SE 3XA3: Module Guide
Euneva

Team 9, Euneva
Mehta, Jash - mehtaj8
Sharma, Aditya - shara24
Ren, Zackary - renx11

April 6, 2021

Contents

List of Tables

List of Figures

Date	Version	Notes
17/03/2021	1.0	Initial version of Module Guide

Table 1: **Revision History**

1 Introduction

1.1 Overview

The goal of this project is to create a To-Do list software capable of collecting updated assignment and quiz information from Avenue to Learn, and displaying this information in an organized manner. This would allow students to keep track of upcoming assignments and quizzes without having to search from class to class. The current To-Do list software is not capable of scraping Avenue to Learn, therefore the software is a very basic rendition of a To-Do list. This document aims to translate the functional and non-functional requirements as seen in the SRS into a direct programming scenario. This document outlines how the behaviour desired in the SRS will be implemented into the code of the game.

1.2 Scope

The final version of Euneva will implement the Avenue to Learn web scraping functionality as well as display the information in an organized and comprehensible manner.

1.3 Purpose

The purpose of this document is to provide a written, organized representation between the code design and the requirements. The software being developed has many working parts that all work simultaneously to create the desired functionality. As a result, this document will outline how the modules must interact and function, allowing the current development team to keep track of individual modules, as well as provide a map to future developers.

2 Anticipated and Unlikely Changes

This section lists possible changes to the system. According to the likeliness of the change, the possible changes are classified into two categories. Anticipated changes are listed in Section 2.1, and unlikely changes are listed in Section 2.2

2.1 Anticipated Changes

Anticipated changes are the source of the information that is to be hidden inside the modules. Ideally, changing one of the anticipated changes will only require changing the one module.

AC1 : The platforms and devices on which the software is running.

AC2 : The appearance of the user interface.

AC3 : The user experience and flow through the app.

2.2 Unlikely Changes

The module design should be as general as possible. However, a general system is more complex. Sometimes this complexity is not necessary. Fixing some design decisions at the system architecture stage can simplify the software design. If these decisions should later need to be changed, then many parts of the design will potentially need to be modified. Hence, it is not intended that these decisions will be changed.

UC1 : Currently the scraping functionality is based on the latest release of Avenue to Learn, therefore if the website being scraped is changed, then the scraping functionality would have to be re-designed.

UC2 : Converting the application from a Model View Controller architecture to a different architectural style.

3 Module Hierarchy

This section presents an overview of the module design. Table 1 describes the modules in a hierarchy. The modules listed as leaves in the hierarchy tree are the modules that will actually be implemented.

M1 : GUI Module.

M2 : Login Module

M3 : Filter Courses Module.

M4 : Assignment Information Module.

M5 : Quiz Information Module.

M6 : Filtering Module.

Level 1	**Level 2**
Hardware-Hiding Module	GUI Module
Behaviour-Hiding Module	Login Module Assignment Information Module Quiz Information Module Filtering Module
Software Decision Module	Filter Courses Module

Table 2: Module Hierarchy

4 Connection Between Requirements and Design

The design of the system is intended to satisfy the requirements developed in the SRS. In this stage, the system is decomposed into modules. The connection between requirements and modules is listed in Table 3.

5 Module Decomposition

5.1 Hardware Hiding Modules

Secrets: The data structures and algorithms 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 accept inputs.

Implemented By: Euneva Development Team

5.1.1 GUI Module (M1)

Secrets: ~~Required behaviour of the application elements.~~React components which execute the expected behaviour of the application.

Services: Includes programs that provide eternally visible behaviour of the system as specified in the software requirements specification (SRS) documents. This module serves as a communication layer between the hardware-hiding module and the software decision module. The programs in this module will need to change if there are changes in the SRS.

Implemented By: Euneva Development Team

5.2 Behaviour-Hiding Module

5.2.1 Login Module (M2)

Secrets: ~~Gathering and using the users input information to log into Avenue to Learn~~React input field which collects user input as a string.

Services: Uses CSS elements to navigate through the website, placing the correct information given by the user into the correct sections of the website.

Implemented By: Euneva Development Team

5.2.2 Filter Courses Module (M3)

Secrets: ~~Filtering the course information for the current semester.~~Filtered course information of a user for the current semester as a JSON object.

Services: Uses CSS elements to navigate through the website, pressing the correct filter tab.

Implemented By: Euneva Development Team

5.2.3 Assignment Information Module (M4)

Secrets: ~~Collects assignment information for every course.~~Assignment information for each course as a JSON object.

