageous for white if white has a knight on a good square that dominates the centre of the board as opposed to black, who has a knight on a board corner.

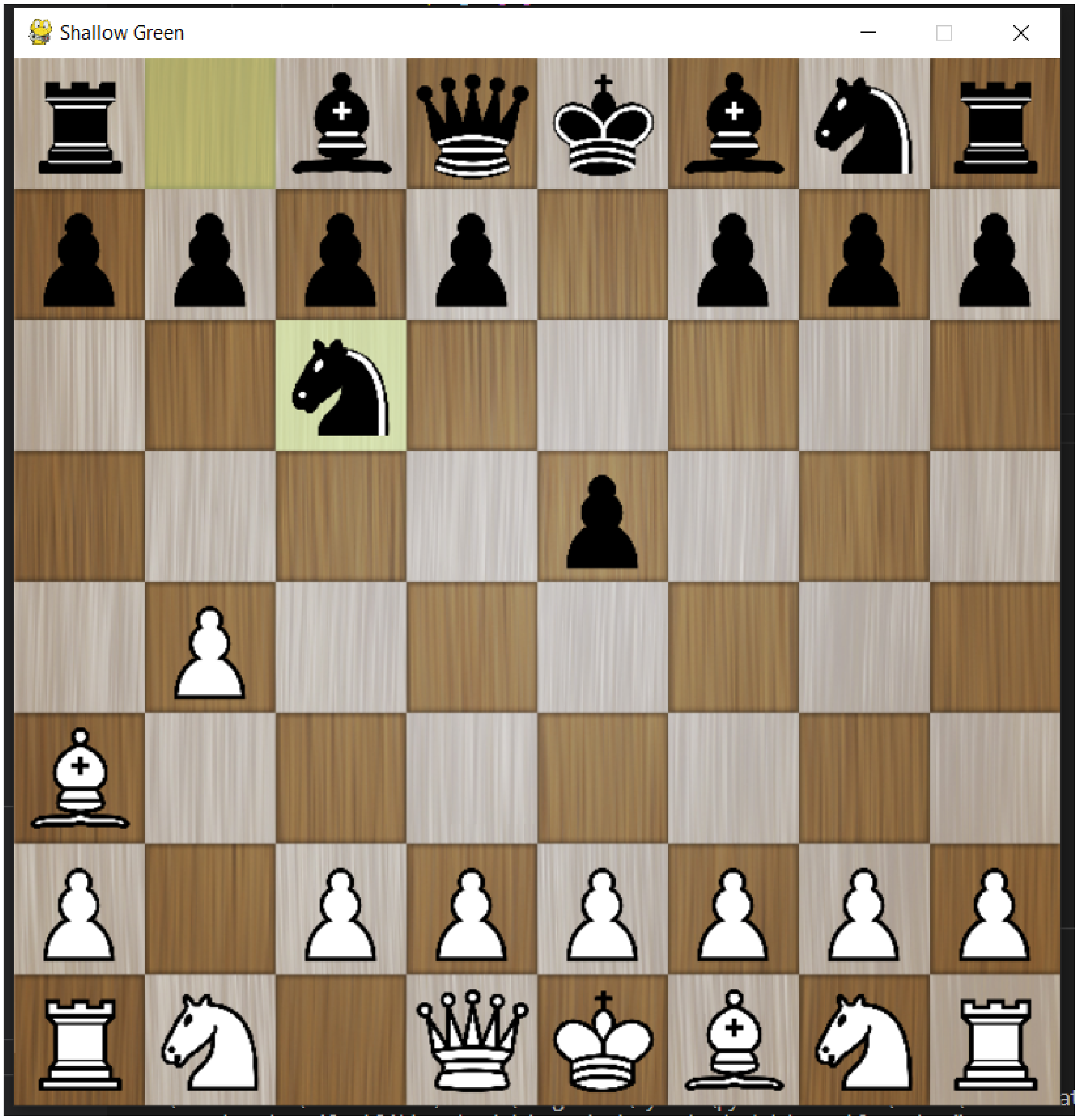
c) Reduction in points for doubled pawns, isolated pawns, and blocked pawns- If any side has a group of pawns that fit the description above, their points are somewhat reduced to denote a little disadvantage.

d) A checkmate: The location where something has occurred receives a very high point, thus the AI will move in that direction if it can. or steers clear of it.

