SE 3XA3: Module Guide
SocialPy

Team 1, SocialPy
Anando Zaman, zamana11
Graeme Woods, woodsg1
Yuvraj Randhawa, randhawy
Due March 18, 2021
Tag DD-Rev.1

April 8, 2021

Contents

List of Tables

List of Figures

Table 1: Revision History

Date	**Developer(s)**	**Change**
March 3, 2021	Anando Zaman	Copy template & completed introductory section
March 5, 2021	Anando Zaman	Completed section 2, 3, 4, & 6.
March 6, 2021	Anando Zaman	Some progress on section 5.
March 13, 2021	Yuvraj Randhawa	5.2.11, 5.2.12, 5.2.13, 5.2.14.
March 13, 2021	Graeme Woods	Completed Section 7 - Uses Hierarchy

1 Introduction

1.1 Introduction

SocialPy is a re-development of an open-source social media application using the Python programming language. The goal of the application is to further enhance the user experience by adding additional features(ex; delete posts, view followers, etc) while providing user security through authentication which was not provided in the original implementation.

1.2 Context

Throughout the re-development of the application, different design techniques have been considered such as modularity and information-hiding in order to develop efficient software. The purpose of this document is to describe the structure of the system and its components as modules. The focus is on the hierarchy of each of the modules along with descriptions of the system architecture via unified modelling language(UML) approaches. This should allow both future designers and maintainers to easily identify different aspects of the software and the inner-workings of how different modules communicate to solve specific tasks/problems. This document is organized as follows:

- Section 2 contains a list of unanticipated and unlikely change
- Section 3 contains the overview of the module hierarchy
- Section 4 contains the Connections Between Requirements and Design.
- Section 5 contains Description of each module
- Section 6 contains two traceability matrices, one for connections between SRS and modules and the another for connections between anticipated changes and modules.
- Section 7 contains information to visualize the relation among all modules.

1.3 Motivation and Design Choices

For large software system designs, it is a good idea to decompose larger modules into smaller submodules in order to enhance code reusability, readability, and structure. It prevents a single-source of failure as the code is partitioned into several different modules conducting a different task rather than being placed into one massive file which would have otherwise been difficult to maintain and follow. Moreover, modular decomposition enables design for future changes/modifications such as adding new modules for additional functionality without disrupting majority of the system. This is important since the application will likely change and have additional features implemented based on the users' feedback. Thus, the team followed the 'design of change' pattern for the SocialPy application.

Our design follows the rules layed out by (Parnases, 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 section lists possible changes for the SocialPy application. There are two categories for changes based on likeliness of the change which are Anticipated changes and unlikely changes. Anticipated changes are planned or probable in foreseeable future of SocialPy while unlikely changes are those that are not planned for the entire life of the application. Below describes these two categories for the application.

2.1 Anticipated Changes

Below is a list of anticipated changes:

AC1: ~~SocialPy must be kept up-to-date with any new Python versions.~~ The format of the display format of the view module as new UI functionality is added, thus the structure in how information is displayed can change.

AC2: SocialPy must be kept up-to-date with any new Firebase API versions.

AC3: ~~The hardware on which the software is running.~~

AC4: The format of the "help" commands as more features are added.

AC5: The CommandParser module as more functionality/commands are added.

AC6: ~~The format of the Firebase(NoSQL) DB branches as more features are added.~~

2.2 Unlikely Changes

Below is a list of unanticipated changes:

UC1: Login/Registeration Authentication system.

UC2: Input/Output devices (Input: keyboard, Output: Screen).

UC3: The *main.py* controller.

UC4: The data query & data display/View modules.

UC5: The account userID information.

UC6: The data structure of followers/following lists.

3 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 2. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

3.1 Posts subsystem modules

M1 AddPost module

M2 DeletePost module

M3 EditPost module

M4 QueryPosts module

M5 ViewPost module

3.1.1 Profile subsystem modules

M6 AddFollowers module

M7 DeleteAccount module

M8 DeleteFollowers module

M9 EditLocation module

M10 EditName module

M11 QueryFollowers module

M12 ViewFollowers module

M13 QueryProfile module

M14 ViewProfile module

3.1.2 Firebase subsystem modules

M15 Authentication module

M16 Firebase_creds module

3.1.3 CommandParser modules

M17 CommandParser module

M18 Main module

Level 1	Level 2
Hardware-Hiding Module	
Behaviour-Hiding Module	M1 M2 M3 M4 M5 M6 M7 M8 M9 M10 M11 M12 M13 M14 M15 M16
Software Decision Module	M17 M18

Table 2: Module Hierarchy

4 Connection Between Requirements and Design

The SRS defines the requirements that the system design must fulfill. In Table 3, the connections between the requirements and modules are listed.

