Software Transfer Document

- for Diabetter, a Diabetes data dashboard

Team members

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

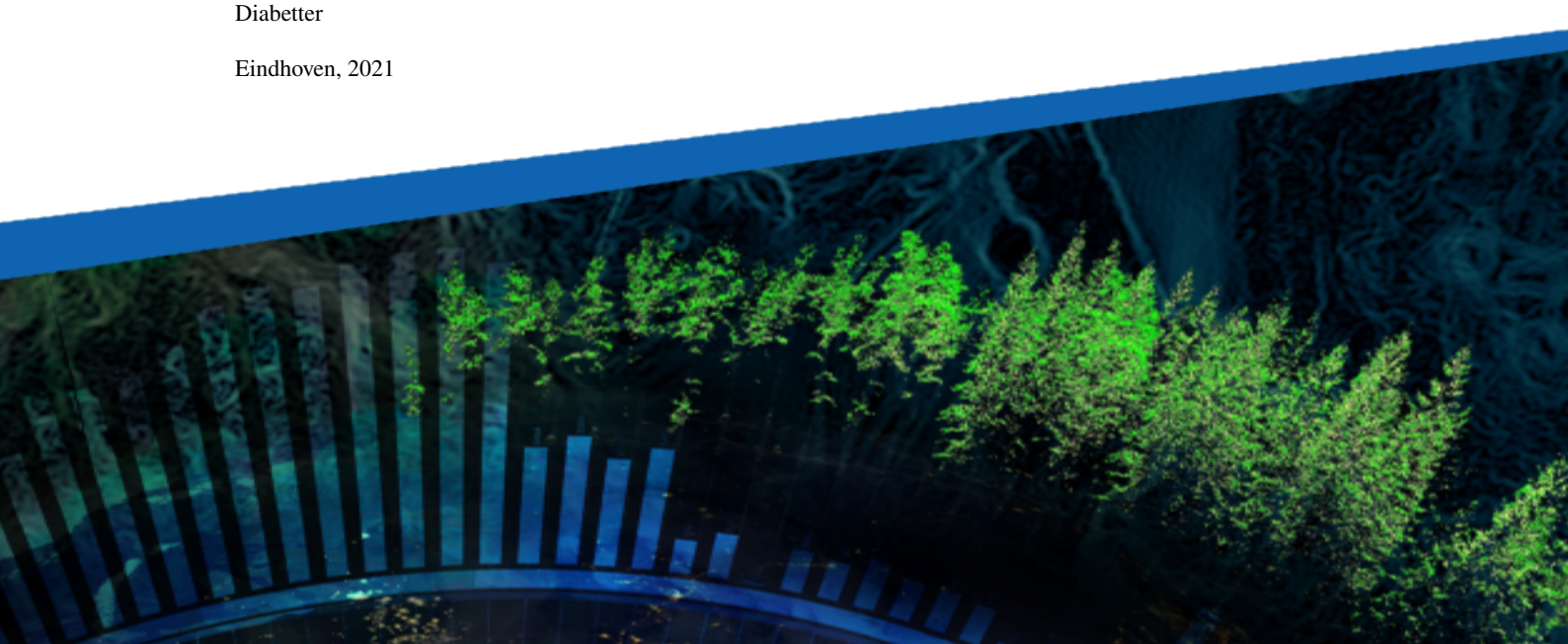
Supervisor
G. Kahraman

Eindhoven University of Technology

Department of Mathematics and Computer Science

Diabetter

Eindhoven, 2021



Abstract

This document is the Software Transfer Document (STD) for the Diabetter dashboard, a web-based data application. It describes how and in which environment the application was built. Furthermore, the steps required to build and install the application are provided. Additionally, the results of the acceptance test are mentioned, as well as the changes and modifications, which were made for the transfer. This document complies with the ESA standard [1].

Contents

Document status sheet and change records

Document title	Software Transfer Document
Document identifier	Diabetter.STD/0.1
Authors	Alexandra Nikolova Vasil Shteriyanov
Document status	Finished, unapproved

Document History

Version	Date	Authors	Description
0.1	30-06-2021	Vasil & Alexandra	First version.

