INNOVATION LAB FOR WEARABLE AND UBIQUITOUS COMPUTING

MACHINE LEARNING AND DATA ANALYTICS LAB

DEPARTMENT OF COMPUTER SCIENCE

Stress+

Students:
Jonas Schüll

Project partner: Lehrstuhl für Gesundheitspsychologie

Scrum master:
Michael Nissen, Matthias Zürl

Date: August 6, 2020

Contents

1 Introduction

This project aims to develop a tool called *Stress+* that simplifies the existing stress test process. Stress is one of the major problems, it often gets overlooked and plays a negative influence on the overall quality of life. Due to its significant public health concern, the demand for a stress test is increasing rapidly. The basic idea behind this project is to overcome the limitations of existing stress test which requires a lot of time, assistance and effort. *Stress+* is a user-friendly tool that allows the participants to take the stress test with their own devices, anytime, anywhere. Based on the Montreal Imaging Stress Task (MIST), this app sets up a challenging environment that induces the psychological stress level of the participants by allowing them to take a mental arithmetic test which is supported by stress-inducing factors like time constraints, scores, feedback etc. The results are shared to doctors which aids them to diagnose and plan any further treatment. The *Stress+* application is also designed for doctors, physiotherapist, sports psychologist, stress counsellors and health insurance organisations to assist their process.

2 Architecture

This chapter describes the architecture of Stress+. The whole architecture was designed in a way that arbitrary stress tests with different modules can be created very easily. It was also taken into account that programming and adding new modules is very simple. Each stress test consists of a pipeline of screens which are shown to the patient successively in the specified order. Additionally, a stress test can contain overlays which are displayed on top of the screens.

