Report November 2025

Author: Ahunanya Israel

Role: Facilitator of WIL students, Developer

Group: WIL students

Number of students: 4

Date: 30 November 2025

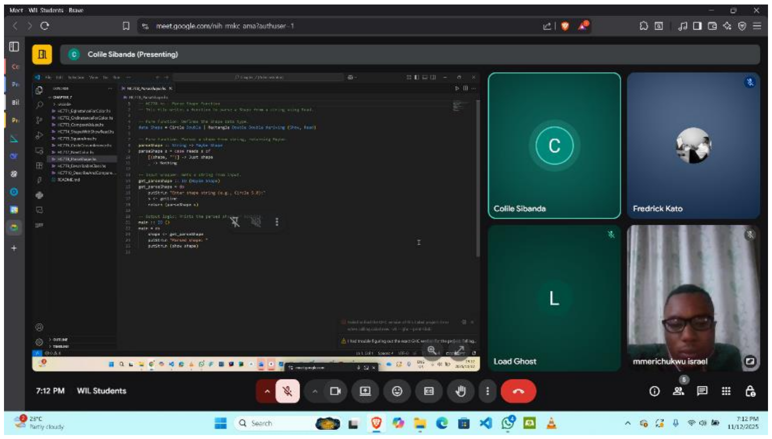
Table of Contents

1. Introduction

This report summarizes the learning activities, progress, and achievements of the WIL (Work Integrated Learning) students for November 2025 under my facilitation at Richfield. The focus this month was on reinforcing Haskell programming skills, particularly around intermediate and advanced topics such as recursion, the Maybe type, and practical application of chapters 7–11 from their study materials. Students also engaged in hands-on software development exercises, including the creation of verification apps and practical problem-solving tasks, fostering their ability to apply functional programming concepts in real-world scenarios. Collaborative practices such as GitHub forking, pull requests, and code review were further emphasized to strengthen students' workflow and teamwork skills. Overall, November was geared towards consolidating foundational knowledge while gradually introducing more complex programming challenges.

2. WIL student Meetings

2.1. Chapter 7, 8 and 9: Practical Questions



```haskell
-- HC77X.hs - Parse Shape Function
-- This file writes a function to parse a Shape from a string using Read.

-- Pure function: Defines the shape data type.
data Shape = Circle Double | Rectangle Double Double deriving (Show, Read)

-- Pure function: Parses a shape from string, returning Maybe.
parseShape :: String -> Maybe Shape
parseShape s = case reads s of
    [(shape, "")] -> Just shape
    _ -> Nothing

-- Input wrapper: Gets a string from input.
get_parseShape :: IO (Maybe Shape)
get_parseShape = do
    putStrLn "Enter shape string (e.g., Circle 5.0):"
    s <- getLine
    return (parseShape s)

-- Output logic: Prints the parsed sh[illegible]
main :: IO ()
main = do
    shape <- get_parseShape
    putStrLn "Parsed shape: "
    putStrLn (show shape)
```

Time: 19:00 – 20:00

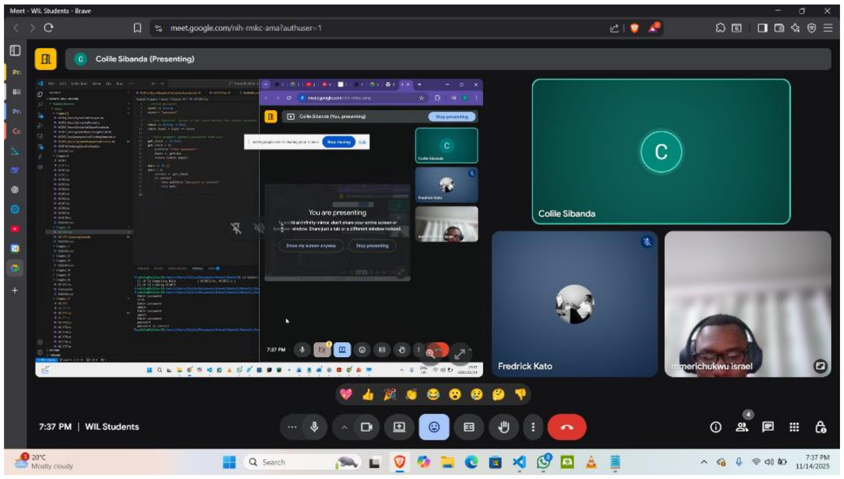
Date: 12-11-2025

Attendance: 3

Expected Attendance: 4

Summary: On this day we went through Chapter 7,8 and 9 and did a practical example of what we had learnt

2.2. Chapter 10 & 11


Meet - WIL Students - Brave
meet.google.com/nih-rmkc-ama?authuser=1
Collie Sibanda (Presenting)
You are presenting
Collie Sibanda
Fredrick Kato
mmerichukwu israel
7:37 PM | WIL Students

Time: 19:00-20:00

Date: 14/11/2025

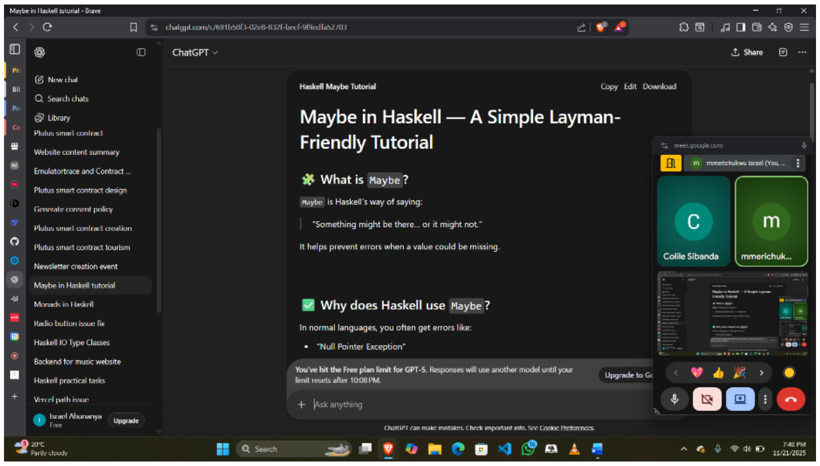
Attendance: 2

Expected Attendance: 4

Summary: Went through chapters 10 and 11 and where given practical questions based on what was learnt.

Achievement of the day: The students where able to create a verification app

2.3. Maybe



Haskell Maybe Tutorial

Maybe in Haskell — A Simple Layman-Friendly Tutorial

What is `Maybe`?

`Maybe` is Haskell's way of saying:

"Something might be there... or it might not."

It helps prevent errors when a value could be missing.

Why does Haskell use `Maybe`?

In normal languages, you often get errors like:

- "Null Pointer Exception"

You've hit the Free plan limit for GPT-5. Responses will use another model until your limit resets after 10:08 PM.

ChatGPT can make mistakes. Check important info. See Cookie Preferences.

