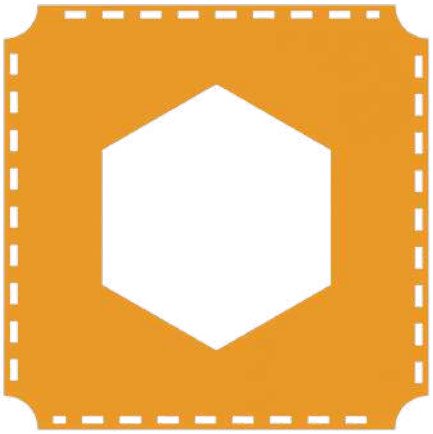


Bitpotz White Paper

v1.0

April 2026

Introduction

Bitpotz is an NFT minting platform that enables users to earn crypto rewards by minting NFTs. Users participate in jackpots by minting NFTs. Each Jackpot is a verified smart contract and uses the native cryptocurrency of the blockchain it is deployed on.

Users participate in Jackpots by minting NFTs called Tickets. These Tickets are tradable, sellable, and never wasted.

There are 5 Jackpots for each cryptocurrency: **HourPot**, **DayPot**, **WeekPot**, **MonthPot** and **GrandPot**. Each Jackpot consists of repeating rounds and each round results in 1 winner receiving the entire balance accumulated in the related smart contract.

Bitpotz algorithms are scheduled and run systematically without any human intervention. Some of the operations are done in smart contracts using Solidty, while others are done on the application side using NodeJS. The randomness in all algorithms is based on NodeJS Math methods.

Jackpots have common features, but each Jackpot also has different features.

Definitions

To clarify the key terms used in this whitepaper, the following definitions are provided:

Ticket: Represents an NFT (Non-Fungible Token). It can be likened to a lottery ticket in the real world, serving as a unique entry for participants.

Jackpot: Represents an NFT Smart Contract. Similar to a lottery draw in the real world, it is responsible for managing and distributing prizes.

Round: Represents the period between the start and end of a Jackpot. It can be compared to the duration of a lottery event in the real world.

Code explanations are simplified for clarity. Bitpotz does not claim that it uses the exact same code.

Common Jackpot Features

The common features of all Jackpots are as follows:

1- Contract Status

Bitpotz smart contracts have the isContractActive property for possible unexpected scenarios. When this feature is false, the contract becomes passive and the relevant Jackpot is paused. If participants try participating during a pause state, transactions are automatically rejected by the contract. It is changeable and announced clearly.

Below is a contract code for setting contract status:

```
/* JACKPOT IS CONTRACT ACTIVE */

bool public isContractActive = true;
```

```
//Activating contract.
function activateContract() external onlyOwner {
   isContractActive = true;
}

//Deactivating contract.
function deactivateContract() external onlyOwner {
   isContractActive = false;
}

/* JACKPOT IS CONTRACT ACTIVE */
```

2- Ticket Price

Ticket Price for each Jackpot is determined by Bitpotz in the native cryptocurrency of the blockchain it is deployed on. In smart contracts it is called nftPrice. It is changeable and announced clearly.

Below is a contract code for setting ticket price:

```
/* JACKPOT TICKET PRICE */

uint256 public nftPrice = 0.0001e18; // 0.0001 BNB

//Changing ticket price.
function setPrice(uint256 newPrice) external onlyOwner {
   nftPrice = newPrice;
}

/* JACKPOT TICKET PRICE */
```

3- Participation & Ticket Minting

Participation in Jackpots is done by transferring the relevant native cryptocurrency to the relevant smart contract. Participants mint Tickets if they transfer the required amount.

Rules:

-Contract owner cannot be a participant.

-A Ticket minting can not exceed the maxNftOnce value.

-A participant can mint an unlimited number of Tickets without exceeding the maxNftOnce value.

-The amount of cryptocurrency sent to the contract for participation must be an exact multiple of the Ticket Price.

*Ticket mintings that do not comply with the rules will be rejected by the contract and the amount sent will be refunded.

An example Ticket minting is as follows;

maxNftOnce = 10

nftPrice = 1 BNB

Sending 5 BNB to the contract address = successfully minted 5 Tickets.

Sending 11 BNB to the contract address = will be rejected by contract because of exceeding the maxNftOnce value.

Sending 5.5 BNB to the contract address = will be rejected by contract because 5.5 is not an exact multiple of the nftPrice.

Below is a contract code for participation & ticket minting:

```
/* JACKPOT PARTICIPATION & MINTING TICKETS */

    receive() external payable nonReentrant {
        if (isContractActive) {
            if (msg.sender != manager) {
                if (
                    msg.value > 0 &&
                    msg.value % nftPrice == 0 &&
                    msg.value / nftPrice <= maxNftOnce
                ) {
                    payPercentages(msg.value);
                    for (uint256 i = 0; i < msg.value / nftPrice; i++) {
                        newTokenId += 1;
                        _safeMint(msg.sender, newTokenId);

                         // Vip control only exists in BNB Jackpots.
                        if (isVip(msg.sender)) {
                            newTokenId += 1;
                            _safeMint(msg.sender, newTokenId);
                        }
                    }
                } else {
                    revert();
                }
            } else {
                revert();
            }
        } else {
            revert();
        }
    }

/* JACKPOT PARTICIPATION & MINTING TICKETS */
```

4- Minting Multiple Tickets at once

Each Jackpot limits the maximum number of Tickets that can be minted at a time. This value is called maxNftOnce in smart contracts. It is changeable and announced clearly.

