**Multiuser Poker Game for Mobile Devices**

December 6, 2018

**Matthew Winterbourn**

School of Computer Science

psymw6@nottingham.ac.uk

4276228

Supervisor:

**Milena Radenkovic**

School of Computer Science

pszmvr@nottingham.ac.uk

*Abstract*

In this dissertation we will present a mobile app designed for a game of poker aimed for ease of use for all types of end user. The app will consist of two components, the mobile app itself, designed to be deployed on Android devices, and a standalone server which the mobile app will connect to in order to play with other users. Throughout development we came to understand the difficulties in providing an engaging user experience for users of varying abilities as well as dealing with the concurrency issues that come with developing a multiplayer server designed to handle many games with multiple players at once.

# **1 Introduction**

* 1. History of Poker

The origins of Poker have been debated at length by the historical community. It is reported that “In the year 969, the Chinese Emperor Mu-tsung played a game of “domino cards” with his wife” (Neal, 2011), which many historians have believed was the first ever game of the card game now known as poker. However, in the 1937 edition of Foster’s complete, R.F. Foster wrote “the game of poker […] is undoubtedly the Persian game of As-Nas” (Foster, 1937). Although the exact origin of Poker is disputed, the modern game of Texas Hold’em we know came to prominence in the 1970s with the World Series of Poker popularising it in a tournament setting (Anon., 2007) and it is this method of play that has become wildly popular in the 21st Century, both in-person and online formats being played and enjoyed globally.

* 1. Project Introduction

From the desktop computer to the mobile phone, the world today is becoming increasingly connected, with over “77% of Adults in the US owning a smartphone in 2018” (Pew Research Center, 2018) and studies showing that “91% of people with disabilities in the US own or use at least one of a cell phone, smartphone or tablet” (Morris, et al., 2016).

Despite this, the Play store presents a relatively sparse offering of mobile games with features designed to make them more accessible to users with issues such as sight or hearing impairment, such as TapBeats which was specifically created with the intention of “being accessible for users who are blind or have low-vision” (Joy & Ricaurte, 2011). This limited offering can be seen on this forum thread (Anon., 2015) which has a short list of aggregated games that support accessibility features on Android, mostly focusing on those that support the Android TalkBack feature for screen reading and does not appear to feature any variation of a Poker game at all.

This project aims to build a multiplayer poker application to be deployed on Android devices. It will be capable of connecting the application client to a host server that will allow multiple users to play against each other and include features to improve the game’s accessibility for new and less able users.

In order to provide this we will build a robust and secure server to handle the connections between users and process as well as another unit to process the core game logic and update the user’s client appropriately. This should be handled in real time and provide handling for instances in which clients abruptly or unexpectedly lose connection to the server so as to provide a smooth flow of gameplay for all users. Furthermore the server will implement select algorithms to ensure the fair and random shuffling of cards between hands as well as to efficiently calculate the outcome of each hand.

On the client side of this application we will face several Human Computer Interaction challenges involving creating a suitable interface for the user to interact with that will be clear and understandable for users of all ability, by implementing optional features that will, for example, providing support for the Android TalkBack feature to navigate the application or an alternative UI layout that emphasises the functional components of the app over a more aesthetically focused layout.

1. **Motivation and Related Work**
   1. Motivation

Mobile gaming has fast become one of the most profitable markets for gaming in the world, with projected revenue for 2018 marking the first time that more than half of all game revenues will come from the mobile segment and with revenue from mobile games alone projected to become a “100 billion dollar market by 2021” (Newzoo, 2018), from this we can see the stunning growth in popularity of mobile games and especially “the rise of multiplayer games which accompanied the advent of third generation network (3G) technologies” (Liu & Li, 2011).

It seems essential that in this new age where anyone can open their phone and expect to load up a multiplayer game and have it work almost anywhere that the game servers supporting this should be able to provide a robust and stable experience in order to retain a player base in today's competitive market. Sensor Tower even projects the Apple app store alone to “reach 5 million apps by 2020” (Sensor Tower, 2016).

As highlighted in the introduction, a large majority of people with disabilities use cellular devices (Morris, et al., 2016), however there is a significant disparity in apps designed to cater this audience.

These key points have convinced me of the need to develop more accessible games for Android and will be a key aspect alongside the creation of a robust and enjoyable multiplayer game of Poker capable of handling multiple game sessions connected together to provide an inclusive experience for all users.

* 1. Related Work

1. Zynga Poker



Fig. 1 Main Menu screen of Zynga Poker App

One of the most popular poker apps available on Android, with over 50 million downloads.

