https://lh4.googleusercontent.com/gKT2McPFNOJ67GHMaDnOq-EZgut7D5GqTKPFr0HtIaNfwQwpYx4vjdEuPS7aIW-rsUWGfca_-2AQd3qO3N0UGL7zrGVI8ZrvnY-48cCiu59v2UONen484H9Qd8tc2xjm2pVsJFzw

**Computer Science and Engineering**

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

**Taskr**

**System Analysis Specification**

**Version 1.0**

Document Number: SAS-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** |
| Franky Cen | Author | 4/11/2016 | Franky Cen |
| Kenan Millet | Author | 4/13/2016 | Kenan Millet |
| Yatin Kaushal | Author | 4/16/2016 | Yatin Kaushal |
| Kenan Millet | Reviewer | 4/17/2016 | Kenan Millet |
| Yatin Kaushal | Reviewer | 4/18/2016 | Yatin Kaushal |
| Franky Cen | Reviewer | 4/18/2016 | Franky Cen |
| Franky Cen | Approver | 4/18/2016 | Franky Cen |

**REVISION LEVEL**

|  |  |  |
| --- | --- | --- |
| **Date** | **Revision Number** | **Purpose** |
| 4/11/2016 | Version 1.0 | Initial creation of the document |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**TABLE OF CONTENTS**

**1.   Introduction ……………………………………………………………………5**  
   1.1   Purpose ……………………………………………………………………………5

**2.   Scope ………………………………………………………………………….5**  
   2.1   Identification ………………………………………………………………………..5  
   2.2   Bounds ……………………………………………………………………………5  
   2.3   Objectives ………………………………………………………………………5  
   2.4   System Overview ……………………………………………………………….6  
   2.5   Document Overview ……………………………………………………………..6

**3.   Reference Documents ………………………………………………………6**

**4.   Business Documents ……………………………………………………….7**   4.1   Technology ………………………………………………………………………7  
   4.2   Economics ………………………………………………………………………7  
   4.3   Regulatory and Legal …………………………………………………………….7  
   4.4   Market Considerations …………………………………………………………….7  
   4.5   Risks and Alternatives …………………………………………………………….7  
   4.6   Human Resources and Training …………………………………………………7

**5.   Logical Architectural Specification ………………………………………8**  
   5.1   Context Diagram …………………………………………………………………8  
   5.2   System Capability Requirements ………………………………………………8  
       5.2.1   Capabilities ………………………………………………………………….8  
       5.2.2   Use Case Diagrams ………………………………………………………..8  
       5.2.3   Use Case Descriptions ………………………………………………………8  
   5.3   User Interface Requirements ……………………………………………………8  
   5.4   Component Architecture ……………….………………………………………9

       5.4.1   Component Descriptions …………………………………………………9-10  
       5.4.2   Component Architecture Diagram ………………………………………..10  
   5.5   Class Diagrams ………………………………………………………………….11  
   5.6   Class Relationships / Interaction Diagrams ……………………………………11  
   5.7   Events ……..…………………………………………………………………….11  
       5.7.1   Motives ………………………………………………………………………11  
       5.7.2   Event Diagrams ……………………………………………………………..11  
   5.8   Activity/State Section …………………………………………………………….11  
   5.9   State Logic …………………………………………………………………………11  
   5.10   Behavior …………………………………………………………………………11  
       5.10.1   Sequence Diagrams ……………………………………………………11  
       5.10.2   Collaboration Diagrams …………………………………………………11  
   5.11   Dictionaries ………………………………………………………………..12-13

**6.   Non-Functional / Operational Specifications …………………………….13**  
   6.1   System External Interface Requirements ……………………………………..14  
   6.2   Safety Requirements …………………………………………………………….14  
   6.3   Security and Privacy Requirements ……………………………………………14  
   6.4   System Environment Requirements ……………………………………………14  
   6.5   Computer Resource Requirements …………………………………………….14  
       6.5.1   Computer Hardware Requirements ……………………………………….14  
       6.5.2   Computer Hardware Resource Requirements ……………………………14  
       6.5.3   Computer Software Requirements ………………………………………14  
       6.5.4   Computer Communications Requirements ………………………………14  
   6.6   System Quality Factors ………………………………………………………..14  
   6.7   Design and Construction Constraints ………………………………………….14

