
Software Requirements Specification

for

**Information system for managing municipality
activities**

Version 1.2 approved

Prepared by Penoiu Cristian

Mitrica Alexandru

Mitri Denis

Facultatea de Automatica, Calculatoare si Electronica din Craiova

26-03-2024

Table of Contents

Table of Contents	ii
Revision History	ii
1. Introduction.....	1
1.1 Purpose.....	1
1.2 Document Conventions.....	1
1.3 Intended Audience and Reading Suggestions	1
1.4 Product Scope	1
1.5 References.....	1
2. Overall Description	2
2.1 Product Perspective.....	2
2.2 Product Functions	2
2.3 User Classes and Characteristics	2
2.4 Operating Environment.....	2
2.5 Design and Implementation Constraints	3
2.6 User Documentation	3
2.7 Assumptions and Dependencies	3
3. External Interface Requirements	4
3.1 User Interfaces	5
3.2 Hardware Interfaces	7
3.3 Software Interfaces	7
3.4 Communications Interfaces	7
4. System Features	7
4.1 System Feature 1.....	
4.2 System Feature 2 (and so on).....	
5. Other Nonfunctional Requirements	13
5.1 Performance Requirements	13
5.2 Safety Requirements	13
5.3 Security Requirements	13
5.4 Software Quality Attributes	13
5.5 Business Rules	14
6. Other Requirements	14
Appendix A: Glossary.....	14
Appendix B: Analysis Models	14
Appendix C: To Be Determined List.....	14

Revision History

Name	Date	Reason For Changes	Version
Penoiu, Mitri, Mitrica	17-03-2024	Technical issues	1.0
Penoiu, Mitri, Mitrica	26-03-2024	Technical and implementation changes	1.1

1. Introduction

1.1 Purpose

This document describes the requirements for an information system that manages activities within a municipality (GAP).

1.2 Document Conventions

This document is drafted using the Arial font. Chapter titles are written in bold font Times size 18, while subchapters use bold font Times, size 14. The descriptive text for each chapter or subchapter is set in font size 11. Acronyms used in the document are defined in a table.

Term/Acronym	Definition
GAP	Municipality Activity Management
CRUD	Create, read, update, delete
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext transfer protocol secure

1.3 Intended Audience and Reading Suggestions

This document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers.

1.4 Product Scope

The system to be developed will be a software product built on a client/server architecture to assist the mayor in carrying out the following specific actions:

Establishing the organizational chart of a municipality;
Communicating with citizens through online consultations;
Managing the flow of documents within the municipality.

1.5 References

<https://chat.openai.com>
<https://www2.hawaii.edu/~walbritt/ics211/materials/standard.htm>
<https://app.diagrams.net/>
<https://stackoverflow.com/>
<https://www.javatpoint.com/php-mysql-login-system>
<https://phpfortech.com/blog/admin-approval-for-user-login-in-php/>
<https://www.codingnepalweb.com/category/navigation-bar/>

2. Overall Description

2.1 Product Perspective

The management of municipality activities is an information system that will assist the mayor of a locality in establishing the organizational chart of the institution, in communicating with the citizens of the city, and in managing the document flow within the municipality.

2.2 Product Functions

Organizational chart establishment in two stages:

- Define existing departments and relationships.
- Determine department architecture and assign employees, prioritizing existing staff to avoid lawsuits.

Consultation module:

- Schedule consultations based on population requests.
- Online chat for individual/group consultations.
- Generate public reports for group sessions; private reports for individual consultations.

Document flow management:

- Enable mayor to access, modify, and sign documents electronically.
- Consider storing, retrieving, scanning, and printing documents.

2.3 User Classes and Characteristics

I identified 3 classes of users that can access the app:

Mayor:

The Mayor class represents an individual serving as the elected head of a municipality. As the chief executive officer, the Mayor is responsible for overseeing the administration of the municipality and representing its interests, being the administrator of the system.

Employee:

Description: Represents an employee working within a department.

User:

Description: Accesses and views municipal activities, manages appointments, and schedules new meetings within the municipality's system.

2.4 Operating Environment

The application will run on any computer that supports XAMPP, including operating systems such as Windows 8, 10, 11. To use this application, it is necessary to use XAMPP as a local server package, which includes Apache, MySQL. Users must have XAMPP installed and configured on

their system to run the application. The minimum recommended version is XAMPP 8.0.30. Users must have an internet connection and one of the following web browsers: Google Chrome: Version 80 or later.

