Goldfish Software Specification 1.0

Team 11

By Kunal Paode, Jeffrey Yu, Tommy Kung, Zachary Qin, Chenhao Yang

Table of Contents

Glossary

1. GUI (Graphical User Interface)

   -an interface that uses icons/menus, and a mouse to manage interaction with the system

2. Doubly linked list

   -a linked data structure that consists of nodes that have pointers to the previous and next node. In which navigation is possible in both directions.

3. Enum

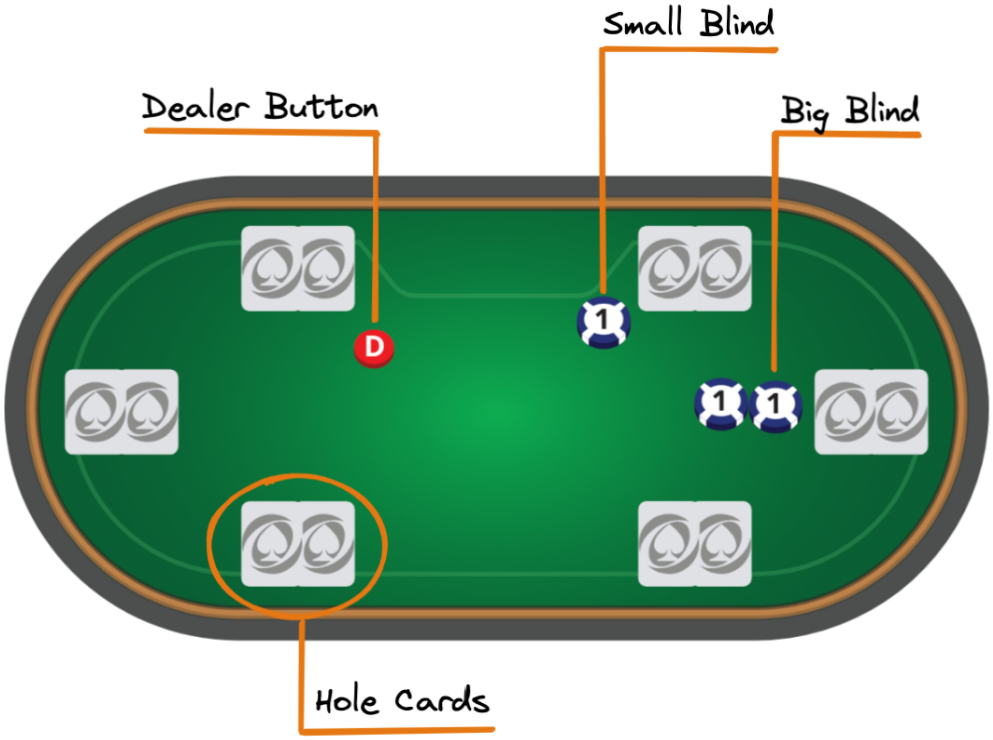
   -a data type that allows assignment of naming conventions to constants

4. Struct

   -structure is another user defined data type available in C that allows to combine data items of different kinds.

5. Gamestate

   -current set of information regarding a Table struct's variables, including all its players information corresponding to it



6. Dealer Button
   A round disc that sits in front of a player and is rotated one seat to the left every match. Helps to determine which player at the table is the acting dealer.

7. Small Blind
   A forced bet that begins the wagering. The player directly to the left of the button posts the small blind. Generally half the amount of the big blind.

8. Big Blind
   A forced bet that begins the wagering. the player directly to the left of the small blind posts the big blind.

9. Hole/Pocket Cards
   Two face down cards dealt to each player at the start of the game. Hole cards are used in combination along with the community cards to build a player's best possible five-card poker hand.

1. Community Cards
   Five face-up cards free for each player to use in combination with their hole cards to build the best possible five-card poker hand.

2. Flop: Round in which the first three community cards are dealt

3. Call: Match the betting amount of the big blind

4. Raise: Increase the bet within specific limits of the game

5. Fold: Throw hand away. Cards go into the muck

6. Check: Pass the action to the next player in hand (do nothing)

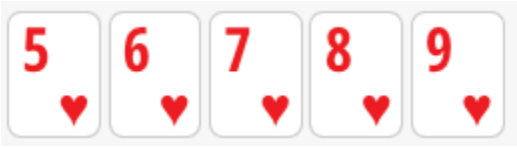
7. All-in: To bet all of a player's money. A player is not forced to bet on proceeding rounds of the current hand as they have no money to bet.

