Unit Test Plan

- for Diabetter, a Diabetes data dashboard

Team members - Group 3

Alexandra Nikolova	- 1339311
Rik Litjens	- 1317059
Vasil Shteriyanov	- 1307282
Gabriela Zanova	- 1307509
Konstantin Velkov	- 1307436
Roel Koopman	- 1299743
Rinse Vlaswinkel	- 1312529
Nikaels Balasovs	- 1250221
Jeroen Gijsbers	- 1305832
Kevin Dirksen	- 1302191

Client & Platform Owner
M. Chaudron & P. Van Gorp

Project Managers
A. Pintea & C. Olteanu

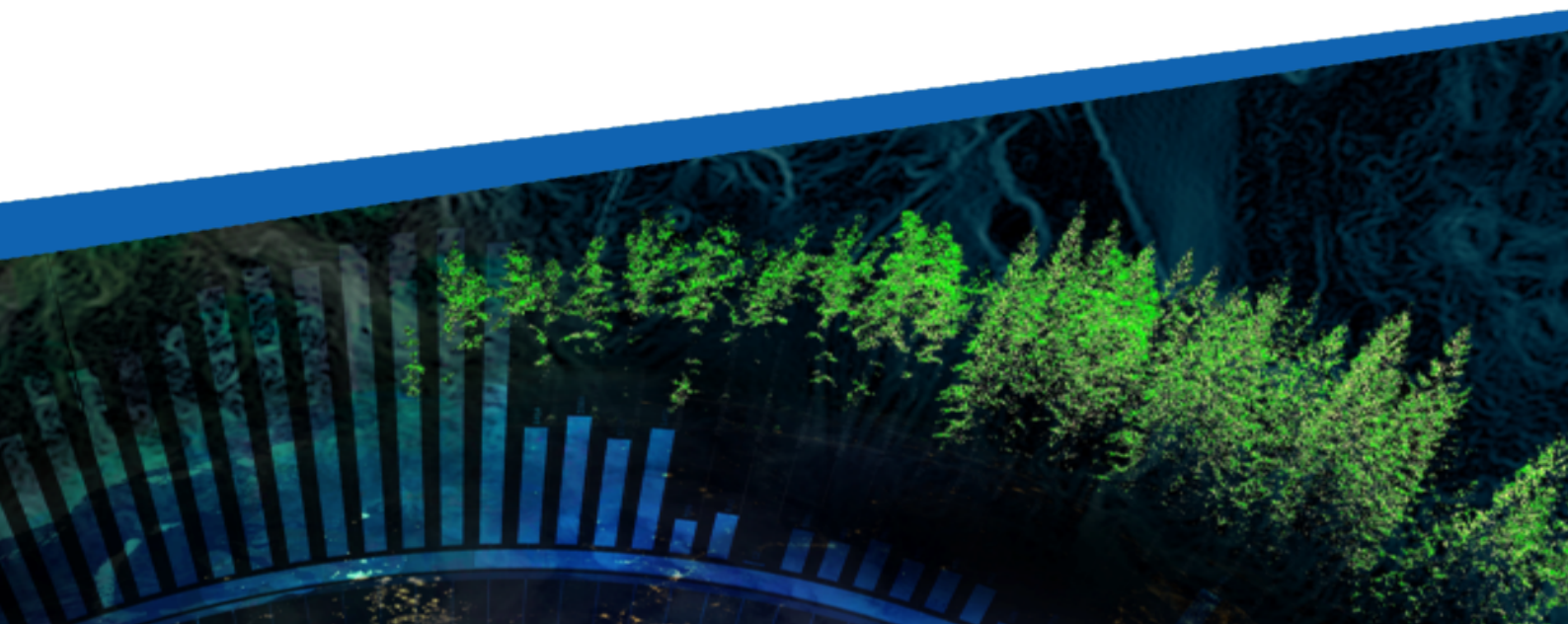
Supervisor
G. Kahraman

Eindhoven University of Technology

Department of Mathematics and Computer Science

Diabetter

Eindhoven, 2021



Abstract

This Unit Test Plan (UTP) elaborates on all unit test procedures for Diabetter, a web-based data dashboard for people living with Diabetes. While using functionalities of the GameBus platform, it does not only give users insights into several aspects of their life but also shows correlations between them. This document is in line with the ESA software standards [1].

Contents

Document status sheet and change records

Document title	Unit test plan
Document identifier	Diabetter.UTP/2.0
Authors	Rinse Vlaswinkel Rik Litjens
Document status	Version 2.0 - Final

Document history

Version	Date	Authors	Description
0.1	17/05/2021	Rinse Vlaswinkel, Rik Litjens	Template, chapters 2 & 3
0.2	20/05/2021	Rinse Vlaswinkel, Rik Litjens	Introduction
0.3	24/05/2021	Rinse Vlaswinkel, Rik Litjens	Chapter 3
1.0	31/05/2021	Rinse Vlaswinkel, Rik Litjens	Chapter 4 + Additions to chapter 3
1.1	06/06/2021	Rinse Vlaswinkel, Rik Litjens	Chapter 3
1.2	11/06/2021	Rinse Vlaswinkel, Rik Litjens	Chapter 3
1.3	14/06/2021	Rinse Vlaswinkel, Rik Litjens	Chapter 3
2.0	28/06/2021	Rinse Vlaswinkel, Rik Litjens	Final version

Change records

Version	Changes
0.1	Added template for all chapters, added sections 2.2, 2.4, 2.5, 2.6, added test cases to chapter 3
0.2	Added content of introduction
0.3	Added more test cases to chapter 3
1.0	Made test procedure and added more test cases to chapter 3
1.1	Added more test cases to chapter 3
1.2	Added more test cases to chapter 3
1.3	Added more test cases to chapter 3
2.0	Finalized section 3, added text to section 1 and 2, added section 5

1 Introduction

1.1 Purpose

In this Unit Test Plan (UTP) all unit tests of the Diabetter dashboard are described. These tests must ensure that all sub-components of the dashboard are working as intended and that their robustness is guaranteed. In order to increase the readability of the document, these tests are subdivided into groups. Besides elaborating on the tests itself, the document will also present the results of the tests as well as the code coverage.

1.2 Overview

This document consists of five sections. The remainder of this first section provides definitions for relevant terms and abbreviations as well as listing all references used in the document. The second section describes all features that are to be tested, the test deliverables, how to prepare and carry out the testing, some constraints/prerequisites on the environment and fail/pass criteria. Section 3 provides a detailed description of all unit tests used in the project and section 4 mentions the test procedures that should be followed to correctly execute all tests. Finally, section five describes the coverage for the program files of our project and gives the result of each test. When all tests are properly run, the results should match these.

1.3 List of definitions and abbreviations

Some of the used terms, phrases, and abbreviations might be ambiguous. Therefore we include all relevant definitions in Table 1 and 2.

1.3.1 Definitions

Mock call	An API call that is not actually executed but simulated by us.

Table 1: Definitions of terms that are used in this document

1.3.2 Abbreviations

GB	GameBus
OD	OneDrive
URL	Uniform Resource Locator (web address)

Table 2: Definitions of abbreviations that are used in this document

1.4 List of references

References

[1] E.B. for software Standardisation and Control. Esa software engineering standards. 1991.

[2] Diabetter. Software design document. 2021.

[3] Diabetter. User requirements document. 2021.

[4] Diabetter. Software user manual. 2021.

[5] Jest. https://www.npmjs.com/package/jest.

[6] Jest fetch mock. https://www.npmjs.com/package/jest-fetch-mock.

[7] Supertest. https://www.npmjs.com/package/supertest.

2 Test plan

2.1 Test items

The application that is being tested is the Diabetter dashboard, a data dashboard that connects several data sources relevant to people living with type 1 Diabetes. More information about the dashboard can be found in the SDD [2], URD [3] and SUM [4]. All backend functionalities were tested and these tests are described in this document. The frontend componentes were not tested through unit tests, but by visual inspection of the website.

2.2 Features to be tested

All backend features that are implemented will be tested, this includes any API calls which will be mocked to ensure proper functionality. Furthemore, test files are used to test the functionality of the file and data parsers and a dedicated test database is created to test the features that involve the local database. Any dependency packages are not tested. The frontend will also not be tested.

2.3 Test deliverables

Before executing the test plan, the following must have been delivered:

1. The finalized version of the Diabetter dashboard (i.e. code and documentation).
2. Sections one through four of this document.

Then, after these have been delivered, the following has to be included in this document:

1. The fifth and final section of this document, containing coverage and test results.
2. A problem report, if any errors occur during the execution of the test plan.

2.4 Testing tasks

In order to successfully run the tasks, the following steps should be taken:

- Download & install the LTS version of Node.js, as of writing, this is version 14.17.0.
- While in the base directory of the project, run `npm install`, this will install all the necessary packages for running and testing the project.

2.5 Environmental needs

To run the tests, Node.js should be installed on the machine running Windows 10 with the steps mentioned above. The necessary packages should also have been installed using npm, the package manager of Node.js. For this, an internet connection is needed. Our test environment uses the Jest [5] testing framework together with Jest Fetch Mock [6] and SuperTest [7] for mocking API requests and testing our own endpoints respectively.

2.6 Test case pass/fail criteria

A test case passes if the actual output is identical to the expected output. The test plan succeeds if all tests pass. The test plan fails if not all tasks described in section 3 of this document pass.

3 Test case specifications

3.1 Server - Basics

3.1.1 GET server root

SERV - 1	GET root
Tests whether sending a get request to the server root returns the expected message	
Test items:	`test/routes.test.ts`
Precondition:	server is running
Input:	-
Output:	whether the expected response is sent back

3.1.2 GET test endpoint

SERV - 2	GET /test endpoint
Tests whether sending a get request to a test endpoint returns the expected message	
Test items:	`test/routes.test.ts`
Precondition:	server is running
Input:	-
Output:	whether the expected response is sent back

3.1.3 GET clean endpoint - cleaning login attempts

SERV - 3	GET /clean endpoint
Tests whether sending a get request to the clean endpoint successfully cleans the login attempts in the database	
Test items:	`test/routes.test.ts`
Precondition:	server is running
Input:	-
Output:	whether the response status is 200, indicating success of the database clean operation

3.2 Authentication

3.2.1 Creating and decoding a JWT

AUTH - 1	createJWT()
Tests whether the creation and decoding of a JWT is correct	
Test items:	`test/auth/auth.test.ts`
Precondition:	token secret is provided
Input:	user ID, access token, refresh token, token secret, expiry time and issuer
Output:	whether the created and decoded JWT contains the input

3.2.2 Starting multiple login attempts

AUTH - 2	startLoginAttempt()
Tests whether starting two login attempts is handled correctly, meaning the first is registered and the second is discarded by returning undefined	
Test items:	`test/auth/auth.test.ts`
Precondition:	-
Input:	user ID
Output:	whether the first login attempt is registered and the second is not

3.2.3 Finishing a non-existing login session

AUTH - 3	finishLoginAttempt()
Tests if finishing a login attempt is rejected when it does not exist	
Test items:	`test/auth/auth.test.ts`
Precondition:	token secret, expiry time and issuer of the JWT are set
Input:	login token
Output:	whether finishing the non-existing login attempt is rejected by returning undefined

3.2.4 Register GameBus callback without open login session

AUTH - 4	registerConnectCallback()
Tests if GameBus callbacks are rejected when there is no open login session	
Test items:	`test/auth/auth.test.ts`
Precondition:	token secret, expiry time and issuer of the JWT are set, no login session is open
Input:	user ID, access token, refresh token
Output:	whether the function rejects the registration of the GameBus callback by returning false

3.2.5 Starting a new login again after GameBus callback

AUTH - 5	startLoginAttempt(), registerConnectCallback(), finishLoginAttempt()
Tests whether starting a new login attempt is rejected after a previous one with registered connect callback is already in progress and that the original attempt can still be finished afterwards	
Test items:	`test/auth/auth.test.ts`
Precondition:	token secret, expiry time and issuer of the JWT are set
Input:	user ID, access token, refresh token
Output:	whether the second login attempt is rejected by returning undefined and the original login attempt can be finished (i.e. a Jwt is created)