   6.8   Personnel-Related Requirements ………………………………………………15

   6.9   Training-Related Requirements ………………………………………………...15

   6.10   Logistic-Related Requirements ………………………………………………...15

   6.11   Packaging Requirements ……………………………………………………...15

**7.   System Test Plan Requirements ………………………………………….15**

**8.   Qualification Provisions …………………………………………………….16**

**9.   Requirements Traceability ………………………………………………….16**

**10.   Rationale ……………………………………………………………………16**

**11.   Notes …………………………………………………………………………16**

**12. Appendices …………………………………………………………………17**

   12.1 Schedule Tracking ……………………………………………………………….17

   12.2 Defect Tracking ………………………………………………………………….18

12.5.1 Context Diagram …………………………………………………………….19

12.5.2.2 Use Case Diagram ………………………………………………………19

12.5.2.3 Use Case Descriptions ……………………………………………...20-30

12.5.4.2 Component Architecture Diagram ………………………………………...31

12.5.5 Class Diagrams ……………………………………………………………31

12.5.6 Class Relationship Diagram ………………………………………………...32

12.5.7.2 Event Diagrams …………………………………………………………...32

12.5.8 Activity / State Section ……………………………………………………….33

12.5.10.1 Sequence Diagrams ……………………………………………………...34

**1. Introduction**

1.1 Purpose

This purpose of this document is to analyze and refine the requirements defined in the System Requirements Specification (SRS). Both the functional and nonfunctional requirements are analyzed. This document specifies what the product must do and lists the constraints on the product. The intended audience for this document are the client, SQA group, project manager, and the project advisor.

**2. Scope**

2.1 Identification

Taskr Software Analysis Specifications Version 1.0

2.2 Bounds

This project will create a product that assists users with scheduling tasks and events. The product will be completed by December 2016. The product will be able to suggest optimal time slots for new tasks to be placed in. The product will also be able to reorganize current schedules and generate new schedules based on existing tasks.

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 was delivered on 04/6/16.

The Software Analysis Specifications was delivered on 04/18/16.

The Software Design document is due 04/25/16.

2.4 System Overview

This product is expected to help users organize their schedules. The product can help create a schedule that most benefits the user. The information for the tasks and schedules are saved both locally and externally on a server. 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, or business use.

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.

2.5 Document Overview

Overview of the contents of this document

Outline what each section contains

This document includes architectural specifications and requirement specifications. The architectural specifications detail the components, internal structure, context diagrams, and Use Cases of the product. The requirements are analyzed to ensure they meet the client’s needs. This document also includes the product’s operational requirements and constraints.

**3. Reference Documents**

Taskr Project Proposal v1.0

Objectives

Rationale

Taskr Software Requirements Specification v1.1

Business Drivers / Documents

Context Diagram

Requirements Traceability

System Scope: Bounds, Objectives, and Overview

System Capability Requirements

Use Cases: Diagram and Descriptions

Other Various Requirements

Taskr Software Project Management Plan v1.0

Test Plans

**4. Business Documents**

4.1 Technology

Taskr is an effective tool for any business to integrate in their workplace. Businesses that use Taskr for scheduling and time management will see an improvement in efficiency.

4.2 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.

4.3 Regulatory and Legal

To be completed at a later date.

4.4 Market Considerations

No applicable business market driver.

4.5 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. |

4.6 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.

**5. Logical Architectural Specification**

5.1 Context Diagram

See Appendix 12.5.1

5.2 System Capability Requirements

The system shall be capable of providing the user with up-to-date information on their personal schedule as well as suggest what the user should work on in regards to their desired tasks. In order for the system to do this, the system must be active and information must be provided by the user. As long as information is given for the system 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 (detailed in the SRS).