8. The Muck: Pile for cards no longer in play

9. Showdown: More than one player remains at the last round, so each remaining player shows their hand to determine who has the best

10. Five-Card Poker Hands (Best to Worst)
    Note: There is no Suit-Ranking in Texas hold'em
        1. Royal Flush: The best possible hand: 10, J, Q, K, A…all same suit

2. Straight Flush: Five cards of the same suit in sequential order



Note: Straight Flush can be ranked best to worst based on highest card

3. Four-of-a-Kind: Any four numerically matching cards

Note: in this example, the King of Diamonds can be replaced with any other card and this hand still applies

Note: Four-of-a-kinds can be ranked from best to worst based on highest four-of-a-kind

4. Full House: Combination of three-of-a-kind and a pair in the same hand

Note: Full Houses can be ranked from best to worst based on highest three-of-a-kind

5. Flush: Five cards of the same suit in any order

Note: a Flush can be ranked from best to worst based on first highest-ranking card, then second, and third, etc…

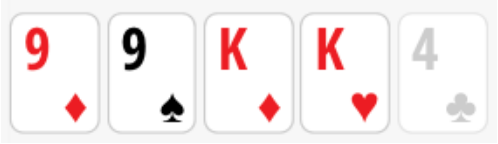
6. Straight: Five cards of any suit, in sequential order

Note: a Straight can be ranked from best to worst based on highest-ranking card

7. Three-of-a-Kind: Any three numerical matching cards

Note: in this example, the Four of Clubs and Five of Hearts can be replaced with any other card and this hand still applies
Note: a Three-of-a-kind can be ranked from best to worst based on highest three-of-a-kind

8. Two Pair: Two different pairs in the same hand
Note: in this example, the Four of Clubs can be replaced with any other card and this hand still applies
Note: a Two Pair can be ranked from best to worst based on highest-ranking pair, then second-highest pair, then highest-ranking final card



9. One Pair: Any two numerically matching cards
Note: in this example, the Three of Spades, Queen of Diamonds, and King of Clubs can be replaced with any other card and this hand still applies
Note: a One Pair can be ranked from best to worst based on highest-ranking pair, then (out of the 3 remaining cards,) highest-ranking cards

10. High Card : The highest ranked card in your hand with an ace being the highest and two being the lowest
Note: in this example, the Two of Hearts, Four of Diamonds, Eight of Diamonds, and Queen of Spades can be replaced with any other card and this hand still applies
Note: a High Card can be ranked from best to worst based on highest-ranking card, then second-highest card, etc…

Player Software Architecture Overview

1.1 Main data types and structures

Main enums:

```
typedef enum {
    heart = 0,
    diamond = 1,
    spade = 2,
    club = 3,
    unknownSuit = 4,
    blankSuit = 5
} Suit;
typedef enum {
    two = 2,
    three = 3,
    four = 4,
    five = 5,
    six = 6,
    seven = 7,
    eight = 8,
    nine = 9,
    ten = 10,
    jack = 11,
    queen = 12,
    king = 13,
    ace = 14,
    unknownNum = 15,
    blankNum = 16
} Num;

typedef enum {
    dealer = 0,
    smallBlind = 1,
    bigBlind = 2,
    none = 3
} Button;

typedef enum {
    null = 0,
```

```
    check = 1,
    call = 2,
    raised = 3,
    fold = 4,
    blind = 5
} MoveChoice;
```

