

Security Toolkit: Interface & Flow

1. Program Start When the program is executed using:

```
go run *.go
```

The application starts and displays the main menu interface:

```
+-----+
|      Security Toolkit App      |
+-----+
| 1. Cipher Module               |
| 2. File Integrity Checker      |
| 0. Exit                       |
+-----+
Choose an option:
```

2. Option 1: Cipher Module Flow Step 1 – Select Cipher Module:

```
1
```

Menu displayed:

```
--- Cipher Module ---
1. Encrypt Message
2. Decrypt Message
Choose an option:
```

Step 2 – Encrypt Message Flow:

```
1
Enter message: HELLO WORLD
Enter key (number of rails): 3
```

Code flow:

```
main.go → encryptRailFence() → display result
```

Output:

Encrypted Message: HOREL OLLWD

Step 3 – Decrypt Message Flow:

2
Enter message: HOREL OLLWD
Enter key (number of rails): 3

Code flow:

main.go → decryptRailFence() → display result

Output:

Decrypted Message: HELLO WORLD

1. Option 2: File Integrity Checker Flow Step 1 – Select File Integrity Checker:

2

Menu displayed:

--- File Integrity Checker ---
1. Generate SHA-256 Hash
2. Verify File Integrity
Choose an option:

Step 2 – Generate Hash Flow:

1
Enter file path: report.txt

Code flow:

main.go → computeSHA256() → display hash

Output:

SHA-256 Hash:
a7c3f1b2e9...

Step 3 – Verify File Flow:

2
Enter file path: report.txt
Enter known hash: a7c3f1b2e9...

Code flow:

main.go → verifyFile() → computeSHA256() → compare hashes

Output (unchanged file):

✓ File is intact

Output (modified file):

✗ File has been modified

1. Exit Program Flow

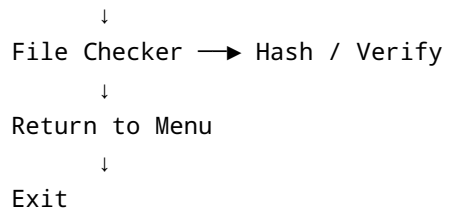
0

Output:

Exiting Security Toolkit. Goodbye!

2. Simple Flow Diagram (Text)

```
Start Program
  ↓
Display Main Menu
  ↓
User Selects Option
  ↓
Cipher Module → Encrypt / Decrypt
```



3. One-Line Explanation

The program runs as a menu-driven command-line application where the user selects a security function, provides input, and the system processes the request before returning to the main menu.