Software Design Description (SDD)

Document Number: 01

Date: 26, April 2018

*DelegationOfTask Module*

Paul Miceli

Professor Tepfenhart

Software Engineering Department

Monmouth University

West Long Branch, NJ 07764-1898

Table of Contents

1 Scope 1

1.1 Identification 1

1.2 System Overview 1

1.3 Document Overview 1

2 References 1

3 Software/Hardware Item-Wide Design Decisions 1

4 Software Item Architectural Design 3

4.1 Software Item Components 3

4.2 Concept of Execution 5

4.3 Interface Design 5

4.3.1 Interface Identification and Diagrams 5

4.3.2 [Project-unique Identifier of Interface] 6

5 Software Item Detailed Design 9

5.1 [Project-unique Identifier of a Software Unit or Designator of a Group of Software Units] 9

6 Requirements Traceability 11

7 Notes 11

# Scope

## Identification

This document details the DelegationOfTask Module for the GettingThingsDone Project.

## System Overview

This system implements the GettingThingsDone technique [1] for task and time management. While it is generally accepted that the only thing necessary for following the GettingThingsDone process is a sheet of paper and pencil, this system is intended to provide additional tools to automate some of the analysis elements of the GettingThingsDone.

The key functionalities will be a Task list and a timer along with management and analysis tools. The system will utilize a calendar for tracking Task items with deadlines.

## Document Overview

This document will present the design for the DelegationOfTask Module for the GettingThingsDone system. This, section 1, gives a scope for the overall document. Section 2 is the references section which identifies the descriptive documents for the GettingThingsDone and the module process along with documents specifying the standards on which the design is based. Section 3 identifies Software/Hardware Design Decisions that need to be made.

Section 4 presents the architectural design for the module. Section 4.1 presents the architecture in terms of the components using UML diagrams and textual specifications of each component functionality. Section 4.2 presents the concept of execution, which demonstrates how the various components work together to meet the requirements. Section 4.3 provides the detailed description of the interfaces used between components and to external systems.

Section 5 provides the detailed design of the module. The module will take what was said in Section 4 and describe the components in more depth.

# References

|  |  |
| --- | --- |
| [1] | K. H. P. M. a. H. T. H. Dubel, *System Design Document For GettingThingsDone System,* 2018. |
| [2] | D. Allen, Getting Things Done - The Art of Stress-Free Productivity, New York: Penguin, 2001. |

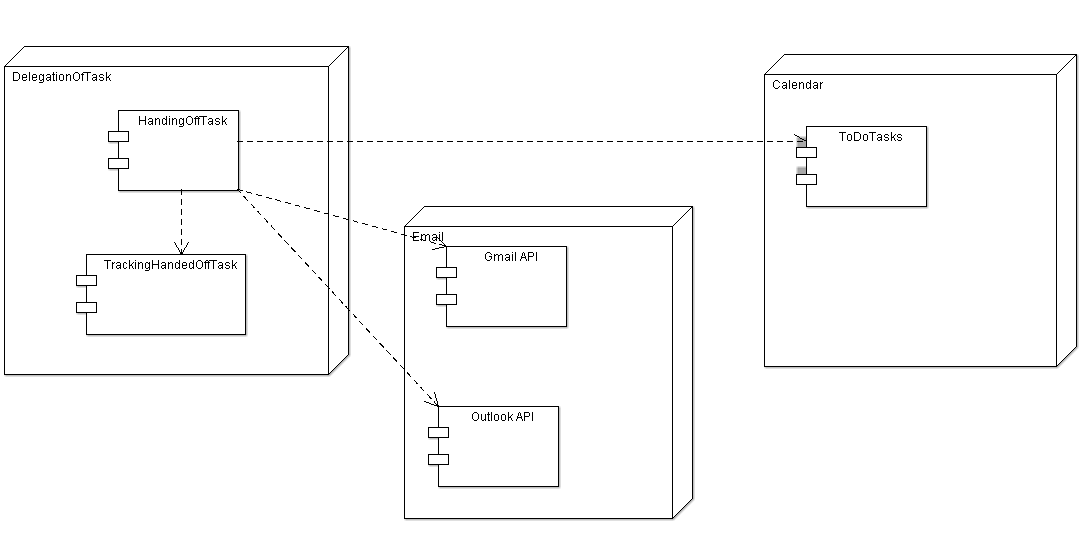
# Software/Hardware Item-Wide Design Decisions

None.

