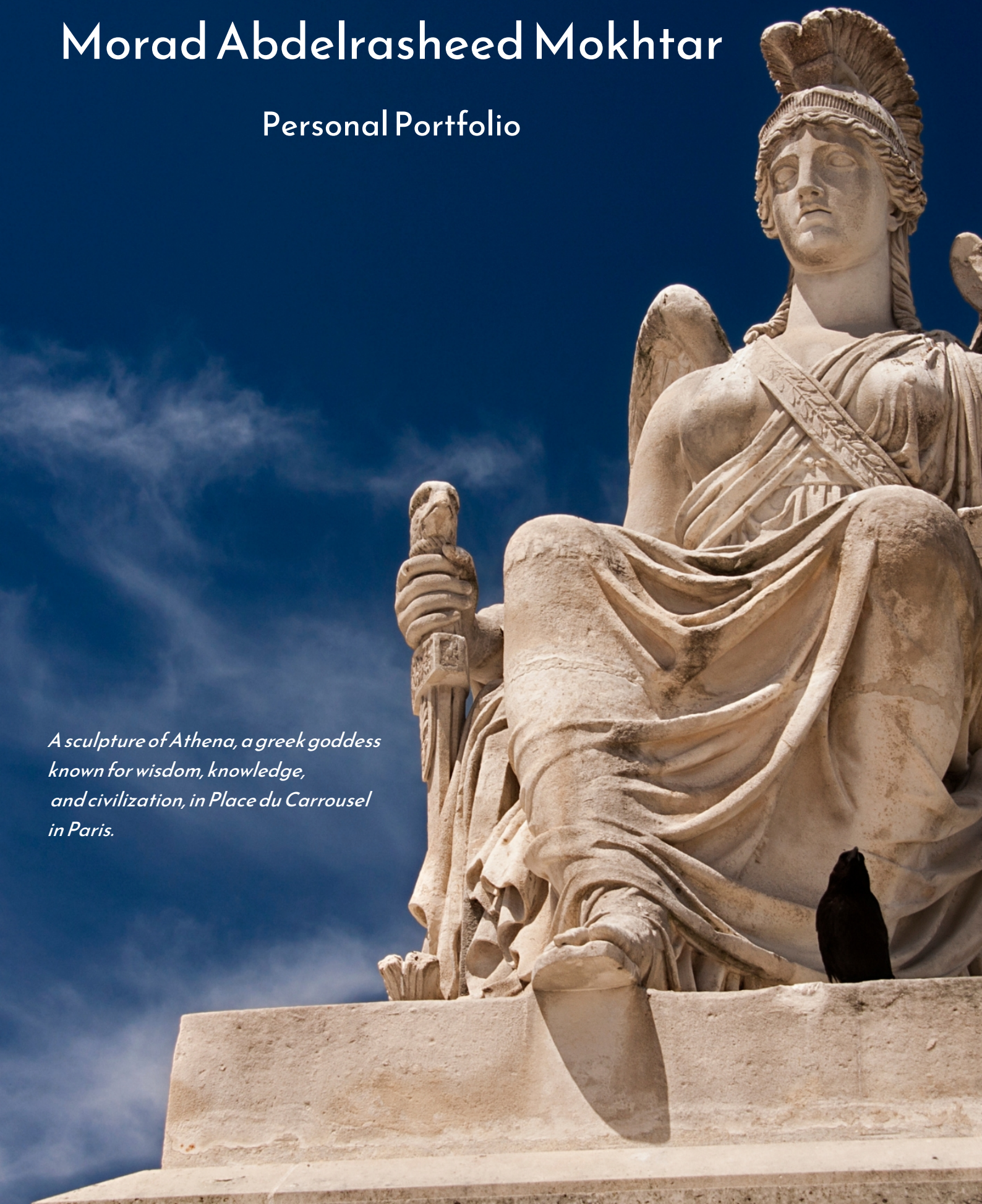
Morad Abdelrasheed Mokhtar

Personal Portfolio



A sculpture of Athena, a greek goddess known for wisdom, knowledge, and civilization, in Place du Carrousel in Paris.

A word from the author

It is known that people have different personalities and ambitions. I believe that our creations and arts are what shows and embraces uniqueness. Either if it is visual art, music, philosophy, or even computer programs, they all have a sense of one's personality traits and life goals. Therefore, we can conclude that a portfolio does not only deliver one's academic experience, but also shows how one's creations both have idiosyncratic traits and how these creations benefited the community one way or another.

For my journey, I was drawn to computational and mathematical sciences by my love of innovation. I have started writing computer programs and exploring the mathematical world since I was six. I thought of computers as a sandbox, or a framework from which you can make your ideas come to life at a low or no cost. But the real personal and academic development started in October of 2018, when I moved from my grandmother's house, due to physical and mental abuse. I discovered a ton of resources and topics to learn. My life revolved around Google Scholar, Wolfram Alpha/Mathematica, and my IDE. Since then, I have not only learned how to code a web application, or an advanced airodynamic simulator, but personality traits and life lessons that changed my mindset, and how I perceive the world around me.

My work was not the result of a socially awkward nerd who likes to code things just to look cool, but the result of someone with an intention to benefit the human kind. You do not have to be a billionaire, or a revolutionist in order to make an impact. As Martin Luther King Jr. said: "*If I cannot do greater things, I can do small things in a great way*".

There are two concepts that ensures the continuity of the human civilization: Ethics, and the pursuit of Knowledge.

