Blocarc Smart Salary Product Business Proposal

DEC 2022

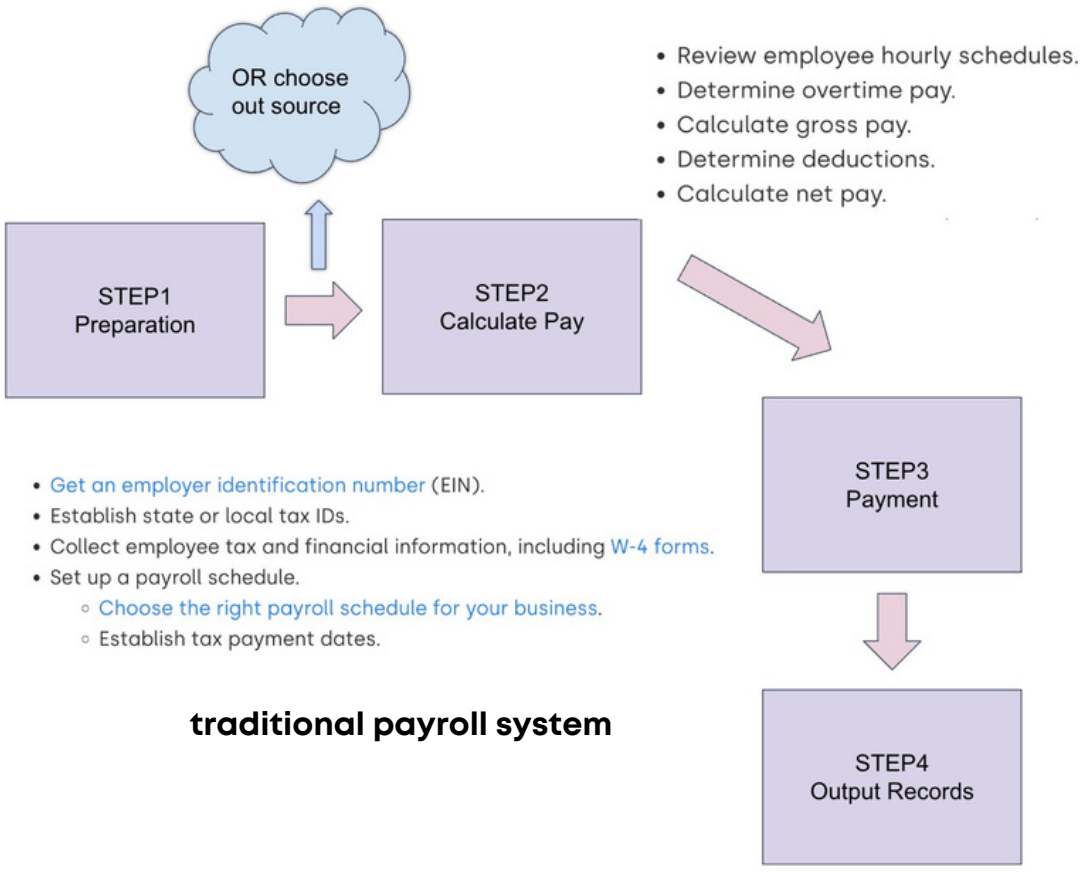
Prepared by

Qian Zha
Caitlin Cao
Xinyi Liang
Ken Chuang

Introduction& Problem Statement

Nowadays in the midst of pandemic, both the employees and employers are struggling with wage payment plan and payroll systems. Particularly, for those client-facing and project-based companies, the employees have been finding it hard to be paid for services performed or minutes of service, while the employers are considering to save cost with third-party payroll agencies and result in delay of payment. In addition, both the biometric data of employees and the company's financial data can be leaked in the middle of this. The problem we are trying to tackle here is how can employees be paid more efficiently without any disputes, how can employers pay them in a more cost-saving, timely, and secure manner, meanwhile assuring that both sides feel their rights are protected and are content throughout the process. Our project is not only different from others, as it is leveraging the smart contract on salary payment system, but also unique, since there has been no blockchain project dealing with payroll issues.

