Team Information

Name: Chess++
Group Members:

- Ahmad Bilal Carini
    - Position: Version Control Management
- Zeferino Araiza-Flores
    - Position: Timeline Management
- Zenas Ortega
    - Position: Chess Management

Communication Tools:

- Microsoft Teams
- School Email

Communication Rules:

- Respond 24 hours from message
- Ping everyone for important messages
- If no response then an email will be sent

Relevant Project Artifacts:
https://github.com/ZeferinoA/Chess_plusplus

Product Information

Abstract:

- This program can simulate a chess game for beginners and advanced players who want to take the game into their own hands. Beginners will be able to learn the basics of how each piece moves up to the complex openings and book moves. Advanced players will be able to improve the game that they have come to learn with pieces they want to add and establish their moves. These players can then connect to the server for sorta-ish online multiplayer.

Goal:

- The goal is to teach users how to play chess, create new pieces and new ways to play, and allow for local server 2 player connectivity or single player games to improve with oneself.

Current practice:

- Chess exists, but can a chess board teach new chess players how to play? How about with new pieces that can do dynamic and interesting moves? What about a

simple CLI chess interface that supports complex moves like Castling and En Passants? Plenty of CLI chess games exist, but do they have the wealth of features that truly makes them worth using over a physical chess board?

Novelty:

- The novelty of this program will be the CLI chess interface that supports complex moves, the addition and use of new pieces and new ways to play the game, and local connectivity.

Do not reinvent the wheel or reimplement something that already exists, unless your approach is different.

Effects:

- New players that want to learn the basics and the complexities of the game will have a place to learn. Players that want a unique twist to the game can create new pieces to use and go against. It will allow all players to have fun either by learning to play for the first time, good ol simple chess, or have crazy modes/new pieces implemented

Technical Approach:

- The program will run in a linux environment and may support other OS and platforms, it will be developed primarily in C++, leveraging the use of rendering libraries for the graphical elements. It will be developed and tested through a command line interface. Git and Github will be used for planning and version control. Pull requests will be used to manage merging.

Risks:

- Creating an intuitive interface for moving pieces could pose a problem. Chess notations exist but require outside knowledge beyond just chess knowledge. Overall, the interface poses a risk as it may end up being too esoteric to understand how the program works if the interface ends up being too complex or limiting.
- Maintaining the novelty of the program despite the ubiquity of chess could pose a challenge. Making novel versions and modes or pieces could add flavor to the project but making sure the game remains balanced with those new pieces or versions will pose a challenge.

- When working through Github, making sure merges don't break or overwrite features that others have created will pose a risk, even though Github allows easy access to rollbacks.'
- Programming lesser known features of chess, such as En Passants, may pose an issue and leaving these features out (because of knowledge limitations or negligence) would remove a core (if obscure) component of the game.

Use Cases (Functional Requirements): Each team member must come up with and describe **at least one** use case of your product. Refer to "1. Use Cases (Functional Requirements) " in Project Requirements Elicitation

1. Learn and Practice Chess, Beginner Level
    a. Actors: User that is a beginner to chess and wants to learn
    b. Triggers: A user selects the lesson option, and begins their level of beginner
    c. Preconditions: User has opened the game, loads on their system, and the standard rule set is embedded in the system
    d. Postconditions (success scenario): The user is able to start the basic lessons in which the program teaches about the pieces and basics
    e. List of steps (success scenario)
        i. User selects the lesson menu
        ii. User selects the basic lesson they want to study
        iii. The system bring up a text line that states what is needed to do with explanation
        iv. User selects the right move
        v. The lesson is complete and the user can now move on to the next lesson
    f. Extensions/variations of the success scenario: The variation/extension of this can be the one used for the advanced setting
    g. Exceptions: A user puts in an invalid input and is told to insert in a certain format. A user selects the wrong move and is told that it is not correct and is retold what is needed

2. Learn and Practice Chess, Advanced Level
    a. Actors: User that is decently experienced with chess and wants to learn more
    b. Triggers: A user selects the lesson option, and begins their level of advanced

c. Preconditions: User has opened the game, loads on their system, and the standard rule set is embedded in the system
d. Postconditions (success scenario): The user is able to start the advanced lessons in which the program teaches about book moves and openings
e. List of steps (success scenario):
    i. User selects the lesson menu
    ii. User selects the advance lesson they want to study
    iii. The system bring up a text line that states what is needed to do with explanation
    iv. User selects the right move
    v. The lesson is complete and the user can now move on to the next lesson
f. Extensions/variations of the success scenario: The variation/extension of this can be the one used for the basic setting
g. Exceptions: A user puts in an invalid input and is told to insert in a certain format. A user selects the wrong move and is told that it is not correct and is retold what is needed

3. Create and Integrate Custom Chess Pieces
    a. Actors: User that wishes to create a novel chess piece to make the game more novel and interesting
    b. Triggers: Loads a menu that offers options that change the move options of a piece and set a custom character/icon to that piece. Saves the piece for the session
    c. Preconditions: User has opened the game and it loads on their system
    d. Postconditions: The piece saves for the session and will load into the next game played. The piece will act with the previously specified behavior
    e. List of steps (success scenario):
        i. Player opens the game and navigates to the create menu
        ii. The user selects from a list of options available and creates a novel piece
        iii. The user selects a symbol for the piece
        iv. The piece saves for the session
        v. The user navigates to the mode selection and selects the option to enable custom pieces
        vi. The custom pieces are available in the mode selected

vii. The pieces function as they were originally described to the system given the available options

f. Extensions/variations of the success scenario
   i. Making sure the pieces do not load into normal modes that do not have custom pieces enabled.

g. Exceptions: failure conditions and scenarios
   i. The pieces do not save
   ii. The pieces do not load into the scenario
   iii. The pieces do not function as described

Non-functional Requirements: Describe at least three non-functional requirements, e.g., related to scalability, usability, security and privacy, etc.