Warmest Regards and wishes,

Table of Contents

Definitions

IDE: Integrated Development Environment

E2EE: End-to-End Encryption

TCP: Transmission Control Protocol

ECB: Electronic Code Book

DBMS: Database Management System

SSL: Secure Sockets Layer

HTTP: Hypertext Transmission Protocol

API: Application Programming Interface

LSB: Least Significant Bit

LVE: Last Variable Equation

MVP: Minimum Viable Product

PDF: Portable Document Format

CVE: Common Vulnerabilities and Exposures

SHA: Secure Hash Algorithm

FIFO: First-in First-out

SQL: Structured Query Language

ISP: Internet Service Provider

AES: Advanced Encryption Standard

PKCS: Public-Key Cryptography Standards

DNS: Domain Name System

TLS: Transport Layer Security

JDK: Java Deverlopment Kit

JSON: Javascript Object Notation

MSB: Most Significant Bit

DBMS: Database Management System

CSV: Comma-separated Values

DOCX: Office Open XML document

CBC: Cipher Block Chaining

TOTP: Time-based one-time password

ITIDA: Information Technology Industry Development Agency

Personal details and Work experience

About Morad

Morad is a computer science and mathematics enthusiast. He is interested in backend development, application security, statistical analysis, number theory, and theoretical computer science. He worked with a variety of programming languages, including Java and C++, which are heavily used in enterprise and large-scale applications. Morad is interested in scientific research, to unravel the mysteries of the universe using mathematical and computational sciences and application. You can contact him at Zelakolase@tuta.io. All of his coding projects are on GitHub at https://github.com/Zelakolase, and the LinkedIn profile is https://linkedin.com/in/0x250.

Work Experience

1. **ZTE Corporation, China** **Sept. 2019**
   Type: Letter of commitment
   Position: Cyber Security Researcher
   Location and/or work model: Remote
   Assessed the security of multiple network router models, such as H108N and H168N, and ZTE Web System. Helped issuing three security patches*.

2. **Zagazig Schools Digitalization Project 'ZSDP', Egypt** **Jan. 2022 – Jan. 2023**
   Type: Student Project
   Position: Leader and Backend Developer
   Location and/or work model: Remote
   Started and managed a student project to provide digitalization solutions for schools in Zagazig city. Succeeded to raise awareness about the benefits of a paperless government and the immense time and financial cost savings. Succeeded to develop a system to safely store documents with an issued IP certificate*.

3. **Technology Innovation and Commercialization Office – Zagazig University, Egypt** **Jun. 2023 – Sept. 2023**
   Type: Volunteer
   Position: Technical Member
   Location and/or work model: Hybrid, Shaibet an Nakareyah, Zagazig City, Al-sharqia Governorate 7120001
   Hosted workshops for children regarding Arduino circuit programming. Recommended funding programs for graduation projects to increase the office output. Participated in the 7th Cairo International Exhibition of Innovation.

7th Cairo International Exhibition of Innovation, Feb. 2023

*: Available in Supplementary Materials

Anonybox

Etymology: (Anony)mous Mail(box)

There exists multiple privacy-focused messaging and mailing applications, such as Signal or Telegram. Unfortunately, there is not much development or interest in building decentralized messaging and mailing apps while using E2EE, examples are limited to Zeronet, for instance.

Anonybox serves as the simplest decentralized mail service while using quantum-resistant encryption algorithms for both network communication and server data storage. It is a terminal-based server and client programs where you can host your own server and communicate with other mail servers, or connect as a client to an existing mail server. The information that can be sent is text only, no attachments or other features. Some might consider it as a disadvantage, but Anonybox is not intended for public use.

Anonybox benefits journalists and those who want to share highly secret text in a decentralized network, far from ISP or government monitoring systems, which involve packet interception techniques. This is especially benefical for those who live in countries with high government censorship and interception, such as Turkey, Egypt, or Syria.

Anonybox uses AES-ECB-PKCS5 Algorithm for symmetric encryption and decryption, and uses Diffie-Hellman Key Exchange Algorithm for the handshake between the server and client, to ensure sharing a secret key without actually revealing it to the public network. Anonybox focuses on software minimalism and code quality, in order to allow other developers to customize Anonybox to their liking, and make other versions of it. Anonybox is built from scratch, even the DBMS, and an internal naming server in order to replace conventional DNS server for Anonybox server domain resolve.

The decentralization of Anonybox comes from the ability to deploy personal servers. Just like the internet in the early days, you can register to an internal naming server, or connecting directly to the destination server, without having to connect to a central server.

GitHub Link: https://github.com/Zelakolase/Anonybox

SimplyWebServer

Etymology: Simply → Simple, Web Server

Very few minimal dynamic web server projects exist. Instead, the market uses popular web servers which are proven to be prone to multiple security vulnerabilities. Moreover, the local configuration of these popular web servers is not beginner-friendly. SimplyWebServer tries to be as simple as possible while providing all necessary functions for developers (eg. SSL/TLS, Static File Optimization, etc..).

SimplyWebServer is mainly targeted for small-sized to medium-sized projects. For example, internal systems, graduation projects, etc.. . The project allows them to peek through the internal server source code while understanding it, allowing them to modify any basic function of the web server to their usecase. Furthermore, you can compile your web application to a binary image using GraalVM. This allows your web application to be deployed on any target operating system easily, depending only on the C library of the system (effictively the core system calls).

SimplyWebServer is not just another project for those who like minimal codebases, it is also a perfect learning opportunity for any college student or a computer enthusiast, to learn how SimplyWebServer understands and parses HTTP requests from clients. The student learns every detail, from TCP connection details, to buffered reading and writing on the TCP socket, to how to deserialize the HTTP request to an object in memory with properties that you can work with. Furthermore, the student learns how to implement multi-threading in a safe way by limiting the number of active threads and setting a timeout for any request that wants to be processed while the active threads equal to the maximum number of threads allowed by SimplyWebServer.

In addition, LOGJAM and Client Renegotiation attacks are mitigated by modifying the JDK properties in runtime. As previously said, it does not only secure SimplyWebServer-built web applications, but it is a learning opportunity for others instead of searching about a mitigation by themselves for hours.

Since SimplyWebServer is made from scratch, every library and file is only dependent on core libraries, adding to the security and compatibility of the project. Additionally, the usage of advanced data structures serves as an example of real-world application, not just a proof of concept or as an example in a Udemy course. Below is an example for a simple web application.