5 Module Decomposition

The modules of SocialPy are decomposed according to the principle of "information hiding" proposed by David Parnas. The goal of the decomposition is to provide a detailed description of how the system operates. 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 AddPost (M1)

Secrets: Format & structure of the Post data.

Services: Converts input data into a post and updates the database..

Implemented By: AddPost.py

5.2.2 DeletePost (M2)

Secrets: Format & The format/structure of the post data and algorithm/method to delete it from the database.

Services: Obtains the input data specifying the post to remove via PostID to QueryPost module, and removes it if exists.

Implemented By: DeletePost.py

5.2.3 EditPost (M3)

Secrets: Format & Data structure of post data and method of editing it.

Services: Queries the data for a given PostID and edits the contents.

Implemented By: EditPost.py

5.2.4 QueryPosts (M4)

Secrets: Format & Data structure of post data and method to query it.

Services: Queries the data for a given PostID.

Implemented By: QueryPosts.py

5.2.5 ViewPost (M5)

Secrets: Format & The format and structure of data being displayed to the screen.

Services: Prints the data queried from a post.

Implemented By: ViewPost.py

5.2.6 AddFollowers (M6)

Secrets: Format & The format and structure of data used for containing the the follower/following names.

Services: Updates the followers/following lists by adding the user_to_follow in the following list, if user is valid and exists.

Implemented By: AddFollowers.py

5.2.7 DeleteFollowers (M7)

Secrets: Format & The format and structure of data used for containing the the follower/following names.

Services: Update the following/followers list by removing the users' name.

Implemented By: DeleteFollowers.py

5.2.8 QueryFollowers (M11)

Secrets: Format & The format and structure of data used for containing the the follower/following names.

Services: Query the followers or following list of a given user if the username is valid.

Implemented By: QueryFollowers.py

5.2.9 ViewFollowers (M12)

Secrets: Format & The format and structure of the followers/following list data displayed to the screen.

Services: Print the follower/following information to the screen.

Implemented By: ViewFollowers.py

5.2.10 DeleteAccount (M8)

Secrets: Format & The format and data structure used to hold the account information.

Services: Permanently removes the account of the user off of the database system.

Implemented By: DeleteAccount.py

5.2.11 EditLocation (M9)

Secrets: Format & The format and data structure used to store the location

Services: Updates the users location.

Implemented By: EditLocation.py

5.2.12 EditName (M10)

Secrets: Format & The format and data structure used to store the name

Services: Update the users name.

Implemented By: EditName.py

5.2.13 QueryProfile (M13)

Secrets: Format & The format and structure of data used for containing the profile information.

Services: Extracts profile related information from the database.

Implemented By: QueryProfile.py

5.2.14 ViewProfile (M14)

Secrets: Format & The format and structure of the profile data displayed to the screen.

Services: Print the profile information to the screen.

Implemented By: ViewProfile.py

5.2.15 Authentication (M15)

Secrets: The format and data structure of the Authentication credentials

Services: Authenticate the user given their account credentials.

Implemented By: Authentication.py

5.2.16 FirebaseCreds (M16)

Secrets: The format and data structure of the Firebase variables.

Services: Establishes a connection to the database and executes commands for communication.

Implemented By: Firebasecreds.py

5.3 Software Decision Module

5.3.1 CommandParser (M17)

Secrets: The format of the command input and method to parse into specific commands.

Services: Provides a means of conducting the different actions of the app(ie; addPost, ViewPost, etc) by parsing the commands.

Implemented By: CommandParser.py

5.3.2 Main (M18)

Secrets: Method to retrieve input data and run as an executable application.

Services: Passes data amongst all the different modules to keep them interconnected in order to facilitate communication.

Implemented By: Main.py

6 Traceability Matrix

This section shows two traceability matrices: between the modules and the requirements and between the modules and the anticipated changes. These traceability matrices are shown on table 3 & 4 respectively.