Time: 19:00-20:00

Date: 08/21/2025

Attendance: 3

Expected Attendance: 4

Summary: On this day we dived into Maybe type in Haskell. When is it is used and how do we use it

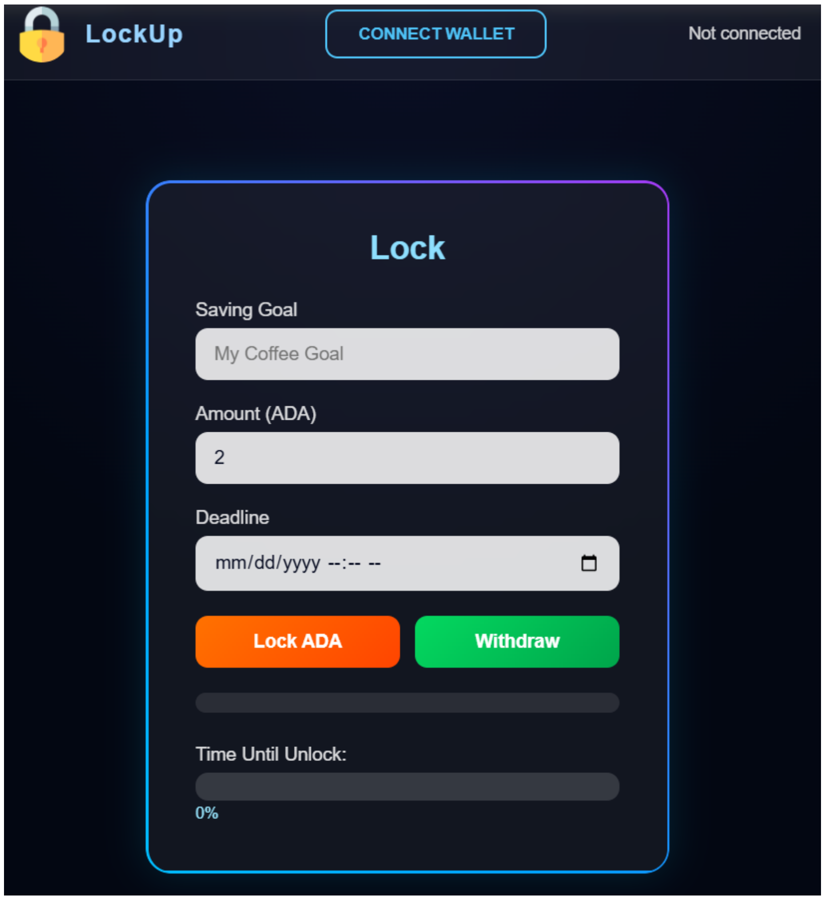
Achievement of the day: Students got a good understanding of what Maybe is in Haskell

3. Performances

Name	Surname	Github	Progress Tokens
Fredrick	Kato	https://github.com/FredrickKcode	21
Colile	Sibanda	https://github.com/Colile1/	18
Siham	Ali	https://github.com/S1ham/	20
Khayelihle	Moyo	https://github.com/loadghost8/	19

4. Decentralized Application

4.1. Covered Call Vault



Saving Goal: My Coffee Goal

Amount (ADA): 2

Deadline: mm/dd/yyyy --:-- --

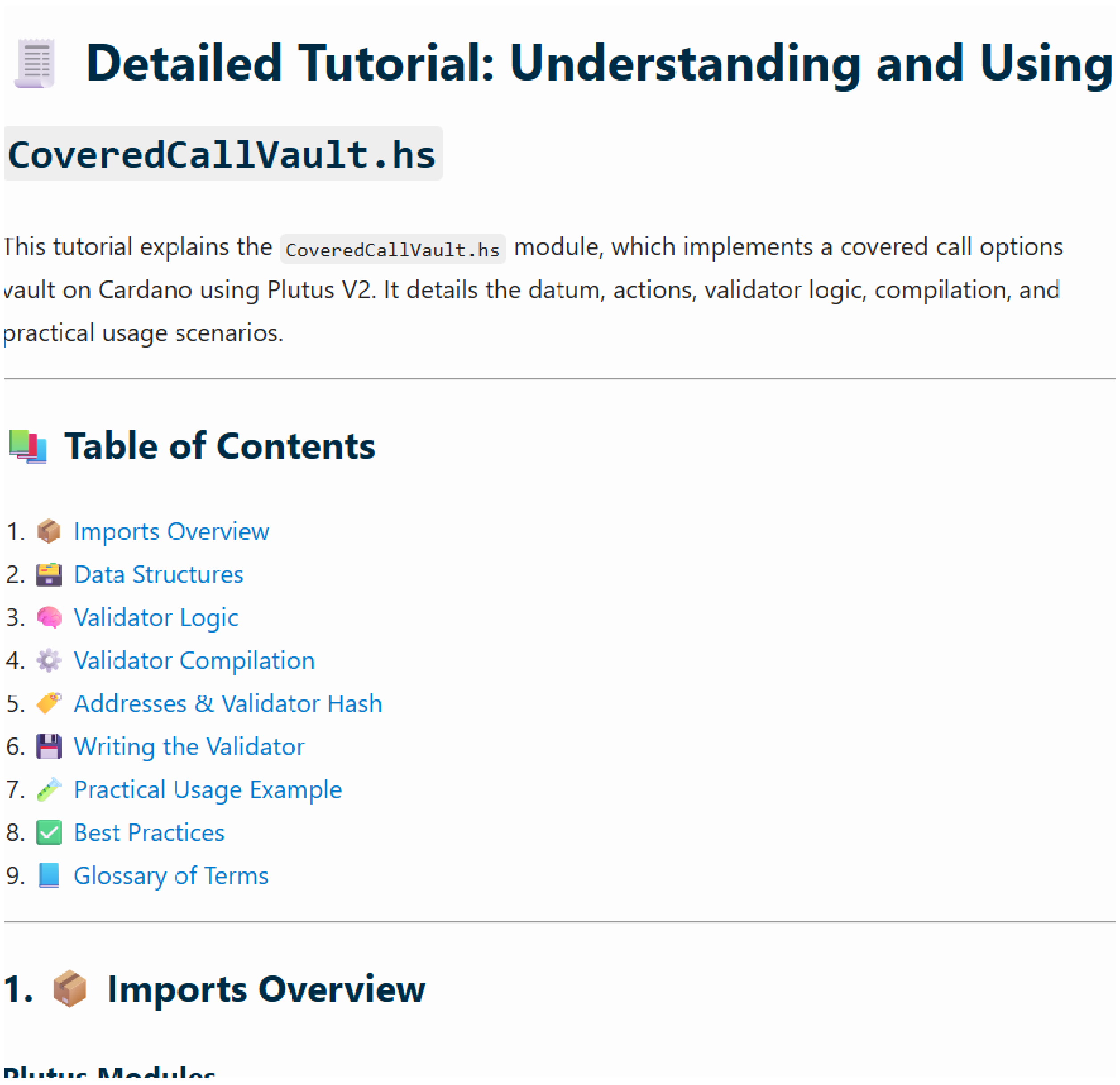
Description: Locks Ada in a smart contract at a set time and can only be unlocked when set date is true

Source Code:

https://github.com/ahunanyaIsrael/lockup

5. Smart Contracts

5.1. Covered Call Vault



🧾 Detailed Tutorial: Understanding and Using `CoveredCallVault.hs`

This tutorial explains the `CoveredCallVault.hs` module, which implements a covered call options vault on Cardano using Plutus V2. It details the datum, actions, validator logic, compilation, and practical usage scenarios.

