Module Guide for Sayyara

Team 31
SFWRENG 4G06
Christopher Andrade
Alyssa Tunney
Kai Zhu
Ethan Vince-Budan
Collin Kan
Harsh Gupta

April 5, 2023

Revision History

Date	Version	Notes
Jan. 18, 2023	Rev 0	Revision 0
April 5, 2023	Rev 1	Revision 1

Table 1: Revision History

Reference Material

This section records information for easy reference.

Abbreviations and Acronyms

symbol	description
AC	Anticipated Change
GUI	Graphical User Interface
M	Module
MG	Module Guide
OS	Operating System
R	Requirement
SC	Scientific Computing
SRS	Software Requirements Specification
Sayyara	Explanation of program name
UC	Unlikely Change

Table 2: Abbreviations and Acronyms

Contents

List of Tables

List of Figures

1 Introduction

Decomposing a system into modules is a commonly accepted approach to developing software. A module is a work assignment for a programmer or programming team. We advocate a decomposition based on the principle of information hiding. This principle supports design for change, because the "secrets" that each module hides represent likely future changes. Design for change is valuable in SC, where modifications are frequent, especially during initial development as the solution space is explored.

Our design follows the rules laid out by Parnes et al., as follows:

- System details that are likely to change independently should be the secrets of separate modules.
- Each data structure is implemented 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.

After completing the first stage of the design, the Software Requirements Specification (SRS), the Module Guide (MG) is developed. The MG specifies the modular structure of the system and is intended to allow both designers and maintainers to easily identify the parts of the software. The potential readers of this document are as follows:

- New project members: This document can be a guide for a new project member to easily understand the overall structure and quickly find the relevant modules they are searching for.
- Maintainers: The hierarchical structure of the module guide improves the maintainers' understanding when they need to make changes to the system. It is important for a maintainer to update the relevant sections of the document after changes have been made.
- Designers: Once the module guide has been written, it can be used to check for consistency, feasibility, and flexibility. Designers can verify the system in various ways, such as consistency among modules, feasibility of the decomposition, and flexibility of the design.

The rest of the document is organized as follows. Section 2 lists the anticipated and unlikely changes of the software requirements. Section 3 summarizes the module decomposition that was constructed according to the likely changes. Section 4 specifies the connections between the software requirements and the modules. Section 5 gives a detailed description of the modules. Section 6 includes two traceability matrices. One checks the completeness of the design against the requirements provided in the SRS. The other shows the relation between anticipated changes and the modules. Section 7 describes the use relation between modules.

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 that hides the associated decision. The approach adapted here is called design for change.

AC1: The specific devices that users access the frontend from.

AC2: The number of total and concurrent users

AC3: The database storage required to accommodate total and concurrent users.

AC4: The network bandwidth required to accommodate total and concurrent users.

AC5: The GUI display resolution and styling.

AC6: Access control methods such as openID authentication support.

AC7: The specific level of time granularity for appointment scheduling.

AC8: The method notification delivery (e.g. email, real-time messaging, etc)

AC9: The hardware that the backend is hosted on.

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 decision 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: Software compatibility for backend (Python, Django) and frontend (Typescript, NextJS)

UC2: Input methods (cursor, keyboard, touch).

UC3: The system will always be able to provide a list of all registered repair shops.

UC4: Users will always have an authentication methods.

UC5: Customers will be able to communicate with shop owners.

3 Module Hierarchy

For a higher resolution diagram check appendix.

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 3. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

Backend Modules	**Frontend Modules**
BE1: Profile	FE1: Auth
BE2: Shop	FE2: Vehicles
BE3: ShopService	FE3: Homepages
BE4: Service	FE4: ShopLookup
BE5: EmployeeAvailability	FE5: CustomerProfile
BE6: VehiclesInfo	FE6: ShopOwnerProfile
BE7: Quote	FE7: EmployeeProfile
BE8: QuoteRequest	FE8: Quote
BE9: Appointment	FE9: QuoteRequest
	FE10: ShopCreate
	FE11: ShopDetails
	FE12: AppContext
	FE13: Appointment

Table 3: List of backend and frontend leaf modules

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 4.

