SE 3XA3: Module Guide
Lines Per Minute (lpm)

Team #16, Lines Per Minute (lpm)
Jay Mody - modyj - 400195508
Jessica Lim - limj31 - 400173669
Maanav Dalal - dalalm1 - 400178115

March 30, 2021

Table 1: Revision History

Date	**Developer(s)**	**Change**
March 13, 2021	Jay/Jessica/Maanav	Initial document, Determine Architecture, Scaffolding code structure
March 16, 2021	Jay/Maanav	Set up Sphyx, Document code
March 16, 2021	Jessica	Document architecture in MG
March 17, 2021	Jay/Jessica/Maanav	Tracability Matrix, Complete MG, Complete MIS
March 27, 2021	Jay/Jessica/Maanav	Revise for R1

Contents

List of Tables

List of Figures

1 Introduction

The goal of the lpm package is to provide a typing tool specialized for code through the command line. The lpm package is based upon the package wpm, and allows for unique simplified interfacing.

The purpose of this document is show a decomposition of the modules that will be used to implement lpm. This document will outline the overall high-level architecture of the program. The architecture is decomposed into several modules to highlight the importance of information hiding.

The document outlines how separation of changes is achieved, as well as the link back to the software requirements. The document outlines anticipated and unlikely changes to ensure information hiding (Section **??**). It also includes a module hierarchy that breaks down a modules (Section **??**), and a decomposition that specifies the secrets of hardware, behavior and software decision modules (Section **??**). All modules are linked too the software specifications via a tractability matrix (Section **??**). The Use Hierarchy diagram can be references for a visual representation (Section **??**).

2 Anticipated and Unlikely Changes

This section outlines anticipated changes in Section **??**, and unlikely changes in Section **??**.

2.1 Anticipated Changes

The following items include potential anticipated changes. These changes mostly involve implementation changes that will not impact the overall functionality and system requirements.

AC1: The logic behind how the game is run may include further abstractions and functions to ensure separation of design.

AC2: The aesthetic of the game may change slightly to ensure the game is accessible while meeting the software requirements.

AC3: The rendering logic for the input/output may change to ensure that the functions for rendering interact seamlessly with the game

AC4: The process of storage and saving code snippets may change to ensure simplicity

AC5: Additional configurations may be added to allow for a more seamless configurable experience

AC6: The logic for how statistics are stored may be adapted to increase speed and reliability.

2.2 Unlikely Changes

The following items describe features of our overall architecture model that will be unlikely to change. These mostly involve interactions between

UC1: The number of modules and how they interact, will likely be consistent

UC2: The statistics being calculated and the statistics functions will likely not change

UC3: The data structures for the different information types will likely stay consistent

UC4: The entry point for the CLI application will be consistent

3 Module Hierarchy

This section includes an overview of the hierarchy for the the design of the modules that will be implemented. The decomposition is summarized in Table **??**

M1: lpm.Commandline Module

M2: lpm.Config Module

M3: lpm.Game Module

M4: lpm.Screen Module

M5: lpm.Snippets Module

M6: lpm.Stats Module

M7: Main (__main__) Module

M8: Hardware-Hiding Module (not implemented)

The following hierarchy can be mapped 1:1 to the MIS specification for items M1-M6. M7 is simply an implementation launcher that calls the lpm.Commandline module (see the Usage flow diagram in Section **??**). M8 is not implemented as it is a hardware hiding module build into the OS. Thus it is separated from the system.

Level 1	**Level 2**
Hardware-Hiding Module	OS & Command Line Interface
Behaviour-Hiding Module	Main (__main__) Module lpm.Commandline Module lpm.Game Module lpm.Screen Module
Software Decision Module	lpm.Stats Module lpm.Config Module lpm.Snippets Module

Table 2: Module Hierarchy

4 Connection Between Requirements and Design

Refer to Section **??** to see a traceability matrix connecting the requirements outlines in the SRS, to the Modules outlined in the Module Guide Decomposition.

5 Module Decomposition

The following section will outline the different modules that were decomposed to ensure information hiding.

