Project 1 Documentation
Blackjack

Course: CSC 17C
Section Code: 47469
Name: Rohan Vasireddy
Instructor: Mark Lehr
Date: 11/30/2025

Repository Link: https://github.com/rohanvasireddy/CSC17C.git

Table of Contents

1. Introduction

This project implements the well-known card game Blackjack as a C++ console program. The main goal is to demonstrate strong STL usage—containers, container adaptors, iterators, and algorithms—while explicitly avoiding std::vector. The program provides both interactive play and a simulation mode that runs many rounds automatically and reports statistics.

2. Game Play and Rules

The game follows standard Blackjack rules. Cards 2–10 count as face value, J/Q/K count as 10, and Ace counts as 11 unless it would cause a bust, in which case it counts as 1.

- Player and dealer are each dealt 2 cards.
- Player chooses Hit (take another card) or Stand (end turn).
- Dealer hits until total is at least 17.
- Bust (>21) loses immediately.
- If neither busts, higher total wins; equal totals is a push (tie).

3. Development Summary

Lines of Code (source)	_____
Comment Lines	_____
Blank Lines (whitespace)	_____
Total Lines (file)	_____
Number of Classes	Deck, Hand, Stats, BlackjackGame (+ Card struct, helpers)
Time Spent (approx)	_____ hours

The project was developed incrementally: implement card/deck/hand logic first, then the interactive game loop, then simulation statistics, then additional STL algorithm and iterator demonstrations.

4. Specifications

4.1 Sample Inputs/Outputs

Interactive mode (example):

=== BLACKJACK ===
1) Play
2) Simulate
3) Quit
> 1

Player: 8♥ 6♠ (Total 14)

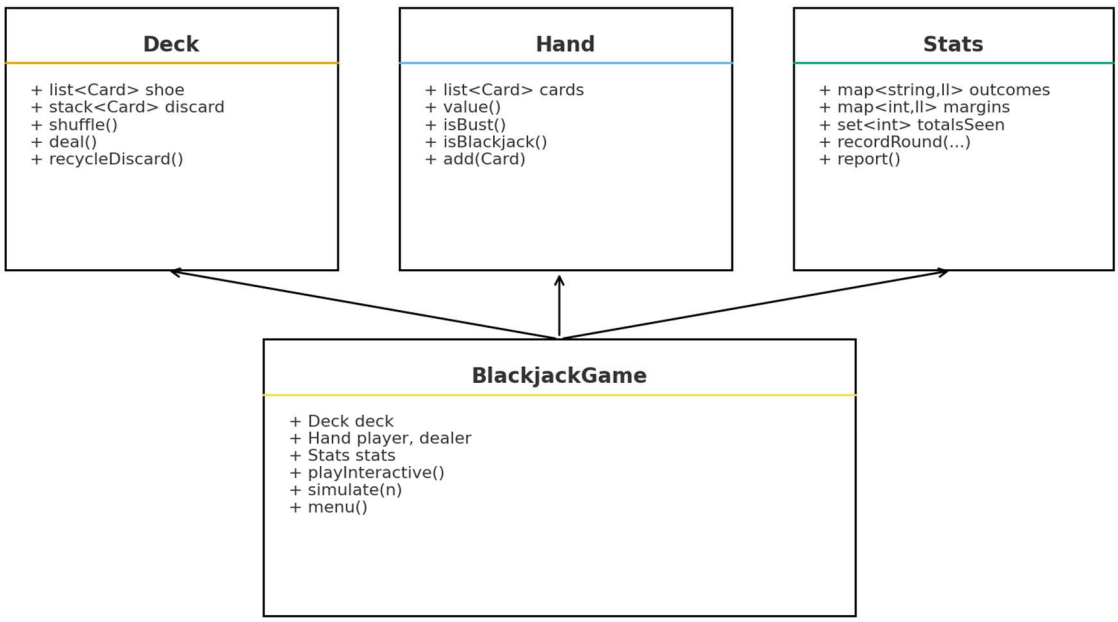
Dealer: [Hidden] K♣
Hit or Stand (H/S): H
Player: 8♥ 6♠ 6♦ (Total 20)
Dealer reveals: 8♣ K♣ (Total 18)
Outcome: WIN

Simulation mode (example):

> 2
Rounds: 10000
WIN: 4088
LOSS: 5001
PUSH: 911
Top margins: +2, -2, +1, -1 ...

