

**Computer Science and Engineering**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Taskr**

**System Requirements Specification**

**Version 1.1**

Document Number: SRS-001

Project Team Number: A12

Project Team Members: Franky Cen Kenan Millet Yatin Kaushal

fc948 kvm237 yk1279

**REVIEW AND APPROVALS**

|  |  |  |  |
| --- | --- | --- | --- |
| **Printed Name and Title** | **Function (Author, Reviewer, Approval)** | **Date** | **Signature** |
| Kenan Millet | Author | 03/05/2016 | Kenan Millet |
| Yatin Kaushal | Author | 02/27/2016 | Yatin Kaushal |
| Franky Cen | Author | 03/1/2016 | Franky Cen |
| Franky Cen | Author | 3/20/2016 | Franky Cen |
| Yatin Kaushal | Author | 3/21/2016 | Yatin Kaushal |
| Kenan Millet | Author | 3/21/2016 | Kenan Millet |
| Franky Cen | Reviewer | 3/23/2016 | Franky Cen |
| Kenan Millet | Reviewer | 3/23/2016 | Kenan Millet |

**REVISION LEVEL**

|  |  |  |
| --- | --- | --- |
| **Date** | **Revision Number** | **Purpose** |
| 2/27/2016 | Version 1.0 | Initial Creation of the Document. All sections except 7.2 and 7.3 were made. |
| 3/23/2016 | Version 1.1 | 7.2 and 7.3 were added. Changes were made to 2.2, 3.1, 6, and 7.1. |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**TABLE OF CONTENTS**

**1. INTRODUCTION …………………………………………………………………………….5**

1.1 Purpose ……………………………………………………………………………………5

**2. SCOPE ……………………………………………………………………………………….5**

2.1 Identification …………………………………………………………………………….5

2.2 Bounds ……………………………………………………………………………………5

2.3 Objectives ……………………………………………………………………………….6

**3. OVERALL SYSTEM OVERVIEW ………………………………………………………….6**

3.1 Context Diagram ………………………………………………………………………….6

3.2 Additional Descriptive Items …………………………………………………………….6

**4. DOCUMENT OVERVIEW ………………………………………………………………….7**

**5. REFERENCE DOCUMENTS ……………………………………………………………….7**

**6. BUSINESS REQUIREMENTS** **………………………………………………………………7**

6.1 Technology ……………………………………………………………………………….7

6.2 Economics ……………………………………………………………………………….7

6.3 Regulatory and Legal ……………………………………………………………………7

[6.4 Market Considerations](#h.1ksv4uv) ……………………………………………………………………8

6.5 Risks and Alternatives ……………………………………………………………………8

6.6 Human Resources and Training ……………………………………………………….8

**7. SPECIFIC REQUIREMENTS ……………………………………………………………….8**

7.1 Functional Descriptive Detailed Requirements ………………………………………8-10

7.2 Requirements Use Cases …………………………………………………………….11

7.3 Non-Functional Descriptive Detailed Requirements ……………………………11-13

**8. SYSTEM TEST PLAN REQUIREMENTS ……………………………………………….13**

**9. QUALIFICATION PROVISIONS** **………………………………………………………….13**

**10. REQUIREMENTS TRACEABILITY ………………………………………………………14**

**11. EVOLUTION OF THE SRS ……………………………………………………………….14**

**12. RATIONALE ……………………………………………………………………………….14**

**13. NOTES ………………………………………………………………………………………14**

**14. APPENDICES** **……………………………………………………………………………….15**

14.1 Schedule Tracking ……………………………………………………………………15

14.2 Defect Tracking ………………………………………………………………………...16

14.3.1 Context Diagram …………………………………………………………………….17

14.7.2.1 Use Case Diagram ………………………………………………………………17

14.7.2.2 Use Case Descriptions ……………………………………………………….18-28

# 1. INTRODUCTION

## 1.1 Purpose

The purpose of this System Requirements Specification (SRS) is to describe the functional and non-functional requirements of the Taskr project, as well as to denote the business and test plan requirements for the development of the project with a level of detail that is relevant to the expected viewer of the given section of the document.

The intended audience of this SRS are the client, development team, and the development team’s advisor. Specifically, sections 1 through 4 are intended for the reference of all three parties (client, development team, and advisor) while the intended audience of the rest of the document is intended for the reference of the development team and their advisor only.