Below is a contract code for setting max NFT at once limit:

```
/* JACKPOT MAX NFT AT ONCE */

uint256 public maxNftOnce = 10;

/* JACKPOT MAX NFT AT ONCE */
```

5- Funding

Fundings aims to provide funding for the platform by taking a certain percentage of commission from the minted tickets. The percentages are determined by Bitpotz and announced clearly. There are 2 funding types in Bitpotz smart contracts:

GrandPot Funding:

It aims to increase the GrandPot prize amount. For every ticket minting made from all Jackpots except GrandPot, a certain amount of cryptocurrency is transferred to the GrandPot contract in the same currency. In smart contracts it is called grandJackpotPercentage. It is changeable and announced clearly.

Below is a contract code for setting GrandPot funding:

```
/* JACKPOT GRANDPOT FUNDING */

uint256 private grandJackpotPercentage = 25; // 25%

//Changing grandpot funding percentage.
function setGrandJackpotPercentage(uint256 percentage) external onlyOwner {
   grandJackpotPercentage = percentage;
}

/* JACKPOT GRANDPOT FUNDING */
```

Platform Funding:

It aims to ensure the sustainability of the platform and fund the fees for transactions made by contract owners such as payToWinner. For every ticket minting made from all Jackpots, a certain amount of cryptocurrency is transferred to Platform Funding Wallet. In smart contracts it is called platformFundingPercentage. It is changeable and announced clearly.

Below is a contract code for setting development funding:

```
/* JACKPOT PLATFORM FUNDING */

uint256 private platformFundingPercentage = 5; // 5%

//Changing platform funding percentage.
function setPlatformFundingPercentage(uint256 percentage) external onlyOwner
 {
   platformFundingPercentage = percentage;
 }


/* JACKPOT PLATFORM FUNDING */
```

Below is a contract code for sending funds:

```
/* JACKPOT SENDING FUNDS */

  function payPercentages(uint256 value) internal {
       uint256 platformFunding = (value / 100) * platformFundingPercentage;
       uint256 grandJackpot = (value / 100) * grandJackpotPercentage;
       payToPlatform(platformFundingWallet, platformFunding);
       payToGrandJackpot(grandJackpotContract, grandJackpot);
   }

//Active for all jackpots.
   function payToPlatform(
       address walletAddress,
       uint256 value
   ) internal returns (bool) {
       (bool success, ) = payable(walletAddress).call{value: value}("");
       return success;
   }

//Active for all jackpots except grandpot.
   function payToGrandJackpot(
       address contractAddress,
       uint256 value
   ) internal returns (bool) {
       (bool success, ) = payable(contractAddress).call{value: value}("");
       return success;
   }

/* JACKPOT SENDING FUNDS */
```

6- Vip ($POTZ Holders)

Vip is the title given to BNB Jackpot participants who have a certain amount of the official platform token $POTZ in their wallet. The certain amount is determined by Bitpotz and called as vipLimit in the smart contract. It is changeable and announced clearly.

- Valid on **BNB Jackpots only**.

- Participants must have at least a **vipLimit** value of $POTZ in the **same** wallet address they used to participate in the Jackpot.

- VIP participants mint **1 more free ticket for every ticket they mint.**

Below is a contract code for setting vip limit and controlling vip status:

```
/* BNB JACKPOT VIP */

uint256 public vipLimit = 10000e18;  // 10000 $POTZ

//Controlling vip status for only BNB Jackpots.
function isVip(address holder) public view returns (bool) {
   IERC20 platformToken = IERC20(platformTokenAddress);
   return platformToken.balanceOf(holder) >= vipLimit;
}

/* BNB JACKPOT VIP */
```

7- Round Ending & Rewarding

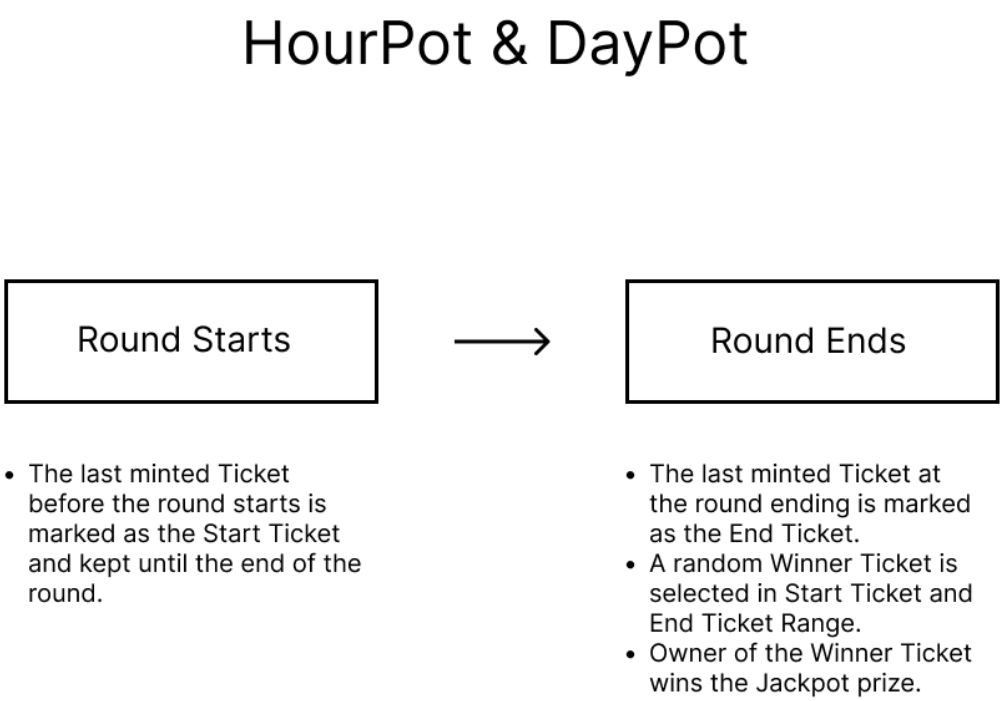
When a round ends, one Winner Ticket is randomly selected and the owner of the Winner Ticket wins **the entire balance** accumulated in the relevant smart contract. It is technically possible for the same ticket to win a Jackpot **repeatedly.** If the winner is not determined for some reason, the round is preserved and restarted.

