Handover Document

Baby Monitor Simulator  
Fontys Hogeschool | Tilburg

|  |  |
| --- | --- |
| **Author**: | Amy Hendriks, Cyrion van Dongen, Tim de Moor, Ben Whelan, Pieter van der Kooij |
| **Version:** | 1.1 |
| **Date**: | 1-06-2023 |

# Version Control

|  |  |  |
| --- | --- | --- |
| **Version (v)** | **Date** | **Changes** |
| 0.1 | 22-05-2023 | Initial Document setup, introduction, system, documentation |
| 0.2 | 08-06-2023 | The System, Documentation, Implementation, Details on BE and MATLAB |
| 0.3 | 12-06-2023 | Frontend code explanation, MATLAB code explanation |
| 1.0 | 15-06-2023 | Frontend code setup |
| 1.1 | 19-06-2023 | More docs |

# Definitions, Acronyms and Abbreviations

|  |  |  |
| --- | --- | --- |
| Term | Abbreviation | Definition |
| Cardiotocography | CTG | Cardiotocography is a technique used to monitor the fetal heartbeat and the uterine contractions during pregnancy and labor. The machine used to perform the monitoring is called a cardiotocograph. |
| MATLAB | - | MATLAB is a programming platform designed specifically for engineers and scientists to analyze and design systems and products that transform our world. |
| FE | Front End | The code behind the web application that will display the graphs. |
| BE | Back End | The server-side code which relays information between the Front End and the MATLAB code. |

Contents

[Version Control 2](#_Toc137461243)

[Definitions, Acronyms and Abbreviations 2](#_Toc137461244)

[1 Introduction 4](#_Toc137461245)

[1.1 Purpose of the system 4](#_Toc137461246)

[1.2 Scope of the system 4](#_Toc137461247)

[1.3 Objectives and success criteria of the project 4](#_Toc137461248)

[2 The system 5](#_Toc137461249)

[2.1 MATLAB model 5](#_Toc137461250)

[2.2 Proposed system for the project 5](#_Toc137461251)

[3 Available documentation 6](#_Toc137461252)

[4 Project implementation 7](#_Toc137461253)

[4.1 Currently implemented 7](#_Toc137461254)

[4.2 Recommended further steps 7](#_Toc137461255)

[5 Git and Git Management 8](#_Toc137461256)

[6 Code Setup 10](#_Toc137461257)

[6.1 Front End 10](#_Toc137461258)

[6.2 Back End 11](#_Toc137461259)

[6.3 MATLAB 12](#_Toc137461260)

[7 Code explanation 21](#_Toc137461261)

[7.1 Front End 21](#_Toc137461262)

[7.2 Back End 22](#_Toc137461263)

[7.3 MATLAB 23](#_Toc137461264)

# 1 Introduction

## 1.1 Purpose of the system

The goal of the system, once implemented, is primarily aimed at providing a tool for healthcare professionals to train their CTG interpretation skills on. Not only should it train them in reading the graphs, but also teach how certain inputs affect said graphs.

## 1.2 Scope of the system

The system will cover the following aspects:

* An intuitive web-application as an interface for the end user, displaying the MATLAB calculations as graphical interpretations and allowing the manipulation of values.
* A backend service to connect with MATLAB for receiving graph calculations to display on the web user-interface.
* Import/export data functionality to allow external data to be read and exported from the UI (User Interface).
* Template selection for default values; starting points for the end-user.

## 1.3 Objectives and success criteria of the project

* A successful translation of MATLAB data through the backend and displayed in the frontend.
* An intuitive user interface that can be used and picked up by any clinician, regardless of their knowledge of computers.

# 2 The system

## 2.1 MATLAB model

A mathematical model has been developed in MATLAB that can be used to describe the physiology of a CTG scan, allowing clinicians to understand how these uterine contractions affect the heartbeat during labor. This system is purely run through MATLAB and the user interface is therefore very developer oriented and not intuitive for a general user.   
This is a problem for training clinicians since many are not familiar with a technology tool such as MATLAB. Besides that, the model will also give more graphs as output than necessary. The application that this project develops should allow clinicians to manipulate values through a simple user-interface, requiring no dependencies and build-tools, thus providing a far more streamlined process than what is currently in place.