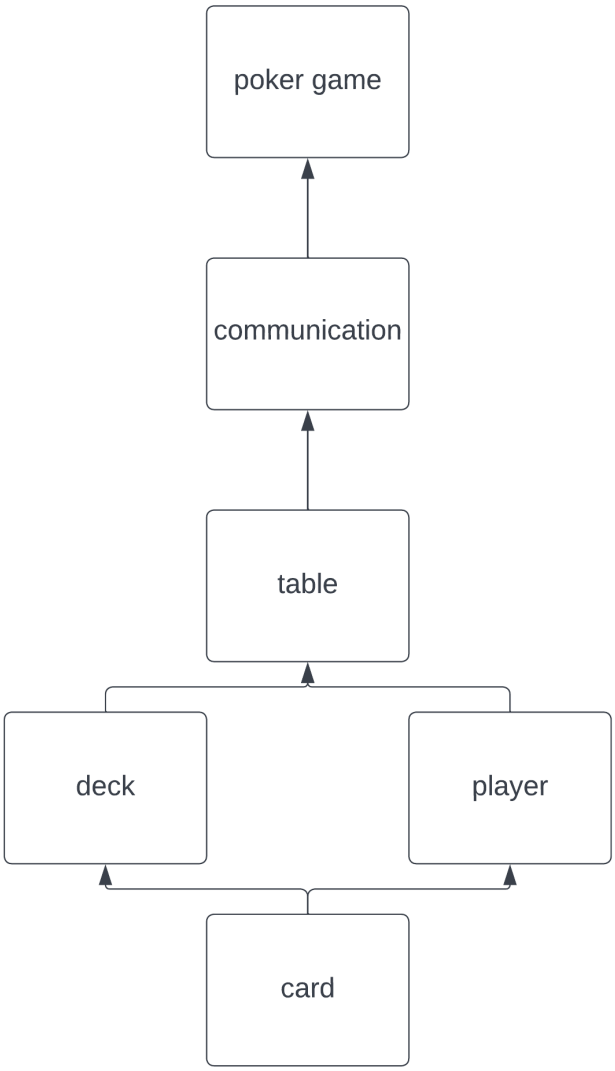
Main structs:

```
typedef struct {
    Suit suit;
    Num num;
} Card;

typedef struct {
    int totCard;
    Card cards[52];
} Deck;

typedef struct {
    Button token;
    int money;
    Card pocket[2];
    int currentBetAmount;
    int position;
    MoveChoice moveChoice;
    char username[50];
} Player;

typedef struct {
    MoveChoice choice;
    int raiseAmount;
} Move;

typedef struct {
    Player* players;
    Deck deck;
    int totPlayers;
```

```
    int totActivePlayers;
    int pot;
    int dealerNum;
    int currentMinBet;
    Card tableCards[5];
} Table;
```