# 2. SCOPE

## 2.1 Identification

Taskr System Requirements Specification Version 1.1

## 2.2 Bounds

This product will help users organize their schedules. Users will provide the software with tasks, which they can put into schedules. Users will also provide the amount of free time they would like. The product uses an algorithm to suggest timeslots for new tasks. The information for the tasks and schedules are saved both locally and externally, on a server. Detailed features of this product are listed in Section 7.

This product will not force users to follow their schedule, or penalize them for doing so.

However, if users do not follow their schedule, this product will adjust its suggestions accordingly. This product can be for personal, commercial, and business use.

## 2.3 Objectives

From a business perspective, this product must be flexible, reliable, usable, and cost-efficient. This product should increase efficiency and save money for businesses.

The deliverables will be delivered as single deliverables on the following dates:

The initial Software Requirements Specifications was delivered on 03/07/16.

The final Software Requirements Specifications was delivered on 03/23/16.

The Software Project Management Plan is due on 04/6/16.

The Software Analysis Specifications is due on 04/18/16.

# 3. OVERALL SYSTEM OVERVIEW

## 3.1 Context Diagram

See Appendix 14.3.1

## 3.2 Additional Descriptive Items

Taskr will allow the user to efficiently schedule several upcoming tasks through use of a calendar view. The application will manage tasks using information such as deadline, urgency ratings, importance rating, desirability rating, breadth rating, and distinguish whether it is a filler (not an urgent task but still takes up time like a meeting or a class) or task (urgent time-based job). The application will allow the user to allocate a minimum daily or weekly free time that the scheduler will keep in mind when generating schedules. When doing so, the scheduler will determine various task scheduling possibilities for daily/weekly/monthly schedules at a time as well as suggest the user times available in their schedule to add a new task.

Only one type of user is kept in mind for interacting with this software. They need to be an individual with several tasks and deadlines that they need assistance in organizing (e.g. college student, businessman). This user would require at the very least minimal technical experience in using applications like a general Calendar app or to-do list app.

Taskr will be constrained by how many tasks the database would be able to hold and the internet connection used to fetch the tasks and load them. Another constraint will be how efficient the algorithm used to schedule the tasks will be. If the algorithm takes too long, the user will be reluctant to use the application.

# 4. DOCUMENT OVERVIEW

This document lists business requirements, specific descriptive requirements, and system test plan requirements first. Next, this document describes how it will be reviewed for quality, how each requirement is traceable from its original source, and how this document will evolve as the system development process progresses. Schedule tracking and defect tracking are located at the end of this document.

# 5. REFERENCE DOCUMENTS

Taskr Project Proposal v1.0

Objectives

Rationale

Taskr Software Requirements Specification v1.0

Business Drivers / Documents

Context Diagram

Requirements Traceability

System Scope: Bounds, Objectives, and Overview

System Capability Requirements

# 6. BUSINESS REQUIREMENTS

## 6.1 Technology

Taskr is an effective tool for any business to integrate in their workplace. Using Taskr as the default application for scheduling and time management, your business is sure to see an improvement in efficiency.

## 6.2 Economics

Taskr improves time management, allowing users of this technology to be more efficient when dealing with several business related tasks. By doing so, money is saved since time is saved.

## 6.3 Regulatory and Legal

To be completed at a later date.

## 6.4 Market Considerations

No applicable business market driver.

## 6.5 Risks and Alternatives

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Risk** | **Priority** | **Detection Method** | **Risk Responder** | **Solution** |
| Lack of demand for the product. | High | Lack of sales | Project Manager, Developers | Find a way to increase interest in the product. |
| System malfunctions | Varies | Customer comments | Project Manager, SQA group | Fix faults, ensure that the product works correctly. |

## 6.6 Human Resources and Training

Developers would need to understand how to create and manage databases in SQL as well as how to communicate with the database with the mobile client side. Training in Android/iOS development necessary to implement client-side application.

# 7. SPECIFIC REQUIREMENTS

## 7.1 Functional Descriptive Detailed Requirements

1. The user must be able to exit out of any menu.

1.1 The application should include a “Cancel” or “Back” icon in every menu except the main menu. When selected, the user will be prompted to confirm this action if there is any unsaved work.

