2DV609

PriTask
Requirements Specification

Author1: Armend Azemi

Author2: Adam Nyman

Author3: Chadi Kawaf

Author4: Donald Nti Appiah-Kubi

Author5: Oliver Rimmi

1. Introduction

This document specifies the different system and user requirements for the application “PriTask”. This document aims to explain what the software will do and how it is going to perform. The product's functionality will be described in this document to meet the stakeholder's needs while implementing the software. A priority feature will be provided in this application, unlike the traditional to-do list apps based on a typical list of tasks.

The purpose of the product is that the user may add their tasks and have them organized in such a way that the user knows what tasks are the most important due to the priority sorting algorithm.

2. System-wide Requirements

Stakeholders

- End-users that benefit from the “PriTask” application.
- Developers: responsible for computing and designing solutions.
- Test lead administers accurate testing during the development process.
- The project manager supervises the project procedure to ensure the documentation and the requirements are updated and completed.

Functional requirements

Requirement SYS1: Create a new list.

- A user must be able to create a list. This list will be used to assign tasks to.

Requirement SYS2: Remove a list.

- A user should be able to remove a list.

Requirement SYS3: Create a new task.

- A user should be able to create a new task and assign it to one of the existing lists.

Requirement SYS4: Edit a task.

- A user should be able to edit all of the information of a task (including the priority).

Requirement SYS5: Delete a task.

- A user should be able to delete a specific task.

Requirement SYS6: Priority.

- Tasks should be sorted by priority.

Requirement DB1: Persistence.

- Whenever some changes are made within the program it should automatically save all the changes in a file locally so that the additions/changes are kept when the user re-opens the program.

Non-functional requirements

Non-functional requirement UI1: Simple user interface.

- The user interface should be simple enough so that a user should be able to use all of the functionalities within 15 minutes.

Non-functional requirement SYS7: The application should be lightweight.

- The application should not exceed more than 0.5gb disk storage. This includes the actual application and the saved files from the user.

Non-functional requirement SYS8: OS Compatible.

- The application should run on all operating systems that support the Java environment.

Systematic 'checklist-based' requirements analysis

	Req. SYS1	Req. SYS2	Req. SYS3	Req. SYS4	Req SYS5	Req. SYS6	Req. DB1
Premature design	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Combined requirements	No	Yes	Yes	No	Yes	No	No
Unnecessary requirements	No	No	No	No	No	No	No
Use of non-standard hardware	No	No	No	No	No	No	No
Conformance with business goals	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Requirements ambiguity	No	No	No	No	No	Yes	Yes
Requirements realism	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Requirements testability	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Requirement Classification 'faceted approach'

	Req. SYS1	Req. SYS2	Req. SYS3	Req. SYS4	Req. SYS5	Req. SYS6	Req. DB1
System	Yes	Yes	Yes	Yes	Yes	Yes	Yes
User interface	Yes	Yes	Yes	Yes	Yes	Yes	No
Database	No	No	No	No	No	No	Yes
Communications	No	No	No	No	No	No	No

Validation of requirements "Checklist"

Question	**Answer**
Are the requirements complete?	Yes
Are the requirements consistent?	Yes
Are the requirements comprehensible?	Yes
Are the requirements ambiguous?	Requirement SYS6 (*Priority*) and Requirement DB1 (*Persistence*) are somewhat ambiguous.
Is the requirements document structured?	Yes
Are the requirements traceable?	Yes
Does the requirements document as a whole, or do the individual requirements conform to defined standards?	Yes

Test Cases for the Requirements

Test Case ID	**TC1**
Priority	High
Main Scenario	Create a new list
Pre-condition	Non
Test Case Description	1. The user wants to create a new list. 2. The user presses the “create list” button. 3. The user enters “List” as the name for the list. 4. The user presses the “save” button.
Expected Results	The user was able to successfully add a list to the table.
Status	Passed

Test Case ID	**TC2**
Priority	High
Main Scenario	Remove a list
Pre-condition	A list must exist within the system
Test Case Description	1. The user wants to delete a specific list. 2. The user selects the list. 3. The user presses the “delete list” button. 4. The user confirms the deletion by pressing the “yes” button.
Expected Results	The user was able to successfully remove the list from the table.
Status	Passed