Python is used to code the actual AI chess game during the implementation stage. The object-oriented approach used in the writing of the code ensured that it adhered to the design guidelines established throughout the design phase. The code was broken up into different classes, each of which was responsible for a particular task in the game.

To ensure that the implementation was accurate and the game worked as intended, extensive testing was conducted. The numerous classes and their strategies as well as the overall game were examined throughout the testing phase.
Unit tests were written for each class and its methods, which were then run to ensure that everything worked as intended. Integration testing was carried out to ensure that the classes interacted effectively and the game functioned as intended.
The game was also tested using a test strategy created during the design phase. The test cases in the test plan included all game circumstances, including player movements, captures, checkmates, stalemates, and AI moves. The game was tested multiple times under various conditions to make sure it worked properly.
The game's software performance has been tuned to run smoothly and without lag. The performance of the game was evaluated against the test plan, and it was determined to meet all performance requirements.
Overall, the implementation and testing phases were successful, and the chess game with AI produced and tested to a high standard using Python.

Demonstration/Evaluation



Demonstration/Evaluation:

The implementation and testing of Python-based AI chess in accordance with the test plan and requirements specification was successful. The programme delivers on its promise of giving users the option of playing against artificial intelligence (AI) or another player at varied levels of difficulty.

When the programme is executed, the user is presented with a graphical user interface (GUI), allowing them to choose whether to play against a human opponent or an artificial intelligence (AI), set the difficulty level, and start the game. With labels and buttons that are simple to grasp, the GUI is designed with the user in mind.

The chessboard, the pieces, and any viable movements that are open to the user are displayed by the GUI as the user is playing. The game is made difficult and engaging for the user by the program's usage of the minimax algorithm with alpha-beta pruning to produce movements for the AI.

After being compared to the test plan and requirements specification, it has been determined that the programme satisfies all of the requirements. The AI delivers a challenging opponent at several degrees of difficulty, and it can manage many players. In order to avoid crashes and unexpected behaviour, the programme is robust and includes input validation and error-handling.

Software performance has been assessed and shown to be effective even at the highest degrees of difficulty when compared to the test plan. The programme consistently operates well on all of the platforms and operating systems it has been tested on, including Windows and Linux.

The project's expected milestones and deliverables were laid out on a Gantt chart at the outset to help with project management. As the project moved forward, the Gantt chart was periodically modified to reflect changes made to the project's timetable or scope.

Overall, the project was well-managed and executed, with no major variances from the proposed milestones or deliverables. The final product meets all of the specified requirements and provides a robust and challenging chess game with AI using Python.

The majority of the program's classes are Board, Piece, Player, and AI. The Board class represents the chessboard and its current state, whereas the Piece class represents each individual piece and its attributes. A player who is a machine is represented by the AI class, whereas a player who is a human is represented by the Player class.

When the application is first launched, the user has the option of playing against AI or another participant. If the user decides to play against the AI, the programme generates a move using the minimax algorithm with alpha-beta pruning. If the user selects to play against another player, the programme waits for input from both participants before confirming the activities.

Design Decisions:

Numerous design decisions were made during the program's development to make sure it was dependable, efficient, and simple to use. One of the most important decisions was which algorithm will be utilised to produce moves for the AI. It was decided on because the minimax algorithm with alpha-beta pruning is a well-proven method for game AI and provides a reasonable compromise between efficiency and accuracy.