5.2.1 Capabilities

Use Cases:

- Change System Settings

- View Tasks

- Create Task

- Modify Task

- Delete Task

- Place Task into a Schedule

- View Schedules

- Create Schedule

- Modify Schedule

- Delete Schedule

- Generate Schedule

5.2.2 Use Case Diagram

See Appendix 12.5.2.2

5.2.3 Use Case Descriptions

See Appendix 12.5.2.3

5.3 User Interface Requirements

The product will be used on mobile smartphones. The interface should be compatible with any smartphone interface.

5.4 Component (Component/Package/Subsystem) Architecture

The system shall operate as modularly as possible. As such, it shall be split up into multiple modules and components. The system shall be split up into an interface module, a core module, a brain module, and a database module. The interface module will have two input groups (user interaction and outputs from the core module) and two output groups (graphical outputs to the screen and functional outputs to the core module). The core module will have three input groups (functional outputs from the interface module, data outputs from the brain module, and data from the database module) and four output groups (functional outputs to the core module, data to be stored by the database module, functional and data outputs to the brain module, and data requests to the database module). The brain module will have one input group (functional requests from the database module) and one output group (data to the core module). The database module will have three input group (data requests from the core module, data to be stored from the core module, and data received from the server) and two output groups (data sent to the core module and data sent to the server).

5.4.1 Component Descriptions

Identify and describe the functionality each system component or subsystem

Core Module:

* This module will be responsible for handling underlying functionality of the inner workings of the system and will also provide an intermediary link between the other three modules. As such, this module will contain the most components.
* The major components of the module are as follows:
  + Fetch: Sends a request to the database module for data and then fetches the data that the database module returns.
  + Store: Sends data to the database module to be stored.
  + Compute: Sends data and functional outputs to the brain module and then fetches the data that the brain module returns. There will be a variety of these components due to the different algorithms that will need to be performed.
  + Serve: Accepts functional outputs from the interface module and then sends the appropriately formatted data packages to the interface module to display.
  + Format: Formats the data received from the brain module to make it easier for the interface module to display effortlessly. There will be a variety of these components due to the different formats in which data will be displayed to the screen.

Interface Module:

* This module will be responsible for relaying user inputs to the core module and displaying formatted data that is passed to it by the core module.
* The major components of the module are as follows:
  + Display: Displays data sent to it by the core module to the screen. There will be multiple types of display components based on various formats of data sent by the core module.
  + Input: Gets user input and stores it within the module temporarily
  + Compile: Determines and temporarily stores a functional output within the module based on stored user input. This will be done through a lookup table.
  + Request: Sends functional output to the core module that was stored through the compile component.

Database Module:

* This module will be responsible for storing, organizing, and sending data to the database module upon request.
* The major components of the module are as follows:
  + Store: Stores data that is passed into it by the core module.
  + Organize: Organizes data when needed. There will be one of these components for each type of data that is stored.
  + Fetch: Fetches data from the database and sends it as output to the data module.
  + Clean: Determines whether or not a section and/or type of data within the database is needed to be organized.
  + Push: Sends data to the server.
  + Pull: Requests and receives data from the server.
  + Data: Holds all permanent data for the product.

Brain Module:

* This module will be responsible for performing any and all algorithms and complex calculations based on data and function inputs given to it by the core module.
* The major components of the module are as follows:
  + Compute: Will perform the specific computation or algorithm that it is told to perform by the functional input from the core module on a given set of data that is passed into it by the core module. There will be many of these components due to the algorithms and computations that are needed in this product.
  + Send: Sends out data that results from the compute component to the core module.

5.4.2 Component Architecture Diagram

See Appendix 12.5.4.2

5.5 Class Diagrams

See Appendix 12.5.5.

5.6 Class Relationships / Interaction Diagrams

See Appendix 12.5.6

5.7 Events

* + 1. Motives

The User changes system settings.

The User requests to view all Tasks.

The User creates a Task.

The User deletes a Task.

The User modifies a Task.

The User places a Task into a Schedule.

The User requests to view all Schedules.

The User creates a Schedule.