3.2.6 Supervisor authentication

AUTH - 6	logToken(), request(), getSupervisors(), getToken(), getApproved(), getChildren(), checkSupervisor(), retractPermission()
Tests whether a full supervisor authentication and de-authentication process is executed as expected	
Test items:	`test/auth/supervisorUtils.test.ts`
Precondition:	database location is specified
Input:	childEmail, supervisorEmail, loginToken
Output:	whether all steps in a full supervisor authentication process are correctly completed and return expected values

3.2.7 Refreshing JWT

AUTH - 7	refreshJWT()
Tests if a JWT can be refreshed	
Test items:	`test/auth/auth.test.ts`
Precondition:	token secret, expiry time and issuer of the JWT are set
Input:	user ID, refresh token
Output:	whether a new JWT is returned

3.2.8 Login attempt with undefined user ID

AUTH - 8	startLoginAttempt()
Tests if a starting a login attempt with undefined user ID is rejected by returning undefined	
Test items:	`test/auth/auth.test.ts`
Precondition:	-
Input:	empty email (converted to undefined user ID)
Output:	whether undefined is returned

3.3 Diabetter database client

3.3.1 Database initialization

DB - 1	initialize()
Tests whether the SQLite database can be created and initialized by creating all tables without errors	
Test items:	`test/db/db.test.ts`
Precondition:	database location is specified
Input:	-
Output:	whether the database is created and initialized without errors

3.3.2 Database - Full login procedure

DB - 2	registerLoginAttempt(), registerCallback(), getLoginAttemptByLoginToken(), removeFinishedLoginAttempt()
Tests whether all database actions in a full login procedure for registering, getting or removing login attempts and callbacks work as expected	
Test items:	`test/db/db.test.ts`
Precondition:	database location is specified
Input:	user ID, login token, expire time, access token, refresh token
Output:	whether all database actions are completed as expected and return expected values

3.3.3 Database - Try to clean non-expired login attempts

DB - 3	cleanLoginAttempts()
Tests whether expired login attempts can be removed and a registered non-expired attempt can still be retrieved afterwards	
Test items:	`test/db/db.test.ts`
Precondition:	database location is specified
Input:	user ID, login token, expire time
Output:	whether the clean function returns true to indicate expired attempts have been removed and that the non-expired attempt can still be retrieved and is defined

3.3.4 Database - Clean expired login attempts

DB - 4	cleanLoginAttempts()
Tests whether expired login attempts can be removed and cannot be retrieved afterwards	
Test items:	`test/db/db.test.ts`
Precondition:	database location is specified
Input:	-
Output:	whether the clean function returns true to indicate expired attempts have been removed and that the expired attempt cannot be retrieved and is therefore undefined

3.3.5 Database - Register GameBus callback for non-existing login attempt

DB - 5	registerCallback()
Tests if registering callbacks for non-existing login attempts is rejected	
Test items:	`test/db/db.test.ts`
Precondition:	database location is specified
Input:	user ID, access token, refresh token
Output:	whether registering the callback is rejected by returning a false indicator

3.3.6 Database - Trying to retrieve non-existing login attempts

DB - 6	getLoginAttemptByPlayerId(), getLoginAttemptByLoginToken()
Tests if trying to retrieve non-existing login attempts is handled correctly	
Test items:	`test/db/db.test.ts`
Precondition:	database location is specified
Input:	user ID, login token
Output:	whether trying to retrieve the login attempt returns undefined for both get functions

3.3.7 Database - Register double login attempt

DB - 7	registerLoginAttempt()
Tests if registering a second login attempt is rejected	
Test items:	`test/db/db.test.ts`
Precondition:	database location is specified
Input:	user ID, login token, expire time
Output:	whether registering a first login attempt is successful (a true indicator is returned) and registering a second login attempt is rejected by returning a false indicator

3.3.8 Database - Register a file parse event and retrieve it

DB - 8	registerFileParse(), getLastUpdate()
Tests if the event of parsing a file can be logged (with user ID, file name and timestamp) and can be retrieved with user ID and file name	
Test items:	`test/db/db.test.ts`
Precondition:	database location is specified
Input:	user ID, file name, timestamp
Output:	whether registering a file parse event is successful and whether it can be retrieved afterwards

3.3.9 Database - Register and update a file parse event and retrieve it

DB - 9	registerFileParse(), getLastUpdate()
Tests if the event of parsing a file can be logged (with user ID, file name and timestamp) and this entry can be updated correctly and can be retrieved with user ID and file name afterwards	
Test items:	`test/db/db.test.ts`
Precondition:	database location is specified
Input:	user ID, file name, timestamp
Output:	whether registering a file parse event is successful, it can be updated and it can be retrieved afterwards

3.3.10 Database - Register and update multiple file parse events and retrieve it

DB - 10	registerFileParse(), getLastUpdate()
Tests if the event of parsing multiple files is logged (with user IDs, file names and timestamps), the entries are updated correctly and can be retrieved with user ID and file name afterwards for different users and files	
Test items:	`test/db/db.test.ts`
Precondition:	database location is specified
Input:	user ID, file name, timestamp
Output:	whether registering a file parse event is successful, it can be updated and it can be retrieved for multiple players

3.3.11 Database - Log token of child

DB - 11	logToken()
Tests if the token of a 'child' (normal) user can be logged such it can be used by the supervisor	
Test items:	`test/db/db.test.ts`
Precondition:	database location is specified
Input:	childEmail, token
Output:	whether the token is successfully logged in the database

3.3.12 Database - Requesting supervisor role

DB - 12	requestSupervisor()
Tests if a supervisor can successfully request a 'child' (normal) user to become their supervisor	
Test items:	`test/db/db.test.ts`
Precondition:	database location is specified
Input:	supervisorEmail, childEmail
Output:	whether the supervisor role has been successfully requested and stored in the database

DB - 13	getRequestedSupervisors()
Tests if a list of aspiring supervisors can be retrieved for a specific 'child' (normal) user	
Test items:	`test/db/db.test.ts`
Precondition:	database location is specified
Input:	childEmail
Output:	whether the list of requesting supervisors is successfully retrieved from the database

3.3.13 Database - Confirming requested supervisor role

DB - 14	confirmSupervisor()
Tests if a supervisor request can be confirmed by the user	
Test items:	`test/db/db.test.ts`
Precondition:	database location is specified
Input:	supervisorEmail, childEmail
Output:	whether a supervisor request can be made and the status of the supervisor request is correctly updated in the database

3.3.14 Database - Get all supervisors for a user

DB - 15	getApprovedSupervisors()
Tests if a list of approved supervisors can be retrieved for a 'child' (normal) user	
Test items:	`test/db/db.test.ts`
Precondition:	database location is specified
Input:	supervisorEmail, childEmail
Output:	whether a supervisor is approved as expected and the correct approved supervisors are retrieved from the database

3.3.15 Database - Check if a user is a supervisor

DB - 16	checkRole()
Tests if the function can correctly indicate whether a user is a supervisor or not. For an actual supervisor, true should be returned whereas a 'child' (normal) user must yield false.	
Test items:	`test/db/db.test.ts`
Precondition:	database location is specified
Input:	supervisorEmail, childEmail
Output:	whether the function correctly retrieves if a user is a supervisor or not from the database for the case when it is and the case when it is not

3.3.16 Database - Get 'child' (normal) users for a supervisor

DB - 17	getChildren()
Tests if a list of 'child' (normal) users that a supervisor supervises can be retrieved	
Test items:	`test/db/db.test.ts`
Precondition:	database location is specified
Input:	supervisorEmail
Output:	whether the correct list of 'child' (normal) users is retrieved from the database

3.3.17 Database - Retract supervisor permission

DB - 18	retractPermission()
Tests if a 'child' (normal) user can succesfully retract the supervisor permissions of one of his supervisors	
Test items:	`test/db/db.test.ts`
Precondition:	database location is specified
Input:	supervisorEmail, childEmail
Output:	whether the supervisor permission has been successfully retracted, meaning the supervisor is not a supervisor anymore (in this test case)

3.3.18 Database - Retrieve non existing parse event

DB - 19	getLastUpdate()
Tests if trying to get the last updated timestamp of a non-existing file for a non-existing user results in 0	
Test items:	`test/db/db.test.ts`
Precondition:	database location is specified
Input:	non-existing user ID, non-existing file name
Output:	whether retrieving a non-existing file parse event gives a timestamp of 0

3.3.19 Database - Execute methods while database does not exist

DB - 20	All db methods that change or get data from the tables
Tests if trying to execute methods on a non-existing database makes the methods return their default on-error values	
Test items:	`test/db/db.test.ts`
Precondition:	database location is specified
Input:	-
Output:	whether the correct on-error values are returned

3.4 GameBus API

For the GameBus API tests, all requests are mocked which means we will solely be looking at the properties of the request for correctness and disregard any response. It is assumed the environment is setup correctly to use the `jest-fetch-mock` [6] library to mock the requests.

3.4.1 Retrieving general activities

GB - 1	getActivityById()
Tests whether the correct request is being made	
Test items:	`test/gb/activity.test.ts`
Precondition:	request is mocked, client is created
Input:	activity ID
Output:	whether the query URL matches the expected URL, the access token is used and the request has only been made once

GB - 2	getActivitiesOnDate()
Tests whether the correct request is being made	
Test items:	`test/gb/activity.test.ts`
Precondition:	request is mocked, client is created
Input:	date
Output:	whether the query URL matches the expected URL, the access token is used and the request has only been made once

GB - 3	getAllActivitiesBetweenDate()
Tests whether the correct request is being made	
Test items:	`test/gb/activity.test.ts`
Precondition:	request is mocked, client is created
Input:	start date, end date
Output:	whether the query URL matches the expected URL, the access token is used and the request has only been made once

GB - 4	getActivitiesOnUnixDate()
Tests whether the correct request is being made	
Test items:	`test/gb/activity.test.ts`
Precondition:	request is mocked, client is created
Input:	date as unix timestamp in milliseconds
Output:	whether the query URL matches the expected URL, the access token is used and the request has only been made once

GB - 5	getAllActivitiesBetweenUnix()
Tests whether the correct request is being made	
Test items:	`test/gb/activity.test.ts`
Precondition:	request is mocked, client is created
Input:	start date and end date as unix timestamps in milliseconds
Output:	whether the query URL matches the expected URL, the access token is used and the request has only been made once

GB - 6	getAllActivitiesWithGd()
Tests whether the correct request is being made	
Test items:	`test/gb/activity.test.ts`
Precondition:	request is mocked, client is created
Input:	list of game descriptor translation keys
Output:	whether the query URL matches the expected URL, the access token is used and the request has only been made once

GB - 7	getAllActivitiesBetweenUnixWithGd()
Tests whether the correct request is being made	
Test items:	`test/gb/activity.test.ts`
Precondition:	request is mocked, client is created
Input:	list of game descriptor translation keys, start & end date (as unix timestamps)
Output:	whether the query URL matches the expected URL, the access token is used and the request has only been made once

GB - 8	getActivitiesOnUnixDateWithGd()
Tests whether the correct request is being made	
Test items:	`test/gb/activity.test.ts`
Precondition:	request is mocked, client is created
Input:	list of game descriptor translation keys, date (as unix timestamp)
Output:	whether the query URL matches the expected URL, the access token is used and the request has only been made once

3.4.2 Retrieving exercise activities

GB - 9	getAllExerciseActivities()
Tests whether the correct request is being made	
Test items:	`test/gb/exercise.test.ts`
Precondition:	request is mocked, client is created
Input:	-
Output:	whether the query URL matches the expected URL, the access token is used and the request has only been made once

GB - 10	getExerciseActivityFromGd()
Tests whether the correct request is being made	
Test items:	`test/gb/exercise.test.ts`
Precondition:	request is mocked, client is created
Input:	list of game descriptor translation keys
Output:	whether the query URL matches the expected URL, the access token is used and the request has only been made once

GB - 11	getExerciseActivityFromGdBetweenUnix()
Tests whether the correct request is being made	
Test items:	`test/gb/exercise.test.ts`
Precondition:	request is mocked, client is created
Input:	list of game descriptor translation keys, start and end timestamps
Output:	whether the query URL matches the expected URL, the access token is used and the request has only been made once