## 2.2 Proposed system for the project

The project is primarily aimed at providing a tool for healthcare professionals to train their CTG interpretation skills on. First of all they should have a simple way to see the graphs in order to learn to read them. Second, the system should provide a simple and understandable way to manipulate the input values. This should help the clinicians understand what values have an impact and how, so that they can understand what to apply when things go awry in the real situations.

**Key Points:**

* The mathematical model has already been fully developed in MATLAB.
  + Fully functional for research users
  + NOT functional for clinical users
* This means that the project is **targeted specifically for clinical end users, as a training tool**, and therefore needs:
  + An **intuitive interface**, easy for clinical users to pick up and use (frontend):
    - Multiple interfaces are also permitted.
    - It is expected that this **user interface will provide graphical representations of the data** from the underlying MATLAB algorithms.
    - Simple to understand and use, since one cannot assume that clinical professionals know their way around computers and applications.
* This new tool needs to **utilize the existing MATLAB mathematical model** (including MATLAB within the project).
  + See: [https://nl.mathworks.com/products/MATLAB/MATLAB-and-other-programming-languages.html](https://nl.mathworks.com/products/matlab/matlab-and-other-programming-languages.html)
  + This program is intended to be used as an educational tool rather than in practice.
* The main data graphs we need to display are:
  + **pO2** – *Partial Oxygen Pressure*   (ONLY AVAILABLE FROM THE SIMULATOR)
  + **MAP** – Mean Arterial Blood Pressure  (ONLY AVAILABLE FROM THE SIMULATOR)
  + **FHR** – *Fetal Heart Rate*  (ALWAYS AVAILABLE IN REAL SITUATIONS)
  + **UP** – *Uterine Contractions*  (ALWAYS AVAILABLE IN REAL SITUATIONS)

Since some variables are not available in real situations and only via MATLAB, these should also be more prominent in our UI. These variables can affect each other. **FHR** & **UP** are what usually form the **CTG**.

# 3 Available documentations

Documents are available on the teams' group, “Baby Monitor Simulator”.

* **Analysis Document**

Project description, functional requirements, non-functional requirements, use cases.

* **API Endpoints Doc**

Short explanations of how Endpoints would work between FE and BE.

* **Technical Design Document**

Diagrams for the designs on how parts of the system would communicate with one another.

* **Test Strategy Document**

Setup plan for which test should exist and how those would be implemented.

* **Baby Monitor Simulator – Project Description**

Initial project description from Fontys.

* **PhD Thesis regarding the MATLAB code**

Research document by Client on how and why the MATLAB code works.

# 4 Project implementation

This chapter gives a brief description of all things that have currently been implemented and what functionalities could be looked at next. For deeper detail on some of the requirements, please refer to the Analysis Document.

The backend streams data over server-sent events to the frontend. This data is generated in the backend in the data generation microservice. This microservice generates mock data now, the goal for the next group is to hook this up to MATLAB to generate point by point data.

## 4.1 Currently implemented

* Basic communication between BE and MATLAB.
  + Read exported MATLAB files
  + Stream point by point data to the frontend
* FE can import and export files with graph data
* FE can read and translate MATLAB data and display it on the graphs.
* Basic communication between FE and BE
  + FE can send values to BE
  + BE can send graphs to FE
* Template scenarios which can be chosen on the dashboard (based on the template scenarios in the MATLAB code)

## 4.2 Recommended further steps

**To implement:**

* Implement a downloadable manual.
* Test for both Front- and Back End.
* When the dashboard screen is resized, the FHR and UP should still be above each other, and with the aspect ratio used in the clinic.
* Option to display a random --yet realistic-- scenario on the dashboard.
* Possibility of further import/export types:
  + CSV
  + EXCEL
  + JSON
* Selectable values that can be implemented on the dashboard, to further generate graphs with:
  + Total Blood Volume of the fetus (*input\_f4.m*, line 77)
  + “Medication Option” to calm a ‘labor storm’
  + Umbilical options (Option 2 is required for scenario 3)
  + …
* Fix the aspect ratio for each diagram on the dashboard (some styling bugs on zoom)

**To keep track of:**

* Update the manual page according to any new work done.
* Update the documentation according to any new work done.

