

# skepa

## Decentralized Scheduled Payments on Arbitrum

skepa. is a decentralized application (dApp) that enables users to create and manage scheduled ETH payments on the Arbitrum Sepolia network. Whether you need to set up one-time future payments or recurring transfers, skepa. handles the automation seamlessly using Chainlink Automation.

---

## Repository

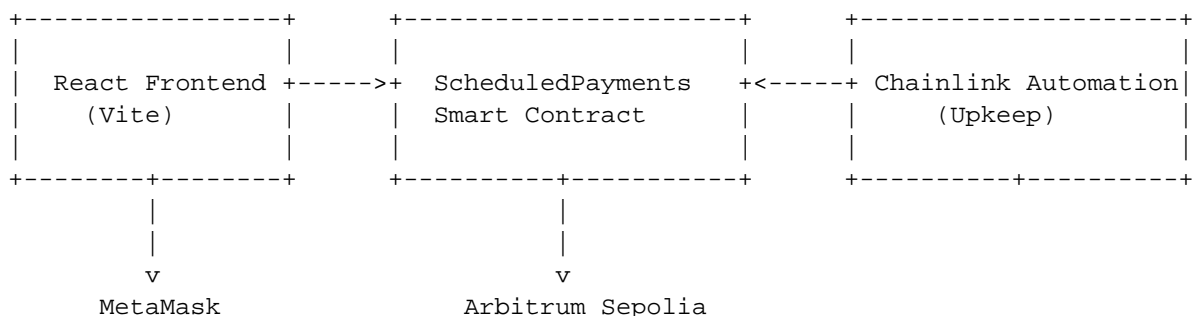
GitHub: <https://github.com/andreivasilescu24/Scheduled-Payments>

## Features

- **One-time Payments** - Schedule a single payment to be executed at a specific future date and time
  - **Recurring Payments** - Set up periodic payments (hourly, daily, weekly, monthly) with a defined number of executions
  - **Escrow System** - Funds are locked in the smart contract upon schedule creation, ensuring payment reliability
  - **Automatic Execution** - Payments are triggered automatically by Chainlink Automation when the scheduled time arrives
  - **Cancellation & Refunds** - Cancel active schedules anytime and receive a refund of remaining funds
  - **Favorite Recipients** - Save frequently used addresses for quick access
  - **Real-time Updates** - UI auto-refreshes to reflect payment executions and status changes
- 

## Technical Documentation

### Architecture Overview



## Smart Contract

The `ScheduledPayments` contract is deployed on Arbitrum Sepolia and implements Chainlink's `AutomationCompatibleInterface`.

### Schedule Structure

```
struct Schedule {
    address payer;           // Creator of the schedule
    address recipient;       // Payment receiver
    uint256 amount;          // ETH amount per execution
    uint256 interval;        // Seconds between executions (0 = one-time)
    uint256 nextExecution;   // Timestamp of next execution
    uint256 executionsLeft;  // Remaining number of payments
    uint256 remainingBalance; // Escrowed ETH remaining
    bool active;             // Schedule status
}
```

### Key Functions

Function	Description
<code>createSchedule(recipient, amount, interval, startTime, executions)</code>	Creates a new payment schedule. Requires sending ETH (principal + 0.5% fee)
<code>cancelSchedule(id)</code>	Cancels an active schedule and refunds remaining balance to the payer
<code>getUserSchedules(user)</code>	Returns all schedules created by a specific address
<code>previewTotalCost(amount, executions)</code>	Calculates total ETH required including protocol fee

### Protocol Fee

A 0.5% fee is charged on the total principal amount at schedule creation. This fee is non-refundable.

$$\text{Total Cost} = (\text{amount} \times \text{executions}) + ((\text{amount} \times \text{executions}) \times 0.5\%)$$

### Chainlink Automation Integration

The contract implements two functions required by Chainlink Automation:

- `checkUpkeep()` - Called by Chainlink nodes to check if any payments are due. Returns `true` if at least one schedule is ready for execution.
- `performUpkeep()` - Executes due payments (up to 10 per call to manage gas). Updates schedule state and transfers ETH to recipients.

### Frontend Stack

Technology	Purpose
React 18	UI framework
Vite	Build tool and dev server
ethers.js v6	Ethereum library for wallet and contract interaction
MetaMask	Wallet provider

## ***Project Structure***

```

contract/
  - Scheduled_Payments.sol    # Main smart contract
  - contract-abi.json        # Contract ABI

frontend/
  - src/
    - components/            # React components
    - hooks/                 # Custom hooks (useWallet, useContract, etc.)
    - utils/                 # Constants and helper functions
    - styles/                # CSS styles
  - index.html

README.md

```

---

## **Getting Started**

### ***Prerequisites***

- Node.js 18+
- MetaMask browser extension
- ETH on Arbitrum Sepolia (for gas and payments)
- LINK tokens on Arbitrum Sepolia (for Chainlink Automation funding)

### ***Installation***

```

# Clone the repository
git clone https://github.com/andreivasilescu24/Scheduled-Payments.git
cd Scheduled-Payments/frontend

# Install dependencies
npm install

# Start development server
npm run dev

```

## ***Contract Deployment***

- 1 Deploy `Scheduled_Payments.sol` to Arbitrum Sepolia using Remix, Hardhat, or Foundry
  - 2 Update `CONTRACT_ADDRESS` in `frontend/src/utils/constants.js`
  - 3 Register the contract with Chainlink Automation (<https://automation.chain.link/>) and fund the upkeep with LINK tokens
- 

## **Resources**

This project was built using the following documentation:

- **Chainlink Automation Documentation** (<https://docs.chain.link/chainlink-automation>) - For implementing automated, decentralized execution of smart contract functions
  - **Solidity Documentation (v0.8.20)** (<https://docs.soliditylang.org/en/v0.8.20/>) - For smart contract development best practices and language reference
- 

## **Network Configuration**

Network: Arbitrum Sepolia