GB - 12	getAllExerciseActivitiesBetweenUnix()
Tests whether the correct request is being made	
Test items:	`test/gb/exercise.test.ts`
Precondition:	request is mocked, client is created
Input:	start and end timestamps
Output:	whether the query URL matches the expected URL, the access token is used and the request has only been made once

GB - 13	getExerciseActivityFromGdOnUnixDate()
Tests whether the correct request is being made	
Test items:	`test/gb/exercise.test.ts`
Precondition:	request is mocked, client is created
Input:	list of game descriptor translation keys, date timestamp
Output:	whether the query URL matches the expected URL, the access token is used and the request has only been made once

GB - 14	getAllExerciseActivitiesOnUnixDate()
Tests whether the correct request is being made	
Test items:	`test/gb/exercise.test.ts`
Precondition:	request is mocked, client is created
Input:	date timestamp
Output:	whether the query URL matches the expected URL, the access token is used and the request has only been made once

3.4.3 Retrieving glucose activities

GB - 15	getGlucoseActivities()
Tests whether the correct request is being made	
Test items:	`test/gb/glucose.test.ts`
Precondition:	request is mocked, client is created
Input:	-
Output:	whether the query URL matches the expected URL, the access token is used and the request has only been made once

GB - 16	getGlucoseActivitiesBetweenUnix()
Tests whether the correct request is being made	
Test items:	`test/gb/glucose.test.ts`
Precondition:	request is mocked, client is created
Input:	start and end date timestamps
Output:	whether the query URL matches the expected URL, the access token is used and the request has only been made once

3.4.4 Retrieving insulin activities

GB - 17	getInsulinActivities()
Tests whether the correct request is being made	
Test items:	`test/gb/insulin.test.ts`
Precondition:	request is mocked, client is created
Input:	-
Output:	whether the query URL matches the expected URL, the access token is used and the request has only been made once

GB - 18	getInsulinActivitiesBetweenUnix()
Tests whether the correct request is being made	
Test items:	`test/gb/insulin.test.ts`
Precondition:	request is mocked, client is created
Input:	start and end date timestamps
Output:	whether the query URL matches the expected URL, the access token is used and the request has only been made once

GB - 19	getInsulinActivitiesOnUnixDate()
Tests whether the correct request is being made	
Test items:	`test/gb/insulin.test.ts`
Precondition:	request is mocked, client is created
Input:	date timestamp
Output:	whether the query URL matches the expected URL, the access token is used and the request has only been made once

3.4.5 Retrieving food activities

GB - 20	getAllFoodActivities()
Tests whether the correct request is being made	
Test items:	`test/gb/food.test.ts`
Precondition:	request is mocked, client is created
Input:	-
Output:	whether the query URL matches the expected URL, the access token is used and the request has only been made once

GB - 21	getFoodActivitiesBetweenUnix()
Tests whether the correct request is being made	
Test items:	`test/gb/food.test.ts`
Precondition:	request is mocked, client is created
Input:	start and end date timestamps
Output:	whether the query URL matches the expected URL, the access token is used and the request has only been made once

GB - 22	getFoodActivitiesOnUnixDate()
Tests whether the correct request is being made	
Test items:	`test/gb/food.test.ts`
Precondition:	request is mocked, client is created
Input:	date timestamp
Output:	whether the query URL matches the expected URL, the access token is used and the request has only been made once

3.4.6 Retrieving mood activities

GB - 23	getAllMoodActivities()
Tests whether the correct request is being made	
Test items:	`test/gb/mood.test.ts`
Precondition:	request is mocked, client is created
Input:	-
Output:	whether the query URL matches the expected URL, the access token is used and the request has only been made once

GB - 24	getMoodActivitiesBetweenUnix()
Tests whether the correct request is being made	
Test items:	`test/gb/mood.test.ts`
Precondition:	request is mocked, client is created
Input:	start and end date timestamps
Output:	whether the query URL matches the expected URL, the access token is used and the request has only been made once

GB - 25	getMoodActivitiesOnUnixDate()
Tests whether the correct request is being made	
Test items:	`test/gb/mood.test.ts`
Precondition:	request is mocked, client is created
Input:	date timestamp
Output:	whether the query URL matches the expected URL, the access token is used and the request has only been made once

3.4.7 Retrieving BMI activities

GB - 26	getBMIActivities()
Tests whether the correct request is being made	
Test items:	`test/gb/bmi.test.ts`
Precondition:	request is mocked, client is created
Input:	-
Output:	whether the query URL matches the expected URL, the access token is used and the request has only been made once

GB - 27	getLatestBMIActivity()
Tests whether the correct request is being made	
Test items:	`test/gb/bmi.test.ts`
Precondition:	request is mocked, client is created
Input:	-
Output:	whether the query URL matches the expected URL, the access token is used and the request has only been made once

3.4.8 Converting response to models

GB - 28	convertResponseToExerciseModels()
Tests whether a response from GameBus is correctly converted to an ExerciseModel	
Test items:	`test/gb/exercise.test.ts`
Precondition:	response is correctly formatted
Input:	response
Output:	whether the response is correctly converted to a model

GB - 29	convertResponseToGlucoseModels()
Tests whether a response from GameBus is correctly converted to an GlucoseModel	
Test items:	`test/gb/glucose.test.ts`
Precondition:	response is correctly formatted
Input:	response
Output:	whether the response is correctly converted to a model

GB - 30	convertResponseToInsulinModels()
Tests whether a single response from GameBus is correctly converted to an InsulinModel with insulin type 0 (Rapid acting)	
Test items:	`test/gb/insulin.test.ts`
Precondition:	response is correctly formatted
Input:	response
Output:	whether the response is correctly converted to a model

GB - 31	convertResponseToInsulinModels()
Tests whether a single response from GameBus is correctly converted to an InsulinModel with insulin type 1 (Long acting)l	
Test items:	`test/gb/insulin.test.ts`
Precondition:	response is correctly formatted
Input:	response
Output:	whether the response is correctly converted to a model

GB - 32	convertResponseToMoodModels()
Tests whether a response from GameBus is correctly converted to a MoodModel	
Test items:	`test/gb/mood.test.ts`
Precondition:	response is correctly formatted
Input:	response
Output:	whether the response is correctly converted to a model

GB - 33	convertResponseToBMIModels()
Tests whether a response from GameBus is correctly converted to a BMIModel	
Test items:	`test/gb/bmi.test.ts`
Precondition:	response is correctly formatted
Input:	response
Output:	whether the response is correctly converted to a model

3.4.9 Sending glucose activities

GB - 34	postSingleGlucoseActivity()
Tests whether a GlucoseModel is correctly posted to GameBus	
Test items:	`test/gb/glucose.test.ts`
Precondition:	request is mocked, client is created
Input:	GlucoseModel
Output:	whether the body of the request is correct, the query URL matches the expected URL, the access token is used and the request has only been made once

GB - 35	postMultipleGlucoseActivities()
Tests whether multiple GlucoseModels are correctly posted to GameBus	
Test items:	`test/gb/glucose.test.ts`
Precondition:	request is mocked, client is created
Input:	GlucoseModels
Output:	whether the body of the request is correct, the query URL matches the expected URL, the access token is used and the request has only been made once

3.4.10 Sending insulin activities

GB - 36	postSingleInsulinActivity()
Tests whether an InsulinModel is correctly posted to GameBus	
Test items:	`test/gb/insulin.test.ts`
Precondition:	request is mocked, client is created
Input:	InsulinModel
Output:	whether the body of the request is correct, the query URL matches the expected URL, the access token is used and the request has only been made once

GB - 37	postMultipleInsulinActivities()
Tests whether multiple InsulinModels are correctly posted to GameBus	
Test items:	`test/gb/insulin.test.ts`
Precondition:	request is mocked, client is created
Input:	InsulinModels
Output:	whether the body of the request is correct, the query URL matches the expected URL, the access token is used and the request has only been made once

GB - 38	putSingleInsulinActivity()
Tests whether an InsulinModel is correctly replaced in GameBus	
Test items:	`test/gb/insulin.test.ts`
Precondition:	request is mocked, client is created
Input:	InsulinModel
Output:	whether the query URL matches the expected URL, the access token is used and the request has only been made once

3.4.11 Sending food activities

GB - 39	postSingleFoodActivity()
Tests whether a FoodModel is correctly posted to GameBus	
Test items:	`test/gb/food.test.ts`
Precondition:	request is mocked, client is created
Input:	FoodModel
Output:	whether the body of the request is correct, the query URL matches the expected URL, the access token is used and the request has only been made once

GB - 40	postMultipleFoodActivities()
Tests whether multiple FoodModels are correctly posted to GameBus	
Test items:	`test/gb/food.test.ts`
Precondition:	request is mocked, client is created
Input:	FoodModels
Output:	whether the body of the request is correct, the query URL matches the expected URL, the access token is used and the request has only been made once

3.4.12 Sending mood activities

GB - 41	postSingleMoodActivity()
Tests whether a MoodModel is correctly posted to GameBus	
Test items:	`test/gb/mood.test.ts`
Precondition:	request is mocked, client is created
Input:	MoodModel
Output:	whether the body of the request is correct, the query URL matches the expected URL, the access token is used and the request has only been made once

GB - 42	postMultipleMoodActivities()
Tests whether multiple MoodModels are correctly posted to GameBus	
Test items:	`test/gb/mood.test.ts`
Precondition:	request is mocked, client is created
Input:	MoodModels
Output:	whether the body of the request is correct, the query URL matches the expected URL, the access token is used and the request has only been made once

GB - 43	putSingleMoodActivity()
Tests whether a MoodModel is correctly replaced in GameBus	
Test items:	`test/gb/mood.test.ts`
Precondition:	request is mocked, client is created
Input:	MoodModel
Output:	whether the query URL matches the expected URL, the access token is used and the request has only been made once

GB - 44	putSingleMoodActivity()
Tests if the correct error message is shown when a MoodModel is being replaced but no activity Id is specified	
Test items:	`test/gb/mood.test.ts`
Precondition:	request is mocked, client is created
Input:	MoodModel without activity Id
Output:	whether the correct error message is thrown

3.4.13 Sending BMI activities

GB - 45	postSingleBMIActivity()
Tests whether a BMIModel is correctly posted to GameBus	
Test items:	`test/gb/bmi.test.ts`
Precondition:	request is mocked, client is created
Input:	BMIModel
Output:	whether the body of the request is correct, the query URL matches the expected URL, the access token is used and the request has only been made once

3.4.14 GameBus user information

GB - 46	getCurrentUser()
Tests if GameBus user information can be correctly retrieved	
Test items:	`test/gb/user.test.ts`
Precondition:	request and response are mocked, client is created
Input:	-
Output:	whether the query URL matches the expected URL, the access token is used, the request has only been made once and the mocked response is returned as expected

GB - 47	disconnectDataProvider()
Tests if a user can be disconnected from a GameBus data provider	
Test items:	`test/gb/user.test.ts`
Precondition:	request and response are mocked, client is created
Input:	-
Output:	whether the query URL matches the expected URL, the access token is used and the request has only been made once

GB - 48	connectDataProvider()
Tests if a user can be connected to a GameBus data provider	
Test items:	`test/gb/user.test.ts`
Precondition:	request and response are mocked, client is created
Input:	-
Output:	whether the query URL matches the expected URL, the access token is used and the request has only been made once

3.5 GameBus challenges and circles

GameBus offers functionality to create challenges and join circles with others. We also use and test these functionalities through mocked requests.

