SOFTWARE PROJECT FINAL REPORT

LiLou DeRivera, Keaton Dinger, Hitesh Kukreja, Nathan Sikon –2823726

Online Casino/Blackjack Group 11

04/20/2025

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1. Purpose and Scope

- Purpose: The purpose of the project is to replicate a digital version of blackjack for purely recreational purposes where no real money is involved. This simulated version of blackjack allows users to play against a 'computer' dealer.

- Scope: As this is a replicated game of blackjack, all traditional rules apply, including the dealing of cards, hitting, standing, checking for busts and wins. The software will include a simple UI that displays the user their options. The user will be playing against a digital dealer to earn more "chips", a non-real currency for the game.

1.2. Product Overview (including capabilities, scenarios for using the product, etc.)

- The software replicates an online blackjack game, only the user plays against the dealer and no real currency is used. The user begins with a pre-determined number of chips. Once the game starts, the user places their bet, and the user and dealer are dealt their cards. After, the user can perform actions such as 'hit', 'stand', 'double down', and 'split'. After making their choice, the dealer gets their cards. Then, the game decides who won and lost.

This target for this software is for those that want a casual blackjack experience without the pressure of gambling with real-world money. It can also teach players the general rules of blackjack for if they would like to play the game in casinos or with friends.

1.3. Structure of the Document

- The document is structured in 7 parts:

1) Introduction – introduces the project with its purpose, scope, and product overview

2) Project Management Plan – Outline pre-requisites for the project. It includes organization of the project, the project's development lifecycle, the identification of any risks associated with the software, and hardware + software resource requirements.

3) Requirement Specifications – This section identifies the stakeholders, use cases (accompanied by a use-case model) and rationale for them, and non-functional requirements

4) Architecture – Identifies the architectural styles, architectural model and rationale for said model, and technology, software, and hardware used.

5) Design – Includes the design of the interface, components design + rationale, database design, and traceability

6) Test Management – Include a list of testcases, traceability of testcases, techniques used for testcase generation, test results and assessments, and a report of defects.

7) Conclusions – Outcomes of the project, what was learned, and what directions can be taken for future development

1.4. Terms, Acronyms, and Abbreviations

- 'bust', when a user's combined numerical score is greater than 21, they lose that game.

- 'blackjack', when a user's initial two cards combine numerically to equal 21, they get a blackjack. When this happens, their bet is rewarded 3/2.

- 'hit', a blackjack term where the user wishes to receive an additional card. User can 'hit' as many times as they wish, until they bust.

- 'stay', a blackjack term where a user forfeits receiving additional cards.

- 'double down', a blackjack term where the user wishes to receive ONLY ONE additional card, while also doubling their bet.

- 'split', a blackjack term where the user wishes to play two separate hands if the user's initial cards are matching numerically and face-type (i.e. jack, queen, king).

2. Project Management Plan

2.1. Project Organization

- Our group is made up of four people (LiLou, Keaton, Hitesh, and Nathan). We all have shared roles on our team, meaning that everyone helps each other out in equal parts in regard to developing, testing, and UI design.

2.2. Lifecycle Model Used

- The lifecycle model that we decided to use as a group was the "Waterfall Model". Because the model is good for project with well-defined, unchanging requirements, it was a great choice for software where we replicate an already-existing game of blackjack.

2.3. Risk Analysis

- Risk Table

RISK NAME	PROBABILITY (Low, Medium, High)	IMPACT (Low, Medium, High)	RM3 POINTER
Browser compatibility issues	Medium	High	Related to UI testing and cross-browser support
Performance issues	Medium	High	Linked to optimization techniques in rendering
Project Delays	High	Medium	Related to time management and milestone tracking
Scope Creep	Medium	High	Connected to project scope management

2.4. Hardware and Software Resource Requirements

- The software is web-based and designed for computer use, not smart phones. For this reason, a computer (Windows, Mac, or Linux) is needed that has access to a web browser.

