Requirements

###### Introduction

ChessMob is a chess game where each player only has their king and a set of one other type of piece. Their choices will be either 6 queens, 7 rooks, 9 bishops, 13 knights, or 31 pawns. This applies a twist to the standard game of chess and will give ChessMob a unique edge in the market of Chess games. ChessMob will enforce piece movement rules and pawn replacement, but not the more specialized rules like castling and en passant.

Our client is Brian Hubbell, a Chess enthusiast, who presented the idea for ChessMob as a mobile application so that people can play while at the airport or on an airplane or wherever else they might like. We analysed several competitors out in the market for elements we could use in our application. The physical game of chess will be used for the board structure, the pieces and for the rules that need to be enforced. We looked at multiplayer games like League of Legends and would like to absorb their ranking system and matchmaking for multiplayer game play on ChessMob. We also looked at Chess applications currently published the iOS app store and the Google Play store, the monetization models used, and how they set up their graphical interfaces.

*Client Requirements*

After client presentations, we sat down and discussed this project in further detail. We agreed that we will be developing the application on the iOS and Android mobile device platforms. While the application will be available for tablets, we will be focusing on the phone paradigm. We have yet to decide whether or not the mobile applications will be free or paid. At a minimum, ChessMob will include player vs computer game play so the user can play wherever they are without the need for another person to be present and willing to play.

Example scenarios in which this software might be used are as follows:

* Playing against computer in airplane or other areas with limited internet connectivity
* Playing against another player either through the internet or local connection
* Same device play where two players pass the device back and forth to play a game
* Player playing against either computer or player where the game is continuous and cannot be saved and picked up at a later time
* Player playing against either computer or player where the game is non continuous, i.e. where the game can be left and picked back up at a later time

In regards to specific requirements of the application, it will require two major components. We will have our Graphical interface and our “Game Master” backend that will keep track of the game. The graphical interface will be the same on both devices and will include a Start screen, Game Setup screen, Board screen, Options Screen, Load Game screen, and a Network/Find Player screen. Screens may be merged, postponed, dropped or created as requirements update.

The backend will include load/save game functionality, a “Game Master” that will control the game state, remove pieces, enforce game rules and declare victory, networking components for player vs player online game play, and board set up based on user inputs. During play, the back end will validate player moves and keep track of the state of the game while the UI will take in inputs and keep the screen up to date. After each move, the back end will analyse the board for stalemate, check, or checkmate conditions.

Cost-Benefit Analysis of Features

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Effort (1-100)** | **Benefit (1-100)** | **Notes** |
| Game rules and setup | 60 | 100 | Basic functionality that is required for the app to operate correctly |
| Interface | 40 | 90 | Attractive graphics, user friendly interface, and tactile feeback |
| Human vs Computer Play | 80 | 100 | Allows user to play without another human player and in areas without internet access |
| Multiple levels of AI | 50 | 90 | Allows users to pick how difficult they want the computer player to be |
| Analysis of different setups | 70 | 90 | Analysis of each possible piece setup to find out how balanced the game is; for example, is 6 queens vs 31 pawns a fair matchup? |
| Balancing | 50 | 90 | Balancing the piece sets to make game play fairer |
| Human vs Human pass play | 30 | 85 | Allows two players to play on one device, passing the device back and forth after each move |
| Human vs Human network play | 60 | 80 | Allows users to play with each other over great distances and adds variety to the game play |
| Ranking | 90 | 90 | Gives users a rank to know how good they are compared to the world |
| Matchmaking System | 90 | 90 | Allows users to only play with people who are of similar skill level online |
| Save/Load Game | 60 | 90 | Allows users to store their game so that they can finish it later |
| multiple games | 70 | 40 | Saving and storing multiple games |
| Port games between devices | 50+? | 30 | Moving your saved games to other devices; low benefit due to fact that games against the computer or another player on the same device will either not be important or can be finished quickly enough to not need transferring and online games should be stored in the cloud |
| Tutorial/Help | 30-60 | 60 | Interface to supply instruction to the user when they have questions about how to play |
| Undo Button | 40 | 60 | Give ability to undo a move |
| Speed Chess Version | 40 | 20 | Augments the rules to make game play faster |
| Play offline | 20 | 90 | Gives ability to play without an internet connection |
| Support iOS platform | 50 | 90 | Develop application for iOS |
| Support Android platform | 50 | 80 | Develop application for Android |
| Support Window 8 platform | ? | 10 | Develop application for Windows 8 |

*Feasibility*

As the idea behind ChessMob is patented by our client and this is the first electronic version to be created, this program will not have to compete against other electronic versions of ChessMob. It will be competitive against the physical game of ChessMob as it allows the user to play without another player and it has the capability to be played in most environments without carrying around a lot of equipment. As this game is adapted from Chess with most of the same rules, Chess players should be able to easily pick up the game and play, making this program competitive by reducing the learning curve. As it is adapted from Chess, Chess programs will be competitors, but the unique setup of the game should make it different enough to compete with those programs. Furthermore, there is no intention to allow for regular Chess play within ChessMob, which will remove it somewhat from direct competition with other Chess programs.

