

**Configuration Manual** 

**MSc Research Project** 

FinTech

# TING YI LIU Student ID: X21177899

**School of Computing** 

National College of Ireland Supervisor: Theo Mendonca

### National College of Ireland MSc Project Submission Sheet School of Computing



Student Name:	TING YI LIU							
Student ID:	X21177899							
Programme:	MSC FINTECH	MSC FINTECH <b>Year:</b> 2022/23						
Module:	MSC Research Project							
Supervisor:	Theo Mendonca							
Submission Due Date:	August 14, 2023							
Project Title:	Innovative User Incentive Models in Blockchain-Based Deckles Bike Sharing System							
Word Count:	1757	Page Count 15						

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

<u>ALL</u> internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	TING YI LIU
Date:	14/08/2023

### PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

## **Configuration Manual**

# Ting Yi Liu x21177899

### 1. Introduction

This user configuration manual outlines in detail how to set up and configure the Bike Sharing Smart Contract an Ethereum-based decentralized application (Dapp). Through a reward and penalty structure, the smart contract promises to change the bike sharing sector by motivating responsible user behavior and boosting active involvement.

### 2. System Requirements

### 2.1. Hardware

- MacBook Pro (Retina, 13-inch, Early 2015)
- Mac OS Big Sur version 11.7
- Memory 8 GB 1867 MHz DDR3
- Storage 128 GB SSD

### 2.2. Software

- Install Ganache: Download and install Ganache from the official website.
- Remix IDE: Open your web browser and go to the Remix IDE website (<u>https://remix.ethereum.org/</u>). Remix is a browser-based IDE, so no additional installation is needed.

### 3. Configuration Steps

### 3.1. Step 1 - Launch Ganache

After installation, launch the Ganache program. It will create a local blockchain network with predefined accounts and 100 test Ether each for transactions.

🤿 Ganache	ACCURATE (B) RACKES (P) INSERVATION (B) CONT OWNERS (B) RACKES (P) INSERVATION (B) CONT OWNERS (B) RACKES (B) CONT WARRANCE (B) OUTCIDE DISCOMPTION (B) F (F) CF (F) CF (B) CONT B) CF (B)		Active first force remote an in- ancessary Concentration (Concentration)	14140 3MID	0) 0
WORKSPACES	#200805 8×67A622791F4497513a75A8b55539357FAc292e14	tutetr 100.00 ETH	tootaat B	anaca B	S
	400E9 0+53E37311e345c776EE2cd7Dc8093135a0Ec28358	160400 1885-88 ETH	14.03.07 0	98073 1	8
fortunate-root (ethereum)	0+CFF02213959Ed4bD8Cf52dB614661214F88cbDa5	199.00 ETH	txcoust 0	nau 2	e
	439855 @×6A4096cb0D876F8Ad3486aFC82e4A5385244Ee1E	188.801 189.00 ETN	ткозант В	нин 3	I
	400000 @×5699c573358b58953A198158eb9EBd55F072bF2b	141067 189.00 ETH	TK DOUBT D	1803 4	ð
	400855 6×D9CC6485Eb4e896d668c977DAA58525Cedb78c39	109.00 ETH	teosar Q	980X 5	C
ETHEREUM ETHEREUM	409855 0+A58E3639d4#A#888#A66F588A5D71845D3D8A#F83	1804CE 180.00 ETH	ticoar D	eeso. G	ð

### **3.2.** Step 2 - Deploy the Smart Contract

Compile and deploy the Bike Sharing Smart Contract to the Ganache network using the Remix IDE development environment. Go to the "Deploy & Run Transactions" tab on the left-hand side. Under the "Environment" section, select "Dev - Ganache Provider" as the environment and enter the connection URL for Ganache (http://127.0.0.1:7545).

& RUN TRANSACTIONS $\checkmark$ $\rightarrow$	🕨 🍳 🍳 旈 Home 💈 Bikesharingsystem.sol	🖇 bikesharingmodeLsol 🗙
MENT 🤟 VM (Merge) 💿 i	Ganache Provider	×
● eddC4 (100 ether)	Note: To run Ganache on your system, run: yarn global add ganache ganache	
	- For more info, visit: <u>Ganache Documentation</u>	
	Ganache JSON-RPC Endpoint:	7F9a8Fb2f55e6)
	http://127.0.0.1:8545	
	<b>V</b> 7585	OK Cancel

**Connect Remx to Ganache** 

In Remix IDE, go to the "Solidity Compiler" tab and select the version of Solidity used in your smart contract. Then, navigate to the "Deploy & Run Transactions" tab again. Click "Deploy" to deploy the contract to the Ganache network.

