Team:

Chris Hitte
Gene Zhang
Brandon Boylan-Peck

Title: Settlers of Catan

Project Summary

For our semester project we will be implementing the popular board game, Settlers of Catan. At the end of the semester, we would like to have a Pass-and-Play version of Settlers of Catan which supports all components of the game (trading, placing settlements, earning resources, etc.) and enforces all rules. We will be focusing on the mechanics of the game over the look and feel. As such not all mechanics may be represented in the UI at the end of the project, but they will all be implemented properly.

Project Requirements

Business Requirements:

N/A

User Requirements				
ID	**Description**	**Actor**	**Topic**	**Priority**
US-01	As a Player, I want to start a new game with the option of 3 players and 4 players.	Player	Game Management	High
US-02	As a Player, I want to load a previously saved game and resume playing.	Player	Game Management	Low
US-03	As a Player, I want to set up my initial pieces at the start of the game.	Player	Setup	High
US-04	As the Current Player, I want to be able to start my turn.	Current Player	Rolling	High
US-05	As the Current Player, if I roll a 7 I want to be able to place the Robber and steal a card from a target Player.	Current Player, Other Player	Rolling	Low
US-06	As a Player, if a 7 is rolled and I have 7 or more cards in my had, I need to be able to discard the proper number of cards.	Player	Rolling	Low
US-07	As the Current Player I want to be able to trade Resource Cards with other Players	Current Player, Other Players	Trading	Med

US-08	As the Current Player I want to be able to trade Resources with the Bank.	Current Player	Bank	Med
US-09	As the Current Player I want to be able to purchase structures.	Current Player	Purchasing	High
US-10	As the Current Player I want to be able to purchase Development Cards	Current Player	Purchasing	Med
US-11	As the Current Player I need to be able to place structures I have purchased.	Current Player	Building	High
US-12	As the Current Player I need to be able to play Development Cards I have purchased.	Current Player	Development Cards	Med

Functional Requirements				
ID	**Description**	**Actor**	**Topic**	**Priority**
FR-01	The System will allow all Players to set up their pieces in order.	System, Players	Setup	High
FR-02	The System will begin each turn by rolling the dice.	System	Rolling	High
FR-03	The System will distribute the correct Resources to the correct Players based on the value of the roll.	System, Players	Resources	High
FR-04	When a Player is selected to be robbed, the System will randomly move one Resource from the target Player's inventory to the Current Player's inventory.	System, Current Player, Other Player	Robber	Low
FR-05	The System validates that the Current Player has the necessary Resources before any purchase is completed.	System, Current Player	Purchasing	High
FR-06	The System validates that a selected location is legal before the Current User builds a structure on it.	System, Current Player	Building	High
FR-07	The System does not allow the Current Player to play a Development Card in the same turn that it was purchased.	System, Current Player	Development Cards	Med
FR-08	The System will save all game state to the database when a game is exited.	System	Game Management	Low
FR-09	The System will reconstruct all game state when a Player loads a previously saved game.	System, Player	Game Management	Low

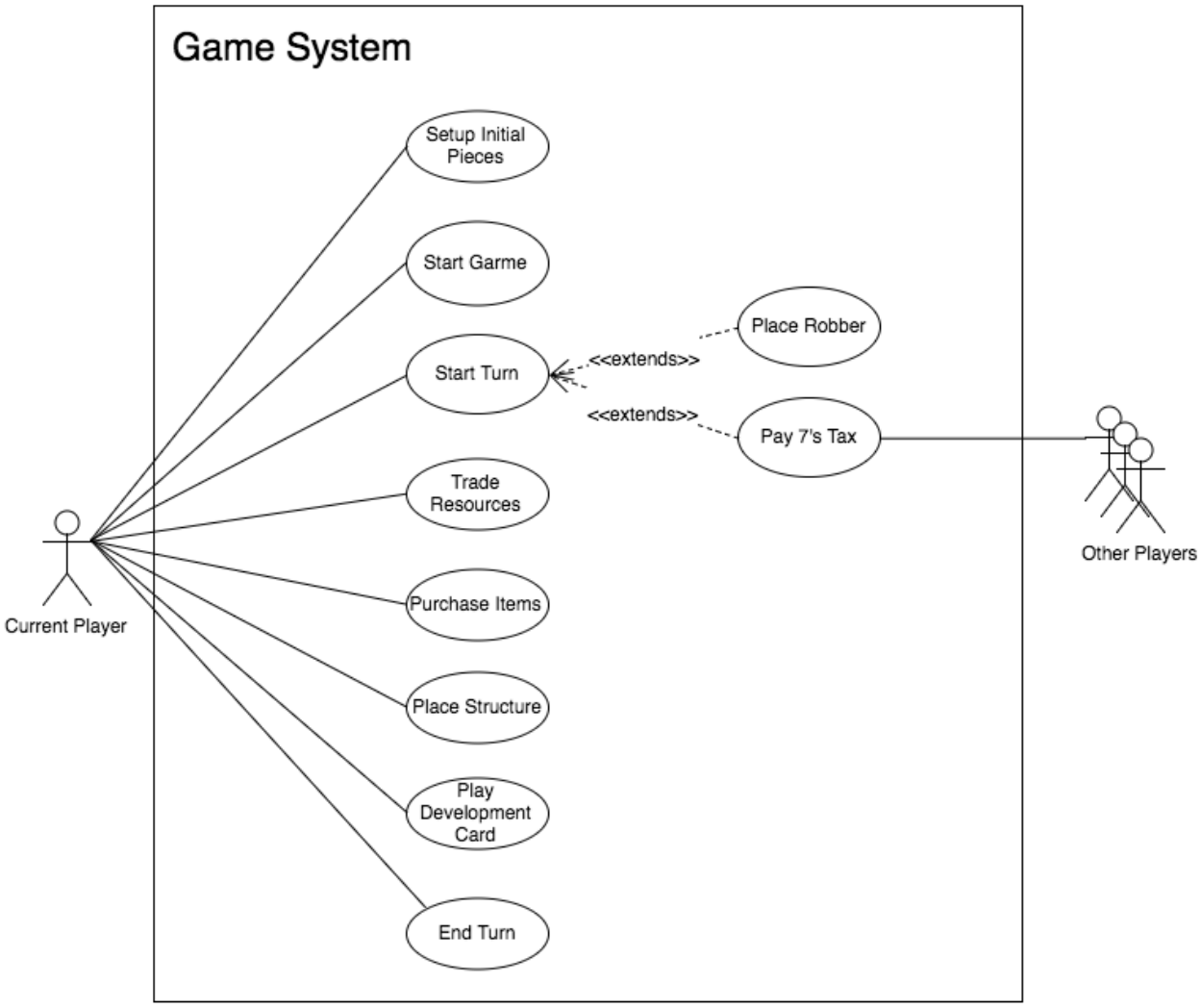
Non-Functional Requirements

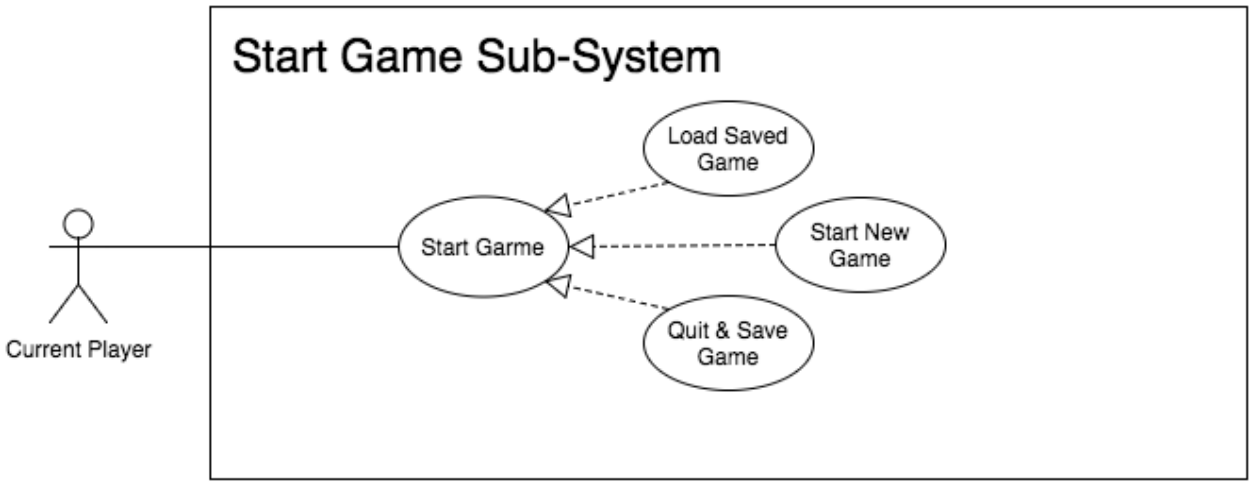
ID	Description	Actor	Topic	Priority
NFR-01	The System should run smoothly enough and quickly enough to support fun game play.	System	Performance	High
NFR-02	All game data such as Resources and Victory Points should be calculated and stored accurately.	System	Reliability	High

