Testkonzept

Energiemanagement-Dashboard

Klassifizierung	intern
Status	in Arbeit
Programmname	EnergieWeb
Projektnummer	EMD-2025-01
Projektleiter	Projektleiter
Version	0.1
Datum	8. Juni 2025
Auftraggeber	Lehrveranstaltung Software Engineering
Autor/Autoren	Kimia Jamei
Verteiler	Kimia Jamei

Änderungsverzeichnis

Version	Datum	Änderung	Autor
0.1	11.06.25	Erstversion Testkonzept – Grundgerüst erstellt und Teststrategie dokumentiert	Kimia Jamei

Tabelle 1: Änderungskontrolle

Beschreibung

Das Projekt ist ein webbasiertes Dashboard zur Erfassung, Analyse und Visualisierung von Energieverbrauchsdaten für Haushaltsgeräte. Es erlaubt die Auswahl von Geräten, Energiearten und Analysealgorithmen über eine benutzerfreundliche Oberfläche und nutzt Observer- und Strategy-Pattern.

1 Testziele

Globale messbare Testziele über alle Testfälle hinweg:

Nr.	Beschreibung	Messgrösse	Priorität*
1	*Gerät hinzufügen funktioniert korrekt*	*Gerät erscheint*	*M*
2	*Energieverbrauch korrekt visualisiert*	*Diagramm korrekt*	*M*
3	Einheitenwahl beeinflusst Anzeige	*Werte umgerechnet*	*1*
4	*Analysefunktion liefert richtige Ausgaben*	*Ergebnis korrekt*	*1*
5	*Zeitraum-Dropdown filtert Verbrauchsdaten korrekt*	*Datenmenge korrekt*	*2*
6	*Diagrammtyp-Auswahl ändert Darstellung korrekt*	*Diagrammtyp wechselt*	*2*
**** Priorität: M = Muss / 1 = hoch, 2 = mittel, 3 = tief***			

Tabelle 2: Übergeordnete Testziele

2 Teststrategie und Teststufen

Zur Sicherstellung der Qualität kommt eine kombinierte Teststrategie aus manuellen UI-Tests und gezielten Modultests zum Einsatz. Die Tests erfolgen in drei aufeinander aufbauenden Stufen:

- Komponententests: Einzelne Module und Klassen werden isoliert auf korrekte Funktion geprüft.
- Integrationstests: Zusammenspiel mehrerer Komponenten wird getestet, insbesondere Datenfluss und Schnittstellen.
- UI-Tests: Die Benutzeroberfläche wird auf korrekte Darstellung und Interaktion überprüft.

Durch die klare Trennung der Teststufen werden Redundanzen vermieden und Fehler frühzeitig erkannt.
Testumgebung: Die Tests erfolgen in einer lokalen Umgebung mit modernen Browsern (Google Chrome, Mozilla Firefox).

3 Testobjekte

Nr.	Objekt	Beschreibung
1	*Gerätekomponente*	*Fügt Geräte mit Energie-Typ hinzu*
2	*ChartObserver*	*Visualisiert Werte mit Einheit und Diagrammtyp*
3	*Strategy-Klassen*	*Berechnen Analysewerte: Durchschnitt, Spitzenwert, Einsparpotenzial*
4	*Devicemanager*	*Verwaltet alle Geräte im System (Hinzufügen, Suchen, Auflisten)*
5	*Zeitraumfilter*	*Filtert Messdaten nach auswählbarem Zeitbereich (z. B. 5 Min, 15 Min, etc.)*

Nr.	Objekt	Beschreibung
6	*Einheit-Auswahl*	*Rechnet Verbrauchswerte je nach Einheit (W, kWh, J) korrekt um*
7	*Diagrammtyp-Auswahl*	*Wechselt zwischen Linien-, Balken- und Punktdiagramm*

Tabelle 3: Testobjekte

4 Testarten

Nr.	Testart	Beschreibung
1	*Modultest (Jest)*	*Automatisierter Test einzelner Klassen und Methoden mittels Jest*
2	*UI-Test (manuell)*	*Interaktives Testen der Benutzeroberfläche im Browser*

Tabelle 4: Testarten

5 Testabdeckung

5.1 Übersicht Testfälle

Nr.	Testobjekt	Testfälle
1	*Gerätekomponente*	*• Neues Gerät hinzufügen (mit Name + Energy-Type) → erscheint im Dashboard als neue Box* *• Gerät hinzufügen mit leerem Namen → kein Gerät wird erstellt (Abbruch)* *• Energie-Typ-Dropdown pro Gerät → ausgewählter Typ wird in Box angezeigt* *• Gerät entfernen (nicht doppelt anlegen)*
2	*ChartObserver*	*• Initiales Chart rendern → Canvas zeigt korrekte Linie über Zufallswerte* *• Zeitraum-Dropdown ändern (1 Min, 5 Min, All) → Chart zeigt nur Werte im gewählten Intervall* *• Diagrammtyp-Dropdown ändern (line/bar/scatter) → Chart wechselt Darstellung* *• Energy-Type-Label aktualisiert sich korrekt*
3	*Strategy-Klassen*	*• AverageStrategy.analyze([100, 200, 300]) → „Durchschnitt: 200.0 W“* *• PeakStrategy.analyze([100, 200, 300]) → „Spitze: 300 W“* *• SavingPotentialStrategy.analyze([100, 200, 300]) → z. B. „Sparpotenzial: –100.0 W“ (korrektes Rechnen)*
4	*DeviceManager*	*• addDevice(device) → manager.getDevice(name) liefert genau dieses Gerät* *• getDevice(unbekannterName) → liefert* `undefined`