Key Features:

* Brief tutorial to introduce the interface
* Friends system which allows importing of friends from Facebook
* Ability to adjust the game speed and table size you would like to play at
* Has a tournament game mode for users to compete in
* Slot machine for earning currency – linked to real money currency purchases

1. World Series of Poker



Fig. 2 Main Menu screen of World Series of Poker App

Named after the annual Las Vegas tournament, this app has over 10 million downloads on the Play Store and offers a robust gameplay experience with an overall clear layout.

Key Features:

* An extensive, narrated tutorial
* A chat system when playing at a Poker table
* Options menu has the option to show current hand strength, as well as disable extra features like the chat
* Friends system with Facebook integration
* A levelling system and VIP system tied to the user’s profile and playtime
* Displays game stats on the user’s profile
* Different levels of tournaments, with increasingly high buy-ins unlocked as the user ‘levels up’
* Can invite friends to join your current table
* Slot machine for earning currency – linked to real money currency purchases

1. World Poker Club



Fig 3. Main Menu Screen of World Poker Club App

Created by CrazyPanda, World Poker Club has a slick layout with labelled buttons for clarity, creating a satisfying and easy to navigate user experience.

Key Features:

* After missing a certain number of turns the user is marked as ‘AFK’ (away from keyboard), and their turn will be automatically skipped, but not removed from the table, until they come back
* Basic gesture controls for gameplay (bet, check, raise, fold etc)
* Scratch card minigame – linked to real money currency purchases
* Customisation of table settings, i.e the number of seats and blind amount
* Manual browsing through a list of tables
* Chat system with prearranged messages

Overall Poker apps tend to share a lot of functionality such as a friends system, user profiles and levelling, and the ability to buy currency to keep playing. With this being the case this project will attempt to build a similarly structured app, but with a focus on the core gameplay – Poker, over pointless minigames and systems designed to entice you into spending money. Furthermore, we will attempt to build a functional interface with a simple and clear layout that is highly learnable for all types of users in order to distinguish from other apps which appear to focus more on aesthetics over a user-friendly interface.

1. **Description of the work**
   1. The work

This app is designed to function on Android devices and intends to provide a base game experience of Texas Hold ‘em Poker without the extraneous additions of features like real money transactions for currency which can help facilitate gambling addictions and instead creating a simple app the is easy to pick up and understand and be used by all types of users, including those who might have restricted ability such as limited sight or motor capacity.

* 1. Functional Requirements

1. **Platform:** The application will run on all Android devices supporting the minimum API level specified in its creation.
2. **Login:** Users will be presented with a choice of methods to login to the app.
   1. Users will be able to login with a Google account

Users will be able to login with a guest account

* + 1. Users will be able to link their Facebook or Google account to the guest account

1. **Accessibility:** Users will have the option to enable features to increase the accessibility of the app.

The app will support the Android TalkBack feature to the best of its ability

The app will support an alternative layout mode which focuses on making the key gameplay elements as visible as possible (henceforth referred to as the ‘functional layout’)