GitHub Link: https://github.com/Zelakolase/SimplyWebServer

```
@Override                                                                   Overrides the abstract function in SimplyWebServer
public HashMap<String, Object> main(HashMap<String, String> headers, byte[] body) {
    HashMap<String, Object> response = new HashMap<>();
    response.put("mime", "text/html");                                      The type of response is HTML text

    if (headers.get("path").equals("/index.html")) {                        If the user requested the homepage, serve it
        response.put("body", HTMLCode.getBytes());
        response.put("code", HTTPCode.OK);                                  with status code 200 OK
    } else if (headers.get("path").equals("/api/login")) {                  If a request is received in the login API endpoint
        HashMap<String, String> LoginInfo = JSON.QHM(new String(body));     Convert the JSON of the POST request structure to a Key, Value Pair
        if (!(LoginInfo.containsKey("username") && LoginInfo.containsKey("password"))) {   If the request doesn't have the attributes 'username', 'password'
            response.put("body", "Invalid Request.".getBytes());
            response.put("code", HTTPCode.BAD_REQUEST);                     Return a 'bad request' response to the user
        } else {                                                            Else, the request is valid
            if (LoginInfo.get("username").equals(username)                  If the value of 'username' is equal to the correct username
                        && LoginInfo.get("password").equals(password)) {    and the value of 'password' is equal to the correct password
                response.put("body", JSON.HMQ(new HashMap<>() {{
                    put("status", "success");
                }}).getBytes());                                            Construct a JSON response of a success message
            } else {
                response.put("body", JSON.HMQ(new HashMap<>() {{
                    put("status", "failed");
                }}).getBytes());                                            Else, Construct a JSON response of a failed message
            }
            response.put("mime", "application/json");                       Either way, the message type is a JSON message
            response.put("code", HTTPCode.OK);                              with a successful response
        }
    } else {                                                                If the path requested is not the home page or the API endpoint
        response.put("body", "File not found".getBytes());
        response.put("code", HTTPCode.NOT_FOUND);                           Return a '404 File not found' response
    }
    response.put("isFile", "0");                                            Example: Custom static header
    response.put("CustomHeaders", new HashMap<String, String>() {{
        put("RandomInteger", String.valueOf(new Random().nextInt(10)));     Example: Random generation of header value in response
    }});
    return response;
}
```

Simuconomy

Etymology: (Simu)lation of E(conomy)

The dynamics of the free market economic model, and its relation with consumer behaviour has opened a new and interesting field of research. This field involves behavioural economics, econometrics, computational economics, and computer science. Advances in this field of research leads in more accurate real-world economic, financial, and monetary decisions.

Previous research in this field has indicated that there might be an optimal tax formula for income and corporate, such formulas are claimed to outperform the Saez formula of optimal labour income tax[1]. Unfortunately, most of these models are computationally extensive due to the usage of artifical intelligence, and do not make proper abstractions of real-world economies.

Simuconomy is one of the few agent-based models that does not use artificial intelligence. The author believes that self-correcting deterministic models are sufficient to replace artificial-intelligence-related methods. Simuconomy uses genetic algorithms to correct for market shock behaviour and optimize the economic stability on a computational level.

One of the innaccuracies that Simuconomy shares with other models is that all agents have some sort of production power. This is inaccurate since real-world agent production is determined by the corporation general productivity. Therefore, this model is only true if all agents are freelancers. However, I am planning to actively research methods to implement current hierarichal economic model in order to get more accurate results, and make proper abstractions to balance between computational irreducibility and real-world accuracy.

In general, designing economic simulation models and analyzing free market behaviour directs us towards more accurate economic decisions. For example, taxation on inheritence, monetary policies and Universal Basic Income, optimal price control policies, and more.

Simuconomy relies on genetic crossover and variability of agents' genetic code, in order to stabilize the entirety of the economic system by the concept of 'Survival of the fittest'. Every agent produces and consumes some units of food, and tries to optimize production to consumption ratio on an individual level.

The genetic makeup of each agent

1. Base Inflator Sensitivity: The rate of change of the price inflator. That gene changes ±1% each iteration in the agent's lifetime.

2. Base Supply Sensitivity: The rate of change of supply capacity.
3. Base Demand Sensitivity: The rate of demand capacity.

The assumptions of Simuconomy

1. There are no monopolies formed. When agents reproduce, their wealth decreases by 35%.
2. Any agent can take up to $10,000, before dying (ie. bankruptcy).

The Panic Coefficient

The panic coefficient is an indicator for the agent to increase production and/or decreases demand, or vice versa. The panic coefficient increases or decreases according to certain conditions.