5.1 Hardware Hiding Modules (M??)

Secrets: The data structure and algorithm used to implement the virtual hardware. The structures and algorithms used to render and accept input and output via the terminal.

Services: Serves as a virtual hardware used by the rest of the system. This module provides the interface between the hardware and the software. These modules will be used to display outputs or to accept inputs through the CLI.

Implemented By: OS

5.2 Behaviour-Hiding Module

Secrets: The contents of the required behaviours for the lpm program.

Services: Includes programs that provide externally 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 (the user interface and inputs/outputs) and the software decision module.

Implemented By: –

5.2.1 Main (__main__) Module (M??)

Secrets: The format and structure of how the initial launch of the application occurs.

Services: Acts as the entry point of the application and triggers the launch of the lpm commmand line interface.

Implemented By: Main (__main__) Module

5.2.2 lpm.Commandline Module (M??)

Secrets: The format and structure of the command-line interface when the program is run.

Services: Reads the command line arguments and converts them appropriately to activate the module so that the program behaves as expected.

Implemented By: lpm.Commandline Module

5.2.3 lpm.Game Module (M??)

Secrets: The format and structure of how the stats module, data modules and screen modules interact with one another.

Services: Provides Game object that runs an instance of the lpm typing interface game.

Implemented By: lpm.Game Module

5.2.4 lpm.Screen Module (M??)

Secrets: The format and structure of how the input gets communicated with the game, and how output is displayed to the user

Services: Provides methods to capture user input and to display the current state of the game via the command line.

Implemented By: lpm.Screen Module

5.3 Software Decision Module

Secrets: The design decision based on mathematical theorems, physical facts, or programming considerations. The secrets of this module are *not* described in the SRS.

Services: Includes data structure and algorithms used in the system that do not provide direct interaction with the user.

Implemented By: –

5.3.1 lpm.Stats Module (M??)

Secrets: The format and structure of how the statistics are calculated

Services: Provides services to calculate LPM, CPM, WPM, and error rate for both a single snippet and over a lifetime

Implemented By: lpm.Stats Module

5.3.2 lpm.Config Module (M??)

Secrets: The format of how configurations effect the operation of the game.

Services: Provides services that will limit the code snippets the user is provided.

Implemented By: lpm.Config Module

5.3.3 lpm.Snippets Module (M??)

Secrets: The format and structure of the code snippets, and how they are organized.

Services: Provides the appropriate code snippets based upon certain filters and requirements. Checks to make sure code snippet is of appropriate language and length.

Implemented By: lpm.Snippets Module

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
FR1	M??, M??
FR2	M??
FR3	M??
FR4	M??
FR5	M??, M??
FR6	M??, M??
FR7	M??, M??, M??
FR8	~~M??~~, M??, M??
FR9	~~M??~~, M??, M??
~~FR10~~	~~M??, M??, M??~~
FR11	M??, M??
FR12	M??, M??, M??
FR13	M??, M??, M??
FR14	M??, M??, M??, ~~M??~~
FR15	M??, M??, M??
FR16	M??, M??
FR17	M??, M??
FR18	M??, M??
FR19	M??, M??
FR20	~~M??,~~ M??, M??
FR21	M??, M??
FR22	M??, M??
FR23	M??
FR24	M??
FR25	M??, M??
FR26	M??, M??
FR27	M??, M??
FR28	M??, M??

Table 3: Trace Between Requirements and Modules

AC	Modules
AC??	M??, M??, M??, M??, M??
AC??	M??, M??
AC??	M??, M??
AC??	M??, M??, M??
AC??	M??, M??, M??
AC??	M??, M??

Table 4: Trace Between Anticipated Changes and Modules

7 Use Hierarchy Between Modules

In this section, the uses hierarchy between the modules is provided. The overall architecture can be broken down into the View (Hardware hiding modules), Controller (Behavior hiding modules) and Data (Software hiding modules).

