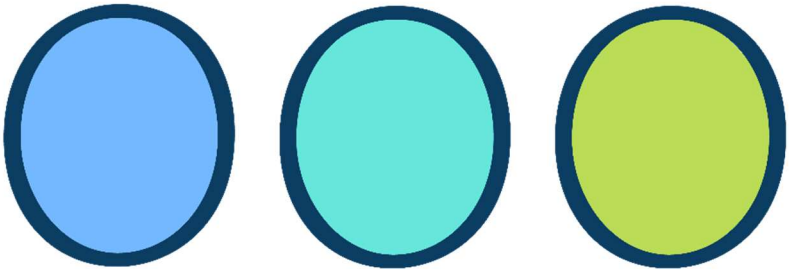
# Eventdoo

## Software Requirement Specification



Bern, 10.12.2019
Version 1. 0

**ESE2019-team 5**

Gillian Cathomas
Wegmatte 11
6460 Altdorf

18-116-624

Vishvaksenan Rasiah
Im Than 1
3150 Schwarzenburg

18-104-992

Lars Ziegler
Stauffacherweg 15
6006 Luzern

15-920-960

Sophie Pfister
Giacomettistrasse 18
3006 Bern

14-118-608

Cyrill Rohrbach
Ischlag 64
3303 Jegenstorf

18-122-002

# Table of Content

1. Introduction ..... 4

1.1 Purpose ..... 4

1.2 Scope ..... 4

1.3 Short Forms ..... 4

1.4 References ..... 5

2 Overall Description ..... 6

2.1 Product Perspective ..... 6

2.1.1 System Interfaces ..... 7

2.1.2 User Interfaces (UI) ..... 7

2.1.3 Hardware Interfaces ..... 7

2.1.4 Software Interfaces ..... 8

2.1.5 Communication Interfaces ..... 8

2.2 Product functions ..... 9

2.3 User Characteristics ..... 10

2.4 Constraints, assumptions and dependencies ..... 10

3. Specific Requirements ..... 11
3.1 Functional Requirements ..... 11
3.1.1 Sign up and log in ..... 11
3.1.2 Update and delete UA ..... 12
3.1.3 Search and request ESs ..... 12
3.1.4 Provide, update and delete ESs ..... 12
3.2 Performance Requirements ..... 13
3.3 Software System Attributes ..... 13
3.3.1 Reliability ..... 13
3.5.2 Availability ..... 13
3.5.3 Security ..... 13
3.5.4 Maintainability ..... 14
3.5.5 Portability ..... 14

# 1. Introduction

## 1.1 Purpose

The purpose of this document is to present a detailed description of Eventdoo, an Ionic/Angular based hybrid application. It will explain the purpose and features of the system, interfaces of the system, what will the system do, the constraints under which it must operate and how the system will react to external stimuli.

## 1.2 Scope

Eventdoo software is an online marketplace for event planning to make it easier. All registered users can provide different kind of services for events (venue, gastronomy, photography, music, entertainment) by creating a service offer. Potential customer can filter those services by category (and subtype), availability, city, price, number of people at the event, and simple text search. When finding a service, they are interested in, they can send a request to the provider. Eventdoo will then connect the provider and the potential customer.

Eventdoo's use is limited to Switzerland and has to be adapted if it's meant to be used internationally.

## 1.3 Short Forms

This document uses the following short forms.

| | |
|---|---|
| BE | backend |
| DB | Database |
| ES | event service |
| FE | Frontend |
| JWT | JSON web token |
| TD | task defintion |
| UA | user account |
| UGL | user guidelines |
| UI | user interface |
| UF | user feedback |

## 1.4 References

| | |
|---|---|
| Git repository | https://github.com/scg-unibe-ch/ese2019-team5 |
| Project description / task definition | https://github.com/scg-unibe-ch/ese2019 |
| Software Requirement Specification | Krazytech, Wikipedia |

# 2 Overall Description

## 2.1 Product Perspective

Any user who wants to make use of or provide ESs has to create an UA. By doing so, the following information about the user is stored in our DB:

- name
- surname
- address
- e-mail-address (which has to be verified)
- hashed version of the password the user defined

By modifying his user profile, the user can store additionally his/her phone number as well as a firm name. A user can add an ES to his/her favorite ones. Also, all ESs requested by the user are stored in our DB and can be shown.

If a user wants to provide an ES he has to create an ES the following information about the provided service must be stored in the DB.

- title
- category: venue, gastronomy, music, entertainment, or photography
- address and perimeter around the address, where the provider is willing to provide his/her/their service
- capacity: the max. number of persons the service can be provided to
- availability: when (on which days) can the service be provided
- a standard price per event/person served/hour (depending on the category)
- a description of the service
- one picture

Optionally, the user can store a subtype of the category (e.g. what kind of food is provided) and any special requirements.