# 5 Git and Git Management

The Git contains four main repositories:  
- **S6BabyMonitor**, which has the Back End code. This is written with Kotlin.  
- **s6-baby-monitor-simulator-webui**, which has the code for the Front End, which uses Vue.

Org: <https://github.com/Baby-Monitor-Simulator>’

Backend: <https://github.com/Baby-Monitor-Simulator/S6BabyMonitor>

Frontend: <https://github.com/Baby-Monitor-Simulator/s6-baby-monitor-simulator-webui>

Docs: <https://github.com/Baby-Monitor-Simulator/wiki.github.io>

- CI/CD is currently highly primitive. Only unit tests and builds are run in build server.

### Commits, PR’s and branches

When making commits make sure the commit message is as clear as possibleA screenshot of a computer program

Description automatically generated with medium confidence

To push changes to the development branch you need to create a pull request. When creating a pull request make sure to select your branch and merge to dev.

A screenshot of a computer program

Description automatically generated with medium confidence

A screenshot of a computer program

Description automatically generated with low confidenceBoth front- and back-end have a main branch and a dev branch. The main branch should be locked so you can’t push to it, but this is not the case. So be careful when pushing commits.

# 6 Code Setup

## 6.1 Front End

### Requirements

* Node
* Package Manager (npm, yarn, pnpm)

### Project Setup

Commands can be found in README.md of repository.

A screenshot of a program

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with medium confidence

### Repository Code

|  |  |
| --- | --- |
| Directory | Purpose |
| .github | Pipelines  GitHub Configurations |
| cypress | E2E Tests |
| node\_modules | Dependencies |
| src | Source Code (see below for details) |

The remaining files in the root directory are configuration files for the various dependencies within the project.

#### Repository Source Code

|  |  |
| --- | --- |
| Directory | Purpose |
| assets | Media assets that need to be loaded into the project. |
| components | Sections of the application for easier overview and reusability. |
| enums | Enum types defined here. |
| i18n | Translations (currently only supporting NL and EN). |
| interfaces | Class or object interfaces. |
| repositories | Proper connections with backend should be handled here. |
| router | Multiple views can be routed here. |
| stores | Global variables can be defined here if required. |
| views | Multiple pages can be defined here if required. |

The *app.vue* file is the root component of the project (top of the tree). The *main.ts* file is responsible for plugin integration into project. If you add additional dependencies, you probably configure them here.

### Important Points

* Make sure to write all strings with the i18n style instead of standard strings in order to support translations:

A picture containing text, font, screenshot, graphics

Description automatically generated

* For new strings, translations need to be added here:

A screen shot of a computer

Description automatically generated with medium confidenceFor example:A screen shot of a computer

Description automatically generated with medium confidence

* The file reader component is brittle in the sense that it only supports .txt files generated from MATLAB.
* The graph implementation is rigid and primitive. For a more advanced implementation you will have to rewrite the component from scratch.
* The overlay hover effect should be improved to be clearer for the user.

## 6.2 Back End

In order to get the data from MATLAB to the backend we build a wrapper class that uses the official MATLAB library for java.

MATLABConnection: This is the wrapper class, this is made in order to make sure that the provided arguments are present in the MATLAB enums.



Enums:

There are a couple of different enums in the module, these are used in the wrapper class.

When adding a value, make sure that the variable name corresponds to its MATLAB counterpart. A picture containing text, screenshot, software

Description automatically generated

Whenever a client (frontend) connects to the backend over SSE, this data will be streamed to the frontend over that connection for each graph. Currently, all clients receive the same data. Another thing to do for the next group is to use the sessions that are in the backend to differentiate between clients and give each their own data based on their own settings.

## 6.3 MATLAB

### MATLAB application

The MATLAB code itself can be found within the Teams Environment. In ‘**Baby Monitor Simulator > Archive**’, the ‘**MATLAB Files.zip**’ contains everything you need.

A picture containing text, screenshot, font, design

Description automatically generatedA MATLAB license can be obtained [through Fontys](https://nl.mathworks.com/academia/tah-portal/fontys-hogescholen-31089256.html). Ask your teacher for the license if the link itself does not work. Through here you can also download the application to use on your device.

To find the project files in MATLAB, select the folder that contains them in the folder browser bar. If the correct folder is selected, the files should appear in the ‘Current Folder’ section.