- Scalability
  - The scalability of the application is needed due to increasing the user base for multiplayer and increasing the board for new pieces added
- Adaptability
  - The application needs to adapt to the user's pieces and rule changes in order to produce seamless gameplay for them
- Usability
  - Since the application is a game, the user experience needs to be friendly and the lessons given by the application needs to be understandable for the user.

External Requirements: Refer to "3. External Requirements" in Project Requirements Elicitation.

- As our application will take user input for moves and chess piece creations we do need to check the external requirements with being robust against errors
- Since the app is in c++, the application will be downloaded and be able to be executed by the user

Major Project Features:

- Major Feature 1: Establishing the logic of the command-line chess, beginning with the creation of the array for the board, game pieces structures, and the rules for the game
- Major Feature 2: Establishing the logic for the chess teacher for basic moves such as how the pieces move and how to check and checkmate
- Major Feature 3: Expanding upon the learning logic for complex move such as special openings and book moves such as castling

- Major Feature 4: Allowing users to create their own chess pieces with their own move set in order to expand the game

Stretch Features:

- Stretch goal 1: Making a local multiplayer environment so that the game is playable between players on the same network but different devices through the use of network sockets and hosting clients (as needed).
- Stretch goal 2: Making a CPU mode. Wherein the player plays against a ‘smart’ cpu capable of determining moves itself, either through the use of machine learning or pretrained open-source models.

Team process description

Describe your quarter-long development process.

- Specify and justify the software toolset you will use.
    - We chose to use C++ since it is the most high performing language and is a comfortable language with our team with all of us knowing how to do it and it is a real challenge with the less libraries
- Define and justify each team member’s role: why does your team need this role filled, and why is a specific team member suited for this role?
    - Our team needs the roles of Version Control Management, Timeline Management, and Chess Management due to the method of the project being tight and compact with the deadline and the necessity of Chess being implemented in our Chess app. The Version Control Manager is needed for the repository and making sure that no code is pushed without proper checks, Ahmad is a great fit for this role for their great availability in checking code, being a leader and mentioning when something needs to be revisited, and they’re avid about maintaining good working code. The Timeline Manager is needed for reaching the deadlines and week schedules that we have for the project, Zeferino is a great fit for this role for the schedule checking with history with managing the project tab of a previous project and the ability to work with a deadline. The basics of the app needed is the chess system so a Chess Manager is needed to make sure the project remains to the core principle, Zenas is needed for this role

for his knowledge in chess and the passion he had carried for pitching this project to the team.

- Provide a schedule for each member (or sub-group) with a measurable, concrete milestone for each week. "Feature 90% done" is not measurable, but "use case 1 works, without any error handling" is.
    - Week 4: The project has begun and the folder directory of github is specified and version control measures have been implemented for all future commits.
    - Week 5: We have a completed the set of elicited scenarios for the beginner level (use-case 1) and advanced level (use-case 2)
    - Week 6: Chess++ is a functional game in the command line, that is, you can play a game of chess against a friend.
    - Week 7: Use-case 1 scenarios are implemented in the game and Use-case 2 scenarios are implemented in the game.
    - Week 8: Use-case 3 is complete, without error handling
    - Week 9: Stretch goals have been evaluated for possibility and those that were deemed possible were completed.
    - Week 10: The project now has error-handling and UI/UX has been evaluated to ensure the game is in a playable state.
- Specify and explain at least three major risks that could prevent you from completing your project.
    - The implementation of Use Case 3 proves too challenging to implement well. That is: the creation of novel pieces causes too many challenges for bug-fixing and getting the pieces into the game in a logical way
    - The implementation of guides and learning systems requires technical knowledge of chess that no one in the group has. The general wide range of skill gaps that chess as a game fosters could lead advanced players to not feel like their skills are truly explained well for advanced scenarios.
    - None of the stretch goals specified are feasible within the 10 week time span of the course and we end up only being proficient enough to implement the basic requirements.

- Describe at what point in your process external feedback (i.e., feedback from outside your project team, including the project manager) will be most useful and how you will get that feedback.
    - The point that external feedback will be processed will be at the start, after the presentation and comments, since this can challenge us to add things that are wanted by the feedback or things to make different that we initially pitched. The next time to gather feedback would be after the creation of our lessons, we want to know what lessons need or are wanted to be included in the application since time is limited for the project and we want the most impact for the userbase. The final time to gather feedback would be after implementation of the creation feature as we want to know about how we can make the creation better for the users or what to add on such as which stretch goal is wanted to be implemented first or if anything past that is wanted.

Timeline:

- Week 1: Begin creating array chess board with pieces
- Week 2: Begin the logic of the chess board for check, checkmate, and others
- Week 3: Create the teaching module for basic moves such as how the pieces move
- Week 4: Expand the teaching module for basic moves such as checking and checkmating
- Week 5: Expand the teaching module for complex moves with openings
- Week 6: Expand the teaching module for complex moves with castling and other strategies
- Week 7: Allow for user creation of pieces and rules for them
- Week 8: Allow for the use of the pieces to be integrated into the board
- Week 9: Attempt stretch goals in order to allow playability with these pieces