2.5. Deliverables and schedule

Phase 1: Planning
. Define project scope, objectives, and constraints
. Identify core functionalities
. Risk assessment
. Development project timeline and milestones

Phase 2: Modeling
. Design architecture
. Create UI wireframes and layout
. Define data structures
. Plan state management

Phase 3: Construction

1. Core game mechanics of Blackjack
2. User interface and interactivity
3. Game logic and refinement

Phase 4: testing and debugging

. Test functionality
. Works across different browsers
. Fix identified problems

Phase 5: Deployment and Feedback

. Test the game out on a web server and gather feedback

3. Requirement Specifications

3.1. Stakeholders for the system

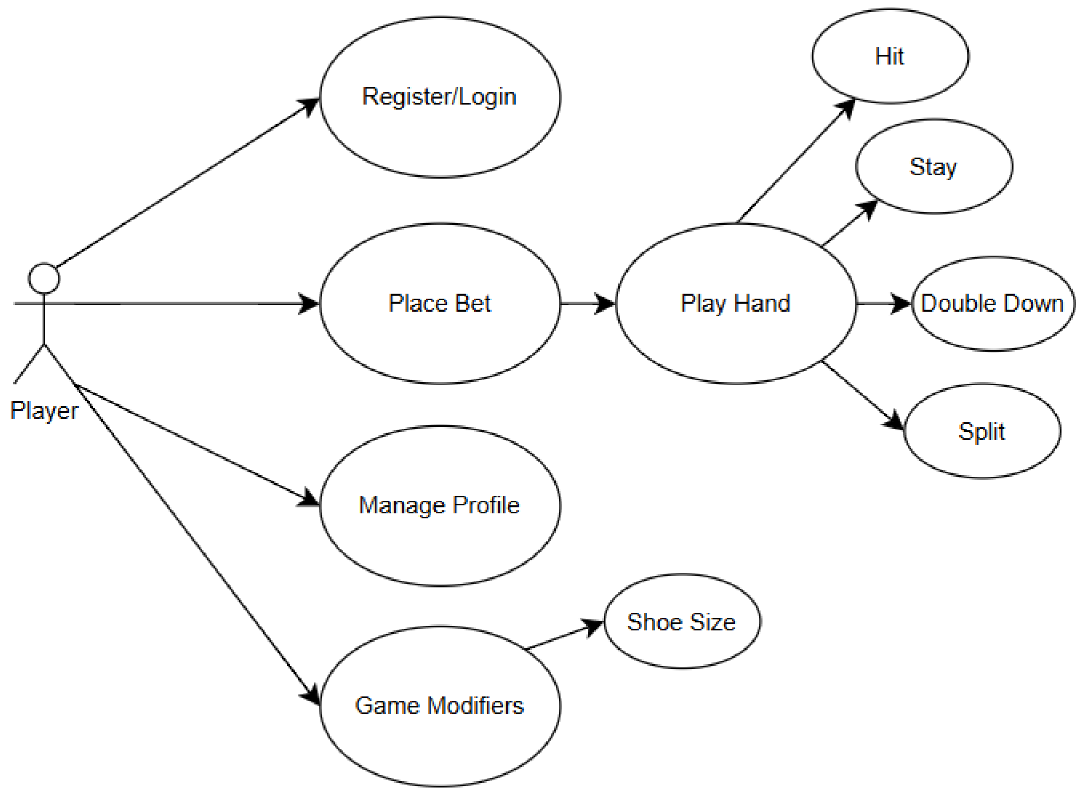
- Project Team: Everyone involved in the development of the software has helped to build and maintain it. Each team member's grade is dependent on the quality and what the team delivers with the software

- End Users: The users that play the game are a big stakeholder as their experience determines the success of the project

- Instructor/grader: Very important in the grading process, as they expect certain deliverables and functionality relating to our specific project.

3.2. Use cases

3.2.1. Graphic use case model



3.2.2. Textual Description for each use case

- Register/Login: Player can create a new account or login to their existing account that holds their chip information

- Place Bet: Player can initiate a game by placing their bet

i) Play Hand: The player is then able to play the game given the available options:

A) Hit

B) Stay

C) Double Down

D) Split

- Mange Profile: The user is able manipulate aspects about their profile, like chip count, is also able to delete their account

- Game Modifiers: The user is able to manipulate some aspects about the game, such as shoe size

3.3. Rationale for your use case model

- The model shows a simple graphic of all the actions that the user has in the software. The reason why an arrow leads from the user to place bet, to play hand, and then the four call options is to show the linear progression of how each function is accessed. For example, you cannot play a hand without first placing a bet.