📚 Table of Contents

---

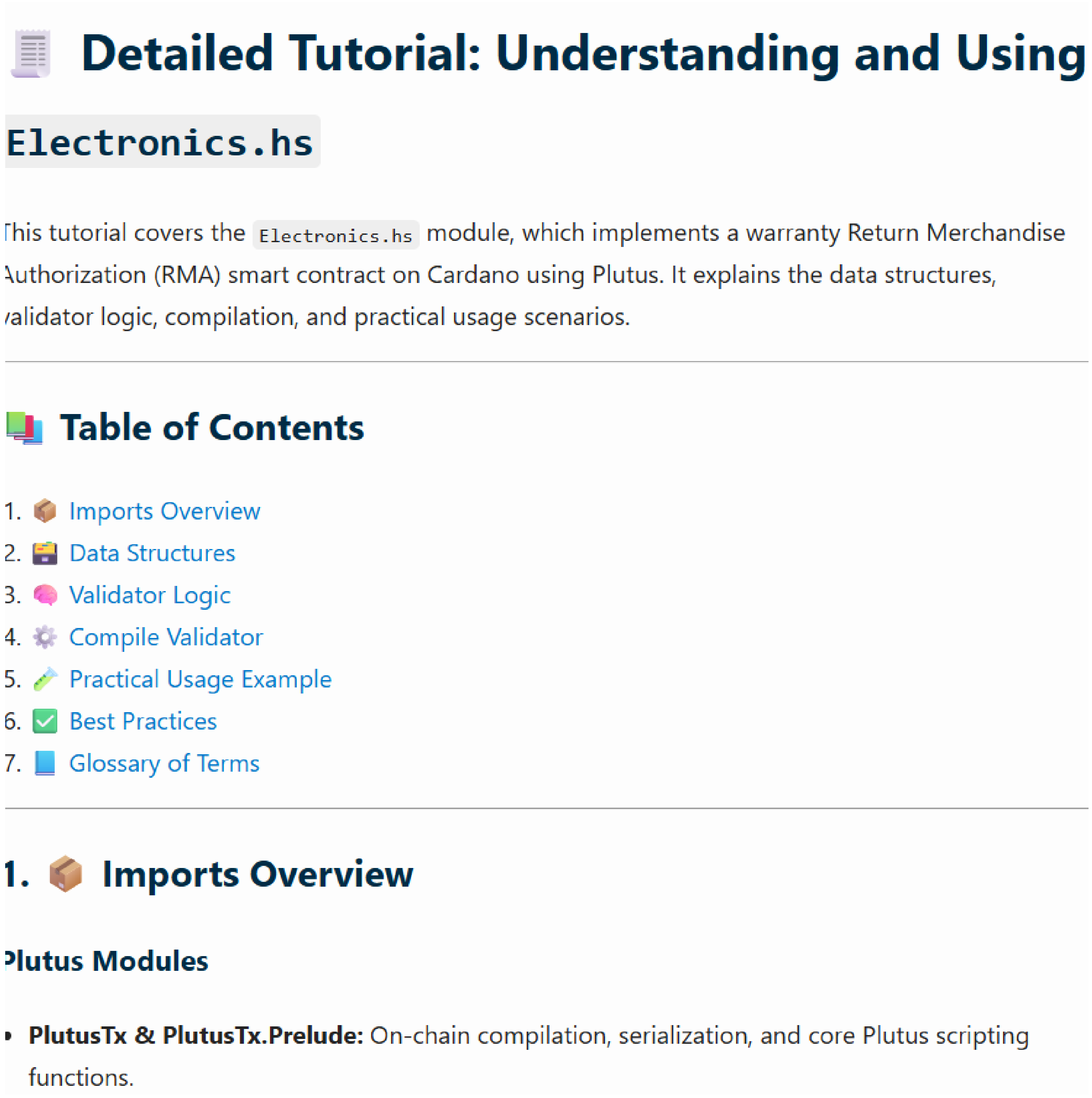
1. 📦 Imports Overview

Plutus Modules

Source Code:

https://github.com/ahunanyaIsrael/plutus-nix/blob/main/code/wspace/tests/CoveredCallVault.hs

5.2. Electronics



🧾 Detailed Tutorial: Understanding and Using `Electronics.hs`

This tutorial covers the `Electronics.hs` module, which implements a warranty Return Merchandise uthorization (RMA) smart contract on Cardano using Plutus. It explains the data structures, alidator logic, compilation, and practical usage scenarios.

