Project Report

Project Name:

A Multi-agent System for Crypto Project Fundrasing

NI YONGXIN   A0231559B

WU YICHEN   A0231544M

ZENG ZIJING   A0231548E

PENG JUNHAO   A0231329L

Menu

1 Business Case

This robot is mainly to obtain the fundraising data of each blockchain project on the Icodrops website through the crawler, and then integrate the useful data through data analysis, process and extract the key information that is helpful for investment, thereby greatly saving researchers' time in the digital currency industry, who used to spend a lot of time writing reports every day to assist in investment decisions.

We developed this robot based on the team members' years of work experience in the blockchain industry, combined with both the actual needs of previous work and the automation skills learned in the classroom, we have completed the development of the robot through many continuous attempts. Although this time we only used one website's fundraising data to generate daily reports, the data of many different websites can also be obtained by our robot in the same way and use the same data processing logic to generate the same type of research report, so this project has great business potential whether it is in terms of scalability or in terms of commercial value.

2 Market Research

If it weren't for the robots, it was certain to take a huge amount of time to obtain all projects data from just one website and analyze it with Excel, draw charts,

summarize the classification, and write it into a research report. It probably takes a few days for each researcher, and after writing the report, it also takes a few hours per week to maintain and update the data, so, this kind of work is usually structured and repetitive. However, since anyone engaged in project financing must pay attention to these newly financed projects every day, and use this as a reference for investment, these complicated and repetitive tasks are quite necessary

Fortunately, with the use of automated robots, researchers in each industry can save a lot of fragmentary time and lead a more productive life. Furthermore, another advantage of using robots is that the program is good at handling structured problems. We will no longer worry about making mistakes due to carelessness, and we can always let our robot generate a beautiful report and send it to us on time.

We took scalability and universality into consideration at the beginning of designing our robot, The crawler part adopts a general framework, which is used to call the browser interface to crawl through python and is suitable for most different data websites. As for our data analysis module, it is also suitable for processing various types of data, and generating charts as well as text descriptions for the given data, likewise, the report generation module and the email sending module can also be used directly on other websites, so our robot is universal and

can be used to write blocks Most data-based structured reports in the chain industry