Nr.	Testobjekt	Testfälle
		• *getAllDevices() nach mehrfachen Hinzufügungen → gibt alle Instanzen in der richtigen Reihenfolge zurück*

Tabelle 5: Testabdeckung

5.2 Beurteilung Testziele und Testabdeckung

Die zentralen Funktionalitäten des Dashboards wurden durch gezielte Modultests sowie manuelle UI-Tests abgedeckt. Die Testfälle decken alle Kernanforderungen ab – insbesondere die Geräteverwaltung, Visualisierung, Analysefunktionen und Einheitenauswahl. Die Testabdeckung wird als ausreichend beurteilt.

6 Testrahmen

6.1 Testvoraussetzungen

Tester, Vorkenntnisse
Für den Test unseres Energiemanagement-Dashboards müssen folgende Bedingungen erfüllt sein:

- Tester-Profil
  Ich(Kimia Jamei) teste das Energiemanagement-Dashboard und übernehme dabei Planung, Durchführung und Dokumentation. Ich verfüge über gute Kenntnisse in TypeScript, DOM-APIs, Chart.js, sowie im Einsatz von Jest und jsdom für automatisierte Tests.
- Hard-/Software
  – Node.js (Version≥16) und npm installiert
  – Lokaler Browser (Chrome oder Firefox) für manuelle UI-Smoketests
  – Terminal und VS Code
- Projekt-Setup
  1. Repository klonen und ins Verzeichnis wechseln
  2. `npm install` ausführen
  3. Build der TS-Quellen: `npx tsc`
  4. Anwendung starten: `npm run start` (lite-server öffnet `public/index.html`)
  5. Automatisierte Tests ausführen: `npm test` (führt Jest-Suits aus)

6.2 Mängelklassifizierung

Die festgestellten Mängel, bzw. die nicht erfüllten Anforderungen (Erwartungen), werden in Klassen von 1 bis 4 eingestuft. Die Klasse 0 findet nur dann Verwendung, wenn ein einwandfreies Ergebnis gesondert ausgewiesen werden soll:

Nr.	Mängelklassen	Beschreibung
0	mängelfrei	Einwandfrei und anforderungsgerecht
1	belangloser Mangel	Verwendung möglich, Brauchbarkeit ist vorhanden, Mängel sollte dennoch nicht vorkommen

Nr.	Mängelklassen	Beschreibung
2	leichter Mangel	Verwendung möglich, Brauchbarkeit ist nur wenig beeinträchtigt
3	schwerer Mangel	Verwendung ist noch möglich, Brauchbarkeit ist stark verringert
4	kritischer Mangel	Unbrauchbar; Wesentliche Funktionalität ist nicht gegeben; Betrieb ist nicht verantwortbar (z.B. sicherheitsspezifisch)

Tabelle 6: Mängelklassen

Die Klassifizierung spiegelt die Folgenschwere und den Aufwand zur Behebung der möglich feststellbaren Mängel. Die Zuordnung der festgestellten Mängel zu einer Mängelklasse gibt grob auch die Priorität vor, in welcher Reihenfolge die Behebung der Mängel angegangen werden soll.

Wird eine Mängelklasse zwischen 1-3 erreicht, kann das System/Produkt unter Vorbehalt abgenommen werden. Zur Behebung der Mängel sind jedoch Massnahmen zu definieren. Eine Nachprüfung ist zwingend.

Werden hingegen Mängel der Klasse 4 festgestellt, kann das System/Produkt nicht abgenommen werden und der Auftragnehmer muss umgehend Massnahmen treffen, um diese Mängel zu beheben. Der Auftragnehmer hat zudem die erneute Abnahme zu veranlassen.

6.3 Start- und Abbruchbedingungen

Vorbedingungen für Teststart

Startbedingungen:

Testumgebung bereitgestellt

Lokaler Web-Server (z. B. via npm run start) läuft und bedient /public-Ordner.

Browser-Instanz (Firefox) geöffnet mit DevTools-Konsole aktiviert.

Codekomponenten kompiliert

TypeScript-Transpilierung erfolgreich (npx tsc ohne Fehler).

Alle Modulimporte aufgelöst, main.js, ChartObserver.js etc. liegen unter public/src/....

Abhängigkeiten installiert

npm install ausgeführt, Chart.js via CDN erreichbar.

Simulationsdaten aktiv

Zufallsdaten-Generator (setInterval) läuft, erste Messwerte sind bereits in den Charts zu sehen.