To understand how each Jackpot round ends, review the Jackpot details grouped below.

```
/* JACKPOT ROUND ENDING WITH A WINNER */

function payToWinner(
       address winner
   ) external onlyOwner nonReentrant returns (bool) {
       uint256 balance = address(this).balance;
       (bool success, ) = payable(winner).call{value: balance}("");
       return success;
   }

/* JACKPOT ROUND ENDING WITH A WINNER */
```

HourPot & DayPot

HourPot and DayPot use the same mechanism. The only difference is that HourPot rounds restart every hour, while DayPot rounds restart every 24 hours.



HourPot rounds restart **every hour**, DayPot rounds restart **every 24 hours**.

4 factors are taken into consideration when determining the winner:

1-Total Supply

It is one of the public methods of Bitpotz smart contracts. The totalSupply method always returns the last minted Ticket ID value and increases by 1 each time a Ticket is minted.

```
/* HOURPOT & DAYPOT TOTAL SUPPLY */

const totalSupply = Jackpot.methods.totalSupply();
//e.g. totalSupply= 1211;

/* HOURPOT & DAYPOT TOTAL SUPPLY */
```

2- Start Ticket

Before each round starts, a constant called Start Ticket stores the last minted Ticket ID. It uses the totalSupply method in the smart contract.

```
/* HOURPOT & DAYPOT START TICKET */

//Calculated and stored before the round starts using totalSupply public
 contract method.
const startTicket = Jackpot.methods.totalSupply();
//e.g. startTicket = 132;

/* HOURPOT & DAYPOT START TICKET */
```

3- End Ticket

End Ticket represents the last minted Ticket ID value when the round ends. It uses the totalSupply method in the smart contract.

```
/* HOURPOT & DAYPOT END TICKET */

//Calculated at the round ending using totalSupply public contract method.
const endTicket = Jackpot.methods.totalSupply();
//e.g. endTicket = 798;

/* HOURPOT & DAYPOT END TICKET */
```

4- Round Ending & Rewarding

When the round ends, a Winner Ticket is randomly selected from the Start Ticket and End Ticket ranges (Start Ticket is not included). The owner of the selected Winner Ticket will be the Winner Address.

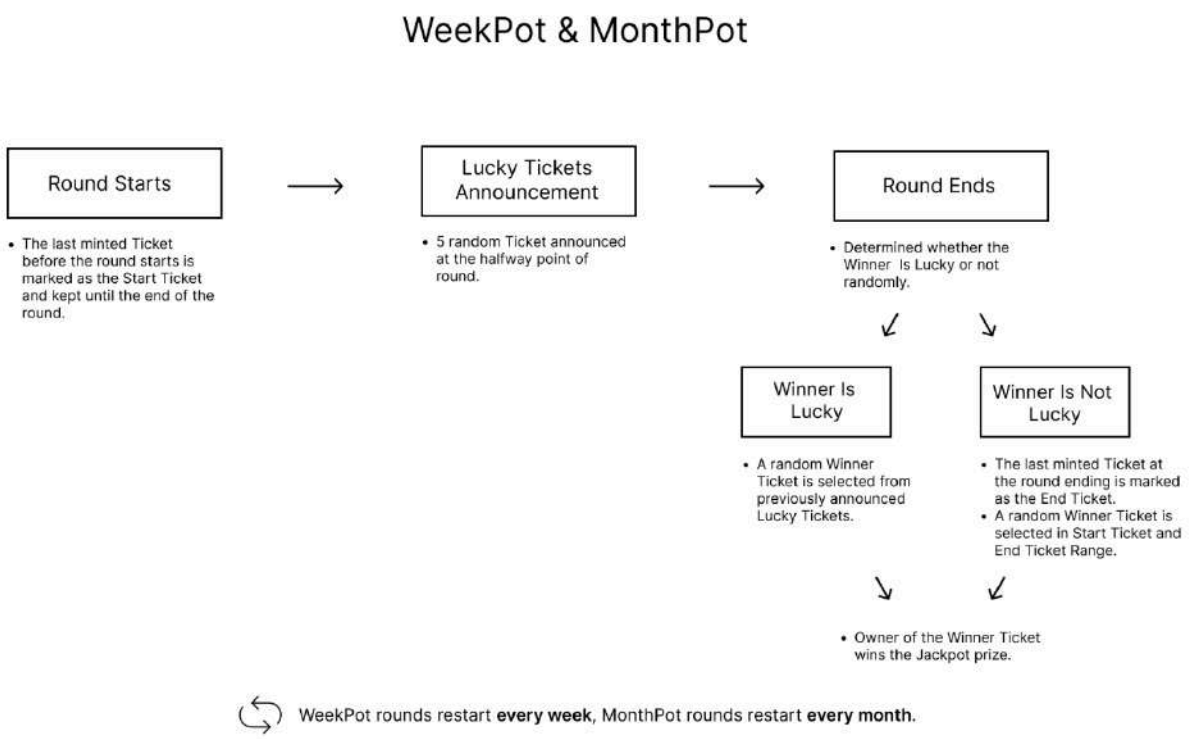
Below is a simplified code for random winner selection:

```
/* HOURPOT & DAYPOT REWARDING */

//Calculated and stored before the round starts using totalSupply public
 contract method.
const startTicket = Jackpot.methods.totalSupply();
//e.g. startTicket = 132;

//Calculated at the round ending using totalSupply public contract method.
const endTicket = Jackpot.methods.totalSupply();
//e.g. endTicket = 798;
```

```
//Function for randomly selecting one of startTicket and endTicket ranges.
function randomTicket(startTicket, endTicket) {

 const totalTicketAmount = endTicket - startTicket;
 const randomMargin = Math.floor(Math.random() * totalTicketAmount) + 1;

 //startTicket is not included.
 return startTicket + randomMargin;
}


//Determining Winner Ticket. (in round Tickets minted from the Jackpot are
included.)
const winnerTicket = randomTicket(startTicket, endTicket);
//e.g. winnerTicket = 465;

//Determining Winner Address using ownerOf public contract method.
const winnerAddress = Jackpot.methods.ownerOf(winnerTicket);
//e.g. winnerAddress = 0xZ87655A4E9AEbc30D50c7523b2429804ba0987a2;


//Sending all accumulated balance in the contract to the winner using
payToWinner private contract method.
Jackpot.methods.payToWinner(winnerAddress);

/* HOURPOT & DAYPOT REWARDING */
```

WeekPot & MonthPot

WeekPot and MonthPot use the same mechanism. The only difference is that WeekPot rounds restart every week**,** while MonthPot rounds restart every month.



