**Project Plan**

***Stress Wearables***

*Manon Peeters-Schaap*

*and Petra Heck*

|  |
| --- |
| **Date** **:** **05-09-2022** |
| **Version** **:** **0.1** |
| **State** **:** **First Draft** |
| **Author** **:** **Group 1** |

**Version History**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Author(s)** | **Changes** | **State** |
| 0.1 | 05-09-2022 | Milan | Testing Strategy Chapter 4. | First Draft. |
| 0.1 | 05-09-2022 | Victoria | Project Organization Chapter 2. | First Draft. |
| 0.1 | 05-09-2022 | Marinda | Project Assignment Chapter 1. | First Draft. |
| 0.1 | 05-09-2022 | Milan | Finances and Risk. Project Budget. | First Draft. |
| 0.1 | 05-09-2022 | Michael | Activities and Time plan | First Draft |
| 0.1 | 05-09-2022 | Ming | Finances and Risk. Risk and Mitigation. | First Draft. |
| 0.2 | 05-09-2022 | Milan | Main Research Question. | Second Draft. |
| 0.2 | 05-09-2022 | Milan | Paragraph numbering. | Second Draft. |
| 0.2 | 05-09-2022 | Michael | Activities and Time plan | Second Draft |

Table of Contents

[1. Project assignment 4](#_Toc125045954)

[1.1 Context 4](#_Toc125045955)

[1.2 Initial Situation 4](#_Toc125045956)

[1.3 Goal of the project 5](#_Toc125045957)

[1.4 Scope and preconditions 5](#_Toc125045958)

[1.4.1 Scope 5](#_Toc125045959)

[1.4.2 Preconditions 5](#_Toc125045960)

[1.5 Strategy 5](#_Toc125045961)

[1.6 Research questions and methodology 6](#_Toc125045962)

[1.6.1 Main research question 6](#_Toc125045963)

[1.6.2 Sub research questions 6](#_Toc125045964)

[1.7 End products 6](#_Toc125045965)

[2. Project organization 8](#_Toc125045966)

[2.1 Stakeholders and team members 8](#_Toc125045967)

[2.2 Communication 9](#_Toc125045968)

[3. Activities and time plan 10](#_Toc125045969)

[3.1 Phases of the project 10](#_Toc125045970)

[3.2 Time plan and milestones 10](#_Toc125045971)

[4. Testing strategy and configuration management 11](#_Toc125045972)

[4.1 Testing strategy 11](#_Toc125045973)

[4.2 Test environment and required resources 12](#_Toc125045974)

[4.3 Configuration management 12](#_Toc125045975)

[5. Finances and risk 13](#_Toc125045976)

[5.1 Project budget 13](#_Toc125045977)

[5.2 Risk and mitigation 13](#_Toc125045978)

[6. Sources 14](#_Toc125045979)

# Project assignment

## Context

The Streaming Wearables and Stress measurement Platform (SWSP) is made for the research project Wearables and Stress. The project is managed by Petra Heck and Manon Peeters-Schaap.

The research project focusses on stress-related health issues and the impact it can have on the lives of people. These health issues can decrease the quality of life and have high social costs. By measuring the stress levels of a person, stress-related health issues can be predicted earlier and circumvented.

The project’s goal is to gain insight into the amount of stress a person is experiencing. This will be calculated through various measurements from wearables devices. The data has to be extracted and normalized, after which it can displayed to an user-end. The data can also be interpreted and used to predict a user’s stress levels. This result is meaningful to health professional, family members of the user and for the user themselves.

## Initial Situation

The SWSP already exists and was created by another group of developers. However, no connection has been implemented between the wearables and the backend, mock-data has been used instead.

The project has been handed over by the previous developers in the following state:

**Frontend**

* A caregiver dashboard: A dashboard that caregivers use to view patient’s stress data.
* An organisation dashboard: A dashboard that organisation admins to manage patients and patient groups.
* A technical health dashboard: A dashboard that shows the system functionality of all microservices, including logs, warnings and errors.

**Backend**

* Caregiver service: The task of this service is to manage the caregivers in the SWSP system
* Feedback service: The service that is responsible for handling feedback on stress measurements.
* Organisation service: The Organisation Service is in charge of managing organizations (adding, updating and removing).
* Patient group service: The Patient Group Service is in charge of managing patient groups (adding, updating and removing). It is also in charge of adding caregivers and patients to patient groups.
* Patient service: The task of this service is to manage the patients in the SWSP system.
* Stress Algorithm service: The service that is responsible for taking in normalized stress data and processing it into processed stress measurements. Currently, only the Heart Rate Variability algorithm is implemented.
* Stress Data Service: The service that is responsible for handling stress data and communicating with the stress measurements database. This service gets processed stress data from the Stress Algorithm Service, stores it in a database and is used by the Caregiver Dashboard to retrieve this stress data.
* Raw data service: A microservice that retrieves the normalized sensor data from it is database.
* Sensor Data service: A microservice that receives the data from the sensors via the mobile app, converts it to normalized data and stores it in it is database.
* Technical Health service: A microservice that monitors the system functionality of all the microservices with a heartbeat and stores their logs, warnings and errors in it is database.