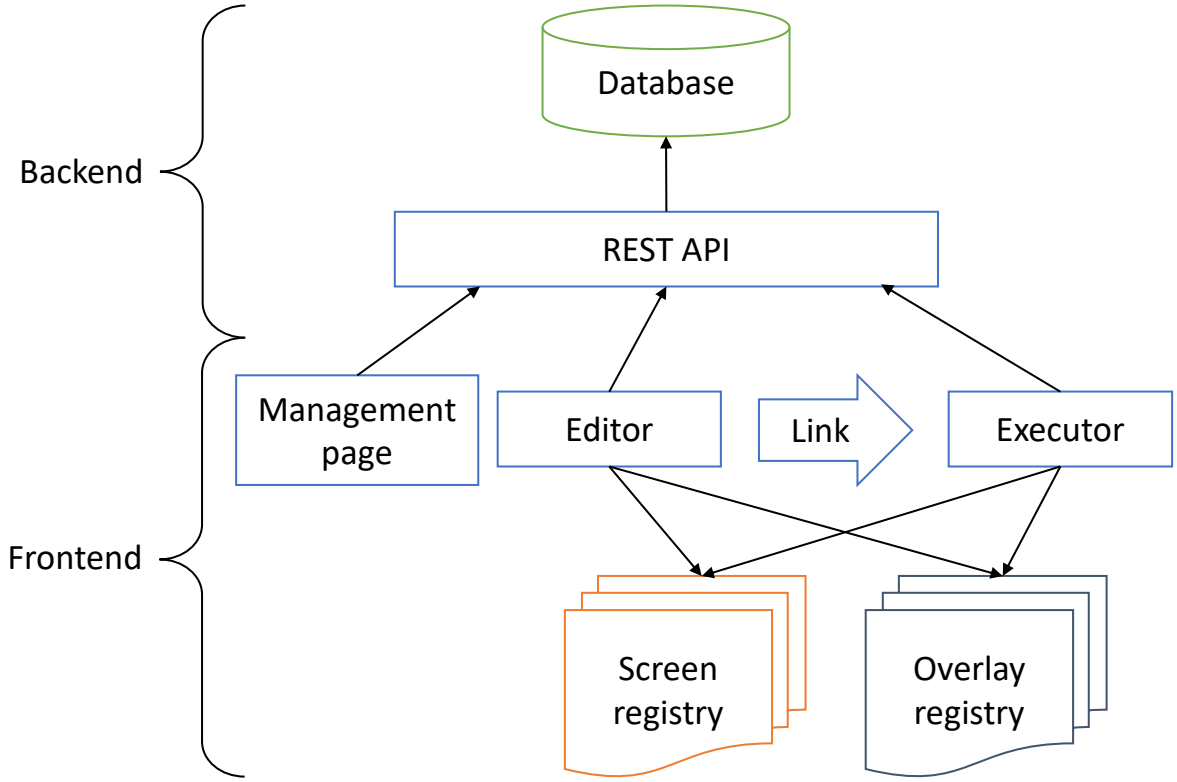

Figure 1: Software architecture

The figure 1 shows the software architecture of Stress+, which developed as a browser-based application and consists of a frontend and a backend. The main frontend components of Stress+ are the *Editor* for creating and editing stress tests and the *Executor* for executing a stress test. On the *Management page* all available stress tests can be managed. To be able to easily add new modules in the future all available screens and overlays are registered in the *Screen registry* and *Overlay registry*. Therefore the *Editor* and *Executor* are developed generically and must query the registries to know which modules are currently present. After saving a stress test in the *Editor* a link is generated, which can be sent to the patient so he can execute the stress test.

The backend is responsible for saving the stress test configurations and the statistics on how the patient performed. Therefore it consists of a REST API, through which the database can be accessed.

2.1 Frontend

The frontend is a Single-page application written in JavaScript with the React framework. The following sections describe the different frontend components in more detail

Screen

The stress test consists of a list of screens that will be displayed successively to the patient. A screen will be displayed fullscreen inside the uses browsers. Each screen has its own settings, which can be adjusted inside the editor. All available screens can be found in chapter 5.

Overlay

The stress test can be equipped with overlays that are displayed on top of the current screen. All overlays are displayed simultaneously during the whole stress pipeline execution, therefore they do not have an order. Each overlay has its own settings, which can be adjusted inside the editor. All available overlays can be found in chapter 6.

Management page

On the management page, all available stress test are displayed. From there you can open a stress test in the editor, delete one or create a new test. Further details can be found in chapter 3.

Editor

The stress tests can be created and edited in the editor. Also, every setting of the screens and overlays can be adjusted within the editor. Each stress test has a unique ID that is generated when it is saved for the first time. With this ID a link is generated that can be sent to patients so they can execute the stress test. From the editor, users can also download all recorded statistics for the current stress test. Further details can be found in chapter 4.

Executor

The executor extracts the unique stress test id from the link and loads the stress test configuration from the backend. Then the executor will display each screen successively to the patient. All Overlays are displayed simultaneously during the whole stress test run. The executor is also responsible to collect records from the screens and persist them in the database. Further details can be found in chapter 7.

2.2 Backend

The backend consists of the Database and the REST API

Database

To save the stress test configurations and the results of a stress test executions a database is used. The data does not have a clear structure, as arbitrary modules can be composed in a stress test. Therefore, the document-oriented NO-SQL database MongoDB is used instead of a relational database.

REST API

The REST API acts as the connection between the frontend and the database. The REST API can be accessed via HTTP and uses JSON documents for transferring the data. The REST API is written in JavaScript on the NodeJS platform and uses the express HTTP framework.

3 Management page

To go to the management page navigate to the home page of Stress+ and click the "Go to the management page" button. A screenshot of the management page can be found in figure 2. All previously saved stress tests are displayed on this page. You can open one of the tests in the editor by clicking on it. Clicking on the trash icon will delete that stress test. To create a new stress test click the "New" button, which will open the editor with a new stress test.