2.5 Design and Implementation Constraints

1. Programming Language: PHP 7.4 or newer.
2. Application Server: XAMPP 8.0.30 will be used for development, with versions that support PHP 7.4 or newer.
3. Database Management: phpMyAdmin for MySQL 5.7 or newer, providing support for new database features.
4. User Interface: HTML5, CSS3, and JavaScript ES6 to ensure a modern and interactive experience.
5. IDE: PHPStorm 2023.2.5 or newer.
6. Testing: PHPUnit 9 or newer.

2.6 User Documentation

All details about this application will be presented in this document.

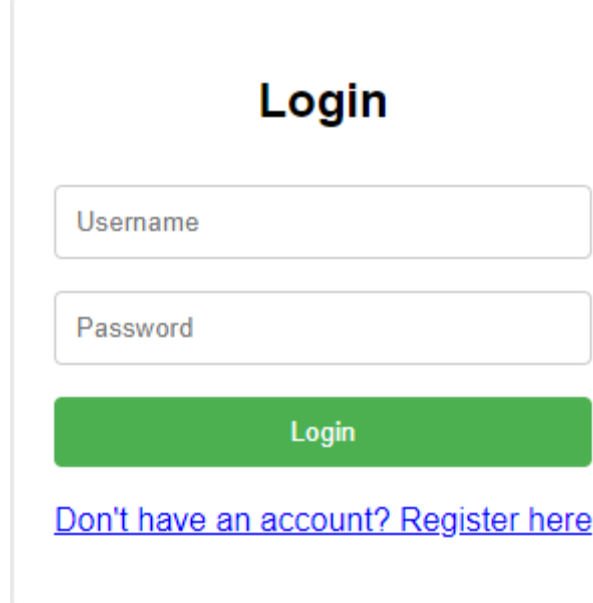
2.7 Assumptions and Dependencies

<not applicable>

3. External Interface Requirements

3.1 User Interfaces

The application will provide intuitive and user-friendly interfaces for various categories of users. A consistent style will be used across all screens of the application, including colors, fonts, and navigation menu. Once the user accesses the web application, they are greeted by the login page, which has 2 fields: one for the username/email and one for the password.



The image shows a login form with a light gray border. At the top, the word "Login" is centered in a bold, dark blue font. Below it are two white input fields with light gray borders. The first field is labeled "Username" in a light gray font. The second field is labeled "Password" in a light gray font. Below these fields is a green rectangular button with the word "Login" in white text. At the bottom of the form, there is a blue hyperlink that reads "Don't have an account? Register here".

If the user doesn't have an account, they can click on "Register Now" and will be redirected to the registration page, where they need to enter details such as: username, email, CNP, password, and password confirmation to verify if the password has been entered twice the same. Two users with the same email or username cannot be created. The CNP is the key for users to create their accounts. If there is an employee of the municipality or another local resident with that CNP in the database and they don't have an account, it means the account will be validated and created; otherwise, they won't be able to create an account, and they will see the corresponding message indicating why they couldn't create an account: either because there is already a user with the entered CNP or because there is no employee with that CNP.

Register

Register

[Already have an account? Login here](#)

After accessing their account, the user will be directed to their respective page based on the role they have.



They can access the features that are created within the page such as: chat online, new schedule, new meeting, edit profile. An example of table of accessing for all the functionalities is this one:

Nume	Prenume	Vârstă	Adresă
Popescu	Ion	35	Str. Primaverii, nr. 10
Ionescu	Maria	28	Str. Libertății, nr. 15

3.2 Hardware Interfaces

<not applicable.>

3.3 Software Interfaces

The connection between the database and server will always be active. This application will include all CRUD operations

3.4 Communications Interfaces

Network communication is carried out through the HTTP/HTTPS protocol for accessing the web interface and interacting with the application server.

4. System Features

4.1 Login

4.1.1 Description and Priority

Allowing users to authenticate in the application.

Priority: High.

4.1.2 Stimulus/Response Sequences

Input: The user accesses the login page and enters the username and password.

Output: The system verifies the entered data. If they are valid, the user is redirected to the corresponding page based on their role. If authentication fails, the system displays an error message.

4.1.3 Functional Requirements

The user enters the username and password.

The user selects the LOGIN button.

REQ-1: There must be a text field for the user to enter the username.

REQ-2: There must be a text field for the user to enter the password.

REQ-3: There must be a button to toggle password visibility.

REQ-4: The password is not displayed, "*" is displayed for each character.

REQ-5: There must be a button for the user to press after entering the data.

REQ-6: If the username is incorrect, the user cannot successfully log in and an appropriate popup message is displayed.

REQ-7: If the password does not match the username, the user cannot successfully log in.

REQ-8: To access the account, the user must enter the username and password.