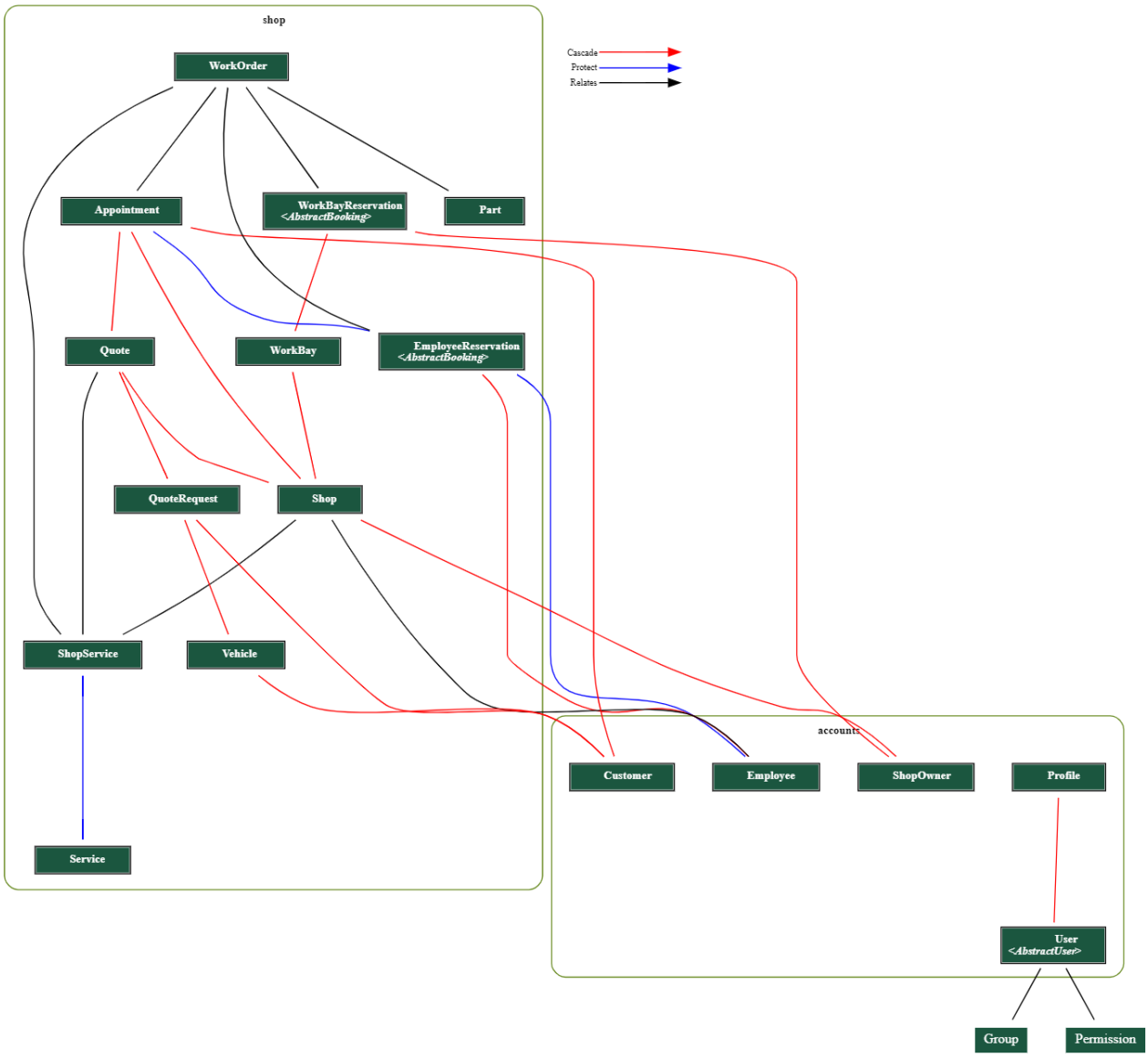

Figure 1: Backend Module Hierarchy Diagram

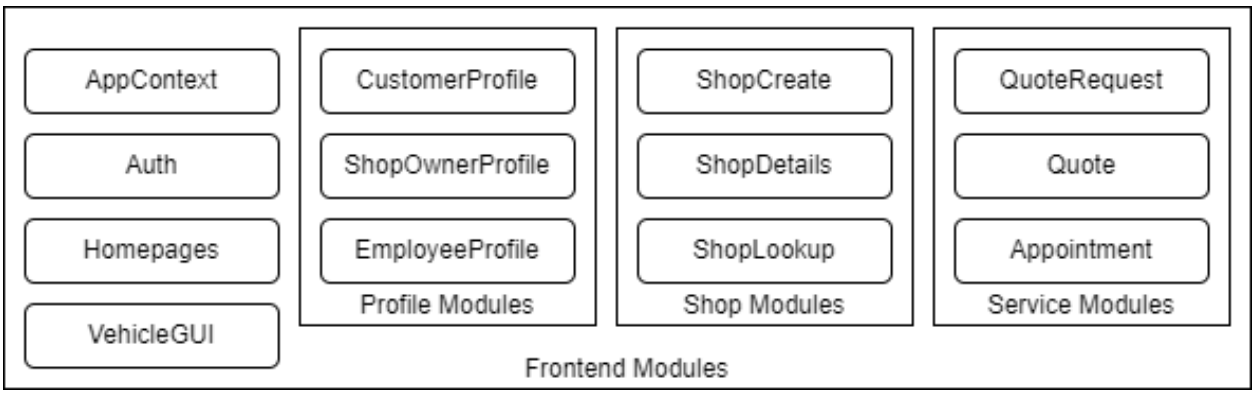

Figure 2: Frontend Module Hierarchy Diagram

- Several requirements in the SRS (i.e. SR1, PAR11, ApR3, etc.) mention different behaviours of the system based on the class of user (employee, customer, shop owner). For this reason the functionality of profile-related actions such as updating account details and changing shop/personal availability have been split on the frontend into multiple class specific modules. This allows for logic across the same user class to be fully encapsulated and easily modified in the future, while maintaining a simple backend architecture.

- A large amount of non-functional requirements for Sayyara specify visual design preferences and usability goals intended to be enforced across the entire system. To achieve this level of consistency and simplify the frontend design process, a framework like React Bootstrap will be used as these components already meet the required design characteristics of the system.

5 Module Decomposition

Modules are decomposed according to the principle of "information hiding" proposed by Parnes et al.. 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. *Sayyara* means the module will be implemented by the Sayyara software.

Only the leaf modules in the hierarchy have to be implemented. If a dash (–) is shown, this means that the module is not a leaf and will not have to be implemented.

5.1 Backend Modules

5.1.1 Profile module (BE1)

Secrets: Holds account profile related information such as name, contact, user type

Services: Methods to create new accounts, update existing information, and reset passwords

Implemented By: Sayyara

Type of Module: Abstract Data Type

5.1.2 Shop Module (BE2)

Secrets: Shop model containing information including name, address, description, email, phone, address, services, employees, and owner account.

Services: Methods to create, update, read, and delete a Shop instance.

Implemented By: Sayyara

Type of Module: Abstract Data Type

5.1.3 ShopService Module (BE3)

Secrets: ShopService model containing information including price, estimated time, and service.

Services: Methods to create, update, read, and delete a ShopService instance.

Implemented By: Sayyara

Type of Module: Abstract Data Type

5.1.4 Service Module (BE4)