3 System model

3.1 Overall Architecture
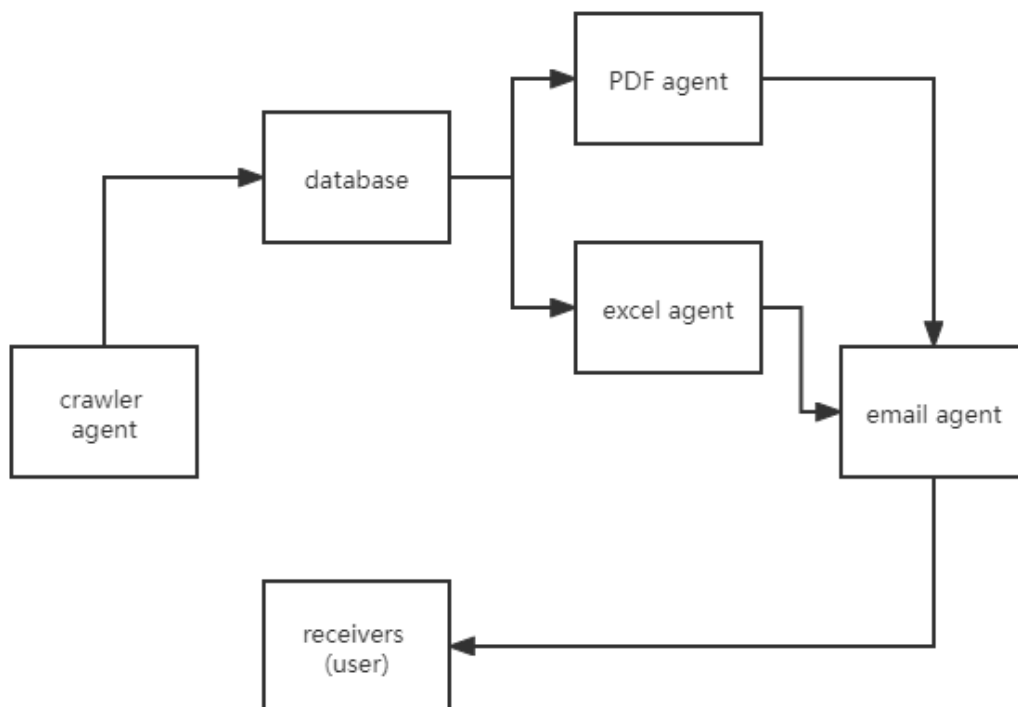
Figure: Overall Architecture of Our Multi-agent System

From the above figure, we can see that our system stores data through a cloud database, and three agents perform all the functions of the whole system. Specifically, crawler agent regularly collects the data we want from the target website and then uploads it to the database; Pdf agent reorganizes the obtained data to generate title, chart and text (organized in latex template), and then it

compiles latex into a typeset PDF; the email agent will automatically send the generated PDF (that is, daily report) to our mailbox.

3.2   System Development
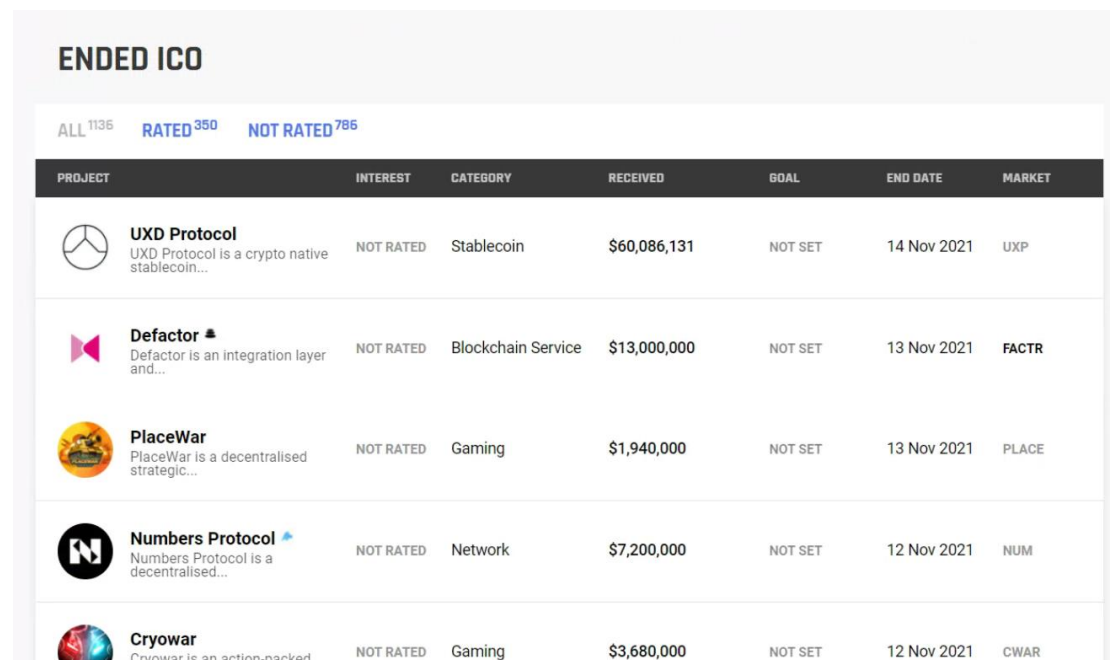
What the tools (python libraries) this system uses are: selenium, bs4, pymysql, requests, lxml, matplotlib, cfscrape, smtplib, email.

In order to run the system smoothly, texlive environment should be equipped locally (the system commands inside PDF agent use this environment to compile latex into PDF), and "chromedriver. exe" should be available in the root directory (crawler agent uses it as a tool to simulate interaction).

3.3   Crawler Agent

This agent uses selenium library and cfscrape library to complete automatic collection. Firstly, it opens the main page through the chrome driver. Since the main page adopts the lazy loading technology of page rendering, the agent constantly sends key simulation requests to the page, simulates the click of the button "END", and constantly checks how many records there are in the total page. When the " END" is clicked many times and the records are no longer increased, it stops sending key requests and capture all data rows on the main page. Then it uses the beautifulsoup library to process the loaded page source code, so that we can directly obtain the data in the target tag through tag name, id, class, etc. Through the pymysql library, these data are eventually inserted into the database

table "ENDEDICO", whose format is "ENDEDICO (PROJECT, INTEREST, CATEGORY, RECEIVED, GOAL, ENDDATE, MARKET, URL)".