6 factors are taken into consideration when determining the winner:

1- Total Supply

It is one of the public methods of Bitpotz smart contracts. The totalSupply method always returns the last minted Ticket ID value and increases by 1 each time a Ticket is minted.

```
/* WEEKPOT & MONTHPOT TOTAL SUPPLY */

const totalSupply = Jackpot.methods.totalSupply();
//e.g. totalSupply= 1211;

/* WEEKPOT & MONTHPOT TOTAL SUPPLY */
```

2- Lucky Tickets

Lucky Tickets represent **5 random Tickets** that are minted from the relevant Jackpot. Lucky Tickets do not have to be minted in the current round, **any Ticket** that is minted from the relevant Jackpot **at any time** is a candidate to become a Lucky Ticket.

Lucky Tickets are announced when the relevant Jackpot round reaches its **halfway point**. For example, the announcement time for Lucky Tickets in the MonthPot is the 15th of the month. Tickets minted **until the announcement time** can become Lucky Tickets. The count of announced Lucky Tickets is 5 by default, it is changeable and announced clearly.

For announcing Lucky Tickets, there must be at least 1 minted Ticket in the contract. If there are 5 or more minted Tickets, a total of 5 random Lucky Tickets are announced, otherwise random 1 Lucky Ticket is announced.

Below is a simplified code for announcing Lucky Tickets:

```
/* WEEKPOT & MONTHPOT LUCKY TICKETS ANNOUNCEMENT */

//Getting Jackpot Total Supply at the announcement time, using totalSupply
public contract method.
//All Tickets (in round or not) minted from Jackpot until the time of
announcement are included.
const totalSupply = Jackpot.methods.totalSupply();
//e.g. totalSupply= 1211;

//Function for randomly selecting Lucky Tickets at a certain amount.
function luckyTickets(luckyTicketsCount, totalSupply) {
  const numbers = new Set();
  while (numbers.size < luckyTicketsCount) {
    const randomNumber = Math.floor(Math.random() * totalSupply) + 1;
    numbers.add(randomNumber);
  }
  return Array.from(numbers);
}

if (totalSupply > 0) {
  let luckyTicketsCount = 1;
  if (totalSupply >= 5) luckyTicketsCount = 5;
  const announcedLuckyTickets = luckyTickets(luckyTicketsCount, totalSupply);

  //The final Lucky Tickets look like below.
  announcedLuckyTickets = [48, 776, 8712, 15, 56];

}

/* WEEKPOT & MONTHPOT LUCKY TICKETS ANNOUNCEMENT */
```

3- Start Ticket

Before each round starts, a constant called Start Ticket stores the last minted Ticket ID. It uses the totalSupply method in the smart contract.

```
/* WEEKPOT & MONTH START TICKET */

//Calculated and stored before the round starts using totalSupply public
contract method.
const startTicket = Jackpot.methods.totalSupply();
//e.g. startTicket = 132;

/* WEEKPOT & MONTH START TICKET */
```

4- End Ticket

End Ticket represents the last minted Ticket ID value when the round ends. It uses the totalSupply method in the smart contract.

```
/* WEEKPOT & MONTHPOT END TICKET */

//Calculated at the round ending using totalSupply public contract method.
const endTicket = Jackpot.methods.totalSupply();
//e.g. endTicket = 798;

/* WEEKPOT & MONTHPOT END TICKET */
```

5- Is Winner Lucky

Is Winner Lucky constant is created when the round ends. It is a boolean value that determines whether the winner will be one of the Lucky Tickets or not.

The default probability of Lucky Tickets winning is 25%. Probabilities are changeable and announced clearly.

Below is a simplified code for determining Is Winner Lucky:

```
/* WEEKPOT & MONTHPOT IS WINNER LUCKY */

const bet = Math.random();
//e.g. bet = 0.15528324740678534;

//25% probability for lucky winner.
const isWinnerLucky = bet < 0.25;

/* WEEKPOT & MONTHPOT IS WINNER LUCKY */
```

6- Round Ending & Rewarding

When the round ends, a random Winner Ticket is selected, taking into account whether the winner is a Lucky Ticket or not. The owner of the selected Winner Ticket will be the Winner Address.

If the winner is a Lucky Ticket, the Winner Ticket will be one of the previously announced Lucky Tickets. Otherwise, it will be one of the Start Ticket and End Ticket ranges (Start Ticket is not included). And the owner of the selected Winner Ticket will be the Winner Address.