Figure 2: Screenshot of the management page

4 Editor

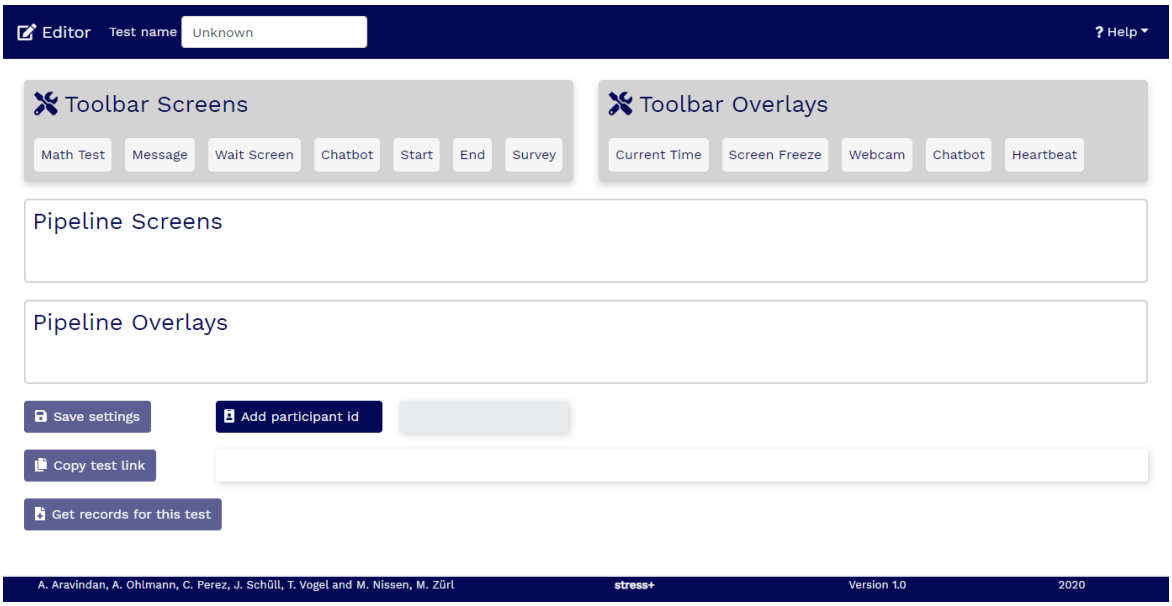

Figure 3: Screenshot of the editor after creating a new stress test

Figure 3 shows a screenshot of the editor with an empty stress test. With the text field on top of the editor, the name of the stress test can be changed. All available screens and overlays are displayed in the corresponding toolbars. From the toolbars, the screen and overlay items can be moved onto the corresponding pipeline with drag and drop. All items in the pipeline will be used when executing this stress test. The stress test can be saved with the "Save settings" button if at least one screen item is present. Each stress test has a unique ID that is generated when it is saved for the first time. With this ID a link is generated that can be sent to patients so they can perform the stress test. This link is displayed next to the "Copy test link" button, which can be clicked to copy the link to the clipboard.
Figure 4 shows a screenshot of the editor with a simple stress test with the name "My stress test". It consists of a "Start", "Math test" and "End" screen and uses the "Webcam" overlay. By clicking the trash icon in the top right corner of pipeline items, the items can be deleted from the pipeline. Each pipeline item has a "Settings" button which can be clicked to open the settings of the corresponding screen or overlay.
Clicking on the "Get records for this rest" button will download all collected test statistics of this test in JSON format. The file will contain an array of test executions. To differentiate test executions a participant id can be added in the text field next to the "Add participant id" label. This participant id will be added to the test link and the test execution result, so executions from different participants can be identified easily.