A screenshot of a computer

Description automatically generated

### Required extensions

In order to make the provided model run, a couple of MATLAB extensions are needed. These are:

* Symbolic math toolbox
* Statistics and machine learning toolbox
* Signal processing toolbox

These can be found on the MATLAB add-on store:

A screenshot of a computer

Description automatically generated with medium confidence

When you open the ‘manage add-on’ window you can see which add-ons you have installed.

A picture containing text, font, screenshot

Description automatically generated

### Path variables

To run MATLAB code in the backend make sure you set the path in environment variables

1. Search for environment variables in explorer

A screenshot of a computer screen

Description automatically generated with medium confidence

1. Select ‘Environment Variables’

Graphical user interface, text, application, email

Description automatically generated

1. Under system variables, select the path variable and edit it

A screenshot of a computer

Description automatically generated with medium confidence

1. Click ‘new’ and add the path to where MATLAB is installed,

The default path should be ‘C:\Program Files\MATLAB\R2022b}\bin’

A screenshot of a computer

Description automatically generated

1. Now start MATLAB and under the main tab click on the set path button

Graphical user interface, text, application

Description automatically generated

1. Add the current folder you’re working in to the list of paths

A screenshot of a computer

Description automatically generated with medium confidence

The Kotlin backend should now be able to find the files needed.

### Calling MATLAB functions

In order to call a MATLAB function from kotlin you have to do a couple of things

1. Custom MATLAB function

Graphical user interface, text

Description automatically generatedA picture containing text

Description automatically generated

1. [area] is the name of the variable that is returned
2. calcArea(x,y) is the function name with the parameters ‘x’ and ‘y’
3. parameter datatypes need to be specified
4. make sure the variable you want to return has the same name as step 1.
5. Make sure that the name of the function you want to call is the same as the MATLAB filename
   1. If called externally, only one function per file can exist. If you want to call multiple functions, you have to make multiple MATLAB files.
6. In order to call the MATLAB function from the kotlin backend you have to use the following syntax…
   1. Always instantiate the MATLAB engine first.
      1. To instantiate the engine you have to import ‘com.mathworks.engine.MATLABEngine’ first
   2. Text

      Description automatically generatedInitialize some variables.
   3. To call a MATLAB function use ‘engine.feval()’, this function then needs the functionname as a string parameter and then the parameters.
   4. In order to run a MATLAB script use the ‘engine.eval’ function
   5. Always make sure to close the engine when you’re done.

### Running the model and getting data

A screenshot of a computer

Description automatically generated

1: go to the folder path in your file explorer where the MATLAB project is saved

2: Select the FMPmodel (NOT FMPmodel2, you can even delete FMPModel2 to be sure you don’t use it)

3: Press the run button

A screenshot of a computer screen

Description automatically generated with low confidenceWhen the model starts running the console will spit out some diagnostic dataA screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generatedAfter the model is done running all the data variables will show on the right hand side of the screen since MATLAB only works with global variables.

MATLAB has a couple of different types of variables, these are:

Doubles: every number is a double.

Arrays: an array of doubles with a size defined by the notation “1x{size} double”

Matrices: an array of arrays. Defined by the notation “{sizeX}x{sizeY} double”

Double click to view the data in a variable

A screenshot of a computer

Description automatically generated with medium confidence

### Exporting data

Unfortunately the exporting of large chunks of data does not work normally via the building export function from MATLAB. Therefore we need to manually export the data via plots.

Once you run the model some graphs will appear. In order to extract the numerical data from these graphs you have to select the entire plot and copy paste the data into a txt file.

To export data:

1: Open a graph and select the ‘brush data’ tool of the graph you want to export.

A picture containing text, font, line, diagram

Description automatically generated

2: Select the entire graph

A picture containing text, font, line, plot

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a graph

Description automatically generated with low confidence3: Right click and select ‘export data to clipboard’ You may note the ‘export brushed’ function,

this function doesn’t work.

4: open notepad and paste the data

# 7 Code explanation

## 7.1 Front End

### File Reader

See component `FileReader.vue`

Basic component that allows uploading of files and then read into JSON format into the store arrays (see `import.ts`). Only supports .txt exported MatLab files at this moment.