1.2 After the confirmation, or if there is no unsaved work, the system should return to the previous menu.

2. The software must provide users with the ability to modify system options and parameters.

2.1 The application should include an “Options” icon. When selected, the options menu will open.

2.2 The user should be able to view all options and parameters of the system.

2.3 The user should be able to change the following parameters:

Minimum Daily Free Time – The amount of free time desired per day.

Minimum Weekly Free Time – The amount of free time desired per week.

Notifications - What the system reminds the user of.

- How often the system reminds the user.

- How the system reminds the user.

3. The software must provide users with the ability to create tasks.

3.1 The application should include a “New Task” icon. When selected, the interface for creating a new task will start.

3.2 The user must be able to add details, notes, and comments for the new task. The details of a task include its name, duration, priority, deadline, urgency, importance, type, and desirability. Also displayed is all schedules that this task is in.

3.3 The application should include “Save” and “Done” icons. Selecting either icon will save the task, and the appropriate action will occur (to be described in the Use Cases).

4. The software must provide users with the ability to view tasks.

4.1 The application should include a “Tasks” icon. When selected, all tasks will be displayed.

4.2 The tasks should initially be displayed in list form. In this form, only basic information about the task will be shown.

4.3 When a task is selected, all of its details will be displayed.

4.4 The user should be able to create a new task within this menu.

4.5 When viewing a single task, the user should be able to modify it.

4.6 The user should be able to transition between viewing all tasks and viewing a single task.

5. The software must provide users with the ability to modify tasks.

5.1 When viewing a task, the application should include an “Edit” icon. When selected, all of the task’s details are shown. The user should be able to change any detail.

5.2 The application should also include a “Save” icon. When selected, any changes will be saved.

5.3 From this menu, the user should be able to place the task into a schedule, as well as delete the task.

6. The software must provide users with the ability to place tasks into schedules.

6.1 When modifying a task, the application should include a “Place” icon. When selected, the user should be able to place the task into a schedule.

6.2 The user should be able to select a schedule in which to place a task.

6.2.1 If there is no existing schedule, the user will be prompted to create one.

6.2.2 If a schedule is selected, the system will recommend several timeslots in which to place the task. This recommendation will take into account all parameters and options of the system.

6.3 The user should be able to select a timeslot in which to place the task.

6.4 There should be a “Save” icon in this menu. When selected, any changes made will be saved.

7. The software must provide users with the ability to delete tasks.

7.1 When modifying a task, the application should include a “Delete” icon. When selected, the user will be prompted to confirm the deletion of a task.

7.2 After the confirmation, the task is deleted from the system.

8. The software must provide users with the ability to create schedules.

8.1 The application should include a “New Schedule” icon. When selected, the interface for creating a new schedule will start.

8.2 The user must be able to add details for the new schedule. The details of a schedule include its name, number of tasks, and applicability. A schedule’s applicability refers to when the schedule is applied. For example, one user could use “Schedule A” this week, and “Schedule B” next week. The user should be able to add personal notes and comments for the schedule.

8.3 The schedule must be saved.

9. The software must be able to generate schedules.

9.1 When creating a schedule, the application should include a “Generate” icon. When selected, the user will be prompted to pick one or more tasks from a list.

9.2 The system must be able to generate a schedule using the picked tasks, using the system parameters as desired constraints.

9.3 The user should then be able to select one or more schedules to save.

1.4 The schedules are saved after the user presses the “Save” icon.

10. The software must provide users with the ability to view schedules.

10.1 The application should include a “Schedules” icon. When selected, all schedules will be displayed.

10.2 The schedule should initially be displayed in list form. In this form, only basic information about the schedule will be shown. More recent schedules will be shown first.

10.3 When a schedule is selected, all of its details and tasks will be displayed.

10.4 The user can then select a schedule to view its details.

11. The software must provide users with the ability to modify schedules.

11.1 When viewing a schedule, the application should include an “Edit” icon. When selected, the user should be able to change any of the schedule details, as well as move tasks around within the schedule.

11.2 The application should include a “Save” icon. When selected, any changes will be saved.

11.3 Users should be able to remove tasks from the schedule.

12. The software must provide users with the ability to remove tasks from schedules.

12.1 When modifying a schedule, the application should include an “Remove” icon. When selected, the system will wait for the user to select a task within the schedule.