📚 Table of Contents

---

1. 📦 Imports Overview

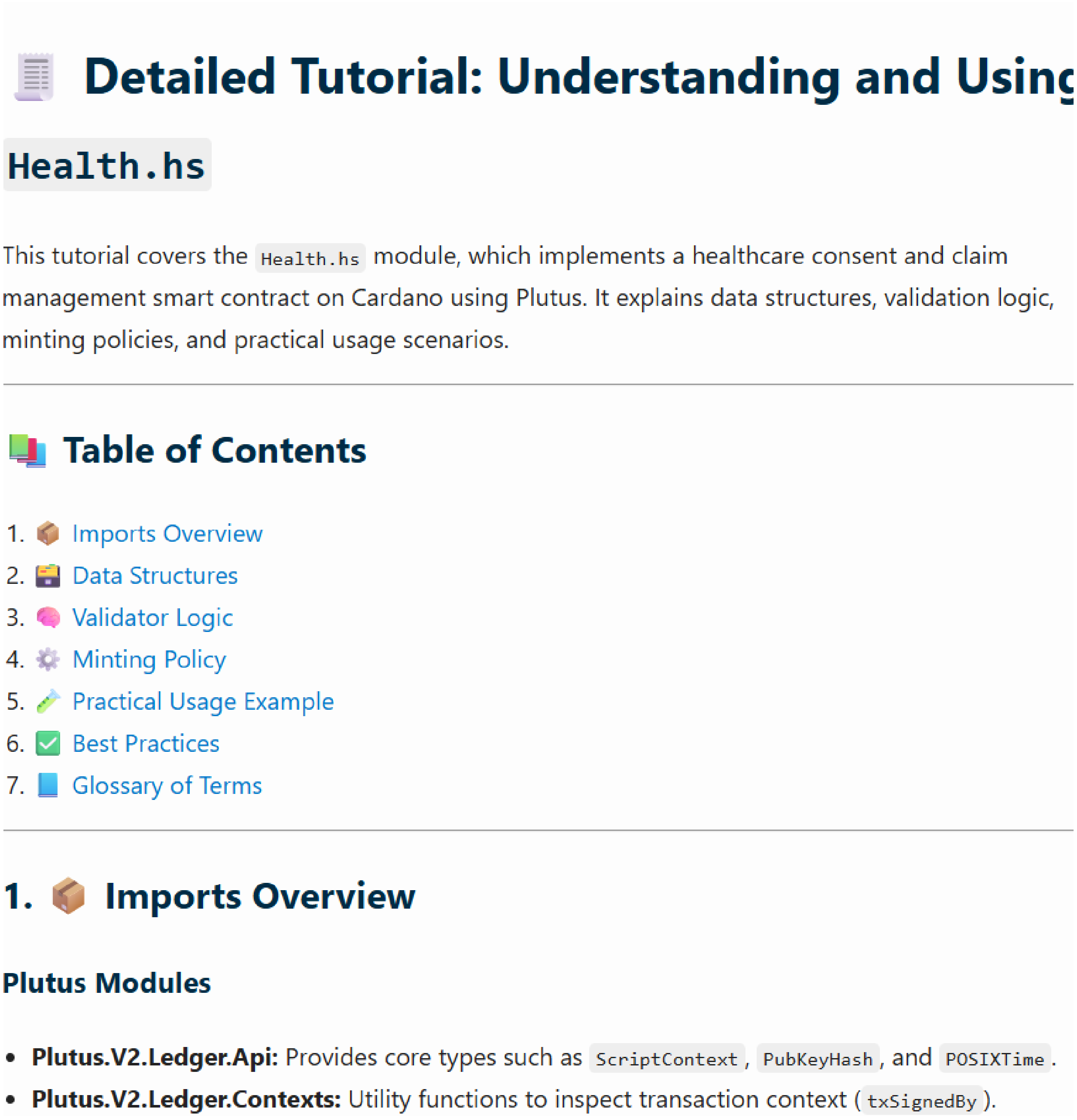
lutus Modules

- **PlutusTx & PlutusTx.Prelude:** On-chain compilation, serialization, and core Plutus scripting functions.

Source Code:

https://github.com/ahunanyaIsrael/plutus-nix/blob/main/code/wspace/tests/Electronics.hs

5.3. Health



🧾 Detailed Tutorial: Understanding and Using

`Health.hs`

This tutorial covers the `Health.hs` module, which implements a healthcare consent and claim management smart contract on Cardano using Plutus. It explains data structures, validation logic, minting policies, and practical usage scenarios.

📚 Table of Contents

---

1. 📦 Imports Overview

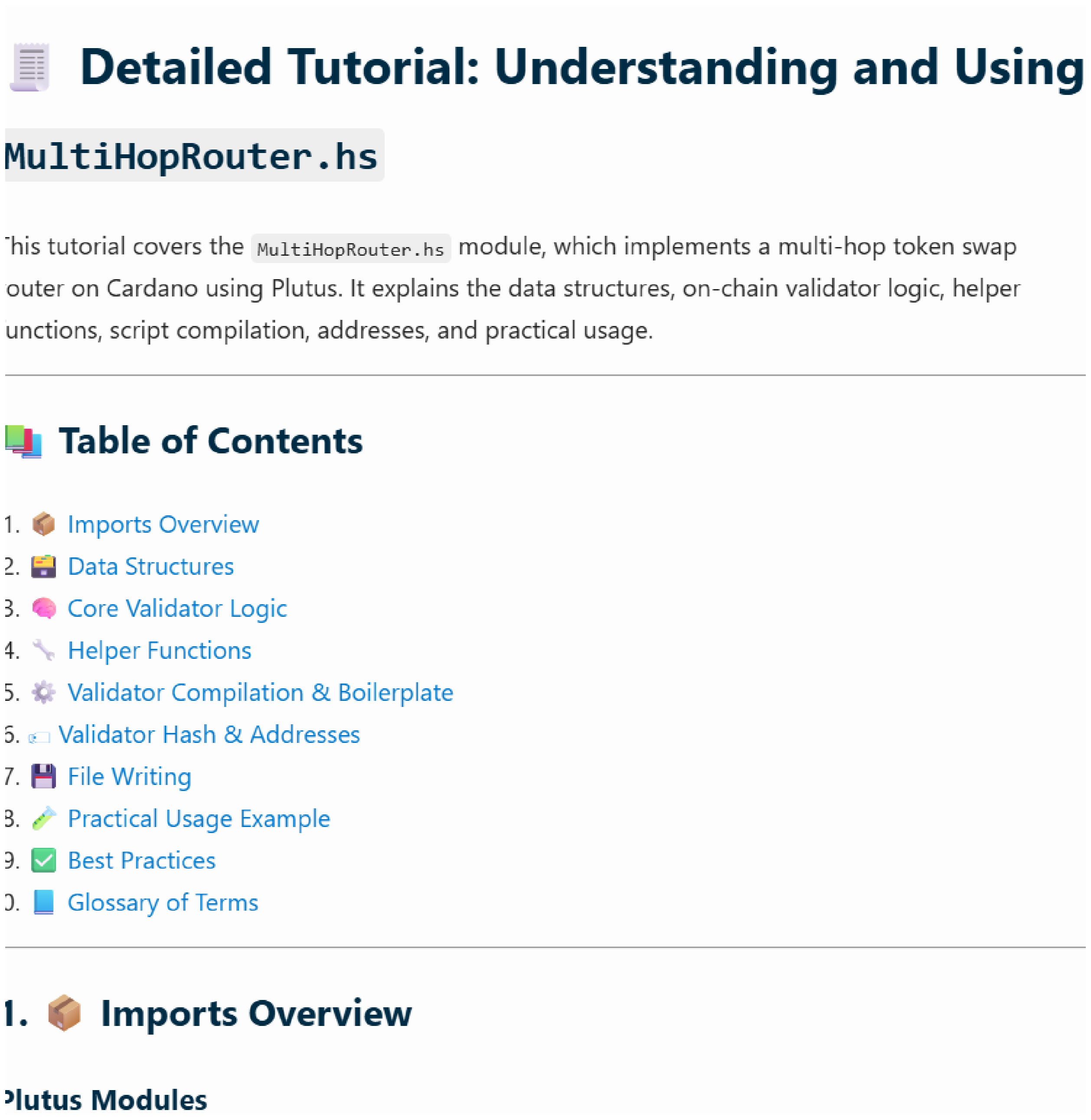
Plutus Modules

- **Plutus.V2.Ledger.Api:** Provides core types such as `ScriptContext`, `PubKeyHash`, and `POSIXTime`.
- **Plutus.V2.Ledger.Contexts:** Utility functions to inspect transaction context (`txSignedBy`).

Source Code:

https://github.com/ahunanyaIsrael/plutus-nix/blob/main/code/wspace/tests/Health.hs

5.4. MultiHopRouter



🧾 Detailed Tutorial: Understanding and Using `MultiHopRouter.hs`

'his tutorial covers the `MultiHopRouter.hs` module, which implements a multi-hop token swap outer on Cardano using Plutus. It explains the data structures, on-chain validator logic, helper unctions, script compilation, addresses, and practical usage.