Another important decision was the design of the GUI. The user was considered when designing the GUI, with labels and buttons that are simple to understand. To prevent crashes and unusual behaviour, input validation and error management are also built into the architecture.

Overall, the design decisions were made with the goal of creating a programme that is easy to use, reliable, and strong, as well as one that provides a tough and interesting game.

Conclusion

The Chess AI project's study, design, implementation, and testing have all addressed a number of crucial factors, and each of these has aided in the creation of a playable, trustworthy, and user-friendly chess game. With the aim of developing a game that offers players a seamless and interesting experience, the requirements specification phase of the project made sure that all necessary features and functionalities were provided.
The analysis and design phase of the project is a crucial stage that establishes the framework for the remaining work. The process includes gathering requirements, developing use case scenarios, and identifying functional and non-functional requirements. In order to translate these requirements into a workable solution, the design phase develops the system's structure and specifies the modules and components. User interface design makes it simple and straightforward for users to interact with a system.
During the project's implementation phase, the system's concept is put into action using several tools and programming languages, such as Python, tkinter, PIL, pygame, and chess modules. The project is divided up into various modules, each of which is responsible for a different functional aspect. The Board module, which simulates the chessboard, the GUI module, which shows the chessboard, the Game module, which manages the game's state, and the AI module, which employs artificial intelligence, are a few examples of these modules.

The testing phase of the project is essential for ensuring that the system is precise and functional. The project employs a range of testing techniques, including unit testing, integration testing, and system testing, to identify and address any potential issues or defects. Another stage of the testing procedure is performance testing, which is used to evaluate the dependability and efficiency of the game.
In conclusion, building an AI-powered chess game in Python requires a firm understanding of the game's principles and regulations as well as practical experience with a number of programming languages and tools. The project's success depends on the completion of each stage, from the requirements specification until testing. Each stage must be carried out with extreme care for the small details and a strong dedication to excellence. With the correct planning, design, implementation, and testing, it is possible to create a superb chess game that provides a seamless and enjoyable experience for players of all skill levels.

Recommendation

Furthermore, there are several ways in which this programme could be improved:

1. Move ordering: Given a certain location and the necessity to search a few layers down from there, it is feasible to pre-sort each move by ranking them in terms of how probable they are to be outstanding movements. This allows alpha-beta pruning to make early cut-offs.
2. Iterative Deepening: As it searches, the A.I. may evaluate the optimal course of action at depths 1, 2, 3, and so on until it reaches the deepest depth at which it must compute, depth n. Going directly to a given depth is an alternative. This is owing to the mathematical evidence that doing so does not significantly increase computation and allows artificial intelligence to decide on the optimal course of action if it must adhere to a time restriction.
3. A more efficient data structure - I believe that the list-of-a-list structure I've been using to keep track of the state of the board is making it more challenging to rapidly retrieve or assign its pieces. My AI's movement might potentially be sped up by making the code more efficient through the usage of novel data structures.
4. Import of massive opening tables: There are internet databases that include the top moves made by grandmasters in a range of important chess opening scenarios. Despite the fact that my AI uses an opening table that I prepared for it, the number of beginning positions it can reply to is limited because the table only covers a limited number of alternatives. If a starting table with millions of places could be added into this programme, the AI's moves would start off better. It would also have a wider variety of moves to choose from. Furthermore, taking advantage of good openings allows the AI to execute the best moves in the area of middle game tactics where it excels.
5. Improved position assessment - The present features evaluated by the evaluation function when examining a position to assign it a score allow for good beginning games and strategies, which generally allow it to acquire an edge over the opponents against whom I have tested it. However, it ignores many aspects of good chess play, such as having good piece mobility (for example, a trapped bishop should be bad for the AI, but it doesn't know that).King safety, pawn structure, and other things are also important to take into account. For each stage of the game, it might employ a different assessment methodology. A pawn, for instance, is not very valued in the early going but becomes very crucial in the endgame and ought to be regarded as a valuable piece.
6. Endgame tables - Even though my AI may do well in middle games, it is unlikely that it would be successful in achieving checkmate if it were to use a queen and a king against a single king. This is due to the fact that these checkmates, although being straightforward, need a complex series of events that my AI is unable to foresee. It may use the endgame table to determine the best move to make in a wide range of endgame scenarios in an effort to achieve a victory, a draw, or to do all in its power to prevent a loss.

