**Java project 5**

# Prompt: Start

I am an undergraduate student that studies information technology. I do not know much about programming and would like to make a Java project to help build my knowledge. Can you be an assistant to help me make a project. I will provide more information in later prompts. Also clarify throughout the process if what was outputted okay. Just make a small, short response.

# Prompt: Context

I am nearly going to graduate and need to do 1 more advanced IT subject. I was advised to take a subject 1 semester as I couldn’t take a leave of absence.

“If you pass your course in semester 2, 2024, you will be left with 1 more advanced IT option course. You can choose either of

Cloud Security or Blockchain fundamentals for this last course. Here are the course guides:

• Cloud Security: http://www1.rmit.edu.au/courses/038407

• Blockchain fundamentals: http://www1.rmit.edu.au/courses/INTE26272450

You mentioned that you want to take LoA for semester 1, because you want to take System Deployment & Operations (COSC2759).”

- Shekhar Kalra

Here is the course information for Cloud Security, one of the courses I need to do enclosed in brackets:

{

Course Title: Cloud Security

Credit Points: 12.00

Course Description

Cloud security is an evolving sub-domain of computer security, network security, and, more broadly, information security. It refers to a broad set of technologies deployed to protect data, applications, and the associated infrastructure of cloud computing.

This course provides a practical survey of both the principles and practice of cloud security. The emphasis of this course is on the underlying principles and techniques of cloud security with examples of how they are applied in practice.

Please note that if you take this course for a bachelor honours program, your overall mark in this course will be one of the course marks that will be used to calculate the weighted average mark (WAM) that will determine your award level. This applies to students who commence enrolment in a bachelor honours program from 1 January 2016 onwards.

Program Learning Outcomes

This course is an option course and not a core course, so that it does not need to contribute to the Program Learning Outcomes.

Course Learning Outcomes

Upon successful completion of this course you should be able to:

1. Demonstrate knowledge of cloud security principles and mechanisms

2. Demonstrate computer programming and configuration skills required to develop a cloud security infrastructure

3. Identify cloud security weaknesses by recognising and discovering threats and vulnerabilities to cloud computing

4. Problem solve how to fix cloud security weaknesses and mitigate security threats to cloud computing

5. Demonstrate knowledge and skills to prepare for industry cloud security certificate exams e.g. CCSK, CCSP

6. Communicate clearly and effectively using the technical language of the field correctly

Overview of Learning Activities

The learning activities included in this course are:

• Lectorials: Key concepts will be explained in lectorials in which course material will be presented and the subject matter will be illustrated with demonstrations and examples;

• Tutorial sessions focus on analysing and problem solving of given scenarios, such as security analysis and secure solutions;

• Computer laboratory sessions provide practices in the application of security techniques.

}

Here is information from one of tutors about what I should to prepare for the subject enclosed in brackets.

{

The programming tasks for this course are expected to be completed in Java or JavaScript. Generally speaking, and without going into too much detail, we use a programming language to apply a cryptographic concept, where the objective is to transform data into a form that appears random, but the original data can be recovered.

As a simplified example of this idea, consider that we have a character string like "Hello World!" and the task is to transform each character, such that the transformation can be reversed. For this task, the programming constructs we would need to use are:

• String handling: obtaining individual characters.

• Basic data types: understanding the relationship between character and integer data types.

• IF statement: can be used to handle characters like '!' and the space character as special cases.

• FOR loop: need a way to iterate over all characters in a string.

• Arithmetic: need to use a mathematical operation that can be reversed.

These constructs are found in either Java or JavaScript. In the case of JavaScript, we also need an API that allows us to get input from a webpage to make the implementation more interactive (that is, the Document Object Model that represents the HTML document). We typically avoid the use of language features like polymorphism or exception handling, because this can distract from the task at hand. For more advanced algorithms, we make use of programming libraries, which are provided as needed.

}

Does this make sense? Just summarise what I provided concisely.