	DEPLOY & RUN TRANSACTIONS	$i \checkmark i$	) (	<b>. e.</b> ĜH	łome	Sikesharingsystem.s	sol 💲 bikesharingmodel.sol 🗙			
<b>~</b>	ENVIRONMENT 👾			// SPDX-L	// SPOX-License-Identifier: MIT					
æ	Dev - Ganache Provider  🥥				addates voites,					
0	Custom (5777) network				sontract BikeChain {					
~	ACCOUNT G			addre	address owner;					
0	0xFdAA607d (100 ether) 💲 🕻				address payable companyWallet; // Company (8) wallet address					
	GAS LIMIT			const	ructor()					
⇒ ∣		10	10 owner = msg.sender;							
	3000000	11	1	companywallet = payable 0x903A/6145/T/389832F/0009T/B802166035A3A01; // Replace this wi						
ж.										
	Theore				d yoursel					
123	0 Ether									
					t Renter					
	CONTRACT				ddress pa	yable walletAddress	s;			
	BikeChain - bikesharingmodel.sol				tring fir	stName;				
					tring las	tName;				
	evm version: london				ool canRe					
	Destau				ool activ	e;				
	Depidy				int balan	ce;				
	Publish to IPFS				int due;	Times				
					une start	LTHC'				

**Deploy the smart contract** 

### 3.3. Step 3 - Interact with the Smart Contract

Users can start interacting with the smart contract using MetaMask to complete transactions and run functions once it has been deployed. Through the smart contract's functionalities, users may do operations such as user registration, starting and ending rentals, making payments, and redeeming rewards.

#### 3.3.1.Add a Renter

To add a new renter, fill in the required parameters (walletAddress, firstName, lastName, canRent, active, balance, due, startTime, endTime) and click on the function button.

	DEPLOY & RU	JN TRANSACTIONS $\checkmark$ $\rightarrow$	) e	د الله Home ک Bikesharingsystem.sol ک bikesharingmodeLsol 🗙			
- 41				// SPOX-License-Identifier: MIT pragma solidity ^0.8.15;			
	addRenter			contract BikeChain {			
Q		0xFdA32C985d330BfBE93		address owner;			
- <b>-</b>		Ting Yi		address payable companyWallet; // Company (8) wallet address			
۵				<pre>constructor() {     owner = msg.sender;</pre>			
Ű.		true		<pre>companyWallet = payable(0x903A761457f7389832F70609f76882166035A3A0); // Replace this with }</pre>			
310		false					
MB				struct Renter {			
				address payable walletAddress; string firstName;			
				string lastName; bool canRent;			
		9		bol active; uint balance;			
		0 transact		uint due; wint startTime; uint endTime:			
				Listen on all transactions Q Search with transaction hash or address			
				ation of BikeChain pending			
4	redeemRewa.	address walletAddres: 🗸	0	[block:1 tx1ndex:0] from: 0xFdAA607d to: BikeChain.(constructor) value: 0 wei dete: 0x600 _ f0031 loge: 0 kueb: 0x531 _ 2101a			
2	startRental	address walletAddres: 🗸					
			>				

Add a Renter

You will get the renter's information stored in the renters mapping. This includes the firstName, lastName, canRent, active, balance, due, startTime, endTime, penalty, and the Rewards struct associated with that walletAddress. The renter's name is "Ting Yi Liu," and they have the ability to rent a bike (canRent: true). However, currently, they are not actively renting a bike (active: false) and have no pending balance (due: 0). There are no previous rental records, so the startTime, endTime, and penalty are all set to 0. Additionally, the renter has not completed any rides or earned any rewards yet (rewards: rideCount: 0, rideDistance: 0, leaderboardRewards: 0, leaderboardPositions: [false, false, false, false]).