Test Case ID	**TC3**
Priority	High
Main Scenario	Create a new task and add it to a list.
Pre-condition	A list must exist within the system
Test Case Description	1. The user wants to add a new task. 2. The user selects a list to add the task to. 3. The user presses the “create task” button. 4. The user presses the “save” button.
Expected Results	The user was able to successfully add a task to a specific list to the table.
Status	Passed

Test Case ID	**TC4**
Priority	High
Main Scenario	Edit a task name
Pre-condition	A list and a task must exist within the system
Test Case Description	1. The user wants to edit an existing task 2. The user selects the list that corresponds to the task that the user wants to edit. 3. The user selects the task. 4. The user presses the “edit task” button. 5. The user edits the task name to “task”. 6. The user presses the “save” button.
Expected Results	The task name was changed to the new value “task”.
Status	Passed

Test Case ID	**TC5**
Priority	High
Main Scenario	Delete a task
Pre-condition	A list and a task must exist within the system
Test Case Description	1. The user chooses the list in which the task is located 2. The user selects the desired task 3. The user presses the delete button 4. The user confirms the deletion by pressing the “yes” button.
Expected Results	The task was successfully removed from the table.
Status	Passed

Test Case ID	**TC6**
Priority	High
Main Scenario	The tasks are sorted correctly by their priority
Pre-condition	A list contains two tasks with the same priory values.
Test Case Description	1. The user selects the first task. 2. The user presses the “Edit Task” button. 3. The user enters a new priority value. 4. The user presses the “Save” button.
Expected Results	The tasks were successfully sorted by their priority values.
Status	Passed

Test Case ID	**TC7**
Priority	Medium
Main Scenario	The lists and tasks persist through application shutdown.
Pre-condition	The application should be running with one list which contains a task.
Test Case Description	1. The user shuts down the application. 2. The user starts the application.
Expected Results	The lists and tasks are successfully displayed after restart.
Status	Passed

3. System Interfaces

3.1. User Interfaces

There are six user interfaces to be implemented. The main interface is going to hold the view of all the existing priority lists and the existing tasks within the list. The main interface is also going to allow the user to select the options to create, delete and edit for both tasks and priority lists.

The second user interface holds the list creation. This interface allows the user to select the name of the list, cancel the creation and confirm creation.

The third user interface holds the task creation. This interface allows the user to select priority, name, deadline, cancel creation and confirm creation.

The fourth user interface is with regard to the deletion of a list. The user can confirm or decline the deletion of the list.

The fifth user interface is with regard to the deletion of a task. The user can confirm or decline the deletion of the task.

The sixth user interface is with regard to the editing of a list. The user can confirm or decline the editing to the list

The seventh user interface is with regard to the editing of a task. The user can confirm or decline the editing to the task.

3.1.1. Look & Feel

The user interfaces should use a color theme which consists of white and a shade of green. These colors were chosen in order to give the user a calming experience when looking at the tasks. The interfaces should also be interactive, which means that whenever the user hovers over a button gets highlighted. Additionally, interfaces other than the main should be presented by replacing the main window.

3.1.2. Layout and Navigation Requirements

The visualization of the existing lists should be grouped on the left half of the main user interface. And on the left hand side of the user interface the table of the existing lists are shown. In the center of the main user interface, the existing tasks will be shown in a sorted order by their respective priority values. The right side of the window contains the task information about the currently selected task.

Below the list table, on the left, we find the “Create List” and “Delete List” buttons. And below the task list, which is located in the center of the main interface window, we find the “Create Task” button. The right side of the main interface window contains the selected task information and holds the buttons for “Edit Task” and “Delete Task”.

3.1.3. Consistency

Buttons which have identical functionality should have a specific look. Hence, the user will be able to identify functionality based on the look of a button. The placement of buttons which have identical functionality should also be placed at consistent places. e.g., the delete task buttons will all look the same and they will also be placed at the same place on the object representing a task.

In cases where the user has to input some data/text there will always be a label which explains what the text field below represents. That way the user will always know where to look when they feel uncertain.

3.1.4. User Personalization & Customization Requirements

There are no requirements regarding the personalization of the application. Hence, the users won't be able to customize any visual aspects of the application.

3.2. Interfaces to External Systems or Devices

3.2.1. Software Interfaces

The host computer must be able to run a Java application.

3.2.2. Hardware Interfaces

There are no specific hardware interface requirements.

3.2.3. Communications Interfaces

This application is monolithic and thus does not use any communication interfaces.

4. Business Rules

The current iteration of PriTask is not in need of any types of business rules.