Services: Uses CSS elements to navigate through the website, collecting assignment names, due dates, and completion statuses for every course.

Implemented By: Euneva Development Team

5.2.4 Quiz Information Module (M5)

Secrets: ~~Collects quiz information for every course.~~Quiz information for each course as a JSON object.

Services: Uses CSS elements to navigate through the website, collecting quiz names, due dates, and completion statuses for every course.

Implemented By: Euneva Development Team

5.3 Software Decision Module

5.3.1 Filtering Module (M6)

Secrets: ~~Filters and combines the collected information, pushing it to the DB.~~Database entries which are combined and filtered and added to the database as JSON objects.

Services: Uses the collected information and filters it for upcoming and incomplete quizzes and assignments, pushing this information to the front end.

Implemented By: Euneva Development Team

6 Traceability Matrix

This section shows two traceability matrices. Between the modules and the requirements, and between the modules and the anticipated changes.

Req.	Modules
Functional Requirements	
FR1	M1
FR2	M1
FR3	M1
FR4	M1
FR5	M1
FR6	M1
FR7	M1
FR8	M2
FR9	M3
FR10	M4
FR11	M5
FR12	M5
FR13	M6
Non-Functional Requirements	
LFA1	M1
LFS1	M1
UHE1	M1
UHP1	M1
UHU1	M1
UHA1	M1
PSL1	M3, M6
PPA1	M3, M6
PRA1	M1, M2
PRA2	M1, M2
PRF1	M2, M3, M6
PCR1	M2
PSE1	M3
EPE1	M1
RIAS1	M1
RR1	M1, M2, M3, M4, M5, M6

Table 3: Trace Between Requirements and Modules

Req.	Modules
Non-Functional Requirements Cont.	
RR2	M1
MSM1	M1, M2, M3, M4, M5, M6
MSM2	M1
MSM3	M1, M2, M3, M4, M5, M6
SR1	M1, M2, M3, M4, M5, M6
ADR1	M1
AR1	M1
AR2	M1, M2, M3, M4, M5, M6
IR1	M1, M2, M3, M4, M5, M6
PR1	M2, M3, M4, M5, M6
PR2	M2, M3, M4, M5, M6
CPC1	M1
CPC2	M1

Table 4: Trace Between Requirements and Modules Cont.

AC	Modules
AC1	M1, M2
AC2	M1, M2
AC3	M1, M2, M3, M4, M5, M6

Table 5: Trace Between Anticipated Changes and Modules

7 User Hierarchy Between Modules

As shown by 1 and 2, the graph representing the relationship between the modules is a directed acyclic graph (DAG). The modules have purposely been designed to mimic a tree due to the way how technologies like React work when rendering information. Simplicity was key when developing this product, so we followed a pipelined approach which is why the module hierarchy is more tree-like rather than graph-like.

Figure 1: Frontend Module Use Hierarchy

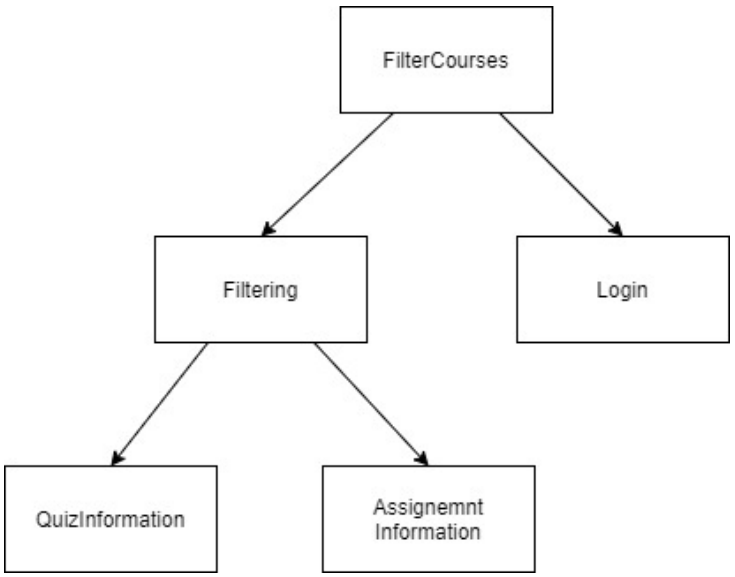

Figure 2: Backend Module Use Hierarchy