	DEPLOY & RUN TRANSACTIONS	<b>~</b> >	🕨 ର୍	<b>ବ,</b> ଲି Home	S Bikesharingsystem.sol	💲 bikesharingmodel.sol 🗙
Cen	getTotalDura address walletAddres:			contract BikeCha	in {	
Q	leaderboard			address owne address paya	r; ble companyWallet; // C	
	monthlyRide address , uint256			constructor(	) { msg.sender;	
	penaltyPerVi_			companyW }	allet = payable(0x903A7	
*	renters 0xFdA32C985d330Bf					
₩ 500	0: address: walletAddress 0xFdA32C98 5d330BfBE93e66f88cFae6A734FA60 7d 1: string: firstName Tion Vi			struct Rente address	r { payable walletAddress;	
	2: string: lastName Liu 3: bool: canRent true			string f string l bool can	irstName; astName; Rent;	
	4: bool: active false 5: uint256: balance 0 6: uint256: due 0			bool act uint bal uint due	ive; ance; ;;	
	7: uint256: startTime 0 8: uint256: endTime 0 9: uint256: penalty 0			uint sta uint end uint pen	rtTime; Time; Walty;	
	<ol> <li>tuple{uint256,uint256,uint256,bool</li> <li>bool): rewards 0,0,0,false,false,fal se,false</li> </ol>			listen on all tr	ansactions Q Sear	rch with transaction hash or address
	rewardPerDi			to BikeChain.ren	ters	J MASA   UXJ0700440
¥	rewardPerRi		CAL L	[call] from: 0x data: 0x644a	rdA32C985d330BfBE93e66f 607d	88cFae6A734FA607d to: BikeChain.renters(address)
	rewardsOf address walletAddress	v	U C		5	

-

#### **Retrieve the details**

#### **3.3.2.Start Renting a Bike**

To start renting a bike, provide the renter's wallet address as a parameter in the "startRental" function and click the function button.



#### Start Renting a Bike

After start a rental. we can check the information. The renter currently cannot rent a bike (canRent: false). They have an ongoing rental session (active: true) that started at startTime: 1691057043 (Unix timestamp). The renter has no pending balance (due: 0), no penalty (penalty: 0), and has not completed any rides or earned rewards yet (rewards: rideCount: 0, rideDistance: 0, leaderboardRewards: 0, leaderboardPositions: [false, false, false]).



"Renters" information after start renting a bike

#### 3.3.3.Deposit

The deposit function allows users (renters) to add funds to their account balance in the smart contract by sending Ether with a transaction. The deposited amount is added to the user's existing balance, which can be used to cover pending dues or future bike rentals. As shown below, we added 5 ETH to deposit.



Deposit

The balanceOfRenter function allows the smart contract to retrieve and display the balance of a specific renter's wallet address within the bike sharing system. And the balanceOfCompany function retrieves the balance of the company's wallet address in the smart contract. As figure below, we can see balance of renter is 5 ETH and balance of company is 100 ETH.

	DEPLOY & RUN TRANSACTION	<b>\S ✓</b> →	🕨 ପ୍ର୍	🛱 Home 💈 Bikesharingsyste	m.sol 💲 bikesharingmodel.sol 🗙						
	BIKECHAIN AT UXFA1AABU7			address owner;							
മ											
~											
Q	addRenter address walletAdd		10	owner = msg.sender;	9834761457f7389832F78689f788821668354348) • // Benlace this with						
	deposit 0xFdA32C985d33	DBf 🗸	12	<pre>}</pre>							
0											
	makePayment address walletAdd										
⇒				<pre>struct Renter {     address payable walletAddress;</pre>							
يف	redeemRewa address walletAdd										
*	startRental 0vEdA32C985d33	ner 🗸		string firstName; string lastName; bool canRent:							
-											
1420	stopRental address walletAdd			bool active;							
				uint balance;							
	balanceOf			uint startTime:							
				uint endTime;							
	balanceOfCo 0x903A761457f73	B9 🗸		uint penalty;							
	A. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.			Rewards rewards;							
	0: UIR256: 10000000000000000000										
	balanceOfRe address walletAdd			struct Rewards {							
	can Pont Bito			listen on all transactions Q							
				ita: Ux/22/1317 RikeChain balanceOfCompany							
	getLeaderbo			bikechain, ba tanceor company							
			CAL [G	all: from: 0xFdA32C985d330BfBE9	3e66f88cFae6A734FA607d to: BikeChain.balanceOfCompany(address)						
×	getTotalDura address walletAdd		L da	ta: 0xa8b5a3a0							

#### Balance

In Ganache, we can see the account balance dropped 5 ETH.