5. System Constraints

PriTask is a desktop application, while developing this application some constraints should be noted, in developing the application, the most popular and common desktop application implementation languages are java and python. GitLab and a suitable integrated development environment (IDE) will be used to implement and share source code with the team developers. Third party applications such as Gson will be used for the persistence part of the application and Maven to enable the use of Gson.

6. Use-Cases

In this chapter the use cases for this application will be discussed. All the most essential use cases have both been stated below as proper use cases. An overview of the system's use cases can also be seen in the diagram below (see figure 1).

From these use cases a class diagram has also been created (see figure 2). The idea of this diagram is to give a high level overview of what the system and its hierarchy should look like.

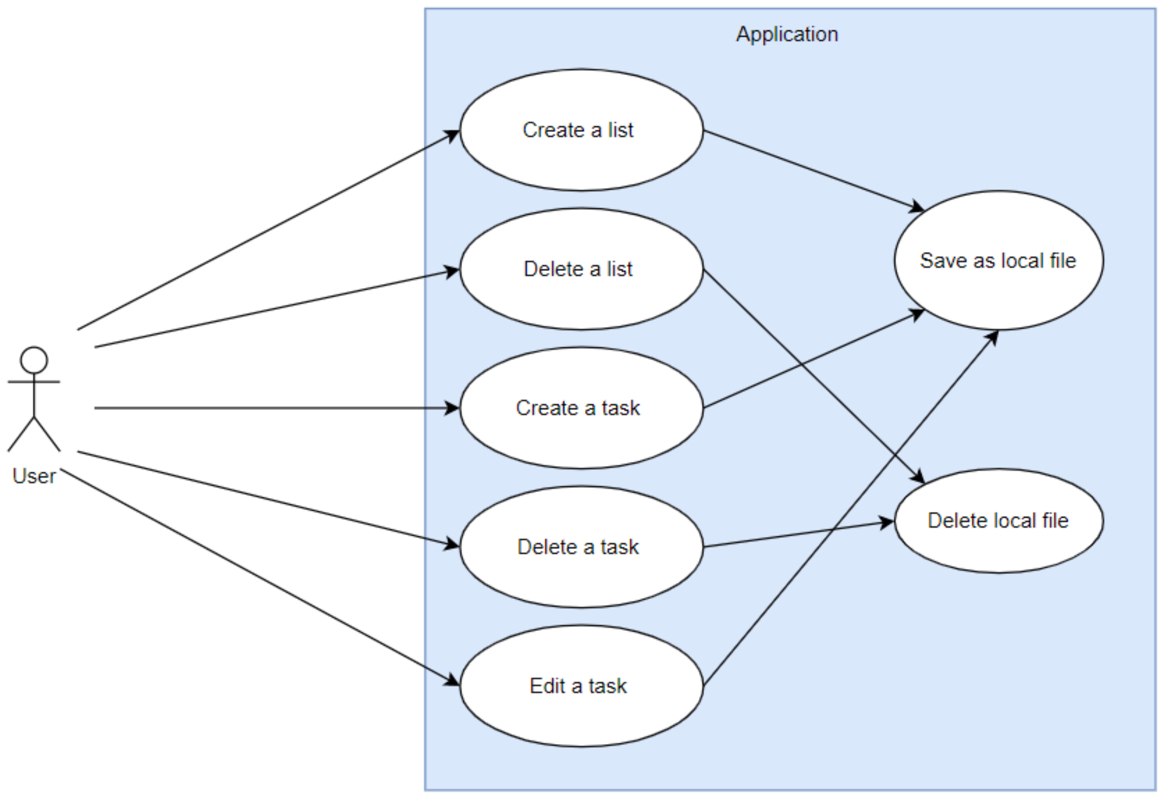

Figure 1: *Use case diagram over all of the most essential use cases of the application.*

