https://lh6.googleusercontent.com/ZbSIWMo4cow4QNJGA6BveANighDY2_y1CMANa5citrT0ZKaBLxYbxfT-zNV4nV6jZSe7GbYkxIvsv2UQbzEC92oVcW-jMaPgbCVKsddyyd-NH5iW-5ids8pNvEGYhTEQ97jAA1bN

**Computer Science and Engineering**

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

**Taskr**

**System Project Management Plan**

**Version 1.1**

Document Number: RAS-001

Project Team Number: B12

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 | 10/01/2016 | Franky Cen |
| Kenan Millet | Author | 10/03/2016 | Kenan Millet |
| Yatin Kaushal | Reviewer | 10/03/2016 | Yatin Kaushal |
| Franky Cen | Reviewer | 10/03/2016 | Franky Cen |
| Kenan Millet | Reviewer | 10/03/2016 | Kenan Millet |
| Franky Cen | Approval | 10/04/2016 | Franky Cen |
| Yatin Kaushal | Author | 10/10/2016 | Yatin Kaushal |
| Franky Cen | Author | 10/10/2016 | Franky Cen |
| Kenan Millet | Author | 10/10/2016 | Kenan Millet |
| Franky Cen | Approval | 10/10/2016 | Franky Cen |

**REVISION LEVEL**

|  |  |  |
| --- | --- | --- |
| **Date** | **Revision Number** | **Purpose** |
| 10/01/2016 | Version 1.0 | Initial creation of the document |
| 10/10/2016 | Version 1.1 | Revision of the document |

**TABLE OF CONTENTS**

**1.   Overview ……………………….………………………………………………………5**

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

**2.   Scope ……………………………………………………………………………………5**

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

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

2.3 Objectives ……………………………………………………………………………5

2.4 System Overview ……………………………………………………………………5

2.5 Document Overview …………………………………………………………………6

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

**4.   Business Requirements ………………………………………………………………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 ………………………………………………………8

**5.   Context Diagram ………………………………………………………………………8**

5.1   High Level ……………………………………………………………………………8

**6.   Functional Requirements Analysis Specification ……………………………8**

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

6.2   System Capability Requirements ……………………………………………………10

6.2.1 Capabilities …………………...………………………………………………10

6.3   User Interface Requirements …………………………………………………………10

6.4   Component Architecture ………………….…………………………………………10

6.4.1 Component Descriptions ……………………………………………………10-12

6.4.2 Component Architecture Diagrams ……………………………………………12

6.5 Class Diagrams ………………………………………………………………………12

6.6 Class Relationship/Interaction Diagrams ……………………………………………12

6.7 Event Section ………………………………………………………………………12

6.7.1 Event Dictionary ………………………………………………………………12

6.7.2 Event Diagrams ………………………………………………………………12

6.8 Activity/State Section ………………………………………………………………12

6.8.1 Activity Diagrams ………………………………………………………………12

6.8.2 Activity Specifications ………………………………………………………12-13

6.9 Sequence Diagrams …………………………………………………………………13

6.10 Collaboration Diagrams ………………………………………………………………13

6.11 Dictionaries …………………………………………………………………………13-15

**7.   Non-Functional/Operational Requirements ……………………………………15**

7.1   System External Interface Requirements ……………………………………………15

7.2   Safety Requirements ……………………….………………………………………15

7.3   Security and Privacy Requirements …………………………………………………15

7.4   System Environment Requirements …………………………………………………15

7.5   Computer Resource Requirements …………………………………………………15

7.5.1 Computer Hardware Requirements ……………………………………………15

7.5.2 Computer Hardware Resource Requirements …………………………………15

7.5.3 Computer Software Requirements ……………………………………………16

7.5.4 Computer Communications Requirements ……………………………………16

7.6   System Quality Factors ……………………………………………………………16

7.7   Design and Construction Constraints …………………………………………………16

7.8   Personnel-Related Requirements ……………………………………………………16

7.9 Training-Related Requirements ………………………………………………………16

7.10 Logistics-Related Requirements ……………………………………………………16

7.11 Packaging Requirements ……………………………………………………………16

**8.   System Test Plan Requirements ……………………………………………………17**

**9.   Qualification Provisions ……………………………………………………………17**

**10. Requirements Traceability …………………………………………………………17**

**11. Rationale …………………………………………………………………………………17**

**12.   Appendix ………………………………………………………………………………18**

12.1   Schedule Tracking …………………………………………………………………18

12.2   Defect Tracking ………………………………………………………………………19

12.5.1 Context Diagram ……………………………………………………………………20

12.6.2.1 Use Case Diagrams and Use Case Descriptions …………………………20-31

12.6.4.2 Component Architecture Diagram ……………………………………………32

12.6.5 Class Diagrams ……………………………………………………………………33

12.6.6 Class Relationships / Interaction Diagram …………………………………………33

12.6.7.2 Event Diagrams …………………………………………………………………34

12.6.8.1 Activity Diagrams …………………………………………………………………35

12.6.9 Sequence Diagram …………………………………………………………………36

**1.   Overview**

**1.1   Purpose**

The purpose of this document is to analyze the requirements stated in the SRS and SAS. Information from both of those documents is used in this document. The requirements of the project are further analyzed in this document.

