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. Create different types of accounts – manager, caretaker, resource planner

* **Manager:** Should be able to create accounts
* **Resource planner:** Should be able to create schedules and assign tasks for the caretakers
* **Caretaker:** Should be able to have a schedule and perform his tasks

1. Contents of the feeding timetable

* Each animal’s diet(food that they eat) and when they should be fed, as well as their favourite food.

1. Contents of breeding timetable

* What 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 task
* Get a list of all animals

1. Keep that of statistics

* How many tickets were sold?
* When were tickets selling more?
* How many tickets were actually used?

1. Store information about animals

* Name and ID
* Year of arrival
* Reason of arrival
* Reason of departure

# Functional & Non-functional requirements

## Functional requirements:

* Implementation of a database that stores account details, information about the animals and the employees and their schedules and timetables
* CRUD for account management
* CRUD for animal management
* CRUD for employee management
* Authentication of the user

## Non-functional requirements:

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. Manager clicks on a create new account button
2. System requires information about the user the account is being created for
3. Manager fills up the information
4. System asks for confirmation
5. Manager confirms

**Post condition:** New account created

**Extensions:**

4a: Empty fields

.1: System displays an empty field message

.2: Manager fills up the missing information

.3: End of use case

4b: Invalid information

.1: System displays invalid information massage

.2: Manager fills up the right information

.3: 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. The user checks the feeding timetable of the animals
2. The user checks if any employee has been assigned to feed the animals
3. The user checks for suitable caretaker
4. The user checks of the caretaker is available for that period of time
5. The user assigns the caretaker to feed the animal
6. The user confirms and saves the information into the caretaker’s schedule

**Post condition:** A schedule has been created

**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:** User

**Main success scenario:**

1. User goes to the settings of the account
2. User updates the information
3. System asks for confirmation
4. User confirms
5. New information is saved into the database

**Post condition:** User account details have been changed

**Extensions:**

3a: Empty fields

1: System displays an empty field message

2: User fills up the missing information

3: End of use case

3a: New information matches the old one

1: System displays same data message

2: User fills up new data

3: End of use case

# UC-04: Caretaker schedule

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

**Actor**: Resource planner

**Main success scenario:**

1. Manager assigns tasks into the caretaker’s schedule
2. System asks for details of the task such as what and when should be done
3. Information saved into DB
4. Information displayed onto caretaker’s schedule

**Post condition:** Caretaker can view his updated schedule

**Extensions:**

4a: Caretaker schedule is full

1: System shows a message informing nothing can be added to that schedule at this time

4b: Invalid information such as setting a task for a previous date

1: System shows a message that a task can’t be assigned to a previous date

# UC-05: Update animal details

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

**Actor:** Manager

**Main success scenario:**

1. Manager searches for an animal by id
2. System displays the animal and its information
3. Manager clicks update name/age/diet/etc
4. System asks user to update specific details
5. System asks user to confirm
6. Animal information updates in the DB

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

**Extensions:**

5a: Fields are empty

1. System displays points towards the empty fields and displays a message they should be
2. Manager inputs data into the empty fields

5b: Information matches previous data

1. System displays a message
2. Manager enters different data