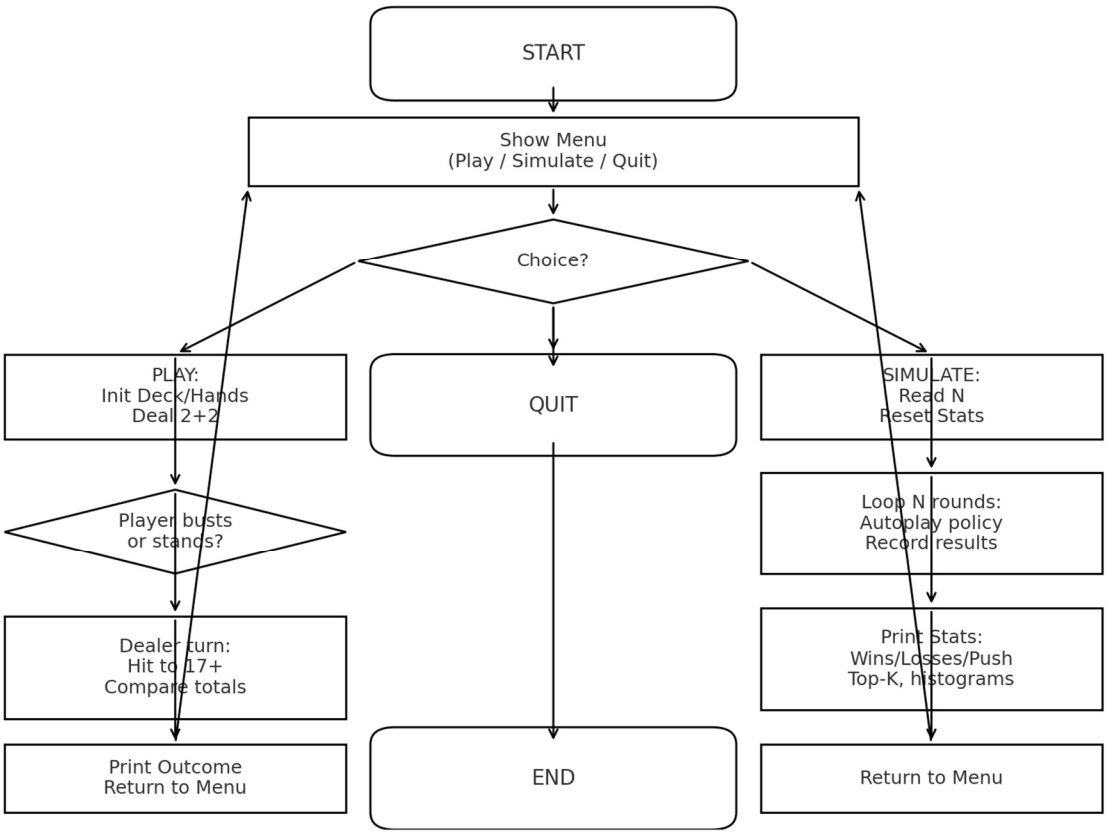
4.2 Class Diagram

High-level class relationships used by the program:



4.3 Activity / Flowchart

Overall program flow and major decision points:



4.4 Pseudocode

Main Menu:

```
loop:
  display menu
  read choice
  if choice == PLAY: playInteractiveRound()
  if choice == SIMULATE: simulate(N)
  if choice == QUIT: break
```

Play Round:

```
initialize deck and hands
deal 2 cards to player and dealer
while player total < 21 and player chooses HIT:
  deal 1 card to player
dealer reveals hidden card
while dealer total < 17:
  deal 1 card to dealer
```

compare totals and print WIN/LOSS/PUSH
update statistics (optional)

Simulate N rounds:

for i in 1..N:
  reset hands
  deal opening cards
  autoplay policy for player (e.g., hit to 16)
  dealer hits to 17
  compute outcome and margin
  stats.record(outcome, margin, totals)
stats.report()

4.5 Major Variables / Data Structures

Key data members and where they are used:

Type	Name	Description	Location
struct	Card {rank, suit}	Represents a single playing card	Card struct / helpers
std::list<Card>	Deck::shoe	Current deck/shoe order for dealing	class Deck
std::stack<Card>	Deck::discard	Discard pile (LIFO)	class Deck
std::list<Card>	Hand::cards	Cards held by player/dealer	class Hand
int	Hand::value()	Computes Blackjack total with Ace adjustment	class Hand
std::unordered_map<std::string,int>	rankValue	Rank -> value lookup (A,J,Q,K,2..10)	helpers / Hand
std::map<std::string,long long>	Stats::outcomes	Histogram for WIN/LOSS/PUSH	class Stats
std::map<int,long long>	Stats::margins	Histogram for (playerTotal - dealerTotal)	class Stats
std::set<int>	Stats::totalsSeen	Unique totals encountered during simulation	class Stats
std::queue<std::string>	dealOrder	Demonstrates FIFO sequencing	BlackjackGame
std::priority_queue<...>	topK	Top-K most common	Stats report

		margins/outcomes	
std::bitset<8>	options	Feature toggles (debug, verbose, etc.)	BlackjackGame
std::set<std::string>	Deck::seenIds	Uniqueness / membership demo for dealt cards	class Deck
std::deque<int>	scratch	Random-access container for sort/binary_search demos	Algorithm/Stats demo

4.6 STL Concepts Used (Containers / Iterators / Algorithms)

This section is written in the same spirit as the example documentation tables, but adapted to C++ STL.

