Project Structure

text

```
src/test/java/
 ├── pages/
|    ├── HomePage.java
|    ├── DreamsDiaryPage.java
|    └── DreamsTotalPage.java
 ├── tests/
|    └── DreamPortalTests.java
 ├── utils/
|    └── TestBase.java
└── resources/
     └── config.properties
```

1. TestBase.java - Core Setup

java

```
package utils;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeClass;
import java.time.Duration;

public class TestBase {
    protected WebDriver driver;
    protected WebDriverWait wait;

    @BeforeClass
    public void setUp() {
        System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");
        driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));
        wait = new WebDriverWait(driver, Duration.ofSeconds(15));
    }

    @AfterClass
```

```
    public void tearDown() {
        if (driver != null) {
            driver.quit();
        }
    }
}
```

2. HomePage.java - Page Object

java

```
package pages;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import java.time.Duration;
import java.util.ArrayList;
import java.util.List;

public class HomePage {
    private WebDriver driver;
    private WebDriverWait wait;

    // Locators
    private By loadingAnimation = By.id("loading");
    private By mainContent = By.id("main-content");
    private By myDreamsButton = By.id("myDreamsBtn");

    public HomePage(WebDriver driver) {
        this.driver = driver;
        this.wait = new WebDriverWait(driver, Duration.ofSeconds(15));
```

```
    }

    public void navigateToHomePage() {
        driver.get("https://arjitnigam.github.io/myDreams/");
    }

    public void verifyLoadingAnimation() {
        // Wait for animation to appear
        WebElement animation =
wait.until(ExpectedConditions.visibilityOfElementLocated(loadingAnimation));

        // Wait for animation to disappear (around 3 seconds)

wait.until(ExpectedConditions.invisibilityOfElementLocated(loadingAnimation))
;
    }

    public void verifyMainContentVisibility() {
        wait.until(ExpectedConditions.visibilityOfElementLocated(mainContent));
        wait.until(ExpectedConditions.elementToBeClickable(myDreamsButton));
    }

    public List<String> clickMyDreamsAndGetTabs() {
        String originalWindow = driver.getWindowHandle();

        // Click the My Dreams button
        driver.findElement(myDreamsButton).click();
```

```
        // Wait for new tabs to open
        wait.until(ExpectedConditions.numberOfWindowsToBe(3));

        // Get all window handles
        List<String> windowHandles = new
ArrayList<>(driver.getWindowHandles());
        List<String> urls = new ArrayList<>();

        // Switch to each new tab and get URL
        for (String handle : windowHandles) {
            if (!handle.equals(originalWindow)) {
                driver.switchTo().window(handle);
                urls.add(driver.getCurrentUrl());
                driver.close(); // Close the tab after getting URL
            }
        }

        // Switch back to original window
        driver.switchTo().window(originalWindow);
        return urls;
    }
}
```