12.2 The user is then prompted to confirm this removal. After the confirmation, the task is removed from the schedule.

13. The software must provide users with the ability to delete schedules.

13.1 When modifying a schedule, the application should include an “Delete” icon. When selected, the user is prompted to confirm this deletion.

13.2 After the confirmation, the schedule is deleted from the system.

## 7.2 Requirements Use Cases

Use Case Diagram: see Appendix 14.7.2.1.

Use Case Descriptions: see Appendix 14.7.2.2.

## 7.3 Non-Functional Descriptive Detailed Requirements

**7.3.1 System Capabilities, conditions, and constraints**

The system shall be capable of providing the user with up-to-date information on their personal schedule as well as suggest to the user what the user should work on in regards to their desired tasks. The conditions under which the system should have these capabilities are that the system be active and informed by the user--as long as information is given to it for it to work with and the system interface is open, the system should be able to provide the user with the results of its actions as described in the functional requirements.

In regards to constraints, the system shall maintain a local database inside the phone’s storage. The size of the database should be as small as possible without losing any information. A copy of the information will also be retained on a server.

**7.3.2 Physical Resource Requirements**

**7.3.2.1 Computer Hardware Requirements**

The system shall be operated from the mobile phone of the user. As such, the device operating the system will need to have capabilities of touch-interaction. This is commonplace with “smart” phones.

**7.3.2.2 Computer Hardware Resource Requirements**

The system shall be minimalistic in its design so that it will use as few resources from its operating device as possible. None of the tasks stated in the functional requirements are resource intensive so this requirement should be met rather easily.

**7.3.2.3 Computer Software Requirements**

The system shall be downloadable via an online application store on either an android operating system on a mobile phone. Although cross-platform capability is a goal, it is not necessary for the first design of the product.

**7.3.3 Environmental Conditions**

The environment of the system is the android operating system. As such, the environmental conditions are not applicable to the system.

**7.3.4 System Performance characteristics**

The system performance is heavily based on the algorithm it uses to organize tasks and schedules. It also uses the algorithm to recommend several timeslots when adding a task to a schedule. If this algorithm is slow, the performance of the system drops significantly.

**7.3.5 Safety Requirements**

As the system exists solely on mobile phones, it is incapable of being a safety hazard to the user. The safety of the user’s information is detailed in section 7.3.6.

**7.3.6 Security and Privacy Requirements**

The system shall keep the user’s information safe by not granting access to the user’s information to any party other than the user. The system shall also not grant the user access to any other party’s information.

**7.3.7 System Human Interfaces**

The system shall notify the user of upcoming deadlines and give recommendations to the user through a graphical interface. The system shall also display a calendar with the user’s tasks when the user requests it. Similarly, upon request, the system shall show a chart organizing the user’s tasks by urgency and importance.

**7.3.8 System Maintainability**

The system shall be made up of very well defined modules which will have as high of cohesion as possible and will have only couplings which are vital to a module’s communication with other modules. Furthermore, each of these modules in the system shall work on their own without the aid of other modules, but shall also be able to work with the other modules as well--this will reduce maintenance time by making testing conditions more segregated between the modules.

**7.3.9 System Quality Factors**

The system quality factors include the speed and helpfulness of the algorithm used. If the algorithm runs quickly, it gets results quickly, and will not force users to wait for the program to perform calculations. If the algorithm gives recommendations that are acceptable to the user, it will have helped the user. The quality of the system depends on how well it can fulfill its intended purpose.

**7.3.10 Design and Construction Constraints**

**7.3.10.1 Life Cycle Model**

The development and maintenance of the system shall follow the Iterative and Incrementation Life Cycle Model.

**7.3.10.2 Policies and standards - Methods, tools, and techniques**

The methods, tools, and techniques of the system will be determined at a later date.

**7.3.11 Personnel-Related Requirements**

The developers of the system shall have the knowledge required to implement the design of the system and fulfill the requirements of this SRS through programming in whatever language is deemed necessary in order to develop the system for the android OS. No certification is necessary for the developers.

The system does not have any particular requirements on the users other than having access to and the knowledge of how to operate a smartphone.

**7.3.12 Training-Related Requirements**

No special training is required for the development of this system beyond knowledge of programming and general programming practices.