**2.   Scope**

**2.1 Identification**

Taskr Requirements Analysis Specification v1.1

**2.2 Bounds**

The users will interact with the system by making requests and receiving formatted data in the form of schedules and tasks. The system will interact with the database, which will store the data relevant to each user in an unformatted, but organized manner.

**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 document was delivered on 03/07/16.

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

The initial Software Project Management Plan was delivered on 04/6/16.

The Software Analysis Specifications document was delivered on 04/11/16.

The Software Design document was delivered on 04/25/16.

The final Software Project Management Plan was delivered on 09/27/2016.

The Requirements and Analysis Specifications will be delivered on 10/04/2016.

The Software Design Description will be delivered on 10/25/2016.

The Design Document will be delivered on 11/25/2016.

**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.

**2.5 Document Overview**

Each section of this document is detailed below:

Scope: The boundaries within which the system will operate as well as the objectives within the project.

Reference Documents: The documents which are referenced by this one.

Business Requirements: The business-related requirements and concerns for the development and marketing solutions for the project.

Context Diagram: A diagram showing the input/output relationship between the various parts of the system.

Functional Requirements Analysis Specification: The specification of requirements detailing what the program should be able to do.

Non-Functional/Optional Requirements: The specification of requirements about the operation of the system rather than those covered in the Functional Requirements.

System Test Plan Requirements: The plan for how the system will be tested during development and after completion.

Qualification Provisions: The criteria that will be used to judge the accuracy and purpose of this document.

Requirements Traceability: Details on how the requirements of this document and the system will be traced over each iteration/revision.

Rationale: The reason for creating the product.

**3.   Reference Documents**

Taskr Project Proposal v1.0 02/17/2016

Objectives, Rationale, Bounds

Taskr Software Requirements Specification v1.1 03/23/2016

Business Requirements

Functional and Nonfunctional Requirements

System Test Plan Requirements

Taskr System Analysis Specification v1.0 04/11/2016

Class Diagrams

Context Diagrams

Architecture Diagrams

Dictionaries

Event Diagrams

**4.   Business Requirements**

**4.1   Technology**

Taskr is an effective tool for any business to apply in their workplace. By using Taskr as the default application for scheduling and time management, your business 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**

Not required

**4.4 Market Considerations**

Taskr is marketed towards college students, career professionals, and businesses who have several tasks of varying priorities and require assistance in effective time management.

**4.5 Risks and Alternatives**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Risk** | **Priority** | **Detection Method** | **Risk Responder** | **Solution** |
| Economic Risk: Lack of demand for the product. | High | Lack of sales | Project Manager, Developers | Find a way to increase interest in the product. |
| Technical Risk: System malfunctions | Varies | Customer comments | Project Manager, SQA group | Fix faults, ensure that the product works correctly. |
| Financial Risk:  Product dev. costs surpasses budget | Medium-High | Project behind schedule and/or extra costs | Project Manager | Re-establish minimum viable product |
| Environmental Risk: User environment is unintuitive | Medium-High | Quality Assurance Testing | SQA group, Developers | Re-design user environment, update documentation accordingly |

**4.6 Human Resources and Training**

Developers will 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.   Context Diagram**

**5.1   High Level**

See Appendix 12.5.1 for Context Diagram

**6.   Functional Requirements Analysis Specification**

**6.1   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.

9.4 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.

**6.2   System Capability Requirements**

**6.2.1 Capabilities**

See Appendix 12.6.2.1 for Case Diagrams

**6.3   User Interface Requirements**

Android OS mobile (other requirements will be available at a later point)

**6.4   Component Architecture**

**6.4.1 Component Descriptions**

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.

**6.4.2 Component Architecture Diagram**

See Appendix 12.6.4.2 for the Component Architecture Diagram

**6.5 Class Diagrams**

See Appendix 12.6.5 for Class Diagrams

**6.6 Class Relationship/Interaction Diagram**

See Appendix 12.6.6 for the Class Relationships / Interaction Diagram

**6.7 Event Section**

**6.7.1 Event Dictionary**

* 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.

**6.7.2 Event Diagrams**

See Appendix 12.6.7.2 for Event Diagrams

**6.8 Activity/State Section**

**6.8.1 Activity Diagrams**

See Appendix 12.6.8.1 for Activity Diagrams

**6.8.2 Activity Specifications**

When the user requests to generate a schedule, the system will ask which tasks or events to add to the new schedule. Using an algorithm, the system will generate a schedule using parameters set by the user.