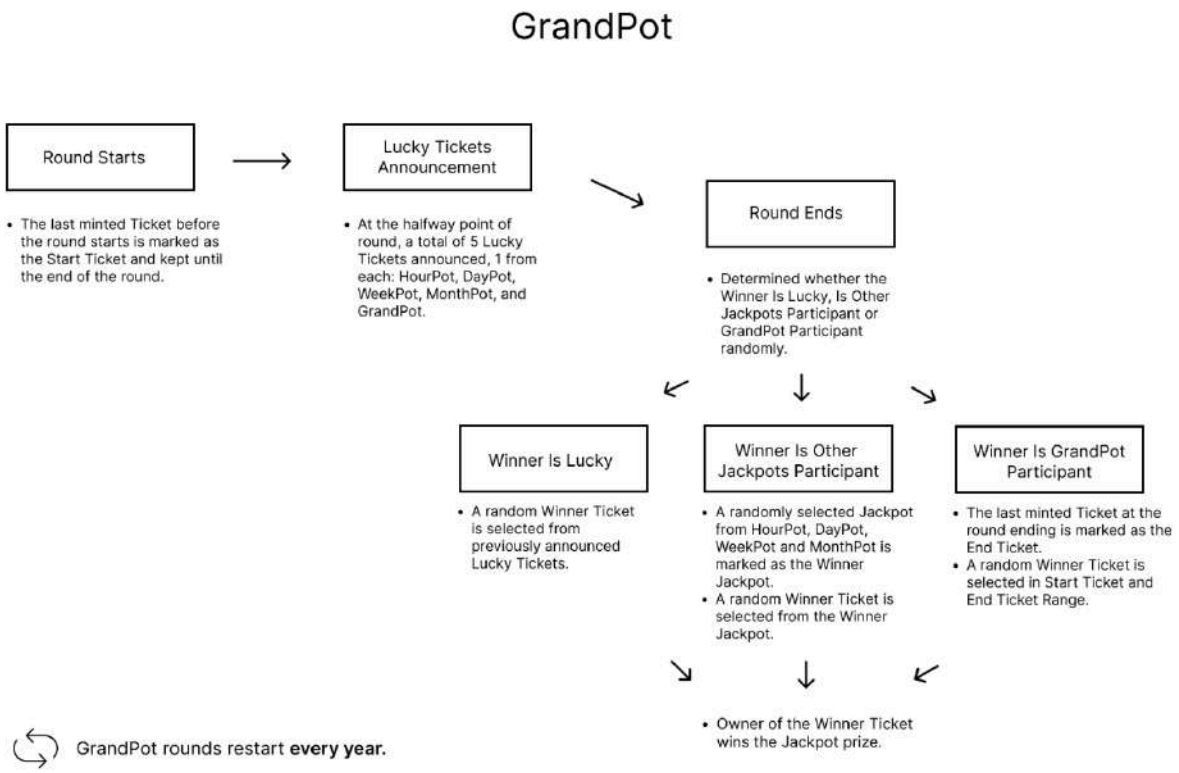
Below is a simplified code for random winner selection considering the Is Winner Lucky parameter:

```
/* WEEKPOT & MONTHPOT REWARDING */

const luckyWinnerBet = Math.random();
//e.g. luckyWinnerBet = 0.15528324740678534;

//25% probability for lucky winner
const isWinnerLucky = luckyWinnerBet < 0.25;

let winnerTicket;
let winnerAddress;

//Winner Is Lucky Case
if (isWinnerLucky) {
  //Previously announced Lucky Tickets.
  announcedLuckyTickets = [48, 776, 8712, 15, 56];

  //Function for randomly selecting one of the Lucky Tickets.
  function randomLuckyTicket(luckyTickets) {
    const randomIndex = Math.floor(Math.random() * luckyTickets.length);
    return luckyTickets[randomIndex];
  }

  //Determining Winner Ticket.
  winnerTicket = randomLuckyTicket(announcedLuckyTickets);
  //e.g. winnerTicket = 15;

  //Determining Winner Address using ownerOf public contract method.
  winnerAddress = Jackpot.methods.ownerOf(winnerTicket);
  //e.g. winnerAddress = 0xZ87655A4E9AEbc30D50c7523b2429804ba0987a2;

  //Winner Is Not Lucky Case
} else {
 //Calculated and stored before the round starts using totalSupply public
 contract method.
 const startTicket = Jackpot.methods.totalSupply();
 //e.g. startTicket = 132;

 //Calculated at the round ending using totalSupply public contract method.
 const endTicket = Jackpot.methods.totalSupply();
 //e.g. endTicket = 798;

 //Function for randomly selecting one of startTicket and endTicket ranges
 except Lucky Tickets.
 function randomTicketExceptLuckyTickets(
   startTicket,
   endTicket,
   luckyTickets
 ) {
```

```
const luckyTicketsSet = new Set(luckyTickets);
   let randomWinner;

   //repeat variable is used for the scenario where there are no participants
 other than Lucky Tickets.
   //in this scenario the winner will also be a Lucky Ticket.
   let repeat = 0;

   do {
     let totalTicketAmount = endTicket - startTicket;
     const randomMargin = Math.floor(Math.random() * totalTicketAmount) + 1;

     //startTicket is not included.
     randomWinner = startTicket + randomMargin;
     repeat++;
   } while (luckyTicketsSet.has(randomWinner) && repeat <= 5);

   return randomWinner;
 }

 //Determining Winner Ticket.
 winnerTicket = randomTicketExceptLuckyTickets(
   startTicket,
   endTicket,
   luckyTickets
 );
 //e.g. winnerTicket = 465;

 //Determining Winner Address using ownerOf public contract method.
 winnerAddress = Jackpot.methods.ownerOf(winnerTicket);
 //e.g. winnerAddress = 0xZ87655A4E9AEbc30D50c7523b2429804ba0987a2;
}

//Sending all accumulated balance in the contract to the winner using
payToWinner private contract method.
Jackpot.methods.payToWinner(winnerAddress);

/* WEEKPOT & MONTHPOT REWARDING */
```

GrandPot

Unlike other Jackpots, the winner of the GrandPot does not have to be a GrandPot participant. HourPot, DayPot, WeekPot and MonthPot participants who minted a Ticket in the same currency at any time also have a chance to win the GrandPot prize.