The app will support gesture based controls

The app will support voice command controls

1. **Connections:** The app will handle connections from multiple clients in order to facilitate a Poker game between users
   1. The server will match users together from a queue to create virtual “Tables” where the game will be played and removing them from the queue
   2. The app will allow users to leave a table at any time, forfeiting any bet chips and returning to the main menu and disconnecting from the server Table
   3. The server will check users current connection status and:
      1. Remove users who have lost connection to the server from the current match
      2. Update connected users with information on the current game state
2. **Algorithms:** The app will implement algorithms in order to calculate the winner of each hand and to shuffle the cards between hands
   1. Durstenfeld’s improved Fisher-Yates shuffle algorithm will be used in order to shuffle the deck of cards
3. **Data storage:** User data will be stored appropriately by the server
   1. User login data will be stored in a database server side to validate login requests
   2. User game data will also be stored server side for security and stability
4. **Game functions:** The app will provide an interface for the user to make game decisions with that will trigger the relevant actions by the server
   1. Users will be able to take the standard actions permitted within a game of Texas Holdem like Check, Bet, Fold etc with only relevant actions being shown in the screen
   2. The server will receive each user’s actions and apply them to the current game state and update accordingly
   3. User’s will have a common virtual currency, which will be used to bet with in game and will be carried between game sessions
      1. Users will be granted an amount of currency at account creation
      2. Users will be able to win and lose this currency in hands
      3. Users will be provided with currency on a daily based, scaled with login activity
5. **Interface:** The app will provide different menus, implemented as “Activities” in Android that can be navigated between to perform certain actions.
   1. The app will provide a login screen for users when the app launches until they choose a method of login
   2. The app will provide a main menu as a hub to provide access to the options, friend, chat, and game help menus.
   3. The app will provide an options menu to adjust options pertaining to accessibility features, general game features, and connected accounts
   4. The app will provide a friend menu for viewing friend’s profiles and adding friends
   5. The app will provide a chat screen for users to talk in from the main menu and the in game Poker Table screen
   6. The app will provide a game help screen to display the rules of poker for beginners, as well as a reminder of hand values
   7. The app will provide a Poker Table screen where the actual game will take place between users
   8. The app will provide a user stats screen displaying statistics saved in the user’s profile data on the server
   9. Non-Functional Requirements
6. **Usability:** The app should feature introductory tutorials designed to introduce key features of the User Interface, as well as how to use the main accessibility options. Otherwise use should be straightforward even for users with low experience with similar apps using highly learnable buttons that are easily recognisable, and minimising the number of activities displayed to the user
7. **Reliability:** The server side implementation will secure the stored data to prevent unauthorised users accessing it, and will be built to be robust and handle numerous connections at once without failure
8. **Performance:** Communication between the client and server should be completed with minimal delay, and the algorithms implemented in the server for calculating hand value and shuffling cards should not cause any noticeable slowdown when handling requests or sending game data. Additionally the app itself should not present any noticeable lag to the user when performing actions such as switching activities in order to maintain a smooth and enjoyable user experience
9. **Implementation:** The app will be created primarily in Java in order to be deployed on the Android Operating System with a corresponding server built in Java to communicate with
10. **Methodology**
    1. Key Features
11. **Google account** **integration:** A staple of modern mobile gaming, account integration allows a smoother user experience, without the hassle of creating any new accounts and the convenience of being able to import their friends as well
12. **Accessibility**
    1. **Change layout mode:** An optional layout mode that focuses on delivering the main functions of the app, i.e. the buttons and removing the unnecessary menu and game graphics
    2. **Gesture control scheme:** The option to navigate between Activities using gestures and control the game with gestures which will aim to be highly learnable and an effective alternative control scheme for the game
    3. **Android TalkBack support:** A function designed to improve the user experience for users who have low-vision or are blind, TalkBack will be supported as fully as possible to be able to navigate and use the app
    4. **Voice command support:** Another form of control for the app, allowing users to interact with the app utilising voice commands, however is held at a lower priority due to the relative complexity of implementation
13. **Virtual-only currency:** The currency to be used in the game, which will have no real monetary value, and will only be possible to earn by playing the game, without the addition of any ‘micro-transactions’ in order to create a ‘pure’ poker game, without the extra clutter of so called ‘mini-games’ that seem to be prominent in the most popular of games and the enticement to spend real money
    1. Development Process

Although the original intent for this project was to adapt Agile methodology in order to utilise Test Driver Development principles and be flexible in changing requirements due to user testing, we discovered early on into the development process that Test Driven Development would not prove to be practical with the work schedule set out. This was due to the fact that the server and client were being developed separately, and not in tandem and as such in order to test whether each component was working as intended the tests written would need to use a dummy server or client to correspond with the component being tested. Essentially this would mean creating 2 servers and 2 clients in order to properly test functionality which would be impractical given the limited time frame for development and so the decision was made to steer away from Test Driven Development and to simply develop bother core systems in parallel. This led to building the corresponding components alongside each other and using these to essentially perform testing on each other, reducing the need for written tests with functionality being shown to work by correct communication between the client and server components being built.

Tests were still developed for certain components of the server, specifically supporting classes like the HandEvaluator class which is intended for use as part of the game logic of the server to determine the winner of hands played by users. As these components were purely intended for use server- side with no communication with the client the testing was fairly straightforward and helped detect and eliminate defects in relatively complex logic with ease.

The algorithm used within the aforementioned HandEvaluator class in order to actually evaluate poker hands is a simple brute force algorithm. The strength of this approach is the relative ease of implementation with there currently being no need to evaluate millions of hands at a large scale in the app build. The main issue with this approach would be the lack of scalability and would need to be overhauled entirely if it was a key focus. After researching into the use of algorithms such as ‘Cactus Kev’s’ 5 card evaluation algorithm and attempting to adapt similar methods involving the use of lookup tables and bit shifting in order to achieve the fastest possible computation of poker hands we determined there would not be enough development time to completely implement such an algorithm and so we had to settle for simply using brute force.