The User requests the system to generate a Schedule.

The User deletes a Schedule.

The User modifies a Schedule.

5.7.2 Event Diagrams

See Appendix 12.5.7.2

5.8 Activity/State (Scenario) Section

See Appendix 12.5.8

5.9 State Logic

To be completed at a later date.

5.10 Behavior

5.10.1 Sequence Diagrams

See Appendix 12.5.10.1

5.10.2 Collaboration Diagrams

To be completed at a later date.

5.11 Dictionaries

Classes: User, Database, Task, Schedule

Attributes: User: Name

Database: List of Tasks

List of Schedules

Task: Name

Duration

Priority

Schedule Name

Number of Tasks

List of Tasks

Methods: User: changeSettings()

viewTasks()

createTask()

deleteTask()

modifyTask()

placeTask()

viewSchedules()

createSchedule()

generateSchedule()

deleteSchedule()

modifySchedule()

Database: modify()

displayTasks()

createTask()

deleteTask()

modifyTask()

placeTask()

displaySchedules()

createSchedule()

generateSchedule()

deleteSchedule()

modifySchedule()

Task: modify()

Schedule: modify()

addTask()

Events: The User changes system settings.

The User requests to view all Tasks.

The User creates a Task.

The User deletes a Task.

The User modifies a Task.

The User places a Task into a Schedule.

The User requests to view all Schedules.

The User creates a Schedule.

The User requests the system to generate a Schedule.

The User deletes a Schedule.

The User modifies a Schedule.

Associations: User-Database: one to one relationship

Database-Task: one to many relationship

Database-Schedule: one to many relationship

**6. Non-Functional / Operational Specifications**

6.1 System External Interface Requirements

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. The system shall include a way for users to perform all actions listed in the Use Cases (Section 5.2.2).

6.2 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 6.3.

6.3 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.

6.4 System Environment Requirements

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

6.5 Computer Resource Requirements

6.5.1 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.

6.5.2 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.

6.5.3 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.

6.5.4 Computer Communications Requirements

The system must be able to communicate with the server. A database is saved both locally and externally. The database shall be synchronized between the local device and the server.

6.6 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.

6.7 Design and Construction Constraints

The development and maintenance of the system shall follow the Iterative and Incrementation Life Cycle Model. The methods, tools, and techniques of the system will be determined at a later date.

6.8 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 users of the system have no particular requirements other than access to an knowledge of how to operate a smartphone.

6.9 Training-Related Requirements

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

6.10 Logistic-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.

6.11 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.

**7. 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 on the 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.

**8. 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.

**9. 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. New requirements are added to the traceability matrix. 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, changes that need to be done in documents can be determined using the traceability matrix. Changes will be requested, reviewed, then approved before being applied.

**10. 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.

**11. Notes**

**12. Appendix**

12.1 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** |
| SPMP | Franky | 13 hours | 16.5 hours | 3.5 hours |
| Kenan | 18 hours | 12 hours | 6 hours |
| Yatin | 10 hours | 7 hours | 3 hours |
| **Total** | **41 hours** | **35.5 hours** | **5.5 hours** |
| SAS | Franky | 15 hours | 13 hours | 2 hours |
| Kenan | 12 hours | 7.5 hours | 4.5 hours |
| Yatin | 15 hours | 8 hours | 7 hours |
| **Total** | **42 hours** | **28.5 hours** | **13.5 hours** |

Cumulative

|  |  |  |  |
| --- | --- | --- | --- |
| **Individual** | **Estimated** | **Actual** | **Difference** |
| Franky | 57 hours | 46.5 hours | 10.5 hours |
| Kenan | 62 hours | 40 hours | 22 hours |
| Yatin | 55 hours | 27 hours | 28 hours |
| **Total** | **174 hours** | **113.5 hours** | **60.5 hours** |