# Answer: Context

Here’s a concise summary:

You need to complete one final advanced IT course, either **Cloud Security** or **Blockchain Fundamentals**, to graduate. You prefer **Cloud Security**, which involves learning about securing cloud infrastructure, identifying threats, fixing vulnerabilities, and preparing for industry certifications (e.g., CCSK, CCSP).

The programming component focuses on applying cryptographic concepts using **Java** or **JavaScript**, involving:

* String handling, loops, and arithmetic for reversible data transformations.
* Basic programming without advanced features like polymorphism.
* JavaScript implementations may involve interactive input via the DOM.

Your preparation should focus on basic programming skills and cryptographic principles. Does this summary align with your understanding? 😊

# Prompt: Selected Project

Yes. Here is a project I want to cover as part of my prepartion:

5.Secure Cloud File Storage Emulator

Difficulty: Hard

Goal: Create a simple program to encrypt a file before uploading it to a simulated cloud directory (use local storage as the "cloud"). Include options to decrypt and download the file.

Skills Covered:

- File I/O

- Encryption and decryption (e.g., AES)

- Basic command-line interface

Now give me details of the project in this order: A summary, a deadline of the project (date), explain step by step what the project is and how it works with examples (do NOT give the code as I want to figure it out myself). The project should be expected to take 4 weeks.

# Answer: Selected Project

**Summary**

The **Secure Cloud File Storage Emulator** is a project that simulates a secure cloud storage system. It encrypts files before storing them locally (acting as the cloud) and allows users to decrypt and retrieve files when needed. The goal is to practice file handling, encryption/decryption techniques (e.g., AES), and creating a basic command-line interface.

**Deadline**

**4 weeks from today:** February 6, 2025.

**Step-by-Step Project Plan**

**Week 1: Research and Setup**

1. **Understand the Key Concepts**:
   * Research AES encryption: How to encrypt and decrypt files securely.
   * Learn about file handling in Java: Reading, writing, and updating files.
2. **Set Up Your Workspace**:
   * Install Java (or ensure it’s updated).
   * Set up your IDE (e.g., IntelliJ IDEA, Eclipse).
   * Organize folders for the project, e.g., /input, /cloud-storage.

**Week 2: Build the Encryption and Decryption Mechanisms**

1. **File Encryption**:
   * Plan how to take an input file and encrypt its contents using AES.
   * Use a secret key for encryption and ensure it is securely stored.
   * Example:  
     Input: "Hello, World!"  
     Encrypted: "3a45b6d..."
2. **File Decryption**:
   * Plan a process to decrypt an encrypted file using the same key.
   * Verify the decrypted content matches the original.
   * Example:  
     Input: "3a45b6d..."  
     Decrypted: "Hello, World!"

**Week 3: Simulate Cloud Storage with File I/O**

1. **Directory Structure**:
   * Create a folder on your local system (e.g., /cloud-storage) to simulate the cloud.
2. **Upload Files**:
   * Write the encrypted file to the /cloud-storage directory.
   * Store metadata (e.g., filename, encryption key) if needed.
3. **Download Files**:
   * Retrieve files from /cloud-storage and decrypt them to their original form.

**Week 4: Create a Command-Line Interface (CLI)**

1. **Interface Design**:
   * Use the command line to interact with the program.
   * Provide options such as:
     + Encrypt and upload a file (encrypt-upload)
     + Decrypt and download a file (decrypt-download)
2. **Example Usage**:
   * **Upload a file**:

java SecureCloud encrypt-upload file.txt

Output: "File encrypted and uploaded successfully!"

* + **Download and decrypt a file**:

java SecureCloud decrypt-download file.txt

Output: "File decrypted and saved locally!"

**End Goal**

By February 6, 2025, you’ll have a working emulator that encrypts files, stores them locally as the "cloud," and allows secure retrieval through a CLI. Let me know how you’d like me to clarify further! 😊

# Prompt: Use Case

I don't understand the project. Can you give me a use-case of how the project would work and at each stage?