### 2.1.1 System Interfaces

The interface between the user and the software depends on the device on which Eventdoo is run. The user can e.g. use the keyboard to enter text and number inputs, a mouse, touchpad or touchscreen to enter input from select options or to navigate.

All mandatory inputs must either be in the correct format (and eventually match some pattern) or correspond to some predefined value. This is evaluated each time when a user requests the processing of the data. If there is any invalid input, an error message is displayed in the UI as a UF.

The interface between FE and BE is implemented by using http requests. FE application either posts some data to BE, e.g. to create a new UA or ES, or gets some data from it, e.g. ESs, to display it. Data is tested before being sent to BE. FE adapts the UI according to the response.

In BE, there is another interface to the DB. It is implemented in the DbService class. This class contains methods to handle all requests.

### 2.1.2 User Interfaces (UI)

As FE is basically an Ionic/Angular application, the UI consists mainly of Ionic UI-components. It is simple and discreet to ensure clarity, usability, and as a result functionality. The main theming is in white and bright, cool colors. Text is mainly black (except for errors and linkages) and is sometimes replaced by icons.

The UI consists of several pages either providing information or asking the user for information (forms). Only start page combines both functions.
The forms are reactive. They provide instant user feedback, if any input does not match the expected format. Also there are loading animations when data is processed as well as pop ups or toasts to confirm successful completion of requested processing.

### 2.1.3 Hardware Interfaces

Eventdoo can be run in any browser supported by Ionic/Angular applications. This is given for all latest Versions of common Browsers such as Google Chrome, Mozilla Firefox, Safari or Oprah. We discovered limitations when it is run in Microsoft Edge.

By using Ionic Cordova, the application can also be run on mobile Android- or iOS-Devices.

### 2.1.4 Software Interfaces

Following are the technologies used for Eventdoo.

| **Software / Technologies** | **Purpose / Description** |
|---|---|
| Angular 8 | Eventdoo is based on Ionic/Angular (given by TD). Angular enables high portability, speed and performance. |
| CSS | CSS was sometimes used to refine styling of Ionic UI-Components or when using "standard" components. It was included directly in the HTML tags. |
| Email (Nodemailer) | Email is used to communicate with registered users. Nodemailer module is used to send the emails from the administrator's email address to customers. |
| Express | Express is used for all the HTTP communication. Thanks to Express we have a REST API that exposes endpoint to accept HTTP requests. |
| HTML | HTML was mainly used to build the UI. Only few of the logic in FE is implemented in HTML. |
| Ionic 4 | Eventdoo is based on Ionic/Angular (given by TD). The UI mainly consists of Ionic UI-Components, so the main styling is defined by them. |
| JSON Web Tokens | JWT is used for session token, verifying the email address of the user, as well as authenticating the user when he forgot his password. |
| PostgreSQL (ts-postgres) | PostgreSQL is the provider of Eventdoo's DB. It defines the connection between Eventdoo's data and business layer. |
| Typescript | In FE, Typescript was used to implement the logic behind the UI, e.g. to send requests to BE and process the responses. |

### 2.1.5 Communication Interfaces

As specified in 2.1.3, Eventdoo support all latest version of common browsers. Eventdoo works with simple electronic forms for signup, login, creating ESs, modifying existing data or adding search attributes when looking for ESs. There are also several types of UF such as loading animations, displayed error messages and confirmation toasts or pop ups to confirm an action has successfully been completed.

## 2.2 Product functions

Eventdoo mainly consists of the following tasks:

- **Connecting people:** This is the main purpose of Eventdoo. It was developed to connect providers of ESs with potential customers. This is realized by email. When a customer is interested in some ES, he / she can send e request to the provider. The provider then gets an email with the information about the customer's event and his / her email address to get in touch.

- **Authorization and authentication**: Non-registered/non-authenticated users can only search and view ESs. They can neither provide nor request them. To do so, they must sign up (create a UA), verify their email address and authenticate themselves by logging in (entering their password).
  Authenticated users can additionally request ESs, add them to their favorite ones, report them to the administrators (when they are violating the UGL) or provide services themselves. Also, they can update their UA and ESs, they provide (with some restrictions) or delete them.

- **Data managing**: All information from UAs and ESs are stored in the DB. They can be updated or deleted by any user authenticated as authorized. All ES requests (per user) are stored as well as any user's favorite ESs.

- **Filter-based searching**: To find ESs more efficiently, user can filter them according to various criteria. Text search is also available.