3.5.1 Creating GameBus challenges

CLCR - 1	postChallenge()
Tests whether a challenge post request is correctly created	
Test items:	`test/gb/challenge.test.ts`
Precondition:	request is mocked, client is created
Input:	Challenge data
Output:	whether the body of the request is correct, the query URL matches the expected URL, the access token is used and the request has only been made once

3.5.2 Adding circles to a challenge

CLCR - 2	postCircleMembership()
Tests whether a circle is correctly added to a challenge	
Test items:	`test/gb/challenge.test.ts`
Precondition:	request is mocked, client is created
Input:	circleIds, challengeIds
Output:	whether the body of the request is correct, the query URL matches the expected URL, the access token is used and the request has only been made once

3.5.3 Get circles for a player

CLCR - 3	getAllCircles()
Tests whether all circle IDs of the circles the player belongs to are correctly requested	
Test items:	`test/gb/circle.test.ts`
Precondition:	request is mocked, client is created
Input:	playerId
Output:	whether the body of the request is correct, the query URL matches the expected URL, the access token is used and the request has only been made once

CLCR - 4	getCircleById()
Tests whether one circle of the circles the player belongs to is correctly requested	
Test items:	`test/gb/circle.test.ts`
Precondition:	request is mocked, client is created
Input:	circleId, playerId
Output:	whether the body of the request is correct, the query URL matches the expected URL, the access token is used and the request has only been made once

CLCR - 5	getAllCirclesLeaderDiabetter()
Tests whether all Diabetter circles where the user is a supervisor are correctly requested	
Test items:	`test/gb/circle.test.ts`
Precondition:	request is mocked, client is created
Input:	playerId
Output:	whether the body of the request is correct, the query URL matches the expected URL, the access token is used and the request has only been made once

3.5.4 Get players in a circle

CLCR - 6	getPlayersForAGivenCircle()
Tests whether the players within a circle are correctly requested	
Test items:	`test/gb/circle.test.ts`
Precondition:	request is mocked, client is created
Input:	circleId
Output:	whether the body of the request is correct, the query URL matches the expected URL, the access token is used and the request has only been made once

3.6 GameBus client

3.6.1 Full response of GameBus request

GBC - 1	GameBusClient.request()
Tests if a request can be made and the response has the correct format	
Test items:	`test/gb/gbClient.test.ts`
Precondition:	request is mocked, client is created
Input:	-
Output:	whether an empty data array is returned within the response

3.6.2 Put request

GBC - 2	GameBusClient.put()
Tests if a put request can be made in the correct way	
Test items:	`test/gb/gbClient.test.ts`
Precondition:	request is mocked, client is created
Input:	-
Output:	whether the query URL matches the expected URL and has only been made once

3.6.3 Post request

GBC - 3	GameBusClient.post()
Tests if a post request can be made in the correct way	
Test items:	`test/gb/gbClient.test.ts`
Precondition:	request is mocked, client is created
Input:	-
Output:	whether the query URL matches the expected URL and has only been made once

3.6.4 Get request

GBC - 4	GameBusClient.get()
Tests if a get request can be made in the correct way	
Test items:	`test/gb/gbClient.test.ts`
Precondition:	request is mocked, client is created
Input:	-
Output:	whether the query URL matches the expected URL and has only been made once.

3.6.5 Sending unauthorized requests

GBC - 5	GameBusClient.request()
Tests if sending requests with a client that is not initiated with a token is rejected by throwing an error	
Test items:	`test/gb/gbClient.test.ts`
Precondition:	request is mocked, client is created
Input:	client without token
Output:	whether an error is thrown if a request is made

GBC - 6	GameBusClient.request()
Tests if sending requests with a client that is initiated with an empty token is rejected by throwing an error	
Test items:	`test/gb/gbClient.test.ts`
Precondition:	request is mocked, client is created
Input:	client with empty token
Output:	whether an error is thrown if a request is made

3.6.6 GameBus client helper methods

GBC - 7	GameBusClient.createHeader()
Tests if the correct error is thrown, when the function is called without proper authorization in place	
Test items:	`test/gb/gbClient.test.ts`
Precondition:	client is created
Input:	-
Output:	whether the correct error is thrown

GBC - 8	GameBusClient.createURL()
Tests if a URL without included queries is correctly created	
Test items:	`test/gb/gbClient.test.ts`
Precondition:	client is created
Input:	-
Output:	whether the created URL matches the expected URL

GBC - 9	GameBusClient.createURL()
Tests if a URL with query parameters is correctly created	
Test items:	`test/gb/gbClient.test.ts`
Precondition:	client is created
Input:	-
Output:	whether the created URL matches the expected URL and includes the query

3.7 Date conversions

3.7.1 String to date format

DATE - 1	getDateFormat()
Tests whether the correct date format is returned from a string	
Test items:	`test/services/utils/dates.test.ts`
Precondition:	correct date format(s) is (are) defined
Input:	date as string
Output:	whether the guessed date format matches the expected date format

3.7.2 String to date

DATE - 2	parseDate()
Tests whether the string correctly parses to the date	
Test items:	`test/services/utils/dates.test.ts`
Precondition:	correct date format is used
Input:	date as string, date format, unix (yes/no)
Output:	whether the string parsed as given date format is equal to the expected date

3.7.3 Unix timestamp to date

DATE - 3	fromUnixMsTime()
Tests whether provided unix timestamp correctly converts to a date	
Test items:	`test/services/utils/dates.test.ts`
Precondition:	correct timestamp format is provided
Input:	unix timestamp (in milliseconds)
Output:	whether the unix timestamp is equal to the expected date

3.7.4 Parsing excel raw date format

DATE - 4	parseExcelDate()
Tests if the raw Excel date format is correctly parsed to a readable format for the first day of January 2008	
Test items:	`test/services/utils/dates.test.ts`
Precondition:	-
Input:	days since 1900 (Excel date format)
Output:	whether the Excel date format is correctly converted

DATE - 5	parseExcelDate()
Tests if the raw Excel date format is correctly parsed to a readable format for the ninth of May 2021	
Test items:	`test/services/utils/dates.test.ts`
Precondition:	-
Input:	days since 1900 (Excel date format)
Output:	whether the Excel date format is correctly converted

3.7.5 Parsing excel raw time format

Due to the inaccurate nature of floats, we have added quite a few tests that seem to test the same functionality. These are to ensure that this inaccurate nature is correctly dealt with, however.

DATE - 6	parseExcelTime()
Tests if the raw Excel time format is correctly parsed to a readable format for 12:00	
Test items:	`test/services/utils/dates.test.ts`
Precondition:	-
Input:	fraction of the day (raw Excel time format)
Output:	whether the Excel time format is correctly converted

DATE - 7	parseExcelTime()
Tests if the raw Excel time format is correctly parsed to a readable format for 20:43	
Test items:	`test/services/utils/dates.test.ts`
Precondition:	-
Input:	fraction of the day (raw Excel time format)
Output:	whether the Excel time format is correctly converted

DATE - 8	parseExcelTime()
Tests if the raw Excel time format is correctly parsed to a readable format for 08:00	
Test items:	`test/services/utils/dates.test.ts`
Precondition:	-
Input:	fraction of the day (raw Excel time format)
Output:	whether the Excel time format is correctly converted

DATE - 9	parseExcelTime()
Tests if the raw Excel time format is correctly parsed to a readable format for 00:00	
Test items:	`test/services/utils/dates.test.ts`
Precondition:	-
Input:	fraction of the day (raw Excel time format)
Output:	whether the Excel time format is correctly converted

DATE - 10	parseExcelTime()
Tests if the raw Excel time format is correctly parsed to a readable format for 10:01	
Test items:	`test/services/utils/dates.test.ts`
Precondition:	-
Input:	fraction of the day (raw Excel time format)
Output:	whether the Excel time format is correctly converted

3.7.6 Parsing excel date and time format - Robustness

DATE - 11	parseExcelTime()
Tests if an error is thrown when a negative day fraction (Excel time format) is given as input	
Test items:	`test/services/utils/dates.test.ts`
Precondition:	-
Input:	negative day fraction (raw Excel time format)
Output:	whether the correct error is thrown

DATE - 12	parseExcelDate()
Tests if an error is thrown when a negative number of days since 1900 (Excel date format) is given as input	
Test items:	`test/services/utils/dates.test.ts`
Precondition:	-
Input:	negative amount of days since 1900 (raw Excel date format)
Output:	whether the correct error is thrown

3.7.7 Testing validity of Unix timestamps

DATE - 13	validUnixTimestamp()
Tests if a valid Unix timestamp is identified as such by the function and true is returned	
Test items:	`test/services/utils/dates.test.ts`
Precondition:	-
Input:	a valid Unix timestamp
Output:	whether the function returns true

DATE - 14	validUnixTimestamp()
Tests if an invalid Unix timestamp is identified as such by the function and an error is thrown	
Test items:	`test/services/utils/dates.test.ts`
Precondition:	-
Input:	an invalid Unix timestamp
Output:	whether the function throws the correct error

3.7.8 Converting excel date and time within objects

DATE - 15	convertExcelDateTimes()
Tests if an array of objects with Excel date or time fields is converted to an array of objects with time in readable format	
Test items:	`test/services/utils/dates.test.ts`
Precondition:	-
Input:	array with object that has excel date and time fields
Output:	whether the Excel date and time formats are correctly converted and returned in an array

DATE - 16	convertExcelDateTimes()
Tests if an array of objects without Excel date or time fields is returned as it is	
Test items:	`test/services/utils/dates.test.ts`
Precondition:	-
Input:	array with object that does not have date and time fields
Output:	whether the array is returned as it is without errors

3.8 Unit conversions

3.8.1 mg/dL to mmol/L

UNITS - 1	convertMG_DLtoMMOL_L()
Tests whether the conversion from mg/dL to mmol/L is correct	
Test items:	`test/services/utils/units.test.ts`
Precondition:	-
Input:	amount in mg/dL
Output:	whether the amount is the same as the expected amount in mmol/L

3.8.2 mmol/L to mg/dL

UNITS - 2	convertMMOL_LtoMG_dL()
Tests whether the conversion from mmol/L to mg/dL is correct	
Test items:	`test/services/utils/units.test.ts`
Precondition:	-
Input:	amount in mmol/L
Output:	whether the amount is the same as the expected amount in mg/dL

3.9 File paths

3.9.1 Get extension from file path

FILE - 1	getFileExtension()
Tests whether the file extension is correctly retrieved from a file path	
Test items:	`test/services/utils/files.test.ts`
Precondition:	-
Input:	File path
Output:	Whether the file extension is as expected

FILE - 2	getFileExtension()
Tests if the file extension function is robust and can handle basic erroneous inputs	
Test items:	`test/services/utils/files.test.ts`
Precondition:	-
Input:	file paths with edge cases (early '.', no extension)
Output:	whether the correct results are returned after inputting the wrong file paths

FILE - 3	getFileExtension()
Tests if the file extension function can also process file paths with backslashes	
Test items:	`test/services/utils/files.test.ts`
Precondition:	-
Input:	file paths with backslashes
Output:	whether the correct file extensions are returned

3.9.2 Get directory from file path

FILE - 4	getFileDirectory()
Tests whether the file directory is correctly retrieved from a file path	
Test items:	`test/services/utils/files.test.ts`
Precondition:	-
Input:	file path
Output:	whether file directory in as expected

3.9.3 Get name from file path

FILE - 5	getFileName()
Tests whether the file name is correctly retrieved from a file path	
Test items:	`test/services/utils/files.test.ts`
Precondition:	-
Input:	file path
Output:	whether file name in as expected

3.10 Data parser general

DAP - 1	DataParser.parse()
Tests whether the correct error is thrown when no file path is provided	
Test items:	`test/services/dataParser.test.ts`
Precondition:	-
Input:	-
Output:	whether the correct error is thrown

DAP - 2	DataParser.parse()
Tests whether the correct error is thrown when no OneDrive token is given	
Test items:	`test/services/dataParser.test.ts`
Precondition:	-
Input:	-
Output:	whether the correct error is thrown

DAP - 3	DataParser.setFilePath()
Tests whether the file path can be set at a later stage	
Test items:	`test/services/dataParser.test.ts`
Precondition:	-
Input:	new file path
Output:	whether the new file path is set correctly

DAP - 4	DataParser.getData()
Tests whether all data from a parser can be retrieved even if null	
Test items:	`test/services/dataParser.test.ts`
Precondition:	-
Input:	-
Output:	whether the correct data type with no data is returned

3.11 Parsing .csv file

3.11.1 Reading .csv file

CSV - 1	CSVParser.parse()
Tests whether reading a .csv file gives the expected result	
Test items:	`test/services/csv.test.ts`, `test/services/data/test.csv`
Precondition:	test.csv file is present in `test/services/data/test.csv`
Input:	file path
Output:	whether the .csv file is read as expected

3.11.2 Robustness of Abbott parser

CSV - 2	AbbottParser.process()
Tests whether providing wrong input data throws an error: 'Wrong input data for processing Abbott data!'	
Test items:	`test/services/Abbott.test.ts`
Precondition:	-
Input:	wrong file path
Output:	whether the correct error was thrown