Abbruchbedingungen:

Schwerwiegende Fehler im Build/Compile (z. B. fehlende Importe, TS-Fehler)

Server startet nicht oder liefert HTTP-Fehler auf /index.html

Chart.js nicht geladen (CDN-Fehler) oder Canvas-Initialisierung schlägt fehl

Keine Messwerte innerhalb von 30 Sekunden nach Teststart (Timer defekt)

Testumgebung instabil (Browser-Crashes, Netzwerkunterbrechung)

In allen Abbruchfällen werden die Tests sofort gestoppt und die Ursache in der Mängelklassifizierung entsprechend (Klasse 2–4) dokumentiert.

7 Testumgebung

Die Tests werden lokal auf meinem Entwicklungsrechner durchgeführt. Verwendete Browser:

Mozilla Firefox (aktuelle Version): Die Build- und Laufzeitumgebung besteht aus Node.js (v18) und lite-server für das Live-Reloading.

8 Testinfrastruktur

8.1 Testsystem

- Betriebssystem: macOS Monterey
- Node.js v18.x, npm v9.x
- Lite-Server zur Auslieferung der statischen public-Dateien
- Jest v29 als Test-Runner für Unit- und Integrationstests

8.2 Testdaten

- Simulierte Verbrauchswerte (Zufallszahlen 100…500 W bzw. umgerechnet)
- Standardgeräte „Kühlschrank“, „Heizung“, „Waschmaschine“
- Diverse Energie-Typen (Strom, Gas, Wasser)

8.3 Testhilfsmittel

- **Jest** für Unit-Tests der Device, DeviceManager und Strategy-Klassen
- **jsdom** (über Jest) für DOM-Tests von ChartObserver
- **Browser-DevTools** zur manuellen UI-Verifikation

9 Testorganisation

Alle Tests führe ich selbst als Entwickler und Tester durch. Code-Reviews und Pair-Programming unterstützen die Testqualität.

10 Testfallbeschreibungen

ID / Bezeichnung	*T-001*	*Füge neues Gerät hinzu (Requirement “Geräte hinzufügen”)*	
Beschreibung	*Prüft, ob Formular ein neues Gerät mit gewähltem Typ anlegt*		
Testvoraussetzung	*Dashboard und Formular offen*		
Testschritte	*1. Form öffnen* *2. Namen „Lampe“ eingeben* *3. Energie-Typ „Strom“ wählen* *4. Submit klicken*		
Erwartetes Ergebnis	*Gerät „Lampe“ erscheint in Liste, Typ „Strom“ wird korrekt angezeigt*		

ID / Bezeichnung	*T-002*	*Chart-Update (Requirement "Chart aktualisieren")*	
Beschreibung	*Prüft, ob ChartObserver nach 3 Sekunden neuen Punkt zeigt*		
Testvoraussetzung	*Gerät „Kühlschrank" vorhanden*		
Testschritte	*1. Dashboard öffnen* *2. 3 Sekunden warten* *3. Chart prüfen*		
Erwartetes Ergebnis	*Mindestens ein neuer Datenpunkt im Linien-Chart sichtbar*		

ID / Bezeichnung	*T-003*	*Einheit-Wechsel (Requirement "Einheiten umrechnen")*	
Beschreibung	*Prüft, ob Dropdown-Einheit Umrechnung anwendet*		
Testvoraussetzung	*Gerät mit Datenpunkten*		
Testschritte	*1. Einheit-Dropdown wählen „kWh"* *2. Chart-Werte prüfen*		
Erwartetes Ergebnis	*Chart-Y-Achse Werte um Faktor 0.001 skaliert (W→kWh)*		

Tabelle 7: Testfallbeschreibung

11 Testplan

Nr.	Aktivität	Verantwortlich	Mitarbeit	Termin
1	*Einrichtung der Testumgebung*	*Ich*	*-*	*11.06.25*
2	*Schreiben der Unit-Tests (Jest)*	*Ich*	*-*	*11.06.25*
3	*Manuelle UI-Tests im Browser*	*Ich*	*-*	*11.06.25*
4	*Integrationstests & Regression*	*Ich*	*-*	*11.06.25*

Tabelle 8: Testplan

Testkonzept
Energiemanagement-Dashboard

Inhaltsverzeichnis

1 Testziele 2
2 Teststrategie und Teststufen 2
3 Testobjekte 2
4 Testarten 3
5 Testabdeckung 3
5.1 Übersicht Testfälle 3
5.2 Beurteilung Testziele und Testabdeckung 4
6 Testrahmen 4
6.1 Testvoraussetzungen 4
6.2 Mängelklassifizierung 4
6.3 Start- und Abbruchbedingungen 6
7 Testumgebung 6
8 Testinfrastruktur 7
8.1 Testsystem 7
8.2 Testdaten 7
8.3 Testhilfsmittel 7
9 Testorganisation 7
10 Testfallbeschreibungen 7
11 Testplan 8

Tabellenverzeichnis

Tabelle 1: Änderungskontrolle 1
Tabelle 2: Übergeordnete Testziele 2
Tabelle 3: Testobjekte 3
Tabelle 4: Testarten 3
Tabelle 5: Testabdeckung 4
Tabelle 6: Mängelklassen 5
Tabelle 7: Testfallbeschreibung 8
Tabelle 8: Testplan 8