Category	Concept / Construct	Where Used
Containers (sequence)	std::list, std::forward_list	Deck::shoe, Hand::cards; tips/help text list
Associative	std::map, std::set, std::unordered_map	Stats histograms, uniqueness checks, rank lookup
Adaptors	std::stack, std::queue, std::priority_queue	discard, deal order, top-K summaries
Iterators	forward, bidirectional, random-access	forward_list/list/set/map/deque iteration
Algorithms (non-mutating)	for_each, find_if, count_if, equal, search	analysis and demo utilities
Algorithms (mutating)	copy, transform, replace, remove_if, fill	data cleanup, formatting, working buffers
Algorithms (organizing)	sort, binary_search, merge, inplace_merge, minmax_element	stats and reporting

5. Build & Run Instructions

Visual Studio 2022 (Developer Prompt):

```
cd C:\path\to\project
cl /std:c++17 Blackjack.cpp
Blackjack.exe
```

G++ (if installed):

```
g++ -std=c++17 Blackjack.cpp -O2 -o blackjack
./blackjack
```

6. References

Blackjack rules reference (general): standard casino Blackjack rules (online references).

C++ STL references: cppreference.com (containers, iterators, algorithms).

Instructor-provided project specification and example documentation formats.

Appendix: Project 1 Checkoff Sheet (STL Requirements)

Use this page during demonstration/grading. Fill in the 'Where' column with your exact function names/line ranges after your final code is posted.

A. STL Containers / Adaptors

Category	STL Type	Purpose	Where in Program	Checked
Sequence	std::list<Card>	Deck shoe order + Hand storage	Deck::shoe, Hand::cards	
Sequence	std::forward_list<string>	slist equivalent (tips/help)	buildTips() / tips iteration	
Bit container	std::bitset<N>	option flags/toggles	options flags / menu toggles	
Associative	std::map<string, ll>	Outcome histogram (ordered)	Stats::outcomes	
Associative	std::map<int, ll>	Margin histogram	Stats::margins	
Associative	std::set<string> / std::set<int>	Uniqueness + membership checks	Deck::seenIds, Stats::totalsSeen	
Associative	std::unordered_map<string,int>	Rank->value lookup (hash)	VALUE_HASH / Hand::value()	
Adaptor	std::stack<Card>	Discard pile (LIFO)	Deck::discard	
Adaptor	std::queue<string>	Dealing order (FIFO)	dealOrder queue	
Adaptor	std::priority_queue<...>	Top-K outcomes/margins	Stats top-K report	

B. Iterator Types Demonstrated

Iterator Category	Iterator / Construct	Purpose	Where in Program	Checked
Trivial (C-style)	array iteration	Build 52-card deck from ranks/suits arrays	Deck::rebuild()	
Input	istream_iterator<int>	Iterator-based input demo	demoInputIterators()	
Output (optional)	ostream_iterator<T>	Iterator-based output demo (if used)	print sequence demo	
Forward	forward_list<string>::iterator	Traverse tips/help list	tips iteration	
Bidirectional	list/set/map iterators	Traverse game state + histograms	Hand/Deck/Stats loops	
Random	deque iterators	sort/binary_search/minmax	Stats algorithm	

Access		/merge demos	demo	

C. STL Algorithms Used (by category)

Category	Algorithm	Purpose	Where in Program	Checked
Non-mutating	for_each	Process/print sequences	printing helpers / demos	
Non-mutating	count_if	Count Aces or conditions	Hand::value()	
Non-mutating	find_if / search / equal	Search/equality demos	AlgorithmLab demos	
Mutating	copy	Copy into working buffers	Stats demo	
Mutating	fill	Initialize buffers	Stats demo	
Mutating	transform	Convert strings / map values	log formatting demo	
Mutating	replace	Token replacement	log formatting demo	
Mutating	remove_if + erase	Filter a container	log filtering demo	
Organization	sort	Sort results	Stats demo (deque)	
Organization	binary_search	Search sorted data	Stats demo	
Organization	merge / inplace_merge	Merge sorted ranges	Stats demo	
Organization	minmax_element	Min/max extraction	Stats demo	

Instructor notes / initials: _____