**Software Requirements Specification for**

**“Sistem informatic distribuit pentru gestionarea activitatii**

**unei asociatii de proprietari”**

**Version 1.0**

**Prepared by Calina Florin, Marcu Andrei, Matei Alexandru, Tolgoi Dragos**

**March 8, 2020**

**Table of Contents**

* **Introduction**
* **Purpose**
* **Product Scope**
* **Intended Audience and Reading Suggestions**
* **Definitions, acronyms, and abbreviations**
* **References**
* **Overall Description**
* **Product Perspective**
* **Product Function**
* **User Classes and Characteristics**
* **Operating Environment**
* **Design and Implementation Constraints**
* **User Documentation**
* **Assumptions and Dependencies**
* **External Interface Requirements**
* **User Interfaces**
* **Hardware Interfaces**
* **Software Interfaces**
* **Communications Interfaces**
* **System Features**
* **Account Login**
* **Account Logout**
* **Admin Add Member**
* **Admin Edit Member**
* **Admin Add Invoice**
* **Admin Generate Reports**
* **Admin Add Pool**
* **Other Non-functional Requirements**
* **Performance Requirements**
* **Safety Requirements**
* **Security Requirements**
* **Software Quality Attributes**
* **Business Rules**
* **Other Requirements**

**1.Introduction**

**1.1 Purpose**

This document describes the requirements for handling the activities of an association of tenants. The system will have a user interface, a database server which will store information about the activity of the association application server.

* **Product Scope**

This software system will be a web application for an association of tenants. Thissystem will be designed to ease the administrators’ management activities and also help the tenants to easily have acces to all their invoices, see how much they have to pay, see their overdue invoices and anything related to their building services.

* **Intended Audience and Reading Suggestions**

This document is for developers, project managers and users.

In the first chapter you will find the purpose and references used to make product.

In the subchapter 2.2 are enumerate the functions realized by the product and o short description of each function, information about the components of each class (tenants, administrator, employees of administration etc.). The attribute and operations of each are presented in subchapter 2.3 through an image.

Chapter 4 describes the functional requirements.

* **Definitions, acronyms, and abbreviations**

This Document was created based on the IEEE template for System Requirement Specification Documents.

N.A – Not available

Font – Arial

Dimension – 11

The document is structured on chapters and each chapter in more subchapters.

The language used for this document is English.

**1.5 References**

IEEE. IEEE Std 830-IEEE Recommended Practice for Software Requirements

<https://github.com/>

<https://www.sourcetreeapp.com/>

<https://angular.io/>

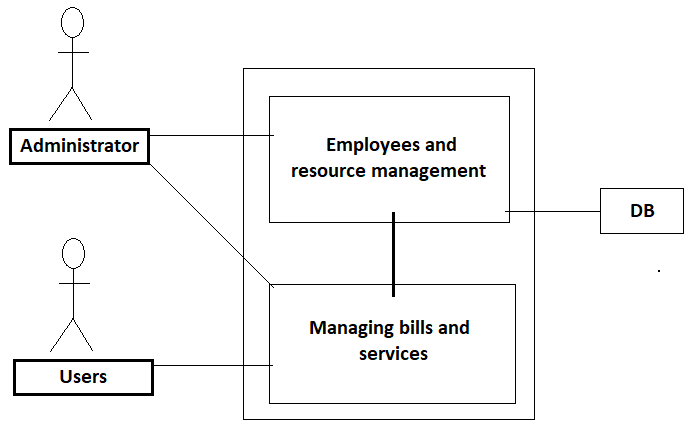
<https://material.angular.io/>

<https://dotnet.microsoft.com/apps/aspnet>

<https://www.microsoft.com/en-us/sql-server/sql-server-downloads>

**2. Overall Description**

**2.1 Product Perspective**



The “Sistem informatic distribuit pentru gestionarea activitatii unei asociatii de proprietari” is a virtually self-contained managing system; however, it will require users to have access to a web browser on their workstation computer. This means that the users of the system do not need to invest in any other software to get the most out of this software system.

**2.2 Product Function**

* **Login**
* **User interface**
* **Administrator/employee interface**
* **Income and expenses report**
* **Poll for different services**
* **Total debt and overdue payments**
* **Monthly report for income**
* **The user can modify the data about his apartment(no. of tenants/ownership etc)**

**2.3 User Classes and Characteristics**

The application users will be employees which will have administrator privileges and tenants which will have regular member accounts.