When the user requests to view all tasks, the system will query the database and present all the tasks in a list.

When the user requests to create a task, the system will ask for the name, start time, duration, priority, and other relevant information of the task. The task is added to the database.

When the user requests to delete a task, the system confirms this request before taking action. If confirmed, the task will be removed from the database and from any schedule that it is in.

When the user requests to modify a task, the system will ask the user which piece of information to modify. The user can also add additional information to tasks. The changes are saved in the database.

When the user requests to place a task in a schedule, the system will display the list of schedules and will ask the user to select one. When a schedule is selected, the system will ask the user to place the task in an timeslot. The schedule change is saved in the database.

When the user requests to view all schedules, the system will query the database and present all schedules in a list.

When the user requests to create a schedule, the system will ask for the name, priority, frequency, and other relevant information of the schedule. The system will then ask the user to add tasks to this empty schedule. The schedule is saved in the database.

When the user requests to delete a schedule, the system confirms this request before taking action. If confirmed, the schedule is removed from the database.

When the user requests to modify a schedule, the system will ask the user which piece of information to modify. The user can also add additional information to schedules. The user can move the timeslots of tasks around in a database. The changes are saved in the database.

**6.9 Sequence Diagrams**

See Appendix 12.6.9 for Sequence Diagrams

**6.10 Collaboration Diagrams**

To be completed at a later date.

**6.11 Dictionaries**

Classes: User, Server, Database, Task, Schedule

Attributes: User: Name

Database: List of Tasks

List of Schedules

Server: Database

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()

Server, 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-Server: one to one relationship

Server-Database: one to one relationship

Database-Task: one to many relationship

Database-Schedule: one to many relationship

**7.   Non-Functional/Operational Requirements**

**7.1   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 (Descriptions located in the SRS)

**7.2   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 the next section: 7.3.

**7.3   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.

**7.4   System Environment Requirements**

The environment of the system is a digital operating system. The development and testing environments will be managed by the project team members. Each team member will have a separate personal environment. Programming tools will be used to develop and test the product. Programmers and developers will have access to all product environments. These environments will be checked to make sure they comply with each other, and that they are adequate for the product. These environments will be maintained by the project team members.

**7.5   Computer Resource Requirements**

**7.5.1 Computer Hardware Requirements**

The system shall be operated from the mobile phone of the user. The device operating the system will need to have capabilities of touch-interaction.

**7.5.2 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 should be resource intensive.

**7.5.3 Computer Software Requirements**

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

**7.5.4 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.

**7.6   System Quality Factors**

The system quality factors include the speed and helpfulness of the algorithm used. If the algorithm runs quickly, then it gets results quickly, and will not force the user 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.

**7.7   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.

**7.8   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.

**7.9 Training-Related Requirements**

Programmers and developers are required to have knowledge of programming and general programming practices.

**7.10 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.

**7.11 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.

**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 on the Use Cases, 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. 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.

**11. 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.

**12.   Appendices**

**12.1   Schedule Tracking**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SRS - Initial | Franky  Kenan  Yatin | 14 hours  12 hours  15 hours | 8 hours  14 hours  7 hours | 6 hours  2 hours  8 hours |
| **Total** | **41 hours** | **29 hours** | **12 hours** |
| SRS - Final | Franky  Kenan  Yatin | 15 hours  20 hours  15 hours | 9 hours  6.5 hours  5 hours | 6 hours  12.5 hours  10 hours |
| **Total** | **50 hours** | **20.5 hours** | **29.5 hours** |
| SPMP - Initial | Franky  Kenan  Yatin | 13 hours  18 hours  10 hours | 16.5 hours  12 hours  7 hours | 3.5 hours  6 hours  3 hours |
| **Total** | **41 hours** | **35.5 hours** | **5.5 hours** |
| SAS | Franky  Kenan  Yatin | 15 hours  12 hours  15 hours | 13 hours  7.5 hours  8 hours | 2 hours  4.5 hours  7 hours |
| **Total** | **42 hours** | **28.5 hours** | **13.5 hours** |
| SPMP - Final | Franky  Kenan  Yatin | 8 hours  3 hours  2 hours | 4 hours  2 hours  1 hours | 4 hours  1 hour  1 hour |
| **Total** | **13 hours** | **7 hours** | **6 hours** |
| RAS | Franky  Kenan  Yatin | 10 hours  4 hours  3 hours | 5 hours  4 hours  2.25 hours | 5 hours  0 hours  0.75 hours |
| **Total** | **17 hours** | **11.25 hours** | **5.75 hours** |

