

# Practice with GitHub and Software Develop

## Description of the Project

The project consists of developing a basic computer security web tool that allows users to perform a simple analysis of passwords to determine their strength. The tool will include a web interface for entering passwords, a password evaluation system, and a feedback function to improve password security.

## Roles

- **Lead/Reviewer:** Responsible for reviewing and approving changes to the code.
- **Programmers:** N programmers assigned to different features of the project.

## Instructions

### 1. Creation of the Project

#### 1. Initialize the repository:

```
git init security-tool  
cd security-tool  
echo "# Security Tool" >> README.md
```

### 2. Initial content

#### 1. Add the initial structure of the project:

```
mkdir -p src/public src/css src/js  
touch src/index.html src/css/styles.css src/js/main.js
```

## 2. Add basic content to `index.html`

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Security Tool</title>
  <link rel="stylesheet" href="css/styles.css">
</head>
<body>
  <h1>Password Strength Checker</h1>
  <input type="password" id="password" placeholder="Enter your password">
  <button id="check">Check Password</button>
  <p id="feedback"></p>
  <script src="js/main.js"></script>
</body>
</html>
```

## 3. Add basic content to `styles.css`

```
body {
  font-family: Arial, sans-serif;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  height: 100vh;
  margin: 0;
}
#password {
  padding: 10px;
  font-size: 16px;
```

```
margin-bottom: 10px;
}
```

#### 4. Add basic content to `main.js`:

```
document.getElementById('check').addEventListener('click', function() {
    const password = document.getElementById('password').value;
    document.getElementById('feedback').textContent = `Password length: ${password.length}`;
});
```

#### 5. Make the first commit, create remote repository in GitHub and add it to your git project, finally upload your files

```
git add .
git commit -m "Initial project structure"
git branch -M main
git remote add origin git@github.com:{user_name}/{repository_name}.git
git push -u origin main
```

### 3. Development in the branch develop

#### 1. Create the develop branch from main:

```
git checkout -b develop
git push origin HEAD
```

#### 4. Add rules protection to main branch

### Protect matching branches

☒ **Require a pull request before merging**  
 When enabled, all commits must be made to a non-protected branch and submitted via a pull request before they can be merged into a branch that matches this rule.

☐ **Require approvals**  
 When enabled, pull requests targeting a matching branch require a number of approvals and no changes requested before they can be merged.

☐ **Dismiss stale pull request approvals when new commits are pushed**  
 New reviewable commits pushed to a matching branch will dismiss pull request review approvals.

☒ **Require review from Code Owners**  
 Require an approved review in pull requests including files with a designated code owner.

☐ **Require approval of the most recent reviewable push**  
 Whether the most recent reviewable push must be approved by someone other than the person who pushed it.

☐ **Require status checks to pass before merging**  
 Choose which [status checks](#) must pass before branches can be merged into a branch that matches this rule. When enabled, commits must first be pushed to another branch, then merged or pushed directly to a branch that matches this rule after status checks have passed.

☐ **Require conversation resolution before merging**  
 When enabled, all conversations on code must be resolved before a pull request can be merged into a branch that matches this rule. [Learn more about requiring conversation completion before merging.](#)

☐ **Require signed commits**  
 Commits pushed to matching branches must have verified signatures.

☐ **Require linear history**  
 Prevent merge commits from being pushed to matching branches.

☐ **Require deployments to succeed before merging**  
 Choose which environments must be successfully deployed to before branches can be merged into a branch that matches this rule.

☒ **Lock branch**  
 Branch is read-only. Users cannot push to the branch.

## 5. Assignment of Features

- **Programmer 1: Password validation**
  - Create branch: `feature/password-validation`
  - Add logic to validate passwords according to simple criteria like length, uppercase, lowercase, numbers, special characters.

- **Programmer 2: Security feedback**

- Create branch: `feature/security-feedback`
- Add detailed messages to help users improve their passwords, for example, research algorithms to identify password strength and show the results with friendly message to user.

- **Programmer 3: Improve the interface**

- Create branch: `feature/ui-enhancements`
- Improve the design and usability of the web interface.

- **Programmer 4: Integration with external library**

- Create branch: `feature/external-library`
- Integrate an external library to analyze passwords.

## 6. Development and Conflict Resolution

This flow is repeated for each of the functionalities that each developer does.

### 1. Development of each feature:

```
git checkout -b feature/password-validation
# First write all the code necessary to fulfill the functionality, then you can execute the following commands

git add .
git commit -m "Add password validation logic"
git push origin feature/password-validation
```

### 2. Pull Request and Review:

- Create a pull request from 'feature/password-validation' to 'develop'.
- The lead/reviewer reviews and approves the pull request.
- **After leader review:** This is a flow to resolve conflicts if there are any, perform rebasing if necessary:

```
git checkout develop
git pull origin develop
git checkout feature/password-validation
git rebase develop
git push -f origin feature/password-validation
```

**3. Integration in develop (Optional, The PR from feature to develop on GitHub does the same):**

```
git checkout develop
git merge feature/password-validation
git push origin develop
```

## **7. Merge develop into main**

**1. First perform thorough tests on develop:**

- Make sure that all the features are working properly and that there are no conflicts.

**2. Merge develop into main:**

```
git checkout main
git merge develop
git push origin main
```