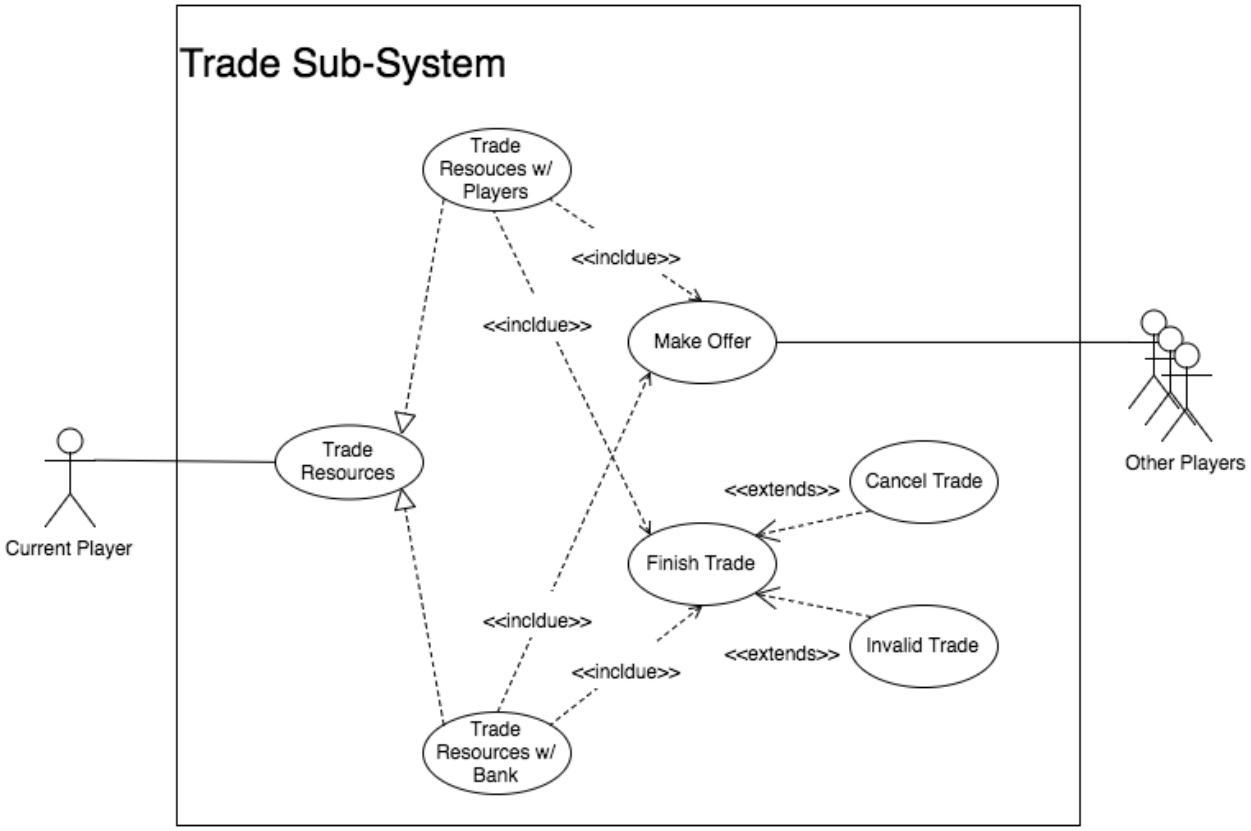
Use Cases

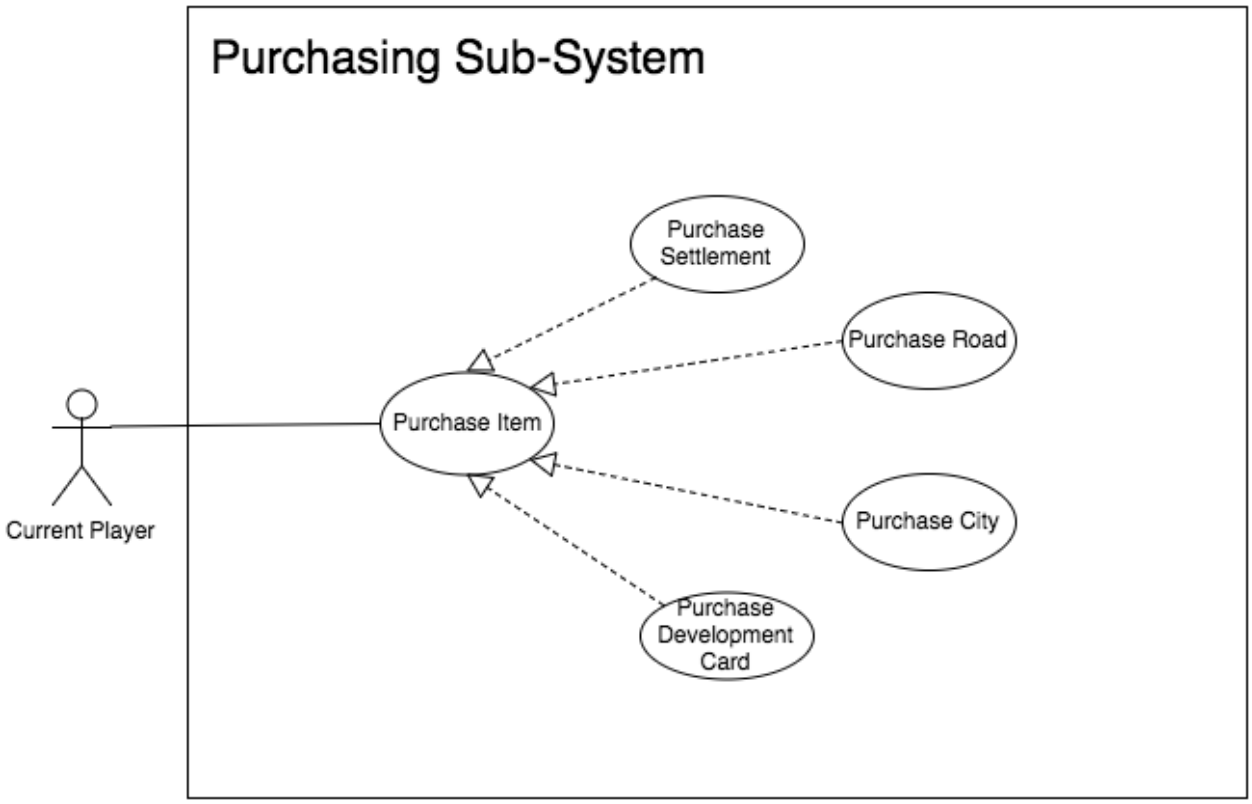
- Actors:
    - Player (general case)
    - Current Player
    - Other Player 1
    - Other Player 2
    - Other Player 3
    - System

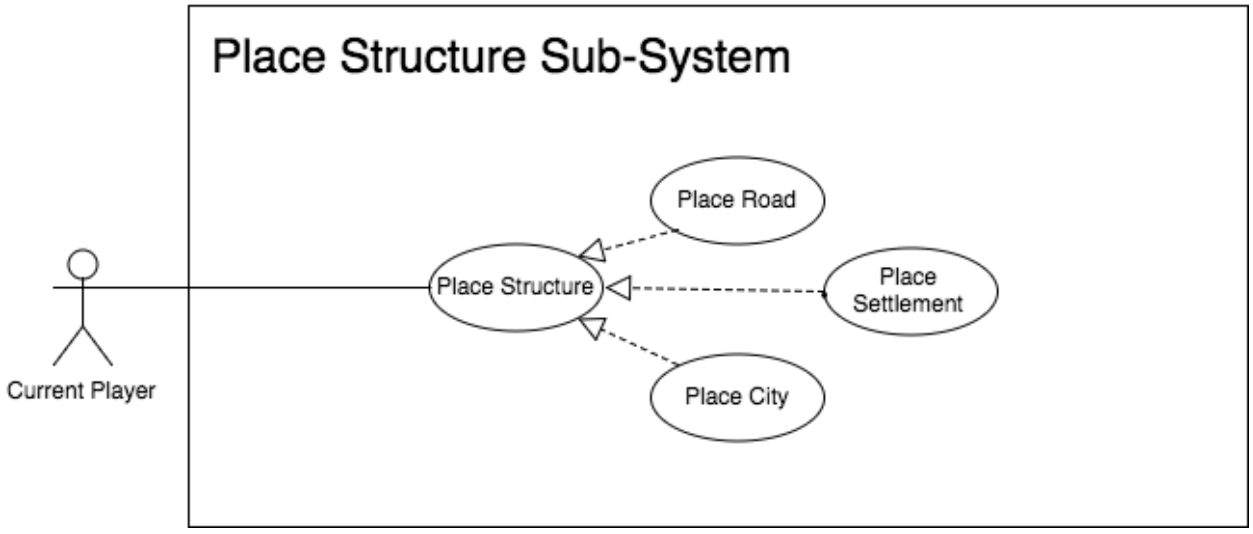
- Use Case Overview

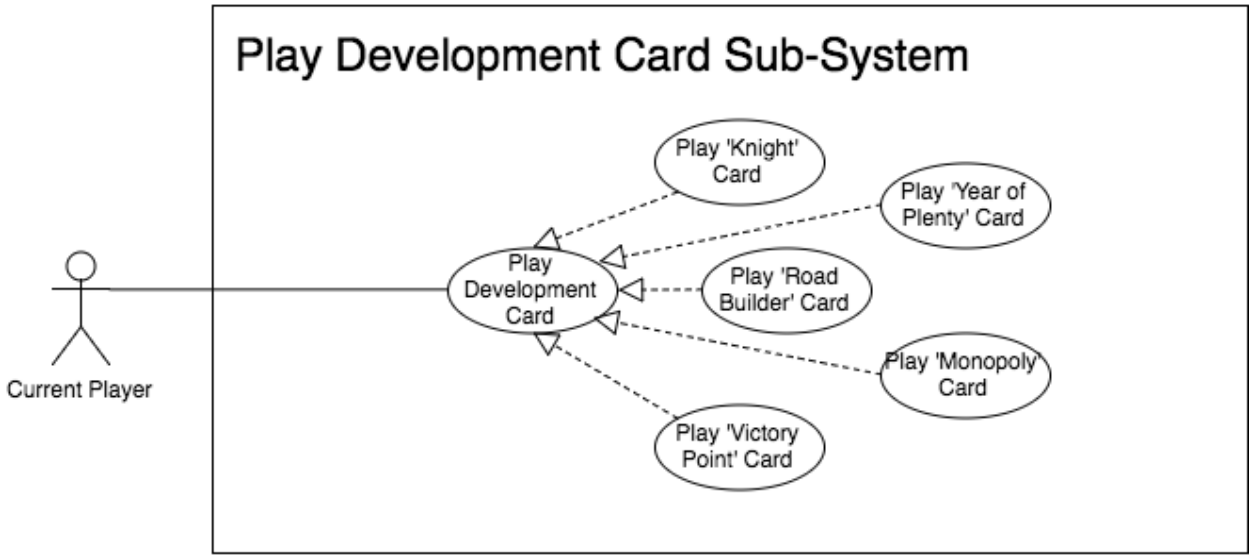



Start Game Sub-System
Load Saved Game
Start Garme
Start New Game
Quit & Save Game
Current Player