The Durstenfeld algorithm for shuffling cards in our Deck object is a concise and elegant solution that goes through the list of items and swaps the last entry that has not been chosen with a random entry until there are no entries left that haven’t been swapped. This solution produces an O(n) complexity which is optimal for the problem it solves.

1. **Design**
   1. Design Overview

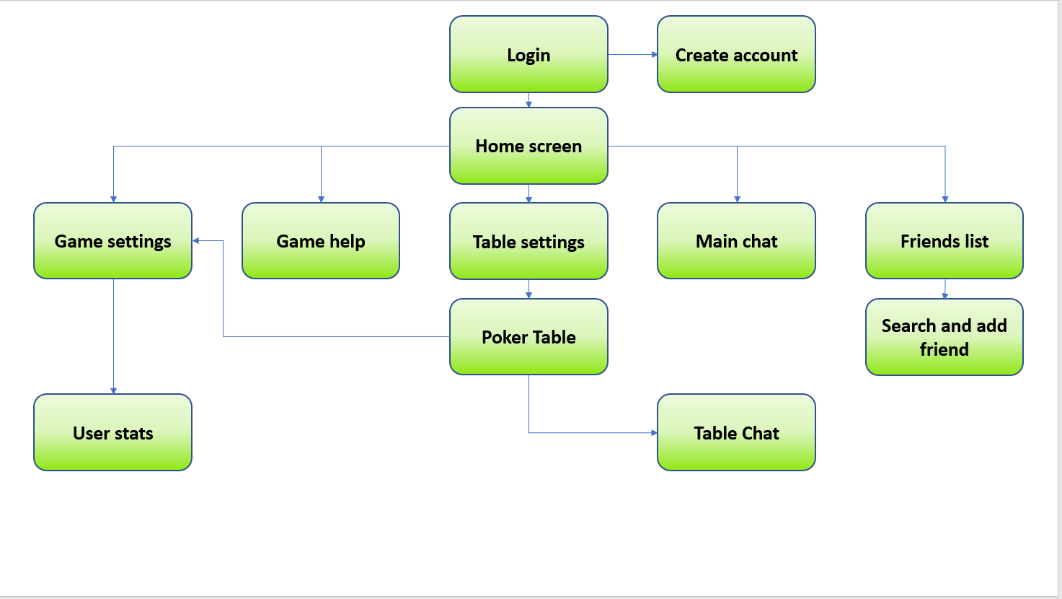


Fig. 4 App screen-relationship diagram

The core design of the project will be in two parts, following a client-server model. The application which will be released on Android devices will act as the client, and interface with the server, which will be developed separately.

The server will be created from scratch in order to have full control over the desired functionality of the system. The server will also have an attached mySQL database to handle user authentication for guest accounts and linked Facebook and Google accounts. This data will be stored online as if it is stored in the Android internal storage database provided it will have to be encrypted and could still potentially be tampered with if the encryption is not strong enough.

The server will have multiple responsibilities in communicating with the client. For all users the first thing it will do is add them to a separate handler thread which will authenticate their login ids and retrieve their profile. After this the client will be able to join a queue which will place them in games with other players and handle the relevant game logic in a separate thread during which the original handler thread will be idle. Users will be able to leave their game and communication will be returned to the original handler thread which can then re-add them to the queue or send them an updated user profile if the user wishes to view their profile.

The client consists of multiple Activities presenting information to the user and communicating with the server when necessary. These Activities are presented as screens to the user and will provide links to navigate between them (Fig. 4).

Development of the server will be done in Java as this will provide the easiest method of communication with the client which will be written using the Android Studio Java implementation and the objected-oriented design will facilitate the use of several key design patterns such as the Model-View-Controller pattern and the Command design pattern, the former to be used as part of the client and the latter as part of a communication protocol between both the client and the server.

* 1. Prototypes

Prototyping the User Interface commenced early in the project, in order to promote an iterative, user focussed design methodology. Low Fidelity prototypes were developed to this end and testing was used to gauge the potential learnability and usability of the prototype. The prototyping was software aided in order to add interaction so that users could get a feel for how the app would run and provide more relevant feedback.

The initial prototype demonstrated the overall layout of the main features of the app (Fig. 5 & 6). This prototype had attempted to make use of the space in the screen to only display information that was seen as important, with the links to other screens placed in the corner, out of the way of the main interface. The initial prototype did not have any designs for the optional ‘functional layout’ outlined in the Requirements section, Requirement 3.2, as it had only been conceived as a result of feedback from users.