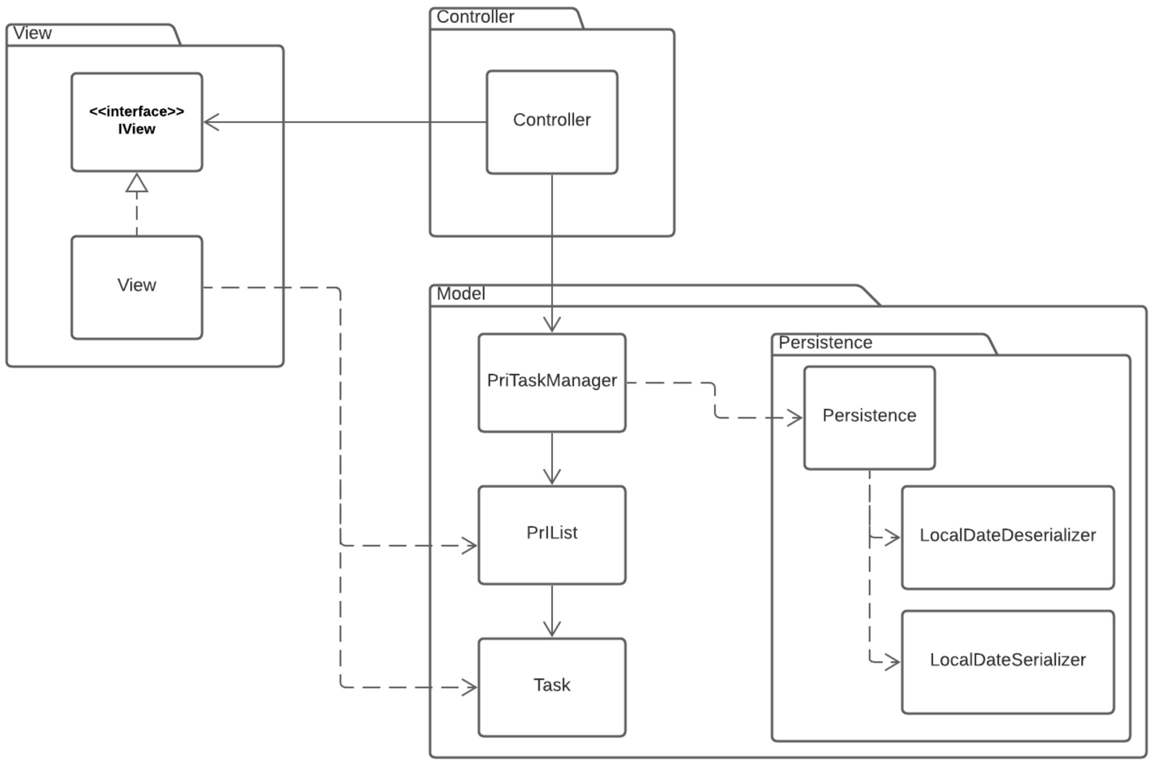

Figure 2: *An UML class diagram of the application and its most important features and the overall high-level architecture of the system.*

6.1. Use-Case: Create a list

6.1.1. Brief Description

The user creates a list that holds tasks.

6.1.2. Actor Brief Descriptions

- User - The user of the system.

6.1.3. Preconditions

1. Client has been started.

6.1.4. Basic Flow of Events

1. User presses the button to create a new list.
2. User is asked to name the list.
3. The user confirms and creates the lists.
4. The list is stored and saved on the local device.

6.1.5. Alternative Flows

2.1. The name is already taken.

2.2 The user is asked to use another name.

6.1.6. Subflows

6.1.7. Key Scenarios

6.1.8. Post-conditions

1. A list is created.

6.1.9. Special Requirements

6.2. Use-Case: Delete a list

6.2.1. Brief Description

The user deletes an existing list.

6.2.2. Actor Brief Descriptions

- User - The user of the system.

6.2.3. Preconditions

1. Client has been started.
2. A list has been created.

6.2.4. Basic Flow of Events

1. User presses the button to delete an existing list.
2. The system prompts the user for a confirmation for deleting the list.
3. The list gets deleted from the local device.

6.2.5. Alternative Flows

2.1. The user does not confirm to delete the list.
2.2 The list does not get deleted.
3.1 The list was not able to be deleted.

6.2.6. Subflows

6.2.7. Key Scenarios

6.2.8. Post-conditions

1. A list is deleted.

6.2.9. Special Requirements

6.3. Use-Case: Create a task

6.3.1. Brief Description

The user creates a task.

6.3.2. Actor Brief Descriptions

- User - The user of the system.

6.3.3. Preconditions

1. Client has been started.
2. A list has been created.

6.3.4. Basic Flow of Events

1. The user choses a list to add a task to it.
2. The user enters a name for the task.
3. The user enters a brief description of the task.
4. The user enters the priority of the task.
5. The user enters the deadline of the task.
6. The task is saved and stored on the local device.