Trade Sub-System
Trade Resouces w/ Players
<<incldue>>
<<incldue>>
Make Offer
Other Players
Trade Resources
Current Player
<<extends>>
Cancel Trade
Finish Trade
<<incldue>>
<<extends>>
Invalid Trade
<<incldue>>
Trade Resources w/ Bank


Purchasing Sub-System
Purchase Settlement
Purchase Road
Purchase Item
Purchase City
Purchase Development Card
Current Player


Place Structure Sub-System
Place Road
Place Structure
Place Settlement
Place City
Current Player



- Use Case Documents

Use Case ID:	UC-01
Use Case Name:	Start New Game
Description:	Player begins new game
Actors:	Player
Pre-conditions:	
Post-conditions:	A new game has been created and started.
Frequency:	Once per game period
Flow of Events:	<table><tr><th></th><th>Actor Action</th><th>System Response</th></tr><tr><td>1</td><td>Player selects a 3 or 4 player game</td><td>System creates a new game for 3 or 4 players</td></tr></table>
Variations:	
Exceptions:	
Developer Notes:	

Use Case ID:	UC-02
Use Case Name:	Setup Initial Pieces

Description:	At the beginning of the game each Player takes turns placing a Road and a Settlement until each player has two Roads and two Settlements on the Board
Actors:	Current Player, Other Players
Pre-conditions:	Game has begun and is in the Setup Phase
Post-conditions:	Each Player has two Settlements and two Roads on the Board
Frequency:	Once at the beginning of the game
Flow of Events:	

	Actor Action	**System Response**
1	Player 1 selects a location for a Settlement and a Road and confirms placement	System verifies the selected location and updates the Board state
2	Player 2 selects a location for a Settlement and a Road and confirms placement	System verifies the selected location and updates the Board state
3	Player 3 selects a location for a Settlement and a Road and confirms placement	System verifies the selected location and updates the Board state
4	Player 4 selects a location for two Settlements and two Roads and confirms placement	System verifies the selected locations and updates the Board state
5	Player 3 selects a location for a Settlement and a Road and confirms placement	System verifies the selected location and updates the Board state
6	Player 2 selects a location for a Settlement and a Road and confirms placement	System verifies the selected location and updates the Board state
7	Player 1 selects a location for a Settlement and a Road and confirms placement	System verifies the selected location and updates the Board state

Variations:	
Exceptions:	1-7. Invalid location selected for Settlement or Road
Developer Notes:	If there are only 3 Players playing then the Flow will proceed as normal until it is Player 3's turn. Player 3 will put down both of his/her Settlements and Roads. After Player 3 is finished, then Player 2 goes, followed by Player 1.

Use Case ID:	UC-03
Use Case Name:	Load Saved Game
Description:	Player loads a saved game

Actors:	Player
Pre-conditions:	There exists a previously saved game on the system
Post-conditions:	The game is resumed.
Frequency:	Once per game period
Flow of Events:	

	Actor Action	System Response
1	Player selects a Saved Game form the list of games on the Main Menu	System loads the Saved Game data from the database and reinitializes the game

Variations:	
Exceptions:	
Developer Notes:	

Use Case ID:	UC-04
Use Case Name:	Start Turn
Description:	Current Player officially begins turn
Actors:	Current Player, Players
Pre-conditions:	Previous Player has ended his/her turn
Post-conditions:	Dice are rolled and resources are distributed, or robber is placed
Frequency:	Once per turn
Flow of Events:	

	Actor Action	System Response
1	User clicks “Start Turn” button	System rolls a 7
2	Players select Resource Cards to discard and confirms their selection	System validates that the appropriate number of Resource Cards have been discarded
3	Current Player picks a hex to move the Robber to	System validates the choice (must be a different tile than where the Robber began)
4	Current Player selects a player who owns a settlement on the selected hex to give up a card	System validates Player selection; randomly picks a Resource Card from that Player’s cards and gives it to the Current Player

Variations:	The system does not roll a 7 and distributes resources normally.
Exceptions:	

Developer Notes:	

Use Case ID:	UC-05
Use Case Name:	Trade Resources w/ Players
Description:	Current Player can open a Trade Menu to trade with other players
Actors:	Current Player, Other Players
Pre-conditions:	Current Player has started his/her turn
Post-conditions:	Trade Menu opens
Frequency:	0..* times per turn
Flow of Events:	

	Actor Action	System Response
1	Current Player clicks the 'Trade' button	System opens the Trading Menu

Variations:	
Exceptions:	
Developer Notes:	

Use Case ID:	UC-06
Use Case Name:	Trade Resources w/ Bank
Description:	Current Player can open a menu to trade Resources with the Bank
Actors:	Current Player
Pre-conditions:	Current Player has started his/her turn
Post-conditions:	Exchange Menu opens
Frequency:	0..* times per turn
Flow of Events:	

	Actor Action	System Response
1	Current Player clicks the 'Exchange' button	System opens the Trade with Bank Menu

Variations:	
Exceptions:	
Developer Notes:	

Use Case ID:	UC-07
Use Case Name:	Make Offer
Description:	Current Player or any Player can designate Resource Cards they are willing to Exchange or Trade
Actors:	Players
Pre-conditions:	Player has opened the Exchange or the Trade menu
Post-conditions:	Some combination of Resources is selected
Frequency:	0..* times per turn
Flow of Events:	

	Actor Action	**System Response**
1	Player designates a combination and quantity of Resources to exchange or trade	

Variations:	
Exceptions:	
Developer Notes:	

Use Case ID:	UC-08
Use Case Name:	Finish Trade
Description:	Current Player completes the Trade
Actors:	Current Player
Pre-conditions:	Current Player has opened either the Exchange or the Trade menu and all necessary Players have made their offer
Post-conditions:	Resource Cards are Exchanged/Traded and all relevant Players' inventories are updated
Frequency:	
Flow of Events:	

	Actor Action	**System Response**
1	Current Player clicks the 'Confirm' button	System validates that the Trade is legal. System updates each relevant Player's inventory. Menu closes.

Variations:	1. Current Player cancels trade
Exceptions:	1. Trade is illegal
Developer Notes:	

Use Case ID:	UC-09

Use Case Name:	Purchase Settlement
Description:	Current Player can purchase a Settlement
Actors:	Current Player
Pre-conditions:	Current Player has started his/her turn
Post-conditions:	Current Player exchanges Resource Cards for a Settlement
Frequency:	0..* times per turn
Flow of Events:	

	Actor Action	System Response
1	Current Player clicks the 'Purchase Settlement' Button	System validates the purchase and removes the required Resources from the Current Player's Inventory. System adds the Settlement to the Current Player's inventory. System switches to the Build Settlement state.

Variations:	
Exceptions:	3. Current Player does not have required Resources
Developer Notes:	

Use Case ID:	UC-10
Use Case Name:	Purchase Road
Description:	Current Player can purchase a Road
Actors:	Current Player
Pre-conditions:	Current Player has started his/her turn
Post-conditions:	Current Player exchanges Resources for a Road
Frequency:	0..* times per turn
Flow of Events:	

	Actor Action	System Response
1	Current Player clicks the 'Purchase Road' Button	System validates the purchase and removes the required Resources from the Current Player's Inventory. System adds the Road to the Current Player's inventory. System switches to the Build Road state.

Variations:	

Exceptions:	3. Current Player does not have required Resources
Developer Notes:	

Use Case ID:	UC-11
Use Case Name:	Purchase City
Description:	Current Player can purchase a City
Actors:	Current Player
Pre-conditions:	Current Player has started his/her turn
Post-conditions:	Current Player exchanges Resources for a City
Frequency:	0..* times per turn
Flow of Events:	

	Actor Action	System Response
1	Current Player clicks the 'Purchase City' Button	System validates the purchase and removes the required Resources from the Current Player's Inventory. System adds the City to the Current Player's inventory. System switches to the Build City state.

Variations:	
Exceptions:	3. Current Player does not have required Resources
Developer Notes:	

Use Case ID:	UC-12
Use Case Name:	Purchase Development Card
Description:	Current Player can purchase a Development Card
Actors:	Current Player
Pre-conditions:	Current Player has started his/her turn
Post-conditions:	Current Player exchanges Resources for a Development Card
Frequency:	0..* times per turn
Flow of Events:	

	Actor Action	System Response
1	Current Player clicks the 'Purchase Development Card' Button	System validates the purchase and removes the required Resources from the Current

			Player's Inventory. System adds the Development Card to the Current Player's inventory.
Variations:			
Exceptions:	3. Current Player does not have required Resources		
Developer Notes:			

Use Case ID:	UC-13		
Use Case Name:	Place Road		
Description:	Place a purchased Road on the Board		
Actors:	Current Player		
Pre-conditions:	Current Player has purchased a Road		
Post-conditions:	The Road has been placed on the Board		
Frequency:	0..* times per turn		
Flow of Events:		**Actor Action**	**System Response**
	1	Current Player selects a location on the board and confirms placement	System validates that the location is an Edge connected to another Edge that also has a Road belonging to the Current Player. System adds the Road at this location. System returns to the Default state.
Variations:			
Exceptions:	1. Invalid location selected		
Developer Notes:			

Use Case ID:	UC-14
Use Case Name:	Place City
Description:	Place a purchased City on the Board
Actors:	Current Player
Pre-conditions:	Current Player has purchased a City
Post-conditions:	The City has been placed on the Board
Frequency:	0..* times per turn

Flow of Events:		**Actor Action**	**System Response**
	1	Current Player selects a location on the board and confirms placement	System validates that the location is a Corner that already contains a Settlement belonging to the Current Player. System adds the City at this location. System returns to the Default state.
Variations:			
Exceptions:	1. Invalid location selected		
Developer Notes:			

Use Case ID:	UC-15		
Use Case Name:	Place Settlement		
Description:	Place a purchased Settlement on the Board		
Actors:	Current Player		
Pre-conditions:	Current Player has purchased a Settlement		
Post-conditions:	The Settlement has been placed on the Board		
Frequency:	0..* times per turn		
Flow of Events:		**Actor Action**	**System Response**
	1	Current Player selects a location on the board and confirms placement	System validates that the location is a Corner that does not belong to any player and that is not within 1 Edge of another Settlement. System adds the Settlement at this location. System returns to the Default state.
Variations:			
Exceptions:	1. Invalid location selected		
Developer Notes:			

Use Case ID:	UC-16
Use Case Name:	Play ‘Knight’ Card
Description:	Play a Knight card from the Current Player’s inventory
Actors:	Current Player, Other Player

Pre-conditions:	Current Player has a Knight card in his/her inventory
Post-conditions:	The Knight card is removed from the Current Player's inventory, the Robber is moved, and a card is stolen from a selected Player if possible
Frequency:	0..* times per turn

Flow of Events:

	Actor Action	System Response
1	Current Player presses the 'Play Knight Card' Button	System validates that the Current Player has a Knight card in his/her inventory that was not purchased this turn. System enters the Play Knight state.
2	Current Player selects a new Hex for the Robber	System validates that the new Hex is not the same as the previous Hex for the Robber. System returns a list of Other Players that share the new Hex.
3	Current Player selects a Player to steal from	System places a random Resource in the Current Player's inventory from the selected Player's inventory. System enters the Default state.