**7.3.13 Logistics-Related Requirements**

The system shall store and retrieve data from a database and will work autonomously without the support of any services physically beyond the domain of the smartphone running it. The system itself shall be obtained by the user for free from the google play store. Any other logistical requirements such as training and packaging should be covered by the other requirements in this SRS.

**7.3.14 Packaging Requirements**

The system shall be packaged as a single mobile application. Although future versions may allow the user to connect to, store data, and retrieve data from servers, this version will store data locally on the mobile phone, resulting in the entire system being contained in one application.

# 8. SYSTEM TEST PLAN REQUIREMENTS

Once Completed and approved, this document will be given to the Software Quality Group, who will develop the test plan and a set of test scenarios (based upon use cases), expected output, execute the tests and report any defects. Each feature of the product will be tested against several scenarios. This testing will ensure that the product functions correctly. The testing will be conducted using an SQL server.

# 9. QUALIFICATION PROVISIONS

This document will be reviewed to make sure that it is correct, unambiguous, complete, consistent, stable, verifiable, modifiable, and traceable. All reviewers will thoroughly evaluate each detail in this document to make sure there are no errors. Reviewers will make sure that the document meets each of the above attributes. If any errors are found, reviewers will leave comments for each part that needs to be amended. Changes will be made to a section only after every reviewer has gone over it. Sections will be reviewed after each correction. This document will be reviewed until there are no errors.

# 10. REQUIREMENTS TRACEABILITY

A traceability matrix will be used to ensure requirement traceability. The traceability matrix is a document in the form of a table that can be used to check whether or not the current requirements are being met. The relationships of each succeeding artifact to their source document will be recorded so that there is both forward and backward traceability. When a requirement is changed in a source document, it will be easy to determine what needs to be changed in other documents.

# 11. EVOLUTION OF THE SRS

This document will change as the system development process progresses. All changes will be recorded in the changelog. The changelog will include an accurate and complete trail of changes. Requested changes will be recorded in the changelog. Changes will be implemented only after review and approval. The requester, reviewer, and approver of the change will also be recorded in the changelog.

# 12. RATIONALE

Time management is a problem that many in first world countries face. The common saying goes that there are only twenty-four hours in the day, but if used efficiently, twenty-four hours can be a very long time.

We believe that the solution to efficient use of time is through the proper mindset. Unfortunately, due to entertainment, stress, friends, family, and many other important but deviating tasks, such a mindset can be very difficult to adopt and equally difficult to maintain. In order to help the population adapt to the mindset of the successful, we will develop an application that acts as a scheduler that can be used to help ease people into managing their time efficiently.

# 13. NOTES

# 14. APPENDICES

## 14.1 Schedule Tracking

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Artifact / Deliverable** | **Individual** | **Estimated** | **Actual** | **Difference** |
| Initial SRS | Franky | 14 hours | 8 hours | 6 hours |
| Kenan | 12 hours | 14 hours | 2 hours |
| Yatin | 15 hours | 7 hours | 8 hours |
| **Total** | **41 hours** | **29 hours** | **12 hours** |
| Final SRS | Franky | 15 hours | 9 hours | 6 hours |
| Kenan | 20 hours | 6.5 hours | 13.5 hours |
| Yatin | 15 hours | 5 hours | 10 hours |
| **Total** | **50 hours** | **20.5 hours** | **29.5 hours** |

**Cumulative**

|  |  |  |  |
| --- | --- | --- | --- |
| **Individual** | **Estimated** | **Actual** | **Difference** |
| Franky | 29 hours | 17 hours | 12 hours |
| Kenan | 32 hours | 20.5 hours | 11.5 hours |
| Yatin | 30 hours | 12 hours | 18 hours |
| **Total** | **91 hours** | **49.5 hours** | **41.5 hours** |

## 

## 14.2 Defect Tracking

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Artifact / Deliverable** | **Individual** | **Estimated** | **Actual** | **Difference** |
| Initial SRS | Franky | 20 faults | 10 faults | 10 faults |
| Kenan | 10 faults | 6 faults | 4 faults |
| Yatin | 10 faults | 3 faults | 7 faults |
| **Total** | **40 faults** | **19 faults** | **21 faults** |
| Final SRS | Franky | 20 faults | 24 faults | 4 faults |
| Kenan | 20 faults | 2 faults | 18 faults |
| Yatin | 15 faults | 3 faults | 12 faults |
| **Total** | **55 faults** | **29 faults** | **26 faults** |