ACCOUNTS (B) BLOCKS (C) TRANSACTIONS (B) CONTRA	ACTS () EVENTS () LO	IGS SEARCH FOR BLOCK NUMBERS OR TX HASHES Q
CURRENT BLOCK GAS PRICE GAS LBUT HARDFORK NETWORK ID RPC S	ERVER MINING STATUS	WORKSPACE
4 20000000000 6721975 MULIRGLACIER 5777 HTT	P://127.0.0.1:7545 AUTOMINING	QUICKSTART SAVE SWITCH
MNEMONIC 👩 bundle soul satisfy orbit dinosaur during egg clock skull	battle mean morning	HD PATH m/44'/60'/0'/0/account_index
ADDRESS	BALANCE	TX COUNT INDEX
Ø×FdA32C985d330BfBE93e66f88cFae6A734FA607d	94.93 ETH	4 0
ADDRESS	BALANCE	TX COUNT INDEX
0×903A761457f73B9832F70609f7BB02166035A3A0	100.00 ETH	0 1
ADDRESS	BALANCE	TX COUNT INDEX
0×2D427E6E159F9718df82a14B077b2912139395b0	100.00 ETH	0 2
ADDRESS	BALANCE	TX COUNT INDEX
0×780262A878b22D4cc0959b57afB6bEa851E2da6E	100.00 ETH	0 3
ADDRESS	BALANCE	TX COUNT INDEX
0×496a5A7Fd41447Aa8815CCE4ec7676d5E76fb32A	100.00 ETH	0 4 CP
ADDRESS	BALANCE	TX COUNT INDEX
0×afEfD997Dc4D8753F4F86787cB19A5037FBc1cdB	100.00 ETH	0 5 CP
ADDRESS 0×0c8B0a2E2091FD4700687d8f59A4Fb28Ae6c9e3b	BALANCE 100.00 ETH	TX COUNT INDEX

### Ganache account balance

	NTS BL	.оскя 🧲	TRANSACTION		CONTRACTS			SEARCH FOR BLOCK NUM	BERS OR TX H	ashes Q
CURRENT BLOCK 4	GAS PRICE 20000000000	GAS LIMIT 6721975	HARDFORK	NETWORK ID 5777	RPC SERVER HTTP://127.0	0.0.1:7545 AU	ING STATUS TOMINING	WORKSPACE QUICKSTART	SAVE	SWITCH
TX HASH <b>0×3eef21</b> FROM ADDRESS 0×FdA32C985	4 <b>6a3bb164</b> 5d330BfBE93e6	<b>0861cb46</b> 6f88cFae6A7	<b>be2c51290f</b> 34FA607d	<b>b8f2737</b> TO CONTRA 0×fA173	<b>20d8650a(</b> CT ADDRESS 0f906EF01280	0 <b>1c725f3db</b> 0332193FA6DaA	9 <b>fe47f1</b> e677ffAaBC7	GAS USED 43089	VALUE 50000000	CONTRACT CALL
TX HASH         CONTRACT         CALL           0×03f83634370652c7cac533f84fd218b73e4e3db7e518d40436f0b49fd798e43d         CONTRACT         CALL           FROM ADDRESS         TO CONTRACT ADDRESS         GAS USED         VALUE           0×FdX32C985d330BfBe93e66f88cFae6A734FA607d         0×fA1730f906EF01280332193FA6DaAe677ffAaBC7         51703         0								CONTRACT CALL		
TX HASH <b>0×0b8049</b> FROM ADDRESS θ×FdA32C985	<b>C8132c0a2</b> 5d330BfBE93e6	<b>5112c358</b> 6f88cFae6A7	<b>871fe3985c</b> 34FA607d	<b>30c3168</b> TO CONTRA 6×fA173	<b>fd9f9f396</b> CT ADDRESS 0f906EF01280	6 <b>905cf6f8b</b> 9332193FA6DaA	e677ffAaBC7	0AS USED 178620	VALUE Ø	CONTRACT CALL
TX HASH <b>0×1b3d3f</b> FROM ADDRESS 0×FdA32C985	<b>73af85f1e</b>	ec6cac80	<b>bcf511cedf</b> 34FA607d	<b>f8917b8</b> CREATED CC 6×fA173	b4ed037d DNTRACT ADDRESS 0f906EF01280	5 <b>ce2644e48</b> 9332193FA6DaA	e677ffAaBC7	GAS USED 3011119	VALUE Ø	CONTRACT CREATION

