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: July 30, 2020

Contents

1 Introduction

This is a short introduction to your project.

2 Architecture

This chapter describes the architecture of Stress+. The whole architecture was designed in a way that arbitrary stress tests can be created with the given modules and new modules can be programmed and added very easily. Each stress test consist of a pipeline of screens which are show to the patient successively in the specified order. Additionally a stress test can contain overlays which are displayed on top of the screens.

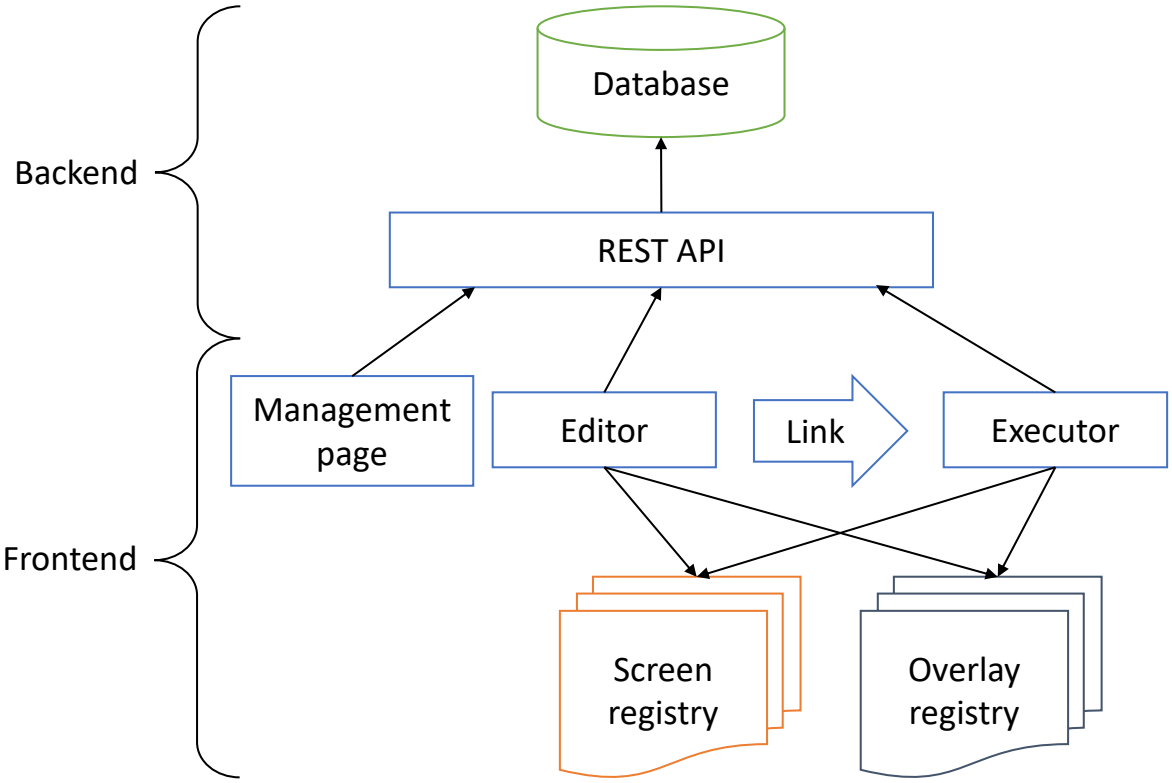

Figure 1: Software architecture

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 availalbe screens and overlays are registered in the *Screen registry* and *Overlay registry*. Therefore the *Editor* and *Executor* are developed in a generic way 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 send 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 the 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 the chapter 6.

Management page

On the management page all availalbe 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 the 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 send 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 the 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 the 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 stress test executions a database is used. Because the data does not have a clear structure, as arbitrary modules can be composed in a stress test, a relational database is not suitable. Instead the document-oriented NO-SQL database MongoDB is used.

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 transfering 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 root url 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 an empty stress test.

Figure 2: Screenshot of the management page

4 Editor

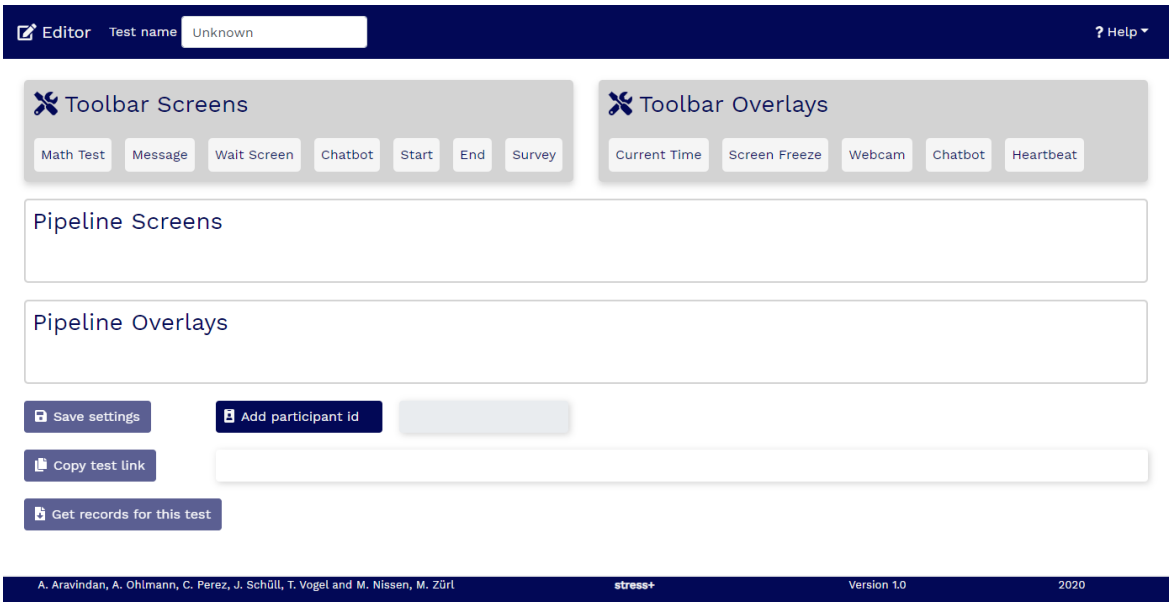

Figure 3: Screenshot of the editor after creating a new stress test

Figure 3 shows a screenshot of the editor after creating a new stress test.
The stress test can be created and edited in the editor with dragging and dropping items from a toolbar to the pipeline. The editor has one toolbar for the screens and one for the overlays. There are also two separate pipelines for the screens and the overlays. Within each item inside a pipeline the user can adjust the settings of the current item. Stress test do have a name that can be changed inside 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 send to patients so they can execute the stress test. The UI also allows the user to download all statistics for the current stress test.

5 Screens

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.

6 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.
Overlays 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.

7 Executor

Screens 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 gets their settings as properties.

8 Frontend

9 Backend