📚 Table of Contents

---

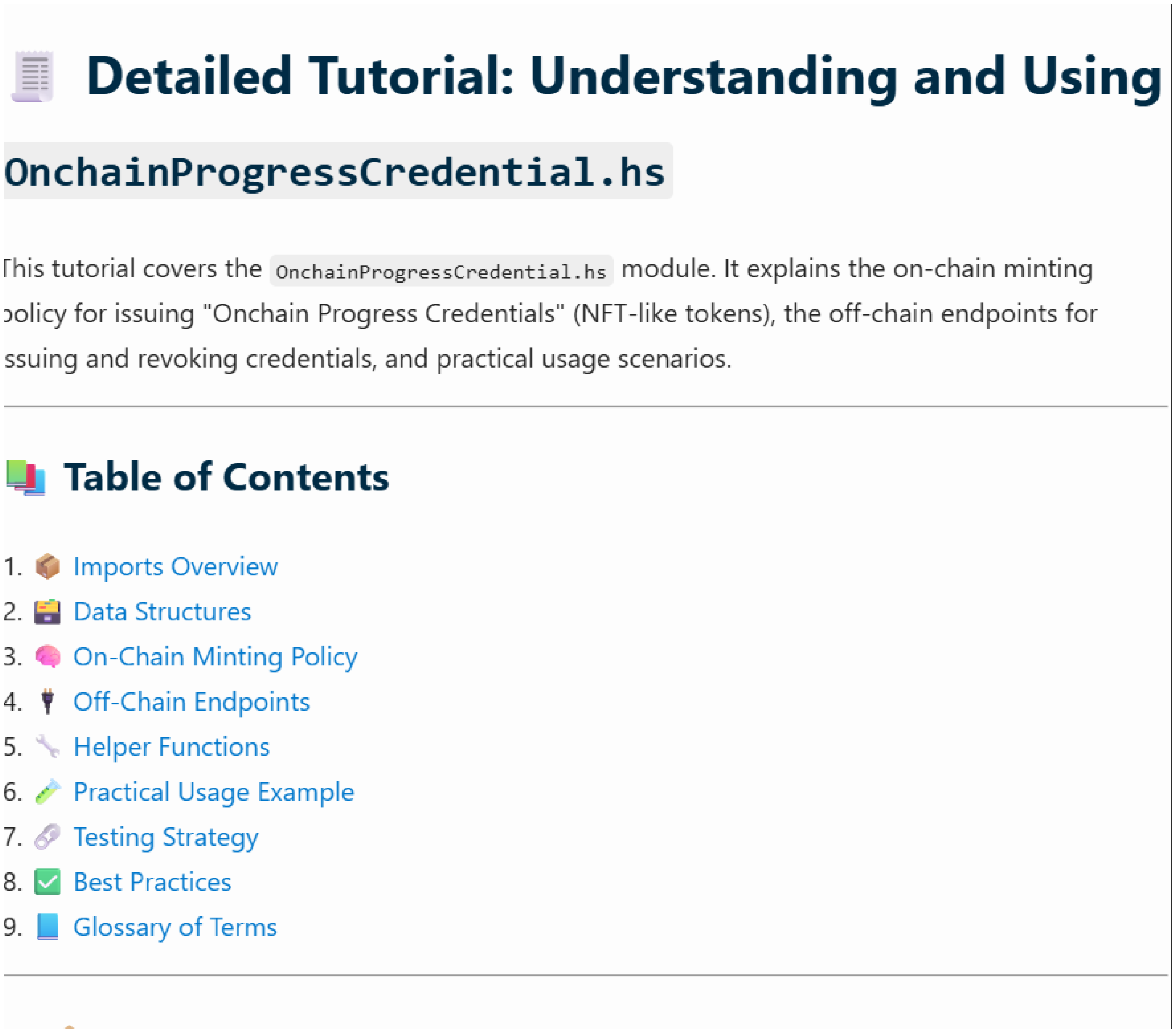
1. 📦 Imports Overview

'lutus Modules

Source Code:

https://github.com/ahunanyaIsrael/plutus-nix/blob/main/code/wspace/tests/MultiHopRouter.hs

5.5. OnchainProgressCredentials



🧾 Detailed Tutorial: Understanding and Using `OnchainProgressCredential.hs`

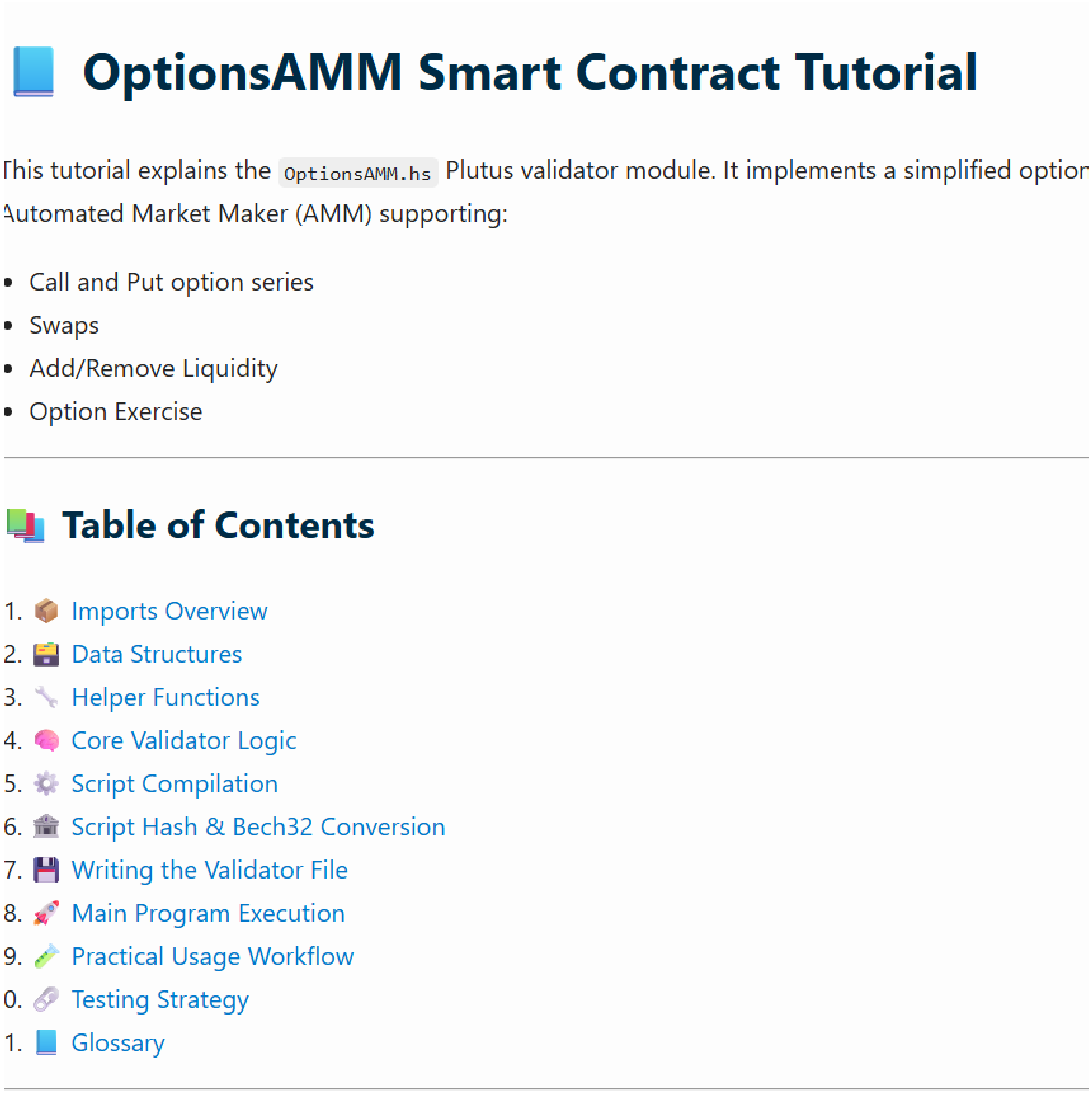
This tutorial covers the `OnchainProgressCredential.hs` module. It explains the on-chain minting olicy for issuing "Onchain Progress Credentials" (NFT-like tokens), the off-chain endpoints for ssuing and revoking credentials, and practical usage scenarios.