3.12 Parsing .xml file

3.12.1 Reading .xml file

XML - 1	XMLParser.parse()
Tests whether reading a .xml file gives the expected result	
Test items:	`test/services/xml.test.ts`, `test/services/data/test.xml`
Precondition:	test.xml file is present in `test/services/data/test.xml`
Input:	file path
Output:	whether the .xml file is read as expected

3.13 Parsing .xlsx file

3.13.1 Parsing a Excel files

XLSX - 1	ExcelParser.parse()
Tests if an Excel file can be correctly parsed and converted to an array of objects	
Test items:	`test/services/xlsx.test.ts`
Precondition:	testExcel.xlsx file is present in `test/services/data/testExcel.xlsx`
Input:	`test/services/data/testExcel.xlsx`
Output:	whether the Excel file is correctly parsed into an object with string values

XLSX - 2	ExcelParser.parse()
Tests if an Excel file can be correctly parsed and converted to an array of objects with raw (non-string) values. This is mainly important for date fields	
Test items:	`test/services/xlsx.test.ts`
Precondition:	testExcel.xlsx file is present in `test/services/data/testExcel.xlsx`
Input:	`test/services/data/testExcel.xlsx`
Output:	whether the Excel file is correctly parsed into an object with string or number values

3.14 Food diary importing

3.14.1 Robustness of food diary parser

FD - 1	FoodDiaryParser.process()
Tests whether providing wrong input data throws an error: 'Wrong input data for processing food diary data!'	
Test items:	`test/services/fooddiary.test.ts`
Precondition:	-
Input:	wrong file path
Output:	whether the correct error was thrown

3.14.2 Automatic date and total insulin fill if missing

FD - 2	FoodDiaryParser.process(), DiaryParser.fillDate(), Parser.computeTotalInsulin()	Food-FoodDiary-

Tests if missing date and total insulin fields are filled in correctly if they are missing and can be filled in within the context

Test items:	`test/services/fooddiary.test.ts`
Precondition:	-
Input:	mocked food diary data after it has been parsed from the file and before preprocessing
Output:	whether the missing fields are filled in correctly

3.14.3 Automatic date, total insulin an time fill if missing

FD - 3	FoodDiaryParser.process(), DiaryParser.fillDate(), Parser.computeTotalInsulin(), Parser.fillTime()	Food-FoodDiary-FoodDiary-

Tests if missing date, time and total insulin fields are filled in correctly if they are missing and can be filled in within the context

Test items:	`test/services/fooddiary.test.ts`
Precondition:	-
Input:	mocked food diary data after it has been parsed from the file and before preprocessing
Output:	whether the missing fields are filled in correctly

3.14.4 Missing first date

FD - 4	FoodDiaryParser.process(), FoodDiaryParser.fillDate()
Tests if the correct error is thrown if the first date within the food diary excel file is not filled in	
Test items:	`test/services/fooddiary.test.ts`
Precondition:	-
Input:	mocked food diary data after it has been parsed from the file and before preprocessing
Output:	whether the correct errors are thrown

3.15 Food importing

3.15.1 Importing Abbott EU data

FOOD - 1	AbbottParser.parse(), FoodParser.process()
Tests whether reading a European Abbott .csv file gives the correct food data	
Test items:	`test/services/food.test.ts`, `test/services/data/abbott_eu.csv`
Precondition:	abbott_eu.csv file is present in `test/services/data/abbott_eu.csv`
Input:	file path, data type to return
Output:	whether the .csv food data is read as expected

3.15.2 Importing Abbott US data

FOOD - 2	AbbottParser.parse(), FoodParser.process()
Tests whether reading an American Abbott .csv file gives the correct food data	
Test items:	`test/services/food.test.ts`, `test/services/data/abbott_us.csv`
Precondition:	abbott_us.csv file is present in `test/services/data/abbott_us.csv`
Input:	file path, data type to return
Output:	whether the .csv food data is read as expected

3.15.3 Importing Eetmeter data

FOOD - 3	EetMeterParser.parse(), FoodParser.process()
Tests whether reading a single Eetmeter entry gives the correct food data	
Test items:	`test/services/food.test.ts`, `test/services/data/eetmeter.xml`
Precondition:	eetmeter.xml file is present in `test/services/data/eetmeter.xml`
Input:	file path
Output:	whether the .xml food data is read as expected

FOOD - 4	EetMeterParser.parse(), FoodParser.process()
Tests whether reading multiple Eetmeter entries gives the correct food data	
Test items:	`test/services/food.test.ts`, `test/services/data/eetmeterMany.xml`
Precondition:	eetmeterMany.xml file is present in `test/services/data/eetmeterMany.xml`
Input:	file path
Output:	whether the .xml food data is read as expected

3.15.4 Importing standardized food diary data without missing values

FOOD - 5	FoodDiaryParser.parse(), FoodParser.process()
Tests whether reading an .xlsx standard food diary file without missing cells gives the correct food data	
Test items:	`test/services/food.test.ts`, `test/services/data/foodDiary_standard.xlsx`
Precondition:	foodDiary_standard.xlsx file is present in `test/services/data/foodDiary_standard.xlsx`
Input:	file path, data type to return
Output:	whether the .xlsx food data is read as expected

3.15.5 Importing standardized food diary data with missing values

FOOD - 6	FoodDiaryParser.parse(), FoodDiaryParser.preprocess(), FoodParser.process()
Tests whether reading an .xlsx standard food diary file with missing cells gives the correct food data and the missing date and time are correctly filled in	
Test items:	`test/services/food.test.ts`, `test/services/data/foodDiary_standard_missing.xlsx`
Precondition:	foodDiary_standard_missing.xlsx file is present in `test/services/data/foodDiary_standard_missing.xlsx`
Input:	file path, data type to return
Output:	whether the .xlsx food data is read as expected

3.15.6 Importing mocked Nightscout response with carbohydrates

FOOD - 7	NightscoutParser.parseTreatment(), FoodParser.process()
Tests whether parsing a mocked response from Nightscout containing a treatment with carbohydrates gives the correct food data	
Test items:	`test/services/food.test.ts`
Precondition:	-
Input:	mocked nightScout treatment + empty entry array + data type to return
Output:	whether the Nightscout food data is read as expected

3.15.7 Importing mocked Nightscout response with food data

FOOD - 8	NightscoutParser.parseTreatment(), FoodParser.process()
Tests whether parsing a mocked response from Nightscout containing a treatment with several food properties (carbs, fat and protein) gives the correct food data	
Test items:	`test/services/food.test.ts`
Precondition:	-
Input:	mocked nightScout treatment + empty entry array + data type to return
Output:	whether the Nightscout food data is read as expected

3.15.8 Food mapper - robustness

FOOD - 9	FoodMapper.mapFood()
Tests if trying to map unsupported food sources results in an error	
Test items:	`test/services/food.test.ts`
Precondition:	-
Input:	unsupported food source
Output:	whether the expected error is thrown

3.16 Glucose importing

3.16.1 Importing Abbott EU data

GLU - 1	AbbottParser.parse(), GlucoseParser.process()
Tests whether reading a European Abbott .csv file gives the correct glucose data	
Test items:	`test/services/glucose.test.ts`, `test/services/data/abbott_eu.csv`
Precondition:	abbott_eu.csv file is present in `test/services/data/abbott_eu.csv`
Input:	file path, data type to return
Output:	whether the .csv glucose data is read as expected

3.16.2 Importing Abbott US data

GLU - 2	AbbottParser.parse(), GlucoseParser.process()
Tests whether reading an American Abbott .csv file gives the correct glucose data	
Test items:	`test/services/glucose.test.ts`, `test/services/data/abbott_us.csv`
Precondition:	abbott_us.csv file is present in `test/services/data/abbott_us.csv`
Input:	file path, data type to return
Output:	whether the .csv glucose data is read as expected

3.16.3 Importing mocked Nightscout response with glucose

GLU - 3	NightscoutParser.parseEntry(), GlucoseParser.process()
Tests whether parsing a mocked response from Nightscout containing an entry with glucose gives the correct glucose data.	
Test items:	`test/services/glucose.test.ts`
Precondition:	-
Input:	mocked nightScout entry + empty treatment array + data type to return
Output:	whether the Nightscout glucose data is read as expected

3.16.4 Glucose mapper - Robustness

GLU - 4	GlucoseMapper.mapGlucose()
Tests if mapping an unsupported glucose source results in an error	
Test items:	`test/services/glucose.test.ts`
Precondition:	-
Input:	unsupported glucose source
Output:	whether the expected error is thrown

3.16.5 Empty glucose model

GLU - 5	emptyGlucoseModel()
Tests if the helper function correctly returns an empty glucose model	
Test items:	`test/services/glucose.test.ts`
Precondition:	-
Input:	-
Output:	whether the expected empty glucose model is returned

3.17 Insulin importing

3.17.1 Importing Abbott EU data

INS - 1	AbbottParser.parse(), InsulinParser.process()
Tests if reading a European Abbott .csv file gives the correct insulin data	
Test items:	`test/services/insulin.test.ts`, `test/services/data/abbott_eu.csv`
Precondition:	abbott_eu.csv file is present in `test/services/data/abbott_eu.csv`
Input:	file path, data type to return
Output:	whether the .csv insulin data is read as expected

3.17.2 Importing Abbott US data

INS - 2	AbbottParser.parse(), InsulinParser.process()
Tests if reading an American Abbott .csv file gives the correct insulin data	
Test items:	`test/services/insulin.test.ts`, `test/services/data/abbott_us.csv`
Precondition:	abbott_us.csv file is present in `test/services/data/abbott_us.csv`
Input:	file path, data type to return
Output:	whether the .csv insulin data is read as expected

3.17.3 Importing standardized food diary data without missing values

INS - 3	FoodDiaryParser.parse(), InsulinParser.process()
Tests if reading an .xlsx standard food diary file without missing cells gives the correct insulin data	
Test items:	`test/services/insulin.test.ts`, `test/services/data/foodDiary_standard.xlsx`
Precondition:	foodDiary_standard.xlsx file is present in `test/services/data/foodDiary_standard.xlsx`
Input:	file path, data type to return
Output:	whether the .xlsx insulin data is read as expected

3.17.4 Importing standardized food diary data with missing values

INS - 4	FoodDiaryParser.parse(), FoodDiaryParser.preprocess(), InsulinParser.process()
Tests if reading an .xlsx standard food diary file with missing cells gives the correct insulin data and the missing date, time and/or insulin field are filled in correctly	
Test items:	`test/services/insulin.test.ts`, `test/services/data/foodDiary_standard_missing.xlsx`
Precondition:	foodDiary_standard_missing.xlsx file is present in `test/services/data/foodDiary_standard_missing.xlsx`
Input:	file path, data type to return
Output:	whether the .xlsx insulin data is read as expected

3.17.5 Importing mocked Nightscout response with insulin

INS - 5	NightscoutParser.parseTreatment(), InsulinParser.process()
Tests if parsing a mocked response from Nightscout containing a treatment with insulin gives the correct insulin data.	
Test items:	`test/services/insulin.test.ts`
Precondition:	-
Input:	mocked nightScout treatment + empty entry array + data type to return
Output:	whether the Nightscout insulin data is read as expected

3.17.6 Insulin mapper - Robustness

INS - 6	InsulinMapper.mapInsulin()
Tests if mapping an unsupported insulin source results in an error	
Test items:	`test/services/insulin.test.ts`
Precondition:	-
Input:	unsupported insulin source
Output:	whether the expected error is thrown

3.18 Food exporting

FEX - 1	FoodParser.post(), FoodParser.process()
Tests if food diary data can be processed to a food model and posted to GameBus	
Test items:	`test/services/food.test.ts`
Precondition:	-
Input:	mocked foodDiary data
Output:	whether the food diary data is processed and posted to GameBus as expected

3.19 Glucose exporting

GEX - 1	GlucoseParser.post(), GlucoseParser.process()
Tests if Abbott data can be processed to a glucose model and posted to GameBus	
Test items:	`test/services/glucose.test.ts`
Precondition:	-
Input:	mocked AbbottData
Output:	whether the AbbottData is processed and posted to GameBus as expected

3.20 Insulin exporting

IEX - 1	InsulinParser.post(), InsulinParser.process()
Tests if food diary data can be processed to an insulin model and posted to GameBus	
Test items:	`test/services/insulin.test.ts`
Precondition:	-
Input:	mocked FoodDiaryData
Output:	whether the FoodDiaryData is processed and posted to GameBus as expected

3.21 Mood exporting and importing

MEX - 1	MoodParser.post(), MoodParser.process()
Tests if a mood model can be posted to GameBus	
Test items:	`test/services/mood.test.ts`
Precondition:	-
Input:	mocked mood model
Output:	whether the mood model is posted to GameBus as expected

MEX - 2	MoodParser.process()
Tests if a mood model parser template is functioning correctly (note that it is not implemented yet as we do not have mood data from other sources)	
Test items:	`test/services/mood.test.ts`
Precondition:	-
Input:	mocked mood model array
Output:	whether the mood input is processed as expected

3.22 Export only new data from input files

When the user uses a food diary multiple times or creates an export at multiple occasions, only the newly added data in these files should be exported from the Diabetter system to GameBus. These tests ensure that this process goes as expected. Before all of these tests, an empty database is created on a test location and after all tests it is deleted. Before each test, the file parse events table is emptied as well.