3. DreamsDiaryPage.java – Page Object

java

```
package pages;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.ui.WebDriverWait;
import java.time.Duration;
import java.util.List;

public class DreamsDiaryPage {
    private WebDriver driver;
    private WebDriverWait wait;

    // Locators
    private By dreamTableRows = By.cssSelector("table tbody tr");
    private By dreamNameColumn = By.cssSelector("td:nth-child(1)");
    private By daysAgoColumn = By.cssSelector("td:nth-child(2)");
    private By dreamTypeColumn = By.cssSelector("td:nth-child(3)");

    public DreamsDiaryPage(WebDriver driver) {
        this.driver = driver;
        this.wait = new WebDriverWait(driver, Duration.ofSeconds(15));
    }
```

```
    public void navigateToDreamsDiary() {
        driver.get("https://arjitnigam.github.io/myDreams/dreams-diary.html");
    }

    public int getDreamEntriesCount() {
        List<WebElement> rows = driver.findElements(dreamTableRows);
        return rows.size();
    }

    public boolean verifyDreamTypes() {
        List<WebElement> typeElements =
driver.findElements(dreamTypeColumn);
        for (WebElement element : typeElements) {
            String type = element.getText().trim();
            if (!type.equals("Good") && !type.equals("Bad")) {
                return false;
            }
        }
        return true;
    }

    public boolean verifyAllColumnsFilled() {
        List<WebElement> rows = driver.findElements(dreamTableRows);

        for (WebElement row : rows) {
            String name = row.findElement(dreamNameColumn).getText().trim();
```

```
            String daysAgo = row.findElement(daysAgoColumn).getText().trim();
            String type = row.findElement(dreamTypeColumn).getText().trim();

            if (name.isEmpty() || daysAgo.isEmpty() || type.isEmpty()) {
                return false;
            }
        }
        return true;
    }

    public int countRecurringDreams() {
        List<WebElement> nameElements =
driver.findElements(dreamNameColumn);
        int recurringCount = 0;

        for (int i = 0; i < nameElements.size(); i++) {
            String currentDream = nameElements.get(i).getText().trim();

            for (int j = i + 1; j < nameElements.size(); j++) {
                if (currentDream.equals(nameElements.get(j).getText().trim())) {
                    recurringCount++;
                    break;
                }
            }
        }
        return recurringCount;
    }
```

```
    public boolean verifySpecificRecurringDreams() {
        List<WebElement> nameElements =
driver.findElements(dreamNameColumn);
        boolean hasFlying = false;
        boolean hasMaze = false;

        for (WebElement element : nameElements) {
            String dreamName = element.getText().trim();
            if (dreamName.equals("Flying over mountains")) hasFlying = true;
            if (dreamName.equals("Lost in maze")) hasMaze = true;
        }

        return hasFlying && hasMaze;
    }
}
```

4. DreamsTotalPage.java – Page Object

java

```
package pages;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.ui.WebDriverWait;
import java.time.Duration;

public class DreamsTotalPage {
    private WebDriver driver;
    private WebDriverWait wait;

    // Locators - Update these based on actual website structure
    private By goodDreamsCount = By.id("goodDreams"); // or appropriate selector
    private By badDreamsCount = By.id("badDreams");
    private By totalDreamsCount = By.id("totalDreams");
    private By recurringDreamsCount = By.id("recurringDreams");

    public DreamsTotalPage(WebDriver driver) {
        this.driver = driver;
        this.wait = new WebDriverWait(driver, Duration.ofSeconds(15));
    }
```

```
    public void navigateToDreamsTotal() {
        driver.get("https://arjitnigam.github.io/myDreams/dreams-total.html");
    }

    public int getGoodDreamsCount() {
        return Integer.parseInt(driver.findElement(goodDreamsCount).getText());
    }

    public int getBadDreamsCount() {
        return Integer.parseInt(driver.findElement(badDreamsCount).getText());
    }

    public int getTotalDreamsCount() {
        return Integer.parseInt(driver.findElement(totalDreamsCount).getText());
    }

    public int getRecurringDreamsCount() {
        return
Integer.parseInt(driver.findElement(recurringDreamsCount).getText());
    }
}
```