Secrets: Service model containing information including name and description.

Services: Methods to create, update, read, and delete a Service instance.

Implemented By: Sayyara

Type of Module: Abstract Data Type

5.1.5 EmployeeAvailability Module (BE5)

Secrets: EmployeeAvailability, EmployeeAvailabilityOccurrence, EmployeeTimeSlot and EmployeeReservation models.

Services: Methods to access the available availability objects, set up time slots and reservations, and approve or reject a reservation.

Implemented By: Sayyara

Type of Module: Abstract Data Type

5.1.6 VehiclesInfo Module (BE6)

Secret: Characteristic details of vehicle types (make, model, year, etc.)

Services: Storage, retrieval and modification of vehicle information

Implemented By: Sayyara

Type of Module: Record

5.1.7 Quote Module (BE7)

Secrets: The information contained in a quote that is sent to an automotive shop, filled in and sent on the frontend by a customer.

Services: Defines the information contained in a quote that is displayed to an automotive shop.

Implemented By: Sayyara

Type of Module: Abstract Data Type

5.1.8 QuoteRequest Module (BE8)

Secrets: The information contained in a quote request that is filled in and sent to automotive shops by customers. This differs from the Quote module as the QuoteRequest module contains additional information relevant to the customer, and contains all shops sent to rather than just one quote per shop.

Services: Defines the information contained in a quote request, modifiable by the customer.

Implemented By: Sayyara

Type of Module: Abstract Data Type

5.1.9 Appointment Module (BE9)

Secrets: Appointment model.

Services: Methods to create an appointment, set the estimated time and price, associate the appointment with a customer and shop, and send a confirmation email.

Implemented By: Sayyara

Type of Module: Record

5.2 Frontend Modules

Secrets: The contents of the required behaviours.

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 and the software decision module. The programs in this module will need to change if there are changes in the SRS.

Implemented By: –

5.2.1 Auth Module (FE1)

Secrets: Contains authentication status and tokens

Services: Methods for connecting to backend APIs related to account authentication and registration

Implemented By: Sayyara

Type of Module: Library

Secrets: Customer, Shop, Quote, and EmployeeReservation models.