### Ganache transaction detail

#### **3.3.4.Stop Renting a Bike**

To stop renting a bike, provide the renter's wallet address, the distance traveled, and whether there was a violation as parameters in the "stopRental" function. Then, click the function button.



Stop renting a bike

The renter's name is Ting Yi Liu. Currently, the renter cannot rent a bike (canRent is false) and is not actively using a bike (active is false). The renter's account balance is 5 Ether. The renter has a pending due amount of 0.01 Ether. The startTime indicates that the renter started a bike rental at a specific timestamp, while endTime shows the end time of the rental. The penalty value is currently 0, indicating no penalty has been applied. The rewards tuple shows that the renter has completed 1 ride (rideCount: 1) and covered a total ride distance of 1 kilometer (rideDistance: 1). The renter has earned 0 leaderboard rewards and holds the first position on the leaderboard (leaderboardPositions: [true, false, false]). Additionally, the renter has no violated the rules (violation: false).



"renters" information after stop renting a bike

canRentBike function is false.



canRentBike function

#### Make payment



make payment function

#### after making payment



#### "renters" information after making a payment

The balance of renter and company



**Balance of renter and company** 

ACCOUNTS (B) BLOCKS (C) TRANSACTIONS (C) CONTRACTS (C) EVENTS (C) SEARCH FOR BLOCK MUMBERS OR TX HASHES (C)			
CURRENT ELOCK GAS PRICE GAS LIMIT HARGFORK NETWORK ID RPC: 6 20000000000 6721975 MUIRGLACIER 5777 HTT	SERVER MINING STATUS WORKSPAC AUTOMINING QUICKST	CE SAVE SWITCH	8
MNEMONIC       Image: Control of the source of			
ADDRESS	BALANCE	TX COUNT INDEX	F
0×FdA32C985d330BfBE93e66f88cFae6A734FA607d	94.93 ETH	6 0	
ADDRESS	BALANCE	TX COUNT INDEX	J.
0×903A761457f73B9832F70609f7BB02166035A3A0	100.01 ETH	O 1	
ADDRESS	BALANCE	TX COUNT INDEX	F
0×2D427E6E159F9718df82a14B077b2912139395b0	100.00 ETH	Θ 2	
ADDRESS	BALANCE	TX COUNT INDEX	F
0×780262A878b22D4cc0959b57afB6bEa851E2da6E	100.00 ETH	Θ 3	
ADDRESS	BALANCE	TX COUNT INDEX	F
0×496a5A7Fd41447Aa8815CCE4ec7676d5E76fb32A	100.00 ETH	0 4	
ADDRESS	BALANCE	TX COUNT INDEX	F
0×afEfD997Dc4D8753F4F86787cB19A5037FBc1cdB	100.00 ETH	Θ 5	
ADDRESS	BALANCE	TX COUNT INDEX	P
0×0c8B0a2E2091FD4700687d8f59A4Fb28Ae6c9e3b	100.00 ETH	0 6	