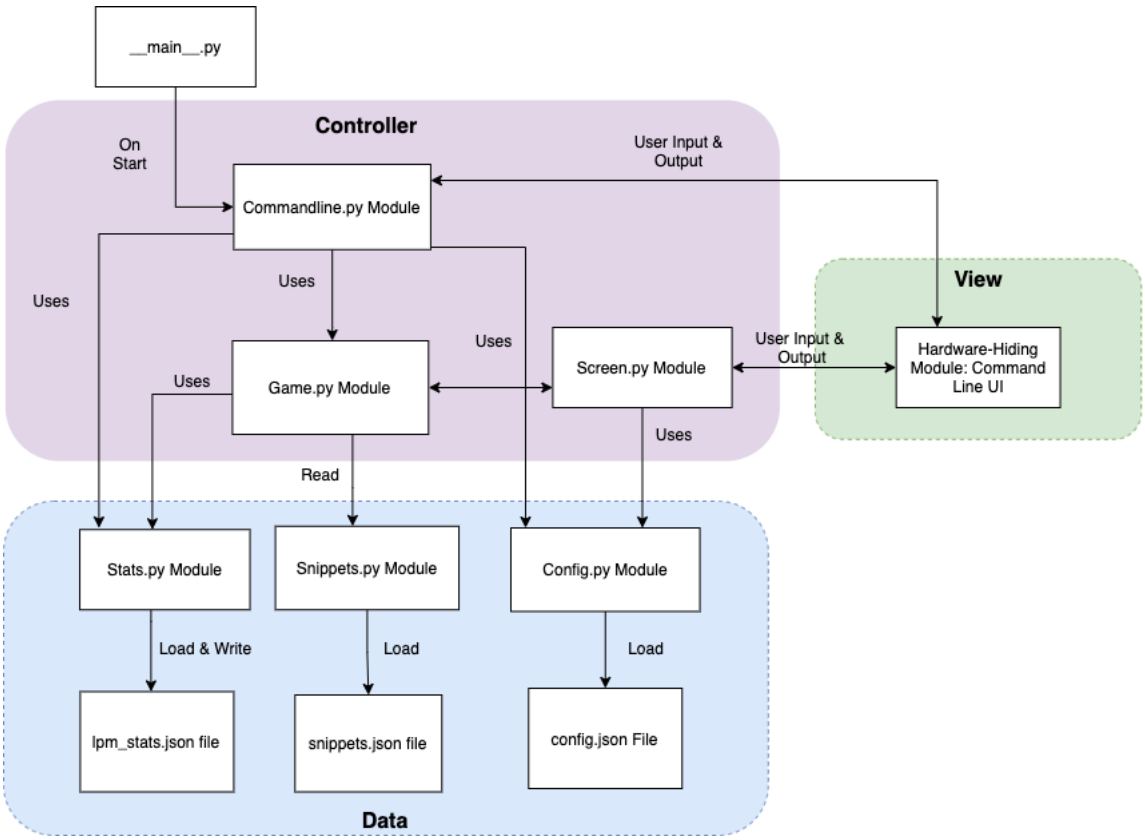

Figure 1: Use hierarchy among modules

8 Gantt Chart

See Gantt Chart with the testing plan at the following link:

A snippet of the relevant portion of the Gantt chart can be found below:

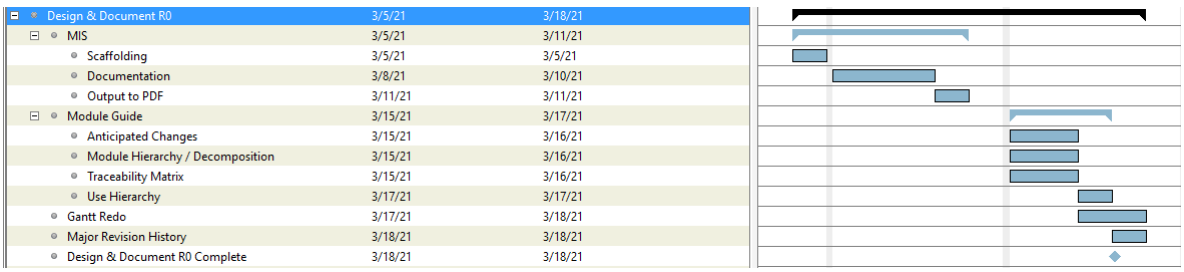

Figure 2: Gantt Chart for Design Document Checkpoint