Variations:	
Exceptions:	1. No Knight card available; Invalid location selected
Developer Notes:	

Use Case ID:	UC-17
Use Case Name:	Play 'Road Builder' Card
Description:	Play a Road Builder card from the Current Player's inventory
Actors:	Current Player
Pre-conditions:	Current Player has a Road Builder card in his/her inventory
Post-conditions:	The Road Builder card is removed from the Current Player's inventory, the Current Player places 2 roads at valid locations on the Board
Frequency:	0..* times per turn

Flow of Events:

	Actor Action	System Response
1	Current Player presses the 'Play Road Builder Card' Button	System validates that the Current Player has a Road Builder card in his/her inventory that was not purchased this turn. System

			enters the Play Road Builder state.
	2	Current Player selects a location for his/her first Road	System validates that the location is an Edge connected to another Edge that also has a Road belonging to the Current Player. System adds the Road at this location.
	3	Current Player selects a location for his/her second Road	System validates that the location is an Edge connected to another Edge that also has a Road belonging to the Current Player. System adds the Road at this location. System returns to the Default state.
Variations:			
Exceptions:	1. No Road Builder card available; Invalid location selected		
Developer Notes:			

Use Case ID:	UC-18		
Use Case Name:	Play 'Monopoly' Card		
Description:	Play a Monopoly card from the Current Player's inventory		
Actors:	Current Player		
Pre-conditions:	Current Player has a Monopoly card in his/her inventory		
Post-conditions:	The Monopoly card is removed from the Current Player's inventory, all Resources of the selected type are removed from the Other Player's inventories and added to the Current Player's inventory		
Frequency:	0..* times per turn		
Flow of Events:		**Actor Action**	**System Response**
	1	Current Player presses the 'Play Monopoly Card' Button	System validates that the Current Player has a Monopoly card in his/her inventory that was not purchased this turn. System enters the Play Monopoly state.
	2	Current Player selects a Resource type	All Resources of the selected type are removed from the Other Player's inventories and added to the Current Player's inventory by the System.

Variations:	
Exceptions:	1. No Monopoly card available
Developer Notes:	

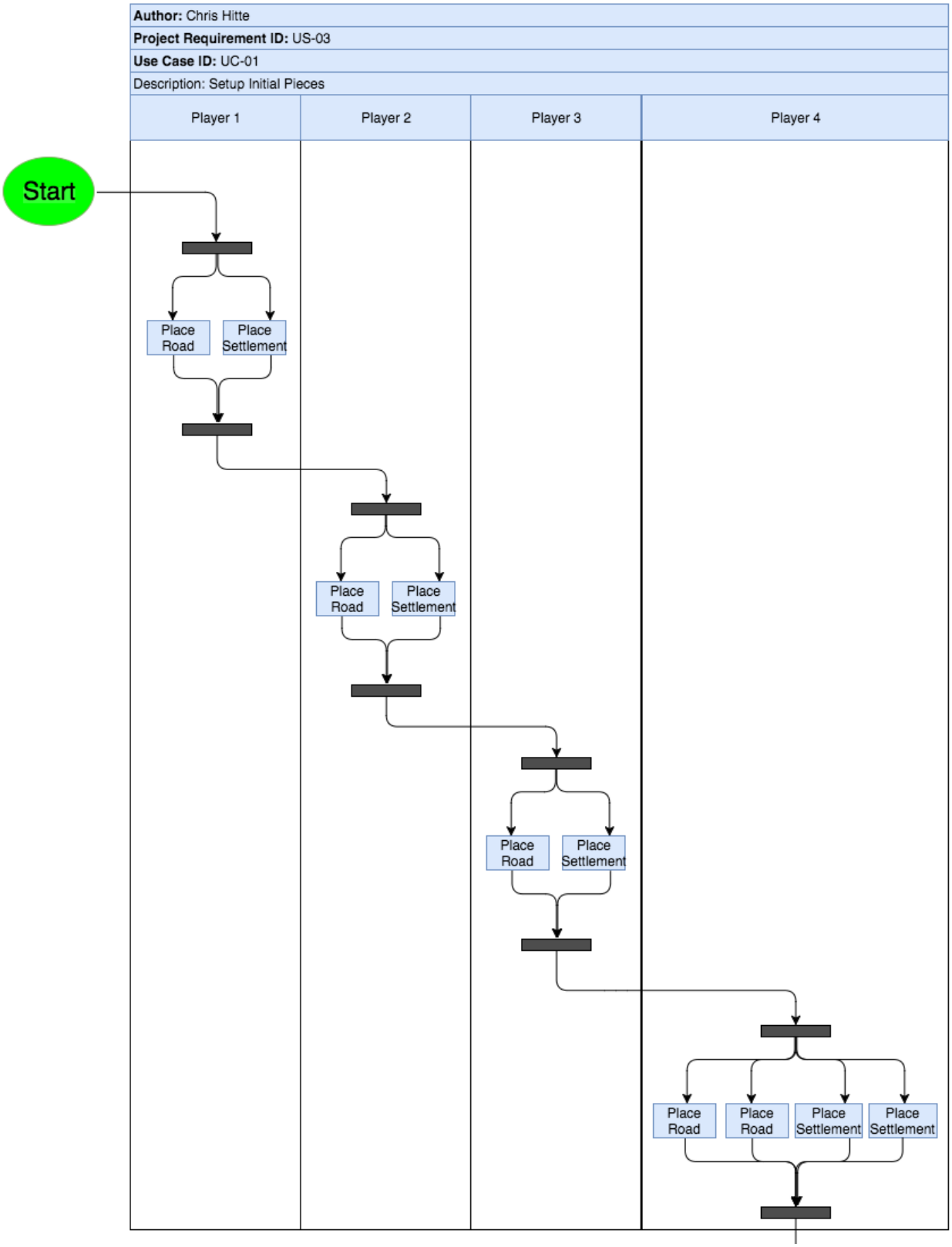
Use Case ID:	UC-19
Use Case Name:	Play 'Year of Plenty' Card
Description:	Play a Year of Plenty card from the Current Player's inventory
Actors:	Current Player
Pre-conditions:	Current Player has a Year of Plenty card in his/her inventory
Post-conditions:	The Year of Plenty card is removed from the Current Player's inventory, the Current Player selects two Resources from the Bank
Frequency:	0..* times per turn

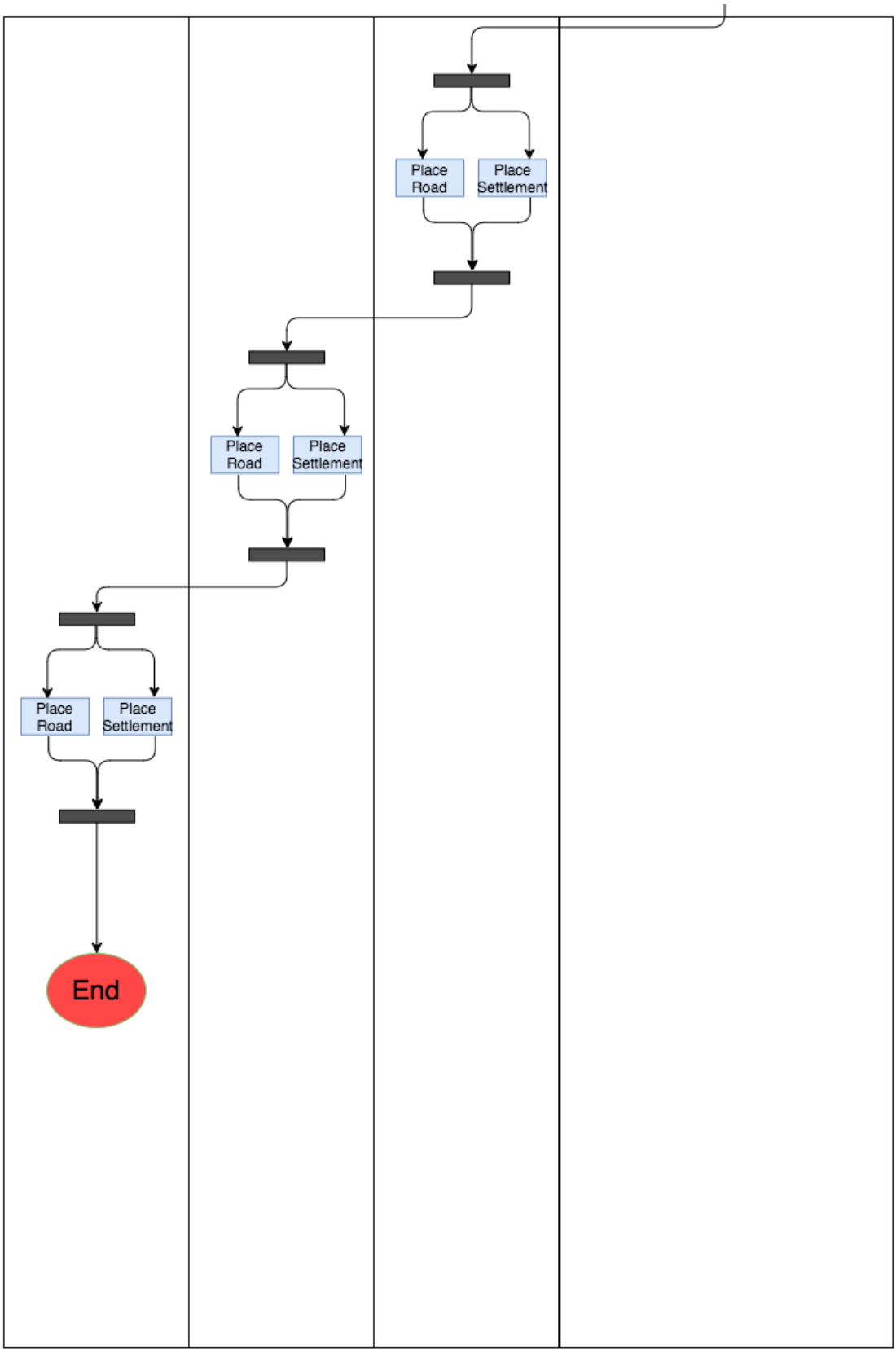
Flow of Events:

	Actor Action	System Response
1	Current Player presses the 'Play Year of Plenty Card' Button	System validates that the Current Player has a Year of Plenty card in his/her inventory that was not purchased this turn. System enters the Play Year of Plenty state.
2	Current Player selects two Resources	System adds the two Resources to the Current Player's inventory from the bank. System enters the Default state

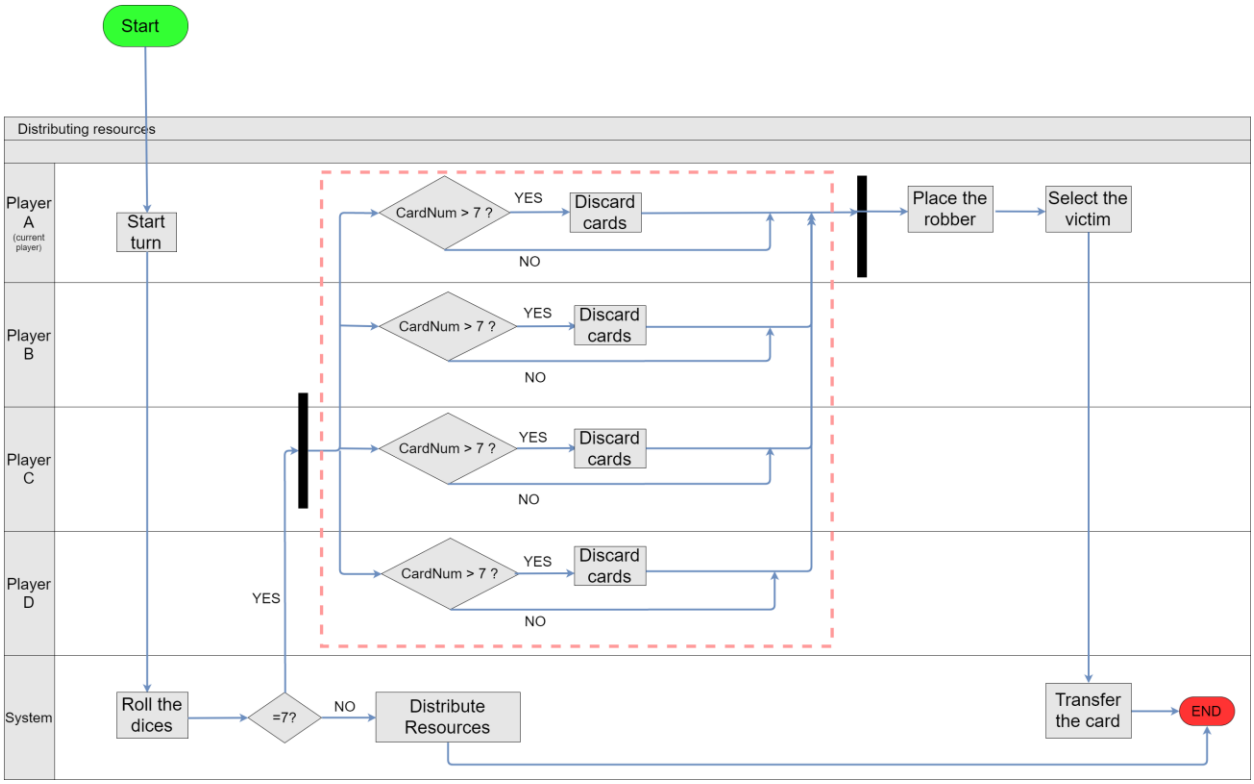
Variations:	
Exceptions:	1. No Year of Plenty card available
Developer Notes:	

Activity Diagrams



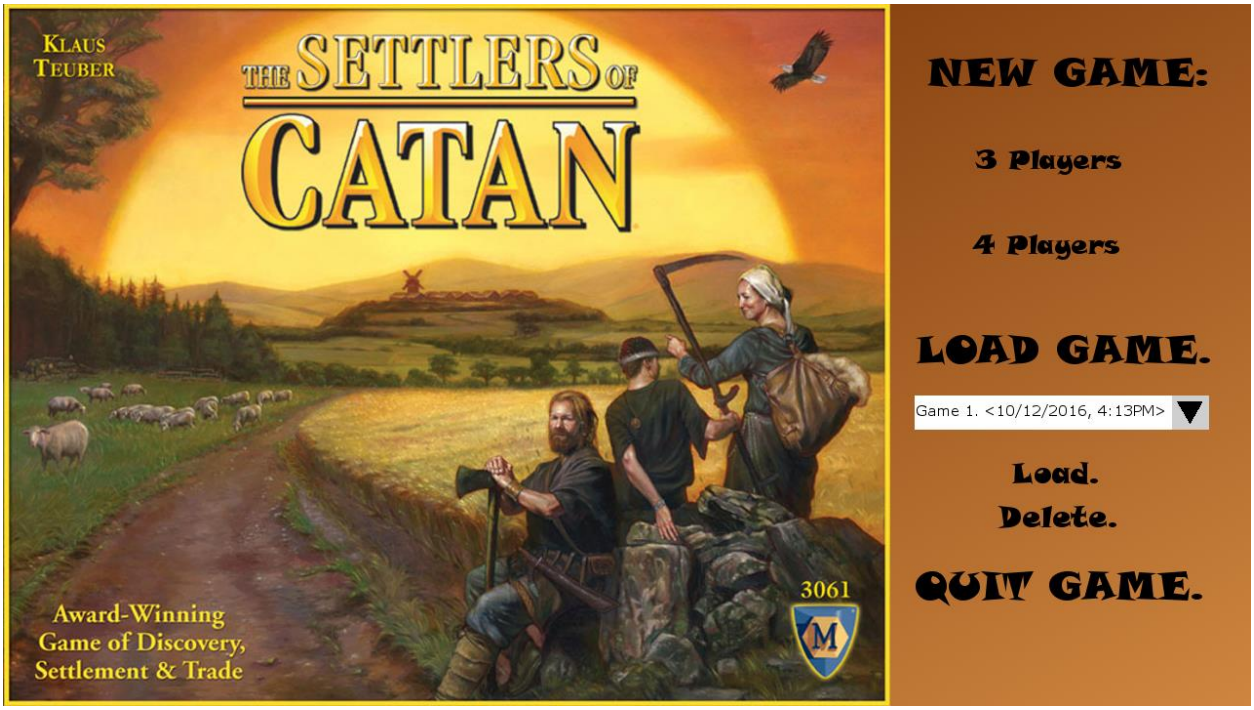

Place Road
Place Settlement
Place Road
Place Settlement
Place Road
Place Settlement
End

- Author: Yijun Zhang
- Description: Distributing resources or placing the robber
- Project Requirement ID: US-05, US-06, FR-03
- User Case ID: UC-04



