CSCI 3060 - Final Project
Phase 2
Frontend Design Document

Descriptions for all methods and classes:

Class	**Description**
Application	The Application class is used to start off each session when the program is run and is used to check if the user is currently logged in.
Members	

User user;
bool running = false;
	All the member attributes of the App class.

Methods	**Description**
Application(){};	The constructor method for the Application class.
~Application(){};	The destructor method for the Application class.
void start();	This method starts off the program by printing the welcome message, and calling the required methods to display the appropriate menu and getting the transaction code from the user for the transaction they wish to do.
void displayMenu();	When called in start(), this method gets the user type from the user that is currently logged in and displays the appropriate menu based on user type.
void handleInput(std::string in);	This method gets the user's input for the transaction they want to do, and validates that the user's input is one of the available transaction options. If the user's input is incorrect (i.e. transaction code is wrong), then the appropriate error message is printed.

Class	**Description**
AvailableItemsFile	The AvailableItemsFile class is used to read, append, update, print the contents of an available items file and is used to verify the

	contents of an available items file.
Members	
std::fstream file	The attribute for the AvailableItemsFile class.
Methods	**Description**
AvailableItemsFile(/* args */);	The constructor method for an AvailableItemsFile object.
~AvailableItemsFile();	The destructor method for an AvailableItemsFile object.
bool appendItemToFile(ItemInfo data);	This method appends the information for the specified item to the available items file, and returns true if successfully appended to the file.
bool itemExistsInFile(ItemInfo data);	This method verifies if the specified item's information currently exists in the available items file and returns true if the item's information is found in the file.
std::vector<ItemInfo> readItemFile();	This method reads in all the contents of the available items file, and returns a vector with all the information. The information for each item is stored as an ItemInfo struct in the vector.
bool deleteItemFromFile(ItemInfo data);	This method deletes the information for the specified item from the available items file and returns true if the information is successfully deleted.

Class	**Description**
Item	The item class is used to manage and update the attributes for each item object, and is used to keep track of whether it has been sold or not.
Members	

char *itemName;
char *sellerName;
int daysRemaining;
double highestBid;
bool sold;
	All the member attributes of the Item class.

Methods	**Description**
Item(/* args */);	The constructor method for an item object.
~Item();	The destructor method for an item object.
void updateBid(double newBidamount);	This method updates the current highest bid for an item using the passed in new bid amount.
void sellItem();	This method ensures that an item may not have any further bids placed and that it will be removed from the Available Items List. It also ensures that the appropriate fund amounts are transferred from the buyer's account to the seller's account.
void updateDaysRemaining(int newNumOfdays);	This Method will update the amount of days remaining to bid for the item.

Class	**Description**
Transaction	The transaction class is used to manage and carry out all the transactions that can be done by a user.
Methods	**Description**
static bool login(User *user);	This method logs a user into the system, based on the passed in user object.
static bool logout(User *user);	This method logs the current user out of the system.
static bool createUser(User *user);	This method allows an admin user to create a new user.
static bool deleteUser(User *user);	This method allows an admin user to delete a specified user.
static bool advertise(User *user);	This method allows all passed in users except of type BS to advertise/put an item up for auction.
static bool bid(User *user);	This method allows all passed in users except of type SS, to place a bid on an item.

static bool refund(User *user);	This method allows an admin user to refund the credit amount to the specified user.
static bool addcredit(User *user);	This method allows users to add a credit amount to a specified user's account.
static bool listitems(User *user);	This method allows users to list all items up for auction in the console.
static bool outputAllActiveAccounts(User *user);	This method allows admin users to output to console all active accounts.
static std::map<std::string, std::function<bool(User *)>> getAdminMapping();	This method is used to return the appropriate menu options available for an admin user.
static std::map<std::string, std::function<bool(User *)>> getFullStandardMapping();	This method is used to return the appropriate menu options available for a full standard user.
static std::map<std::string, std::function<bool(User *)>> getSellStandardMapping();	This method is used to return the appropriate menu options available for a sell standard user.
static std::map<std::string, std::function<bool(User *)>> getBuyStandardMapping();	This method is used to return the appropriate menu options available for a buy standard user.
static std::map<std::string, std::function<bool(User *)>> getLoginMapping();	This method is used to return the appropriate menu option that is available to the user at the start of the program(only login is available).

Class	**Description**
TransactionFile	The TransactionFile class is used to read, append, update, print the contents of a transaction file and is used to verify the contents of a transaction file.
Members	
std::fstream file;	The attribute for the TransactionFile class.
Methods	**Description**
TransactionFile(/* args */);	The constructor method for a TransactionFile object.