# Software Item Architectural Design

This section of the document specifies the architecture for the DelegationOfTask module under the Webservice Scripts package in the GettingThingsDone System in terms of hardware and software components, the concept of operations, and the interfaces within the system.

## Software Item Components

­

The DelegationofTask component will handle sending the emails to those whom were delegated to handle the task. It will utilize the GMail and Outlook API’s for sending out and receiving emails.

It is a part of the Webservice Scripts package in the GettingThingsDone System.

### DelegationOfTask

There are two methods that handle the DelegationOfTask to a user.

1. HandingOffTask – Identifying the person that will be doing the task
2. TrackingHandedOffTask – Figuring out the status of the task that the user is responsible for.

#### HandingOffTask

The task will be handed off by emailing the person that is responsible for the task. This will be done via the Gmail and Outlook API’s.

#### TrackingHandedOffTask

This tracks the handoff to the other person to do the task. Once the email is sent to the user that is to be doing the task the module will need to organize the tasks. Categories such as “Pending”, “Waiting For”, and “Completed”. “Waiting For” is when the User is responsible for replying on an update for the task. The update can come in the form of an email, in this case the system will need to update to show the current status of the task. “Pending” is the state that will come after “Waiting For” this is the state that shows that the user responsible for the task needs to update on the status. “Completed” is the state when the task is finally completed.

### Gmail API

This module utilizes the Gmail API provided by Google. This assists the system with the emailing the user with the task that they are responsible for.