# Answer: Use Case

Sure! Let me walk you through a **real-world use case** of how the **Secure Cloud File Storage Emulator** works, breaking it into stages:

**Use Case Scenario**

**You are a user who wants to securely store a confidential document ("passwords.txt") in the cloud and retrieve it later.**

**Stage 1: File Upload (Encryption and Storage)**

1. **Initial Input**:  
   You have a file called passwords.txt containing sensitive data like:

Gmail: myemail@gmail.com | Password: 123456

Bank: bankuser | PIN: 7890

1. **Action**:  
   Run the program from the command line with an "upload" option:

java SecureCloud encrypt-upload passwords.txt

1. **Processing**:
   * The program reads the contents of passwords.txt.
   * It generates an **encryption key** (a secure string) to encode the file using AES encryption.
   * The encrypted data might look like this:

7GdhU3k9mNpOq2e1Rf...

* + The encrypted file is saved in a simulated "cloud" folder, e.g., /cloud-storage/passwords.enc.

1. **Output**:  
   The program informs you:

File "passwords.txt" encrypted and uploaded successfully to the cloud.

**Stage 2: Retrieving the File (Decryption and Download)**

1. **Need**:  
   You now want to retrieve and decrypt the passwords.txt file.
2. **Action**:  
   Run the program with a "download" option:

java SecureCloud decrypt-download passwords.txt

1. **Processing**:
   * The program locates the encrypted file in the /cloud-storage directory (passwords.enc).
   * It uses the **encryption key** to decrypt the file, recovering the original data:

yaml

Copy code

Gmail: myemail@gmail.com | Password: 123456

Bank: bankuser | PIN: 7890

* + The decrypted file is saved to your local directory as passwords-decrypted.txt.

1. **Output**:  
   The program informs you:

arduino

Copy code

File "passwords.txt" decrypted and saved as "passwords-decrypted.txt".

**Stage 3: Program Workflow Example**

1. **Encrypt and Upload**:
   * Input: passwords.txt
   * Encrypted File Stored: /cloud-storage/passwords.enc
2. **Decrypt and Download**:
   * Input: /cloud-storage/passwords.enc
   * Output: passwords-decrypted.txt (restores original content)

**Why is This Useful?**

* You securely store files in a "cloud" directory without exposing raw data.
* Even if someone accesses the /cloud-storage directory, the files are encrypted and meaningless without the key.
* It mimics a real-world secure file storage system, like Google Drive or Dropbox, with encryption.

Does this use case help clarify? 😊

# Prompt: Clarification on stages

I sort of understand. There are 2 stages. The first is uploading. The program first searches for the file we want to encrypt (in this case passwords.txt). Next an encryption key is generated and saved. The second stage would be decrypting the file. Just to clarify is the file hardcoded or do I select it?

# Answer: Clarification on stages

Great question! The file is **not hardcoded**—you get to **select it**.

Here’s how it works step by step in each stage:

**Stage 1: File Upload (Encryption and Storage)**

1. **You Select the File**:
   * When running the program, you specify the file you want to encrypt and upload. For example:

bash

Copy code

java SecureCloud encrypt-upload passwords.txt

* + Here, you’ve chosen passwords.txt, but you could specify any file (e.g., notes.docx, image.jpg).

1. **What Happens Next?**
   * The program locates the file using the name/path you provided (passwords.txt in this example).
   * If the file doesn’t exist or the name is incorrect, the program will alert you with an error message.

**Stage 2: File Retrieval (Decryption and Download)**

1. **You Select the Encrypted File**:
   * Similarly, when retrieving the file, you specify which encrypted file to decrypt. For example:

java SecureCloud decrypt-download passwords.txt

* + The program looks for the corresponding encrypted version (e.g., /cloud-storage/passwords.enc).