3.4. Non-functional requirements

- Some non-functional requirements include:

i) Blackjack Simulation Logic (game functionality)

ii) Game outcomes must be provably fair (outcomes must be achieved logically in the bounds of blackjack logic)

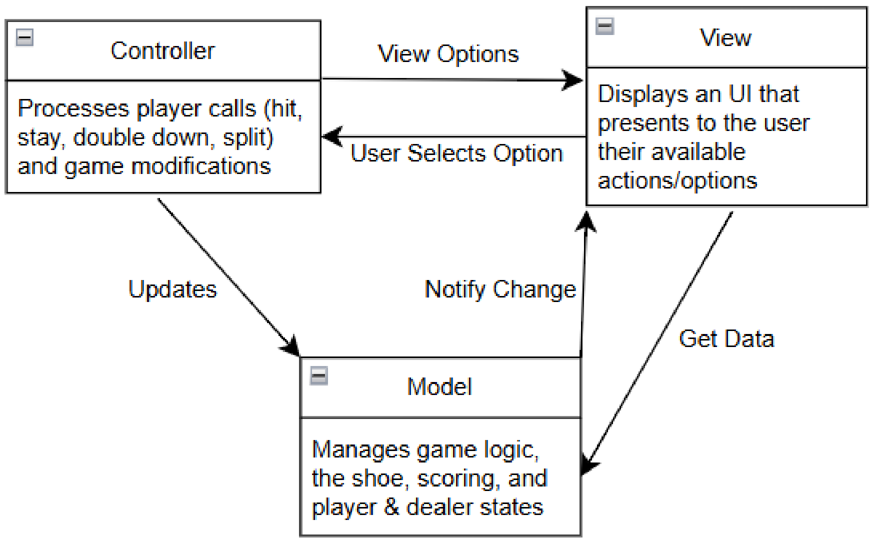
iii) Performance (Response time should be quick without noticeable delay)

4. Architecture

4.1. Architectural style(s) used

- Model-View-Controller

4.2. Architectural model (includes components and their interactions)


Controller
Processes player calls (hit, stay, double down, split) and game modifications
View Options
User Selects Option
View
Displays an UI that presents to the user their available actions/options
Updates
Notify Change
Get Data
Model
Manages game logic, the shoe, scoring, and player & dealer states

-

4.3. Technology, software, and hardware used

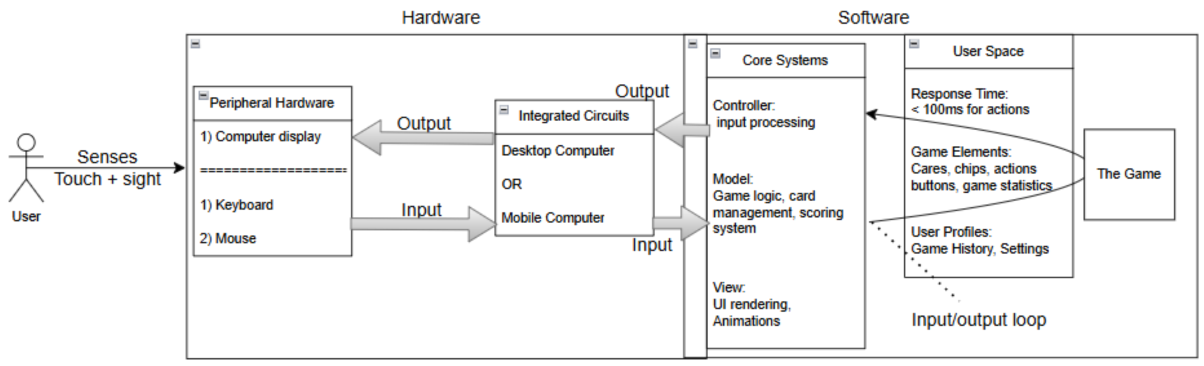
- No external/physical technology was used. What was used to create the software was CSS to format the webpage, and JavaScript for the interactivity and dynamic behavior of the web page, and then HTML to define the structure and content of the webpage.