Document Change Record

Version	Date	Version	Changes
0.1	30-06-2021	Vasil & Alexandra	First version.

1 Introduction

1.1 Purpose

The Software Transfer Document represents a detailed explanation of the procedure for building and installing the Diabetter dashboard. It is intended for the customer and any other third parties interested in expanding the project. The document will allow the readers to independently continue the development of the application. Furthermore, it will include modifications to the software which were completed during the transfer phase.

1.2 Scope

The Diabetter dashboard is the outcome of the final Software Engineering Project of Computer Science students at the Eindhoven University of Technology. The application is developed for the client M. Chaudron and GameBus platform owner P. Van Gorp. GameBus [2] is a platform that promotes a mentally and physically healthy lifestyle in an engaging manner.

The Diabetter dashboard is a web application. The main purpose of the product is to visualize data considering the factors of interest for Diabetes type 1 patients in a simple and intuitive way. Enabling analyses and showing possible interactions between these factors is an important feature of the application. Furthermore, GameBus functionalities like authentication and data storage are part of the product.

1.3 List of definitions

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

Diabetter	The developed web application, including the source code.
GameBus data provider	Data source for GameBus.
GameBus game descriptor	A list of properties, with their corresponding type.

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

1.3.2 Abbreviations

ATP	Acceptance Test Plan
CSV	Comma Separated Values File
ESA	European Space Agency
NPM	Node Package Manager
PDF	Portable Document Format
SDD	Software Design Document
STD	Software Transfer Document
URD	User Requirements Document
UTP	Unit Test Plan

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] GameBus. Gamebus.https://blog.gamebus.eu/. 2021.

[3] Jest. Jest.https://jestjs.io/. 2021.

[4] Node.js.https://nodejs.org/en/. 2021.

[5] Diabetter. Acceptance test plan. 2021.

[6] Diabetter. Software design document. 2021.

[7] Diabetter. Software user manual. 2021.

[8] Diabetter. Unit test plan. 2021.

[9] Diabetter. User requirements document. 2021.

2 Build Procedure

This section explains how the Diabetter web application can be built. The building can be done on systems running Windows 10. Other systems may work as well, such as machines running macOS and Linux, but we cannot guarantee this. Git was used for version control. Jest [3] is used for testing the back-end code. The build process should take less than 2 minutes. NPM will return some build warnings, but these do not affect the overall project.

2.1 Prerequisites

The submission zip has been unzipped. In order to build the software Node.js version, 14.17.1 LTS [4] needs to be installed.

2.2 Back-end

The back end is written in Typescript programming language. The JavaScript runtime environment Node.js is used.

1. Navigate to the `SEP-backend` directory in the unzipped submission folder and open the command prompt in it
   Alternatively:
   Go to the directory where you want to put the backend files and open the command prompt in it. Clone the backend repository by issuing the following command:

   git clone https://github.com/RoelMK/SEP-backend.git

   Afterwards, navigate to the `SEP-backend` directory and open the command prompt in it.

2. Run the following terminal command to install the necessary dependencies

   npm install

2.3 Front-end

The front end is written in HTML, Javascript, and CSS using the Vue framework. The following steps need to be executed to build the front-end.

1. Navigate to the `SEP-frontend/diabetter-dashboard` directory in the unzipped submission folder and open the command prompt in it.

   Alternatively:
   Go to the directory where you want to put the frontend files and open the command prompt in it. Clone the frontend repository by issuing the following command:

git clone https://github.com/RoelMK/SEP-frontend.git

Afterwards, navigate to the `SEP-frontend/diabetter-dashboard` directory in the command prompt.

2. Run the following terminal command to install the necessary dependencies

npm install

3. Run the following terminal command, which compiles and minifies for production

npm run build

4. The build folder called $dist$ is generated in the `SEP-frontend/diabetter-dashboard` directory. The folder can be used to host the front end of the application.

3 Installation Procedure

This section will provide installation instructions for our software. After the building and installation are done the front-end and the back-end can be run. The installation can be done on systems running Windows. Other systems may work as well, such as machines running macOS and Linux, but we cannot guarantee this. No problems should arise during the installation. The time needed to install depends on whether the user wants to modify the configuration. If no modifications to the configuration are made, the installation should take less than 2 minutes.