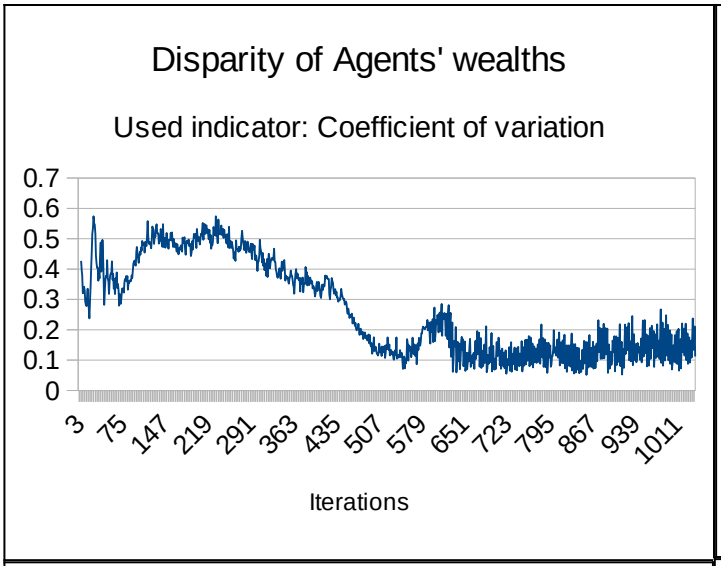
The next page will include an example

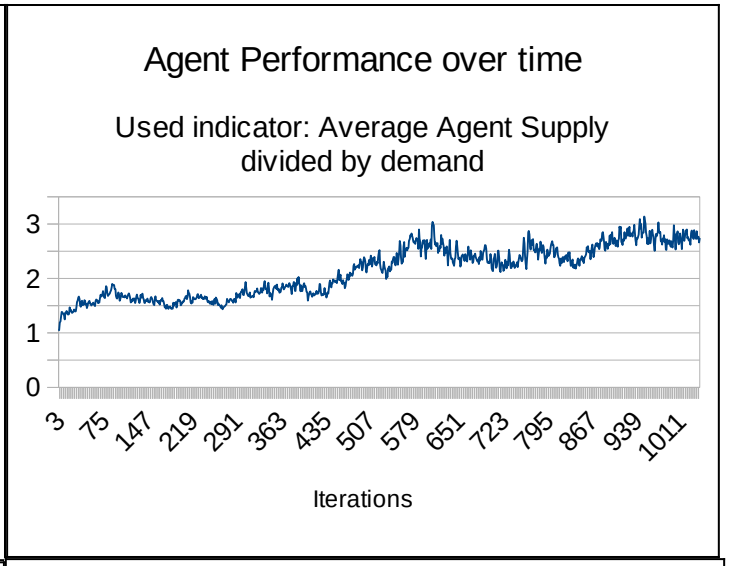
GitHub Link: https://github.com/Zelakolase/Simuconomy

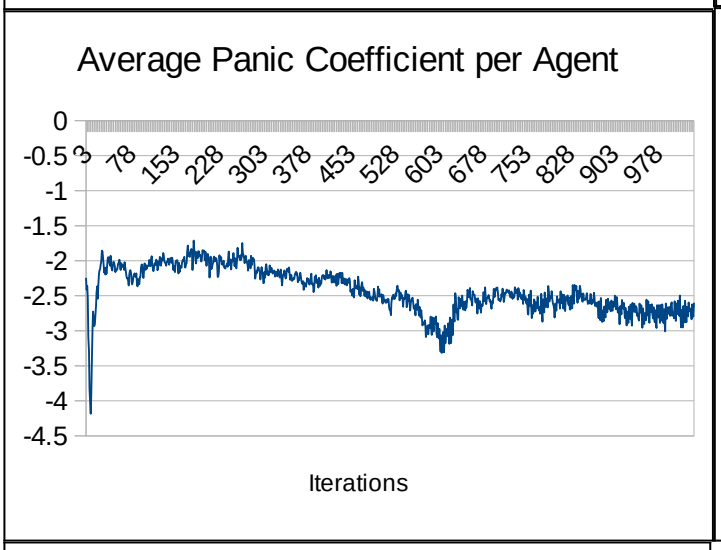
[1]: https://blog.salesforceairesearch.com/the-ai-economist

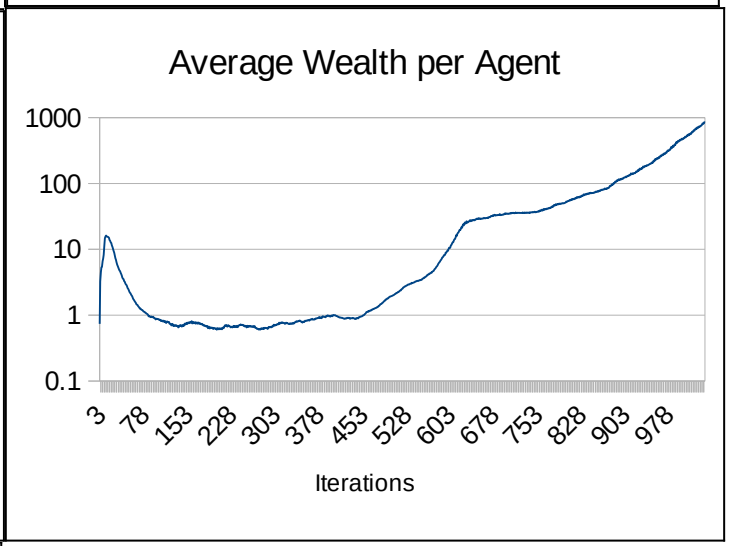
Example

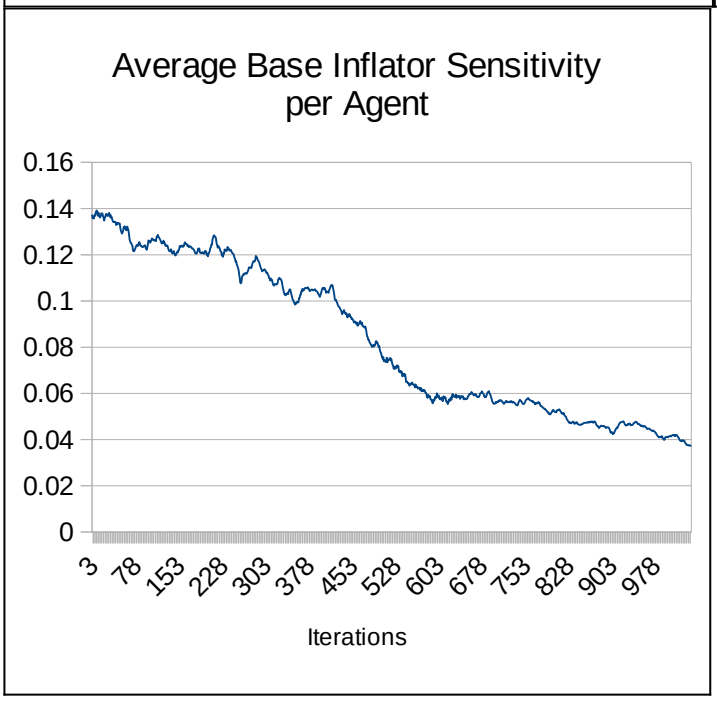
In this example, the population is capped at 500, starting population is 50, with inflation target of 0%, and genetic variation for each crossover up to 25%. The inflation control method is increasing/decreasing wealth of agents.