1.2 Major software components

- Diagram of module hierarchy



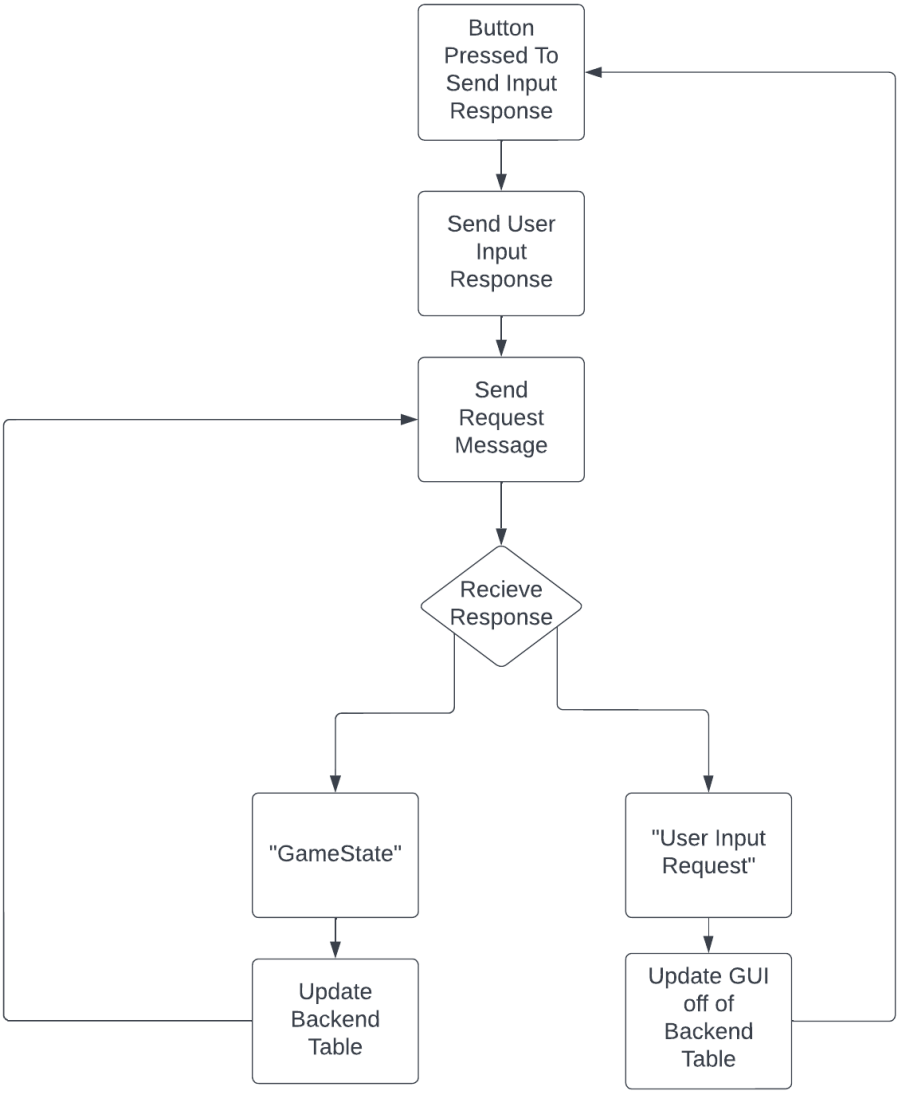
1.3 Module interfaces

- API of major module functions

-The Client GUI accepts the numbers of players and table position when the user clicked on the setting buttons. The move buttons on the bottom right declares what moves then user is doing and click send to send their move to the server. The green square shows when it is your move again as it switches between red and green.

1.4 Overall program control flow

Client Control Flow



Poker Server Software Architecture Overview

2.1 Main data types and structures

Main enums:

```
typedef enum {
    heart = 0,
    diamond = 1,
    spade = 2,
    club = 3,
    unknownSuit = 4,
    blankSuit = 5
} Suit;
typedef enum {
    two = 2,
    three = 3,
    four = 4,
    five = 5,
    six = 6,
    seven = 7,
    eight = 8,
    nine = 9,
    ten = 10,
    jack = 11,
    queen = 12,
    king = 13,
    ace = 14,
    unknownNum = 15,
    blankNum = 16
} Num;

typedef enum {
    dealer = 0,
    smallBlind = 1,
    bigBlind = 2,
    none = 3
} Button;

typedef enum {
    null = 0,
    check = 1,
    call = 2,
    raised = 3,
    fold = 4,
```

```
    blind = 5
} MoveChoice;
```

Main structs:

```
typedef struct {
    Suit suit;
    Num num;
} Card;

typedef struct {
    int totCard;
    Card cards[52];
} Deck;

typedef struct {
    Button token;
    int money;
    Card pocket[2];
    int currentBetAmount;
    int position;
    MoveChoice moveChoice;
    char username[50];
} Player;

typedef struct {
    MoveChoice choice;
    int raiseAmount;
} Move;

typedef struct {
    Player* players;
    Deck deck;
    int totPlayers;
    int totActivePlayers;
    int pot;
    int dealerNum;
    int currentMinBet;
```

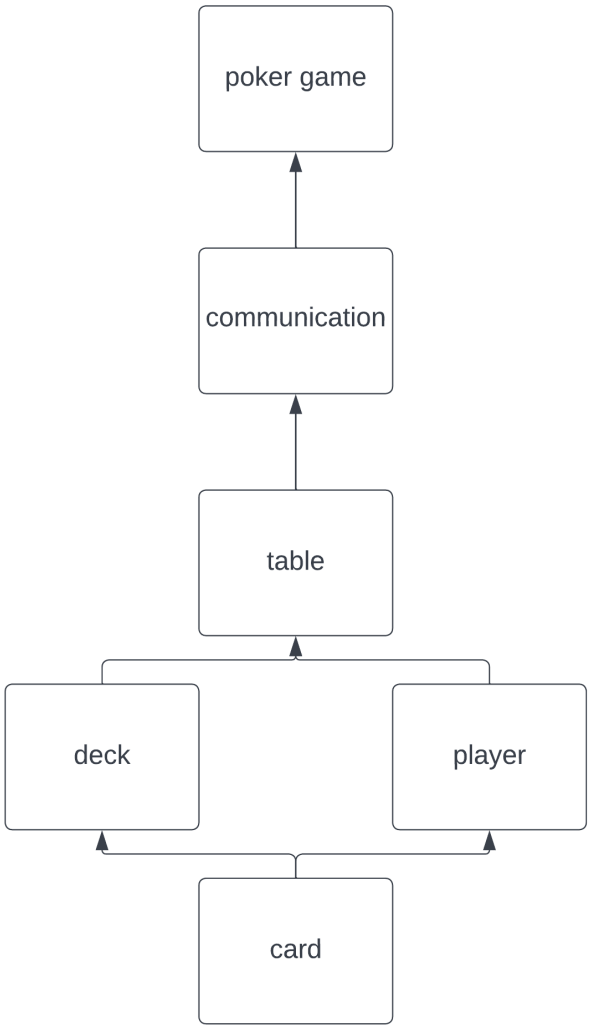
```
    Card tableCards[5];
} Table;
```