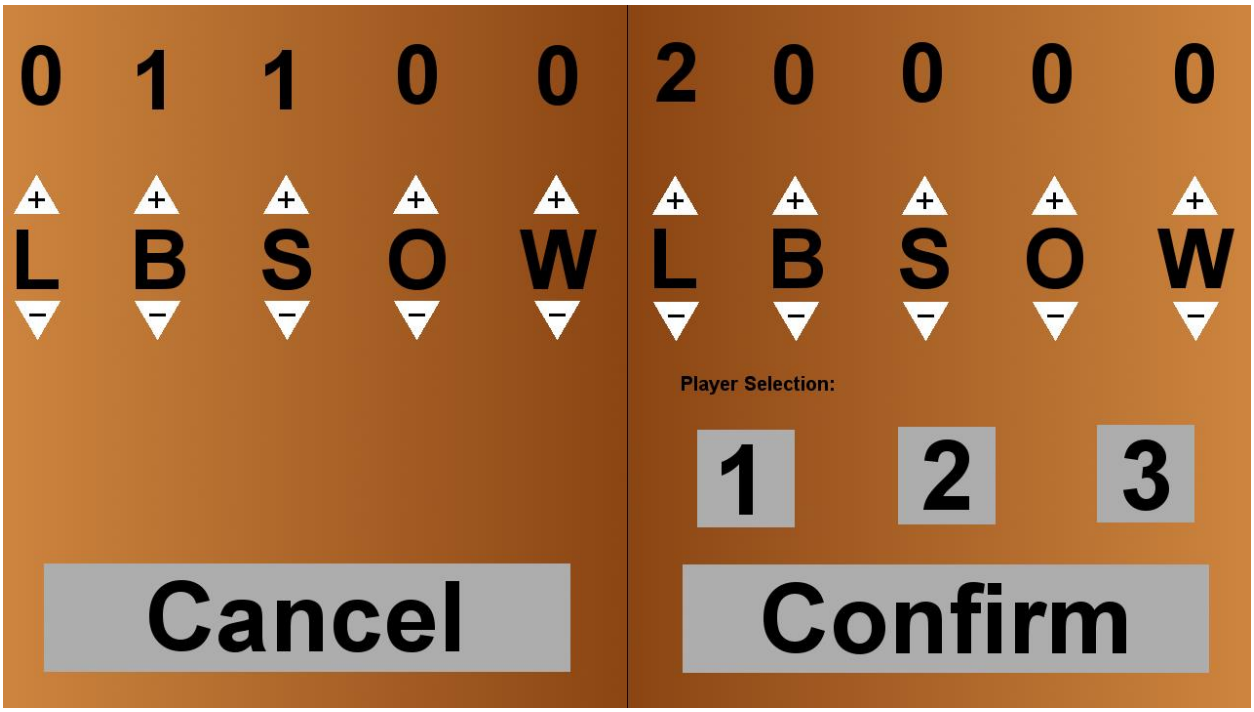
UI Mockups

- Main Menu UI

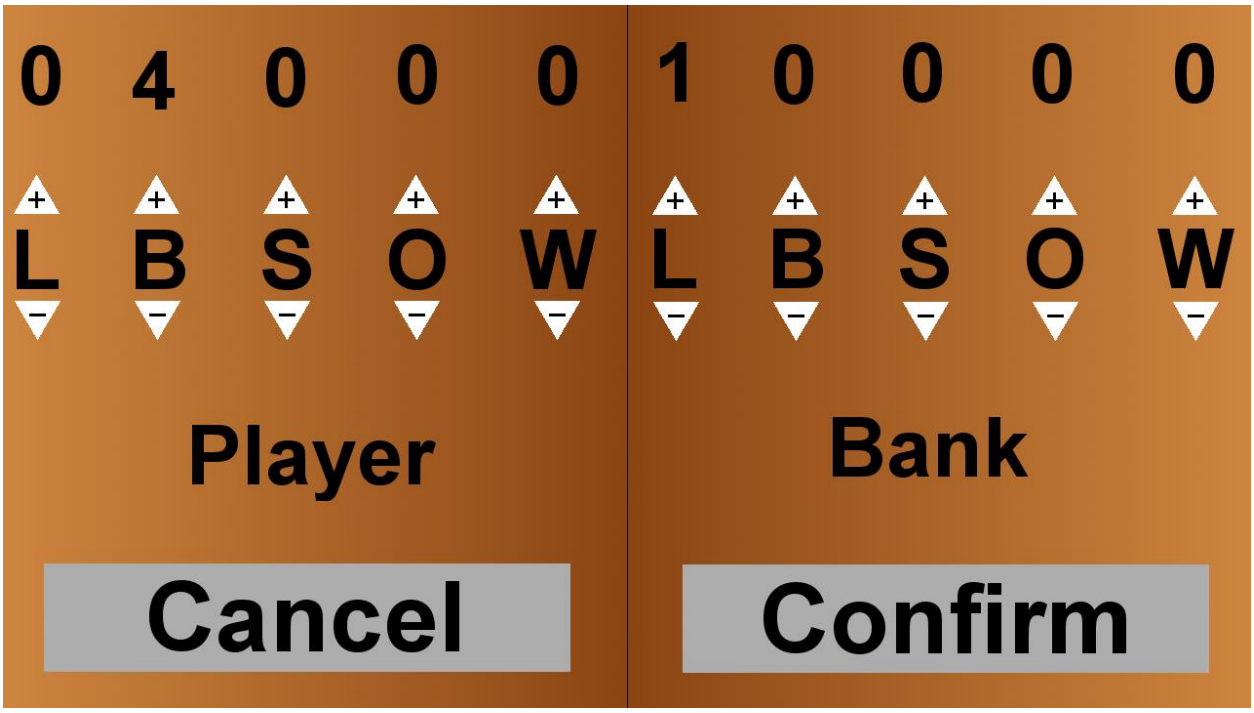


- Game UI

- Trade Player UI



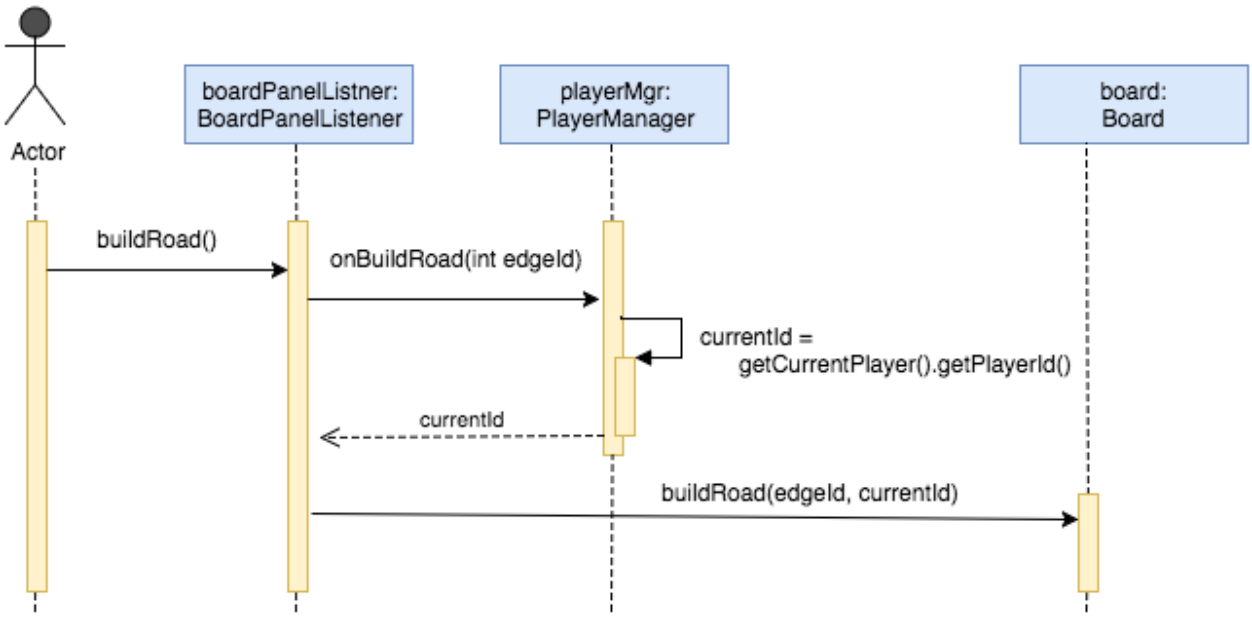
- Trade Bank UI



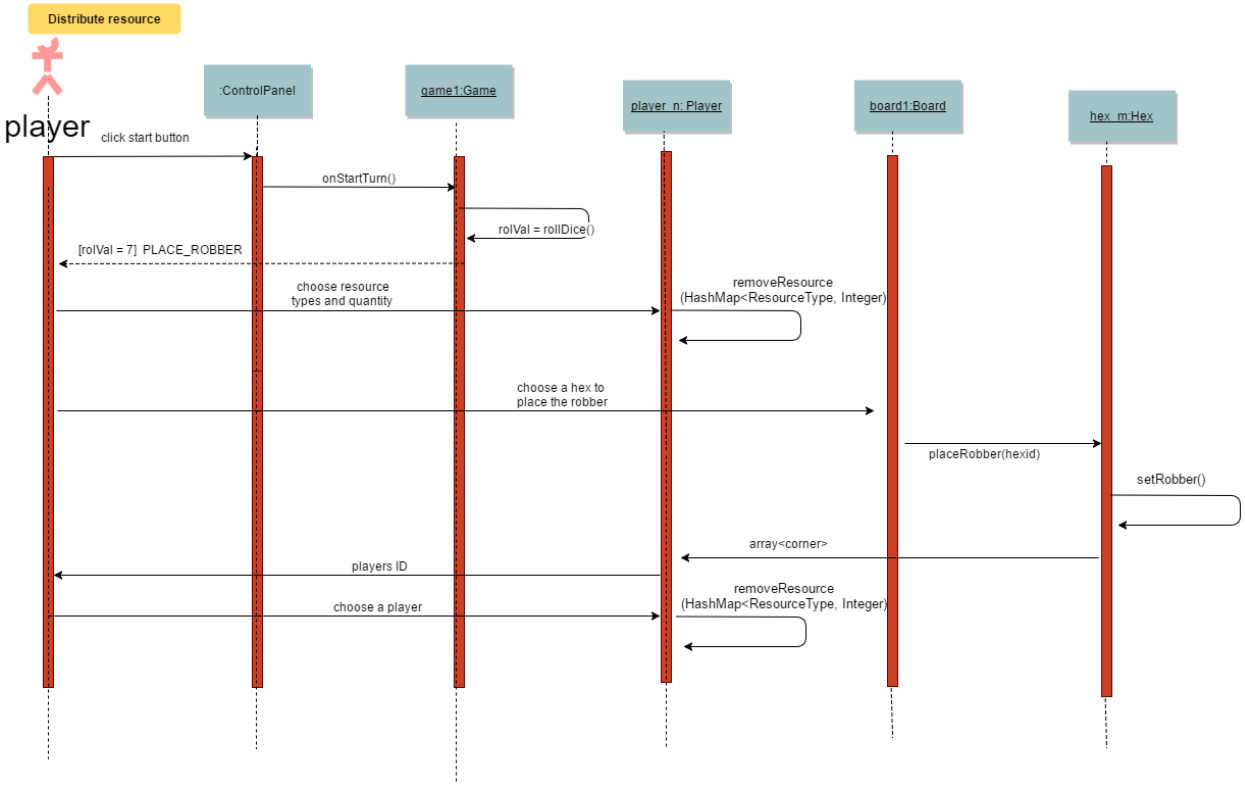
- Steal From UI

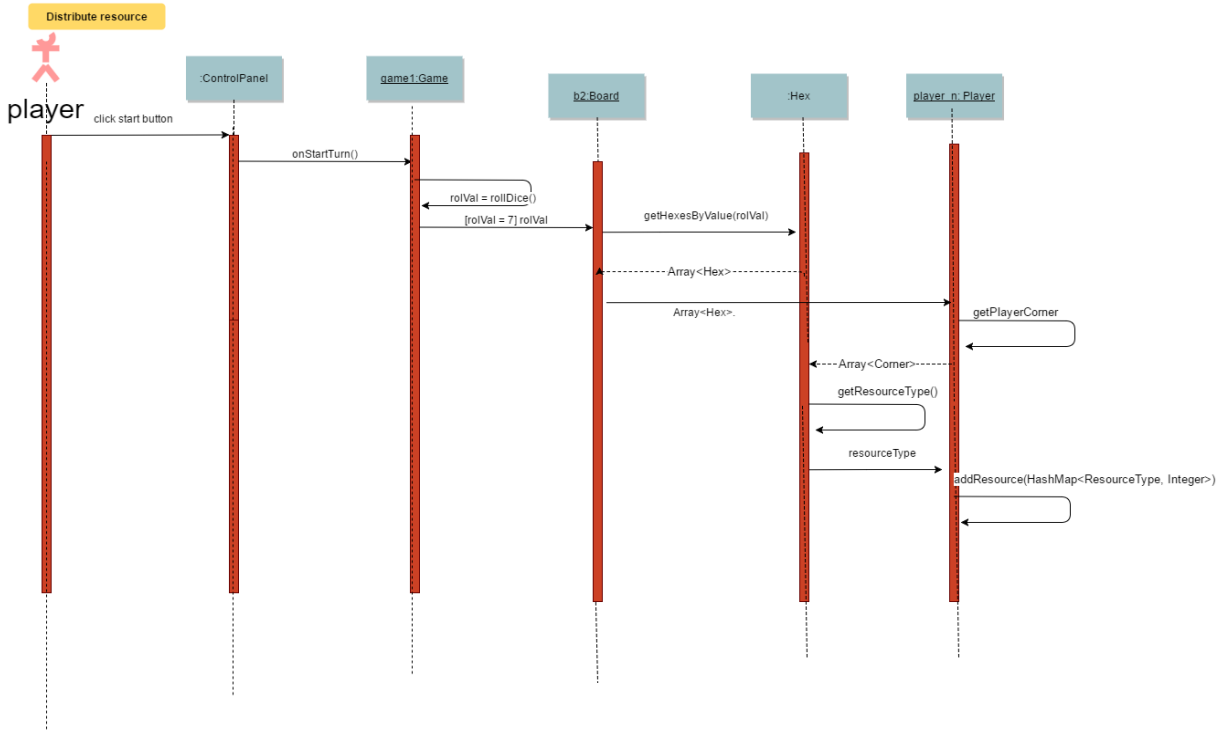
User Interactions

- Author: Chris Hitte