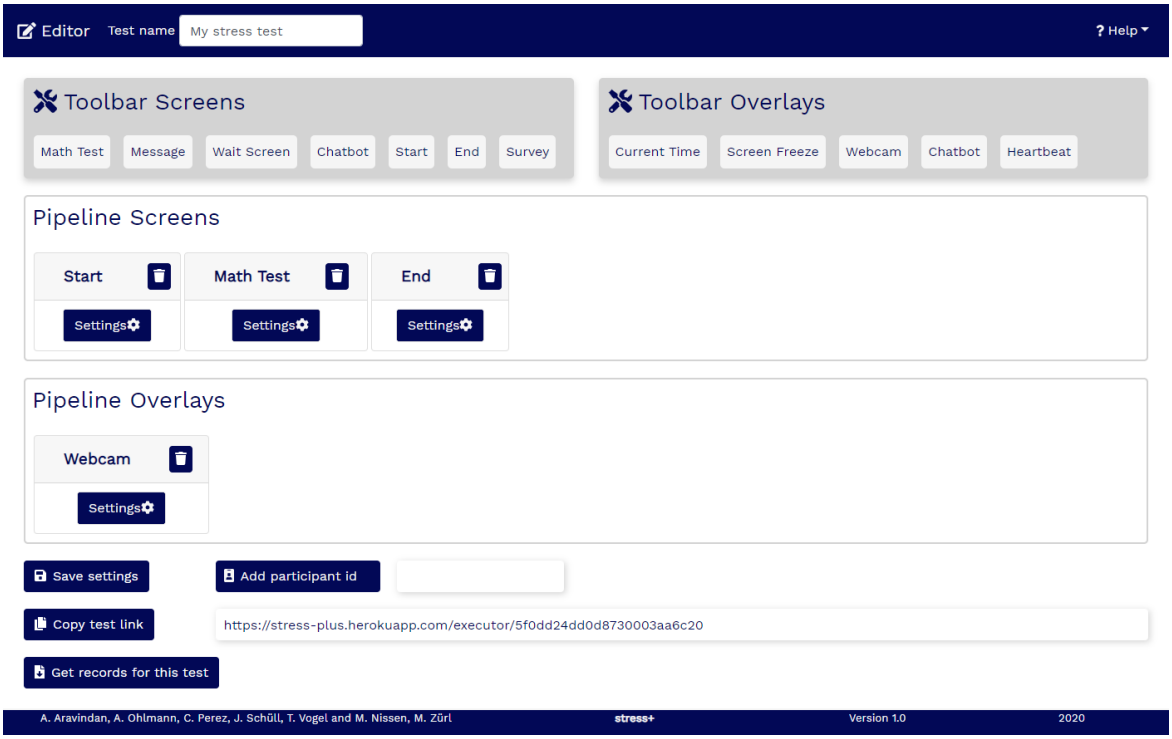

Figure 4: Screenshot of the editor with a simple stress test

5 Screens

This chapter describes all the available screens.

5.1 Math test

The math test screen is the main stress-inducing factor of the stress test. The stress is induced by mathematic questions. For each question, the user has only a limited time to answer the question. If the time is over or the has given the wrong answer, a negative feedback message is displayed. If the user has answered the math question correctly a positive feedback message is displayed. The feedback message is visible for a short amount of time. After this short pause, a new question is displayed. Math questions are taken out of a pool of 40 arithmetic expressions. Before every math test, the array of questions is mixed randomly.