Function Types:

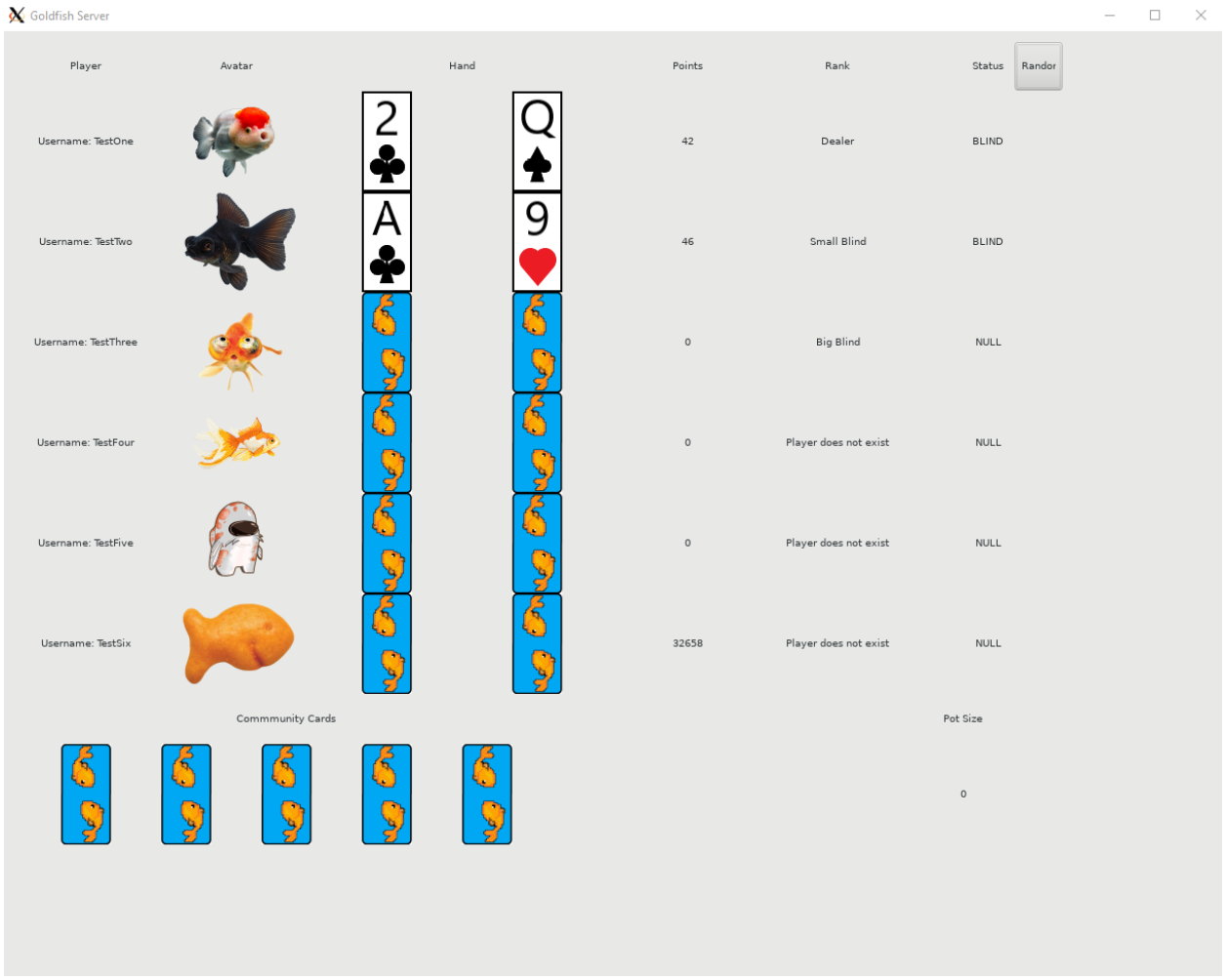
```
typedef void (*TimeoutHandler)(Table table);

//New Server
typedef Move (*ClientHandler_V2)(int DataSocketFD, int playerNum, int
*quasi_shutdown, Table table);

typedef Move (*MoveGetter)(int ServSocketFD, int Timeout, fd_set*
ActiveFDs, Table table, int playerNum, ClientHandler_V2 HandleClient);
```