3.22.1 Parsing same Abbott export file should return nothing

EON - 1	AbbottParser.parse()
Tests if glucose data is correctly retrieved the first time, but not retrieved the second time since it is not new	
Test items:	`test/services/updateNewest.test.ts`
Precondition:	database location is specified, `test/services/data/abbott_eu.csv` file is present
Input:	`test/services/data/abbott_eu.csv`
Output:	whether expected glucose data and an empty list are returned

3.22.2 Parsing same food diary should return nothing

EON - 2	FoodDiaryParser.parse()
Tests if food data is correctly retrieved the first time, but not retrieved the second time since it is not new	
Test items:	`test/services/updateNewest.test.ts`
Precondition:	database location is specified, test/services/data/foodDiary_standard_missing_table.xlsx file is present
Input:	test/services/data/foodDiary_standard_missing_table.xlsx
Output:	whether expected food data and nothing are returned

3.22.3 Parsing same Eetmeter export should return nothing

EON - 3	EetmeterParser.parse()
Tests if food data is correctly retrieved the first time, but not retrieved the second time since it is not new	
Test items:	`test/services/updateNewest.test.ts`
Precondition:	database location is specified, `test/services/data/eetmeterMany.xml` file is present
Input:	`test/services/data/eetmeterMany.xml`
Output:	whether expected food data and nothing are returned

3.22.4 Update newest on the food and insulin parser

EON - 4	FoodParser.process(), InsulinParser.process()
Tests whether raw foodDiaryData is properly filtered on the timestamp of last update, leaving only newer entries in	
Test items:	`test/services/updateNewest.test.ts`
Precondition:	-
Input:	dummy food diary data with 2 entries, dummy user info, timestamp between the two entries
Output:	Whether only the entry after the timestamp has been returned

3.22.5 Update newest on the glucose parser

EON - 5	GlucoseParser.process()
Tests if raw foodDiaryData is properly filtered on the timestamp of last update, leaving only newer entries in	
Test items:	`test/services/updateNewest.test.ts`
Precondition:	-
Input:	dummy Abbott data with 2 entries, dummy user info, timestamp between the two entries
Output:	whether only the entry after the timestamp has been returned

3.22.6 Update newest on a ModelParser but no last update timestamp has been set

EON - 6	FoodParser.process()
Tests if all data is returned if no last update timestamp has been set, whereas another parameter states it should filter on newest updates. The result should be that all entries are returned	
Test items:	`test/services/updateNewest.test.ts`
Precondition:	-
Input:	dummy FoodDiary data with 1 entry, dummy user info
Output:	whether all data has been returned

3.22.7 Update newest on a ModelParser but last update timestamp has been set to 0

EON - 7	FoodParser.process()
Tests if all data is returned if the last update timestamp has been set to 0, whereas another parameter states it should filter on newest updates. The result should be that all entries are returned.	
Test items:	`test/services/updateNewest.test.ts`
Precondition:	-
Input:	dummy FoodDiary data with 1 entry, dummy user info
Output:	whether all data has been returned

3.23 OneDrive API

As with the GameBus API tests, all requests are mocked.

3.23.1 Reading table values

ONED - 1	OneDriveClient.getTableValues()
Tests whether a table from OneDrive can be read	
Test items:	`test/onedrive/odClient.test.ts`
Precondition:	request is mocked
Input:	-
Output:	whether the table is read as expected

3.23.2 Trying to read non-existing table values

ONED - 2	OneDriveClient.getTableValues()
Tests if trying to read a non-existing table returns an empty array	
Test items:	`test/onedrive/odClient.test.ts`
Precondition:	request is mocked
Input:	-
Output:	whether an empty array is returned

3.23.3 Reading range values

ONED - 3	OneDriveClient.getRangeValues()
Tests whether the selected range can be read from OneDrive	
Test items:	`test/onedrive/odClient.test.ts`
Precondition:	request is mocked
Input:	range of cells to read (top left, bottom right)
Output:	whether the table range is read as expected

3.23.4 Reading range values as text

ONED - 4	OneDriveClient.getRangeText()
Tests whether the selected range can be read from OneDrive as text	
Test items:	`test/onedrive/odClient.test.ts`
Precondition:	request is mocked
Input:	range of cells to read (top left, bottom right)
Output:	whether the table range is read as expected

3.23.5 Reading table lists

ONED - 5	OneDriveClient.getTableList()
Tests the lists of tables in OneDrive is read as expected	
Test items:	`test/onedrive/odClient.test.ts`
Precondition:	request is mocked
Input:	-
Output:	whether the table list is read as expected

3.23.6 Getting file at root directory

ONED - 6	OneDriveClient.getFile()
Tests if the file is retrieved given the input	
Test items:	`test/onedrive/odClient.test.ts`
Precondition:	request is mocked
Input:	client token, name of file, path of file
Output:	info about file (if it exists)

3.23.7 Debug boolean printing True False

ONED - 7	OneDriveClient.doPrint(), OneDriveClient.printDeep()
Checks whether the debugging tools of the OneDriveClient are called as intended, in this case when debug mode is on (true) and deep printing is off (false)	
Test items:	`test/onedrive/odClient.test.ts`
Precondition:	-
Input:	booleans true and false for indicating debug mode without deep printing
Output:	whether the debugging tools print as intended

3.23.8 Debug boolean printing False True

ONED - 8	OneDriveClient.doPrint(), OneDriveClient.printDeep()
Checks whether the debugging tools of the OneDriveClient are called as intended, in this case when debug mode is off (false) and deep printing is on (true)	
Test items:	`test/onedrive/odClient.test.ts`
Precondition:	-
Input:	booleans false and true for indicating no debug mode with deep printing
Output:	whether the debugging tools print as intended

3.23.9 Debug boolean printing True True

ONED - 9	OneDriveClient.doPrint(), OneDriveClient.printDeep()
Checks whether the debugging tools of the OneDriveClient are called as intended, in this case when debug mode is on (true) and deep printing is on (true)	
Test items:	`test/onedrive/odClient.test.ts`
Precondition:	-
Input:	booleans true and true for indicating debug mode with deep printing
Output:	whether the debugging tools print as intended

3.23.10 Onedrive .xslx parsing - import standardized food diary with missing values from a onedrive

ONED - 10	OneDriveExcelParser.parse(), OneDriveExcelParser.assignKeys(), utils/date: parseExcelDate(), parseExcelTime()
Uses mocked 2D array .xlsx response data from the OneDrive client and checks if this is correctly parsed and keys are assigned	
Test items:	`test/onedrive/oneDriveExcelParser.test.ts`
Precondition:	-
Input:	dummy file path, mocked response data
Output:	whether the correct keys are assigned to the instances and the dates and times are correctly parsed

3.23.11 Assigning keys to raw OneDrive data

ONED - 11	OneDriveExcelParser.assignKeys()
Tests if the correct keys are assigned to the raw OneDrive data (2D array) and an array of objects is returned	
Test items:	`test/onedrive/oneDriveExcelParser.test.ts`
Precondition:	-
Input:	mocked raw OneDrive food diary data (2D array), food diary keys
Output:	whether the correct keys are assigned to the items of the data array

3.23.12 Assigning wrong keys to raw OneDrive data

ONED - 12	OneDriveExcelParser.assignKeys()
Tests if assigning non-compatible keys to the raw OneDrive data (2D array) results in an error	
Test items:	`test/onedrive/oneDriveExcelParser.test.ts`
Precondition:	-
Input:	mocked raw OneDrive food diary data (2D array), Abbott keys
Output:	whether the expected error is thrown

3.23.13 Assigning no keys to raw OneDrive data

ONED - 13	OneDriveExcelParser.assignKeys()
Tests if assigning no/undefined keys to the raw OneDrive data (2D array) results in an error	
Test items:	`test/onedrive/oneDriveExcelParser.test.ts`
Precondition:	-
Input:	mocked raw OneDrive food diary data (2D array), undefined keys
Output:	whether the expected error is thrown

ONED - 14	OneDriveClient
Tests if constructor default values are set correctly	
Test items:	`test/onedrive/odClient.test.ts`
Precondition:	-
Input:	-
Output:	whether the default values of the client are set correctly

3.23.14 Generating redirect URL

ONED - 15	generateRedirectURL()
Tests if OneDrive account information is correctly converted to a callback URL that contains this data	
Test items:	`test/onedrive/odClient.test.ts`
Precondition:	mocked OneDrive token model
Input:	-
Output:	whether the input account information is correctly converted into the URL

3.24 Nightscout API

As with the GameBus API tests, all requests are mocked.

3.24.1 Posting a Nightscout entry

NS - 1	NightscoutClient.postEntry()
Uses a mocked request to test if the Nightscout client can post a glucose entry	
Test items:	`test/Nightscout/nsClient.test.ts`
Precondition:	request is mocked
Input:	mocked input entry
Output:	whether the entry is posted as expected

3.24.2 Posting a Nightscout treatment

NS - 2	NightscoutClient.postTreatment()
Uses a mocked request to test if the Nightscout client can post a treatment	
Test items:	`test/Nightscout/nsClient.test.ts`
Precondition:	request is mocked
Input:	mocked input treatment
Output:	whether the treatment is posted as expected

3.24.3 Getting Nightscout entries

NS - 3	NightscoutClient.getEntries()
Uses a mocked request to test if the Nightscout client can get entries from a Nightscout host	
Test items:	`test/Nightscout/nsClient.test.ts`
Precondition:	request is mocked
Input:	-
Output:	whether the entries are requested as expected

3.24.4 Getting Nightscout treatments

NS - 4	NightscoutClient.getTreatments()
Uses a mocked request to test if the Nightscout client can get treatments from a Nightscout host	
Test items:	`test/Nightscout/nsClient.test.ts`
Precondition:	request is mocked
Input:	-
Output:	whether the treatments are requested as expected

3.24.5 Getting glucose unit

NS - 5	NightscoutClient.getGlucoseUnit()
Uses a mocked request to test if the Nightscout client can get the used glucose unit from a Nightscout host	
Test items:	`test/Nightscout/nsClient.test.ts`
Precondition:	request is mocked
Input:	-
Output:	whether the glucose unit that is used on the Nightscout website is correctly requested

3.25 Files endpoint

To test our own endpoints, we used the `supertest` [7] package which can test HTTP responses. For posting to our endpoint, users need to specify the data format (Abbott, food diary, etc.) and provide a file to upload.

