SE 3XA3: Module Guide
Save The Date

Karuka Khurana (khurak1)
Utsharga Rozario (rozariou)
Samarth Kumar (kumars38)
Dhruv Cheemakurti (cheemakd)

March 18, 2022

Contents

List of Tables

List of Figures

Table 1: Revision History

Date	**Developer(s)**	**Change**
March 15, 2021	Karuka Khurana	Copy template & completed introductory section
March 16, 2021	Karuka Khurana	Completed section 5.
March 18, 2021	Karuka Khurana	Completed section 7, 8.
March 18, 2021	Utsharga Rozario	Completed section 7 Diagram.
March 18, 2021	Dhruv Cheemakurti	Completed section 2, 3, 4, 6.

1 Introduction

1.1 Overview

Save The Date is a re-development of an open-source Notion application using React and the Python programming language. Our project aims to further enhance the user experience by adding an additional feature to scrape an uploaded PDF document of a course outline and visualize outstanding dates that are important for students to keep in mind. We hope to promote organization and time management by efficiently displaying important deadlines rather than student's having to scroll through multiple syllabuses. On top of the original implementation, we will include a PDF upload feature, an option to provide a page range and the option to begin scraping the document. Once this is complete, a dashboard will be created that includes all important dates and deadlines.

1.2 Context

A Software Requirements Specification (SRS) document was created which outlined functional and non-functional requirements this project must fulfill. The purpose of this document is to give a high-level description of the structure of the system which is broken down into modules. Using a unified modelling language (UML) approach, this document will describe the system architecture. Decomposing code into modules will allow for a better understanding of how different aspects of the program communicate with each other to solve a problem. This document has the following sections:

- Section 2 outlines any anticipated and unlikely changes
- Second 3 outlines the overview of the module hierarchy
- Section 4 outlines the connections between requirements and design
- Section 5 includes the module decomposition
- Section 6 includes two traceability matrices, one for connections between SRS and modules and the another for connections between anticipated changes and modules.
- Section 7 includes a hierarchical diagram to visualize the relationship among the modules

1.3 Design Principle

Modular decomposition is beneficial for many different reasons. Decomposing larger modules into smaller ones ensures reusability, readability, and structure and allows for easy traceability when debugging. In this project, decomposition of the modules is based off of the principle of information hiding (Parnas, 1972). Each of the modules will hide a secret that can easily be changed.

Our design follows the rules layed out by (Parnas, 1972), as follows:

- System details that are likely to change independently should be the secrets of separate modules.
- Each data structure is used in only one module.
- Any other program that requires information stored in a module's data structures must obtain it by calling access programs belonging to that module.

2 Anticipated and Unlikely Changes

This sections lists possible changes for the application SaveTheDate. The changes that may occur for the development of an application can be categorized into anticipated and unlikely changes. Anticipated changes are planned or foreseeable whereas unlikely changes are those that were not planned at all. We describe the two categories below.

2.1 Anticipated Changes

AC1: The type of the output of PDFScraper module might change

AC2: The format of input data

AC3: The PDFScraper module might change as more functionality is added.

AC4: The functionality loadPDF module might change as more features will be added.

2.2 Unlikely Changes

UC1: There will be a source of input data external to software

UC2: The system interfacing with background application

UC3: Input/output devices

UC4: The data structures of PDFScraper module output.

3 Module Hierarchy

3.1 PDF Scraping subsystem

M1 PDFScraper Module

M2 ScrapePDFUser module

M3 loadPDF module

M4 image module

3.2 Table generation subsystem

M5 editableTable module

M6 Table module

M7 Cell module

M8 Text module

M9 Color module

M10 Header module

M11 Relationship module

M12 util's module

3.3 Page generation subsystem

M13 button module

M14 selectMenu module

M15 editablePage module

M16 editableBlock module

M17 Uid module

M18 CaretHelpers module

4 Connection Between Requirements and Design

The system design is meant to fulfill all the requirements that were outlined in the SRS. In this stage, the system is decomposed into modules and Table 3 lists the connection between requirements and modules.

5 Module Decomposition

The **Secrets** field in a module decomposition is a brief statement of the design decision hidden by the module. The **Services** field specifies what the module will do without documenting how to do it. For each module, a suggestion for the implementing software is given under the **Implemented By** title. If the entry is **OS**, this means that the module is provided by the operating system or by standard programming language libraries.

5.1 Hardware Hiding Modules

Secrets: The data structure and algorithm used to implement the virtual hardware.