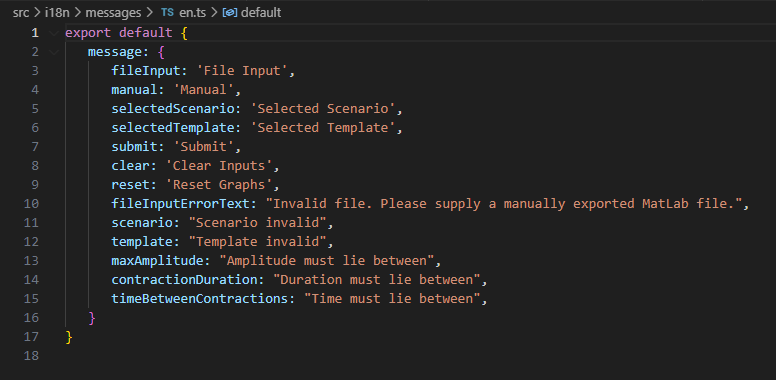
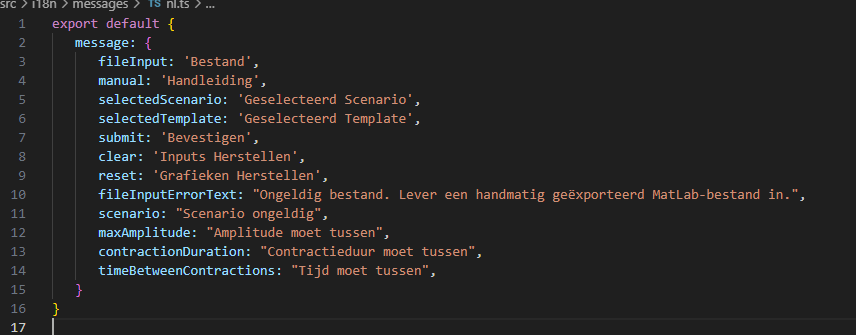
### Sending data to backend

Currently not supported, use something like Axios for example.

### Language support

To add new languages go to ‘src > i18n > Index.ts’. Make sure to also create a new language file under `src/i18n/messages/tr.ts` (for example if u wanted to add Turkish).

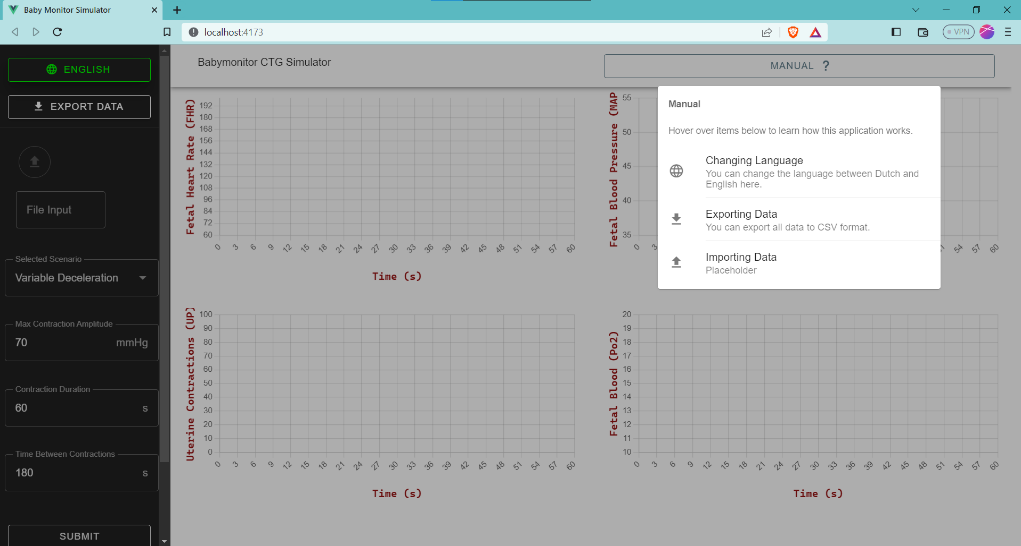
In the individual language files you can add the words:



If a new language is added you will need to make some changes to the Language button, since it just switches between the 2 languages right now, instead of selecting a language (hard coded, but simple drop-down menu will solve this).