Introduction&
Problem Statement



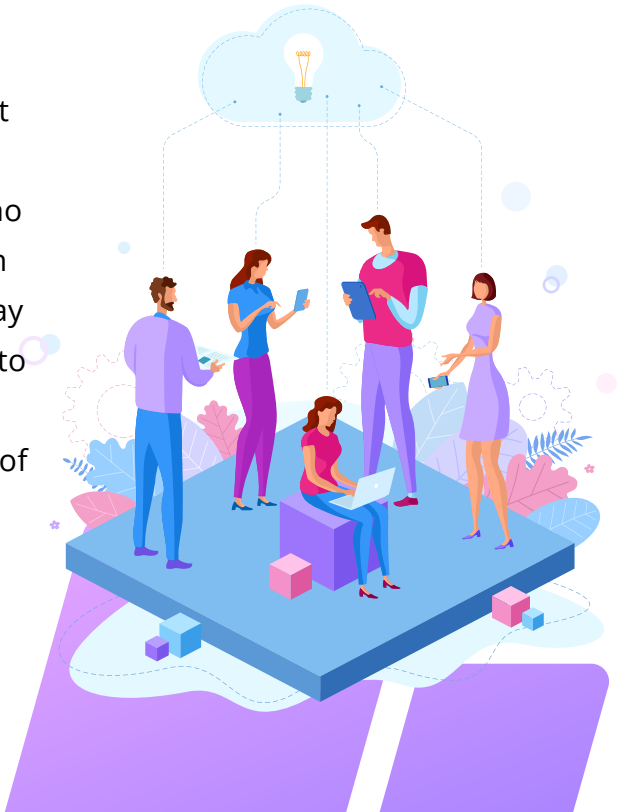
traditional payroll system

Market Analysis

While our smart salary product could be applied through large and small companies all over the world, our primary target market are mid-size and large-size companies.

Traditionally, large companies handle their employees' payroll through a third party, so we can circumvent the middleman by using smart contracts.

- Currently, payments are slow and laborious, taking up to 45 days to be agreed and arrive. Our smart salary product promises daily payments triggered by smart contracts. Large companies with a lot of employees are more willing to use our smart salary product because the delays and inefficiencies of salary payment can be dealt with.
- Specifically, it saves a lot of money paid as a fee to the third parties, increases security as the company's employee database remains within the company and need not be shared with the third party.
- Our blockchain-based online employment salary payment system tends to protect the rights and interests of both employees and employers. With our product, payments no longer have to be constrained to regular payroll runs with specific cut-off dates. On one side, smart contracts can pay wages to employees in real time, which allow employees to receive immediate compensation for their labor. On the other hand, with our smart payment product, employers of large and mid size companies can reduce company accounting costs.



Product Idea&MVP

PART 1 WHAT IS OUR PRODUCT

The core value of our smart salary product is to use the compulsion of smart contracts and blockchain to provide employees with certainty of reliability, avoid salary arrears and miscalculation; automate salary distribution for companies, reduce administrative costs, and improve efficiency. Therefore, MVP will mainly focus on realizing the smart contracting of salary payment terms and practicing decentralized, automatic, mandatory and accurate salary payment.

PART 2 HOW OUR MVP SOLVE THE PROBLEM

Our application works based on blockchain and smart contracts, uploading employee's salary-related data to the blockchain, and using smart contracts to perform accurate calculations and punctual payments in full accordance with the terms agreed upon by the company and employees. In the salary payment under the smart contract mechanisms, the enterprise adjusts the salary payment behavior mode from the traditional "active salary payment" to "authorized salary deduction". The product, therefore, can resolve issues such as inaccurate wage calculations and untimely payment, which better protect the rights and interests of employees. From the company's perspective, it also saves huge amount of manpower, operation and technical costs.

Product Idea&MVP

PART 3 WHY USE OUR PRODUCT?

From the data we researched, there is currently no salary payment system that relies on smart contracts, so we cannot make a comparison. However we intend to differentiate ourselves from other potential similar product. Our products are not limited to just developing the core payroll system, we also plan to expand into a complete smart payroll system. For example, we hope to help companies pay wages in multiple currencies, which is especially useful for international companies. Companies no longer need to hold multiple currencies and face related currency risks, and employees also have more convenient currency choices. For another example, we also plan to realize the real-time calculation and payment of employee wages. At present, most companies settle wages for the first month in the middle of the second month. This is because of the need for auditing and accounting, but it causes the time difference of employee wages. Through real-time settlement, we can allow employees to get their own labor remuneration in minutes or even smaller units, realize constant settlement and incentives, reduce corporate costs and better meet employee needs. In addition, there are many functions to be realized, and our overall goal is to provide a complete full-process solution. It is the comprehensiveness of this full-process approach that sets us apart from potential competitors.