## Goal of the project

The goal of our project is to create a fully functioning prototype. A fully functioning prototype includes the full process from a user wearing a wearable to users being able to see live data on an application (mobile or web). For this to be realized a connection must be made from the wearable to the backend so that data can be collected from the wearable. Once the data from the wearable has been collected, the data needs to be sent to the user interface.

## Scope and preconditions

### Scope

|  |  |
| --- | --- |
| **Inside scope:** | **Outside scope:** |
| 1. **Backend**: Connecting the wearables with the backend, services for patients to see their own data and comment on stress data | 1. Software maintenance after project delivery |
| 1. **Frontend**: Creating a Progressive Web App (PWA) for patients to view data, making dashboard styling unified. | 1. User manuals |
| 1. Improving maintainability of entire platform |  |

### Preconditions

**Deadline**

The project must be delivered by 22 January 2023. A minimum of 52 days has been allocated to work on the end product.

**Programming languages and Frameworks**

The backend of the project will make use of the .Net framework and will be written in C# since the project has already been set up this way by the previous developers who worked on the project.

The front-end for the project will make use of React as the web application framework since it has also already been set up by the previous developers. The frontend will be created using JavaScript, HTML and CSS.

## Strategy

This project will be done with Agile using the SCRUM framework, which will involve discovering requirements and developing solutions through the collaborative effort of self-organizing and cross-functional teams and their customer/end user. Since tests would be running throughout the sprints to ensure the product integrity, the choice to utilise the agile approach was driven by the requirement to ensure that the software is always prepared for deployment.

The SCRUM framework for project management is structured in such a way that feedback is gotten from the stakeholders at the end of each sprint, guaranteeing the flexibility and adaptivity of the development team. Additionally, because every member of the development team is familiar with this organisational structure, greater project concentration is assured.

## Research questions and methodology

### Main research question

How can a functioning prototype visualise data, recorded from a wearable, on an interactive dashboard that is accessible from a mobile device and the web?

### Sub research questions

**Question 1:** How can data be collected from the wearables?

Research method: Literature Study (Library), Document Analysis (Field), Prototyping (Workshop) (HBO-i, 2022)

**Question 2:** What is the best way to visualise the data to the end user?

Research method: Literature Study (Library), Group Brainstorm (Workshop), Prototyping (Workshop) (HBO-i, 2022)

**Question 3:** How can the platform be accessible by a large amount of users at one time?

Research method: Literature Study (Library) and IT architecture sketching (Workshop) (HBO-i, 2022)

**Question 4:** How can the platform be made to be mobile friendly by converting to or making a Progressive Web App (PWA)?

Research method: Literature Study (Library), Prototyping (Workshop) (HBO-i, 2022)

## End products

The final product will consist of a modification of the SWSP’s existing backend and frontend, as well as adding new functionality.

The backend will be modified to process data directly from wearable devices . A new functionality will be created that enables non-organisation users to access wearables data.

A new frontend will be created for non-organisation users to access and view data from wearables. This frontend will be made to be a Progressive Web App (PWA) which enables the app to be downloaded on a mobile device. The existing frontend for the Caregiver dashboard will be modified to be a PWA to allow access from a mobile device. The styling of all dashboards will be modified to look more cohesive.

All existing code will also be modified to improve maintainability.