10 factors are taken into consideration when determining the winner:

1- Total Supply

It is one of the public methods of Bitpotz smart contracts. The totalSupply method always returns the last minted Ticket ID value and increases by 1 each time a Ticket is minted.

```
/* GRANDPOT TOTAL SUPPLY */

const totalSupply = Jackpot.methods.totalSupply();
//e.g. totalSupply= 1211;

/* GRANDPOT TOTAL SUPPLY */
```

2- Lucky Tickets

GrandPot Lucky Tickets are announced by randomly selecting one Lucky Ticket from each Jackpot in the same currency. Lucky Tickets **do not** have to be minted in the current round, **any Ticket that is minted from any Jackpot in the same currency at any time** is a candidate to become a GrandPot Lucky Ticket.

Lucky Tickets are announced when the jackpot round reaches **halfway point**. The announcement time for Lucky Tickets for GrandPot is the 6th month of the year. Tickets that are minted **until the announcement time** can be Lucky Tickets.

GrandPot Lucky Tickets are announced as **minimum 0** and **maximum 5**. The number of Lucky Tickets here is related to the number of participants in each Jackpot in the same currency. If there is at least 1 participant in each Jackpot, it means that a total of 5 Lucky Tickets will be announced. The maximum count of announced Lucky Tickets is 5 by default, it is changeable and announced clearly.

Below is a simplified code for announcing Grandpot Lucky Tickets:

```
/* GRANDPOT LUCKY TICKETS ANNOUNCEMENT */

//Function for randomly selecting Lucky Tickets at a certain amount. (In
GrandPot, luckyTicketsCount = 1 for each Jackpot.)
function luckyTickets(luckyTicketsCount, totalSupply) {
 const numbers = new Set();
 while (numbers.size < count) {
   const randomNumber = Math.floor(Math.random() * totalSupply) + 1;
   numbers.add(randomNumber);
 }
 return Array.from(numbers);
}
const announcedLuckyTickets = [];

//Getting HourPot Total Supply in the same currency at the announcement time,
using totalSupply public contract method.
//All Tickets (in round or not) minted from HourPot until the time of
announcement are included.
const hourpotTotalSupply = HourPot.methods.totalSupply();
//e.g. hourpotTotalSupply = 135;

if (hourpotTotalSupply > 0)
 announcedLuckyTickets.push({
   contract: HourPot,
   NftID: luckyTickets(1, hourpotTotalSupply)[0],
 });

//Getting DayPot Total Supply in the same currency at the announcement time,
using totalSupply public contract method.
//All Tickets (in round or not) minted from DayPot until the time of
announcement are included.
const daypotTotalSupply = DayPot.methods.totalSupply();
//e.g. daypotTotalSupply = 246;

if (daypotTotalSupply > 0)
 announcedLuckyTickets.push({
   contract: DayPot,
   NftID: luckyTickets(1, daypotTotalSupply)[0],
 });
```

```
//Getting WeekPot Total Supply in the same currency at the announcement time,
using totalSupply public contract method.
//All Tickets(in round or not) minted from WeekPot until the time of
announcement are included.
const weekpotTotalSupply = WeekPot.methods.totalSupply();
//e.g. weekpotTotalSupply = 357;

if (weekpotTotalSupply > 0)
 announcedLuckyTickets.push({
   contract: WeekPot,
   NftID: luckyTickets(1, weekpotTotalSupply)[0],
 });

//Getting MonthPot Total Supply in the same currency at the announcement
time, using totalSupply public contract method.
//All Tickets (in round or not) minted from MonthPot until the time of
announcement are included.
const monthpotTotalSupply = MonthPot.methods.totalSupply();
//e.g. monthpotTotalSupply = 468;

if (monthpotTotalSupply > 0)
 announcedLuckyTickets.push({
   contract: MonthPot,
   NftID: luckyTickets(1, monthpotTotalSupply)[0],
 });

//Getting GrandPot Total Supply in the same currency at the announcement
time, using totalSupply public contract method.
//All Tickets (in round or not) minted from GrandPot until the time of
announcement are included.
const grandpotTotalSupply = GrandPot.methods.totalSupply();
//e.g. grandpotTotalSupply = 579;

if (grandpotTotalSupply > 0)
 announcedLuckyTickets.push({
   contract: GrandPot,
   NftID: luckyTickets(1, grandpotTotalSupply)[0],
 });

//The final Lucky Tickets looks like below.
announcedLuckyTickets = [
 { contract: HourPot, NftID: 12 },
 { contract: DayPot, NftID: 45 },
 { contract: WeekPot, NftID: 156 },
 { contract: MonthPot, NftID: 267},
 { contract: GrandPot, NftID: 378},
];

/* GRANDPOT LUCKY TICKETS ANNOUNCEMENT */
```

4- Others

Others represent other Jackpots **in the same currency** other than the GrandPot. (HourPot, DayPot, WeekPot and MonthPot.) Used to select a random winner from Jackpots other than the GrandPot when the round ends.

```
/* GRANDPOT OTHERS */

//All Jackpots except GrandPot in the same currency.
const others = [HourPot, DayPot, WeekPot, MonthPot];

/* GRANDPOT OTHERS */
```

5- Start Ticket

Before each round starts, a constant called Start Ticket stores the last minted Ticket ID. It uses the totalSupply method in the smart contract.

```
/* GRANDPOT START TICKET */

//Calculated and stored before the round starts using totalSupply public
contract method.
const startTicket = GrandPot.methods.totalSupply();
//e.g. startTicket = 132;

/* GRANDPOT START TICKET */
```

6- End Ticket

End Ticket represents the last minted Ticket ID value when the round ends. It uses the totalSupply method in the smart contract.

```
/* GRANDPOT END TICKET */

//Calculated at the round ending using totalSupply public contract method.
const endTicket = GrandPot.methods.totalSupply();
//e.g. endTicket = 798;

/* GRANDPOT END TICKET */
```

7- Is Winner Lucky

Is Winner Lucky constant is created when the round ends. It is a boolean value that determines whether the winner will be one of the Lucky Tickets or not.