Conclusions

Based on the average panic coefficient progression over time, we can conclude that low market panic is the optimal market policy for all agents. The concept of wealth redistribution for the next generation of agents (ie. productive units) is beneficial for lower wealth disparity and overall wealth increment. Lower inflation rise is correlated for higher performing agents, correlation is evident between iteration 400 and 650.

MORAD ABDELRASHEED MOKHTAR ALI

CycleBruteforce 'Collatz Conjecture'

Collatz Conjecture is one of the most famous unsolved problems in number theory. Lothar Collatz introduced the conjecture in 1937. Since then, many generalizations were hypothesized in order to understand how to solve Collatz Conjecture. Assuming a sequence $T_n=\{(T_{n-1}|2,\frac{T_{n-1}}{2})\vee(T_{n-1}\nmid 2,3\times T_{n-1}+1)\}$, does this sequence always approach one?

Many papers in recent years picked up on the nature of dividing over two, and concluded a one series of numbers that always collapse to one, that is $2^n, n\in\mathbb{Z}$. Therefore, further expansion and discovery of other number series that collapse to one will bring the coverage of all whole numbers to 100%, solving the Conjecture.

My paper [DOI: 10.21203/rs.3.rs-2288122/v1], explains the nature of divisions by two, and how we can derive equations from this nature to get infinite cycles or diverging seeds (hence, the name of the research). The results are promising and will form the basis for the final solution. I have successfully constructed an infinite set of equations for the generalization of Collatz Conjecture, where the scope of the problem is narrowed to integer bruteforce for a multivariate equation.

A demo was published in the Wolfram Foundation Notebook Archive [https://notebookarchive.org/id/2023-07-47q58wm/], extending my work to the generalization of Collatz Conjecture $qX+c$. The general intuition is that T_n collapses to an odd number until it jumps to $3x+1$, this visual image creates a new perception of the problem. Therefore, it is possible to construct an equation to see how many divisions by two for every $3x+1$ jump can be made in order to collapse T_n to one. This is exactly what was discussed both in the paper and in the archived notebook.

In the repository, a java program was made and supplied with the original paper to allow others to easily develop and build upon my project. The program uses all CPU resources via multi-threading in order to speed up the calculation process. Two optimization techniques were used to speed it up:

1. The usage of square-and-multiply algorithm to calculate any $x^y.x, y\in\mathbb{Z}$ with fewer steps, instead of the conventional repeated multiplication algorithm.

2. The usage of bitshifting in calculating $2^n, n\in\mathbb{Z}$, which is a manipulation in the LSB and MSB of the binary representation of the number n.

Patterns were found in the calculation and detection of Collatz cycles and diverging seeds, these were programmed in the java program to allow it to scale to infinite-level cycles. Primarily, the two functions of tail and LVE equations described in the paper, are in *src/lib/Algo.java*.

The potential of this work is to detect cycles easily, instead of the conventional bruteforce methods. Therefore, Collatz Conjecture is no longer a mathematical problem, but a computational one. The future scaling of hardware resources globally will allow mathematicians to search for huge number ranges efficiently, and every second of execution without a result is a proof that Collatz Conjecture is true.

GitHub Link: https://github.com/zelakolase/CycleBruteforce

SparkDB

Etymology: Spark as for high speed, (D)ata (B)ase

For a long time, SQL has dominated the database management field in Computer Science and Engineering. As applications deal with more data and higher demand over time, new solutions were needed to decrease the latency of reading and writing on the database. This demand created a supply of memory-cache databases such as 'Redis'. Furthermore, multiple DBMSes were developed to tackle certain problems (eg. PostgresSQL, MongoDB, MariaDB, etc..).

A key disadvantage that was not addressed was the abstractness of DBMSes. This led to a new demand of modular and abstract DBMS that can be shaped for nearly any use case. This simplifies the process of choosing the "right" DBMS for your application, and allows for MVP development at low time cost. That was a key issue that SparkDB is trying to solve.

SparkDB is a bloatware-free DBMS, no unused core features. SparkDB is a merely advanced method to manipulate CSV files, which is an extremely powerful tool for data analysis and beyond. SparkDB is a local object in Java that allows you to navigate through the CSV structure. It is not a library, but a free-to-use one Java code file that you can copy and paste in your project, which is the simplest integration process possible.

I have written a guide in the code repository explaining how to use it, so I will not redo it here. Instead, I will focus on the inner workings of SparkDB and how it can be scaled for nearly every use case.

Every CSV has two main things: column titles, and values. SparkDB is essentially a highly optimized Hash Map with a key and value, such that the key is the column title, and the value is an array of values under the column title. It allows for data duplication and parsing from AES-encrypted CSV files. Then, you can hypothesize that every table is essentially a CSV file, and you can build JSON-to-CSV structure code, TCP interface, or any other modification by yourself. You can also shard data based on a prefix/suffix and use SparkDB as the navigator. In short, SparkDB is the core, building block for your customized use case, either a memory cache database, or a simple write-to-disk relational database.

One big advantage for SparkDB over Redis is that you are not limited to pairs of information, like a conventional hashmap. Instead, SparkDB takes it a step further, and allows you to have unlimited columns per table. Maximum rows for a single table is around 2.15 billion.

In order to get values, modify, or add data to a SparkDB object, you use function calls, not SQL-like queries. This is a much simpler and minimal process. By calling .toString() function, you translate the modified SparkDB memory structure to a CSV form that you can store into permanent disk.

Just like SQL-based solutions, you can make relationships between tables via a key, that you get to choose. The main difference is that you code relationships yourself, giving developers the full control over database and code behaviour.

GitHub link: https://github.com/NaDeSys/SparkDB

Electronic Document System

For a long time, Egypt has been suffering from high-cost low-efficiency spending. This issue exists across all sectors in Egypt. Furthermore, The cost of doing business or any process in government in general takes a lot of time, effort, and possible financial expense. As countries are moving forward with digitalization and paperless governments for efficient use of resources, such as United Arab Emirates, or Estonia, and as cyber attacks increase in intensity, a digital transformation movement is needed for Egypt to allow for budget savings, fast government processes, accountability, and lower cost of doing business.