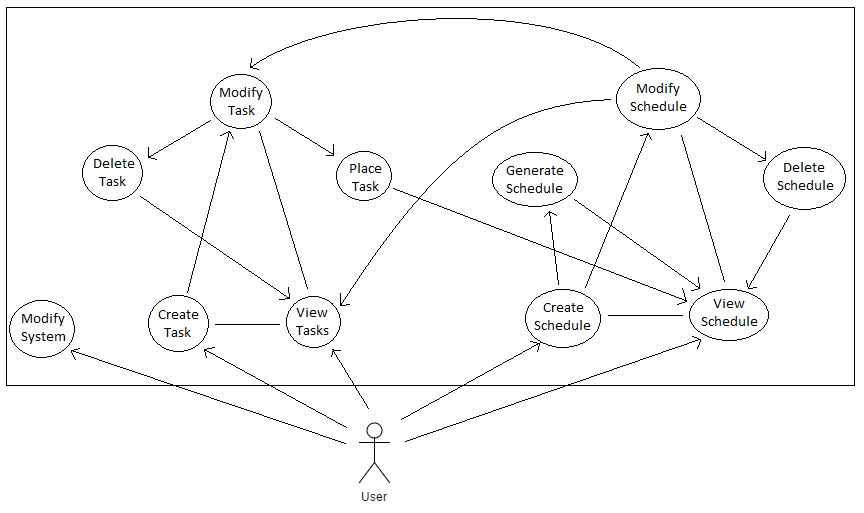
**Cumulative**

|  |  |  |  |
| --- | --- | --- | --- |
| **Individual** | **Estimated** | **Actual** | **Difference** |
| Franky | 40 faults | 34 faults | 6 faults |
| Kenan | 30 faults | 8 faults | 22 faults |
| Yatin | 25 faults | 6 faults | 19 faults |

**14.3.1 Context Diagram**

https://lh6.googleusercontent.com/WBMC_9TbtMGNUWVkSBb7Vx97RjsbQ9SHmEgRwcn4deaBAT8Yutu43EmcCeGpPtr-ntxMlneDDJLdsL5J_kAkHcFfyrTvIZuLVBhJnR9jh-rUv1qUHfErsiXT-Eyzv3uw-xjHtScL

**14.7.2.1 Use Case Diagram**



**14.1.3 Use Case Descriptions**

|  |  |  |
| --- | --- | --- |
| **Create Task** | | |
| **Description** | The user creates a new task, which is saved. | |
| **Pre-Conditions** |  | |
| **Flows** | **Basic / Normal Flows** | 1. The Use Case starts when the user presses the New Task icon.  2. The user can now add details to this Task, such as the name, duration, and priority. The user can also add personal notes and comments.  3. The user presses the Save or Done icon. The task is saved and this Use Case ends. See Extension Points below (3.1 and 3.2). |
| **Alternative Flows** | 1. User presses the Cancel icon. The Task is not saved, and this Use Case ends.  2. The Application closes prematurely. The task is not saved, and this Use Case ends. |
| **Post-Conditions** |  | |
| **Special Requirements** |  | |
| **Extension Points** | 3.1.  If the Save icon is pressed, the Modify Task Use Case starts. The created task will be able to be modified.  3.2.  If the Done icon is pressed, the View Task Use Case starts. | |

|  |  |  |
| --- | --- | --- |
| **View Tasks** | | |
| **Description** | The user views all tasks. | |
| **Pre-Conditions** |  | |
| **Flows** | **Basic / Normal Flows** | 1. The Use Case starts when the user presses the Tasks icon.  2. A list of all tasks are displayed. Basic information for each task is shown. More recent tasks are listed first.  3. When the user selects a task, only that task will be shown. All details of that task will be displayed.  4. When viewing a single task, if the user presses the Back icon, the list of all tasks will be shown. |
| **Alternative Flows** | 1. User presses the Back icon. The software exits to the main menu.  2. The Application closes prematurely. The Use Case then ends. |
| **Post-Conditions** |  | |
| **Special Requirements** |  | |
| **Extension Points** | 1.  When viewing a single task, if the user presses the Edit icon, this Use Case ends and the Modify Task Use Case starts.  2.  When viewing the list of tasks, if the user presses the New Task icon, this Use Case ends and the Create Task Use Case starts. | |

