URS document

User Requirements Specification

**Group members:**

Radolina Petrova (4443527)

Stoil Yonchev (4435567)

Martin Todorov (4491114)

Danilo Gutesa (4334558)

Table of Contents

[Agreements and decision with the client 2](#_Toc96006021)

[Functional & Non-functional requirements 3](#_Toc96006022)

[Use cases 4](#_Toc96006023)

# Agreements and decision with the client

1. Creation of different types of accounts – manager, caretaker, resource planner

* **Manager:** Creation of accounts, manage all the information about the animals and the employees
* **Resource planner:** Creation of schedules and assigning tasks to the caretakers
* **Caretaker:** Viewing the schedules

1. Contents of the feeding timetable

* The time the animal must be fed

1. Contents of breeding timetable

* Time of the year the animals are breedable. Keep track if breeding has been successful and if an offspring has occurred

1. Save account personal information regarding:

* First and last name, address, phone and email, emergency contact information, BSN, unique 5-digit ID
* Contract information: tasks that an employee performs, start date of employment, duration of the contract, FTE

1. Have a search functionality

* Look up employees by name and specialisation
* Get a list of all animals
* Look up feeding timetable of the animals

1. Store information about animals

* Name of the species and the animal itself
* ID of the animal
* Year of and reason for arrival
* Year of and reason for departure
* Location and cage number
* Diet
* Birthdate

# Functional & Non-functional requirements

## Functional requirements:

* CRUD for account management
* CRUD for animal management
* CRUD for employee management
* Authentication and authorisation of the user
* Data logging

## Non-functional requirements:

* Performance
* Scalability

Use cases

UC-01: Manager should be able to create an account

**Pre-condition:** User needs to be logged in manager account

**Actor:** Manager

**Main success scenario:**

1. Actor clicks on a create new account button
2. System requires information about the user the account is being created for
3. Actor fills up the information and confirms
4. System sends a confirmation massage that new account is created

**Extensions:**

4a: Empty fields

.1: System displays an empty field message

.2: Actor fills up the missing information and confirms

.3: System sends a confirmation massage that new account is created

.4: End of use case

4b: Invalid information

.1: System displays invalid information massage

.2: Actor fills up the right information and confirms

.3: System sends a confirmation massage that new account is created

.4: End of use case

## UC-02: Resource planners should be able to create schedules

**Pre-condition:** User needs to be logged in

**Actor:** Resource planner

**Main success scenario:**

1. Actor goes to the feeding timetable of a certain animal
2. Actor assigns a caretaker to feed the animal
3. Actor confirms
4. System saves the information into the caretaker’s schedule

**Extensions:**

2a: There is caretaker assigned to every animal

.1: System shows full schedule

.2: End of use case

3a: There is no available caretaker

.1: System displays no available caretakers

.2: End of use case

## UC-03: Updating account details

**Pre-condition:** User needs to be logged in their account

**Actor:** Manager, Resource-planner, Caretaker

**Main success scenario:**

1. Actor goes to the settings of the account
2. Actor updates the information and confirms
3. System sends a confirmation massage that the information is updated

**Extensions:**

3a: Empty fields

.1: System displays an empty field message

.2: User fills up the missing information and confirms

.3: System sends a confirmation massage that the information is updated

.4: End of use case

3a: New information matches the old one

.1: System displays same data message

.2: User fills up new data and confirms

.3: System sends a confirmation massage that the information is updated

.4: End of use case

# UC-04: Manager should be able to update animal information

**Pre-condition:** User needs to be logged in a manager account

**Actor:** Manager

**Main success scenario:**

1. Actor searches for the animal, which information they want to change
2. System displays the animal and its information
3. Manager selects the information they want to update and confirms
4. System sends a confirmation massage

**Post condition:** Manager has updated the details of an animal

**Extensions:**

5a: Fields are empty

.1: System displays an empty field message

.2: Actor fills up the missing information and confirms

.3: System sends a confirmation massage