REQ-9: Each user has a well-defined role: mayor/employee/user.

REQ-10: After successful login, the user will be redirected to the respective page for each role.

4.2 Sign In

4.2.1 Description and Priority

Allowing new users to create an account in the application.

Priority: High.

4.2.2 Stimulus/Response Sequences

Input: The user fills in the necessary registration information: username, email, password, password confirmation, and CNP.

Output: If the information is correct and valid, the account is created, and the user is directed to the login page to log in. Otherwise, the system displays specific error messages.

4.2.3 Functional Requirements

The user enters the username, email, password, password confirmation, and CNP.

The user selects the Register button.

REQ-11: There must be text fields for the user to enter the username, email, and CNP.

REQ-12: There must be two text fields for the user to enter the password and password confirmation; if the passwords do not match, registration cannot be completed.

REQ-13: If the CNP is already associated with a username, registration cannot be completed, and the user will see a corresponding error message.

REQ-14: The system should validate the input data provided by the user during registration, ensuring that all required fields are filled correctly and adhering to any format or length restrictions.

REQ-15: There must be a button for the user to press after entering the data.

REQ-16: If the username/email is not unique, registration cannot be completed.

REQ-17: After registration is completed, the respective user will automatically have the role of user and will be redirected to the login page.

REQ-18: The password must contain at least 8 characters.

REQ-19: The password will be encrypted..

4.3 Sign Out

4.3.1 Description and Priority

Allowing users to log out of the application.

Priority: High.

4.3.2 Stimulus/Response Sequences

Input: The user clicks on the logout button available in the application interface.

Output: The user is logged out and redirected to the login page with a confirmation message of the logout.

4.3.3 Functional Requirements

The user logs out by pressing the "Logout" button.

REQ-20: There must be a button for the user to log out.

REQ-21: After pressing the button, the user is redirected to the login page.

REQ-22: After logout, a confirmation popup message will be displayed.

4.4 User Profile

4.4.1 Description and Priority

Users can view and update their personal information from their profile.

Priority: LOW.

4.4.2 Stimulus/Response Sequences

Input: The user accesses the "My Profile" section of the application and modifies the desired information: phone number, email, password.

Output: The system updates the information in the database and confirms the changes made to the user.

4.4.3 Functional Requirements

The user can view their current profile data.

The user can modify some data.

REQ-23: There must be text fields for updating the phone number and email address.

REQ-24: There must be two text fields for entering the new password and its confirmation; if the passwords do not match, the update cannot be completed.

REQ-25: There must be a save button.

REQ-23: If the new email entered is already registered in the system, the update cannot be completed, and the user will receive a corresponding popup message.

REQ-24: After successfully updating the profile, the user will receive a popup message.

REQ-25: The new password must contain at least 8 characters.

REQ-26: The password will be encrypted.

4.5 Organizational Chart Establishment:

4.5.1 Description and Priority

Create a module where administrators can define and manage departments.

Allow administrators to establish relationships between departments (e.g., hierarchical structure, reporting lines).

Priority: High.

4.5.2 Stimulus/Response Sequences

Input: The administrator defines the department architecture and assigns existing employees to appropriate roles.

Output: The system updates the information and validates the assignments, avoiding personnel conflicts and confirming the changes to the administrator.

4.5.3 Functional Requirements

REQ-27: The departments of the municipality will be chosen from a list of predefined names. New departments can also be added and saved.

REQ-28: Each department will have a graphical representation, and the relationships between departments will be represented by oriented lines annotated with the type of relationship.

REQ-29: The graphical entity representing a department will allow for the addition and visualization of information regarding the individuals who will work within the department and the subordinate relationships between them.

REQ-30: After validation of the information, the system shall update it accordingly, ensuring data accuracy and integrity.

REQ-31: The system should prioritize existing staff assignments to avoid personnel conflicts.

REQ-32: Upon successful update, the system shall confirm the changes to the administrator, providing feedback on the modifications made.

4.6 Schedule Consultations Based on Population Requests:

4.6.1 Description and Priority

Create a scheduling system where users can request consultations.

High priority.

4.6.2 Stimulus/Response Sequences

Input: The user requests a consultation, specifying time and consultant preferences.

Output: The system validates the request and forecasts the scheduling, confirming the consultation appointment to the user.

4.6.3 Functional Requirements

REQ-33: The system shall implement a scheduling system enabling users to request consultations.

REQ-34: Users must be able to specify the desired time and consultant preferences when requesting consultations.

REQ-35: The system shall prioritize the scheduling of consultations based on user requests and population needs.