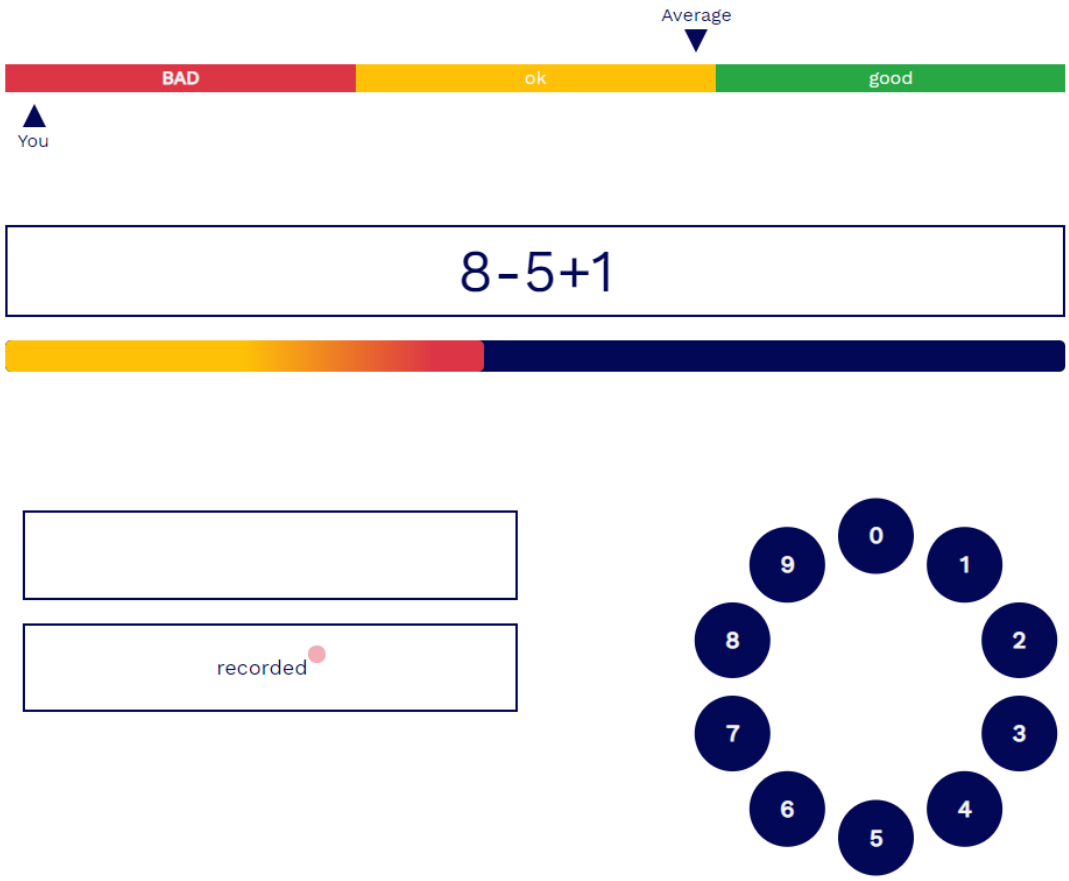

Figure 5: Screenshot of the math test screen

Figure 5 shows a screenshot of the math test screen. At the top, the level bar is displayed, which shows two indicators "You" and "Average". The "You" indicator shows the real performance of the participants. The "Average" indicator is faked. It is increasing if the participant has given a correct answer and is decreasing if the participant has given an incorrect answer. Below the math questions, the progress

bar is displayed. It shows how long the participant has time for the current time. The user must give their answers with the dail pad at the right bottom of the screen. The feedback is given in the blue box on the left of the screen. Now all settings of the math screen are described.

Enable sound If enabled, feedback will be given also via sound. To induce more stress an annoying sound is played while the user has time to answer the math question. Note: users can turn their volume down, so do not rely on the sound to be heard.

Activate Control This option enables the control mode. During control mode, the level bar and progress bar are hidden.

Total test time The total duration in seconds of the math test.

Answer timeout The duration in seconds for answering one math question.

Time between questions The duration in seconds of the short pause after the feedback of the current question has been given to the user.

Difficulty The initial difficulty of the math questions. The table 1 lists all possible difficulty levels. The difficulty level is increased one level if the participant has answered three questions correctly. The level is decreased by one level if the participant has answered three questions incorrectly.

level	number of operands	possible operators	example
0	2	+, −	$7-2$
1	3	+, −	$2-5+6$
2	3	+, −, *	$3*5-9$
3	4	+, −, *	$2*4-2*3$
4	4	+, −, *, /	$3*6/2-2$

Table 1: Difficulty levels for the math test

5.2 Message

The Message screen shows a simple message to the user. The user can go to the next screen by clicking a button on the screen. The title, the message and the button text of this screen can be adjusted.

5.3 Wait Screen

The Wait screen shows a simple message to the user, like the Message screen. But the user has to wait a specific time until the next screen is displayed. The user can not manually go to the next screen. The title, the message and the timeout in seconds of this screen can be adjusted.

5.4 Chatbot

This screen provides a very basic chatbot functionality. Stress test creators can define the chat messages the chatbot will show to the participant. After the chatbot has sent a message to the participant, he can send his answer to the chatbot. After the last message from the chatbot, a "Click to continue" button will be displayed, which will take the participant to the next screen. The answer a participant has given will be added to the statistics of this test execution.

5.5 Start

This screen can be used as the first screen. It displays the Stress+ logo and can show a custom message.

5.6 End

This screen can be used as the last screen. It displayes the Stress+ logo and can show a custom message.
Note: The end screen has no mechanism to go to the next screen. Therefore it should not be placed in the middle of the pipeline. If the end screen is not present a blank page will be shown as the last screen.