|  |  |  |
| --- | --- | --- |
| **Modify Task** | | |
| **Description** | The user changes the details of a task. | |
| **Pre-Conditions** | There user must be viewing an existing task. | |
| **Flows** | **Basic / Normal Flows** | 1. This Use Case starts when the user presses the Edit icon when viewing a task, or when selecting a task when modifying a schedule.  2. The task’s details are shown. The user can click on any detail to modify it. The user can also add notes, or comments.  3. The user presses the Save icon and the changes are saved. |
| **Alternative Flows** | 1. User presses the Cancel icon. The modifications are not saved, and this Use Case ends.  2. The Application closes prematurely. The Task is not saved, and this Use Case ends. |
| **Post-Conditions** |  | |
| **Special Requirements** |  | |
| **Extension Points** | 1.  If the user presses the Back icon, this Use Case ends and system returns to the previous menu. If the previous menu was the list of all tasks, the View Tasks Use Case starts. If the previous menu was a schedule, the Modify Schedule Use Case starts.  2.  If the user presses the Delete icon, this Use Case ends and the Delete Task Use Case starts.  3.  If the user presses the Place icon, this Use Case ends and the Place Task Use Case starts. | |

|  |  |  |
| --- | --- | --- |
| **Place Task** | | |
| **Description** | The user places a task into a schedule. | |
| **Pre-Conditions** | The user must have selected a task to be modified. The task must have a set duration. | |
| **Flows** | **Basic / Normal Flows** | 1. This Use Case starts when the user presses the Place icon.  2. The user can add the selected task to an existing schedule or to a new schedule. The user selects which schedule the task is to be added to.  3. When selecting an existing schedule, the software will recommend several timeslots in which the task can be placed.  4. After confirming the placement of a task, this Use Case ends. |
| **Alternative Flows** | 1. User presses the Cancel icon. This Use Case ends.  2. The Application closes prematurely. This Use Case ends.  3.  In Step 2, if there is no existing schedule, the user will be asked to create one. |
| **Post-Conditions** |  | |
| **Special Requirements** |  | |
| **Extension Points** | When this Use Case ends, the View Schedule Use Case starts. | |

|  |  |  |
| --- | --- | --- |
| **Delete Task** | | |
| **Description** | The user deletes a task. | |
| **Pre-Conditions** | The user must have selected an existing task to be modified. | |
| **Flows** | **Basic / Normal Flows** | 1. This Use Case starts when the user presses the Delete icon when modifying a task.  2. After confirming a deletion, the task is removed, and this Use Case ends. |
| **Alternative Flows** | 1. This Use Case ends when the user presses the Cancel icon.  2. This Use Case ends when the Application closes prematurely. |
| **Post-Conditions** |  | |
| **Special Requirements** |  | |
| **Extension Points** | When this Use Case ends, the View Tasks Use Case starts. | |

|  |  |  |
| --- | --- | --- |
| **Create Schedule** | | |
| **Description** | The user creates a new schedule, which is saved. | |
| **Pre-Conditions** |  | |
| **Flows** | **Basic / Normal Flows** | 1. This Use Case starts when user presses the New Schedule icon.  2. The user can now add details to this schedule.  3. The user presses the Save or Done icon. The schedule is saved and this Use Case ends. See Extension Points below (3.1 and 3.2). |
| **Alternative Flows** | 1. User presses the Cancel icon. The Task is not saved, and this Use Case ends.  2. The Application closes prematurely. The Task is not saved, and this Use Case ends. |
| **Post-Conditions** |  | |
| **Special Requirements** |  | |
| **Extension Points** | 3.1.  If the Save icon is pressed, the Modify Schedule Use Case starts. The created task will be able to be modified.  3.2.  If the Done icon is pressed, the View Task Use Case starts. | |