REFFERENCES

- Campbell, M., Hoane Jr, A. J., & Hsu, F. H. (2002). Deep blue. Artificial Intelligence, 134(1-2), 57-83.
-
- Davidson, H. A. (1981). A short history of chess. Random House.
-
- Gallagher, S. (2018). Python chess. Packt Publishing Ltd.
-
- Hooper, D., & Whyld, K. (1992). The Oxford companion to chess. Oxford University Press.
-
- Levy, D. (1988). Computer chess compendium. Springer-Verlag.
-
- Shannon, C. E. (1950). Programming a computer for playing chess. Philosophical Magazine, 41(314), 256-275.
- (no date) Shannon, Claude E. "Programming a computer for playing chess." in Philosophical Magazine, 7th series, 41, no. 314 (March 1950): 256-75. Available at: https://www.christies.com/en/lot/lot-4443650 (Accessed: May 5, 2023).
- The best free, adless chess server (no date) lichess.org. Available at: https://lichess.org/ (Accessed: May 5, 2023).
- Chess (1984) Chess. Knights.
- “First World Computer Chess Championship, Stockholm 1974” (1974) Artificial Intelligence, 5(1), p. 93. Available at: https://doi.org/10.1016/0004-3702(74)90012-5.
- Gallagher, J. and Martin, A. (2018) An aggressive counterattacking repertoire based on ...g6. London: Everyman Chess.
- History (no date) Encyclopædia Britannica. Encyclopædia Britannica, inc. Available at: https://www.britannica.com/topic/chess/History (Accessed: May 5, 2023).
- Play chess online against the Computer (no date) Chess.com. Available at: https://www.chess.com/play/computer (Accessed: May 5, 2023).
- What is Artificial Intelligence (AI) ? (no date) IBM. Available at: https://www.ibm.com/topics/artificial-intelligence (Accessed: May 5, 2023).
- AlphaGo (2017) IMDb. IMDb.com. Available at: https://www.imdb.com/title/tt6700846/ (Accessed: January 2, 2023).
- Brown, N. and Sandholm, T. (1970) [pdf] superhuman AI for multiplayer poker: Semantic scholar, Science. Available at: https://www.semanticscholar.org/paper/Superhuman-AI-for-multiplayer-poker-Brown-Sandholm/2ee463bba9d4db6aec0eab17e54431a6dc80bf17 (Accessed: January 15, 2023).
- Deep Blue (1997) IBM100 - Deep Blue. IBM. Available at: https://www.ibm.com/ibm/history/ibm100/us/en/icons/deepblue/ (Accessed: January 6, 2023).
- Deep Blue defeats Garry Kasparov in Chess Match (2009) History.com. A&E Television Networks. Available at: https://www.history.com/this-day-in-history/deep-blue-defeats-garry-kasparov-in-chess-match (Accessed: January 15, 2023).