📚 Table of Contents

Source Code:

https://github.com/ahunanyaIsrael/plutus-nix/blob/main/code/wspace/tests/OnchainProgress.hs

5.6. Option AMM



📘 OptionsAMM Smart Contract Tutorial

This tutorial explains the `OptionsAMM.hs` Plutus validator module. It implements a simplified optior Automated Market Maker (AMM) supporting:

- Call and Put option series
- Swaps
- Add/Remove Liquidity
- Option Exercise

📚 Table of Contents

Source Code:

https://github.com/ahunanyaIsrael/plutus-nix/blob/main/code/wspace/tests/OptionsAMM.hs

5.7. Safe Swap

🔐 SafeSwap Smart Contract – Detailed Tutorial & Explanation

The `SafeSwap` smart contract is designed to enable secure multi-hop token swaps on Cardano using oute validation, minimum-output protection, and strict deadline enforcement. This tutorial explains very component of the code and how it works on-chain.

📚 Table of Contents

---

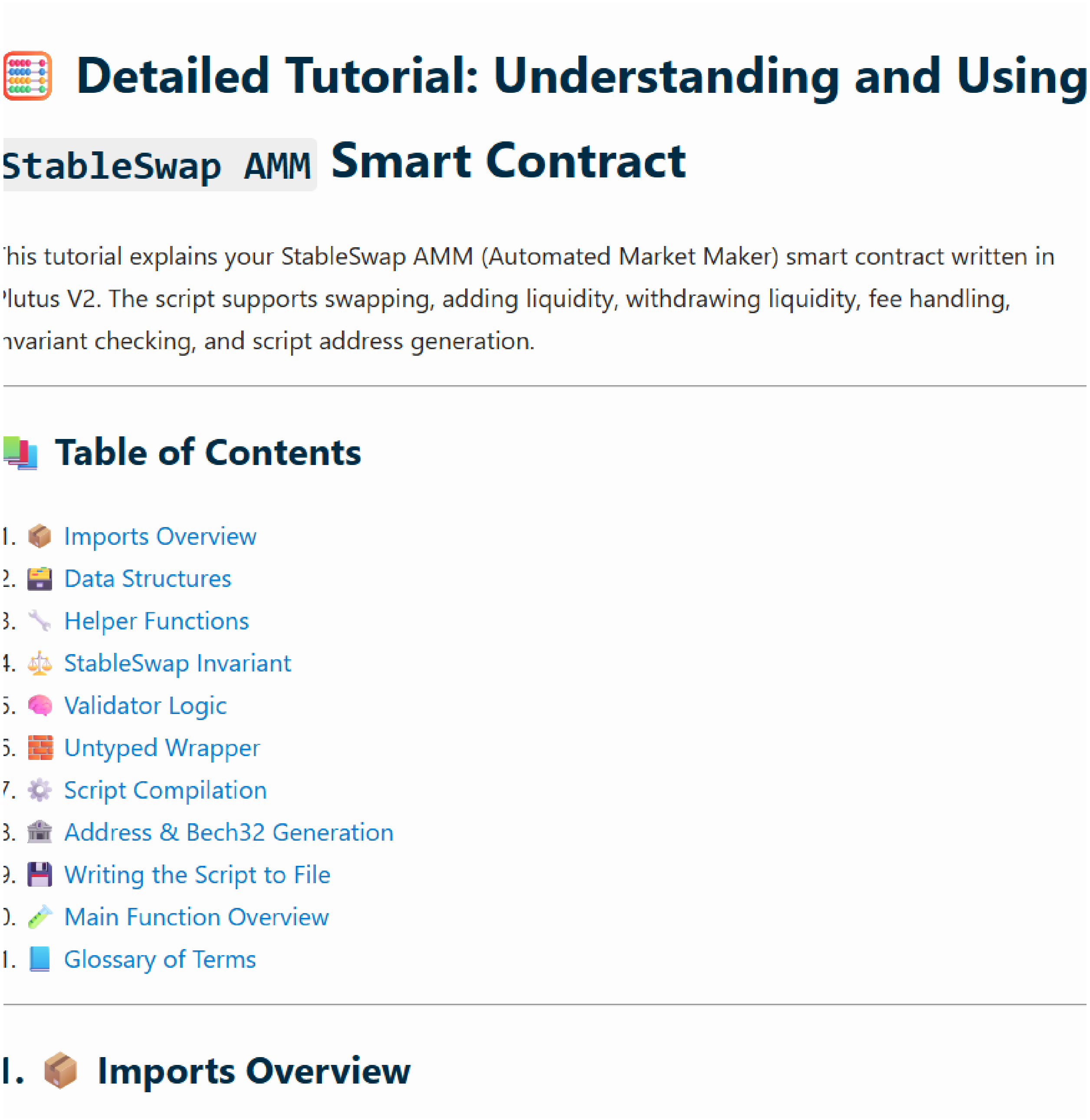
1. 📦 Imports Overview

The SafeSwap contract uses a mix of:

Source Code:

https://github.com/ahunanyaIsrael/plutus-nix/blob/main/code/wspace/tests/SafeSwap.hs

5.8. Stable Swap



🧮 Detailed Tutorial: Understanding and Using `StableSwap AMM` Smart Contract

his tutorial explains your StableSwap AMM (Automated Market Maker) smart contract written in lutus V2. The script supports swapping, adding liquidity, withdrawing liquidity, fee handling, nvariant checking, and script address generation.