Req.	Modules
FR1	M16
FR2	M15, M16
FR3	M15, M16
FR4	M4, M5
FR5	M4, M2
FR6	M15, M16
FR7	M11, M12
FR8	M2
FR9	M3, M4
FR10	M17
FR11	M13, M14
FR12	M9
FR13	M10
FR14	M7
FR15	M8
FR16	M11, M12
NFR1	M18, M5, M12, M14
NFR2	M17
NFR3	M18
NFR5	M17, M18
NFR6	M18, M5, M12, M14
NFR7	M4
NFR8	M15
NFR9	M4
NFR10	M1
NFR12	M1, M2, M3, M4, M5
NFR16	M16
NFR19	M18
NFR24	M17
NFR25	M17
NFR28	M15
NFR29	M15, M4, M5
NFR30	M15, M1
NFR31	M15
NFR32	M18, M5, M12, M14

Table 3: Trace Between Requirements and Modules

AC	Modules
AC1	None
AC2	None
AC3	None
AC4	M17
AC5	M17
AC6	None

Table 4: Trace Between Anticipated Changes and Modules

7 Use Hierarchy Between Modules

In this section, the uses hierarchy between modules is provided. Figure 1 illustrates 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 subset of the system, and modules in the higher level of the hierarchy are essentially simpler because they use modules from the lower levels.

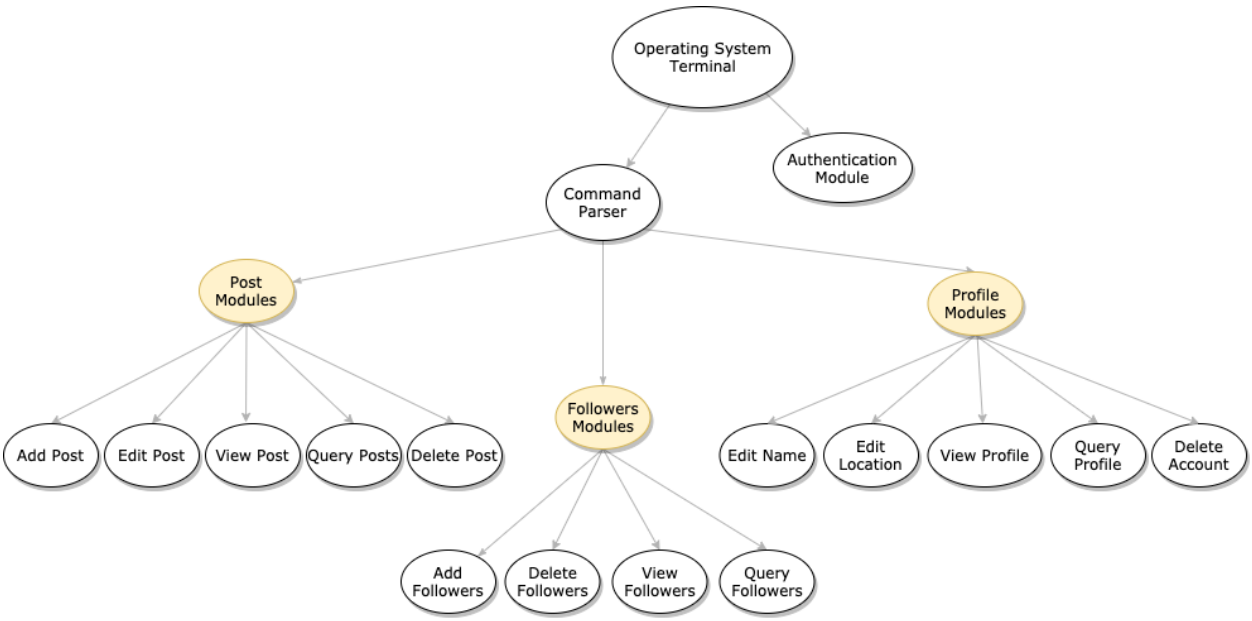

Figure 1: Use hierarchy among modules

8 Schedule

Implementation dates of the modules along with team members responsible are available in the Gantt chart. Specific testing activities and development are listed in detail there.

Gantt chart provided here: https://gitlab.cas.mcmaster.ca/zamana11/se3xa3-project/-/tree/master/ProjectSchedule.

References

David L. Panas. On the crieria 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.