4.4. Rationale for your architectural style and model

- Model-View-Controller was used because the software requires three logical parts of handling game logic, displaying UI, and having player actions update the model, and so on. In the instance of blackjack, it is especially helpful to have the 'Model' manage game logic and the different states, 'View' to handle UI and other visualizations, and 'Controller' to process player inputs to update the model accordingly. Additionally, the separation of the three parts makes testing and maintaining the software much easier and simplistic.

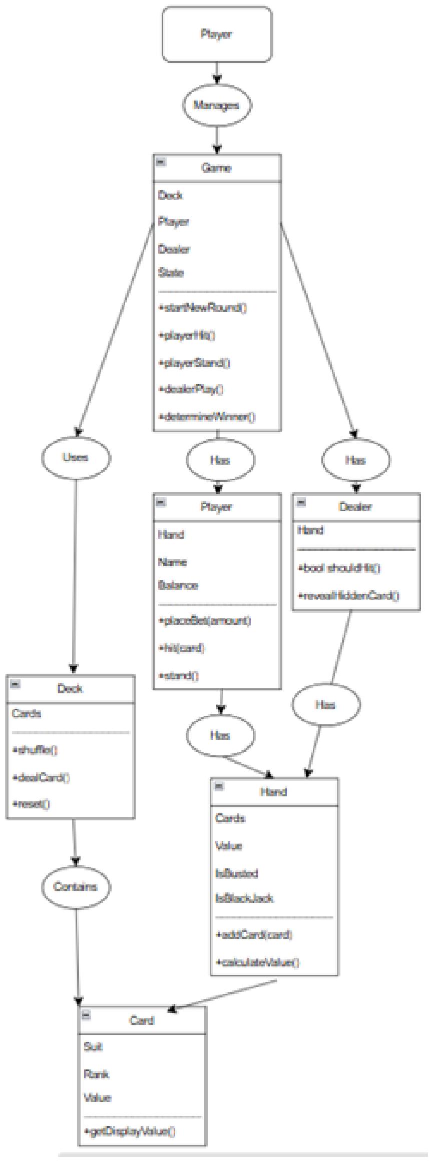
5. Design

5.1. User Interface design

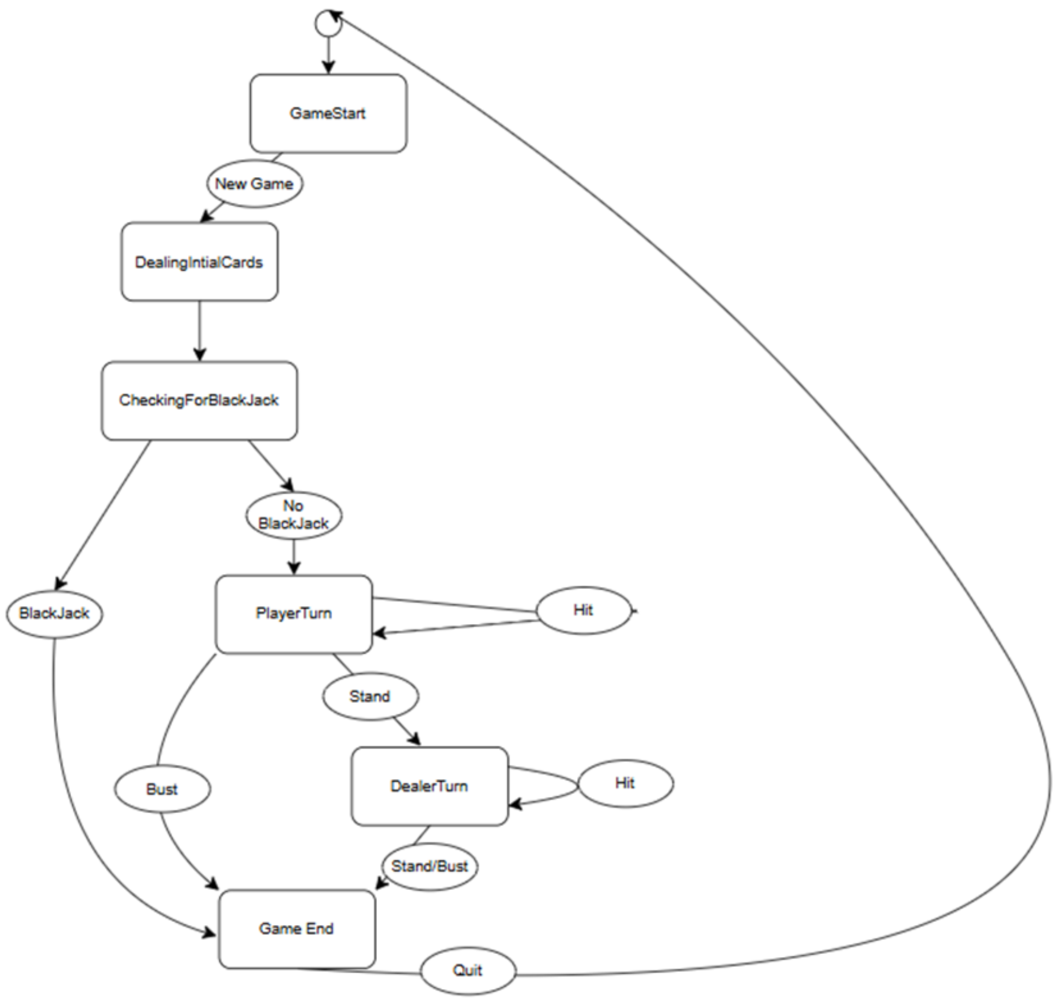

Hardware
Software
User
Senses
Touch + sight
Peripheral Hardware
1) Computer display
1) Keyboard
2) Mouse
Output
Input
Integrated Circuits
Desktop Computer
OR
Mobile Computer
Output
Input
Core Systems
Controller: input processing
Model: Game logic, card management, scoring system
View: UI rendering, Animations
User Space
Response Time: < 100ms for actions
Game Elements: Cares, chips, actions buttons, game statistics
User Profiles: Game History, Settings
The Game
Input/output loop

5.2. Components design (static and dynamic models of each component)

-Static


Player
Manages
Game
Deck
Player
Dealer
State
+startNewRound()
+playerHit()
+playerStand()
+dealerPlay()
+determineWinner()
Uses
Has
Has
Player
Hand
Name
Balance
+placeBet(amount)
+hit(card)
+stand()
Dealer
Hand
+bool shouldHit()
+revealHiddenCard()
Deck
Cards
+shuffle()
+dealCard()
+reset()
Has
Has
Hand
Cards
Value
IsBusted
IsBlackJack
+addCard(card)
+calculateValue()
Contains
Card
Suit
Rank
Value
+getDisplayValue()

-Dynamic



5.3. Database design

- In the database, the software takes login information, username and password, from the user. The user is able to create a new account by using a unique username and a password to go with it. If the user attempts to create an account with an already existing username or incorrectly enters their password, that account cannot be created or accessed. The accounts hold player information, being chip count.

5.4. Rationale for your detailed design models

- The static model shows a linear path of the software's structure during a game of blackjack. The model depicts attributes, the entities that house them, and all of their relationships as they interact.

The dynamic model illustrates accurately shows the execution loop of the software. It shows the beginning and end from when the user first starts a new game, the choices that they're able to make, and if they enter the win/lose state.