5. DreamPortalTests.java – Test Class

java

```
package tests;

import org.testng.Assert;
import org.testng.annotations.Test;
import pages.HomePage;
import pages.DreamsDiaryPage;
import pages.DreamsTotalPage;
import utils.TestBase;
import java.util.List;

public class DreamPortalTests extends TestBase {

    @Test(priority = 1)
    public void testHomePageFunctionality() {
        HomePage homePage = new HomePage(driver);

        // Navigate to home page
        homePage.navigateToHomePage();

        // Verify loading animation
        homePage.verifyLoadingAnimation();

        // Verify main content visibility
```

```
        homePage.verifyMainContentVisibility();

        // Click My Dreams and verify tabs
        List<String> tabUrls = homePage.clickMyDreamsAndGetTabs();
        Assert.assertEquals(tabUrls.size(), 2, "Should open exactly 2 new tabs");

        // Verify URLs contain the expected pages
        boolean hasDiary = tabUrls.stream().anyMatch(url -> url.contains("dreams-diary"));
        boolean hasTotal = tabUrls.stream().anyMatch(url -> url.contains("dreams-total"));
        Assert.assertTrue(hasDiary && hasTotal, "Should open both dreams-diary and dreams-total pages");
    }

    @Test(priority = 2)
    public void testDreamsDiaryPage() {
        DreamsDiaryPage diaryPage = new DreamsDiaryPage(driver);
        diaryPage.navigateToDreamsDiary();

        // Verify exactly 10 dream entries
        int entryCount = diaryPage.getDreamEntriesCount();
        Assert.assertEquals(entryCount, 10, "Should have exactly 10 dream entries");

        // Verify dream types are only Good or Bad
        boolean validTypes = diaryPage.verifyDreamTypes();
```

```
        Assert.assertTrue(validTypes, "All dream types should be either 'Good' or
'Bad'");


        // Verify all columns are filled
        boolean allColumnsFilled = diaryPage.verifyAllColumnsFilled();
        Assert.assertTrue(allColumnsFilled, "All columns should be filled for each
row");
    }


    @Test(priority = 3)
    public void testDreamsTotalPage() {
        DreamsTotalPage totalPage = new DreamsTotalPage(driver);
        totalPage.navigateToDreamsTotal();


        // Verify statistics
        Assert.assertEquals(totalPage.getGoodDreamsCount(), 6, "Good dreams
count should be 6");
        Assert.assertEquals(totalPage.getBadDreamsCount(), 4, "Bad dreams
count should be 4");
        Assert.assertEquals(totalPage.getTotalDreamsCount(), 10, "Total dreams
count should be 10");
        Assert.assertEquals(totalPage.getRecurringDreamsCount(), 2, "Recurring
dreams count should be 2");
    }


    @Test(priority = 4)
    public void testRecurringDreamsLogic() {
        DreamsDiaryPage diaryPage = new DreamsDiaryPage(driver);
```

```
        diaryPage.navigateToDreamsDiary();

        // Verify recurring dreams logic
        int recurringCount = diaryPage.countRecurringDreams();
        Assert.assertEquals(recurringCount, 2, "Should identify 2 recurring dreams");

        // Verify specific recurring dreams
        boolean hasSpecificRecurring = diaryPage.verifySpecificRecurringDreams();
        Assert.assertTrue(hasSpecificRecurring, "Should contain 'Flying over mountains' and 'Lost in maze'");
    }
}
```

6. pom.xml - Maven Dependencies

```
xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.dreamportal</groupId>
    <artifactId>dream-portal-tests</artifactId>
    <version>1.0.0</version>

    <properties>
        <maven.compiler.source>11</maven.compiler.source>
        <maven.compiler.target>11</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
```

```
</properties>

<dependencies>
    <!-- Selenium -->
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>4.15.0</version>
    </dependency>

    <!-- TestNG -->
    <dependency>
        <groupId>org.testng</groupId>
        <artifactId>testng</artifactId>
        <version>7.8.0</version>
    </dependency>

    <!-- Allure Reporting -->
    <dependency>
        <groupId>io.qameta.allure</groupId>
        <artifactId>allure-testng</artifactId>
        <version>2.24.0</version>
    </dependency>
</dependencies>

<build>
```

```
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>3.1.0</version>
        <configuration>
          <testFailureIgnore>false</testFailureIgnore>
          <argLine>-
javaagent:${settings.localRepository}/org/aspectj/aspectjweaver/1.9.7/aspectj
weaver-1.9.7.jar</argLine>
        </configuration>
        <dependencies>
          <dependency>
            <groupId>org.aspectj</groupId>
            <artifactId>aspectjweaver</artifactId>
            <version>1.9.7</version>
          </dependency>
        </dependencies>
      </plugin>
    </plugins>
  </build>
</project>
```

7. README.md

markdown

 # Dream Portal Automated Tests

Overview

Selenium Java automation framework for testing the Dream Portal website.

Prerequisites

- Java 11+

- Maven

- Chrome Browser

Setup

1. Clone the repository

2. Update ChromeDriver path in `TestBase.java`

3. Run: `mvn clean test`

Test Reports

- Allure reports: `mvn allure:serve`

- HTML reports generated in `target/surefire-reports`

Project Structure

Uses Page Object Model (POM) design pattern for maintainable and scalable tests.