**Cumulative**

|  |  |  |  |
| --- | --- | --- | --- |
| **Individual** | **Estimated** | **Actual** | **Difference** |
| Franky  Kenan  Yatin | 75 hours  69 hours  60 hours | 55.5 hours  45.75 hours  29.5 hours | 19.5 hours  23.25 hours  30.5 hours |
| **Total** | **204 hours** | **128.5 hours** | **75.5 hours** |

**12.2   Defect Tracking**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Artifact / Deliverable** | **Individual** | **Estimated** | **Actual** | **Difference** |
| SRS - Initial | Franky  Kenan  Yatin | 20 faults  10 faults  10 faults | 10 faults  6 faults  3 faults | 10 faults  4 faults  7 faults |
| **Total** | **40 faults** | **19 faults** | **21 faults** |
| SRS - Final | Franky  Kenan  Yatin | 20 faults  20 faults  15 faults | 24 faults  2 faults  3 faults | 4 faults  18 faults  12 faults |
| **Total** | **55 faults** | **29 faults** | **26 faults** |
| SPMP - Initial | Franky  Kenan  Yatin | 20 faults  20 faults  20 faults | 16 faults  17 faults  4 faults | 4 faults  3 faults  16 faults |
| **Total** | **60 faults** | **37 faults** | **23 faults** |
| SAS | Franky  Kenan  Yatin | 15 faults  20 faults  18 faults | 13 faults  12 faults  15 faults | 2 faults  8 faults  3 faults |
| **Total** | **53 faults** | **40 faults** | **13 faults** |
| SPMP - Final | Franky  Kenan  Yatin | 15 faults  10 faults  10 faults | 10 faults  4 faults  3 faults | 5 faults  6 faults  7 faults |
| **Total** | **35 faults** | **15 faults** | **18 faults** |
| SPMP - Final | Franky  Kenan  Yatin | 15 faults  20 faults  5 faults | 14 faults  12 faults  1 fault | 1 fault  8 faults  4 faults |
| **Total** | **40 faults** | **27 faults** | **13 faults** |

**Cumulative**

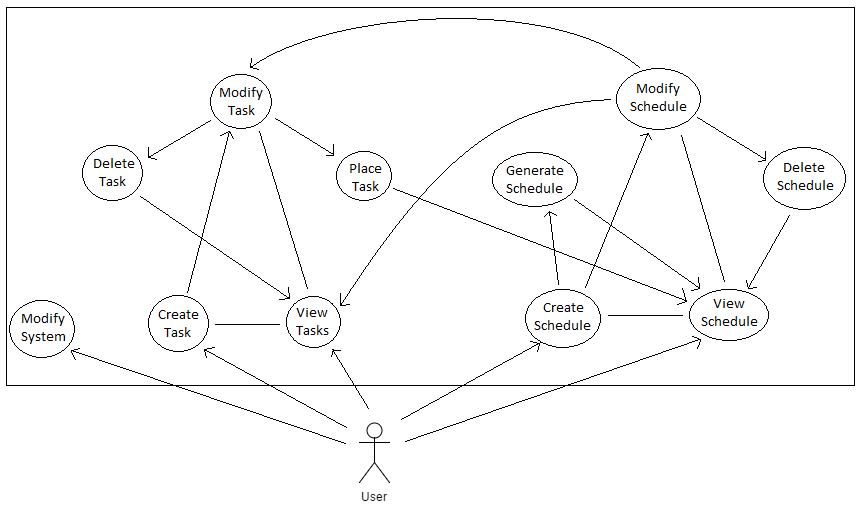
|  |  |  |  |
| --- | --- | --- | --- |
| **Individual** | **Estimated** | **Actual** | **Difference** |
| Franky  Kenan  Yatin | 105 faults  100 faults  73 faults | 87 faults  53 faults  28 faults | 18 faults  47 faults  45 faults |
| **Total** | **283 faults** | **169 faults** | **114 faults** |

**12.5.1 Context Diagram**



**12.6.2.1 Use Case Diagram and Use Case Descriptions**

**Use Case Diagram:**



**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** |  | |

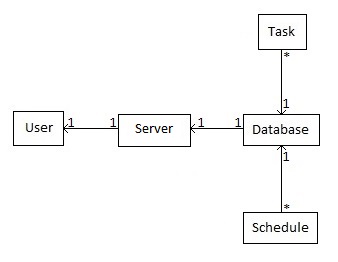
**12.6.4.2 Component Architecture Diagram**

Component Architecture Diagram.png

**12.6.5 Class Diagrams**



**12.6.6 Class Relationship / Interaction Diagram**



Users have a one to one relationship with the Server.

The Server has a one to one relationship with the Database.  
The Database has a one to many relationship to both Tasks and to Schedules.

**12.6.7.2 Event Diagram**



**12.6.8.1 Activity Diagrams**



**12.6.9 Sequence Diagram**