#### Ganache account balance

#### can rent true



#### canRentBike function

### 4. Smart Contract Sodility Code:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.15;
contract BikeChain {
  address owner;
  address payable companyWallet; // Company (B) wallet address
  constructor() {
     owner = msg.sender;
     companyWallet = payable(0x903A761457f73B9832F70609f7BB02166035A3A0); // Replace this with
the actual company wallet address
  }
  // Add yourself as a Renter
  struct Renter {
     address payable walletAddress;
    string firstName;
     string lastName;
    bool canRent;
    bool active;
    uint balance;
     uint due;
     uint startTime:
    uint endTime;
    uint penalty;
     Rewards rewards;
  }
  struct Rewards {
     uint rideCount; // Number of rides
     uint rideDistance; // Total ride distance in kilometers
     uint leaderboardRewards; // Rewards for leaderboard winners
    bool[3] leaderboardPositions; // Keeps track of leaderboard positions for top 3 riders
    bool violation;
  }
  mapping (address => Renter) public renters;
    mapping (address => mapping(uint => uint)) public monthlyRideCount; // Monthly ride count for each
user
  uint256 public leaderboardReward = 5 ether; // Reward for leaderboard winners
  uint256 public rewardPerRideCount = 0.1 ether; // Reward per 10 rides
  uint256 public rewardPerDistance = 0.1 ether; // Reward per 10 kilometers
  uint256 public penaltyPerViolation = 0.5 ether; // Penalty for rule violation
  // Create a new Rewards struct
  function createRewards() private pure returns (Rewards memory) {
     return Rewards(0, 0, 0, [false, false, false], false);
  }
  function addRenter(
     address payable walletAddress,
    string memory firstName,
    string memory lastName,
    bool canRent,
    bool active.
     uint balance,
```

```
uint due
  uint startTime,
  uint endTime
  ) public {
  renters[walletAddress] = Renter(
     walletAddress,
     firstName,
     lastName,
     canRent,
     active,
    balance.
     due.
     startTime,
     endTime,
     0,
     createRewards() // Use the createRewards function to initialize the Rewards struct
     );
}
// Start rent a bike
function startRental(address walletAddress) public {
  require(renters[walletAddress].due == 0, "You have a pending balance.");
  require(renters[walletAddress].canRent == true, "You cannot rent at this time.");
  renters[walletAddress].active = true;
  renters[walletAddress].startTime = block.timestamp;
  renters[walletAddress].canRent = false;
}
// Stop rent a bike
function stopRental(address walletAddress, uint256 distance, bool violation) public {
  require(renters[walletAddress].active == true, "Please start rent a bike first.");
  renters[walletAddress].active = false;
  renters[walletAddress].endTime = block.timestamp;
  setDue(walletAddress);
  updateRewards(walletAddress, distance, violation); // Call updateRewards function when return bike
  updateMonthlyRideCount(walletAddress); // Update monthly ride count
  updateLeaderboard(walletAddress); // Update leaderboard positions
  updatePenaltyViolation(walletAddress, violation); // Update penalty violation
}
// Get total duration of bike use
function renterTimespan(uint startTime, uint endTime) internal pure returns(uint) {
  return endTime - startTime;
}
function getTotalDuration(address walletAddress) public view returns(uint) {
  require(renters[walletAddress].active == false, "Bike is currently rented.");
  uint timespan = renterTimespan(renters[walletAddress].startTime, renters[walletAddress].endTime);
  uint timespanInMinutes = timespan / 60;
  return timespanInMinutes;
}
// Get Contract balance
function balanceOf() view public returns(uint) {
  return address(this).balance;
}
// Get Renter's balance
function balanceOfRenter(address walletAddress) public view returns(uint) {
  return renters[walletAddress].balance;
```