~TransactionFile();	The destructor method for a TransactionFile object.
bool appendToFile(TransactionInfo data);	This method appends the information for the specified transaction to the transaction file, and returns true if successfully appended to the file.
bool transactionExistsInFile(TransactionInfo data);	This method verifies if the specified transaction's information currently exists in the transactions file and returns true if the transaction's information is found in the file.
std::vector<TransactionInfo> readTransactionFile();	This method reads in all the contents of the transactions file, and returns a vector with all the information.
bool deleteLineFromFile(TransactionInfo data);	This method deletes the information for the specified transaction from the transactions file and returns true if the information is successfully deleted.

Class	**Description**
User	The User class is used to construct each user object, and manage its current status to determine whether or not it is currently logged in.
Members	

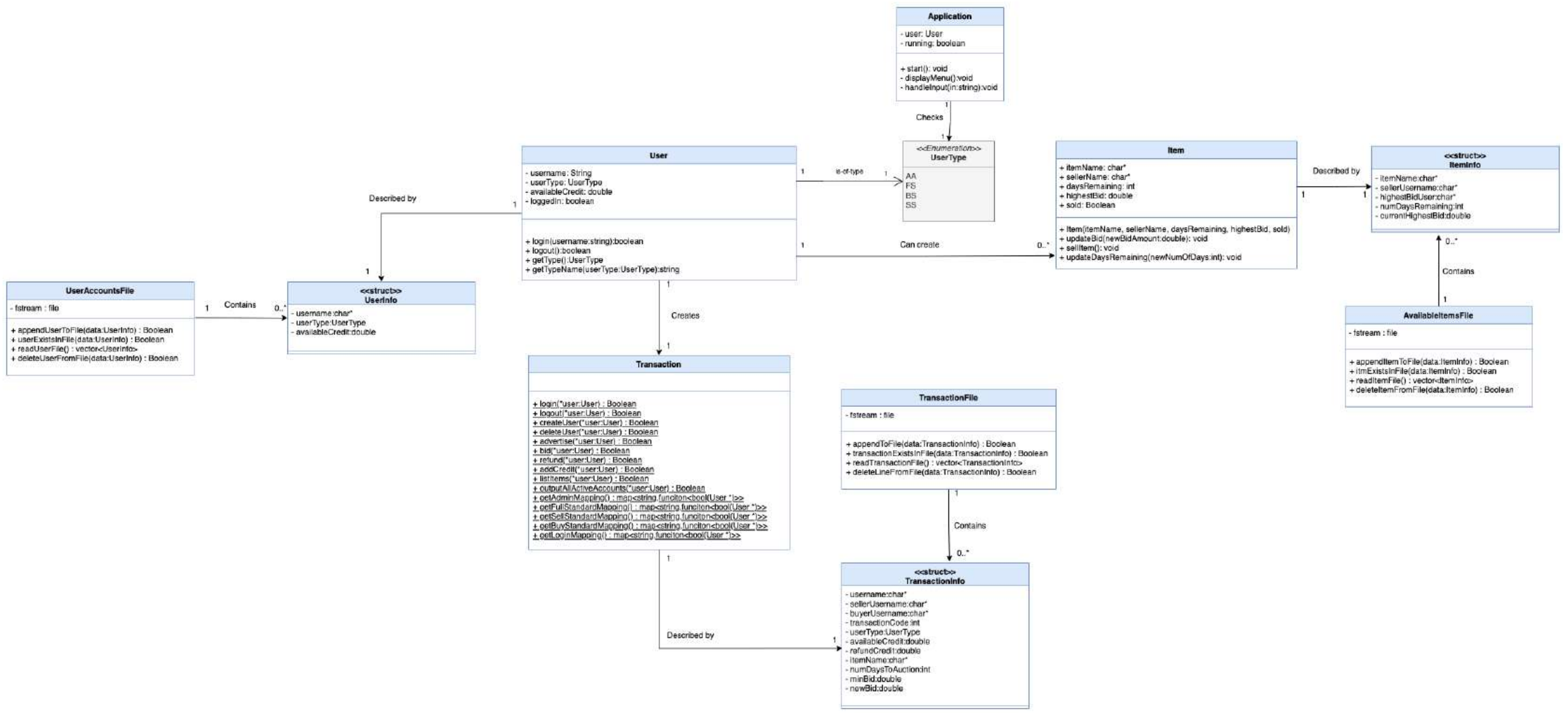
UserType userType;
std::string username;
double availableCredit;
bool loggedIn;
	All the member attributes of the User class.