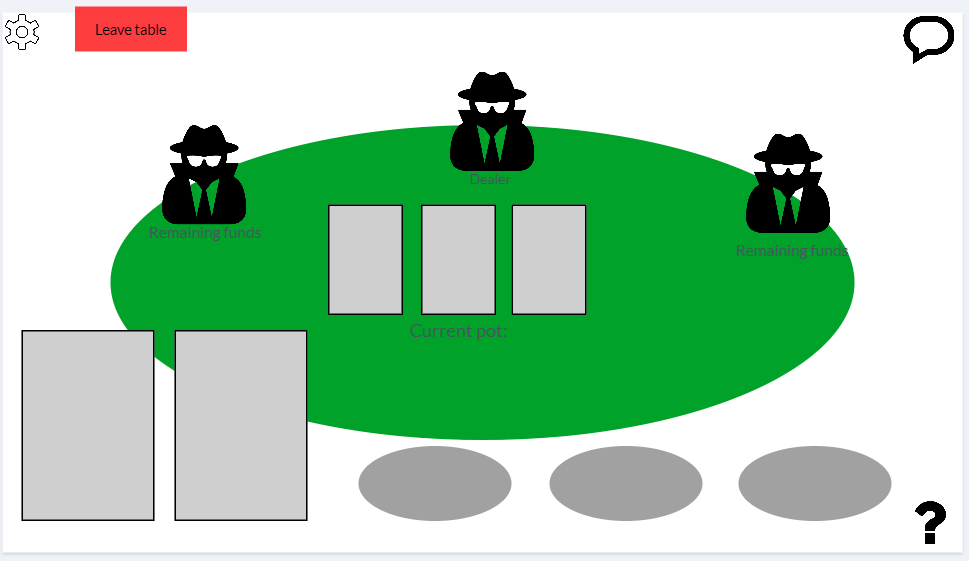


Fig 5. Table screen of initial app prototype

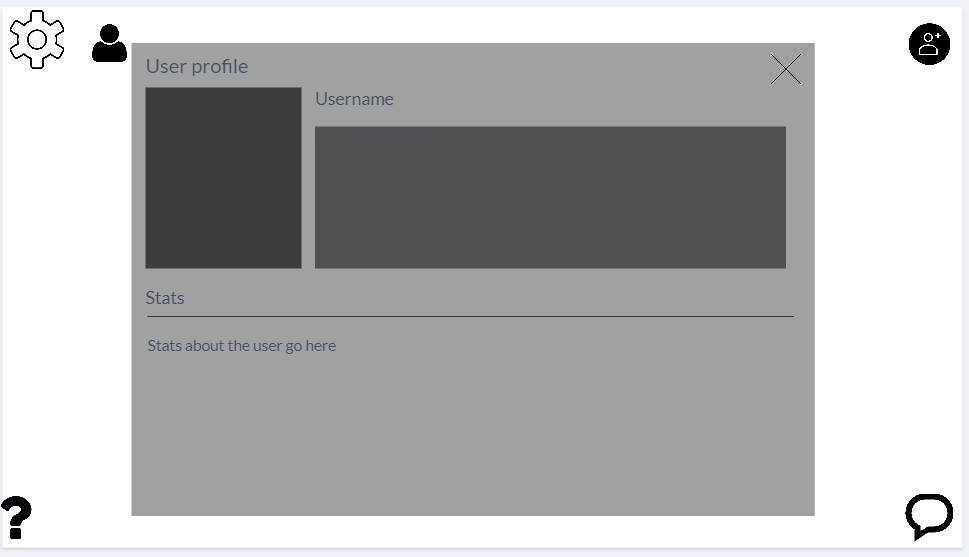


Fig. 6 User profile screen of initial app prototype

This prototype was changed after a round of testing and feedback as a result of the formative evaluation process. The 2nd prototype introduced the aforementioned ‘functional layout’ which had a higher focus on making every functional aspect (buttons) the main focus on the menu screens, and further increase their visibility in the game screen (Fig. 7).