Important note: This project is developed under Zagazig Schools Digitalization Project under the 'Work experience' section. This project is co-founded by three technical students and one non-technical student to promote digital transformation. Unlike other projects, the author 'Morad' does not claim that he coded all of the project. However, he administered it and was acting as both the software architect and backend developer of the project.

Current government efforts towards digital transformation have multiple issues, including not having a disaster recovery plan, depending on centralized architecture, and weak cyber security for systems. These issues do not only affect the economic situation but also the security of people's data. Furthermore, Egypt overspends in making agreements with technological companies to administer the digitalization process, which is both not efficient and not secure enough.

In light of all these issues, we have developed a maximum-security high-performance document management system for entities. The problems can be divided into three main categories:

Paper usage

The extensive usage of paper in the education sector *alone* increases the import volume by 3.66%, and adds an additional 750M+ EGP of cost on the education sector. Furthermore, paper demand and industry accelerates deforestation, air pollution, and is a significant contributor to GHG emissions, according to a paper published in 1999 [Subak, S., Craighill, A. The contribution of the paper cycle to global warming.]. It is also note-worthy that paper accounts for 26% of total waste at landfills worldwide. Paper can be damaged by various ways, including burning or cutting, adding to the risk of losing information.

Government Seal

Government seals on documents serve a purpose, which is verifing authenticity. Unfortunately, such seals can be forged, and it is hard to verify authenticity from the document author. Furthermore, the document can be edited after the seal, and you cannot trace back who sealed the document if there are multiple seal holders. These issues and more affect the achievability of the original purpose. Therefore, a digital replacement is required.

Submission Process

The cost of submitting a request, verifying a document, or any paper-involved process is typically slow. This issue makes scalability for population growth expensive, and the increased risk of losing information and inefficient searching. In addition, the cost of doing business rises, and economic growth decreases by a non-negligible percentage.

The solution is a simple, minimal, decentralized, and purpose-specific document management system. Our project has achieved all of these goals, and addressed every problem discussed. Each entity has its own storage and processing server, with its own unique ID system, that can be connected via a reverse proxy. Furthermore, such systems can be connected via local secured networks to upper management levels in order to request and exchange information faster than traditional on-site requests. The submission process is simplified between an author and a verifier (formerly known as the seal holder or owner), where the author sends the document to the verifier, who is a system user.

The government seal is replaced with a non-repeatable randomly-generated 9-digit code. These codes are unique for every document, which means that you can easily verify the authenticity of the document by comparing it to the secured non-modifiable version on the entity server. The 9-digit code can be used by any member of the public to get information about the document (eg. who wrote it, who verified it, and verification date), download the original version of the document, or compare the PDF/DOCX versions letter by letter, which can be automatically done by the system.

Electronic Document System does not use conventional software for the server (eg. nginx, apache, SQL-based solutions, etc..). Instead, the server is entirely built from scratch. We use SimplyWebServer for the HTTPS request handling and SparkDB for database management (with sharding), both projects were previously discussed. Therefore, the server cannot be attacked with any famous CVE attack.

Furthermore, we use AES 256-bit CBC PKCS5 symmetric encryption algorithm with additional SHA-3 512-bit hashing for verifiers' credentials, and the encryption algorithm alone for the databases, configurations, documents, and frontend files, to mitigate physical attacks. We implemented TOTP for verifiers' authentication for added security*. In order to maintain a reasonable size for the server, we limited the number of documents allowed on one server to 10 million, where the oldest document is removed for the newest document (ie. FIFO).

The communication between the client (eg. member of the public, or a verifier) and the server is secured by TLSv1.3 with two vulnerabilities fixed: Client Renegotiation and LOGJAM.

The software is licensed in Egypt as an intellectual property under deposit number 4123 in ITIDA, under the Egyptian Intellectual Property Protection law no. 86 of 2002.

The (en/de)cryption key of the databases, documents, configurations, verifiers' credentials, and frontend files are called 'Server Key'. It is non-modifiable. In addition, There is an offline configuration mode to add, modify, or delete users (ie. verifiers) on the system, it is not an online page due to security risks.

GitHub Link: https://github.com/Zelakolase/EDS-MOE

*: Upon the informal request of the Ministry of communications, Egypt

GCKit

Etymology: (G)raph (C)lustering Kit

Multiple algorithms have attempted to tackle a difficult problem, that is, detection of clusters in a network 'hereinafter called graph'. For the protein-protein interaction network case, network nodes are proteins and connections 'hereinafter called edges' are the interaction evidence and strength between each protein.

Unfortuantely, using unsupervised learning algorithms for cluster and community detection is not intuitive, since proteins need to be plotted on a 2D chart. Therefore, a walk-based algorithm is needed to efficiently tackle this issue. Multiple algorithms are currently used in the pharmachological field, such as Markov-Clustering algorithm, the louvain method, Infomap, k-means, and CliXO.

The only satisfiable high-performing algorithm is CliXO. A new algorithm is needed to tackle this issue from scratch. Progress in cluster detection for protein-protein interaction networks opens a huge opportunity in accurately categorizing diseases by the chain of protein irregularties and reactions that caused the disease, and the development of new drugs that can save billions of lives worldwide and solve chronic diseases' root causes. I was introduced to the problem by Dr. Mahmoud Elbatreek in the faculty of pharmacy, Zagazig University during my work in the university and the fruitful discussions that we had on the integration of mathematical and computational sciences with pharmachology.

Proteins are divided into 3 categories, heads, normal proteins, and transition proteins. Head proteins can be categorized into a single cluster, transition proteins cannot be categorized into a single cluster, and normal proteins are half-way through the categorization. Clusters are heirarchial, where for a certain cluster, there is a head node 'master' and normal nodes 'members, or slaves'. A protein can join multiple clusters at once (including transition proteins).