Services: Methods to create an appointment, set the estimated time and price, and associate the appointment with a customer and shop.

Implemented By: Sayyara

Type of Module: Record

5.2.2 Vehicles Module (FE2)

Secrets: Information presented to the user regarding vehicles, and event handling of the vehicles page

Services: GUI output containing appropriate information/elements for vehicle management

Implemented By: Sayyara

Type of Module: Abstract Object

5.2.3 Homepages Module (FE3)

Secrets: Event handling and GUI output formatting of the website landing page, for all classes of user

Services: GUI output containing homepage-related information appropriate for the user type

Implemented By: Sayyara

Type of Module: Abstract Object

5.2.4 ShopLookup Module (FE4)

Secrets: Shop model

Services: Provides users with a way to search for a shop by name or postal code. Users can also filter by distance.

Implemented By: Sayyara

Type of Module: Abstract Object

5.2.5 CustomerProfile Module (FE5)

Secrets: Event handling and GUI output formatting of the customer profile page

Services: GUI output containing customer profile related information

Implemented By: Sayyara

Type of Module: -

5.2.6 ShopOwnerProfile Module (FE6)

Secrets: Event handling and GUI output formatting of the shop owner profile page

Services: GUI output containing shop owner profile related information

Implemented By: Sayyara

Type of Module: Abstract Object

5.2.7 EmployeeProfile Module (FE7)

Secrets: Event handling and GUI output formatting of the employee profile page

Services: GUI output containing employee profile related information

Implemented By: Sayyara

Type of Module: Abstract Object

5.2.8 Quote Module (FE8)

Secrets: Event handling and GUI output formatting of the shop quote page

Services: GUI output containing shop quote related information. The shop employee is able to view and respond to new quote requests submitted by customers and can view the history of all quotes for said shop.

Implemented By: Sayyara

Type of Module: Abstract Object

5.2.9 QuoteRequest Module (FE9)

Secrets: Event handling and GUI output formatting of the customer quote request page

Services: GUI output containing customer quote request related information. The customer is also able to fill out new quote requests, and edit or delete existing quote requests.

Implemented By: Sayyara

Type of Module: Abstract Object

5.2.10 ShopCreate Module (FE10)

Secrets: Event handling and GUI output formatting of the ShopCreate page

Services: GUI output containing fields for the shop owner to fill in and create their Shop instance in the backend

Implemented By: Sayyara

Type of Module: Abstract Object

5.2.11 ShopDetails Module (FE11)

Secrets: Shop Service, and Service models.

Services: Provides customer with the types of services a certain shop may offer.

Implemented By: Sayyara

Type of Module: Abstract Object

5.2.12 AppContext Module (FE12)

Secrets: Holds application-level states including authentication

Services: Methods for setting application-level variables

Implemented By: Sayyara

Type of Module: Abstract Object

5.2.13 Appointment Module (FE13)

Secrets: Appointment model.

Services: Methods to create an appointment, set the estimated time and price, and associate the appointment with a customer and shop.

Implemented By: Sayyara

Type of Module: Abstract Object

6 Traceability Matrix

This section shows two traceability matrices: between the modules and the functional requirements and the modules and the anticipated changes.

Req.	Modules
AuR1	BE1, FE1, FE12
AuR2	BE1, FE1, FE12
AuR3	BE1, FE1
SR1	BE2, BE3, FE6, FE10
SR2	BE2, BE3, FE4, FE11, FE6
PAR1	BE2, FE6
PAR2	BE2, FE6
PAR3	BE1, FE5, FE6, FE7
PAR4	BE2, FE4, FE11
PAR5	BE2, FE6
PAR6	BE5, FE11, FE6
PAR7	BE1, FE6
PAR8	BE1, FE6
PAR9	BE1, FE1, FE7
PAR10	FE7
ApR1	BE2, BE5, FE6
ApR2	BE7, BE8, BE9, FE8, FE9
ApR3	BE7, BE8, BE9, FE8, FE9
ApR4	BE8, FE9
QR1	BE7, BE8, FE8, FE9
QR2	FE9
QR3	BE7, BE8, FE8, FE9

Table 4: Trace Between Functional Requirements and Modules

AC	Modules
AC1	FE1, FE2, FE3, FE4, FE11, FE5, FE6, FE7, FE8, FE9, FE10 FE12, FE13, FE??
AC2	BE1
AC3	BE1, BE2, BE3, BE4, BE5, BE6, BE7, BE8, BE9, BE??
AC4	BE1, BE2, BE3, BE4, BE5, BE6, BE7, BE8, BE9, BE??
AC5	FE1, FE2, FE3, FE4, FE11, FE5, FE6, FE7, FE8, FE9, FE10 FE12, FE13, FE??
AC6	FE1, FE12
AC7	BE9, FE13
AC8	BE5, BE7, BE8, BE9, BE??
AC9	BE1, BE2, BE3, BE4, BE5, BE6, BE7, BE8, BE9, BE??