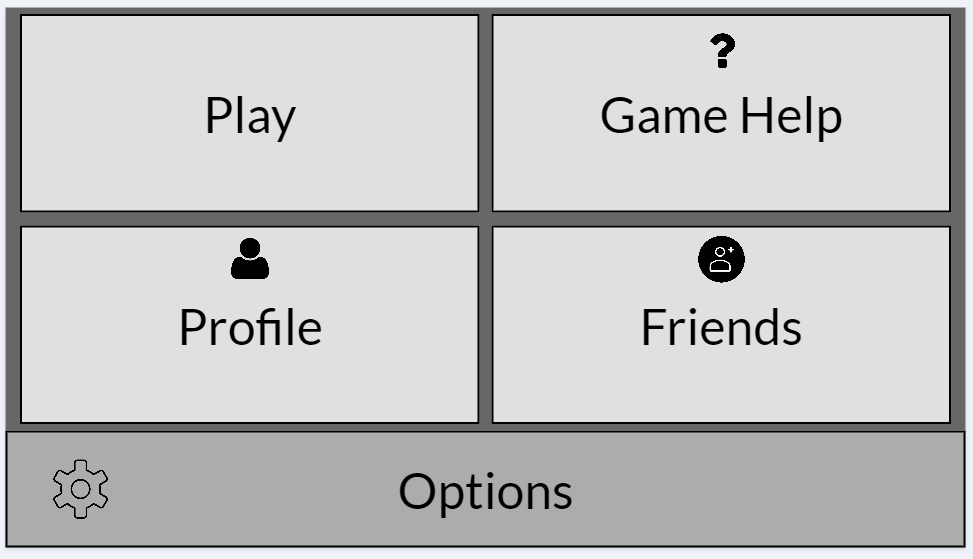


Fig. 7 ‘Functional layout’ Main Menu screen of prototype 2

This prototype was further refined to simply display text, without the associated icons as user’s felt they had no use when the function was already made clear with the text description.

An issue raised in the evaluation of this prototype was the lack of chat option in the functional layout, however we decided that the type of user that would elect to use the more visibility-centric layout is likely to either be visually-impaired or prefer to just focus on the actual game experience and as such the lack of chat visibility was not an issue.

Another major change in this prototype occurred in the normal layout of the main menu screen, with the icons that were initially in the corners representing links to different activities being moved to a navigation bar at the bottom of the screen, with a brief text description reminiscent of the design seen in Fig.3 in Section 2, which in particular was evaluated as being much clearer and easier to learn for new users who might be unfamiliar with similar technologies. Accompanying this was similar changes to the actual game screen, which also favoured larger buttons and clearer presentation whilst sacrificing aesthetic layout (Fig. 8).

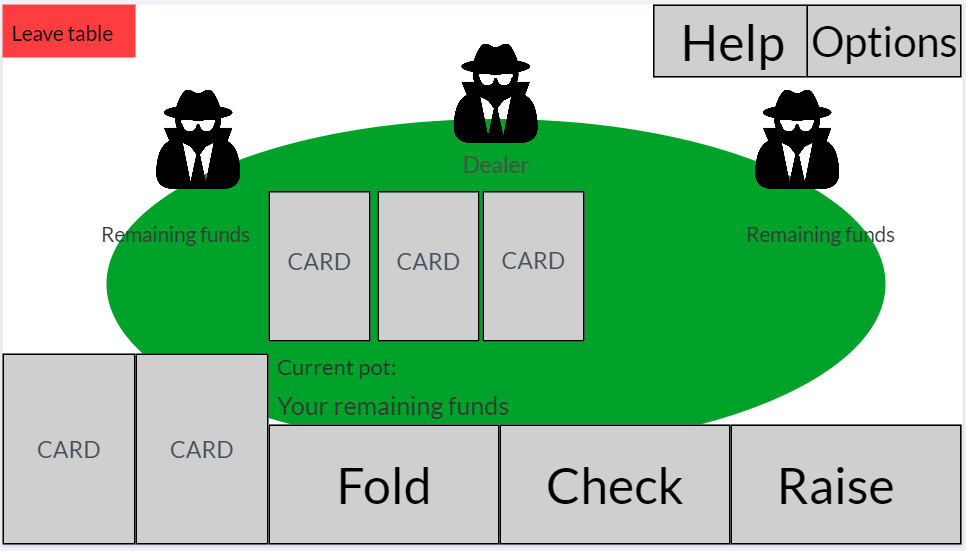


Fig. 8 ‘Functional layout’ Table screen of prototype 2

* 1. User Testing

Initial testing was conducted with a small user group of 3 people, in order to keep the feedback manageable and be able to review it quickly to implement changes to the prototype.

The test group comprised of 3 different types of user, the inexperienced, moderately experienced and experienced smartphone user. Testing proceeded in the form of a cooperative evaluation, and users were encouraged to share their thoughts as they navigated the interface, following a ‘think aloud’ protocol, from which notes were made. The advantage of this sort of testing is that the tester can see the user’s first reactions to the prototype and better understand the concerns and problems that might arise than if the user simply filled in a questionnaire.