**Product Breakdown Structure (PBS)**

*Diagram

Description automatically generated*

# Project organization

## Stakeholders and team members

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Role** | **Functions** | **Availability** |
| Petra Heck (p.heck@fontys.nl) | Product Owner | Senior Researcher AI Engineering - Fontys ICT | **Thursdays & Fridays** 1:00 p.m. - 16:00 p.m. |
| Manon Peeters-Schaap (m.peeters@fontys.nl) | Product Owner | Fontys Paramedische Hogeschool – Senior Researcher Wearables & Stress | **Thursdays & Fridays** 1:00 p.m. - 16:00 p.m. |
| Nico Kuipers  (nico.kuijpers@fontys.nl) | School supervisor (Technical) | Coordinator Eindhoven, Course Developer, Teacher | **Wednesday** 9:00 a.m. - 12:00 p.m.  **Thursday** 9:00 a.m. - 16:00 p.m. |
| Jacco Snoeren  (j.snoeren@fontys.nl) | School supervisor (Technical) | Teacher | **Monday** 9:00 a.m. - 16:00 p.m. (at TQ)  **Thursday** 9:00 a.m. - 16:00 p.m. |
| Georgiana Manolache | Semester Coach | Teacher | **Monday** 9:00 a.m. - 16:00 p.m. (at TQ) |
| Milan Koster van Groos  m.kostervangroos@student.fontys.nl | Developer | Software Developer | **Monday** 9:00 a.m. - 16:00 p.m.  **Wednesday** 1:00 p.m. - 16:00 p.m.  **Friday** 9:00 a.m. - 16:00 p.m. |
| Michael Osuntuyi  m.osuntuyi@student.fontys.nl | Developer | Software Developer | **Monday** 9:00 a.m. - 16:00 p.m.  **Wednesday** 1:00 p.m. - 16:00 p.m.  **Friday** 9:00 a.m. - 16:00 p.m. |
| Ming Janssen  ming.janssen@student.fontys.nl | Developer | Software Developer | **Monday** 9:00 a.m. - 16:00 p.m.  **Wednesday** 1:00 p.m. - 16:00 p.m.  **Friday** 9:00 a.m. - 16:00 p.m. |
| Marinda Boshoff  m.boshoff@student.fontys.nl | Developer | Software Developer | **Monday** 9:00 a.m. - 16:00 p.m.  **Wednesday** 1:00 p.m. - 16:00 p.m.  **Friday** 9:00 a.m. - 16:00 p.m. |
| Victoria Fong  v.fong@student.fontys.nl | Developer | Software Developer | **Monday** 9:00 a.m. - 16:00 p.m.  **Wednesday** 1:00 p.m. - 16:00 p.m.  **Friday** 9:00 a.m. - 16:00 p.m. |

## Communication

Communication regarding with the **product owners and the school technical supervisors** is it at least once a week in person at TQ on Thursday's afternoon from 1:00 p.m. to 2:00 p.m. These meetings focus on progress updates and discussions for weekly tasks to showcase to the stakeholders.

Urgent updates and communications throughout the week are done in a separate channel “Stakeholders” in Microsoft Teams to all the stakeholders.

Communication regarding with the **developers** is weekly on Mondays and Thursdays full day and Wednesdays in the morning from 9:00 a.m. to 12:00 p.m. On these days are the stand-ups set to starting at 9:00 a.m. to being able to attend both physically at TQ and online on Microsoft Teams.

These meetings are to discuss tasks each developer will work on, progress and setbacks/troubles ran into.

Communication regarding with the **semester coach** is it at least once a week in person at TQ on Monday’s afternoon from 2:00 p.m. to 2:30 p.m. These meetings are focused on discussing the progress of the group project with guidance of the semester coach for keeping in track of reaching the learning outcomes.

# Activities and time plan

## Phases of the project

Sprint 0: At this stage of the software development cycle, we will determine the current problem for which a solution Is needed and review all materials provided by stakeholders and prior development teams, to acquire a proper understanding of what has been done and what needs to be done.

Sprint 1: we will start to construct a transition strategy or plan in order to assume charge of the project. We will also start to plan out the technical parts of the project for implementation (UML, Architecture diagram, ERD).

Sprint 2-4: Using the documentations from the design software phase as a guide, we will work on the programming of the software solution throughout each sprint to ensure that no developer deviates from the intended path.

To ensure that every item on our product backlog is completed for each sprint, the tasks, which will involve code refactoring of the previous teams' code, the development of new features and creation of testing, deployment environment, will be split among all developers.

Sprint 5: Creation of final documents (Hand-Over document, complete research document), final software testing(evaluation) and project reflection.

Final delivery: Project application and documentation handover.

## Time plan and milestones

|  |  |  |  |
| --- | --- | --- | --- |
| **Phasing** | **Effort (days)** | **Start date** | **Finish date** |
| Sprint 0 | 7,5 | 29-8-2022 | 18-9-2022 |
| Sprint 1 | 7,5 | 19-9-2022 | 9-10-2022 |
| Sprint 2 | 7,5 | 10-10-2022 | 6-11-2022 |
| Sprint 3 | 7,5 | 7-11-2022 | 27-11-2022 |
| Sprint 4 | 7,5 | 28-11-2022 | 18-12-2022 |
| Sprint 5 | 7,5 | 19-12-2022 | 15-1-2023 |
| Final delivery | 7,5 | 22-1-2023 |  |

**Total Effort:** 52.5 days

# Testing strategy and configuration management

## Testing strategy

**Unit Testing**

Unit tests is a method where individual units of code are tested. Unit tests should be limited to one class.