**2.4 Operating Environment**

As mentioned before the app will require users to have access to a web browser on their workstation computer like Google Chrome(v 82.0) or Mozilla Firefox(v 73.0), Internet Explorer(v 11, 10, 9), Edge(v 80.0.361.62)

**2.5 Design and Implementation Constraints**

Creating a user interface which is both effective and easily navigable will pose a difficult challenge. Other constraints such as a weak web-host and slow internet browsers (“Internet Explorer”) are also worth considering. Our app is meant to be quick and responsive, even when dealing with large groups and transactions, so each feature must be designed and implemented with efficiency in mind.

Backend language: C#

Frontend: HTML, CSS, Typescript

Framework: Angular, ASP.NET Core

**2.6 User Documentation**

NA

**2.7 Assumptions and Dependencies**

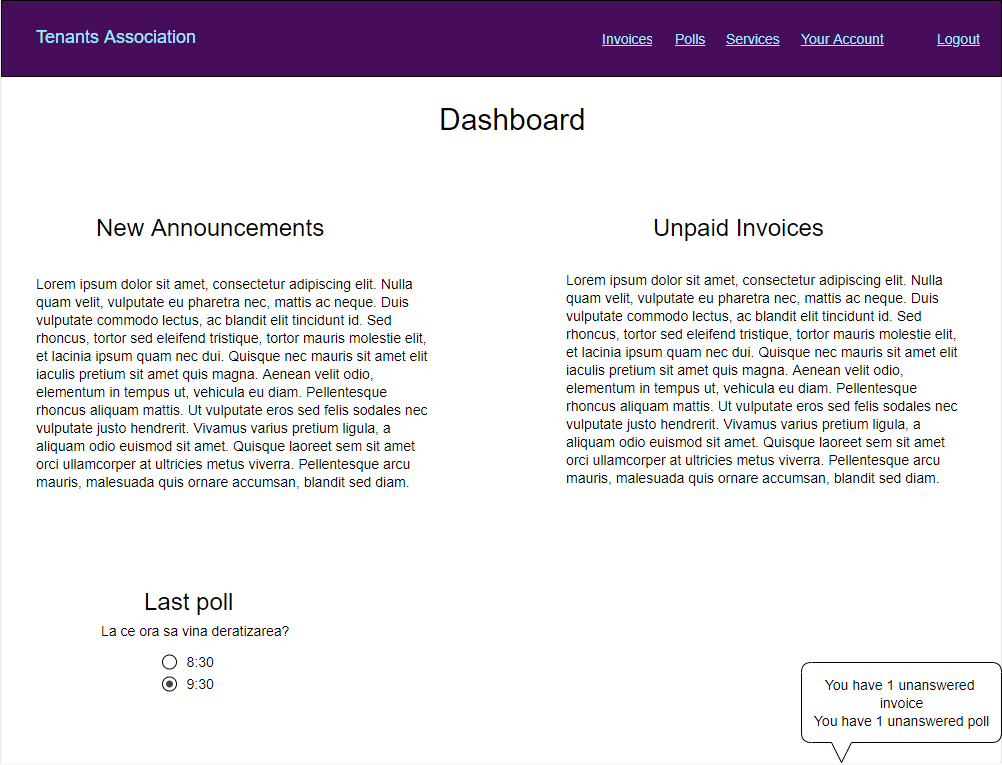
The web application assumes that the user has a computer with an Internet connection and a web browser to access the app. The system may not behave correctly when used with internet browsers other than Firefox and Google Chrome.