1. **How Does It Match Files?**
   * The program uses the original filename you provided (passwords.txt) to locate the encrypted file.
   * It decrypts this file and saves it locally as passwords-decrypted.txt.

**Why Is This Better?**

By letting you select the file, the program becomes flexible—you can use it with **any file you need to secure**, not just a hardcoded one.

Does this explanation make it clearer? 😊

# Prompt: Is using JFileChooser allowed?

When locating the file, is it okay to use JFileChooser?

# Answer: Is using JFileChooser allowed?

Yes, absolutely! Using JFileChooser is a great choice for file selection in a Java program. It provides a **graphical user interface** (GUI) for browsing and selecting files, which can make your program more user-friendly.

**How It Would Work**

Here’s how you might integrate JFileChooser into your project:

1. **Stage 1 (Encryption)**:
   * Instead of typing the file name in the command line, you could launch a JFileChooser dialog to allow the user to browse and select a file for encryption.
   * Example Workflow:
     + The user clicks a "Select File to Encrypt" button.
     + The JFileChooser dialog opens, and the user picks a file.
     + The program encrypts and uploads the selected file.
2. **Stage 2 (Decryption)**:
   * Similarly, when decrypting, the user can select the encrypted file from a JFileChooser dialog.
   * Example Workflow:
     + The user clicks a "Select Encrypted File to Decrypt" button.
     + The JFileChooser dialog opens, and the user picks the encrypted file.
     + The program decrypts it and saves the output.

**Benefits of Using JFileChooser**

* **User-Friendly**: No need to type file paths—users can visually browse their file system.
* **Error Prevention**: Avoids issues like typos in file names or paths.
* **Flexibility**: Works well for desktop applications where GUIs are preferred.

**Example Integration Idea**

* **Command-Line Version**: You can still allow users to specify file names through the command line if they prefer.
* **GUI Version**: Offer a JFileChooser option for those who want a more interactive experience.

Would you like to explore how to integrate JFileChooser, or do you prefer to stick to command-line file selection for simplicity? 😊

# Prompt: Cloud storage

No you don't need to explore how to integrate JFileChooser or command line file selection. I'm wondering for the cloud storage directory do I just create a folder locally (E.g. cloud) ?

# Answer: Cloud storage

Yes, creating a local folder (e.g., named **cloud**) is a perfectly fine way to simulate the "cloud storage directory." This keeps things simple while allowing you to emulate cloud storage functionality.

**How It Works**

1. **Create a Local Folder**:
   * You can manually create a folder (e.g., cloud) in your project directory or another location on your machine.
   * Alternatively, your program can automatically create this folder if it doesn’t already exist using Java’s File API. For example:
     + **Folder Path**: ./cloud/ (relative to your project directory).
     + **Absolute Path**: C:/Users/YourName/Documents/cloud/.
2. **Store Encrypted Files**:
   * When encrypting and "uploading" a file, save the encrypted version inside this folder with a unique or descriptive name. For example:
     + Original file: passwords.txt.
     + Encrypted file: ./cloud/passwords.enc.
3. **Retrieve Files from the Folder**:
   * When decrypting a file, your program will look in the cloud folder to locate the encrypted file and perform the decryption process.

**Why Use a Local Folder for Cloud Simulation?**

* **Simplicity**: No need to set up actual cloud services while still mimicking the concept of cloud storage.
* **Flexibility**: You can test the encryption/decryption functionality locally without needing an internet connection.
* **Organization**: Storing all encrypted files in one dedicated folder makes them easy to locate and manage.

**Things to Consider:**

* **Folder Naming and Location**: Use a consistent name (e.g., cloud) and ensure it’s in an accessible location. If users might run your program in different environments, consider asking them where to create the folder.
* **File Naming**: Use descriptive and unique file names to prevent overwriting files (e.g., append a timestamp or hash to the filename).
* **Error Handling**: Check if the cloud folder exists. If it doesn’t, create it automatically to avoid errors.

Does this approach align with what you were thinking? 😊