5.7 Survey

The survey screen can display other websites inside the stress test. This can be useful to display Google Forms inside a stress test. A button will always be displayed at the top of the screen to navigate to the next screen.

6 Overlays

This chapter describes all the available screens. For easy positioning of the overlays, an overlay can have the optional property `position` which controls the position of the overlay on the screen.

6.1 Current Time

This overlay just displays the current time with hours, minutes and seconds.

6.2 Screen Freeze

The screen freeze overlay will block all user input for a configurable amount of time to induce more stress. The freeze will start after a configurable amount of time. The timer will run as soon as the first screen has started and does not take into account how long a user may wait until he clicks the next button on the Start or Message screen. Therefore it can not be controlled on which screen the freeze happens if a screen with a next button is in the stress test.

6.3 Webcam

The Webcam overlay will display the live recordings of the user's webcam on the screen. To be able to access the webcam the user must give the Stress+ website the permission to do so. Some operating systems might also restrict access to the webcam. It is also generally not possible that two programs access the webcam simultaneously. If the webcam can not be accessed, the stress test will continue without displaying the live recordings of the webcam.

6.4 Chatbot

This is an overlay version of the Chatbot screen, that can be opened and closed via button. For further details consult the Chatbot screen documentation.

6.5 Heartbeat

This overlay will display the heart rate next to a heart image with a bumping animation. The heartrate is completely faked and will slightly increase over time. This overlay can be displayed delayed by specifying a duration in seconds after which it will become visible.

7 Executor

The executor is responsible for handling the execution of a stress test. The id of the stress test to be executed is contained in the link generated by the editor. With this id, the executor loads the stress test configuration from the backend and starts displaying the screens and overlays. The screen's React component will receive the settings and a callback function as properties. The callback function must be called when the current screen is finished, so the executor can show the next screen. All Overlays are displayed simultaneously and each overlay's React component gets its settings as properties.
The executor is also responsible for collecting and saving all records for the statistics. The records are batched together for each screen. They are all sent together to the backend when the screen changes to the next one. Therefore, records for the end screen will not be sent and saved in the database.
Currently, there are three types of records. The math test screen will add records for each answer of the participant. The chatbot screen will add a record containing the messages from the participant. Thirdly, the mouse cursor position will be added to the records once per second.

8 Deployment

The easiest way to deploy Stress+ is using a Docker container. To build the docker image a `Dockerfile` is present in the root directory. The `Dockerfile` will build the frontend and configure the backend to also serve the frontend. Run the following command in the root directory of this repository to build the docker image with the name `stress-plus`.

```
docker build -t stress-plus .
```

After the docker image is built it can be started with:

```
docker run
  --detach
  --name stress-plus
  --publish 80:80
  --env MONGODB_URI=<mongodb-uri>
  stress-plus
```

Stress+ is now running on port 80 of the host machine. By default the backend will listen on port 80 of the container, but this can be changed by setting the `PORT` environment variable. The backend requires a running MongoDB database, therefore you must set the `MONGODB_URI` environment variable. The value should have the following schema:

```
mongodb://<username>:<password>@<host>:<port>/<database-name>
```

For more information on how to run docker containers refer to the docker documentation.

9 Development

9.1 Adding new Screens

Evey screens must be registered in the screen registry to be available. Following properties must be specified when registering a screen:

- The unique name of the screen
- The React component for displaying the screen while executing
- The React component for displaying the settings of the screen inside the editor
- The initial settings of the screen

When a screen is displayed during the stress test execution, its React component will receive two properties. The `settings` properties will contain the settings. The `onFinished` property is a callback function, which must be called when the current screen is finished and the next screen one should be displayed.

9.2 Adding new Overlays

For easy positioning of the overlays, an overlay can have the optional property `position` which controls the position of the overlay on the screen.
Every Overlay must be registered in the overlay registry to be available. Following properties must be specified when registering an overlay:

- The unique name of the overlay
- The React component for displaying the overlay while executing
- The React component for displaying the settings of the overlay inside the editor
- The initial settings of the overlay

When an overlay is displayed during the stress test execution, its React component will receive its settings inside the props.