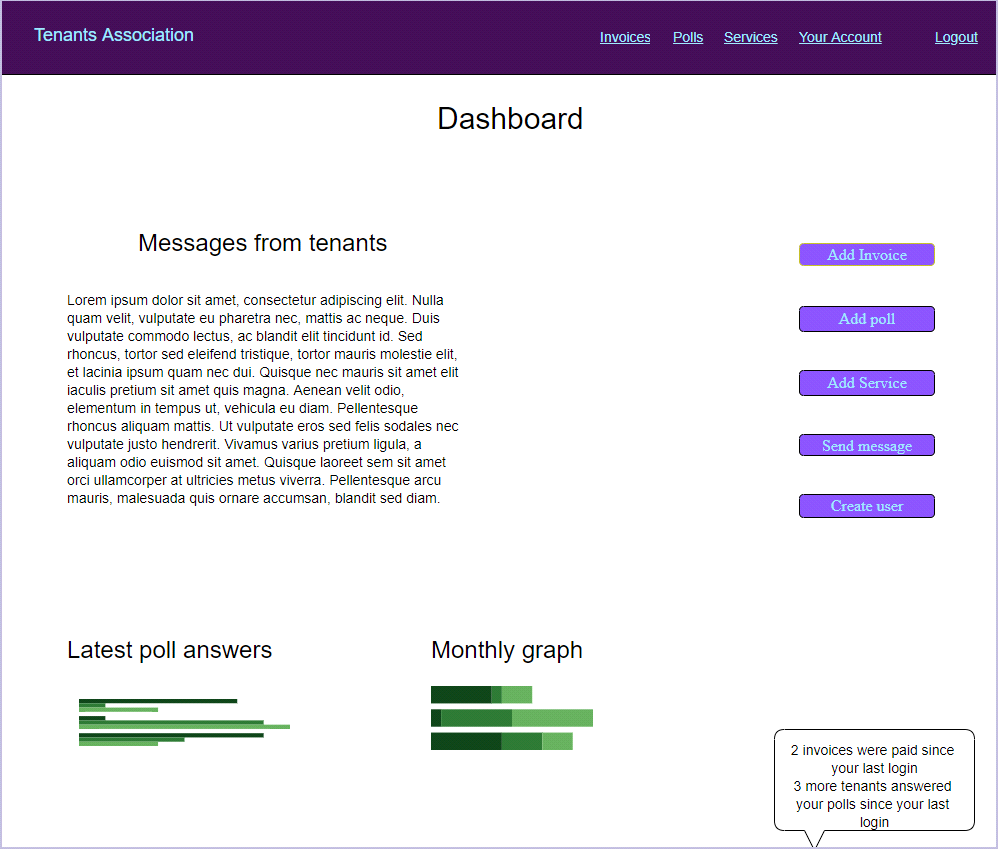
**3. External Interface Requirements**

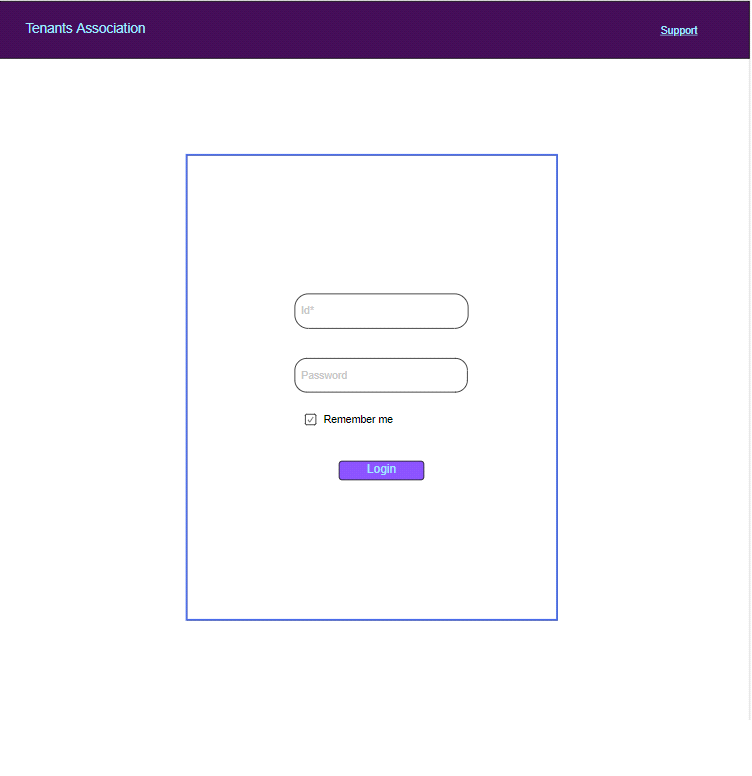
**3.1 User Interfaces**

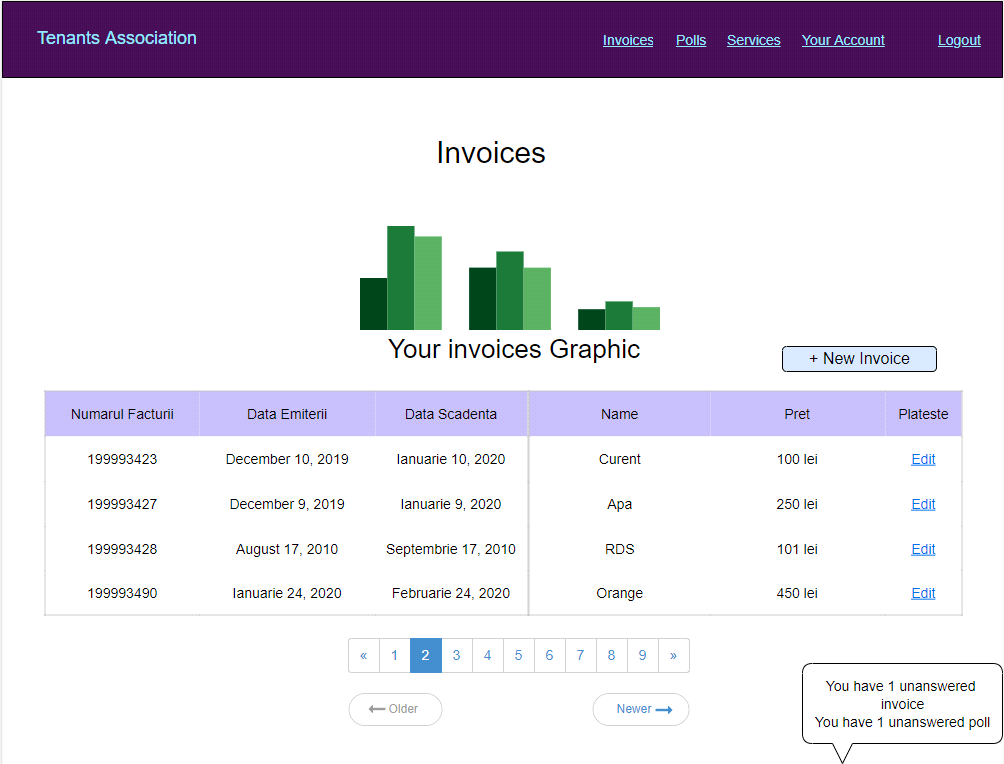
The layout should be as follows:

* The header which displays:
* the logo
* the currently logged user
* the “Logout” and “Manage Account” menu