3.25.1 Posting without specified data format

FILEP - 1	Post /upload endpoint
Tests if a file upload that does not specify any file format is rejected with status 400	
Test items:	`test/routes.test.ts`
Precondition:	server is running
Input:	no parameter + random file
Output:	whether status 400 is returned

3.25.2 Posting an unsupported data format

FILEP - 2	Post /upload endpoint
Tests whether the server rejects uploading an unsupported data format by responding with status 400	
Test items:	`test/routes.test.ts`
Precondition:	server is running
Input:	unsupported data format + random file
Output:	whether status 400 is returned

3.25.3 Posting an unsupported file extension

FILEP - 3	Post /upload endpoint
Tests if the server rejects uploading a supported data format, but with in an unsupported file type (recognised by extension) by responding with status 400	
Test items:	`test/routes.test.ts`
Precondition:	server is running
Input:	supported data format + file with unsupported extension
Output:	whether status 400 is returned

3.25.4 Posting file with wrong contents

FILEP - 4	Post /upload endpoint
Tests if uploading with Abbott format and a supported file extension is rejected when the contents of the file do not match the expected contents	
Test items:	`test/routes.test.ts`
Precondition:	server is running
Input:	supported data format 'Abbott' + file with wrong contents
Output:	whether status 400 is returned

FILEP - 5	Post /upload endpoint
Tests if uploading with food diary format and a supported file extension is rejected when the contents of the file do not match the expected contents	
Test items:	`test/routes.test.ts`
Precondition:	server is running
Input:	supported data format 'food diary' + file with wrong contents
Output:	whether status 400 is returned

3.26 Data endpoint

The first couple of tests go over the robustness of the endpoint by testing error messages. Unfortunately, we do not find it appropriate to store actual authorization tokens, because of both privacy issues and the fact that they expire. The rest of the tests go over the underlying functionalities of the data endpoint that can be thoroughly tested. The requests to GameBus are mocked, however, because the earlier mentioned token is missing.

3.26.1 Test error responses

DEP - 1	/data GET endpoint
Tests if sending requests to the data endpoint results in a 401 unauthorized response if no authorization header is specified	
Test items:	`test/routes.test.ts`
Precondition:	server is running
Input:	-
Output:	whether the correct 401 MEP - response is sent back

DEP - 2	/data GET endpoint
Tests if sending requests to the data endpoint without necessary query parameters results in a 400 bad request response code	
Test items:	`test/routes.test.ts`
Precondition:	server is running
Input:	-
Output:	whether the correct 400 response is sent back

3.26.2 Empty request

DEP - 3	DataEndpoint.retrieveData()
Tests if a request that contains no datatypes returns an empty object	
Test items:	`test/services/dataEndpoint.test.ts`
Precondition:	GameBus requests are mocked
Input:	empty dataTypes list
Output:	whether an empty object is returned

3.26.3 Request glucose data

DEP - 4	DataEndpoint.retrieveData()
Tests if a request that contains the glucose datatype in the dataTypes list returns an object with an empty glucose data array and no other keys are included	
Test items:	`test/services/dataEndpoint.test.ts`
Precondition:	GameBus requests are mocked
Input:	data types list with glucose
Output:	whether an object that contains the glucose key with an empty list as value is returned

3.26.4 Request insulin data

DEP - 5	DataEndpoint.retrieveData()
Tests if a request that contains the insulin datatype in the dataTypes list returns an object with an empty insulin data array and no other keys are included	
Test items:	`test/services/dataEndpoint.test.ts`
Precondition:	GameBus requests are mocked
Input:	data types list with insulin
Output:	whether an object that contains the insulin key with an empty list as value is returned

3.26.5 Request mood data

DEP - 6	DataEndpoint.retrieveData()
Tests if a request that contains the mood datatype in the dataTypes list returns an object with an empty mood data array and no other keys are included	
Test items:	`test/services/dataEndpoint.test.ts`
Precondition:	GameBus requests are mocked
Input:	data types list with mood
Output:	whether an object that contains the mood key with an empty list as value is returned

3.26.6 Request food data

DEP - 7	DataEndpoint.retrieveData()
Tests if a request that contains the food datatype in the dataTypes list returns an object with an empty food data array and no other keys are included	
Test items:	`test/services/dataEndpoint.test.ts`
Precondition:	GameBus requests are mocked
Input:	data types list with food
Output:	whether an object that contains the food key with an empty list as value is returned

3.26.7 Request exercise without parameters

DEP - 8	DataEndpoint.retrieveData()
Tests if a request that contains the exercise datatype in the dataTypes list but with no specified exerciseTypes array returns an object with an empty exercise data array and no other keys are included	
Test items:	`test/services/dataEndpoint.test.ts`
Precondition:	GameBus requests are mocked
Input:	data types list with exercise
Output:	whether an object that contains the exercise key with an empty list as value is returned

3.26.8 Request exercise with parameters

DEP - 9	DataEndpoint.retrieveData()
Tests if a request that contains both the exercise datatype in the dataTypes list and a specified exerciseTypes list returns an object with an empty exercise data array and no other keys are included	
Test items:	`test/services/dataEndpoint.test.ts`
Precondition:	GameBus requests are mocked
Input:	data types list with exercise and exerciseTypes is included
Output:	whether an object that contains the exercise key with an empty list as value is returned

3.26.9 Request all data types

DEP - 10	DataEndpoint.retrieveData()
Tests if a request that contains all data types in the dataTypes list returns an object with empty data arrays for all possible dataTypes	
Test items:	`test/services/dataEndpoint.test.ts`
Precondition:	GameBus requests are mocked
Input:	data types list with all data types
Output:	whether an object that contains all keys with an empty list as values is returned

3.26.10 Parsing data type list

The following test cases test if the data type list parameter (in comma separated string format) is correctly parsed to an array of the DataType enum.

DEP - 11	DataEndpoint.parseDataTypes()
Tests if an empty string is parsed to an empty data type array	
Test items:	`test/services/dataEndpoint.test.ts`
Precondition:	-
Input:	empty data types list
Output:	whether an empty array is returned

DEP - 12	DataEndpoint.parseDataTypes()
Tests if a single element in the list is parsed to an array with the element as DataType enum type	
Test items:	`test/services/dataEndpoint.test.ts`
Precondition:	-
Input:	data types list with a single entry
Output:	whether an array with the correct DataType element is returned

DEP - 13	DataEndpoint.parseDataTypes()
Tests if multiple elements in the list are parsed to an array with the elements as DataType enum type	
Test items:	`test/services/dataEndpoint.test.ts`
Precondition:	-
Input:	data types list with multiple entries
Output:	whether an array with the correct elements is returned

3.26.11 Parsing data type list - robustness

DEP - 14	DataEndpoint.parseDataTypes()
Tests if duplicates in the data types list are not included in the parsed array	
Test items:	`test/services/dataEndpoint.test.ts`
Precondition:	-
Input:	data types list with multiple entries and duplicates
Output:	whether an array with the correct elements is returned without duplicates

DEP - 15	DataEndpoint.parseDataTypes()
Tests if entries in the data types list that do not exist in the DataType enum are left out of the parsed array	
Test items:	`test/services/dataEndpoint.test.ts`
Precondition:	-
Input:	data types list with multiple entries and non-existing types
Output:	whether an array with the correct elements is returned without non-existent entries

DEP - 16	DataEndpoint.parseDataTypes()
Tests if entries in the data types list that have irregular capitalization or superfluous whitespace are correctly parsed	
Test items:	`test/services/dataEndpoint.test.ts`
Precondition:	-
Input:	data types list with multiple entries and irregular whitespace and capitalization
Output:	whether a list with the unusually formatted but correct elements is parsed to an array without any of the elements missing

3.26.12 Parsing exercise type list

The following test cases test if the exercise types list parameter (in comma separated string format) is correctly parsed to an array of the ExerciseGameDescriptorNames enum.

DEP - 17	DataEndpoint.parseExerciseTypes()
Tests if an empty string is parsed to an empty exercise types array	
Test items:	`test/services/dataEndpoint.test.ts`
Precondition:	-
Input:	empty exercise types list
Output:	whether an empty array is returned

DEP - 18	DataEndpoint.parseExerciseTypes()
Tests if a single element in the list is parsed to an array with the element as ExerciseGameDescriptorNames enum type	
Test items:	`test/services/dataEndpoint.test.ts`
Precondition:	-
Input:	exercise types list with one entry
Output:	whether an array with the correct ExerciseGameDescriptorNames element is returned

DEP - 19	DataEndpoint.parseExerciseTypes()
Tests if multiple elements in the list are parsed to an array with the elements as ExerciseGameDescriptor-Names enum type	
Test items:	`test/services/dataEndpoint.test.ts`
Precondition:	-
Input:	exercise types list with multiple entries
Output:	whether an array with the correct elements is returned

3.26.13 Parsing exercise type list - robustness

DEP - 20	DataEndpoint.parseExerciseTypes()
Tests if duplicates in the exercise types list are not included in the parsed array	
Test items:	`test/services/dataEndpoint.test.ts`
Precondition:	-
Input:	exercise types list with multiple entries and duplicates
Output:	whether an array with the correct elements is returned without duplicates

DEP - 21	DataEndpoint.parseExerciseTypes()
Tests if entries in the exercise types list that do not exist in the ExerciseGameDescriptorNames enum are left out of the parsed array	
Test items:	`test/services/dataEndpoint.test.ts`
Precondition:	-
Input:	exercise types list with multiple entries and non-existing types
Output:	whether an array with the correct elements is returned without non-existent entries

DEP - 22	DataEndpoint.parseExerciseTypes()
Tests if entries in the exercise types list that have irregular capitalization or superfluous whitespace are correctly parsed	
Test items:	`test/services/dataEndpoint.test.ts`
Precondition:	-
Input:	exercise types list with multiple entries and irregular capitalization and whitespace
Output:	whether a list with the unusually formatted but correct elements is parsed to an array without any of the elements missing

3.26.14 Testing the union format

Besides retrieving data per data type, the data endpoint has an option to unify all retrieved data into one array of entries that are identified by timestamp. This process of unification also needs test cases.

DEP - 23	DataEndpoint.unionData()
Tests if multiple data types with the same timestamp can be unified into an array with one object	
Test items:	`test/services/dataEndpoint.test.ts`
Precondition:	-
Input:	mocked array with data for several data types but with the same timestamp
Output:	whether an array with one object that contains all data fields is created

DEP - 24	DataEndpoint.unionData()
Tests if multiple data types with the two different timestamps can be unified into an array with two objects	
Test items:	`test/services/dataEndpoint.test.ts`
Precondition:	-
Input:	mocked array with data for several data types with two different timestamps
Output:	whether an array with two objects that contain all data fields is created

3.26.15 Post mood data

MEP - 1	post to /mood endpoint
Tests whether the /mood endpoint works as intended and mood data can be posted to it	
Test items:	`test/routes.test.ts`
Precondition:	-
Input:	mocked moodModel data
Output:	whether the mood data is correctly posted to the endpoint

MEP - 2	post existing activity to /mood endpoint
Tests whether the /mood endpoint works as intended and existing mood data can be posted to it and overwritten	
Test items:	`test/routes.test.ts`
Precondition:	input has activity ID from GameBus
Input:	mocked moodModel data with activity ID
Output:	whether the mood data is correctly posted to the endpoint

MEP - 3	post activity to /mood endpoint
Tests whether the /mood endpoint rejects requests if no valid timestamp, arousal or valence is specified	
Test items:	`test/routes.test.ts`
Precondition:	-
Input:	mocked authentication token
Output:	whether status code 400 is returned

MEP - 4	post existing activity to /mood endpoint
Tests whether the /mood endpoint rejects put (edit) requests if no valid timestamp, arousal or valence is specified	
Test items:	`test/routes.test.ts`
Precondition:	input has activity ID from GameBus
Input:	mocked authentication token
Output:	whether status code 400 is returned

3.26.16 Post insulin data

IEP - 1	post to /insulin endpoint
Tests whether the /insulin endpoint works as intended and insulin data can be posted to it	
Test items:	`test/routes.test.ts`
Precondition:	-
Input:	mocked insulinModel data
Output:	Whether the insulin data is correctly posted to the endpoint

3.26.17 Post insulin data

IEP - 2	post to /insulin endpoint
Tests whether posting to the /insulin endpoint is rejected if no valid timestamp, insulin amount and insulin type is specified	
Test items:	`test/routes.test.ts`
Precondition:	-
Input:	mocked authentication token
Output:	Whether status code 400 is returned

3.26.18 Update insulin data

IEP - 3	post existing insulin entry to /insulin endpoint
Tests whether putting (sending edit requests) to the /insulin endpoint is rejected if no valid timestamp, insulin amount and insulin type is specified	
Test items:	`test/routes.test.ts`
Precondition:	input has activity ID from GameBus
Input:	mocked authentication token
Output:	Whether status code 400 is returned