6.3.5. Alternative Flows

5.1 The deadline has already expired.
5.2 The user is asked to repeat step 5.

6.3.6. Subflows

6.3.7. Key Scenarios

6.3.8. Post-conditions

1. A task is created.

6.3.9. Special Requirements

6.4. Use-Case: Delete a task

6.4.1. Brief Description

The user deletes a task.

6.4.2. Actor Brief Descriptions

- User - The user of the system.

6.4.3. Preconditions

1. Client has been started.
2. A list has been created.
3. A task has been created.

6.4.4. Basic Flow of Events

1. The user selects a list in which the desired task is located.
2. The user selects the task and presses the delete button.
3. The system prompts the user for a confirmation for deleting the task.
4. The task gets deleted from the local device.

6.4.5. Alternative Flows

6.4.6. Subflows

6.4.7. Key Scenarios

6.4.8. Post-conditions

2. A task is created.

6.4.9. Special Requirements

6.5. Use-Case: Edit a task

6.4.1. Brief Description

The user edits a task.

6.4.2. Actor Brief Descriptions

- User - The user of the system.

6.4.3. Preconditions

1. Client has been started.
2. A list has been created.
3. A task has been created.

6.4.4. Basic Flow of Events

1. The user selects a list in which the desired task is located.
2. The user selects the task and presses the edit button.
3. The system prompts the user for a confirmation for editing the task.
4. The task is saved and stored on the local device with the new edits.

6.4.5. Alternative Flows

3.1 The user does not confirm the prompt.

3.2 The changes are disregarded.

6.4.6. Subflows

6.4.7. Key Scenarios

6.4.8. Post-conditions

3. A task is edited.

6.4.9. Special Requirements

7. Performance modelling and analysis

The performance modelling that has been used for this project was a queuing network model created using JMT. The model was constructed using the UML diagram presented in chapter 6 of this document (see figure 2). The idea in the model (see figure 3) is that all inputs from the user will begin by being handled by the controller that then will send the request forward to the other components in the system. As long as the user keeps using the system, the system will send the user back to the controller whenever a request from the user has been handled and when the user's last request has been handled the session will end.

Analysis

In this model we can immediately see that there are a few components that will be used in all requests which means that these are probably the components that have the biggest risk of becoming a bottleneck for the system.

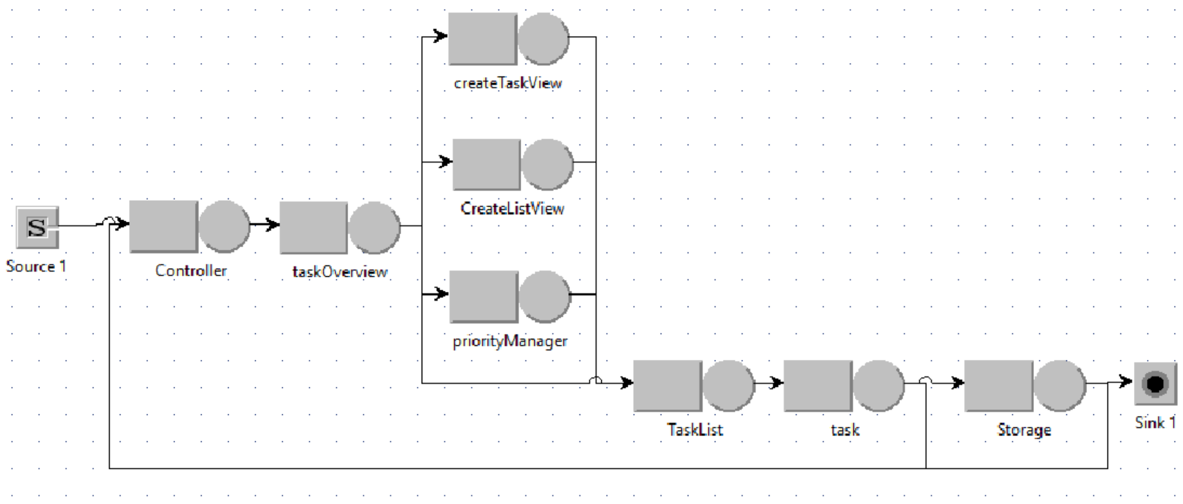

Figure 3: Queuing network model of the system. The idea is that everything starts with the user entering an input that is picked up by the controller and then sent to the various other components within the system.