* The body, which displays
* the menu, immediately under the header
* the content(events already created)
* The footer, which displays
* contact
* copyright









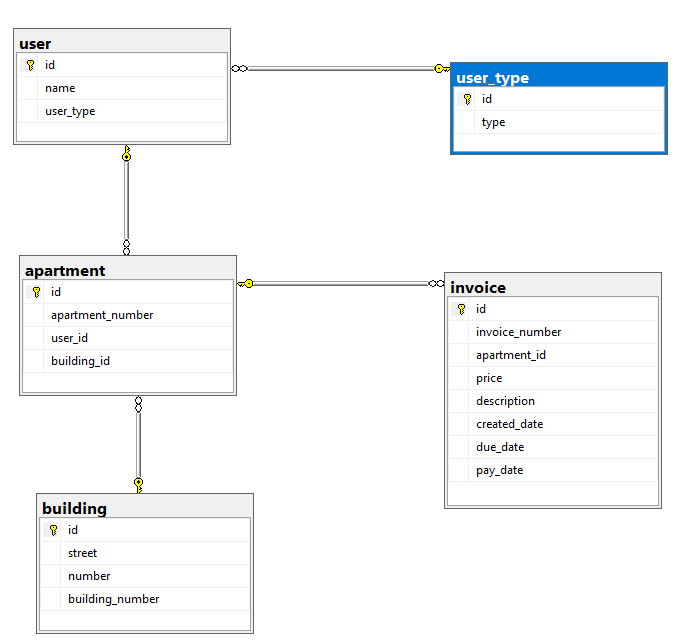
There will be two different user interfaces that will accompany this website: one for the users and one for the administrators.