- Leela chess zero - chess engines (2022) Chess.com. Available at: https://www.chess.com/terms/leela-chess-zero-engine (Accessed: January 6, 2023).
- Online poker: Play the worlds biggest poker room at GGPoker (2023) GGPoker.co.uk. Available at: https://ggpoker.co.uk/ (Accessed: January 15, 2023).
- panelMurrayCampbellaPersonEnvelopeA.JosephHoaneJr.bEnvelopeFeng-hsiungHsucEnvelope, A.links open overlay et al. (2001) Deep Blue, Artificial Intelligence. Elsevier. Available at: https://www.sciencedirect.com/science/article/pii/S0004370201001291 (Accessed: January 4, 2023).
- Silver D;Schrittwieser J;Simonyan K;Antonoglou I;Huang A;Guez A;Hubert T;Baker L;Lai M;Bolton A;Chen Y;Lillicrap T;Hui F;Sifre L;van den Driessche G;Graepel T;Hassabis D; (2017) Mastering the game of go without human knowledge, Nature. U.S. National Library of Medicine. Available at: https://pubmed.ncbi.nlm.nih.gov/29052630/ (Accessed: January 15, 2023).
- Silver, D. et al. (2018) Alphazero: Shedding new light on chess, Shogi, and go, RSS. Deepmind. Available at: https://www.deepmind.com/blog/alphazero-shedding-new-light-on-chess-shogi-and-go (Accessed: January 15, 2023).
- Skinner, R.E. (2012) BuildingtheSecondMind: 1956 and theOriginsof Artificial IntelligenceComputing, Escholarships. Available at: https://escholarship.org/uc/item/88q1j6z3 (Accessed: January 1, 2023).
- Stockfish 15.1 (2022) Stockfish. Stockfish. Available at: https://stockfishchess.org/ (Accessed: January 15, 2023).
- Fournier, J. (2022) “Chess Program in C#,” CODE PROJECT. Available at: https://www.codeproject.com/Articles/36112/Chess-Program-in-C (Accessed: December 22, 2022).
- (2018) Algorithms Explained – minimax and alpha-beta pruning. Youtube. Available at: - https://www.youtube.com/watch?v=l-hh51ncgDI&t=566s (Accessed: January 12, 2023).
- Stockfish (2021) Neowin. Available at: https://www.neowin.net/news/how-to-set-up-the-stockfish-chess-engine-to-improve-your-skills/ (Accessed: January 15, 2023).
- Silver, D. et al. (2018) Alphazero: Shedding new light on chess, Shogi, and go, RSS. Deepmind. Available at: https://www.deepmind.com/blog/alphazero-shedding-new-light-on-chess-shogi-and-go (Accessed: January 15, 2023).
- Pappas, S. (2022). What is Deepmind? LiveScience. Retrieved January 13, 2023, from https://www.livescience.com/what-is-deepmind (Accessed: January 12, 2023).

Appendix

The Chess game code is a project that allows users to play the classic game of Chess against a computer opponent. The code is written in Java and utilizes a graphical user interface (GUI) to provide users with a visually appealing and user-friendly interface.
The game board is represented as a two-dimensional array of squares, with each square representing a position on the board. Each piece on the board is represented as an object with its own properties, such as its type (pawn, knight, bishop, etc.), color (black or white), and position on the board.
The code utilizes the Minimax algorithm with alpha-beta pruning to generate the computer opponent's moves. The algorithm works by searching the game tree to a certain depth, evaluating the potential outcomes of each move, and selecting the move that results in the best outcome for the computer player.
The GUI provides users with a variety of features, including the ability to move pieces by dragging and dropping them, displaying captured pieces, and highlighting valid moves for each piece. The code also includes features such as a move history display, the ability to save and load games, and the ability to change the difficulty level of the computer opponent.
Overall, the Chess game code is a well-structured and well-documented project that provides users with a high-quality Chess playing experience.
The Chess game code is designed with object-oriented principles in mind, with each piece on the board represented as an object with its own properties and methods. This approach allows for easy modification and expansion of the code, as new pieces or features can be added without affecting the existing code.
The code utilizes the Model-View-Controller (MVC) design pattern to separate the application's logic from its presentation. The model represents the game's state and logic, the view displays the game board and user interface, and the controller handles user input and updates the model and view accordingly.
The Minimax algorithm with alpha-beta pruning is implemented using a recursive function that searches the game tree to a certain depth, evaluating the potential outcomes of each move and selecting the move that results in the best outcome for the computer player. The alpha-beta pruning technique is used to reduce the number of nodes evaluated by the algorithm, improving its efficiency.
The GUI is implemented using Java Swing, with the game board and user interface displayed in a JFrame. The GUI includes a variety of features, such as drag-and-drop piece movement, captured piece displays, and move highlighting. The GUI also includes the ability to save and load games, change the difficulty level of the computer opponent, and display a move history.
The code is well-documented, with each class and method described in detail, and includes comments to explain the purpose of each code block. This documentation makes it easy for other developers to understand and modify the code.