- Description: Setup Initial Pieces
- Project Requirement ID: US-03
- User Case ID: UC_01

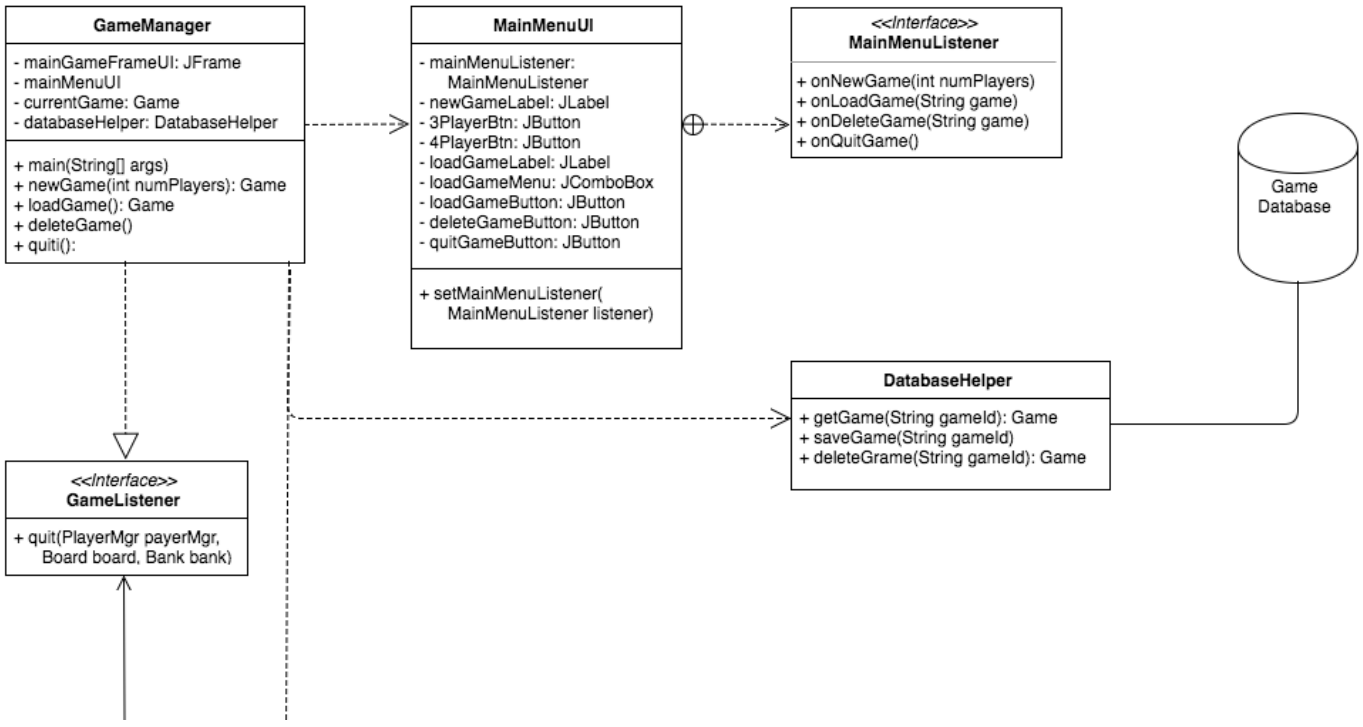


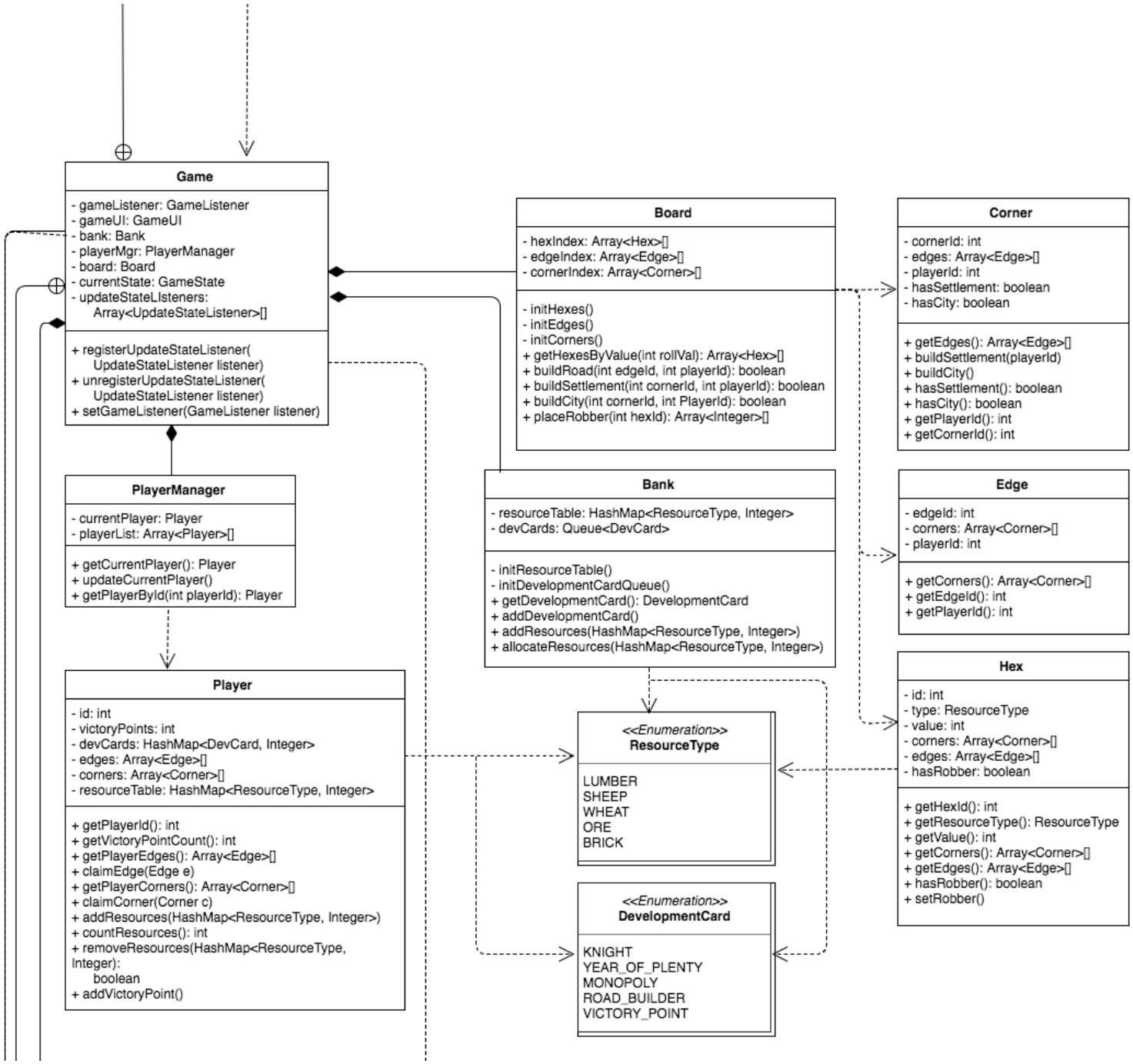
- Author: Yijun Zhang
- Description: (1) placing the robber (2) Distributing resources (2 if-else branches)
- Project Requirement ID: US-05, US-06,FR-03
- User Case ID: UC-04