2.2 Major software components

- Diagram of module hierarchy

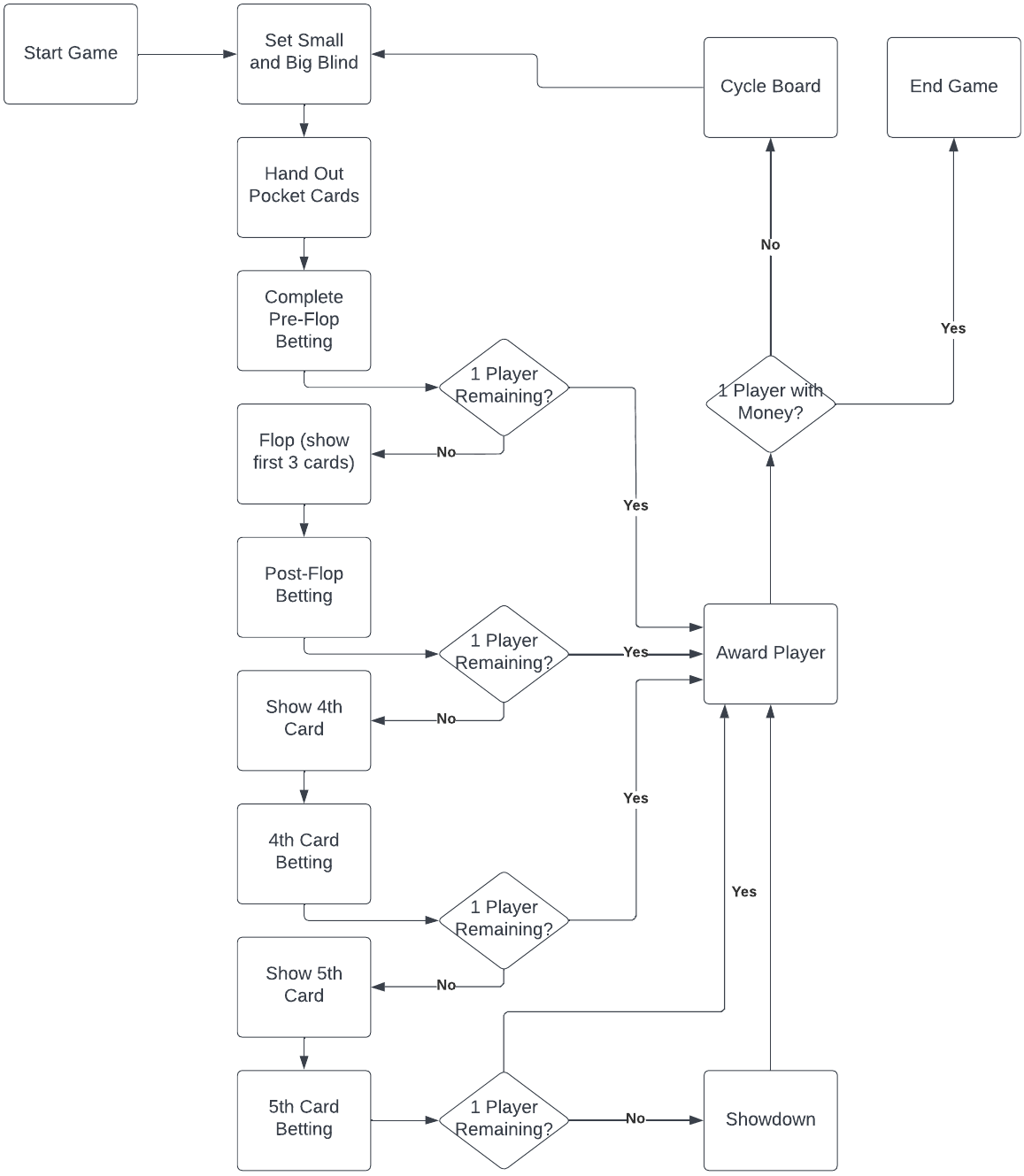


2.3 Module interfaces



2.4 Overall program control flow

Server Game Control Flow:



Installation

3.1. System Requirements

Linux with display function
std=c11 enabled
gtk library installed

3.2 Setup and configuration

Download tar.gz file and Makefile
Extract tar.gz file with command “gtar xvzf”
[Linux] Type “make all” to build program
[Linux] One Player host the server Type “./bin/GoldfishServer 10111”
[Linux] Type how many players (1- 6)
[Linux] Type “./bin/GoldfishClient [Servername] 10111”

3.3 Uninstalling

[Linux] Type “make clean”
[Linux] Type “rm Makefile”

Documentation of Packages

4.1 Data structures

```
typedef struct {
    Suit suit;
    Num num;
} Card;

typedef struct {
    int totCard;
    Card cards[52];
} Deck;

typedef struct {
    Button token;
    int money;
    Card pocket[2];
    int currentBetAmount;
    int position;
    MoveChoice moveChoice;
    char username[50];
} Player;

typedef struct {
    MoveChoice choice;
    int raiseAmount;
} Move;

typedef struct {
    Player* players;
    Deck deck;
    int totPlayers;
    int totActivePlayers;
    int pot;
    int dealerNum;
    int currentMinBet;
    Card tableCards[5];
} Table;
```

The Table structure above is used to represent the current state of the game at any given time. The server will scan through the Table struct, converting each value into a string which will be sent over to the client where the information will be parsed back into its own local Table struct. The Move struct is also used as a part of the game state as it represents the player's choice on what move to make during their turn.

4.2 Functions and parameters

```
//Contains entirety of game logic, returns a int representing winning
player
int ModifiedServerMainLoop_V3(int ServSocketFD, ClientHandler_V2
HandleClient, TimeoutHandler HandleTimeout, int Timeout, Table table);

//handles entire betting round by iterating through table until all
players have either bet the same amount, folded, or all-in'ed. Both sends
and requests player info using HandleClient and shutdownLoop
Table BettingRoundV2(Table table, int startingPlayer, MoveGetter
shutdownLoop, int ServSocketFD, int Timeout,fd_set* ActiveFDs,
ClientHandler_V2 HandleClient, TimeoutHandler HandleTimeout);

//used by BettingRoundV2 to send current gamestate to all clients whose
turn its not currently is, and a input request for the client whose turn
it is
Move modifiedShutdownLoop(int ServSocketFD, int Timeout, fd_set*
ActiveFDs, Table table, int playerNum,
ClientHandler_V2 HandleClient, TimeoutHandler HandleTimeout);

//is used by modifiedShutdownLoop to send and receive data to/from all
clients
Move ProcessRequest_V2(int DataSocketFD, int playerNum, int
*quasi_shutdown, Table table);
```

```
//helper function that can find the word after the nth occurrence a given
keyword within a string
char* findWord_V2(const char *text, const char *keyword, int n, char
*output);

//takes in a Table struct and parses through it to return a large string
containing all of the Table's information (see example in software spec in
4.3)
char *SendGameState(Table table, int PlayerNum, char *SendBuf);

//parses through a string to initialize all values of a Table struct
Table SetTable(Table table, int playerNum, char* string, int endGame);

//sets big and small blind bets
Table SetBigAndSmallBlind(Table table, int smallblind, int bigblind);

//returns a position of a player who is on the table who has money
int GetNextPlayerWithMoney(Table table, int playersAhead,int startingPos);

//returns position of player with best hand
int BestHand(Table table);
```

4.3 Communication Protocol

The clients will constantly send messages to the server in form “REQUEST_MESSAGE 3” where 3 is replaced with whatever player number they are. If it is not the client’s turn, the server will send back what is known as a Gamestate response. Below is an example of such a message for a game of 6 players, labeled 0 to 5. The message first states the table properties, then each player’s properties. The match is currently in the preflop betting stage, and so all table cards are set to 16-5 to denote an unknown card. SetTable() takes in a player number for one of its arguments, and so based on this value, the local table for each client will initialize the pocket cards for other players to be unknown even though the text below shows how the server will technically give away the other player’s cards.

