Playwright Advanced Env + Encrypted Credentials Setup

```
==========================================
1) Recommended Project Folder Structure
==========================================

project-root/
  |- playwright.config.ts
  |- package.json
  |- .env.qa
  |- .env.dev
  |- .env.prod
  |- .env.secrets        (encryption key etc., not committed)
  |
  |- src/
      |- config/
      |    |- envLoader.ts
      |
      |- enums/
      |    |- UserType.ts
      |
      |- utils/
      |    |- CryptoUtil.ts
      |    |- UserCredentials.ts
      |
      |- pages/
      |    |- LoginPage.ts
      |
      |- tests/
           |- login.spec.ts


==========================================
2) Multi-Environment .env Files
==========================================

Example: .env.qa

BASE_URL=https://qa.your-app.com
USER_FA01_NAME=FA01-neo3694
USER_FA01_PASSWORD_ENC=<encrypted-text-here>
USER_SE12558_NAME=se12558
USER_SE12558_PASSWORD_ENC=<encrypted-text-here>

Example: .env.dev

BASE_URL=https://dev.your-app.com
USER_FA01_NAME=dev-fa01-user
USER_FA01_PASSWORD_ENC=<encrypted-text-here>


==========================================
3) Secrets File for Encryption Key
==========================================

.env.secrets (never commit to git)

CRYPTO_KEY=0123456789abcdef0123456789abcdef
CRYPTO_IV=abcdef9876543210abcdef9876543210

Key is 32 bytes hex for AES-256.
IV is 16 bytes hex.


==========================================
4) Enum for User Types
==========================================

File: src/enums/UserType.ts

export enum UserType {
  FA01 = "FA01",
  SE12558 = "SE12558",
  TEST0987 = "TEST0987"
}
```

==
5) Environment Loader (reads --env argument)
==

File: src/config/envLoader.ts

```
import * as dotenv from "dotenv";
import * as path from "path";

// Get --env=QA or default to QA
function resolveEnv(): string {
  const arg = process.argv.find(a => a.startsWith("--env="));
  if (arg) {
    return arg.split("=")[1];
  }
  return process.env.ENV || "QA";
}

const activeEnv = resolveEnv();
process.env.ENV = activeEnv; // make available everywhere

// Load environment specific file like .env.qa
const envFile = path.resolve(process.cwd(), `.env.${activeEnv.toLowerCase()}`);
dotenv.config({ path: envFile });

// Load secrets (key, iv etc.)
const secretsFile = path.resolve(process.cwd(), ".env.secrets");
dotenv.config({ path: secretsFile });

export const CURRENT_ENV = activeEnv;
```

==
6) Crypto Utility for Encryption / Decryption
==

File: src/utils/CryptoUtil.ts

```
import crypto from "crypto";

const algorithm = "aes-256-cbc";

function getKeyIv() {
  const keyHex = process.env.CRYPTO_KEY;
  const ivHex = process.env.CRYPTO_IV;

  if (!keyHex || !ivHex) {
    throw new Error("CRYPTO_KEY or CRYPTO_IV not set in .env.secrets");
  }

  return {
    key: Buffer.from(keyHex, "hex"),
    iv: Buffer.from(ivHex, "hex")
  };
}

// Encrypt (use in a small node script, not in tests)
export function encrypt(plain: string): string {
  const { key, iv } = getKeyIv();
  const cipher = crypto.createCipheriv(algorithm, key, iv);
  let encrypted = cipher.update(plain, "utf8", "base64");
  encrypted += cipher.final("base64");
  return encrypted;
}

// Decrypt (used at runtime in tests)
export function decrypt(enc: string): string {
  const { key, iv } = getKeyIv();
  const decipher = crypto.createDecipheriv(algorithm, key, iv);
  let decrypted = decipher.update(enc, "base64", "utf8");
  decrypted += decipher.final("utf8");
  return decrypted;
}


==========================================
7) UserCredentials Using Env + Decryption
==========================================

File: src/utils/UserCredentials.ts

import { UserType } from "../enums/UserType";
import { decrypt } from "./CryptoUtil";
import "../config/envLoader"; // ensure env files are loaded

export class UserCredentials {

  static getCredentials(user: UserType) {
    const usernameKey = `USER_${user}_NAME`;
    const passwordKey = `USER_${user}_PASSWORD_ENC`;

    const username = process.env[usernameKey];
    const passwordEnc = process.env[passwordKey];

    if (!username || !passwordEnc) {
      throw new Error(`Credentials not found for user: ${user}`);
    }

    const password = decrypt(passwordEnc);
    return { username, password };
  }
}


==========================================
8) Login Page Using Enum-based Credentials
==========================================

File: src/pages/LoginPage.ts

import { Page } from "@playwright/test";
import { UserType } from "../enums/UserType";
import { UserCredentials } from "../utils/UserCredentials";

export class LoginPage {
  constructor(private page: Page) {}

  async goto() {
    const baseUrl = process.env.BASE_URL;
    if (!baseUrl) {
      throw new Error("BASE_URL is not defined in env file");
    }
    await this.page.goto(baseUrl);
  }

  async loginAs(user: UserType) {
    const { username, password } = UserCredentials.getCredentials(user);
    await this.page.fill("#username", username);
    await this.page.fill("#password", password);
    await this.page.click("#loginButton");
  }
}


==========================================
9) Playwright Config: Ensure Env Loader Runs
==========================================

File: playwright.config.ts

import { PlaywrightTestConfig } from "@playwright/test";
import "./src/config/envLoader"; // loads env based on --env

const config: PlaywrightTestConfig = {
  use: {
    baseURL: process.env.BASE_URL,
    headless: true
  },
  // you can still define projects here if needed
```

```
};

export default config;
```

10) Test File Example

File: src/tests/login.spec.ts

```
import { test } from "@playwright/test";
import { LoginPage } from "../pages/LoginPage";
import { UserType } from "../enums/UserType";

test("Login with FA01 user", async ({ page }) => {
  const loginPage = new LoginPage(page);
  await loginPage.goto();
  await loginPage.loginAs(UserType.FA01);
});
```

11) How to Run with Different Environments

QA environment:

```
npx playwright test --env=QA
```

DEV environment:

```
npx playwright test --env=DEV
```

PROD environment (if needed):

```
npx playwright test --env=PROD
```

12) Notes and Best Practices

- Do not commit .env.qa, .env.dev, .env.prod with real passwords.
- Never commit .env.secrets (add it to .gitignore).
- Use encrypt() in a separate small node script to generate encrypted password values.
- Only decrypted values are used at runtime via decrypt().
- Tests just work with UserType enum, no direct username / password in test code.