📚 Table of Contents

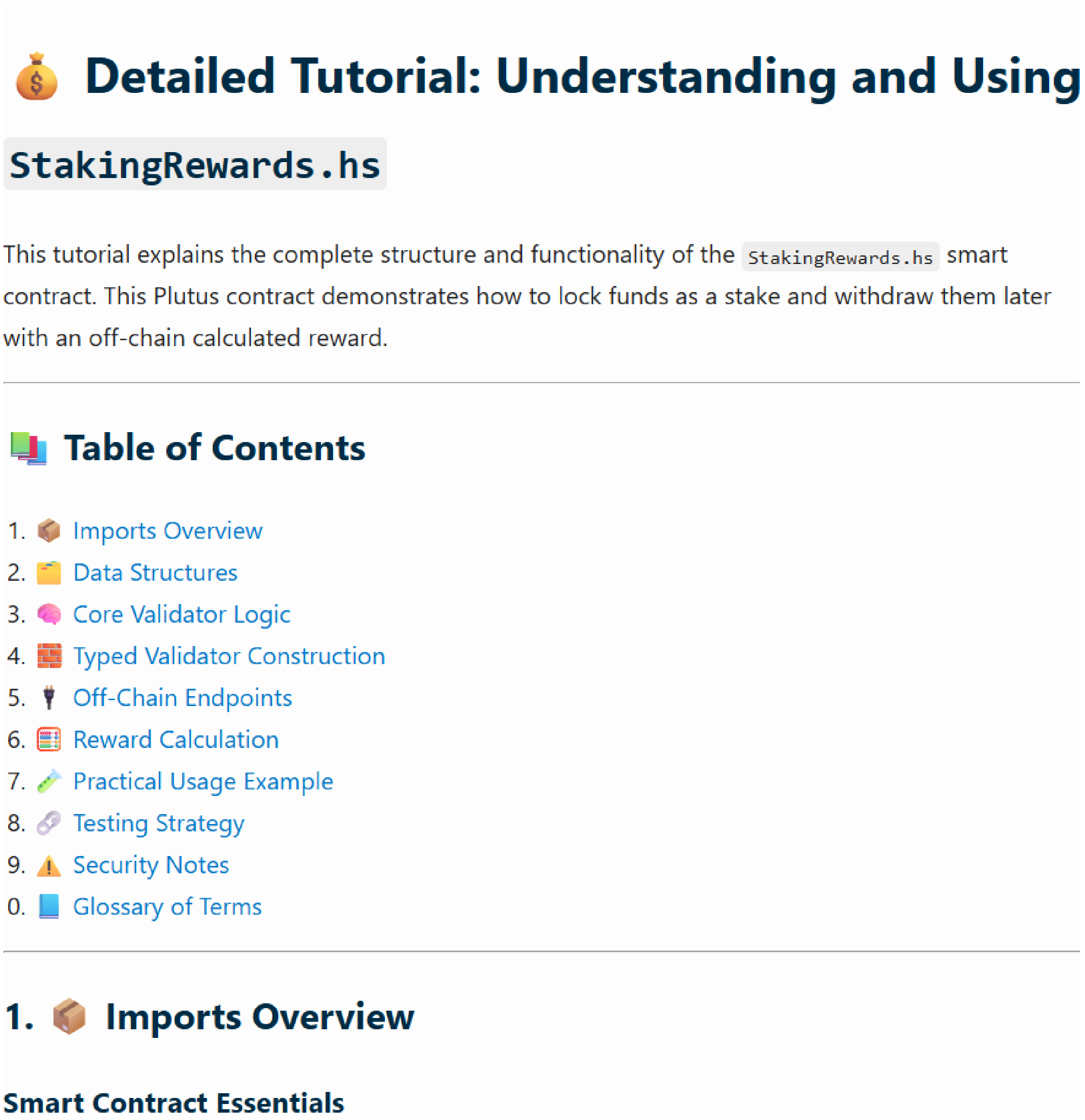
---

1. 📦 Imports Overview

Source Code:

https://github.com/ahunanyaIsrael/plutus-nix/blob/main/code/wspace/tests/StableSwap.hs

5.9. Staking Rewards



💰 Detailed Tutorial: Understanding and Using `StakingRewards.hs`

This tutorial explains the complete structure and functionality of the `StakingRewards.hs` smart contract. This Plutus contract demonstrates how to lock funds as a stake and withdraw them later with an off-chain calculated reward.