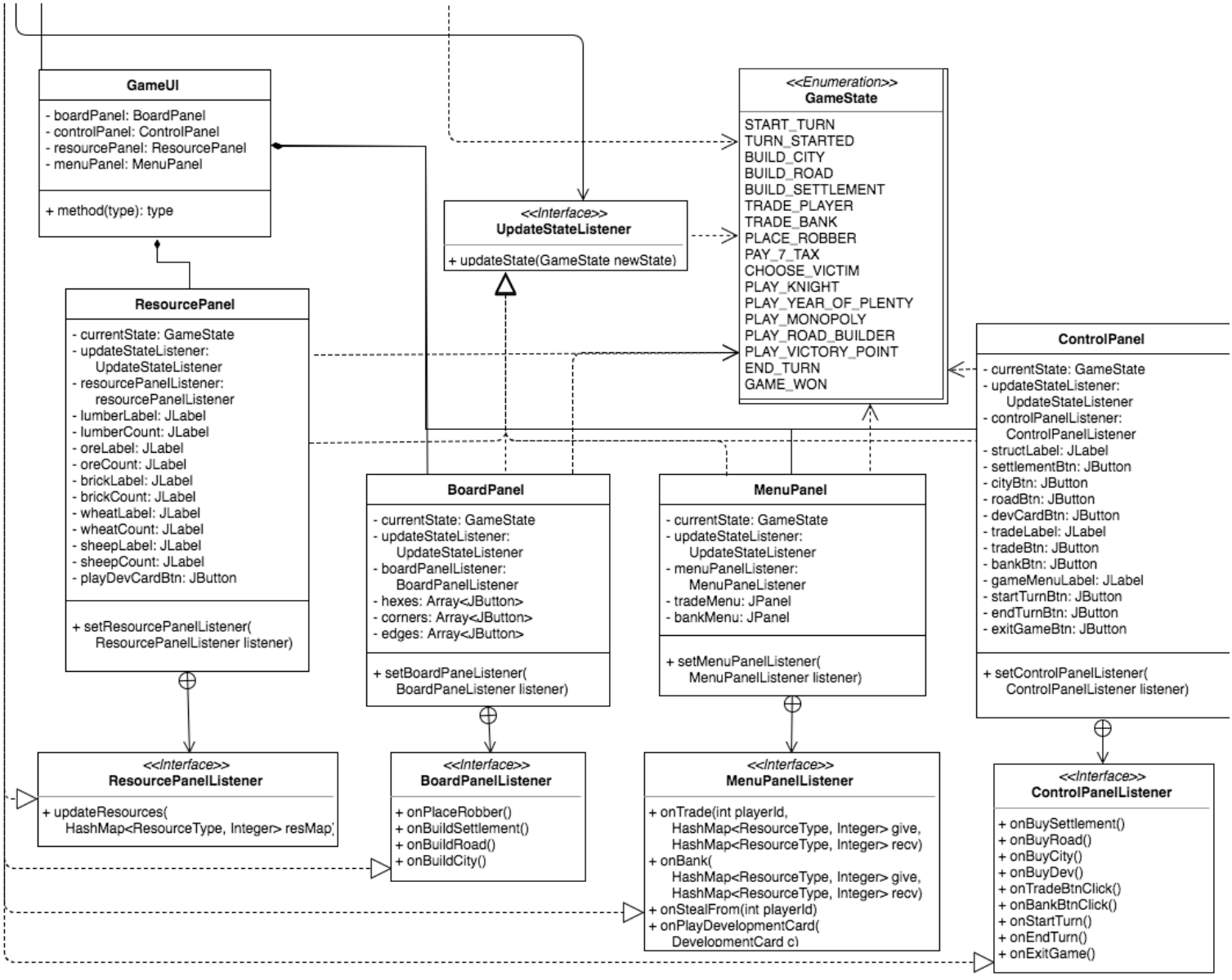




Class Diagram




Game
- gameListener: GameListener
- gameUI: GameUI
- bank: Bank
- playerMgr: PlayerManager
- board: Board
- currentState: GameState
- updateStateLIsteners:
Array<UpdateStateListener>[]
+ registerUpdateStateListener(
UpdateStateListener listener)
+ unregisterUpdateStateListener(
UpdateStateListener listener)
+ setGameListener(GameListener listener)
Board
- hexIndex: Array<Hex>[]
- edgeIndex: Array<Edge>[]
- cornerIndex: Array<Corner>[]
- initHexes()
- initEdges()
- initCorners()
+ getHexesByValue(int rollVal): Array<Hex>[]
+ buildRoad(int edgeId, int playerId): boolean
+ buildSettlement(int cornerId, int playerId): boolean
+ buildCity(int cornerId, int PlayerId): boolean
+ placeRobber(int hexId): Array<Integer>[]
Corner
- cornerId: int
- edges: Array<Edge>[]
- playerId: int
- hasSettlement: boolean
- hasCity: boolean
+ getEdges(): Array<Edge>[]
+ buildSettlement(playerId)
+ buildCity()
+ hasSettlement(): boolean
+ hasCity(): boolean
+ getPlayerId(): int
+ getCornerId(): int
PlayerManager
- currentPlayer: Player
- playerList: Array<Player>[]
+ getCurrentPlayer(): Player
+ updateCurrentPlayer()
+ getPlayerById(int playerId): Player
Bank
- resourceTable: HashMap<ResourceType, Integer>
- devCards: Queue<DevCard>
- initResourceTable()
- initDevelopmentCardQueue()
+ getDevelopmentCard(): DevelopmentCard
+ addDevelopmentCard()
+ addResources(HashMap<ResourceType, Integer>)
+ allocateResources(HashMap<ResourceType, Integer>)
Edge
- edgeId: int
- corners: Array<Corner>[]
- playerId: int
+ getCorners(): Array<Corner>[]
+ getEdgeId(): int
+ getPlayerId(): int
Player
- id: int
- victoryPoints: int
- devCards: HashMap<DevCard, Integer>
- edges: Array<Edge>[]
- corners: Array<Corner>[]
- resourceTable: HashMap<ResourceType, Integer>
+ getPlayerId(): int
+ getVictoryPointCount(): int
+ getPlayerEdges(): Array<Edge>[]
+ claimEdge(Edge e)
+ getPlayerCorners(): Array<Corner>[]
+ claimCorner(Corner c)
+ addResources(HashMap<ResourceType, Integer>)
+ countResources(): int
+ removeResources(HashMap<ResourceType,
Integer):
boolean
+ addVictoryPoint()
<<Enumeration>>
ResourceType
LUMBER
SHEEP
WHEAT
ORE
BRICK
Hex
- id: int
- type: ResourceType
- value: int
- corners: Array<Corner>[]
- edges: Array<Edge>[]
- hasRobber: boolean
+ getHexId(): int
+ getResourceType(): ResourceType
+ getValue(): int
+ getCorners(): Array<Corner>[]
+ getEdges(): Array<Edge>[]
+ hasRobber(): boolean
+ setRobber()
<<Enumeration>>
DevelopmentCard
KNIGHT
YEAR_OF_PLENTY
MONOPOLY
ROAD_BUILDER
VICTORY_POINT


GameUI
- boardPanel: BoardPanel
- controlPanel: ControlPanel
- resourcePanel: ResourcePanel
- menuPanel: MenuPanel
+ method(type): type
<<Interface>>
UpdateStateListener
+ updateState(GameState newState)
<<Enumeration>>
GameState
START_TURN
TURN_STARTED
BUILD_CITY
BUILD_ROAD
BUILD_SETTLEMENT
TRADE_PLAYER
TRADE_BANK
PLACE_ROBBER
PAY_7_TAX
CHOOSE_VICTIM
PLAY_KNIGHT
PLAY_YEAR_OF_PLENTY
PLAY_MONOPOLY
PLAY_ROAD_BUILDER
PLAY_VICTORY_POINT
END_TURN
GAME_WON
ResourcePanel
- currentState: GameState
- updateStateListener: UpdateStateListener
- resourcePanelListener: resourcePanelListener
- lumberLabel: JLabel
- lumberCount: JLabel
- oreLabel: JLabel
- oreCount: JLabel
- brickLabel: JLabel
- brickCount: JLabel
- wheatLabel: JLabel
- wheatCount: JLabel
- sheepLabel: JLabel
- sheepCount: JLabel
- playDevCardBtn: JButton
+ setResourcePanelListener(ResourcePanelListener listener)
BoardPanel
- currentState: GameState
- updateStateListener: UpdateStateListener
- boardPanelListener: BoardPanelListener
- hexes: Array<JButton>
- corners: Array<JButton>
- edges: Array<JButton>
+ setBoardPaneListener(BoardPaneListener listener)
MenuPanel
- currentState: GameState
- updateStateListener: UpdateStateListener
- menuPanelListener: MenuPaneListener
- tradeMenu: JPanel
- bankMenu: JPanel
+ setMenuPanelListener(MenuPanelListener listener)
ControlPanel
- currentState: GameState
- updateStateListener: UpdateStateListener
- controlPanelListener: ControlPanelListener
- structLabel: JLabel
- settlementBtn: JButton
- cityBtn: JButton
- roadBtn: JButton
- devCardBtn: JButton
- tradeLabel: JLabel
- tradeBtn: JButton
- bankBtn: JButton
- gameMenuLabel: JLabel
- startTurnBtn: JButton
- endTurnBtn: JButton
- exitGameBtn: JButton
+ setControlPanelListener(ControlPanelListener listener)
<<Interface>>
ResourcePanelListener
+ updateResources(HashMap<ResourceType, Integer> resMap)
<<Interface>>
BoardPanelListener
+ onPlaceRobber()
+ onBuildSettlement()
+ onBuildRoad()
+ onBuildCity()
<<Interface>>
MenuPanelListener
+ onTrade(int playerId, HashMap<ResourceType, Integer> give, HashMap<ResourceType, Integer> recv)
+ onBank(HashMap<ResourceType, Integer> give, HashMap<ResourceType, Integer> recv)
+ onStealFrom(int playerId)
+ onPlayDevelopmentCard(DevelopmentCard c)
<<Interface>>
ControlPanelListener
+ onBuySettlement()
+ onBuyRoad()
+ onBuyCity()
+ onBuyDev()
+ onTradeBtnClick()
+ onBankBtnClick()
+ onStartTurn()
+ onEndTurn()
+ onExitGame()

Data Usage

Our system will utilize a MySql Database to store all relevant game data. This will facilitate the saving and resuming of games.