Methods	**Description**
User();	The constructor method for a user object.
~User();	The destructor method for a user object.
bool login(std::string username);	Based on the passed in username, this method sets the user type for each user to log them in and to print the appropriate MENU.
bool logout();	This method sets the usertype to NONE, and

	the boolean value of “loggedin” to “false” when a user decides to log out.
UserType getType() { return userType; }	This method returns the user type of the current user.
static std::string getTypeName(UserType userType);	This method returns the full user type name, based on the passed in user type for the current user.

Class	**Description**
UserAccountsFile	The UserAccountsFile class is used to read, append, update, print the contents of a UserAccountsFile and is used to verify the contents of a UserAccounts file.
Members	
std::fstream file;	The attribute for the UserAccountsFile class.
Methods	**Description**
UserAccountsFile(/* args */);	The constructor method for a UserAccountsFile object.
~UserAccountsFile();	The destructor method for a UserAccountsFile object.
bool appendUserToFile(UserInfo data);	This method appends the information for the specified user to the user accounts file, and returns true if successfully appended to the file.
bool userExistsInFile(UserInfo data);	This method verifies if the specified user’s information currently exists in the user accounts file and returns true if the user’s information is found in the file.
std::vector<UserInfo> readUserFile();	This method reads in all the contents of the user accounts file, and returns a vector with all the information.
bool deleteUserFromFile(UserInfo data);	This method deletes the information for the specified user from the user accounts file and returns true if the information is successfully deleted.

CSCI 3060 - Phase 2


Application
- user: User
- running: boolean
+ start(): void
- displayMenu():void
- handleInput(in:string):void
Checks
1
1
<<Enumeration>>
UserType
AA
FS
BS
SS
User
- username: String
- userType: UserType
- availableCredit: double
- loggedIn: boolean
+ login(username:string):boolean
+ logout():boolean
+ getType():UserType
+ getTypeName(userType:UserType):string
1
is-of-type
1
1
Can create
0..*
Item
+ itemName: char*
+ sellerName: char*
+ daysRemaining: int
+ highestBid: double
+ sold: Boolean
+ Item(itemName, sellerName, daysRemaining, highestBid, sold)
+ updateBid(newBidAmount:double): void
+ sellItem(): void
+ updateDaysRemaining(newNumOfDays:int): void
Described by
1
1
<<struct>>
ItemInfo
- itemName:char*
- sellerUsername:char*
- highestBidUser:char*
- numDaysRemaining:int
- currentHighestBid:double
0..*
Contains
1
AvailableItemsFile
- fstream : file
+ appendItemToFile(data:ItemInfo) : Boolean
+ itmExistsInFile(data:ItemInfo) : Boolean
+ readItemFile() : vector<ItemInfo>
+ deleteItemFromFile(data:ItemInfo) : Boolean
Described by
1
1
UserAccountsFile
- fstream : file
+ appendUserToFile(data:UserInfo) : Boolean
+ userExistsInFile(data:UserInfo) : Boolean
+ readUserFile() : vector<UserInfo>
+ deleteUserFromFile(data:UserInfo) : Boolean
1
Contains
0..*
<<struct>>
UserInfo
- username:char*
- userType:UserType
- availableCredit:double
1
Creates
1
Transaction
+ login(*user:User) : Boolean
+ logout(*user:User) : Boolean
+ createUser(*user:User) : Boolean
+ deleteUser(*user:User) : Boolean
+ advertise(*user:User) : Boolean
+ bid(*user:User) : Boolean
+ refund(*user:User) : Boolean
+ addCredit(*user:User) : Boolean
+ listItems(*user:User) : Boolean
+ outputAllActiveAccounts(*user:User) : Boolean
+ getAdminMapping() : map<string,funciton<bool(User *)>>
+ getFullStandardMapping() : map<string,funciton<bool(User *)>>
+ getSellStandardMapping() : map<string,funciton<bool(User *)>>
+ getBuyStandardMapping() : map<string,funciton<bool(User *)>>
+ getLoginMapping() : map<string,funciton<bool(User *)>>
1
Described by
1
TransactionFile
- fstream : file
+ appendToFile(data:TransactionInfo) : Boolean
+ transactionExistsInFile(data:TransactionInfo) : Boolean
+ readTransactionFile() : vector<TransactionInfo>
+ deleteLineFromFile(data:TransactionInfo) : Boolean
1
Contains
0..*
<<struct>>
TransactionInfo
- username:char*
- sellerUsername:char*
- buyerUsername:char*
- transactionCode:int
- userType:UserType
- availableCredit:double
- refundCredit:double
- itemName:char*
- numDaysToAuction:int
- minBid:double
- newBid:double