### Manual

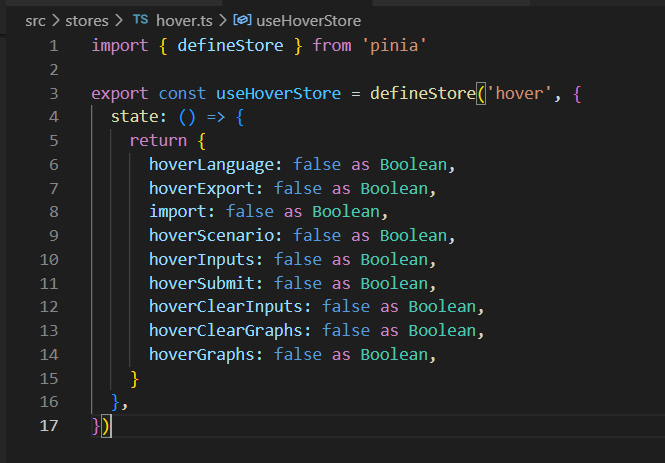
The manual uses an overlay to highlight the items you hover over in the manual.



If I hover over Changing language, the language button on the left highlights in green. You can add items to this in the ‘components > Overlay.vue’ file



You can add cases to the switch to add extra items. Also add the items to the ‘hoverStore’, the store makes the items globally available so that you can use the hover function on them when you add them to the v-list items.



### Front to Back API connection

The backend has a SSE route over which users can receive the data for each graph in the frontend. The frontend simply connects to this route and starts parsing this data which is in JSON format.

A screen shot of a computer program

Description automatically generated with low confidence

Furthermore, the backend has a settings endpoint through which the frontend can PATCH Its own individual settings.

## 7.2 Back End

### MATLAB wrapper

A screen shot of a computer code

Description automatically generated with low confidence

In this example we first instantiate the MATLAB wrapper class which instantiates the MATLAB engine that connects to a running MATLAB engine if present. Else the engine will instantiate a new MATLAB engine.

Second we give the MATLAB engine the model we want to run.

After that, the model spits out the same diagnostic data to the IDE console.

A picture containing text, screenshot, font

Description automatically generated

Note: in this example we use a synchronous approach, but this MATLAB system is compatible with async.

After the model is done running the code, it will continue by getting the single variable ‘oxcycle ’ and prints it to the console.

|  |  |
| --- | --- |
| IDE console | MATLAB console |
| A screenshot of a computer  Description automatically generated with medium confidence | A white background with black text  Description automatically generated with low confidence |

## 7.3 MATLAB

The MATLAB simulation has a couple of different options, these can be found at the top of the ‘*FMPmodel*’ file:

A screenshot of a computer program

Description automatically generated with medium confidence

Do note that there is an ‘*FMPmodel2*’. This one, however, does not work properly and should be deleted.

Most of these variables should be kept at their current values, the only values that currently need to be manipulated for this project are the ‘*scen*’ variable and the ‘*ncyclemax*’ variables. The *ncyclemax* describes how long the simulation should run (the amount of heartbeats the simulation lasts for). The *scen* var controls the different scenarios that are present, these scenarios are:

A screenshot of a computer program

Description automatically generated with medium confidence0: Normal CTG  
1: Early Decelerations  
2: Late Decelerations  
3: Variable Decelerations

Later in the file is a reference to ’*update\_f*’, the file which will handle further calculations for the fetus. This file will then , based on the selected ‘*scen*’, further redirect the calculations to one more file:

‘*ed.m*’ for *scen = 1*  
‘*ld.m*’ for *scen = 2*‘*vd.m*’ for *scen = 3*

It is these files that will do most of the calculations for the graphs.

### Optimizing:

Currently the MATLAB model is extremely slow, there is a possibility to optimize the code by using the profiler.

A screenshot of a computer

Description automatically generated

In this example above the *update\_f* function takes the most time, so lets zoom in to that function.

A picture containing text, screenshot, number, font

Description automatically generated

Within *update\_f* line 150 takes the most time, so again lets zoom into that function

A screenshot of a computer

Description automatically generated with medium confidence

This whole function takes about 26 full seconds to execute, so this needs to be optimized a bit without altering the functionality.