Product Idea&MVP

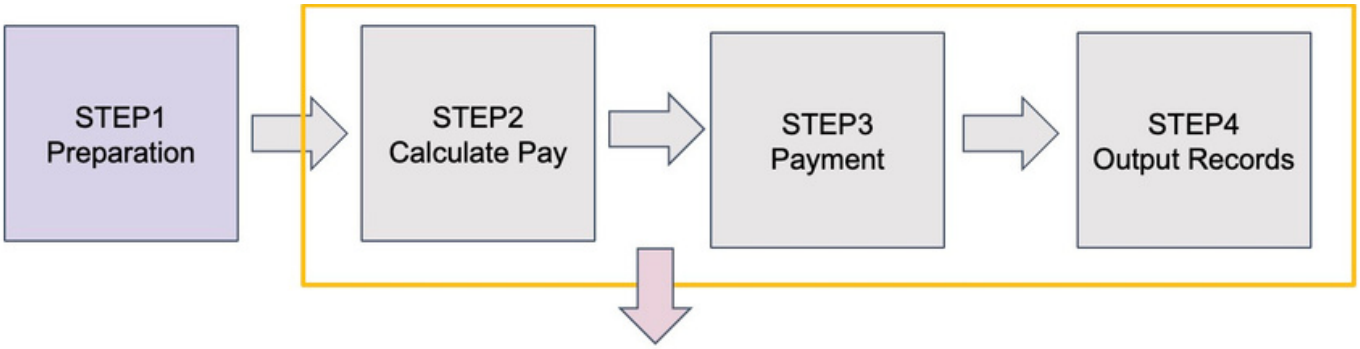
PART 4 WHY OUR PRODUCT SHOULD USE BLOCKCHAIN

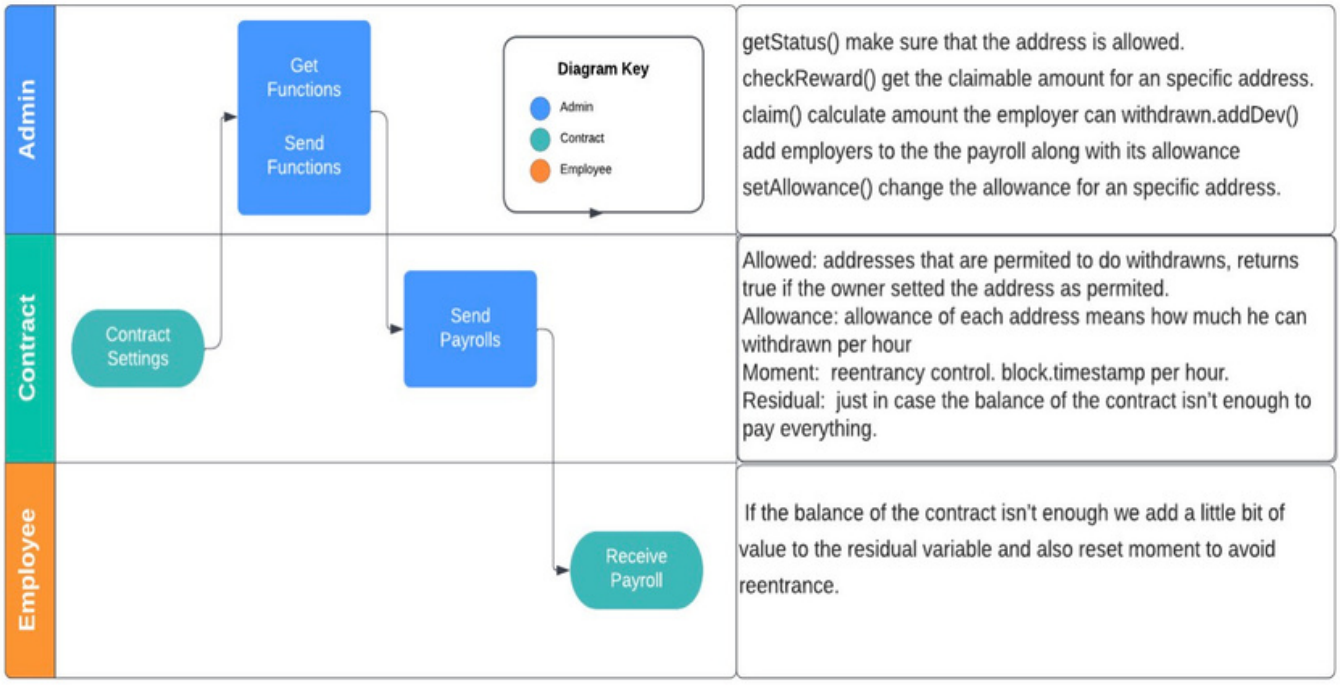
Blockchain technology has a great potential to change payroll processes, and is necessary for the smart salary product. The blockchain security property is one of the keys of our solution.

- "In blockchain technology, cryptography used extensively to sign data in order to prove that a transaction was approved by the owner of the funds." This property builds trust between the company and the employees making sure all the employees are getting the right amount of their share.
- Also Blockchain offers faster settlement compared to the traditional banking system. The employees will be able to get their paycheck without time delay.
- Employees can pick which kind of currency they prefer to receive.
- Blockchain transaction ledgers are easily trackable in a decentralized system. It helps the transaction history to be more transparent and unquestionable. Employee payroll management system has to comply with a lot of regulations and blockchain will help to reduce the discrepancies and save time for the HR department.