Figure: Target Website (main page)

In addition to the information collection of the main page, for each row of data collected, we need to click in to obtain the corresponding detail page and collect the information from several page modules corresponding to the detail page. Before we start, we designed a piece of code to create an IP pool, which will be used in subsequent specific pages to bypass the detection of anti-crawler mechanism. Finally, all available IPs will be stored in the file " ip_proxy.txt" in the form of {http: 'IP address'}. We use cfscrape to create a scraper dedicated to cloud flare defense. For each new target url, we use a new IP from the IP pool to assemble a new request header. Note that if the same IP makes frequent requests to a website, this IP will be blocked. For each page (their templates are the same), we obtain the data from the corresponding page block and store it in a large variable. Through the pymysql library, these data are eventually inserted into the

database table "ENDEDICO_DETAl", whose format is "ENDEDICO_DETAIL (project, logo, description, fund_goal, main_entries, social_links, our_rating, token_sale_details, market_return, short_review, additional_links)".
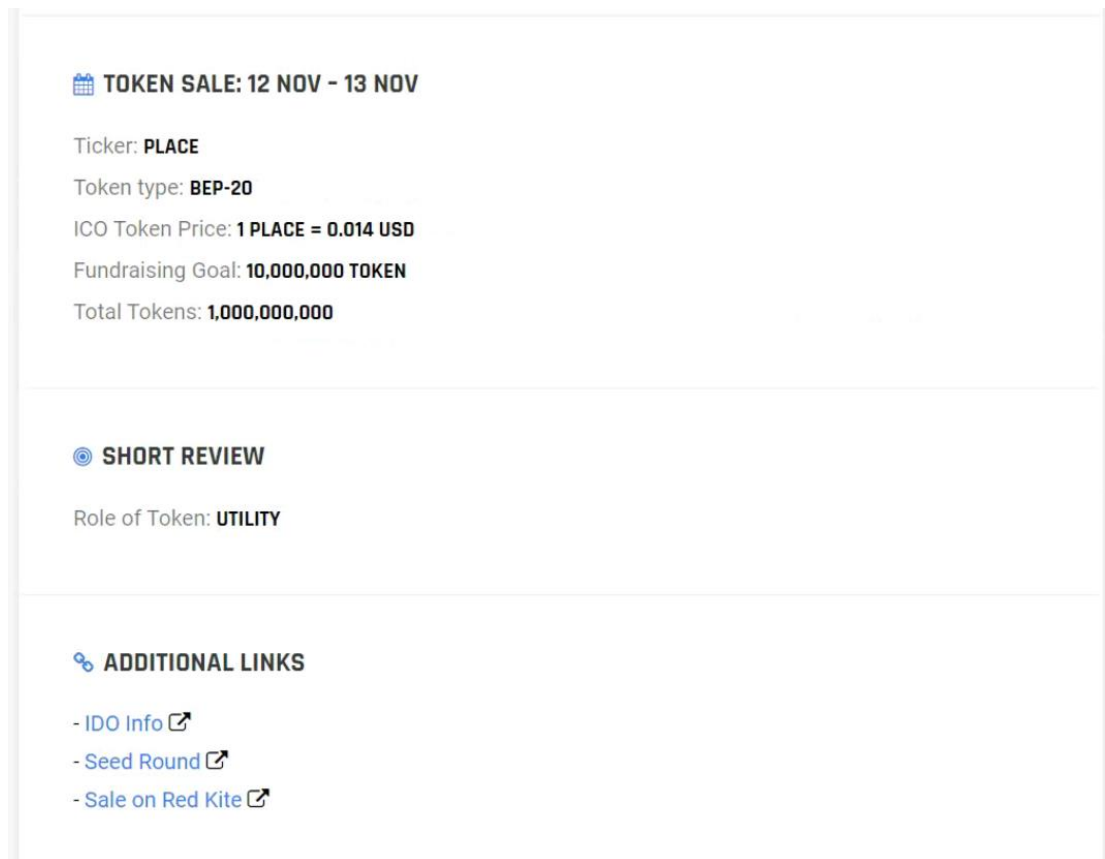
Figure: Target Website (detail page)

Before the above two data uploading processes, we will clear the corresponding database to ensure that the data is up-to-date and there is no duplication.