3.1 Prerequisites

The building procedure from the previous section has been complete successfully. Node.js version 14.17.1 LTS is installed.

3.2 Back-end

1. The back-end code we provide includes a `.env` file and a `.env.example` file in the `SEP-backend` directory.
2. Make sure these files are present. In case `.env` file is not present create a file called `.env` in the `SEP-backend` directory.
3. Make sure all variables present in the `.env.example` file are present in the `.env` file.
4. Make sure the following information is present in the `.env` file:

   TEST_TOKEN=aaaaaaaa-bbbbb-cccc-dddd-fffffffffff
   NODE_ENV=production
   TOKEN_SECRET=h834fh834fh834th834thu9q34t
   TOKEN_ISSUER=http://diabetter.win.tue.nl
   TOKEN_EXPIRES_IN=30d
   DATABASE=localdb.db
   GAMEBUS_TOKEN_SECRET=2slbwyBzfgzUIvXZ
   AZURE_CLIENT_SECRET=ai5.BQ2LiE2-_XFk3l22 0d5WgMLt_1x2I
   AZURE_CLIENT_ID=68181ff8-6dca-43f3-b2aa-0b7687d8e4f0
   GAMEBUS_CLIENT_AUTH_HEADER=RHVlbCBMaW5rczpFUTVBSG8xVDIwRVVNa0hXUnFVNg==
   ONEDRIVE_FRONTEND_REDIRECT=http://localhost:8080/profile
   ONEDRIVE_BACKEND_REDIRECT=http://localhost:5000/onedrive/redirect
   PORT=5000

5. The above configuration only works for our own GameBus data provider ("Daily_run") with pre-defined game descriptors. In case the user wants to modify some these fields, he can do so by changing the values of each field, such as the port field. However, changes may require the setup of a new data provider. A detailed description of how the configuration can be changed can be found in the `readme` file in the `SEP-backend` directory.

6. Open the command prompt in the `SEP-backend` directory
7. Run the following command to start the compiled server

npm run start

If all of the steps above are completed the server is running.

3.3 Front-end

1. The front-end code we provide includes a `.env` file and a `.env.example` file in the `SEP-frontend/diabetter-dashboard` directory.
2. Make sure these files are present. In case these files are not present go to step 4.
3. Make sure all variables present in the `.env.example` file are present in the `.env` file.
4. Make sure the following information is present in the `.env` file:

   VUE_APP_I18N_LOCALE=en
   VUE_APP_I18N_FALLBACK_LOCALE=en

   In case no such file is present, create a file called `.env` in the `SEP-frontend/diabetter-dashboard` directory. Afterwards, copy the variables and their values mentioned above and paste them in that file. Finally, save the file.
5. Open the command prompt in the `SEP-frontend/SEP-frontend` directory
6. Run the following command to run the front-end

npm run serve

4 Configuration Item List

This section presents a list of all the deliverables of the Diabetter project. All items on this list shall be delivered in a zip archive called `diabetter.zip`. Additionally, a `readme` file will be included, which will explain the file structure and which source code should be checked and which not. The provided documentation is in PDF format. The documents can be located in the `documentation` folder, the presentations are in the `presentations` folder, the frontend and the backend code is in the `code` folder, and the poster is in the `poster` folder.

4.1 Documentation

The documents, which were produced for Diabetter:

- Acceptance Test Plan (ATP) [5]
- Software Design Document (SDD) [6]
- Software Transfer Document (STD)
- Software User Manual (SUM) [7]
- Unit Test Plan (UTP) [8]
- User Requirements Document (URD) [9]

4.2 Presentations

The presentations produced for Diabetter:

- Intermediate Presentation
- Final Presentation

4.3 Poster

The poster in pdf format, which was produced for Diabetter.

4.4 Code

The code, which was produced for Diabetter:

- Diabetter Front-end
- Diabetter Back-end

5 Acceptance Test Report Summary

The acceptance test was conducted on 18/06/2021 from 14:30 to 16:30. The acceptance test was executed using ATP version 0.3. All team members were present, as well as the client and one of the managers. During the test, all test cases were passed according to their specification in ATP 0.3. However, the client had problems with the implementation of some "Should have" and "Could have" requirements as his interpretation of those requirements did not match the team's interpretation. As a result, the client requested some small changes and the team agreed to them. Despite this problem, the acceptance test was considered successful as all the "Must have" requirements were met.