5.5. Traceability from requirements to detailed design models

Requirement ID	Component	Design Component	Status	Requirement Type	Test Case ID
REQ-1	User Authentication	User Management 3.2.1	Finished	Functional	TC-101: Test login scenarios
REQ-2	Betting System	Betting Engine 3.2.1	Finished	Functional	TC-201: Verify bet validation
REQ-3	Blackjack Logic	Game Logic 3.2.1	Finished	Non-Functional	TC-301: Verify by playtesting
REQ-4	Chip Management	Virtual Wallet 3.2.2	Finished	Functional	TC-401: Verify Currency Updates
REQ-5	Game outcomes provably fair	Game Logic 2.2	Finished	Non-Functional	TC-501: Verify game RNG

6. Test Management

6.1. A complete list of system test cases

Test Case	Description	Input	Expected Output	Pass/Fail
TC01	Place valid bet	$50	Game starts with 50$ dedcuted	Functional
TC02	Place bet more than bank	2000$	Alert shown, no deduction	Functional
TC03	Invalid bet amount	0$	Alert shown, no deduction	Functional
TC04	Deal cards to player and dealer	Start game	2 cards for player, dealer starts with 1 visible	Functional
TC05	Player hits and stays	Hit once, then stay	Dealer plays, results displayed	Functional

TC06	Player hits and busts	Hit repeatedly	Player score > 21, can’t hit again	Functional
TC07	Player wins	20 vs dealer 18	Win message, bank updated	Functional
TC08	Player loses	16 vs dealer 20	Loss message, bank updated	Functional
TC09	Tie	18 vs 18	Tie message, refund bet	Functional
TC10	Double down without funds	Bet max, try double	Alert, no action	Functional
TC11	Double Down	Double with sufficient bank	One more card, double bet,auto stay	Non-functional
TC12	Split with different cards			Non-functional
TC13	Split with same cards	Two identical cards	Two hands played seperately	Non-functional
TC14	Music plays	Turns on and stays in	Music playing in game	Functional

6.2. Traceability of test cases to use cases

Use Case	Related Test Cases

UC01-Place Bet	TC01, TC02, TC03
UC02- Start Game	TC04
UC03- Player Actions (Hit/Stay)	TC05, TC06
UC04- Evaluate Results	TC07, TC08, TC09
UC05- Double Down	TC10, TC11
UC06 -Split	TC12, TC13
UC07- Settings (Music)	TC14

6.3. Techniques used for test case generation

- Equiavalence Partitioning: Tested valid vs invalid bet amounts, edge cases (eg. 0$, max value)
- Boundary Value Analysis: Bet at $10 (min), $1000 (max), bank limits
- State Transition Testing: Checked state changes between game start, hit, stay, and new round
- Exploratory Testing: Manual tests around double down and split
- Use Case Based Testing: Ensured each user interaction was covered with related test cases

6.4. Test results and assessments (how good are your test cases? How good is your software?)

Total Test Cases: 14

Passed: 11

Failed: 3

Assessment:

Strong – All primary cases and UI elements tested.

Test Effectiveness: High – Issues such as money logic, edge behaviors, and invalid input handling were accounted for

Software Quality:

Stability: High – Most core features perform reliably

Usability: User friendly with intuive controls and feedback besides music which is not too tedious

Maintainability: Good – code structure is modular and testable

6.5. Defects reports

In some cases, the cards load slower than when the outline appears. This could be a defect due to the load times or just processing power of one's computer.

7. Conclusions

7.1. Outcomes of the project (are all goals achieved?)

All outcomes have been achieved. Shoe size was reinterpreted as a settings bar that could manipulate some items on the website. This includes things like music.

7.2. Lessons learned

Developing a website that works smoothly and works well is difficult. Starting with an earlier design plan and sticking with it is a great tool to have when developing an application though. Having all code mapped out and each step you took so you can find any error was useful after halfway into development. We kept track of each step we needed to take and succeeded in making a working product. Bit more than what we could chew with adding features and double down is being extremely buggy so we removed it for now.

7.3. Future development

Expanding our product to have a greater purpose. Make more casino games and rework the money system where you have chips instead of hard cash. When developing the game too we will create our original music and allow ourselves to organically grow into a staple fake gambling site.

References

https://bicyclecards.com/how-to-play/blackjack/

https://www.gamblingsites.org/casino/blackjack/

https://www.w3schools.com/howto/howto_make_a_website.asp

https://github.com/

https://pixabay.com/music/search/casino/