Product Idea&MVP

PART 5 Business Flow





Minimum-Single Purpose Demo

Methodology

We will use Solidity to develop out smart contract.

- Contract-oriented, high-level language for implementing smart contracts.
- Influenced by C++, Python and JavaScript and is designed to target the Ethereum Virtual Machine (EVM)
- Statically typed, supports inheritance, libraries and complex user-defined types among other features.

We plan to use the deploy and test function on Remix to test whether the contract function works, whether the result meets the expectation, and whether there are any errors. Virtual digital currency can be used for testing.

Our smart contract has 13 functions as of now.

1.changeCompanydetail	2.addFunds
3.contractBalance	4.addEmployee
5.removeEmployee	6.getEmployeeAddress
7.setEmployeesalary	8.getEmployeesalary
9.getEmployeebalance	10.setEmployeepayment
11. payrollSystem	12.etherlocal
13.destroy	

```
// SPDX-License-Identifier: GPL-3.0-or-later
pragma solidity >=0.4.4<0.9.0;

contract BinancepayrollSystem {

    address public company;
    address payable employeeAddress;
    struct employee {
            uint ID;           //employee ID
            string name;       //employee name
            string department; //employee department
            uint salary;       //employee salary to sent
            uint balance;      //employee account balance
            string lastPaytime;//employee's salary last paytime
            bool active;       //employee's status active
            string salarytype;
        }
    mapping(address => employee) employeeDetails; //mapping employee address to employee information detail
    address[] employeeList;

    //payroll constructor: set the company as payroll sender
     constructor () public {
        company = msg.sender;
    }

    //access control modifier
    modifier onlyCompany{
        require (msg.sender == company,
        "Only company owner can call this."
        );
        _;
    }
```

Employee's information

mapping employee address to employee detail

set the company as the sender

access control

Minimum-Single Purpose Demo

```
function changeCompanydetail(address _companyAddress, uint balance,
                             uint newfunds,string memory lastaddtime) public {
    companyDetail[ _companyAddress] = company(balance,newfunds,lastaddtime);
}
```

change the detailed information about the company

```
function contractBalance() view public returns (uint) {
    return address(this).balance;
}
```

check the smart contract balance

```
function addEmployee(address _employeeAddress, uint ID, string memory name,
string memory department, string memory country,uint balance,uint salary,
string memory lastPaytime,bool active,uint exchangerate, uint payment) public {
    employeeDetails[ _employeeAddress] = employee(ID,name,department,country,balance,salary,
                                                  lastPaytime,active,exchangerate,payment);
    employeeList.push(_employeeAddress);
}
```

adding an employee to the company's list of employes if not already registered

```
function addFunds() public payable{
    emit deposit(msg.sender, msg.value);
}
```

add funds to the smart contract account

```
function removeEmployee(address _employeeAddress) onlyCompany public{
    employeeDetails[ _employeeAddress].active = false;
}
```

remove an employee if the employee leaves the company

```
function getEmployeeAddress(uint _employeeDetails) public view returns (address){
    return employeeList[_employeeDetails];
}
```

get an employee's address from the company's employee list

```
function setEmployeesalary(address _employee, uint _salary) public payable returns (uint){
    require(_salary > 0, "salary must be greater than 0");
    return (employeeDetails[_employee].salary=_salary);
}
```

set the employee's salary based on the address

```
function getEmployeesalary(address _employee) public view returns (uint){
    return (employeeDetails[_employee].salary);
}
```

get employee's salary based on the address

```
function getEmployeeBalance(address _employee) public payable returns (uint){
    return (employeeDetails[_employee].balance=employeeDetails[_employee].salary +
            employeeDetails[_employee].balance);
}
```

get employee's new salary balance by adding the current amount to the balance in the account

```
function setEmployeepayment(address _employee, uint _balance) external payable returns (uint){
    return (employeeDetails[_employee].payment= _balance);
}
```

set the payment as the new salary balance

```
function payrollSystem(address _employee) external payable returns (uint){
    //send the payroll to the employee
    payable(companyAddress).transfer(employeeDetails[_employee].payment);
    return (employeeDetails[_employee].payment);
}
```

transfer salary to the employee

```
function etherlocal(address _employee) external payable returns (uint){
 return (employeeDetails[_employee].payment = employeeDetails[_employee].payment *
                                              employeeDetails[_employee].exchangeRate);
}
```

convert salary based on whichever country's exchange rate the user wants