## 2.3 User Characteristics

Eventdoo was developed for people who are used to using the internet. All users should have competence in dealing with digital media. They should be mature enough to check their email regularly as this is how Eventdoo connects people.

Users using Eventdoo as customers should be able to use search functions (filters) and buy things online (e.g. clothes, flights, holidays). A typical customer is a person or a group of persons (or also firms) who are planning an (big) event, e.g. a wedding, a birthday party or corporate party, which requires some planning. Eventdoo is meant to make this easier as all services can be found on one web page.

User using Eventdoo as providers should additionally be able to use digital media to advertise their ESs. Also, we expect all users to check their emails regularly as this is how we connect people. Typical providers are persons and firms who work in event business or do so as a hobby. They have to stick to Switzerland's labor laws – e.g. all people working during ESs must be older than 16 years old and people under 18 years old are not allowed to work after 10PM.

There is no difference between UAs of customers and clients to enable to switch easily between roles. They can all be providers and users at the same time.

## 2.4 Constraints, assumptions and dependencies

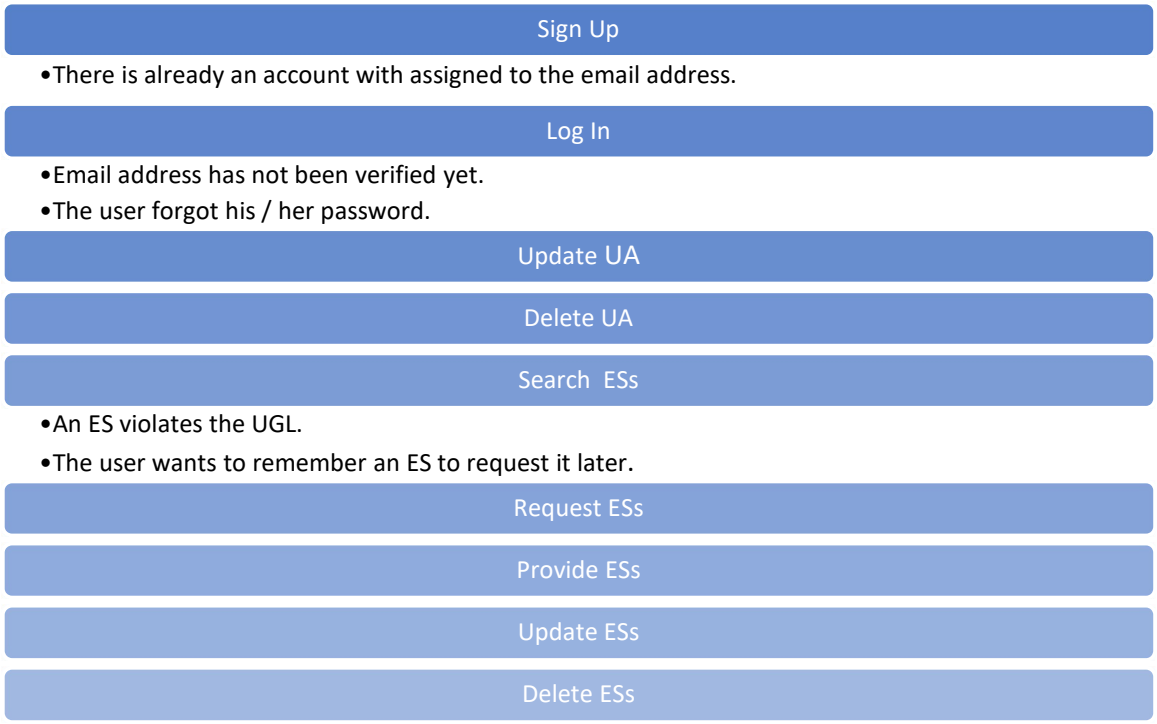
As Eventdoo can be run in any common web browser supported by Ionic/Angular applications, it can be run in almost all operating environments and on almost all devices, given a supported web browser is installed on the device. Therefore, Eventdoo can be used by almost all people. But as we just mentioned above, users are expected to be intermediate users of the world wide web. Important constraints regarding the users are privacy and security: Users are only able to allowed data if the are authenticated to be authorized to do so.

Our application can be run on every DB server which supports PostgreSQL as a provider. Therefor the server can run on every operating system supporting PostgreSQL.

# 3. Specific Requirements

## 3.1 Functional Requirements

Functional requirements are defined by use cases. The following diagram shows key scenarios as well as the most important edge scenarios. Those will be discussed afterwards.