3.27 Supervisor / user role endpoints

3.27.1 Logging a user's token

SEP - 1	post to /supervisor/logToken
Tests if logging a user's token without providing an email and token is rejected	
Test items:	`test/routes.test.ts`
Precondition:	-
Input:	mocked user authorization token
Output:	whether the expected response error code is returned

3.27.2 Requesting supervisor role

SEP - 2	post to /supervisor/request
Tests if requesting the supervisor role without providing the emails of the supervisor and 'child' (normal) user is rejected	
Test items:	`test/routes.test.ts`
Precondition:	-
Input:	mocked user authorization token
Output:	whether the expected response error code is returned

3.27.3 Getting a user's token

SEP - 3	get from /supervisor/getToken
Tests if getting a normal user's token as a supervisor is rejected if no emails for the supervisor and 'child' (normal) user are specified	
Test items:	`test/routes.test.ts`
Precondition:	-
Input:	mocked user authorization token
Output:	whether the expected response error code is returned

3.27.4 Getting aspiring supervisors as normal user

SEP - 4	get from /supervisor/getSupervisors
Tests if getting a list of aspiring supervisors is rejected if no 'child' (normal) user email is specified	
Test items:	`test/routes.test.ts`
Precondition:	-
Input:	mocked user authorization token
Output:	whether the expected response error code is returned

3.27.5 Getting approved supervisors as normal user

SEP - 5	get from /supervisor/getSupervisors
Tests if getting a list of approved supervisors is rejected if no 'child' (normal) user email is specified	
Test items:	`test/routes.test.ts`
Precondition:	-
Input:	mocked user authorization token
Output:	whether the expected response error code is returned

3.27.6 Getting normal users that are supervised by a specific supervisor

SEP - 6	get from /supervisor/getChildren
Tests if getting a list of 'child' (normal) users for a supervisor is rejected if no supervisor email is specified	
Test items:	`test/routes.test.ts`
Precondition:	-
Input:	mocked user authorization token
Output:	whether the expected response error code is returned

3.27.7 Rejecting supervisor permission

SEP - 7	post to /supervisor/retractPermission
Tests if retracting a supervisor role is rejected if no emails for the normal and supervisor user are specified	
Test items:	`test/routes.test.ts`
Precondition:	-
Input:	mocked user authorization token
Output:	whether the expected response error code is returned

3.27.8 Getting role

SEP - 8	get from /supervisor/role
Tests if retrieving a user's role is rejected if no user email is specified	
Test items:	`test/routes.test.ts`
Precondition:	-
Input:	mocked user authorization token
Output:	whether the expected response error code is returned

3.27.9 Full supervisor endpoint functionality

SEP - 9	get from /supervisor/role
Tests if a sequence of steps, executed in real life use cases, is succesfully executed	
Test items:	`test/routes.test.ts`
Precondition:	-
Input:	mocked user authorization token
Output:	whether all steps return response code 200 and returned data is as expected

3.28 Authentication endpoint

3.28.1 Login sequence

AEP - 1	get from /login
Tests if starting a login attempt is rejected if no user email is specified	
Test items:	`test/routes.test.ts`
Precondition:	-
Input:	mocked user authorization token
Output:	whether response code 400 is returned

AEP - 2	get from /login
Tests if ending a login attempt is rejected if no login token is specified	
Test items:	`test/routes.test.ts`
Precondition:	-
Input:	mocked user authorization token
Output:	whether response code 400 is returned

3.28.2 Register GameBus callback

AEP - 3	post to /gamebus/callback
Tests if registering a GameBus callback is rejected if the necessary user information is missing	
Test items:	`test/routes.test.ts`
Precondition:	-
Input:	mocked user authorization token
Output:	whether response code 400 is returned

AEP - 4	post to /gamebus/callback
Tests if registering a GameBus callback is rejected if there is no ongoing login attempt at the time of registration	
Test items:	`test/routes.test.ts`
Precondition:	-
Input:	mocked user authorization token
Output:	whether response code 400 is returned

3.29 GameBus activity endpoint

ACEP - 1	post to /activities/delete
Tests if deleting a GameBus activity is rejected if no activity ID is specified	
Test items:	`test/routes.test.ts`
Precondition:	-
Input:	mocked user authorization token
Output:	whether response code 400 is returned

3.30 Nightscout endpoint

NSEP - 1	get from /nightscout
Tests if getting data from nightscout is rejected if no Nightscout website URL is specified	
Test items:	`test/routes.test.ts`
Precondition:	-
Input:	mocked user authorization token
Output:	whether response code 400 is returned

3.31 OneDrive endpoint

ODEP - 1	get from /onedrive/onedrive
Tests if getting food diary data from OneDrive is rejected if no OneDrive token and file path are specified	
Test items:	`test/routes.test.ts`
Precondition:	-
Input:	mocked user authorization token
Output:	whether response code 400 is returned

ODEP - 2	get from /onedrive/login
Tests if logging in to OneDrive through Diabetter is rejected if no saved session is provided and no authorization URL to start the login procedure can be found	
Test items:	`test/routes.test.ts`
Precondition:	-
Input:	mocked user authorization token
Output:	whether response code 403 is returned

ODEP - 3	get from /onedrive/redirect
Tests if the redirect endpoint does not successfully redirect if no valid code is provided from OneDrive	
Test items:	`test/routes.test.ts`
Precondition:	-
Input:	mocked user authorization token
Output:	whether response code 400 is returned

ODEP - 4	get from /onedrive/displayTokens
Tests if the helper endpoint that displays retrieved OneDrive tokens is robust and only works if the necessary tokens are provided	
Test items:	`test/routes.test.ts`
Precondition:	-
Input:	mocked user authorization token
Output:	whether response code 400 is returned

3.32 Profile endpoint

As the get endpoint entirely relies on the JWT (authentication token) and on GameBus, we cannot make any tests for the get endpoint itself as we do not want to store actual user tokens that are coupled to a real account. The fake access token we use can still be used to test a default response that is given if no user information is found.

PEP - 1	post to /profile
Tests if posting user information to GameBus is rejected if the mandatory parameters weight, length and age are not specified	
Test items:	`test/routes.test.ts`
Precondition:	-
Input:	Mocked user authorization token
Output:	Whether response code 400 is returned

PEP - 2	post to /profile
Tests if posting user information to GameBus is rejected if the values for the mandatory parameters weight, length and age are invalid	
Test items:	`test/routes.test.ts`
Precondition:	-
Input:	mocked user authorization token
Output:	whether response code 400 is returned

PEP - 3	get from /profile
Tests if retrieving profile data from GameBus with the fake account results in retrieving default (nulled) profile data	
Test items:	`test/routes.test.ts`
Precondition:	-
Input:	mocked user authorization token
Output:	whether status code 200 and the expected default response data is returned

4 Test procedures

4.1 Unit test procedure - Diabetter backend

4.1.1 Purpose

This test procedure describes how to execute all tests for the Diabetter backend. The results and coverage reports generated after completing this procedure help determine if and with which certainly it can be stated that the program works as intended.

4.1.2 procedure steps

1. Open a terminal in the root folder of the project
2. Run the command `npm install` to install all necessary packages (make sure the device is connected to the internet for this step)
3. Run the command `npm test` to execute all tests. This will also generate a new coverage report.

5 Test reports

This section contains the results of executing the test suites that contain the test cases described in section 3 with the procedure of section 4. Furthermore, code coverage is provided.

5.1 Results

The outcome of all test cases is provided in a separate document included with the code, this document can be found in `test-report.html` in the root folder of the backend project. An overview of all test suite results can be found in figure 1 and 2.

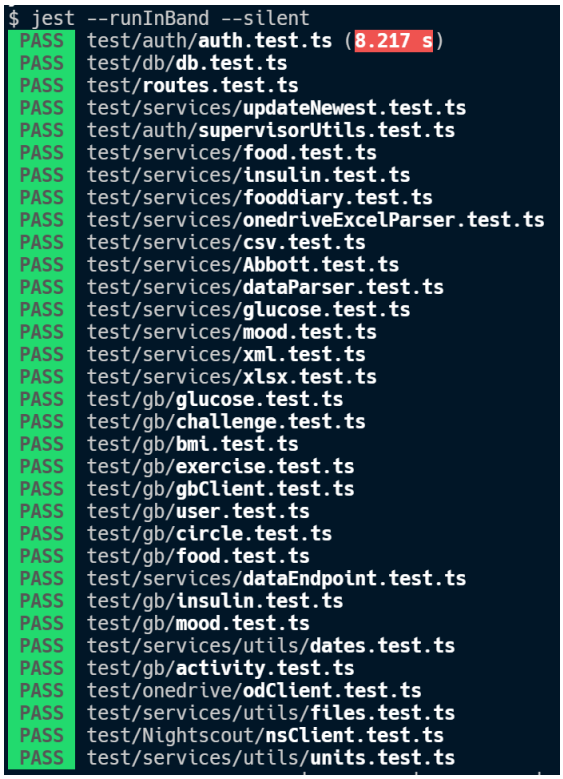


Figure 1: All test suites that are executed pass

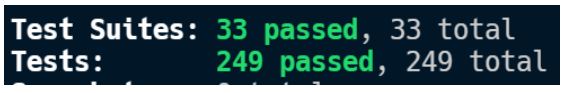


Figure 2: All tests and test suites

5.2 Coverage

The coverage report is automatically generated by our testing framework, Jest, and will be generated upon running the test cases. This test coverage report will also be included in the final code hand-in and can be found in `test/coverage/lcov-report/index.html`. From this report, it is possible to click on each section and see exactly which lines are and are not covered. While most files have excellent coverage, there are a few files for which we were not able to reach full coverage. For the database client, we were unable to cover the error branches since we could not consistently cause these errors. For the endpoints, we did not want to test the full endpoint since we did not want to send the actual request (as this would involve GameBus and we wanted to keep the platform out of our tests), so we only tested the 4xx responses. For the data parsers we were again unable to cover certain branches because of error handling that we could not cause ourselves consistently. Figure 3 shows the total test coverage from the test suites.

All files

93.67% Statements 7578/8090 **78.55%** Branches 692/881 **95.24%** Functions 280/294 **93.67%** Lines 7578/8090

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

File	Statements		Branches		Functions		Lines	
src	100%	48/48	100%	1/1	100%	0/0	100%	48/48
src/db	87.8%	367/418	67.35%	33/49	100%	22/22	87.8%	367/418
src/gb	95.83%	299/312	81.48%	22/27	100%	18/18	95.83%	299/312
src/gb/auth	100%	41/41	100%	3/3	100%	2/2	100%	41/41
src/gb/objects	96.06%	2416/2515	94.35%	217/230	94.74%	90/95	96.06%	2416/2515
src/middlewares	100%	8/8	100%	0/0	100%	0/0	100%	8/8
src/nightscout	100%	150/150	81.82%	9/11	100%	8/8	100%	150/150
src/onedrive	100%	340/340	87.8%	36/41	100%	15/15	100%	340/340
src/routes	87.06%	868/997	59.6%	59/99	100%	1/1	87.06%	868/997
src/services	100%	448/448	91.67%	55/60	100%	16/16	100%	448/448
src/services/dataParsers	90.18%	1056/1171	56.36%	93/165	87.5%	35/40	90.18%	1056/1171
src/services/fileParsers	90.94%	291/320	85.29%	29/34	93.33%	14/15	90.94%	291/320
src/services/food	99.12%	226/228	88%	22/25	100%	10/10	99.12%	226/228
src/services/glucose	95.13%	215/226	62.96%	17/27	100%	9/9	95.13%	215/226
src/services/insulin	94.44%	187/198	70%	14/20	88.89%	8/9	94.44%	187/198
src/services/mood	100%	54/54	100%	6/6	100%	4/4	100%	54/54
src/services/utils	97.77%	307/314	91.38%	53/58	100%	15/15	97.77%	307/314
src/utils	85.1%	257/302	92%	23/25	86.67%	13/15	85.1%	257/302

Code coverage generated by istanbul at Mon Jun 28 2021 11:52:58 GMT+0200 (Central European Summer Time)

Figure 3: Overall test coverage