Services: Serves as a virtual hardware used by the rest of the system. This module provides the interface between the hardware and the software. So, the system can use it to display outputs or to accept inputs.

Implemented By: OS

5.2 Behaviour-Hiding Module

5.2.1 Table Module (M6)

Secrets: Format and structure of the Notion Table

Services: Renders a table within a Notion page

Implemented By: Table.js

5.2.2 Relationship Module (M9)

Secrets: Format and structure of the relationship in a cell of a table.

Services: Renders a relationship within a table on a Notion page.

Implemented By: Relationship.js

5.2.3 Text Module (M10)

Secrets: Format and structure of the text within in a cell of a table.

Services: Returns a colour as a string within a table on a Notion page.

Implemented By: Text.js

5.2.4 Colors Module (M11)

Secrets: Format and structure of colors within in a cell of a table.

Services: Renders a color relationship within a table on a Notion page.

Implemented By: colors.js

5.2.5 uid Module (M12)

Secrets: Format and structure of uid generated for EditablePage and EditableBlock.

Services: Renders a uid and outputs a string to other functions.

Implemented By: uid.js

5.2.6 caretHelpers Module (M13)

Secrets: Format and Structure of determining coordinates for EditableBlock module.

Services: Retrieves and sets the coordinates.

Implemented By: caretHelpers.js

5.2.7 utils Module (M14)

Secrets: Format and Structure of determining IDs and random colours.

Services: Generates a string output for a short ID and a random colour.

Implemented By: utils.js

5.3 Software Decision Module

5.3.1 editablePage Module (M1)

Secrets: Format and Structure of a single Notion page.

Services: Represents the functions that can be executed on a Notion page which includes editing, adding, and deleting.

Implemented By: editablePage.js

5.3.2 editableBlock Module (M2)

Secrets: Format and Structure of a block on a single Notion page.

Services: Represents the functions that can be executed within a block on a Notion page which includes editing, adding, and deleting.

Implemented By: editableBlock.js

5.3.3 selectMenu Module (M3)

Secrets: Format and Structure of a menu bar on a single Notion page.

Services: Represents the functions that can be executed within a menu on a Notion page which includes selecting.

Implemented By: selectMenue.js

5.3.4 button Module (M4)

Secrets: Format and Structure of a button on menu bar on a single Notion page.

Services: Represents the functions that can be executed on a block within a Notion page.

Implemented By:

5.3.5 editableTable Module (M5)

Secrets: Format and Structure of a table on a single Notion page.

Services: Represents the functions that can be executed within a menu on a Notion page which includes adding, editing, deleting.

Implemented By: editableTable.js

5.3.6 Header Module (M7)

Secrets: Format and Structure of a header of a table on a single Notion page.

Services: Represents the functions that can be executed within a menu on a Notion page which includes selecting.

Implemented By: Header.js

5.3.7 Cell Module (M8)

Secrets: Format and Structure of a cell of a table on a single Notion page.

Services: Represents the functions that can be executed within a menu on a Notion page which includes adding, editing, deleting.

Implemented By: Cell.js

5.3.8 loadPDF Module (M15)

Secrets: Accepts the uploaded PDF file.

Services: Runs through the PDF file to load it in order to scrape the document.

Implemented By: loadPDF.js

5.3.9 scrapePDFUser Module (M16)

Secrets: Format and Structure of a pop-up window.

Services: Allows a user to add a course name and select a PDF file from the pop-up window.

Implemented By: scrapePDFUser.js

5.3.10 image Module (M17)

Secrets: Format and Structure of an image on a Notion page.

Services: Represents the functions that can be executed on an image on a Notion page which includes adding the image and rendering it.

Implemented By: image.js

5.3.11 PDFScraper Module (M18)

Secrets: Algorithm to scrape a PDF document.

Services: Handles obtaining important dates and task information from the uploaded PDF file.

Implemented By: scraper.py

6 Traceability Matrix

This sections shows two traceability matrices: one between modules and requirements, and another between modules and anticipated changes.