REQ-36: Upon receiving a consultation request, the system shall validate the input data to ensure accuracy.

REQ-37: The system shall forecast the scheduling of consultations and confirm the appointment details to the user.

4.7 Online Chat for Individual/Group Consultations:

4.7.1 Description and Priority

Develop a real-time chat feature for individual and group consultations, allowing users to interact with consultants or experts.

High priority.

4.7.2 Stimulus/Response Sequences

Input: Users access the online chat to initiate individual or group discussions.

Output: The system opens the chat and facilitates real-time communication between the user and consultant.

4.7.3 Functional Requirements

REQ-38: The system must provide a real-time chat feature.

REQ-39: Users should be able to initiate both individual and group discussions.

REQ-40: The chat feature must support seamless communication between users and consultants.

REQ-41: The chat interface should include features for sending and receiving messages in real-time.

4.8 Generate Public and Private Reports:

4.8.1 Description and Priority

Design functionality to generate public reports for group sessions, summarizing key findings or recommendations. Implement features to generate private reports for individual consultations, ensuring confidentiality and privacy.

High priority.

4.8.2 Stimulus/Response Sequences

Input: Users generate public or private reports for individual or group sessions.

Output: The system generates the requested reports and provides them to the user.

4.8.3 Functional Requirements

REQ-42: The system shall provide functionality to generate public reports for group sessions.

REQ-43: Users must be able to generate private reports for individual consultations.

REQ-44: The system shall ensure confidentiality and privacy of private reports.

REQ-45: Upon user request, the system shall generate the requested reports accurately.

REQ-46: The generated reports shall summarize key findings or recommendations from the sessions.

REQ-47: The system shall provide the generated reports to the user in a timely manner.

4.9 Document Flow Management:

4.9.1 Description and Priority

Develop a secure document management system that allows the mayor and authorized personnel to access, modify, and electronically sign documents. Implement features for storing, retrieving, scanning, and printing documents as needed, ensuring efficient document flow within the system.

High priority.

4.9.2 Stimulus/Response Sequences

Input: The document management system receives requests from the mayor and authorized personnel to access, modify, electronically sign, store, retrieve, scan, or print documents.

Output: The document management system provides access to documents, allows for modification and electronic signing, stores documents securely, retrieves documents upon request, scans documents as needed, and prints documents efficiently.

4.9.3 Functional Requirements

REQ-48: The document management system shall provide functionality for authorized personnel, including the mayor, to access documents.

REQ-49: Authorized personnel must be able to modify and sign documents within the system.

REQ-50: The documents will be stored by categories: informative documents, documents for approval, and documents that can be modified for approval.

REQ-51: Documents can be retrieved by the document name, owner's name, or by searching for all documents in a specific category.

REQ-52: Documents shall be securely stored within the document management system.

REQ-53: Authorized personnel shall be able to retrieve documents from the system upon request.

REQ-54: The system shall have the capability to scan documents as needed.

REQ-55: Documents shall be printed efficiently upon request from authorized personnel.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

Server System Features:

The system on which the application server runs must meet or exceed the following technical specifications to ensure smooth and efficient operation of the services provided:

Processor: Dual-Core 2.0 GHz or faster;

RAM: minimum 2GB;

Available storage space: minimum 5GB for application data storage.

Stable internet connection with a minimum speed of 100 Mbps.

Response Time

The application must be designed to respond to user requests within the following time limits:

Web page loading time: no more than 15 seconds under normal usage conditions;

Report generation: maximum of 30 seconds.

5.2 Safety Requirements

<not applicable.>

5.3 Security Requirements

We will store the username and password in the database.

Passwords will be encrypted in the database.

Only the administrator and the mayor can view employee data.

The application will implement security measures to prevent SQL injection attacks.

5.4 Software Quality Attributes

1. Use descriptive and appropriate names for all identifiers (variables, method names, class names, constants, etc.).
2. Each file has the name of the public class contained within it.
3. Put a single space before every "{".
4. Class names start with an upper case letter.
5. Short comments will be written using "//" ..
6. Use two blank lines to separate each method within a class definition. Use one blank line to separate logical sections of code within a method.
7. Separate all binary operators, such as "+", "-", "*", "/", "%", etc., with a single space. The exception is unary operators, such as "++", "--", unary minus "-", etc, which do not need to be separated with a single space.
8. The code must be properly indented.
9. Lines of code should be kept short, generally less than 80 or 100 characters wide.
10. Each public class is contained in a separate file.

5.5 Business Rules

<not applicable>

6. Other Requirements

Appendix A: Glossary

Appendix B: Analysis Models

Appendix C: To Be Determined List