We concluded that GCKit outperformed every popular algorithm except CliXO with a small margin, all of the generated clusters were significantly enriched with a p-value < 1.0e-16 (for Amyotrophic Lateral Sclerosis). This is a promising result for disease modulation and further pharamchological discoveries. In addition, we try to follow ISO-25010 for code quality in order to allow developers to customize our work to their use cases, and to achieve high maintainability.

Note: The description provided may not fully reflect the work, the author 'Morad' has stopped updating the public codebase since August 2023, due to internal algorithmic development with further significant results that are in pre-clinical testing and paper publication. The author apologizes for the inability to provide updated information.

GitHub link: https://github.com/Zelakolase/GCKit

Supplementary Material

List of CVE IDs

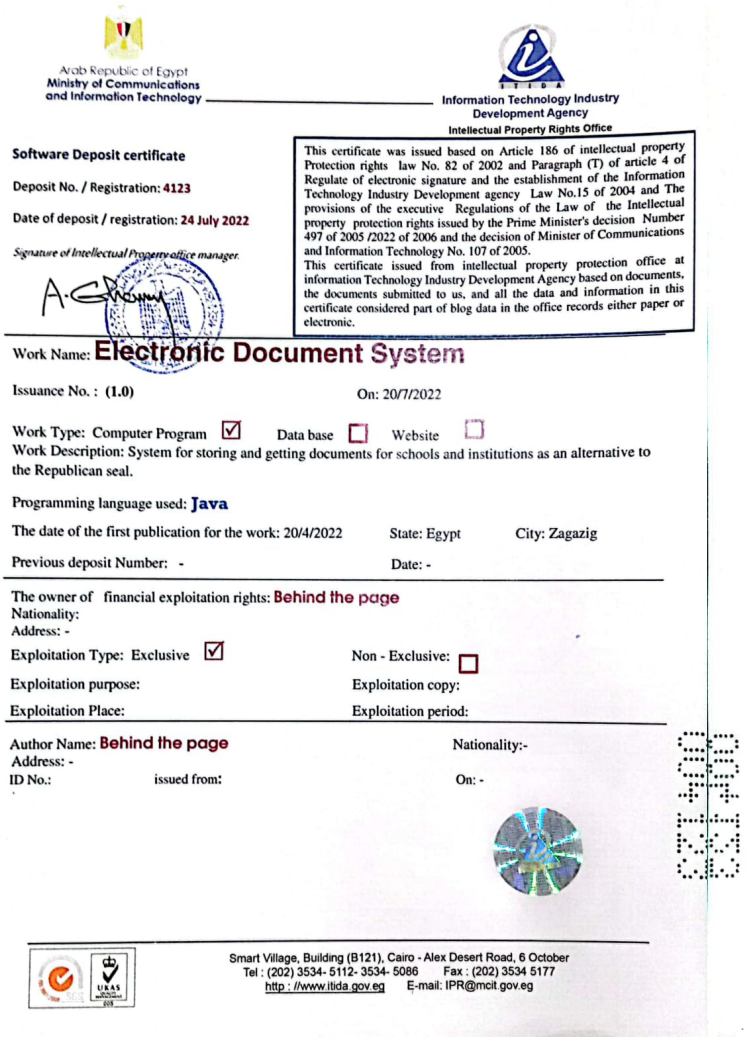
CVE ID	Announcement Link
CVE-2021-21729	https://support.zte.com.cn/support/news/NewsDetail.aspx?newsId=1014904
CVE-2021-21730	https://support.zte.com.cn/support/news/LoopholeInfoDetail.aspx?newsId=1014864
CVE-2019-3420	http://support.zte.com.cn/support/news/LoopholeInfoDetail.aspx?newsId=1011802

List of Certificates

Certificate name	Certificate ID	Issuer
Research and Proposal Writing in the sciences	kDa1EwWa80	INASP
Web Application Penetration Testing	C-0b2d1b3c23-1d0c451	Cybrary
Social Engineering and Manipulation	C-0b2d1b3c23-2ed9ba	
Phishing	C-0b2d1b3c23-1e6a3e480	
Intro to Infosec	C-0b2d1b3c23-faaf1a8e9	
End user: Network Security	SC-0b2d1b3c23-420d47	
Cybersecurity fundamentals	C-0b2d1b3c23-413d9a3	
Corporate Cybersecurity Management	C-0b2d1b3c23-123bddd8	
Computer hacking and forensics	C-0b2d1b3c23-8a34401	
Chief Information Security Officer	C-0b2d1b3c23-7ac6511d	

IP Certificate – English Version

FRONT



Arab Republic of Egypt
Ministry of Communications and Information Technology

Information Technology Industry Development Agency
Intellectual Property Rights Office

Software Deposit certificate

Deposit No. / Registration: **4123**

Date of deposit / registration: **24 July 2022**

Signature of Intellectual Property office manager.

This certificate was issued based on Article 186 of intellectual property Protection rights law No. 82 of 2002 and Paragraph (T) of article 4 of Regulate of electronic signature and the establishment of the Information Technology Industry Development agency Law No.15 of 2004 and The provisions of the executive Regulations of the Law of the Intellectual property protection rights issued by the Prime Minister's decision Number 497 of 2005 /2022 of 2006 and the decision of Minister of Communications and Information Technology No. 107 of 2005.
This certificate issued from intellectual property protection office at information Technology Industry Development Agency based on documents, the documents submitted to us, and all the data and information in this certificate considered part of blog data in the office records either paper or electronic.

Work Name: **Electronic Document System**

Issuance No. : **(1.0)** On: 20/7/2022

Work Type: Computer Program ☑ Data base ☐ Website ☐
Work Description: System for storing and getting documents for schools and institutions as an alternative to the Republican seal.