The testing process involved a set of 3 tasks for each user, designed to allow them to interact with all the interfaces present in the prototype. The time taken to complete each task was noted and compared between users to understand how usable the system is. The user’s comments as they were performing the tasks were also noted to understand what users think during use. After the tasks were completed users were asked to give 2 ratings, the first for how obvious the steps that needed to be taken to complete each task was, and the second an overall rating on how usable the felt the system was.

Further testing will be conducted later in development of the front end in order to ensure that it is built to be as user friendly and bug free as possible. This testing will be both user testing as well as functional testing using JUnit as part of the Test Driven Development methodology.

1. **Implementation**
   1. Backend Server

**Language and dependencies:** The Backend Server was written using Java SE 11, being the latest release of the powerful Java platform intended for long term support by Oracle which provides security and bug fixing updates for years into the future which will be useful to reduce the need for refactoring with new Java releases such as the jump between Java 9 and 10 and will be useful for maintaining the project over time.

Application dependencies include the Google API for Java Clients (Google, 2019) which facilitates the usage of Google Account integration using tokens that can be verified through their API to retrieve user’s details and securely log them into their accounts. The other key dependency is the SQL Java Database Connection (JDBC) package (Microsoft, 2019) which allows for Java code to connect to MySQL databases, crucial for integrating the SQL database that is used for storing user accounts and their associated details.

**Database connection:** The server contains 2 classes encapsulating the core logic of accessing the database using the JDBC package. The base class, SQLDatabaseConnection consists of basic methods to connect and disconnect to the database server (Fig. 9), as well as a method to create SQL statements to be executed on the database.

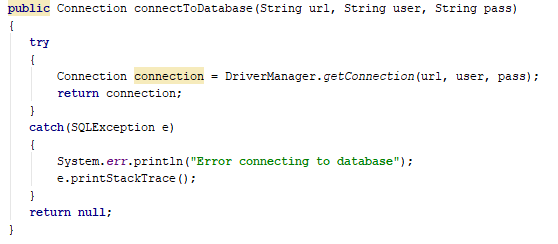


Fig. 9 Example method from the base database connection class

The second class uses inheritance in order to implement these basic connection functions and extends the functionality to create database queries that are specific to the server’s needs. This class serves as the interface object for server threads to access the database. This object allows for the abstraction of database communication in the main server logic as all the queries are handled within the class and all the handler need do is call the appropriate method, this is used every time a client connects to the server to retrieve their user details for example (Fig. 10). A separate query object is maintained for each handler thread and only connects to the database when necessary in order to reduce overhead making such calls.

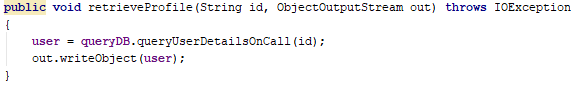
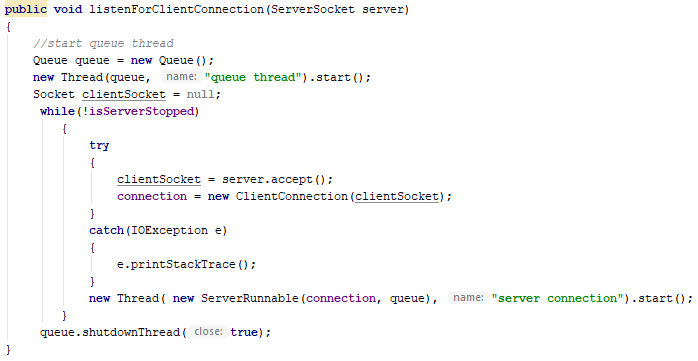


Fig. 10 Example method using the database connection object to retrieve a user’s profile details

**Handler thread:** The server handles new connections by continuously accepting new connections in a main thread loop and for each connection a new ClientConnection object is created which contains the socket and associated output and input streams that will be used anytime communication between the server and client is required for the lifetime of the connection and then adding them to their own thread handler by passing the ClientConnection object into a separate thread which will exists as long as the connection is not closed (Fig. 11)

Fig. 11 Accepting new connections to the server and passing them to their own handler thread

The handler thread is key in providing the client with requests outside of the actual game, these requests include retrieving the user’s details on login (Fig. 12), obtaining an updated user profile when viewing their profile, linking their guest account to a google account, and adding them into the queue to join a game. Since each thread operates independently for each user there are no concurrency issues with this solution however a drawback of this is that we were unable to prevent users from being logged in with the same account on multiple devices.

