

Login Base - Full Developer Guide

■ 1) Environment Setup (.env)

```
# ADMIN
ADMIN_USERNAME=admin_user
ADMIN_PASSWORD=admin_pass

# APPROVER
APPROVER_USERNAME=approver_user
APPROVER_PASSWORD=approver_pass

# END USER
USER_USERNAME=end_user_user
USER_PASSWORD=user_pass

# URL Config
QA_SSO_LOGIN_URL=https://yourdomain/login
QA_BASE_URL=https://yourdomain/home
```

■■ 2) Config File (appConfig.ts)

```
import dotenv from "dotenv";
dotenv.config();

export default {
  loginUrl: process.env.QA_SSO_LOGIN_URL || "",
  baseUrl: process.env.QA_BASE_URL || "",

  // ===== Role based credentials =====
  admin: {
    username: process.env.ADMIN_USERNAME || "",
    password: process.env.ADMIN_PASSWORD || "",
  },
  approver: {
    username: process.env.APPROVER_USERNAME || "",
    password: process.env.APPROVER_PASSWORD || "",
  },
  user: {
    username: process.env.USER_USERNAME || "",
    password: process.env.USER_PASSWORD || "",
  }
};
```

■ 3) BasePage (Reusable Actions)

```
import { Page, Locator, expect } from "@playwright/test";

// BasePage = parent class for reusable UI actions
export class BasePage {
    constructor(protected page: Page) {}

    // Navigate to any given URL and wait until network is idle
    async goto(url: string) {
        await this.page.goto(url, { waitUntil: "networkidle" });
    }

    // Safe click - waits until element becomes visible first
    async click(locator: Locator) {
        await expect(locator).toBeVisible();
        await locator.click();
    }

    // Safe type - waits until visible and fills value
    async type(locator: Locator, text: string) {
        await expect(locator).toBeVisible();
        await locator.fill(text);
    }
}
```

■ 4) BaseLogin (Role-Based Login)

```
import { Page, Locator } from "@playwright/test";
import { BasePage } from "./BasePage";
import appConfig from "../config/appConfig";
import { UserRole } from "../enums/UserRole";

// BaseLogin = Now supports role-based login using ENV credentials
export class BaseLogin extends BasePage {

    protected usernameField: Locator;
    protected passwordField: Locator;
    protected loginBtn: Locator;

    constructor(page: Page) {
        super(page);

        this.usernameField = page.locator("#inputUsername");
        this.passwordField = page.locator("#inputPassword");
        this.loginBtn = page.getByRole("button", { name: "Login" });
    }

    // Navigate to login page URL from .env
    async openLoginPage() {
        await this.goto(appConfig.loginUrl);
    }

    // Generic login method with provided credentials
    async login(username: string, password: string) {
        await this.type(this.usernameField, username);
        await this.type(this.passwordField, password);
        await this.click(this.loginBtn);
    }

    // ■ Login using role-based env credentials
    async loginAs(role: UserRole) {
        const creds = appConfig[role];
        await this.openLoginPage();
        await this.login(creds.username, creds.password);
    }
}
```

■ 5) UserRole Enum

```
export enum UserRole {  
    ADMIN = "admin",  
    APPROVER = "approver",  
    USER = "user",  
}
```

■ 6) Test Implementation with Role Login

```
import { test } from "@playwright/test";
import { BaseLogin } from "../pageObjects/BaseLogin";
import { UserRole } from "../enums/UserRole";

test("Login as Admin", async ({ page }) => {
  const login = new BaseLogin(page);
  await login.loginAs(UserRole.ADMIN);
});

test("Login as Approver", async ({ page }) => {
  const login = new BaseLogin(page);
  await login.loginAs(UserRole.APPROVER);
});

test("Login as End User", async ({ page }) => {
  const login = new BaseLogin(page);
  await login.loginAs(UserRole.USER);
});
```