The default probability of Lucky Tickets winning is 25%. Probabilities are changeable and announced clearly.

Below is a simplified code for determining Is Winner Lucky:

```
/* GRANDPOT IS WINNER LUCKY */

const bet = Math.random();
//e.g. bet = 0.15528324740678534;

//25% probability for lucky winner.
const isWinnerLucky = bet < 0.25;

/* GRANDPOT IS WINNER LUCKY */
```

8- Is Winner Others

Is Winner Others constant is created when the round ends. It is a boolean value that determines whether the winner will be one of the Others or not.

The default probability of Others winning is 25%. Probabilities are changeable and announced clearly.

Below is a simplified code for determining Is Winner Others:

```
/* GRANDPOT IS WINNER OTHERS */

const bet = Math.random();
//e.g. bet = 0.32528324740678534;

//25% probability for other winner.
const isWinnerOthers = bet >= 0.25 && bet < 0.5;

/* GRANDPOT IS WINNER OTHERS */
```

9- Is Winner Self

Self represents the GrandPot current round participants. Is Winner Self constant is created when the round ends. It is a boolean value that determines whether the winner will be one of Self or not.

The default probability of Self winning is 50%. Probabilities are changeable and announced clearly.

Below is a simplified code for determining Is Winner Self:

```
/* GRANDPOT IS WINNER SELF */

const bet = Math.random();
```

```
//e.g. bet = 0.85528324740678534;

//50% probability for self winner.
const isWinnerSelf = bet >= 0.5;

/* GRANDPOT IS WINNER SELF */
```

10- Round Ending & Rewarding

When the round ends, a Winner Jackpot and a Winner Ticket is randomly selected taking into account whether the winner is a Lucky Ticket, an Others Ticket, or a Self Ticket. The owner of the selected Winner Ticket will be the Winner Address.

If the winner is a Lucky Ticket:

-Winner Jackpot will be the contract value of a randomly selected ticket from Previously Announced Lucky Tickets.

-Winner Ticket will be the NftID value of the Winner Jackpot.

-Winner Address will be the owner of the Winner Ticket.

If the winner is an Others Ticket:

-Winner Jackpot will be a randomly selected contract from HourPot, DayPot, WeekPot and MonthPot.

-Winner Ticket will be a random Ticket ID minted from the Winner Jackpot.

-Winner Address will be the owner of the Winner Ticket.

If the winner is a Self Ticket:

-Winner Jackpot will be GrandPot.

-Winner Ticket will be random one of the Start Ticket and End Ticket ranges (Start Ticket is not included).

-Winner Address will be the owner of the Winner Ticket.

Below is a simplified code for random winner selection considering parameters: Is Winner Others, Is Winner Lucky and Is Winner Self:

```
/* GRANDPOT REWARDING */

const bet = Math.random();
//e.g. bet = 0.15528324740678534;

//25% probability for lucky winner
const isWinnerLucky = bet < 0.25;
```

```
//25% probability for other jackpots participants
const isWinnerOthers = bet >= 0.25 && bet < 0.5;

//50% probability for grandpot participants
const isWinnerSelf = bet >= 0.5;

let winnerJackpot;
let winnerTicket;
let winnerAddress;

//Winner Is Lucky Case
if (isWinnerLucky) {
 //previously announced Lucky Ticket objects.
 announcedLuckyTickets = [
   { contract: HourPot, NftID: 1237 },
   { contract: DayPot, NftID: 66 },
   { contract: WeekPot, NftID: 837 },
   { contract: MonthPot, NftID: 9821 },
   { contract: GrandPot, NftID: 14 },
 ];

 //Randomly selecting one of the Lucky Ticket objects.
 const randomLuckyJackpotIndex = Math.floor(
   Math.random() * announcedLuckyTickets.length
 );
 //e.g. randomLuckyJackpotIndex = 2;

 const luckyJackpotObject = announcedLuckyTickets[randomLuckyJackpotIndex];
 //e.g. luckyJackpotObject= { contract: HourPot, NftID: 1237 };

 //Determining Winner Jackpot.
 winnerJackpot = luckyJackpotObject.contract;
 //e.g. winnerJackpot = HourPot;

 //Determining Winner Ticket.
 winnerTicket = luckyJackpotObject.NftID;
 //e.g. winnerTicket = 1237;


 //Determining Winner Address using ownerOf public contract method.
 winnerAddress = winnerJackpot.ownerOf(winnerTicket);
 //e.g. winnerAddress = 0xZ87655A4E9AEbc30D50c7523b2429804ba0987a2;

 //Winner Is Others Case
} else if (isWinnerOthers) {
 //All jackpots except GrandPot.
 const otherJackpots = [HourPot, DayPot, WeekPot, MonthPot];

 //Randomly selecting one of Others.
 const randomOthersJackpotIndex = Math.floor(
   Math.random() * otherJackpots.length
```

```
);
 //e.g. randomOthersJackpotIndex = 1;

 //Determining Winner Jackpot.
 winnerJackpot = otherJackpots[randomOthersJackpotIndex];
 //e.g. winnerJackpot = DayPot;

 //Getting Winner Jackpot Total Supply using totalSupply public contract
method.
 const winnerJackpotTotalSupply = winnerJackpot.methods.totalSupply();
 //e.g. winnerJackpotTotalSupply = 246;

//Function for randomly selecting one of startTicket and endTicket ranges
function randomTicket(startTicket, endTicket) {
   const totalTicketAmount = endTicket - startTicket;
   const randomMargin = Math.floor(Math.random() * totalTicketAmount) + 1;

   //startTicket is not included.
   return startTicket + randomMargin;
 }

 //Determining Winner Ticket. (All NFTs minted from WinnerJackpot are
included.)
 winnerTicket = randomTicket(1, winnerJackpotTotalSupply);
 //e.g. winnerTicket = 151;

 //Determining Winner Address using ownerOf public contract method.
 winnerAddress = winnerJackpot.ownerOf(winnerTicket);
 //e.g. winnerAddress = 0xZ87655A4E9AEbc30D50c7523b2429804ba0987a2;

 //Winner Is Self Case
} else if (isWinnerSelf) {
 //Determining Winner Jackpot.
 winnerJackpot = GrandPot;

 //Calculated and stored before the round starts using totalSupply public
contract method.
 const startTicket = GrandPot.methods.totalSupply();
 //e.g. startTicket = 132;

 //Calculated at the round ending using totalSupply public contract method.
 const endTicket = GrandPot.methods.totalSupply();
 //e.g. endTicket = 798;

 //Function for randomly selecting one of startTicket and endTicket ranges.
 function randomTicket(startTicket, endTicket) {
   const totalTicketAmount = endTicket - startTicket;
   const randomMargin = Math.floor(Math.random() * totalTicketAmount) + 1;

   //startTicket is not included.
   return startTicket + randomMargin;
```

```
}

 //Determining Winner Ticket. (in round NFTs minted from GrandPot are
included.)
 winnerTicket = randomTicket(startTicket, endTicket);
 //e.g. winnerTicket = 545;

 //Determining Winner Address using ownerOf public contract method.
 winnerAddress = winnerJackpot.ownerOf(winnerTicket);
 //e.g. winnerAddress = 0xZ87655A4E9AEbc30D50c7523b2429804ba0987a2;
}

//Sending all accumulated balance in the contract to the winner using
payToWinner private contract method.
GrandPot.methods.payToWinner(winnerAddress);

/* GRANDPOT REWARDING */
```