3.4 PDF Agent

The agent queries the latest data from the database and returns. Based on these data, it organizes the description of the report, generates the corresponding chart (PNG format) through the matplotlib library, and finally outputs all these as a tex file called "Daily Report".

Next, it calls multiple system command lines "os.system("bmeps -c xxx.png xxx.eps")" to convert all PNG images to EPS format that supported by latex. With the prepared tex file and the corresponding EPS format chart, it subsequently calls

the following system command lines to generate the final PDF.

```
os.system("C:/texlive/2020/bin/win32/xelatex Daily_Report.tex")
```

The first step above is to generate the corresponding DVI through tex, and the second step is to convert the DVI into the PDF we want to see. The illustration of the generated PDF is as below:
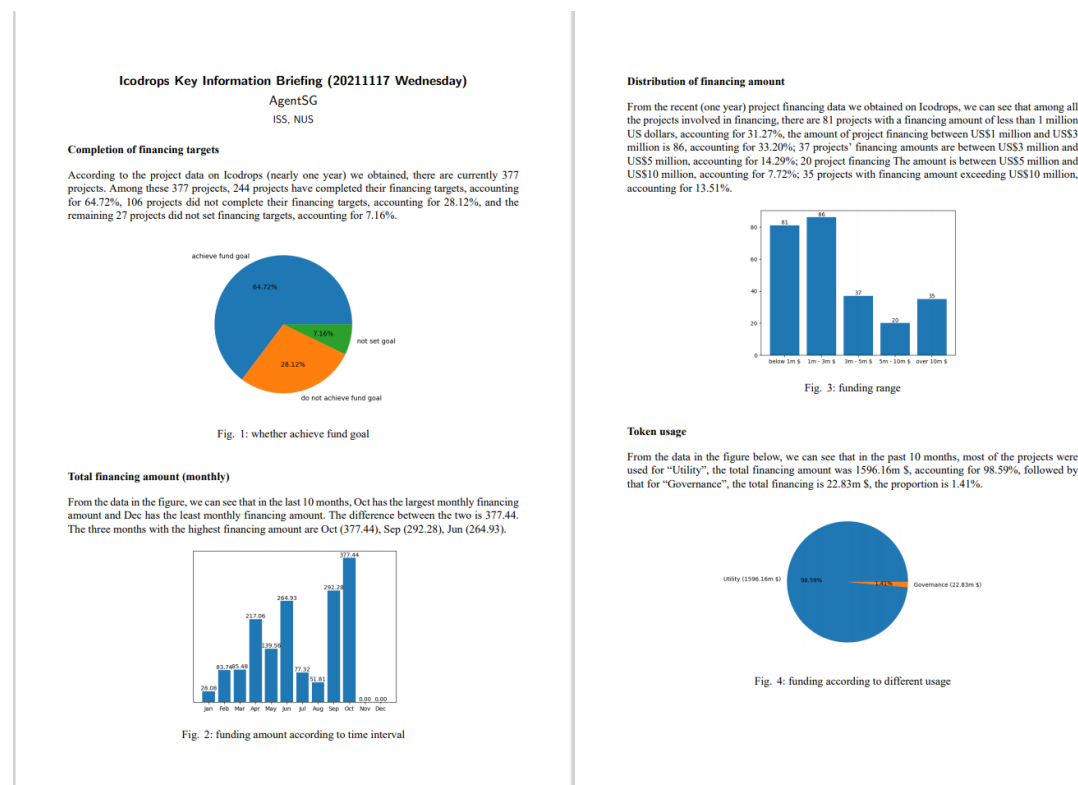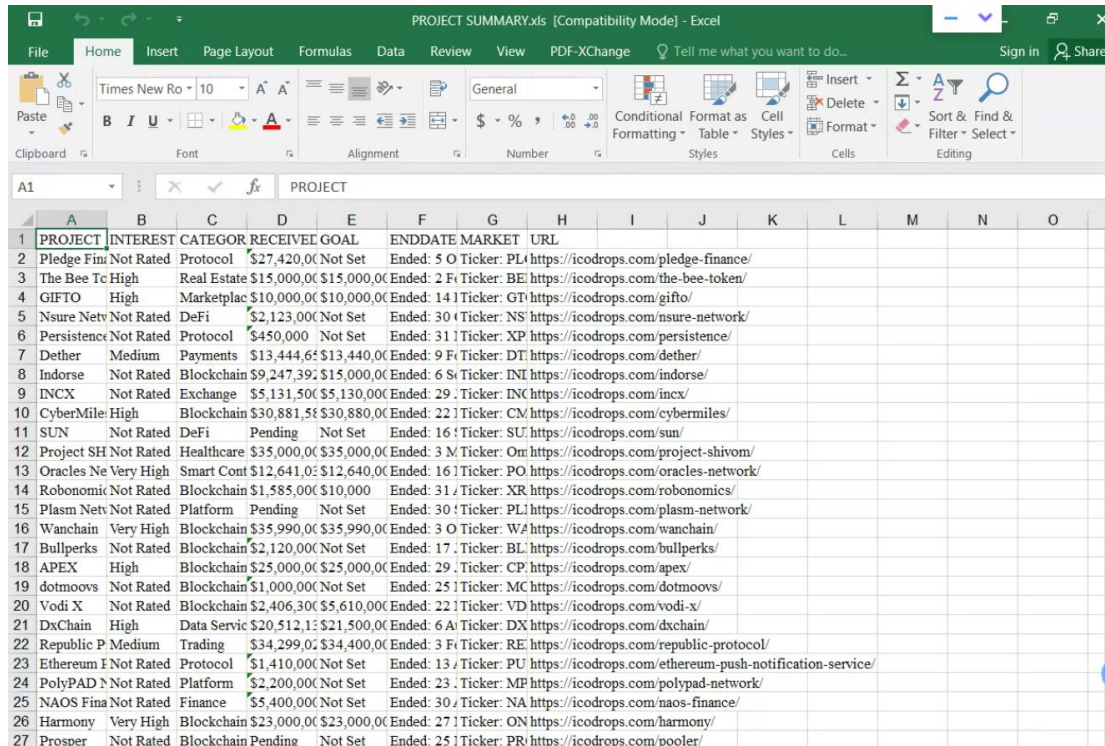
Figure: Daily Report (PDF)