|  |  |  |
| --- | --- | --- |
| **Generate Schedule** | | |
| **Description** | The user asks the system to generate a schedule with a number of given tasks. | |
| **Pre-Conditions** |  | |
| **Flows** | **Basic / Normal Flows** | 1. This Use Case starts when user presses the Generate icon when in the schedule creating menu.  2. The user selects tasks from a list.  3. The system generates several schedules.3  4. The user then selects one or more schedules to save.  5. This Use Case ends when the user presses the Done icon. |
| **Alternative Flows** | 1. User presses the Cancel icon. The Task is not saved, and this Use Case ends.  2. The Application closes prematurely. The Task is not saved, and this Use Case ends. |
| **Post-Conditions** |  | |
| **Special Requirements** |  | |
| **Extension Points** | If the Done icon is pressed, the View Schedule Use Case starts. | |

|  |  |  |
| --- | --- | --- |
| **View Schedules** | | |
| **Description** | The user views all schedules. | |
| **Pre-Conditions** |  | |
| **Flows** | **Basic / Normal Flows** | 1. The Use Case starts when the user presses the Schedules icon.  2. A list of all schedules are displayed. Basic information for each schedule is shown. More recent schedules are listed first. The user can switch the view mode to see a picture that represents each schedule.  3. When the user selects a schedule, only that schedule will be shown. All details of the schedule will be displayed.  4. When viewing a single schedule, if the user presses the Back icon, the list of all schedules will be shown. |
| **Alternative Flows** | 1. User presses the Back icon. The software exits to the main menu.  2. The Application closes prematurely. The Use Case then ends. |
| **Post-Conditions** |  | |
| **Special Requirements** |  | |
| **Extension Points** | 1.  If the user presses the Edit icon when viewing a single schedule, this Use Case ends and the Modify Schedule Use Case starts.  2.  If the user presses the New Schedule icon when viewing the list of all schedules, this Use Case ends and the Create Schedule Use Case starts. | |

|  |  |  |
| --- | --- | --- |
| **Modify Schedule** | | |
| **Description** | The user changes a schedule’s details. | |
| **Pre-Conditions** | The user must be viewing an existing schedule. | |
| **Flows** | **Basic / Normal Flows** | 1. This Use Case starts when the user presses the Edit icon when viewing a schedule.  2. The schedule’s details are shown. The user can select any detail to modify it. The user can also add notes and comments. The tasks within the schedule are also shown.  3. The user can also remove tasks from this schedule by pressing the Remove icon, then selecting a task. The user can also move tasks around within the schedule.  4. The user presses the Save icon and the changes are saved. |
| **Alternative Flows** | 1. User presses the Cancel icon. The Task is not saved, and this Use Case ends.  2. The Application closes prematurely. The Task is not saved, and this Use Case ends. |
| **Post-Conditions** |  | |
| **Special Requirements** |  | |
| **Extension Points** | 1.  If the user presses the Back icon, this Use Case ends and system returns to the View Schedules Use Case.  2.  If the user presses the Delete icon, this Use Case ends and the Delete Schedule Use Case starts.  3.  If the user selects a task, this Use Case ends and the Modify Task Use Case starts.  4.  If the user selects the Add icon, this Use Case ends and the View Tasks Use Case starts. | |

|  |  |  |
| --- | --- | --- |
| **Delete Schedule** | | |
| **Description** | The user deletes a schedule from the program. | |
| **Pre-Conditions** | The user must have selected a schedule to be modified. | |
| **Flows** | **Basic / Normal Flows** | 1. This Use Case starts when the user presses the Delete icon when modifying a schedule.  2.  After confirming a deletion, the schedule is removed and this Use Case ends. |
| **Alternative Flows** | 1. User presses the Cancel icon. The Task is not saved, and this Use Case ends.  2. The Application closes prematurely. The Task is not saved, and this Use Case ends. |
| **Post-Conditions** |  | |
| **Special Requirements** |  | |
| **Extension Points** | When this Use Case ends, the View Schedules Use Case starts. | |

|  |  |  |
| --- | --- | --- |
| **Modify System** | | |
| **Description** | The user changes system parameters. | |
| **Pre-Conditions** |  | |
| **Flows** | **Basic / Normal Flows** | 1. This Use Case starts when the user presses the Options icon.  2. The system’s options and parameters are shown. The user can select any option or parameter to modify it.  3. The user presses the Save icon and the changes are saved. |
| **Alternative Flows** | 1. User presses the Cancel icon. The changes are not saved, and this Use Case ends.  2. The Application closes prematurely. The changes are not saved, and this Use Case ends. |
| **Post-Conditions** |  | |
| **Special Requirements** |  | |
| **Extension Points** |  | |