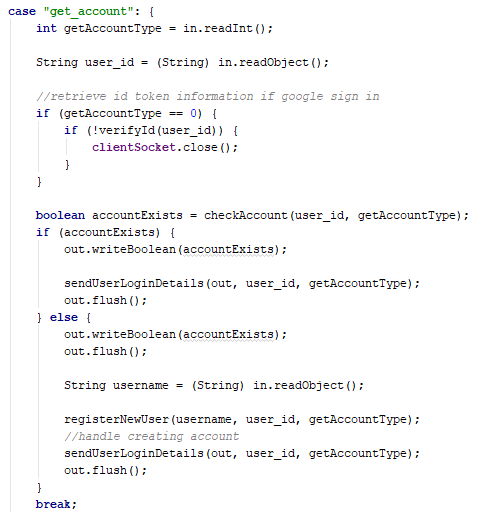


Fig. 12 Example of handler thread responding to a client request to retrieve their user account

The largest issue encountered with implementing this handler thread was to be able to smoothly remove and restore control to it when a user joined and exited a game. The issue that existed initially was the handler thread taking priority in using the client’s input and output streams even after they had been passed to the appropriate game thread. In order to fix this we stopped the handler thread from iterating over its main loop reading input from the client whilst the client was considered ‘ingame’. This produced a further issue that meant that after the client left the game the handler thread did not resume communication as it had no way of telling when the client was no longer ‘ingame’. The fix to this was to introduce callbacks from the game thread that notified the handler thread of the client leaving allowing it to resume the main loop for client communication.

**Game objects:** In order to

* 1. Mobile Client

Asd

* 1. Database

The database structure is relatively simple, consisting of 2 tables; the ‘users’ table and the ‘details’ table. These store all the relevant account data required for users to play the game and have their stats persist between sessions. It also provides a layer of security to prevent exploits as it is all stored on a server and accessible by the main backend server, so that users are not able to edit their own details or spoof them in some manner to fool the server.

The ‘users’ table holds the google user id, guest user id, username, last login date, and currency amounts for each user, these are essential fields for each user with at least one of google or guest user id needing to be provided for each user to be able to retrieve their credentials and details.

The ‘details’ table holds non essential fields that correspond to certain statistics measured as each user plays the game. These include the amount of hands played and win rate, among others and are not necessary to play the game hence are stored in a separate table.

1. **Evaluation**
   1. User testing
2. **Summary and Reflections**
   1. Project Management
   2. Achievements and Contributions
   3. Project Appraisal
   4. Future Considerations

# References

Anon., 2007. *World Series of Poker Retrospective: Horseshoe History.* [Online]   
Available at: https://gaming.unlv.edu/WSOP/history.html  
[Accessed 14 November 2018].

Anon., 2015. *AudioGames.net Forum.* [Online]   
Available at: http://forum.audiogames.net/topic/13872/android-accessible-games-please/page/2/  
[Accessed 10 November 2018].

Foster, R. F., 1937. *Foster's Complete Hoyle: An Encyclopedia of Games.* New York: Frederick A. Stokes.

Google, 2019. *Google API Java Client.* [Online]   
Available at: https://developers.google.com/api-client-library/java/  
[Accessed January 2019].

Joy, K. & Ricaurte, J., 2011. TapBeats: Accessible and mobile casual gaming. p. 2.

Liu, Y. & Li, H., 2011. Exploring the Impact of Use Context on Mobile Hedonic Services Adoption: An Empirical Study on Mobile Gaming in China. *Computers in Human Behavior,* 27(2), p. 9.

Morris, J. T., Sweatman, W. M. & Jones, M. L., 2016. Smartphone Use and Activities by People with Disabilities: User Survey 2016. *The Journal on Technology and People with Disabilities,* p. 18.

Neal, A., 2011. *The Evolution of Poker, From Origins to Texas Holdem,* s.l.: Ezine Articles, http://ezinearticles.com/?The-­Evolution-­of-­Poker,-­From-­Origins-­to-­Texas-­Holdem&id=6544601.

Newzoo, 2018. *Global Games Market Report,* s.l.: Newzoo.

Pew Research Center, 2018. *pewinternet.org.* [Online]   
Available at: http://www.pewinternet.org/fact-sheet/mobile/  
[Accessed 15 November 2018].

Sensor Tower, 2016. *Apple's App Store Will Hit 5 Million Apps By 2020, More Than Doubling Its Current Size.* [Online]   
Available at: https://sensortower.com/blog/app-store-growth-forecast-2020  
[Accessed 12 November 2018].