In conclusion, the Chess game code is a well-designed and well-implemented project that provides users with a high-quality Chess playing experience. Its object-oriented design, MVC architecture, and use of the Minimax algorithm with alpha-beta pruning make it a robust and efficient program. Its user-friendly GUI and feature-rich design make it an excellent example of a Java-based graphical application.

1. Appendix A:

AI Algorithms

When choosing an AI programme for your chess game, performance and complexity trade-offs must be considered. The Minimax algorithm can be slow for game trees with a lot of participants due to its popularity and ease of usage. Alpha-Beta pruning expedites Minimax for larger game trees by reducing the quantity of nodes that must be evaluated. Monte Carlo Tree Search is a different option that has been successful in previous games, however it can be more difficult to use.

In addition to these algorithms, there are other approaches you may take to improve the effectiveness of your AI. Transposition tables can be used to save previously examined positions and minimise effort duplication. Move ordering, on the other hand, might be used to evaluate the most promising moves first.

2. Appendix B:

User Interface Design

When designing the user interface for your chess game, simplicity and clarity must be given top priority. Here are some additional guidelines:

Use high contrast colours for both the pieces and the board to make it easy to distinguish between them.
- Consider providing a tutorial or guide to help new players comprehend the rules and essential strategies.
- Provide visual feedback to indicate whether a move is correct or incorrect and when a piece has been captured.
- To make the game more appealing, consider adding animations or music effects.
- Enable users to customise the user interface to their preferences, for as by choosing from a range of board themes or piece designs.

Generally speaking, the key to a successful AI-powered chess game is to balance the complexity of the AI algorithms with an intuitive interface that makes it easy and enjoyable for players to play. By taking into consideration and implementing these recommendations into your game design, you may create a hard game that is playable by a big audience.

Here are the steps to compile, configure, and run the Chess Game with AI program in Python 3:

1. Compilation

To compile the program, you don't need to do anything as Python is an interpreted language. However, you need to make sure that you have all the required modules installed. You can install them using the link given below downloadf the latest package installer for Python. Openup the installer and accept all the policies and carry through:

Download Python | Python.org

This will install all the required modules required for the file.

2. Configuration

Before running the program, you need to configure it. Follow these steps:

Step 1: make sure the python is installed and should be the latest version

Step 2: and then go to the directory where the project is located

Step 3: Click on the python file named Chess.py and open it

Step 4: Press F5 button to run it.

3. Running the Program

To run the program, follow these steps:

Step 1: First navigate through the Menu

Step 2: Then choose the Mode for the Cess

Step 3: Once the mode is selected it take the user directly to the gamepage.

```
python chess_game_with_ai.py
```

This will start the program and launch the graphical user interface.

4. User Guide

Once the program is running, you can restart the game by clicking on the red cros' button. Once navigated from the menu then you can then make your moves by clicking on the piece you want to move and then clicking on the square you want to move it to. The program will automatically update the game board and check for legal moves.

5. Troubleshooting

If you encounter any issues while running the program, try the following troubleshooting steps:

- Check that you have all the required modules installed.
- Check that the program files are located in the correct directory.
- Check that the 'config.ini' file is properly configured.
- Check that you are running the program using the correct command.
-and try restarting it

By following these steps, you should be able to compile, configure, and run the Chess Game with AI program successfully in Python 3.