```
}
```

```
// Get the balance of the company's wallet
  function balanceOfCompany(address companyAddress) public view returns (uint) {
    return companyAddress.balance;
  }
  // Set Due amount
  function setDue(address walletAddress) internal {
    uint timespanMinutes = getTotalDuration(walletAddress);
    uint fiveMinuteIncrements = timespanMinutes / 5;
    renters[walletAddress].due = fiveMinuteIncrements * 0.01 ether;
  }
  function canRentBike(address walletAddress) public view returns(bool) {
    return renters[walletAddress].canRent;
  }
  // Deposit
  function deposit(address walletAddress) payable public {
    renters[walletAddress].balance += msg.value;
  }
  // Make Payment
  function makePayment(address walletAddress) payable public {
    require(renters[walletAddress].due > 0, "You do not have anything due at this time.");
     require(renters[walletAddress].balance > msg.value, "You do not have enough funds to cover payment.
Please make a deposit.");
    // Transfer the due amount to the company wallet
    companyWallet.transfer(renters[walletAddress].due);
    renters[walletAddress].balance -= renters[walletAddress].due;
    renters[walletAddress].canRent = true;
    renters[walletAddress].due = 0;
    renters[walletAddress].startTime = 0;
    renters[walletAddress].endTime = 0;
  }
  // Update rewards
  function updateRewards(address walletAddress, uint256 distance, bool violation) internal {
    renters[walletAddress].rewards.rideCount++;
    renters[walletAddress].rewards.rideDistance += distance;
    uint256 rideCountRewards = (renters[walletAddress].rewards.rideCount / 10) * rewardPerRideCount;
    uint256 distanceRewards = (renters[walletAddress].rewards.rideDistance / 10) * rewardPerDistance;
    uint256 penalty = violation ? penaltyPerViolation : 0;
    renters[walletAddress].balance += rideCountRewards + distanceRewards - penalty;
  }
  // Update monthly ride count
  function updateMonthlyRideCount(address walletAddress) internal {
    uint256 currentMonth = (block.timestamp / 1 days) / 30; // Assuming 30 days per month
    monthlyRideCount[walletAddress][currentMonth]++;
  }
  // Update leaderboard positions
  function updateLeaderboard(address walletAddress) internal {
    bool[3] memory positions = renters[walletAddress].rewards.leaderboardPositions;
```

```
// Check if the user has entered top 3 positions
    for (uint256 i = 0; i < 3; i++) {
       if (!positions[i]) {
                            if (i == 0 \parallel (i > 0 && monthlyRideCount[walletAddress][currentMonth] >
monthlyRideCount[renters[walletAddress].walletAddress][currentMonth])) {
            for (uint256 j = 2; j > i; j--) {
                            renters[renters[walletAddress].walletAddress].rewards.leaderboardPositions[j] =
renters[renters[walletAddress].walletAddress].rewards.leaderboardPositions[j - 1];
            renters[renters[walletAddress].walletAddress].rewards.leaderboardPositions[i] = true;
            break:
         }
       }
     }
    // Check if the user has won a leaderboard reward
    if (positions[0] && renters[walletAddress].rewards.leaderboardRewards < leaderboardReward) {
       renters[walletAddress].rewards.leaderboardRewards = leaderboardReward;
       renters[walletAddress].balance += leaderboardReward;
     }
  }
  // Update penalty violation
  function updatePenaltyViolation(address walletAddress, bool violation) internal {
  if (violation) {
    renters[walletAddress].penalty += penaltyPerViolation;
    renters[walletAddress].balance -= penaltyPerViolation;
     }
  }
  // Get rewards balance
  function rewardsOf(address walletAddress) public view returns(uint) {
    Renter memory renter = renters[walletAddress]:
    uint256 rideCountRewards = (renter.rewards.rideCount / 10) * rewardPerRideCount;
    uint256 distanceRewards = (renter.rewards.rideDistance / 10) * rewardPerDistance;
                  return rideCountRewards + distanceRewards + renter.rewards.leaderboardRewards -
penaltyPerViolation;
  }
  // Redeem rewards
  function redeemRewards(address walletAddress) public {
    uint rewards = rewardsOf(walletAddress);
    require(rewards > 0, "No rewards available");
    renters[walletAddress].balance = 0:
    payable(walletAddress).transfer(rewards);
    renters[walletAddress].canRent = true;
  }
  // Update reward per ride count
  // function updateRewardPerRideCount(uint256 reward) public {
  // require(msg.sender == owner, "Only owner can update reward per ride count.");
  // rewardPerRideCount = reward;
  // }
  // Update reward per distance
  // function updateRewardPerDistance(uint256 reward) public {
  // require(msg.sender == owner, "Only owner can update reward per distance.");
  // rewardPerDistance = reward;
  // }
```

```
// Update leaderboard reward
// function updateLeaderboardReward(uint256 _reward) public {
// require(msg.sender == owner, "Only owner can update leaderboard reward.");
// leaderboardReward = _reward;
// }
// Update penalty
// function updatePenaltyPerViolation(uint256 penalty) public {
// require(msg.sender == owner, "Only owner can update penalty.");
// penaltyPerViolation = _penalty;
// }
```

// }

// Get current leaderboard positions

function getLeaderboardPositions() public view returns(bool[3] memory) {
 return renters[msg.sender].rewards.leaderboardPositions;
}

}