The "Should have" and "Could have" requirements, which were not implemented and were not present in ATP version 0.3 are the following:

GEN 0.2, GEN 0.3
USER 2.4, USER 4.3, USER 4.4, USER 5.5
HM 1.2 - HM 1.12, HM 2.12, HM 2.15 - HM 2.20, HM 2.22, HM 4.8, HM 4.9, HM 4.18 - HM 4.20
ACT 0.1 - ACT 0.3, ACT 1.7 - ACT 1.10, ACT 2.5 - ACT 2.9, ACT 3.1 - ACT 3.5, ACT 3.7
FOOD 0.2, FOOD 1.2, FOOD 1.4 - FOOD 1.10, FOOD 2.5, FOOD 3.2
EMOT 0.4, EMOT 1.5
PS 0.5
PERF 0.5 - PERF 0.14
REL 0.1
ENV 0.3 - ENV 0.10
DATA 0.7 - DATA 0.15, DATA 1.5, DATA 1.6, DATA 1.8 - DATA 1.16

All other requirements from the URD were implemented and present in the test procedure for ATP 0.3.

6 Software Problem Reports

During the acceptance test, the customer encountered an issue with the profile page. The sliders for the glucose level ranges in the Profile page could overlap, which was undesirable. However, this problem did not affect the requirements related to the sliders. Another problem, which did not affect the success of the test cases was a misalignment in the axis pointer. Additionally, the totals of the insulin/carbohydrates/calories/glucose for a specified time frame were not implemented fully. However, the test for the requirements related to this functionality was not present in ATP 0.3 as the requirements were not "Must haves".

7 Software Change Requests

After the acceptance test the client requested changes, some of which were outside of the scope of the URD:

- Sliders for glucose ranges cannot overlap (user experience)
- When selecting an exercise, filter the visualization 2 hours before and after the activity instead of only the time of the activity (adopting the client's interpretation of the requirement)
- Fix the axis pointer in the visualization (user experience)
- Compute total insulin/carbohydrates/calories/glucose for a specified time frame
- Combined filtering on multiple tables at the same time
- Visualizations for aggregates
- Select multiple table entries and show them all at once

Some of these issues were fixed and they are described in the next section. The other changes were not implemented due to time constraints or due to constraints regarding the chosen libraries.

8 Software Modification Reports

This section contains the software modifications and fixes to existing features, which were completed during the transfer phase. These changes were requested and approved by the client.

8.1 Non-overlapping sliders

Originally, although the sliders passed the requirements related to them, the slider for each glucose type could take any value despite the values of other glucose types. This was of course undesirable. As a result, the team decided to fix this issue. A function was applied to each slider, which checks the current value of a slider. If the value overlaps with the intervals set by other sliders, the function will set the value of the slider to the smallest or largest allowed value considering other intervals.

8.2 When selecting an exercise, filter the visualization 2 hours before and after the activity.

Before the change, when an exercise was selected from the table, the visualization would be filtered on the exact time of the activity. However, the client felt it would be better if we extended the filter to two hours before and after an exercise. To do this, we subtracted 2 hours from the starting moment of the exercise and added 2 hours to the ending moment of the exercise.

8.3 Fix the axis pointer of the visualization overview

During the acceptance test, the axis pointer in the visualization overview was misaligned for the individual sub-visualizations. Although this problem did not cause any requirement to fail, the team decided to fix this issue. This axis pointer was fixed and the alignment was adjusted properly for each sub-visualization.

8.4 Compute total insulin/carbohydrates/calories/glucose for a specified time frame

During the acceptance test, the off statistics of insulin/carbohydrates/calories/glucose only worked for a specified date interval, but not for a specified time interval. The team decided to fix this issue as the client requested this. Currently, the input data, which is passed to the statistics functions, is filtered based on the chosen time frame. Thus, the statistics are now updated based on a specified time frame in addition to specified dates.