GAMESTATE(1) TABLE POT: 12 MINIMUM_BET: 8 DEALER_NUM: 0 TOTAL_ACTIVE_PLAYERS: 6 TOTAL_PLAYERS: 6 TABLE_CARD_1: 16-5 TABLE_CARD_2: 16-5 TABLE_CARD_3: 16-5 TABLE_CARD_4: 16-5 TABLE_CARD_5: 16-5 PLAYER_NUM: 0 USERNAME: POCKET_1: 4-0 POCKET_2: 10-0 MONEY: 50 BET_AMOUNT: 0 MOVE_CHOICE: 0 TOKEN: 0 PLAYER_NUM: 1 USERNAME: POCKET_1: 2-1 POCKET_2: 11-1 MONEY: 46 BET_AMOUNT: 4 MOVE_CHOICE: 5 TOKEN: 1 PLAYER_NUM: 2 USERNAME: POCKET_1: 8-3 POCKET_2: 14-3 MONEY: 42 BET_AMOUNT: 8 MOVE_CHOICE: 5 TOKEN: 2 PLAYER_NUM: 3 USERNAME: POCKET_1: 8-2 POCKET_2: 8-0 MONEY: 50 BET_AMOUNT: 0 MOVE_CHOICE: 0 TOKEN: 3 PLAYER_NUM: 4 USERNAME: POCKET_1: 5-2 POCKET_2: 13-2 MONEY: 50 BET_AMOUNT: 0 MOVE_CHOICE: 0 TOKEN: 3 PLAYER_NUM: 5 USERNAME: POCKET_1: 12-0 POCKET_2: 3-3 MONEY: 50 BET_AMOUNT: 0 MOVE_CHOICE: 0 TOKEN: 3

If it is the client’s turn, the server will send back a message that takes the form of “USER_INPUT_REQUEST”. This tells the client to get user input from the GUI. The client will that send back a message that looks something like “PLAYER_NUM 3 CHOICE 3 RAISE_AMOUNT 23” where 3 in this case is the player number, choice denotes a call, raise, fold, check, and raise amount is the amount raised (is ignored if choice is not to raise).

Development Plan and Timeline

5.1 Partitioning of tasks

Week	**Goals**
1	User Manual
2	Software Specifications Begin initial development of data structures - Specify - Some - Structures
3	Start testing client-server architecture Start the testing of the game Alpha Goldfish Software Release
4	Implement extra functions (?) Continue the testing of the game Modify changes Beta Goldfish Software Release
5	Modify any last-minute changes Final Goldfish Software Release

5.2 Team member responsibilities

Team member	**Responsibility**
Jeffrey	Server GUI, Client GUI
Zachary	Game Logic, Server/Client Communication
Kunal	Game Logic, Server/Client Communication
Tommy	Server GUI, Client GUI
Chenhao	Server/Client Communication

Back Matter

Copyright

References

1. https://developer-old.gnome.org/gtk2/stable/index.html (For GTK library)
2. https://www.gtk.org/ (For GTK library)
3. https://www.thisiscolossal.com/2019/06/tsubaki-goldfish/ (Reference Goldfish Img)
4. https://www.nationalgeographic.com/animals/fish/facts/goldfish (Reference Goldfish Img)
5. https://www.aqueon.com/resources/care-guides/goldfish (Reference Goldfish Img)
6. https://www.deviantart.com/cutiepasta-rose/art/Among-Us-Coy-Fish-856305887 (Reference Goldfish Img)
7. https://transparencyhoe.tumblr.com/post/619635121411244032/pixel-goldfish-goldfish-symbolizes-luck-2020 (Reference Goldfish Img)

Index

1. GUI: Graphical User Interface

Def. An interface that uses icons/menus, and a mouse to manage interaction with the system

2. Client

   Def. Hardware/Software that request access to a service hosted by a server

3. Server

   Def. Network, computer program, or device the processes requests from a client connecting to the same port

4. Port

   Def. A number designated to a single process on a given IP address

5. Process

   Def. An application running on a computer

6. IP address

   Def. A string of characters representing a device over the internet

7. Client Server Architecture

   Def. Software applications communicate via the Internet

8. Protocol

   Def. A set of rules for transmitting data between devices

9. Socket

   Def. An endpoint of a two way communication link between two programs

   a. bind(): associate the socket to an IP address
   b. listen(): the readiness to accept client connection request
   c. connect(): establish a connection between client and server
   d. accept(): accept the connection request from the client
   e. write(): write data to the server
   f. read(): reads the data from the client
   g. close(): shuts down the socket
   h. exit(): terminate connection with the client