The ability to develop applications for the iOS and Android platforms will be necessary. Within our group, we already have some experience with both Android and iOS development, but for the level of work this program requires, more knowledge will be needed. In addition, we intend to use a C++ backend for development of both of these platforms, which requires knowledge of interfacing the backend with the different platforms, something that is not part of our expertise as of yet. We will gain this expertise through the use of online tutorials, forums, and documentation along with some experimentation. For implementing network player versus player, we need knowledge on how the platforms connect with the network and how to develop networked programs. This is another area we have some knowledge in, but not enough for developing the program without further learning. We will gain this expertise in the same manner. Another technical area that we require more expertise in is mobile graphics. We are all in a computer graphics course currently, which will helps us gain the expertise and any further expertise needed we will gain through tutorials, forums, documentation, and experimentation.

Not all of the features previously listed will be implemented due to time constraints. We should be able to implement enough of the features to have a useable and competitive program. The minimum feature set that we need to implement for this program are: game logic, a working interface with board and pieces graphics, a computer player to play against, and development on iOS platform. These features, while challenging, should be able to be implement them within the time restraints. Other important features we believe we can implement are development for the Android platform and the capability to save and load games. Another feature that would be highly desirable would be player versus player, but we are unsure of the feasibility of this feature due to time constraints.

###### Lifecycle Model

We will be using the Agile Lifecycle Model for our application since we need to be able to adjust should our client’s needs change during development. We might also need to be able to make changes late in the development process, especially if our client is unavailable and we make a decision that needs to be changed later. Another feature of Agile we will be using is the setting of milestones for our project so that we have a more clearly defined development schedule, are able to keep our application bug free, and to prevent feature creeping.

We currently do not plan on making a rapid prototype due to the fact that most of the heavy logic and calculations will be in our backend while our interface design can be tested through the use of a storyboard. As such, we will be drawing up a basic storyboard for the application and then getting our client for his opinion on color and positioning of elements within each screen.

We chose agile over alternative models because it fits our situation better than anything of the others. Build-and-fix is simply horrible. The waterfall and Rapid Prototype models are too rigid and won’t allow for flexibility when requirements change or a redesign needs to be worked out. The project does not really lend itself to Incremental since the only real increments would be “interface complete” and “game is functional”. And Synchronize-and-Stabilize could be awkward depending on when developers can work on the project and since we will be parallelizing the interface and backend development, we will be able to synchronize and stabilize once both pieces are developed enough to be integrated with each other.

###### Initial User-Interface

At first we will have a drawing of each of the screens needed in the application to present to our client, in the form of a storyboard, in order to ensure that the visual components of our application are satisfactory to him. For a rapid prototype, it would be difficult to have it more complex than a drawing because most of the code handling the Chess game will be in our backend. We have created some images that represent our home screen and our game play screen which are located [here](https://docs.google.com/drawings/d/1OC69UDxWSUKMLhelk5DpuZ_559EO5T8hQqMBBBJD6_U/edit?usp=sharing).

###### Testing During Requirements Phase

In order to identify bugs or misconceptions early, we will be submitting a survey to our client in order to make sure that we are not missing any details needed at this point in the development process. This document is located [here](https://docs.google.com/document/d/1S7MouVSBshIkRt9rLWvUAC0NMd86ubZF2TxVSs0f9ew/edit?usp=sharing) and will be updated once we receive the survey feedback from our client. We will also be reviewing our requirements to identify redundancies and contradictions. This will hopefully help identify faults and design flaws early in our project which will make the development run more smoothly. We also have some notes from our initial meeting with the client stored in git repository and some on paper that are in Ace’s possession.

###### Presentation of Requirements to Client

We have sent a thorough survey to further clarify the client’s needs and help to identify misconceptions or missing elements that we missed during our initial meeting with our client. Included in this survey is the list from the cost benefit analysis of features. We have also sent a proposal for the UI of the application to the client. He will respond with desired changes. Our survey is located [here](https://docs.google.com/document/d/1S7MouVSBshIkRt9rLWvUAC0NMd86ubZF2TxVSs0f9ew/edit?usp=sharing) and our UI proposal is located [here](https://docs.google.com/drawings/d/1OC69UDxWSUKMLhelk5DpuZ_559EO5T8hQqMBBBJD6_U/edit?usp=sharing).