12.2 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** |
| SPMP | Franky | 20 faults | 16 faults | 4 faults |
| Kenan | 20 faults | 17 faults | 3 faults |
| Yatin | 20 faults | 4 faults | 16 faults |
| **Total** | **60 faults** | **37 faults** | **23 faults** |
| SAS | Franky | 15 faults | 13 faults | 2 faults |
| Kenan | 20 faults | 12 faults | 8 faults |
| Yatin | 18 faults | 15 faults | 3 faults |
| **Total** | **53 faults** | **40 faults** | **13 faults** |

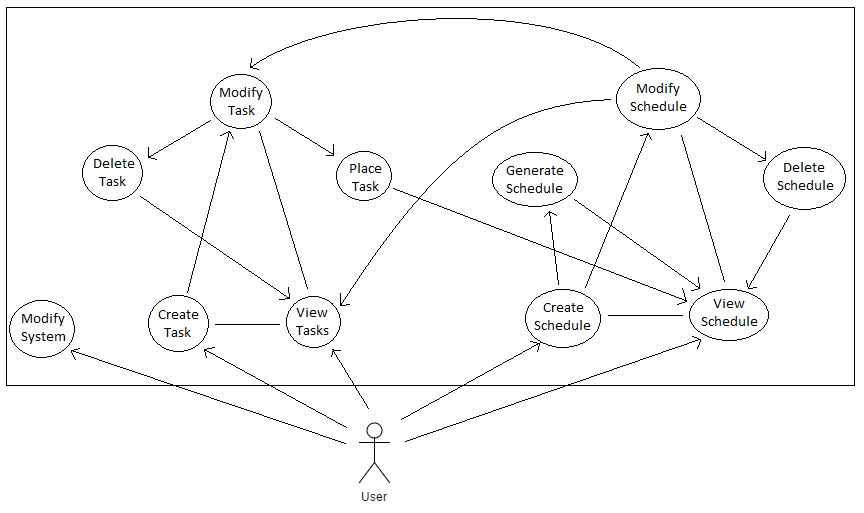
Cumulative

|  |  |  |  |
| --- | --- | --- | --- |
| **Individual** | **Estimated** | **Actual** | **Difference** |
| Franky | 75 faults | 63 faults | 12 faults |
| Kenan | 70 faults | 37 faults | 33 faults |
| Yatin | 63 faults | 40 faults | 23 faults |
| **Total** | **208 faults** | **140 faults** | **68 faults** |

12.5.1 Context Diagram

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

12.5.2.2 Use Case Diagram



12.5.2.3 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** |  | |

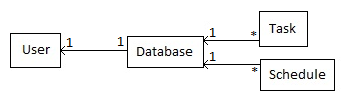
12.5.4.2 Component Architecture Diagram

Component Architecture Diagram.png

12.5.5 Class Diagrams

https://lh4.googleusercontent.com/_4D2Keae9f_D4NwzpJVujh4onCYt3TUf4edd_PSQwmGqo3kliC0x-JpxpWa-I1qYrU7v-lLpiWkt0SKu7hrTXJJ_xUtHmljteUFXf1CmOZRNyN48zrzpTieTVwfGT1a0-JLzHXvj

12.5.6 Class Relationship Diagram



12.5.7.2 Event Diagrams

https://lh5.googleusercontent.com/wk0AI1jS9UmjnTba0_KkKkxVIg7vUaYLKdkkC-L8xSkCOmC8gWGIrlnIceH-wQ44D8D6494CqEb74KKFP214JTrq5VBwYXOEOpQtnAlVQwgEKbPimc3KOlgqCnbhWF6MXs8TUOiq

12.5.8 Activity / State Section

https://lh4.googleusercontent.com/8OQ3yMwEdHP7bE7Ki92jf38-h8EM64vEp4bu2ibQRL_fJOiJYLwbTuatQHCrQGPjvY4FA_WKm9ZILF4ftmQmzFMFn98Sf7LwKCqFJZFPioKcunAqF7PkPCRo-HlqwlJFqLWmjYS7

12.5.10.1 Sequence Diagrams

E:\School\College\8 - Spring 2016\1 - CS 4513 - Software Engineering\Project\3 - SAS\Sequence Diagram.jpg