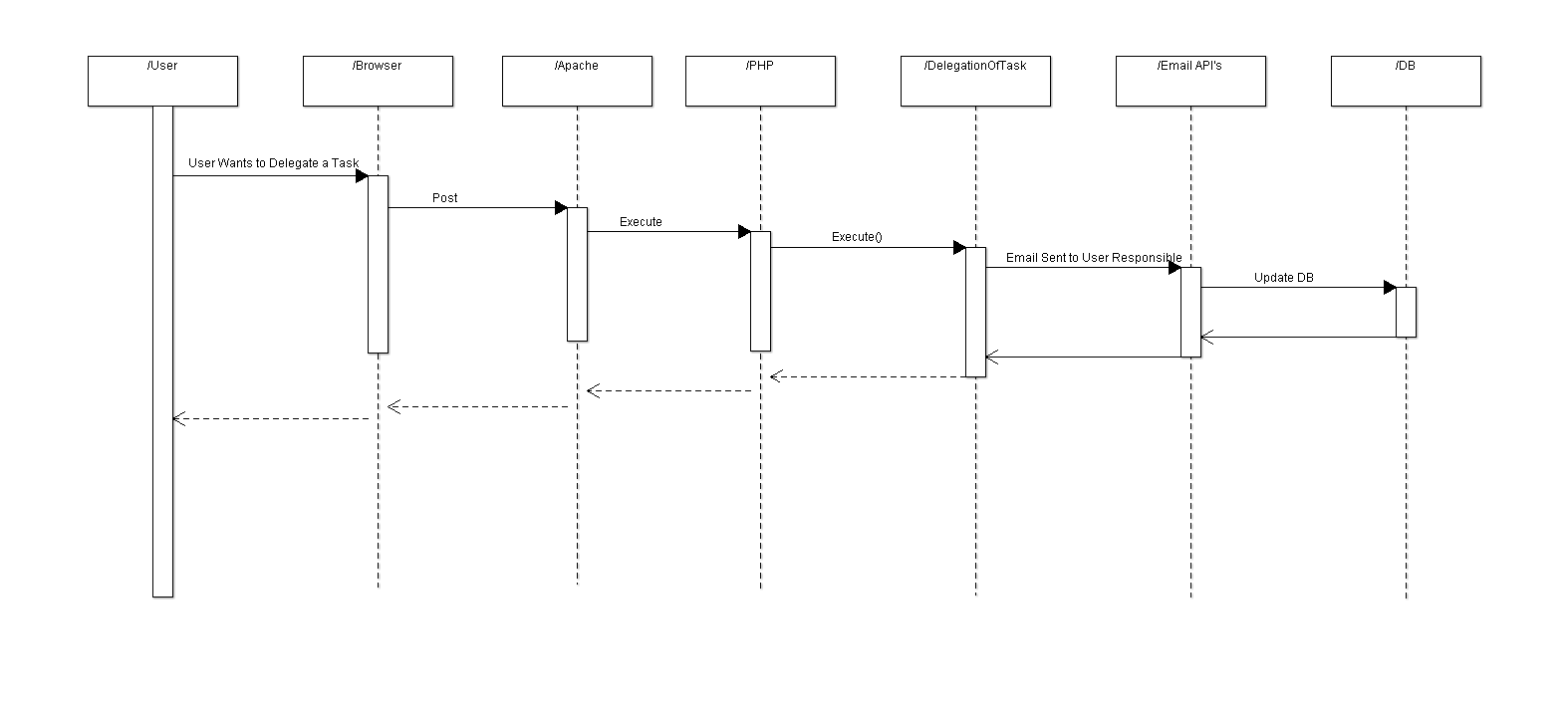
### Outlook API

This module utilizes the Outlook Rest API. To assist with the emailing of the tasks that the user is assigned to. Specifically assisting with the email accounts that are associated with Exchange Online, Office 365, Hotmail, Live, MSN, Outlook and Passport Accounts.

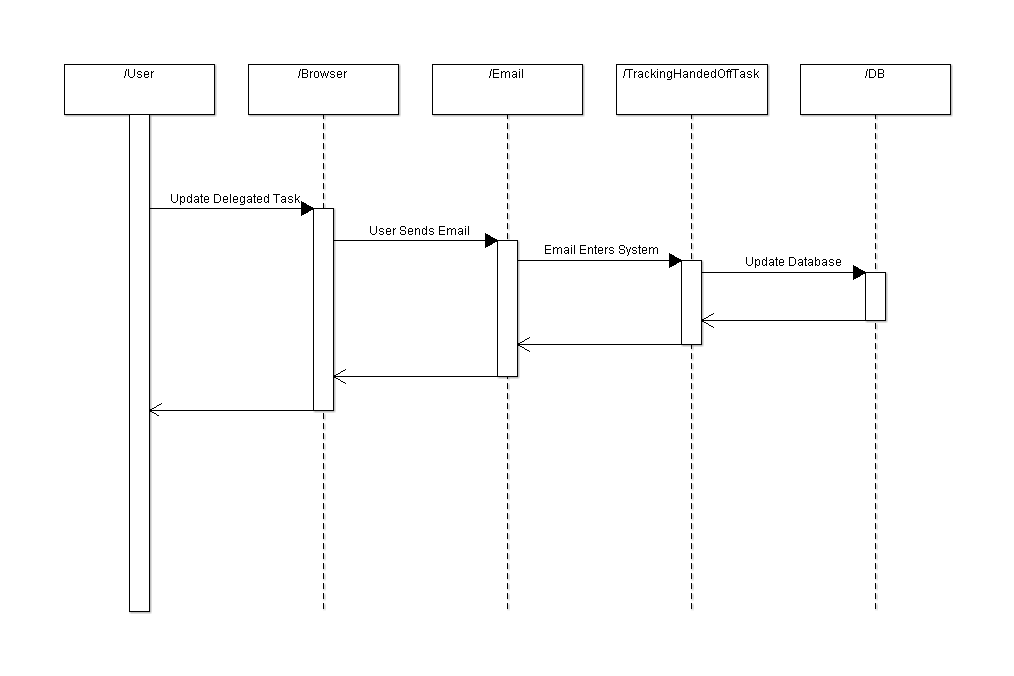
### ToDoTasks

The DelegationOfTask Module needs access to the ToDoTasks to identify the tasks that need to be done, which are in the Calendar.

## Concept of Execution



This diagram depicts the execution of when a user wants to delegate a task to a user. The chain moves from the Apache and PHP connection to talk to the database to executing the DelegateTask() script and sending the corresponding email via the Email API’s, then updating the Database with the new information.



