**JAVA PROJECT 3: PROMPT**

# PROMPT

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.

**RESPONSE**

Of course! I'd be happy to assist you in building a Java project step by step. Just let me know your idea or what kind of project you'd like to create, and we can start from there. I'll also clarify each step as we go. 😊

# 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 concisely summarise what I have said.

**PROMPT**

I already generated a plan. I plan to do this project next: Password Hashing Simulator

Difficulty: Moderate

Goal: Build a program that simulates password hashing using a cryptographic hash function like SHA-256 (use Java’s built-in libraries for hashing).

o Skills Covered:

 Using external libraries (java.security package)

 Reading user input

 Understanding hash algorithms

**Answer**

Great choice! A **Password Hashing Simulator** is an excellent project to strengthen your understanding of cryptographic concepts and Java programming. Here’s a simple breakdown of how we can approach it:

**Steps:**

1. **Understand the Basics**:
   * Research SHA-256 and why hashing is used for passwords.
   * Learn about the java.security.MessageDigest class.
2. **Set Up Your Project**:
   * Create a new Java project.
   * Include comments to document your work as you go.
3. **Core Features**:
   * **User Input**: Prompt the user to enter a password.
   * **Hashing**: Use the MessageDigest class to hash the password with SHA-256.
   * **Output**: Display the original password and its hashed value.
4. **Optional Enhancements**:
   * Add a salt to make the hash more secure.
   * Simulate storing and verifying passwords.

Would you like me to create a project outline or write a starter code snippet for you?

**PROMPT**

Create a project outline and details of the project. The project should take 2 weeks. Also list the start date (today) and the due date (2 weeks from now) at the top under the summary. Also include examples outlining step by step hwo the project should work.

**Answer**

**Password Hashing Simulator Project**

**Start Date:** December 12, 2024  
**Due Date:** December 26, 2024

**Summary**

The Password Hashing Simulator project is designed to simulate how password hashing works using the SHA-256 cryptographic hash function. The project will involve reading user input, applying the hash algorithm, and displaying the hashed output. Additional features like adding salts and verifying passwords will provide further learning opportunities.

**Project Outline**

**Week 1: Understanding Basics and Core Implementation**

1. **Day 1-2: Research**
   * Study cryptographic hash functions (focus on SHA-256).
   * Understand Java's MessageDigest class from the java.security package.
2. **Day 3-4: Basic Setup**
   * Create a new Java project.
   * Set up a main class and prepare to take user input using Scanner.
3. **Day 5-7: Core Implementation**
   * Implement the password hashing functionality using MessageDigest.
   * Display both the original password and the hashed value in a readable format.
   * Test the program with different passwords.

**Week 2: Enhancements and Finalization**

1. **Day 8-10: Adding Salts**
   * Modify the program to generate a random salt for each password.
   * Hash the password + salt combination.
   * Store the salt and hashed password for later use.
2. **Day 11-12: Password Verification**
   * Implement a feature to verify a password by hashing the input and comparing it to a stored hash.
   * Test with correct and incorrect password inputs.
3. **Day 13: Documentation**
   * Write clear comments explaining your code.
   * Create a README file with project details, examples, and instructions.
4. **Day 14: Final Testing and Submission**
   * Test the program thoroughly for edge cases (e.g., empty passwords, special characters).
   * Ensure all requirements are met.
   * Submit the project.

**Detailed Project Flow (Step by Step)**

**Step 1: User Input**

* **Prompt:** "Enter a password to hash: "
* **Input Example:** mySecurePassword123

**Step 2: Generate Hash**

* Use MessageDigest to hash the input using SHA-256.
* Output Example:

Original Password: mySecurePassword123

Hashed Password: 5e884898da28047151d0e56f8dc6292773603d0d6aabbdd2f88f76e8b066d0a8

**Step 3: Add Salt (Optional Enhancement)**

* Generate a random salt using SecureRandom.
* Combine the salt with the password before hashing.

**Output Example:**

Original Password: mySecurePassword123

Salt: XyZ12

Hashed Password: d9f9a9775dc7e3309f52e4af6eec8e9d88b0833d8fd324038edf1e95c1501021

**Step 4: Verify Password (Optional Enhancement)**

* Prompt the user to enter the password again for verification.
* Hash the input with the stored salt and compare it with the stored hash.

**Output Examples:**

Correct Password:

Password Verified Successfully!

Incorrect Password:

Password Verification Failed.