LU05.A01 Frontend Unit-Testing ergänzen

Erfassen Sie hierfür einen Task, sofern noch keiner existiert. Verweisen Sie in allen Commit-Messages dieser Aufgabe auf den entsprechenden Task.

Installieren Sie ein Testframework für JS/TS-Unit-Tests Ihrer Wahl (Bsp. Jest, Vitest).

```
npm install -D ...
```

Ergänzen Sie Ihr `packacke.json`, so dass Sie das Testen analog mit dem Formatieren und Linten mit einem Framework-unabhängigen Scriptnamen abstrahieren.

Jest	**Vitest**
`"scripts": { "format": "prettier --write .", "format-check": "prettier --check .", "lint": "eslint . --fix", "lint-check": "eslint .", "test": "jest", "test-coverage": "jest --coverage", "tsc": "tsc" }`	`"scripts": { "format": "prettier --write .", "format-check": "prettier --check .", "lint": "eslint . --fix", "lint-check": "eslint .", "test": "vitest", "test-coverage": "vitest run --coverage", "tsc": "tsc" }`

Fügen Sie in Ihrem Projekt einen entsprechenden Unit-Test hinzu. Die im Theorieteil erwähnten Repositories können Ihnen womöglich als Hilfe dienen.

Gliedern Sie Ihren Test unbedingt in die 3 Abschnitte `Arrange`, `Act`, `Assert`:

```
    // Arrange
    const testObject = ...;

    // Act
    const result = calculate(testObject);

    // Assert
    expect(result).not.toBeNull();
    expect(result.textContent).toBe('Test');
```

Es empfiehlt sich den Test lokal laufend laufen lassen, bevor Sie ihn committen.

```
npm run test
```

Ergänzen Sie Ihre `.github/workflows/frontend_job.yml`-Datei mit folgenden Schritt:

```
      - name: Install dependencies
        run: npm ci

      - name: Run Formatter check
        run: npm run format-check
```

```
      - name: Run Linter check
        run: npm run lint-check

      - name: Run tests
        run: npm run test-coverage
```