Table 5: Trace Between Anticipated Changes and Modules

7 Use Hierarchy Between Modules

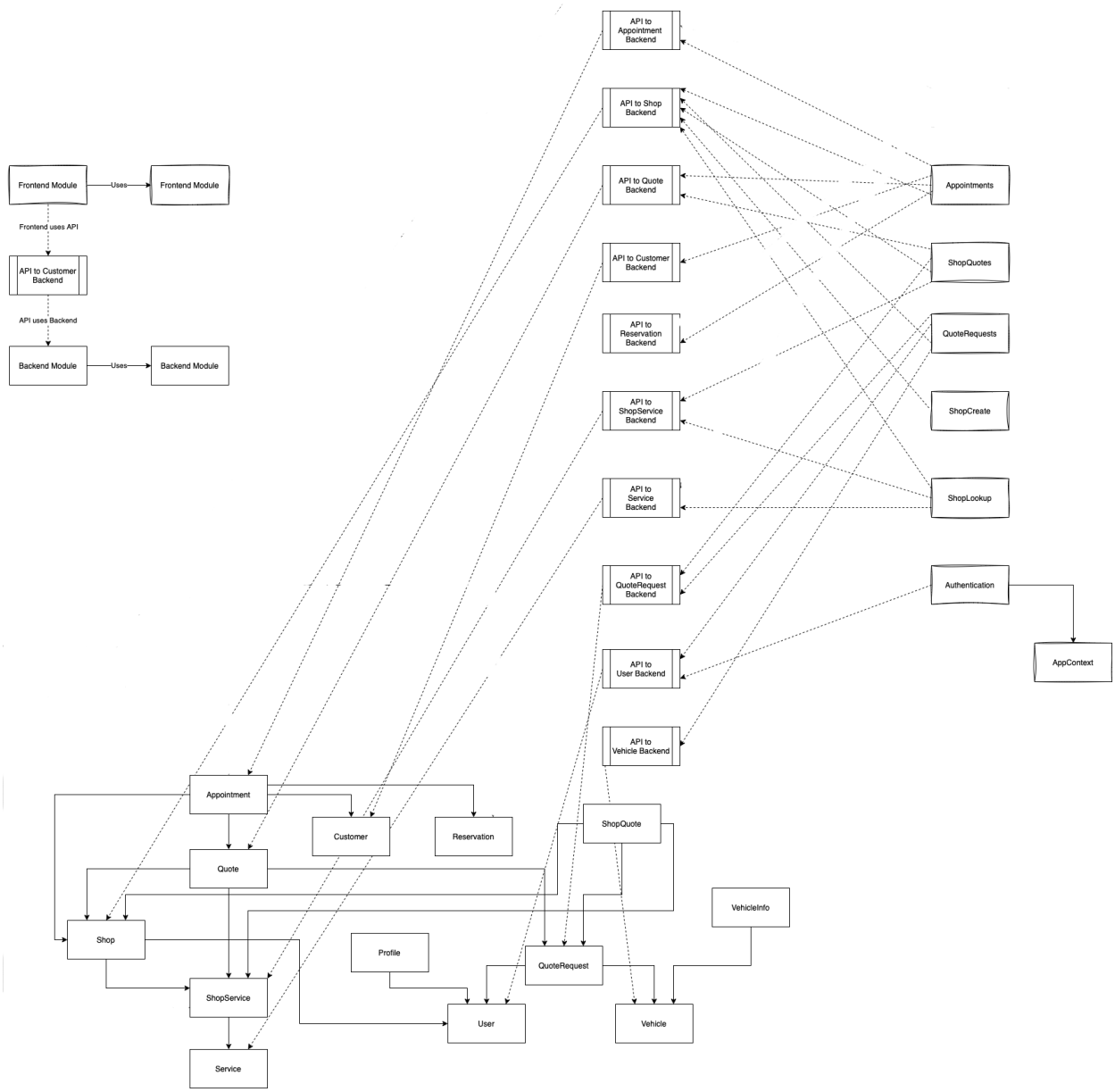

Figure 3: Use hierarchy among modules