**Escrow**

**Dependencies**: ethersjs

**Create job**

Call smart contract function initializeJob() with the parameters,

1. **Structure**

struct JobCreation{

        uint256 jobId;

        address freelancer;

        uint96 payment;

        uint256 fees;

        uint256 startTime;

        uint256 endTime;

    }

* jobId: Concatenation of contractor address and an autoincrement number or current timestamp.

CODE:

const getJobId = (userAddress, number) =>{

  return ethers.utils.solidityPack([ "address", "uint96" ], [userAddress, number]); }

* Freelancer : address of the freelancer in hex format
* Payment (in USDT): The payment for the job which will be stored in the escrow
* Fees: The fees to be paid by the contractor during job creation. It will be set by platform (like 5%, 10%).
* startTime: The start time of the validity of the signature provided by the platform
* endTime: The end time of the validity of the signature provided by the platform

1. **Signature**

const signJobCreation = async(signer, contractor, \_jobId,\_freelancer, \_payment, \_fees,\_startTime,\_endTime) => {

  const domain = {

      name: PROJECT\_NAME,

      version: VERSION,

      chainId: CHAIN\_ID,

      verifyingContract: verifyingContract

  };

  // The named list of all type definitions

  const types = {

    jobCreation: [

      {name: "contractor", type: "address"},

      {name: "jobId", type: "uint256"},

      {name: "freelancer", type: "address"},

      {name: "payment", type: "uint256" },

      {name: "fees", type: "uint256"},

      {name: "startTime", type: "uint256"},

      {name: "endTime", type: "uint256"}

    ]

  };

  const value = {

    contractor: contractor.address,

    jobId: \_jobId,

    freelancer: \_freelancer,

    payment: \_payment,

    fees: \_fees,

    startTime: \_startTime,

    endTime: \_endTime

  };

  let signature = signer.\_signTypedData(domain, types, value);

      return signature;

}

**Release payment**

Function arguments:

(uint256 jobId, uint256 paymentToRelease)

* jobId: Concatenation of contractor address and an autoincrement number or current timestamp.

CODE:

const getJobId = (userAddress, number) =>{

  return ethers.utils.solidityPack([ "address", "uint96" ], [userAddress, number]); }

* paymentToRelease: The amount in USDT to be release at any given point of time, by the contractor, which can then be claimed by the freelancer.

**Claim payment**

Function arguments:

struct JobInteract{

        uint256 jobId;

        uint256 fees;

        uint256 startTime;

        uint256 endTime;

    }

* jobId: Concatenation of contractor address and an autoincrement number or current timestamp.

CODE:

const getJobId = (userAddress, number) =>{

  return ethers.utils.solidityPack([ "address", "uint96" ], [userAddress, number]); }

* Fees: The fees to be paid by the contractor during job creation. It will be set by platform (like 5%, 10%).
* startTime: The start time of the validity of the signature provided by the platform
* endTime: The end time of the validity of the signature provided by the platform

1. Signature

const signJobClaim = async(signer,\_jobId,\_fees,\_startTime,\_endTime) => {

  const domain = {

    name: PROJECT\_NAME,

    version: VERSION,

    chainId: CHAIN\_ID,

    verifyingContract: verifyingContract

};

// The named list of all type definitions

const types = {

  jobClaim: [

    {name: "jobId", type: "uint256"},

    {name: "fees", type: "uint256"},

    {name: "startTime", type: "uint256"},

    {name: "endTime", type: "uint256"}

  ]

};

const value = {

  jobId: \_jobId,

  fees: \_fees,

  startTime: \_startTime,

  endTime: \_endTime

};

let signature = signer.\_signTypedData(domain, types, value);

    return signature;

}

**Cancel Job**

Function arguments:

1. 1. struct JobInteract{
   2. uint256 jobId;
   3. uint256 fees;
   4. uint256 startTime;
   5. uint256 endTime;
   6. }

* jobId: Concatenation of contractor address and an autoincrement number or current timestamp.

CODE:

const getJobId = (userAddress, number) =>{

  return ethers.utils.solidityPack([ "address", "uint96" ], [userAddress, number]); }

* Fees: The fees to be paid by the contractor during job creation. It will be set by platform (like 5%, 10%).
* startTime: The start time of the validity of the signature provided by the platform
* endTime: The end time of the validity of the signature provided by the platform

1. Signature

const signJobCancellation = async(signer, \_jobId, \_FEES, \_startTime, \_endTime) => {

  const domain = {

    name: PROJECT\_NAME,

    version: VERSION,

    chainId: CHAIN\_ID,

    verifyingContract: verifyingContract

};

// The named list of all type definitions

const types = {

  jobCancellation: [

    {name: "jobId", type: "uint256"},

    {name: "fees", type: "uint256"},

    {name: "startTime", type: "uint256"},

    {name: "endTime", type: "uint256"}

  ]

};

const value = {

  jobId: \_jobId,

  fees: \_FEES,

  startTime: \_startTime,

  endTime: \_endTime

};

let signature = signer.\_signTypedData(domain, types, value);

    return signature;

}