This diagram depicts the TrackingHandedOffTask script. This will allow task to be updated via email. The chain moves from the user to executing the TrackingHandedOffTask script and sending the corresponding email via the Email API’s, then updating the Database with the new information. The new status will be the output of this sequence diagram.

## Interface Design

### Interface Identification and Diagrams

The following are the interfaces:

* DelegationOfTask to the Database
* DelegationOfTask to the Email API’s
* Apache server

### Apache

Apache server that launches PHP and activates the script and interacts with the database.

### Database

DelegationOfTask to the Database

### Email API

# Software Item Detailed Design

[This section shall be divided into the following paragraphs to describe each software unit of the software item. If part or all of the design depends upon system states or modes, this dependency shall be indicated. If design information falls into more than one paragraph, it may be presented once and referenced from the other paragraphs. Design conventions needed to understand the design shall be presented or referenced. Interface characteristics of software units may be described here, in Section 4, or in Interface Design Descriptions (IDDs). Software units that are databases, or that are used to access or manipulate databases, may be described here or in Database Design Descriptions (DBDDs).]

DelegationForm

## DelegationOfTask

Purpose of this node is to determine where tasks will be dealt with. If the action will be taking more than two minutes of time then the task will be delegated and the status of the task is then “Waiting for”; waiting for a person to do the task.

(page 139 flowchart)

### HandedOffTask

Algorithm (flowchart) – The user will go on the form to delegate the task, then the user will pick a person, the person that was chosen will get an email

This will be programmed in PHP, Sql, HTML/CSS

### TrackingHandedOffTask

Algorithm (flowchart) – The user that delegated the task will be “Waiting For” or “Pending”, “Someday/Maybe” list

This will be programmed in PHP, Sql, HTMl/CSS.

[This paragraph shall identify a software unit by project-unique identifier and shall describe the unit. The description shall include the following information, as applicable. Alternatively, this paragraph may designate a group of software units and identify and describe the software units in subparagraphs. Software units that contain other software units may reference the descriptions of those units rather than repeating information.

1. Unit design decisions, if any, such as algorithms to be used, if not previously selected.
2. Any constraints, limitations, or unusual features in the design of the software unit.
3. The programming language to be used and rationale for its use if other than the specified software language.
4. If the software unit consists of or contains procedural commands (such as menu selections in a database management system (DBMS) for defining forms and reports, on-line DBMS queries for database access and manipulation, input to a graphical user interface (GUI) builder for automated code generation, commands to the operating system, or shell scripts), a list of the procedural commands and references to user manuals or other documents that explain them.
5. If the software unit contains, receives, or outputs data, a description of its inputs, outputs, and other data elements and data element assemblies, as applicable. Paragraph 4.3.x of this DOCUMENT provides a list of topics to be covered, as applicable. Data local to the software unit may be described separately from data input to or output from the software unit. If the software unit is a database, a corresponding Database Design Description (DBDD) shall be referenced; interface characteristics may be provided here or by referencing Section 4 or the corresponding Interface Design Description(s).
6. If the software unit contains logic, the logic to be used by the software unit, including, as applicable:
7. Conditions in effect within the software unit when its execution is initiated.
8. Conditions under which control is passed to other software units.
9. Response and response time to each input, including data conversion, renaming, and date transfer operations.
10. Sequence of operations and dynamically controlled sequencing during the software unit’s operation, including:
11. The method for sequence control
12. The logic and input conditions of that method, such as timing variations, priority assignments
13. Data transfer in and out of memory
14. The sensing of discrete input signals, and timing relationships between interrupt operations within the software unit
15. Exception and error handling.]

## Email

Gmail API and Outlook API are responsible for the outgoing emails when a task is delegated.

# Requirements Traceability

[This section shall contain:

1. Traceability from each software unit identified in this SDD to the software item requirements allocated to it. (Alternatively, this traceability may be provided in 4.1.).
2. Traceability from each software item requirement to the software units to which it is allocated.]

# Notes

[This section shall contain any general information that aids in understanding this document (e.g., background information, glossary, rationale). This section shall include an alphabetical listing of all acronyms, abbreviations, and their meanings as used in this document and a list of any terms and definitions needed to understand this document.]