📚 Table of Contents

---

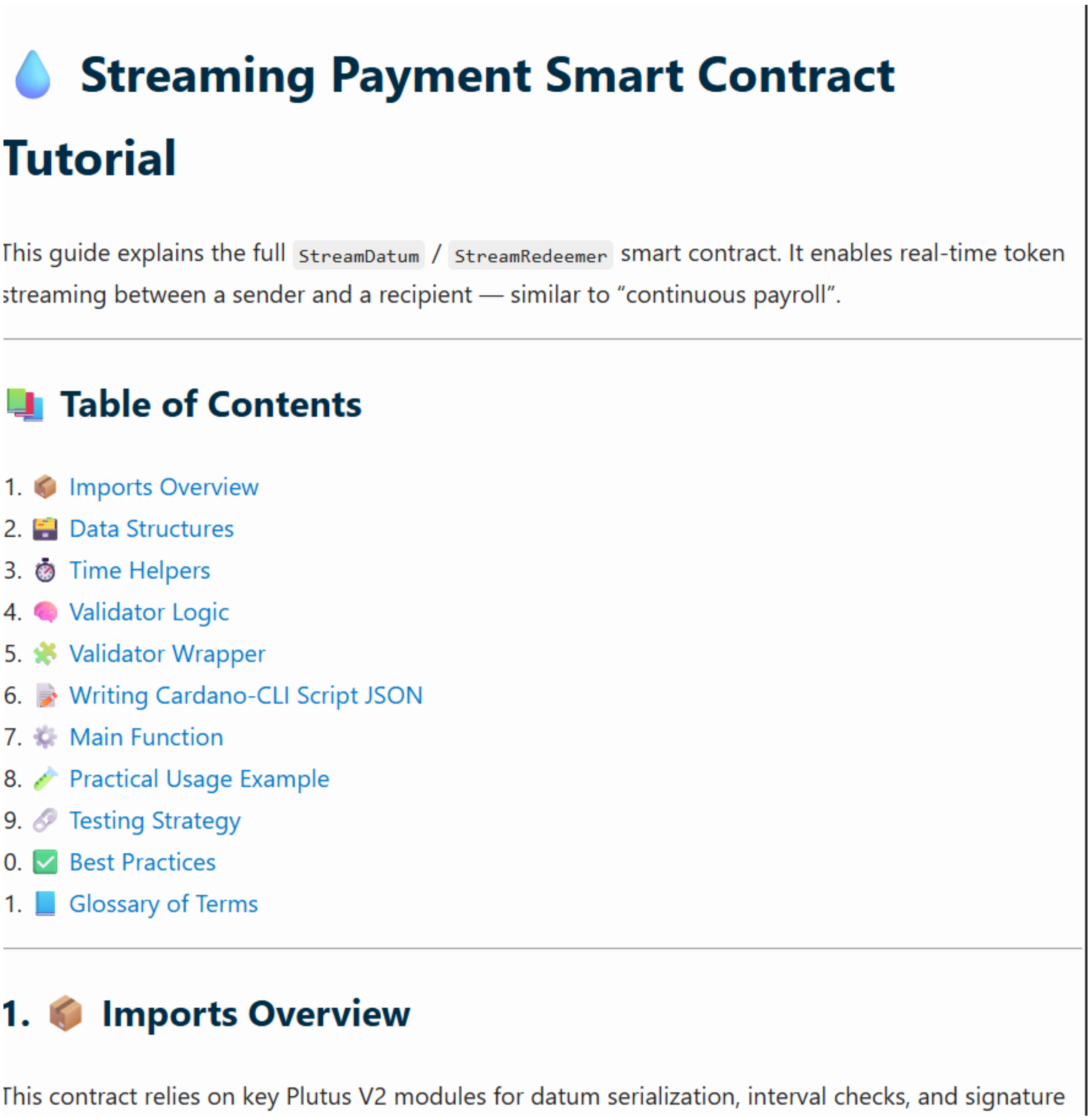
1. 📦 Imports Overview

Smart Contract Essentials

Source Code:

https://github.com/ahunanyaIsrael/plutus-nix/blob/main/code/wspace/tests/StakingRewards.hs

5.10. Stream Vault



💧 Streaming Payment Smart Contract Tutorial

This guide explains the full `StreamDatum` / `StreamRedeemer` smart contract. It enables real-time token streaming between a sender and a recipient — similar to "continuous payroll".

📚 Table of Contents

---

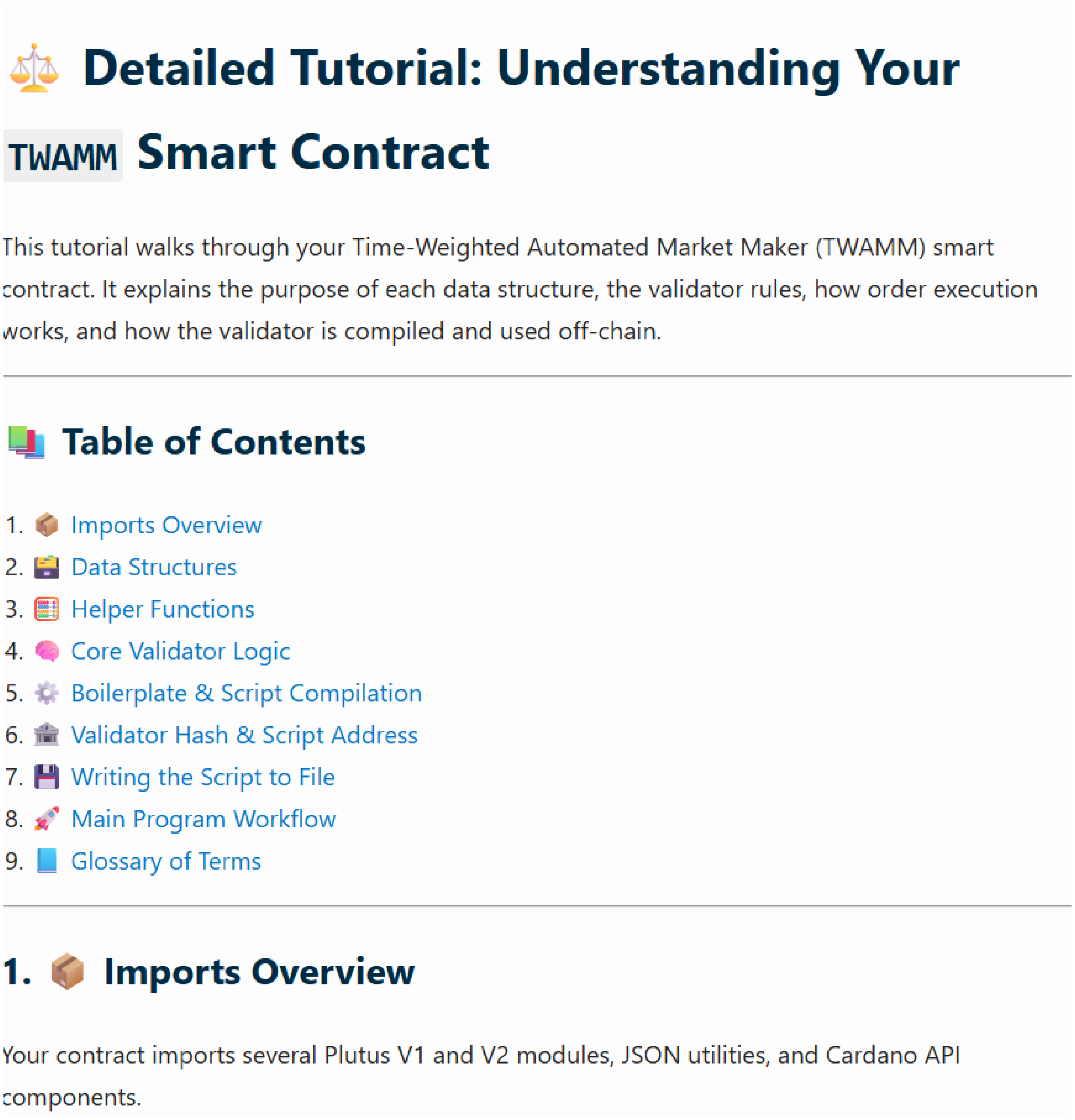
1. 📦 Imports Overview

This contract relies on key Plutus V2 modules for datum serialization, interval checks, and signature

Source Code:

https://github.com/ahunanyaIsrael/plutus-nix/blob/main/code/wspace/tests/StreamVault.hs

5.11. TWAMM



⚖️ Detailed Tutorial: Understanding Your `TWAMM` Smart Contract

This tutorial walks through your Time-Weighted Automated Market Maker (TWAMM) smart contract. It explains the purpose of each data structure, the validator rules, how order execution works, and how the validator is compiled and used off-chain.

📚 Table of Contents

---

1. 📦 Imports Overview

Your contract imports several Plutus V1 and V2 modules, JSON utilities, and Cardano API components.

Source Code:

https://github.com/ahunanyaIsrael/plutus-nix/blob/main/code/wspace/tests/TWAMM.hs

6. Conclusion

In conclusion, November 2025 was a productive month marked by active participation, skill consolidation, and practical learning. Students demonstrated a stronger grasp of Haskell concepts such as the Maybe type, recursion, and practical coding applications from chapters 7–11. Despite occasional attendance fluctuations, the students actively engaged in hands-on exercises and successfully developed functional applications like verification apps. Additionally, their familiarity with collaborative tools and GitHub workflows improved, preparing them for real-world development environments. The consistent combination of theoretical understanding and practical implementation continued to enhance the students' programming capabilities and confidence, setting a strong foundation for future projects in decentralized applications and smart contract development.