3.5   Excel Agent

In order to provide users with more complete analysis, in addition to the above report generated by PDF agent, we also read two tables in the database and save them in two different Excel files.
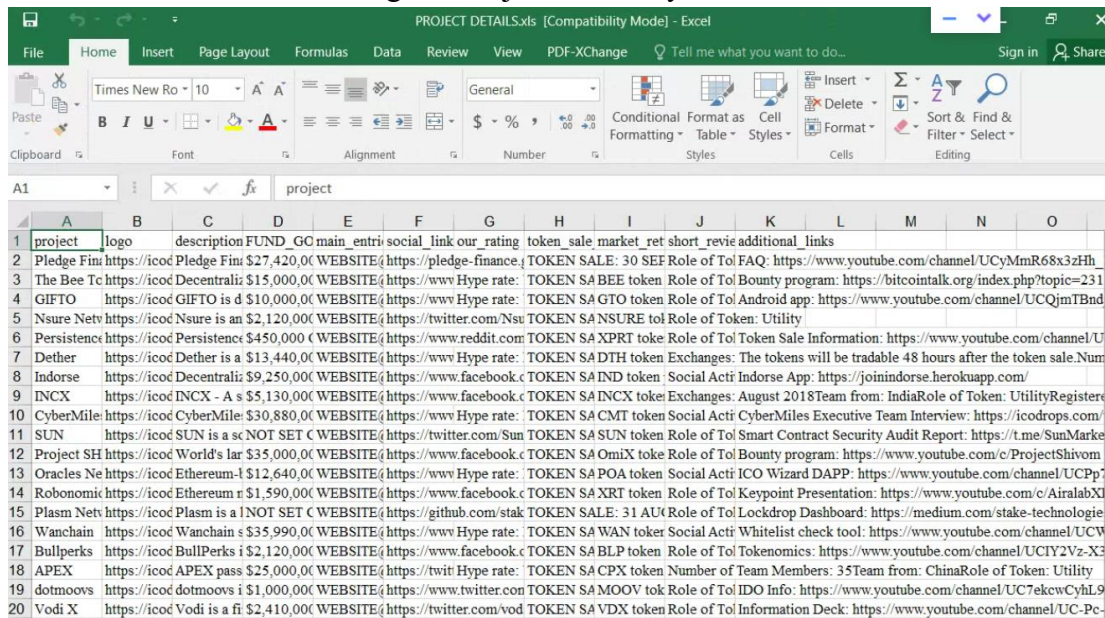
The pymysql library and xlwt library are used here. We use the former to connect to the database and read the corresponding table data, and use the latter to

continuously write data to Excel files according to cell location, and finally generate 'PROJECT SUMMARY.xls ' and 'PROJECT DETAILS.xls', as shown in the following figures:

Figure: Project Summary Excel

Figure: Project Detail Excel

3.6 Email Agent

The agent is based on smtplib library and email library. Firstly, it sets the server host, port and corresponding authorization code. Next, it defines sender email and receiver emails, as well as the email title and email content, and then calls send_email function to start sending email.

In addition to setting the encoding, title and content, an important feature of the send_ email function is that it can read the generated "Daily Report. pdf", "PROJECT SUMMARY.xls" and "PROJECT DETAILS.xls" in the form of binary text and add it to the email via adding attachment. The agent supports sending this email to multiple users at the same time. The effects presented on the user end are as follows:
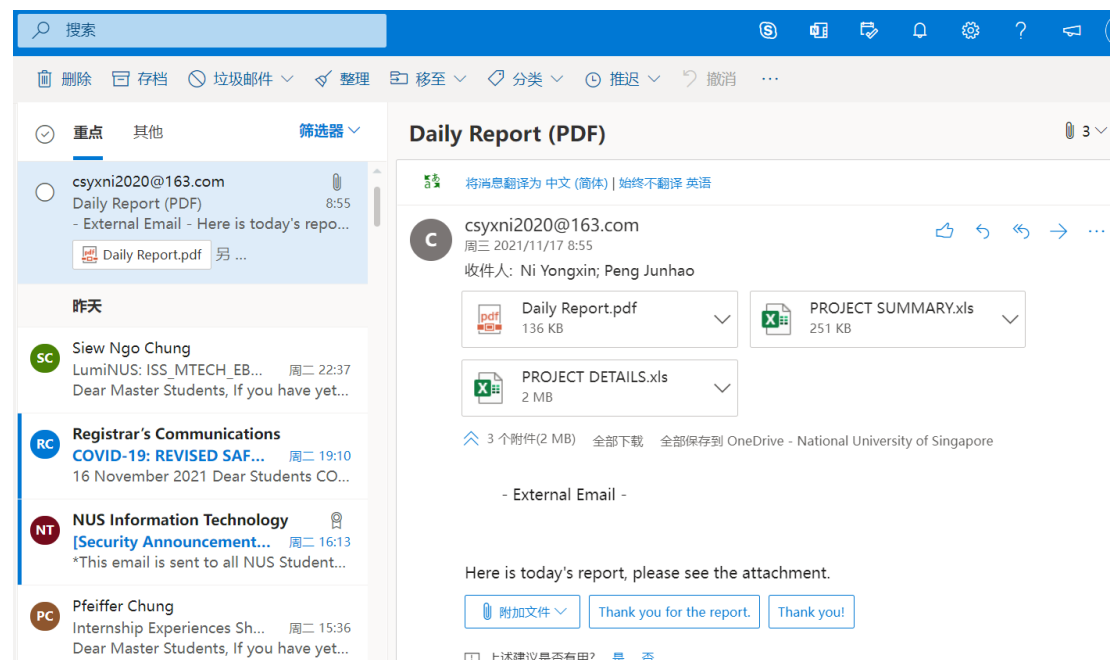

Figure: Email sent by agent

4 Effect Demonstration

The effect is shown in the video. Run the crawler agent, we can see that a browser is automatically opened and pages are automatically turned down (triggering lazy loading of web pages). The agent first collects all records of the main page, and then opens their corresponding detailed pages one by one according to these records. The data of the main page and the detailed page will be uploaded to the two tables of the cloud database respectively. Then we run the email agent, which starts the excel agent and pdf agent first., the two generate Excel files and PDF reports respectively based on the latest data (there are various descriptions and legends in the PDF report, which are established based on the crawled data). Finally, the email agent summarizes these attachments and sends them to our mailbox at the same time.

Please see the video in the attachment.