The Claim of "Never Wasted NFTs"

This situation can be likened to a lottery ticket in the real world that is valid for a **lifetime** and has the potential to win Jackpots **repeatedly**. This claim is based on two things:

1- Lucky Tickets Context

Jackpots that have the feature of Lucky Tickets (WeekPot, MonthPot, and GrandPot) include all tickets minted up to the announcement time from the relevant contracts when determining the Lucky Tickets. **This means any ticket minted from these 3 jackpots at any time can win a Jackpot at any time**.

To understand the logic behind Lucky Tickets, review the Lucky Tickets & Is Winner Lucky titles mentioned above.

Examples of Never Wasted NFTs in the context of Lucky Tickets:

Example 1:

-A participant minted a BNB MonthPot Ticket with ID 135, on June 13, 2025.

-BNB MonthPot announced the Tickets with IDs [15,67,135,726,44] as Lucky Tickets on September 15, 2030.

-On October 1, 2030, the BNB MonthPot round ended and resulted in Winner Is Lucky, Winner Ticket = 135.

-MonthPot Ticket with ID 135 won a jackpot about 6 years after it was minted.

Example 2:

-A participant minted a BNB HourPot Ticket with ID 246, on November 12, 2025.

-BNB GrandPot announced [{ contract: HourPot, NftID: 246 },…others] as Lucky Tickets on June 15, 2030.

-On January 1, 2031, the BNB GrandPot round ended and resulted in Winner Is Lucky, Winner Jackpot = HourPot, Winner Ticket = 246.

-HourPot Ticket with ID 246 won a Jackpot about 6 years after it was minted.

2- GrandPot Context

When a GrandPot round ends, in case of “Winner Is Others”, all Tickets minted from the Winner Jackpot are included when determining the Winner Ticket. **This means any ticket minted from any jackpot at any time can win a GrandPot prize at any time.**

To understand the logic behind GrandPot Others, review the Others & Is Winner Others titles mentioned above.

Example:

-A participant minted a BNB WeekPot Ticket with ID 357, on July 15, 2025.

-On January 1, 2031, the BNB GrandPot round ended and resulted in Winner Is Others, Winner Jackpot = WeekPot, Winner Ticket = 357.

-WeekPot Ticket with ID 357 won a jackpot about 6 years after it was minted.

Trading Bitpotz Tickets

Each Bitpotz Ticket is an NFT, and these NFTs can be sold and traded on the blockchain or on NFT marketplaces. So, why would you buy someone else's Ticket instead of minting a new one?

Bitpotz continues to add various features to make ticket trading advantageous, but for now, the most attractive reason for buying someone else’s Ticket is the Lucky Tickets feature.

To understand the logic behind Lucky Tickets, review the Lucky Tickets & Is Winner Lucky titles mentioned above.

An example of the advantage of buying a Lucky Ticket:

-Person X minted a BNB MonthPot Ticket with ID 135 for $5 on June 1, 2025.

-BNB MonthPot announced the Tickets with IDs [15,67,135,726,44] as Lucky Tickets on June 15, 2025.

-Person X saw that the BNB MonthPot prize had risen to $6000 on June 25, 2025. And Person X knows that this Ticket had a high chance of winning the MonthPot prize, so Person X put it up for sale at a price of $50.

-Person Y saw the Ticket ID 135 for sale on June 29, 2025 and purchased it to gain a high chance of winning before the MonthPot round ended.

-On July 1, 2025, BNB MonthPot round ended and resulted in Winner Is Lucky, Winner Ticket = 135.

-Person X made a profit of $45 without taking any risks, Person Y paid a higher amount than the regular Ticket price to buy the risk and won the MonthPot Prize.

$POTZ Token

The $POTZ token is the official utility token of the Bitpotz platform. The token's use cases both within and outside the platform will continue to be diversified by Bitpotz.

The most important current use case is the **Vip** feature. The Vip feature is initially valid only for BNB Jackpots. The Vip conditions are changeable and announced clearly.

To understand the logic behind Vip, review the Vip ($POTZ Holders) section mentioned above.

Tokenomics is announced on the website.