The administrators will be able to add users and remove them, to add bills , to create pools etc.

The users will be able to see the bills, answer to the pools created by the administrators etc.

**3.2 Hardware Interfaces**

NA

**3.3 Software Interfaces**

The application is connected to a database (SQL Server). The database will communicate with the server, and the server will send the information to the client application. This a mock-up picture of the

Databases.

**3.4 Communications Interfaces**

HTTP will be the communication protocol that our app will use.

**4. System Features**

**4.1 Account Login**

4.1.1 Functional Requirements

• The system shall allow a user to access an account given by the administrators.

• The system will allow the user to change his password

• The system shall confirm the username and password are acceptable.

• The system shall store the information in the database.

All fields are mandatory and are to be written in the text boxes offered by the login interface. If the request passes, the user have access to the app. If the request doesn’t pass the user will be informed with the error and can try again.

**4.2 Account Logout**

4.2.1 Stimulus: Click "Logout" Button

4.2.2 Functional Requirements

•The system shall allow the registered and logged-in user to exit his/her account, so that access to operations requiring a user to be logged in are now disabled.

**4.3 Admin Add Member**

4.3.1 Stimulus: Click "Add Member" Button

4.3.2 Functional Requirements

•The system shall allow a logged-in admin to create new member accounts and new admin accounts.

* Only admins can add new admins.
* The system shall require a username from the logged-in admin

The username field shall be written in a text box offered by the admin interface.

**4.4 Admin Edit User**

4.4.1 Stimulus: Click "Edit User" Button

4.4.2 Functional Requirements

• The system shall allow a logged-in admin to edit a registered member

• The system shall require a username from the logged-in admin

• The system shall display all the information regarding that user with the possibility of modifying data such as user, password, privileges.

• After those modifications the system shall inform the respective member that his information/details have been modified

The username field shall be written in a text box offered by the admin interface.

**4.5 Admin Add Invoice**

4.5.1 Stimulus: Click "Add Invoice" Button

4.5.2 Functional Requirements

• The system shall allow a logged-in admin to add/modify an invoice.

• The system shall attribute it to an user.

The resource name field shall be written in a text box offered by the admin interface.

**4.6 Admin Generate Reports**

4.6.1 Stimulus: Click "Generate Reports" Button

4.6.2 Functional Requirements

* The system shall allow a logged-in admin to generate a report.
* The admin will be able to select between three types of reports:
* Income reports
* Overdue reports
* Payment reports
* Each type of report will display the corresponding data:
* The income reports will display the name of the user and a corresponding due amount.
* The overdue reports will display how much the user have to pay from the previous months.
* The payment reports will display all the payments that a user has to make

**4.7 Admin Create Pool**

4.7.1 Stimulus: Click "Create Pool" Button

4.7.2 Functional Requirements

• The system shall allow a logged-in admin to create a new event: tenants meeting event, upcoming service(pest control etc.)

• The user will have to check in at the current event or vote for a different date.

**5. Other Nonfunctional Requirements**

**5.1 Performance Requirements**

The product shall take initial load time depending on the internet connection strength which also depends on the media-device from which the product is run. The performance shall depend upon hardware components of the client/customer.

**5.2 Safety Requirements**

NA

**5.3 Security Requirements**

The system shall use secure sockets in all transactions that include any confidential customer information. The system shall confirm all transactions with the customer’s web browser.

**5.4 Software Quality Attributes**

Availability: the system is available 24 hours a day.

Portability: a user can log in to the system at any time.

Reliability: the system can be used by multiple users concurrently. Any user can access the system using even a low performance PC.

Rule 1(required) Source code should have comments.

Rule 2(advisory) No identifier name should be reused.

Rule 3(required) All automatic variables shall have been assigned a value before being used

Rule 4(advisory) The increment (++) and decrement (--) operators should not be mixed with other operators in an expression

Rule 5(advisory) Write only one declaration per line

Rule 6(advisory) Use meaningful names for variables

Rule 7(required) All if … else if constructs shall be terminated with an else clause.

Rule 8(required) All constants must be declared with capital letters and underline between words.

Rule 9(required) The operands of a logical && or || shall be primary-expressions.

Rule 10(required) There shall be no unreachable code.