### **Dev Log Structure:**

1. **Start the Document**:
   1. Title the document: **"Dev Log"**.

### **Section 1: Version Overview**

This section should explain:

* **What will be made in each version** of your project.
* **Link it to functional and non-functional requirements** (what the project should do and how well it should do it).
* **Why each feature is included** in the version.

#### **Example:**

**Version 1:**

* **Functional Requirements**: Basic user login system with email verification.
* **Non-Functional Requirements**: System should be secure with proper password encryption.
* **Reasoning**: User authentication is essential for the application. Email verification is added for security.

### **Section 2: What Has Been Made in Each Version**

* **Screenshots of code** from each version.
* **Changes**: If anything was modified:
  + What was changed?
  + Why was it changed?
  + What was the impact of the change?

#### **Example:**

**Version 1 (Initial Login Page)**:

* **Code Screenshot**: (Include code screenshot here)
* **Change Made**: Added validation to ensure the email is in the correct format.
* **Reason**: Users were able to enter invalid emails, which caused errors.
* **Impact**: The login process is now more user-friendly and secure.

### **Section 3: Evaluate Each Version**

* After documenting each version, **evaluate**:
  + What worked well?
  + What didn’t work well and why?
  + What changes were made in response to issues?

### **Section 4: "If I Did Another Version"**

* At the end of your Dev Log, **discuss** what you would do in the next version:
  + Features to add.
  + Changes you would make.
  + Improvements you would focus on.

#### **Example:**

**If I Did Another Version**:

* **Feature to Add**: Add a user profile page where users can update their information.
* **Improvement**: Improve the security of password storage by implementing two-factor authentication (2FA).

### **Section 5: Code Quality**

* Ensure the **code is robust** and **secure**:
  + Make sure your code is designed to handle unexpected input or errors.
  + **Comment your code** thoroughly, explaining what each part does.

### **Section 6: Testing Table**

* Start a **testing table** early in the project. This table should track the tests you’ve run and their outcomes.
* **Record important tests** and include **screenshots** for evidence.
* In your **PowerPoint presentation**, link each screenshot to a test.

#### **Example of a Testing Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Name** | **Description** | **Test Outcome** | **Screenshot** |
| Login Function Test | Test if the login form accepts valid inputs | Passed | (Insert Screenshot) |
| Email Validation Test | Test if the email input only accepts valid emails | Failed (Invalid email accepted) | (Insert Screenshot) |
| Password Encryption Test | Test if passwords are stored securely | Passed | (Insert Screenshot) |
| User Authentication Test | Test if the user can successfully log in | Passed | (Insert Screenshot) |

### **Final Notes:**

* Focus on **important tests**, especially those that were problematic and needed fixing.
* Keep **screenshots in your PowerPoint** and link each screenshot to the corresponding test.
* The Dev Log should be a clear record of your development process, showing both what you did and how you tested it.