**CSE 310 – Applied Programming**

**Module Plan**

|  |  |
| --- | --- |
| **Name:** |  |
| **Date:** | 1/29/2025 |
| **Teacher:** | Bro. Manley |
| **Module # (1-6):** | 2 |

1. Identify which module you have selected to work on. Place an “X” under the “Selected Module” column.

|  |  |
| --- | --- |
| **Modules** | **Selected Module** |
| Cloud Databases |  |
| Data Analysis |  |
| Game Framework |  |
| GIS Mapping |  |
| Mobile App |  |
| Networking |  |
| SQL Relational Databases |  |
| Web Apps |  |
| Language – C++ |  |
| Language – Java |  |
| Language – Kotlin |  |
| Language – R |  |
| Language – Erlang |  |
| Language – JavaScript |  |
| Language – C# |  |
| Language - TypeScript |  |
| Language – Rust | x |
| Choose Your Own Adventure |  |

1. At a high level, describe the software you plan to create that will fulfill the requirements of this module. This may change as you learn more about the technology or language you are learning.

The program is an asymmetric cryptography tool comprised of three parts: keymaker, encrypter, and decrypter. It uses elliptic curve cryptography to cipher and decipher data using the asymmetric keysets that it generates.

1. Create a detailed schedule using the table below to complete your selected module during this Sprint. Include details such as what (task), when (time), where (location), and duration. You are expected to spend 24 hours every Sprint working on this individual module and other activities in the course. Time spent on this individual module should be at least 12 hours.

|  |  |  |
| --- | --- | --- |
|  | **First Week of Sprint** | **Second Week of Sprint** |
| **Monday** | Foundational Research. 1 hour in class. | File reading and writing, encryption and successful decryption. 1 hour in class, 2 hours individual. |
| **Tuesday** | Follow-up work. 2 hours individual. | Follow-up work. 2 hours individual. |
| **Wednesday** | Advanced Research, Version Control Setup, and stubbing out of functions to be implemented. 1 hour in class, 1 hour individual. | Key generation. 1 hour in class, 2 hours individual. |
| **Thursday** | Follow-up work. 2 hours individual. | Follow-up and final polish. 3 hours on team call. |
| **Friday** | Mathematic Implementation, Stubbing out of all functions, I/O Implementation. 1 hour in class, 1 hour individual. | Presentation. 1 hour in class. |
| **Saturday** | Follow-up work. 3 hours individual. |  |

9 15

1. Identify at least two risks that you feel will make it difficult to succeed in this module. Identify an action plan to overcome each of these risks.

The biggest challenge is the mathematical understanding of ECC to implement it in code. Except maybe in software engineering firms, engineers will have ample support from the business side of the company to explain how complex features are intended to work. In our case, we are trying to teach ourselves one of the most complex algorithms in cryptography. We need to study hard and use our resources to understand this complex principle.

Another big risk is related to Rust. Without knowing the language, we have the risk of it not being able to accommodate the software and math of large numbers. If we run into these problems, we will research libraries available that could extend the capabilities of basic Rust.