Programming language used: **Java**

The date of the first publication for the work: 20/4/2022 State: Egypt City: Zagazig

Previous deposit Number: - Date: -

The owner of financial exploitation rights: **Behind the page**
Nationality:
Address: -

Exploitation Type: Exclusive ☑ Non - Exclusive: ☐

Exploitation purpose: Exploitation copy:

Exploitation Place: Exploitation period:

Author Name: **Behind the page** Nationality:-
Address: -
ID No.: issued from: On: -

Smart Village, Building (B121), Cairo - Alex Desert Road, 6 October
Tel : (202) 3534- 5112- 3534- 5086 Fax : (202) 3534 5177
http : //www.itida.gov.eg E-mail: IPR@mcit.gov.eg

BACK

Authors & owners of financial exploitation rights for the program

"Electronic Document System"

Name	ID Number	Address	Nationality
Andrew Roushdy Mourad Youssef	[redacted] Issued from: Zagazig first On: 01/2021	[redacted] Sharkia Governorate	Egyptian
Mourad Abdel Rashid Moukhtar Ali	[redacted] Issued from: Zagazig second On: 03/2021	[redacted] Sharkia Governorate	Egyptian

IP Certificate – Arabic Version

FRONT

جمهورية مصر العربية
وزارة الاتصالات وتكنولوجيا المعلومات

هيئة تنمية صناعة تكنولوجيا المعلومات
مكتب حماية حقوق الملكية الفكرية

شهادة إيداع وتسجيل مصنف حاسب آلى

رقم الإيداع / التسجيل
٠٠ ٤١٢٣

استخرجت هذه الشهادة بناءاً على المادة رقم (١٨٦) من قانون حماية الحقوق الملكية الفكرية رقم ٨٢ لسنة ٢٠٠٢ ، والفقرة (ط) من المادة رقم (٤) من قانون تنظيم التوقيع الإلكتروني وإنشاء هيئة تنمية صناعة تكنولوجيا المعلومات رقم ١٥ لسنة ٢٠٠٤ ، وعلى أحكام اللائحة التنفيذية لقانون حماية حقوق الملكية الفكرية الصادرة بقرارى السيد رئيس مجلس الوزراء أرقام ٤٩٧ لسنة ٢٠٠٥ ، ٢٢٠٢ لسنة ٢٠٠٦ وعلى قرار السيد وزير الإتصالات وتكنولوجيا المعلومات رقم ١٠٧ لسنة ٢٠٠٥

وقد صدرت هذه الشهادة عن مكتب حماية حقوق الملكية الفكرية بهيئة تنمية صناعة تكنولوجيا المعلومات بناءاً على المستندات والوثائق المقدمة إلينا ، وتعتبر كل البيانات والمعلومات الواردة بها جزء من البيانات المدونة بسجلات المكتب الورقية والإلكترونية .

تاريخ الإيداع / التسجيل : ٢٤ يوليو ٢٠٢٢

مدير مكتب
حماية حقوق الملكية الفكرية

تاريخ الإصدار : ٢٠٢٢/٠٧/٢٤

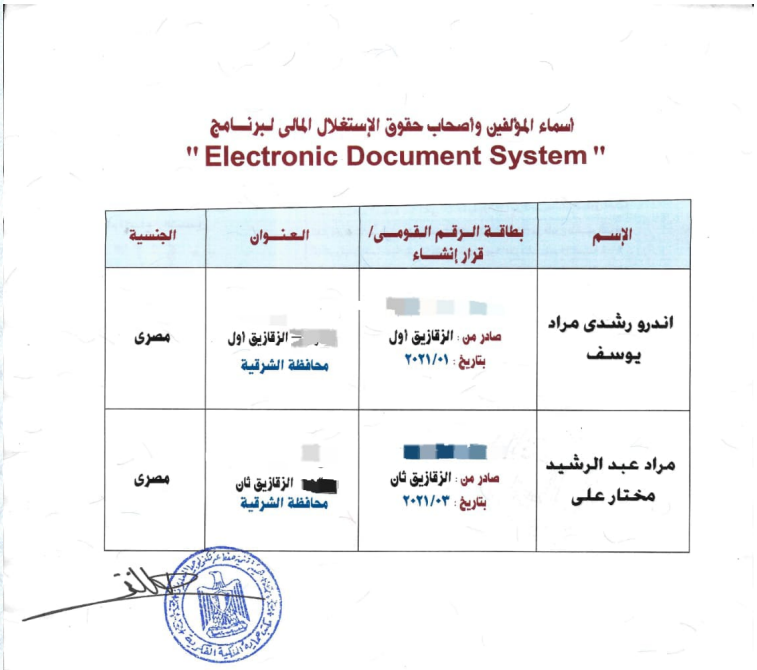
Electronic Document System

إسم المصنف :
رقم الإصدارة : (1.0) تاريخها : ٢٠٢٢/٧/٢٠
طبيعة المصنف : ☑ برنامج حاسب آلى ☐ قاعدة بيانات ☐ موقع على الإنترنت
وصف المصنف : نظام لتخزين وجلب المستندات للمدارس والمؤسسات كبديل للختم الجمهورى.
لغة البرمجة المُستخدمة: Java.
تاريخ النشر الأول للمصنف : ٢٠٢٢/٤/٢٠ الدولة : مصر المدينة : الزقازيق
رقم الإيداع السابق : — بتاريخ : —

إسم صاحب حقوق الإستغلال المالى: خلف الصفحة الجنسية : —
العنوان : —
طبيعة الإستغلال : ☑ إستثمارى ☐ غير إستثمارى
غرض الإستغلال : — صورة الإستغلال : —
مكان الإستغلال : — مدة الإستغلال : —

إسم المؤلف : خلف الصفحة الجنسية : —
العنوان : —
إثبات شخصية رقم : — صادر من : — بتاريخ : —

BACK



أسماء المؤلفين وأصحاب حقوق الإستغلال المالى لبرنامج
" Electronic Document System "

الإسم	بطاقة الرقم القومى / قرار إنشاء	العنوان	الجنسية
اندرو رشدى مراد يوسف	صادر من : الزقازيق أول بتاريخ : ٢٠٢١/٠١	الزقازيق أول محافظة الشرقية	مصرى
مراد عبد الرشيد مختار على	صادر من : الزقازيق ثان بتاريخ : ٢٠٢١/٠٣	الزقازيق ثان محافظة الشرقية	مصرى