Req	Modules
FR1	4
FR2	15
FR3	18
FR4	6, 8, 10
FR5	6
FR6	6
FR7	5
FR8	1
FR9	15
FR10	15
FR11	27
FR12	4
LF1	1
LF2	1
UH1	4
UH2	3,4
UH3	1
PE1	16
PE2	16
PE3	16
PE4	6
PE5	N/A

Table 2: Trace Between Requirements and Modules

7 Use Hierarchy Between Modules

Figure ? visualizes the use relation between the modules. It can be seen that the graph is a directed acyclic graph (DAG). Each level of the hierarchy offers a testable and usable

AC	Modules
AC1	18
AC2	15
AC3	N/A
AC4	15

Table 3: Trace Between Anticipated Changes and Modules

subset of the system, and modules in the higher level of the hierarchy are essentially simpler because they use modules from the lower levels.

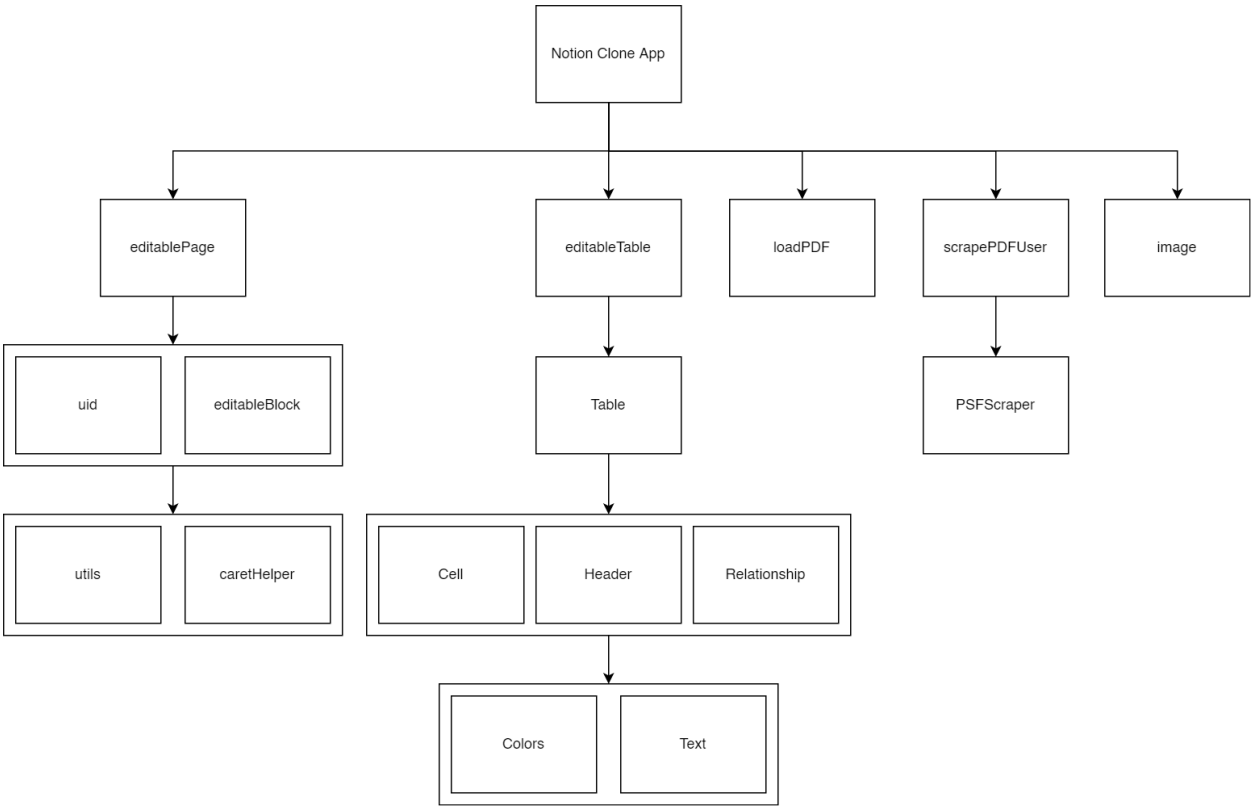

Figure 1: Use Hierarchy Among Modules

8 Schedule

Implementation dates of the modules along with the team members responsible are available in the Gantt chart.

Gantt Chart: https://gitlab.cas.mcmaster.ca/se3xa3_l03_g17/se3xa3_l03_g17/-/blob/main/ProjectSchedule/3XA3_L03_G17_GanttChart.pdf

9 References

David L. Panas. On the criteria to be used in decomposing systems into modules. Comm.ACM, 15(2):1053 1058, December 1972

D.L. Parnas, P.C. Clement, and D. M. Weiss, The modular structure of complex systems. In International Conference on Software Engineering, pages 408-419, 1984.