**Component Testing**

This testing strategy mocks clients of the service. By sending valid and invalid requests to a REST service, we can verify that the service returns the intended responses.

This is especially challenging in a microservice architecture, as we wish to test the microservice in isolation. Service virtualisation tools may be used to simulate other microservices and simplify the test environment.

**End-To-End Testing or System Testing**

The goal of end-to-end testing is to simulate a real user experience from start to finish as they navigate through the application. It does not only validate the UI elements, but also ensures that the sub-systems work and behave as expected.

While the process is often slow and high maintenance, end-to-end testing offers many benefits. It ensures the UI is in line the with the systems behind it and verifies that all microservices and working together correctly in a business requirement context.

**Security Testing**

Static application security testing, or SAST, can be used to test an application for security vulnerabilities. SAST scans each line of code but has shortcomings when it comes to microservices. It does not test for proper communication between microservices.

This can potentially be solved by end-to-end DAST testing, or dynamic application security testing.

**Quality Testing**

Code quality can be tested using a static code analysis tool such as SonarQube or SonarCloud. Quality Testing can be used to ensure a level test coverage and can be integrated into the CI/CD pipeline.

**Performance Testing**

*Performance testing gathers all the tests that verify an application’s speed, robustness, reliability, and correct sizing. It examines several indicators such as a browser, page and network response times, server query processing time, number of acceptable concurrent users architected, CPU memory consumption, and number/type of errors which may be encountered when using an application.*

(Performance testing, best practices, metrics & more, n.d.)

## Test environment and required resources

We plan to use the CI/CD environment within GitHub actions for DTAP or Development, Testing, Acceptance and Production. In addition, SonarCloud can be used for quality and security testing.

Postman may be used for component testing and cypress may be used for end-to-end testing.

**Continuous Integration (CI)**

When a feature has been completed, a pull request (PR) is created which another developer will review. Once the PR is approved a merge into the development/main branch is done which will trigger the CI pipeline. The CI pipeline will build, run tests and push a new docker image into the container registry.

**Continuous Deployment (CD)**

As per discussion with the stakeholders, the project will not be deployed until the end of the semester. At this point, the project will be deployed to a company of their choosing. In the meantime, the project will be run on local machines.

## Configuration management

Within GitHub an organisation will be used to store all repositories. Each microservices and front-end client will be given its own repository with its own pipeline.

**Branching Strategy**

The branching strategy used is Git Feature. Git Feature is a git branching model that uses feature branches and multiple primary branches. All feature development should take places in a dedicated branch instead of the main branch. (Atlassian, 2022)

1. A development branch is created from the main branch.
2. For all user stories, feature branches are created from the development branch.
3. When a feature is completed and tested and the pull request (PR) approved, the feature branch is merged into the development branch.
4. Then a release branch is created from the development branch and merged into the main branch.

# Finances and risk

## Project budget

We will attempt to limit the project budget as possible by using facilities provided by Fontys, such as Azure Active Directory, as well as free services such as GitHub Actions.

If a budget is required, the possibilities will be discussed with the stakeholders. If it requires contribution from the group members, every member needs to agree and the cost will be split evenly.

## Risk and mitigation

|  |  |  |
| --- | --- | --- |
| **Risk** | **Prevention activities** | **Mitigation activities** |
| Illness or unavailability. | N/A. | Message the project group as soon as possible and discuss what can be done furthermore |
| Delay in travel time. | If with public transport, check the apps in advance to notify the project group. | Stand-up can be rescheduled. If the delay is severe (1 hour+), the available members can hold a stand-up and inform the unavailable group member of the results. |
| Deadlines not met, | Using Scrum gives us an early indication when it is unlikely the sprint goals will be achieved. | Notify all the stakeholders of the concerns. Discuss whether the scope can be changed. |
| No Fontys facilities such as classroom (Wi-Fi at school), ISSD, Smartboard. | N/A. | Work using MS Teams in case of no access to the buildings of Fontys. |
| Demo version of the project fails/breaks right before or during the presentation. | Make a clip of the features planned of that sprint and what we worked on. Put this clip in a general space so all the stakeholders can access this and see the previous clip as well. | Do not make last minute changes to the master branch before the demo. |
| Dropout. | Inform the stakeholders of the status of the points of concern every time there is an update. | Inform the stakeholders about the dropout and discuss whether the scope is still in reach. |

# Sources

Atlassian. (2022). Git Feature Branch Workflow. *https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow*.

*Performance testing, best practices, metrics & more*. (n.d.). Retrieved from Tricentis: https://www.tricentis.com/learn/performance-testing