### 3.1.1 Sign up and log in

Sign up and log in refer to the functional requirements of authorization and authentication. Users must be authenticated to ensure they are only accessing data they are authorized to. That's why the first step to use Eventdoo is to create an UA by filling in a form (signing up). The input must be valid. Especially, every UA must be assigned to another email address. This must be checked by Eventdoo software. Only if all data is valid, it can be stored in the DB. Also, the email address must be verified by the user before he / she can log in. This is realized by sending an email to the address entered in the signup form.

To authenticate the user, he / she has to login using the email address and a password as defined in the signup process. If the user has not verified the email address yet, he / she cannot log in. If the email got lost, the user can trigger another email.

If the user forgot the password, he / she can request to reset it. An email will be sent to the correspondent email address with a link to a page where the user can set a new password. The user is authenticated by a JWT.

### 3.1.2 Update and delete UA

Updating and deleting UAs are parts of the data managing function of Eventdoo. Once the user is authenticated (has logged in), he / she is authorized to modify or delete his / her UA. This is essential as some data (e.g. the address) can change. Other data cannot not be changed due to practical reasons (e.g. the email address is used to connect providers of ESs with potential customers). The user can also add additional information about him- /herself to the DB. If the user does not want to use Eventdoo anymore, he / she can delete his / her UA. All data related to the user is deleted from the DB then.

### 3.1.3 Search and request ESs

Searching ESs refers to the filter-based searching function of Eventdoo. All users (also non-authenticated ones) can use it. They can filter them according to different criteria: category and subtype, availability, city, price, capacity and text search. Authenticated users can additionally add ESs, they are interested in, to their favorites. Also, they can report ESs if they violate the UGL in their opinion. Then, an administrator will manually check the ES an delete it if he agrees with the user reporting it. But this refers to the data managing function of Eventdoo.

Requesting ESs belongs to the function of connecting people. Only authenticated users can use it. When he / she does so, Eventdoo then sends an email to the provider as well as a confirmation to the possible customer. Also, the request is stored in the DB, so the user can view it later.

### 3.1.4 Provide, update and delete ESs

All those features refer to the data managing function of Eventdoo. Each authenticated user can provide up to 5 ESs. If he wants to provide another one, he has to delete one.

If the provider wants to adapt one of his / her ESs, he she can do so, assuming that these are minor changes. If he / she wants to change the ES completely, the user has to delete the ES and create a new one.

## 3.2 Performance Requirements

The purpose of performance is providing good usability. Therefore, performance is used when…

- loading pages. This is ensured by Angular framework.
- creating and updating data (UAs, ESs). This is ensured by user-friendly forms.
- searching for ESs. This is ensured by suitable, purposeful filters and dynamic search implementation.

Of course, performance also depends on external factors such as bandwidth of the internet connection or the connection to the DB server as well as the device Eventdoo is run on.

## 3.3 Software System Attributes

### 3.3.1 Reliability

Eventdoo does not provide a lot of extraordinary features, but the features implemented are very reliable. If a user does not provoke errors by his acts (e.g. signing up twice with the same email address), the system does not fail – except for external reasons such as lack of internet connection, failure when connecting to the DB server etc.

### 3.5.2 Availability

Eventdoo was created for Switzerland, but it's available worldwide, if a device is connected to the internet. Unfortunately, the bandwidth to the DB server is limited and therefore, performance decreases if a lot of users try to access it at the same time. To mitigate this effect, Eventdoo could be used with additional DB servers (increase redundancy) or the bandwidth could be increased.

### 3.5.3 Security

Sensible data such as user-IDs, passwords etc. are always encrypted before being sent from FE to BE or vice versa. To do so, we are using JWTs which are based on RSA-256 encryption and SHA-256.

Also someone cannot just manually try to get to the profile of a user by entering all the right URL parameters, because Eventdoo works with session tokens.

### 3.5.4 Maintainability

In FE, authentication and hashing are implemented in services which can be accessed by all pages and components. This enables that those functions can easily be modified (e.g. to increase security by changing to a better hashing function). There are also some components (e.g. header) are used by several other pages and can be modified easily without destroying it. Surely, maintainability could be increased if more services (e.g. for http requests) would be implemented.

In BE, there are also services (e.g. dbService) which execute important functions for diverse controllers and increases maintainability. Also, the builder pattern was used several times to e.g. build an UA. This would allow us to add more “types” of objects or to work with inheritance to create different kind of objects (e.g. customer and provider account).

### 3.5.5 Portability

As Eventdoo is an Ionic/Angular-Application, it can already be run on most devices if a common browser is installed. Also, it can easily converted into an Android or iOS application by using Ionic Cordova. Some parts of the UI and logic (e.g. when uploading pictures) have to be adapted/optimized as well as the index.js file in FE has to be modified by adding a proxy.