Iteration 1:

Table of Contents:

ADD Step 1: Review Inputs

In this step of the ADD method, we will review the inputs and show which requirements will be the drivers, as shown below.

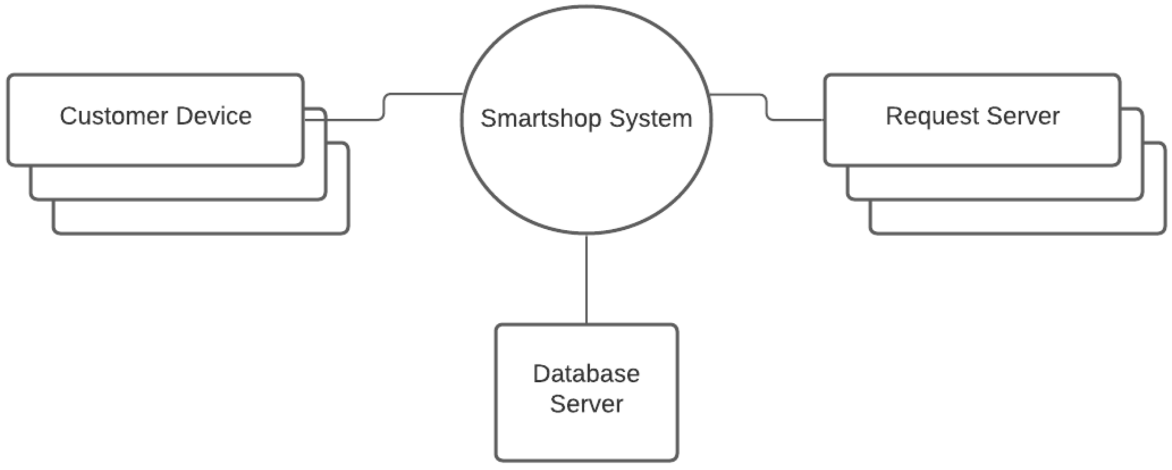
Category	**Details**
Design Purpose	The purpose is to produce a detailed design to allow the construction of the application in the most efficient manner
Primary Functional requirements	The primary use cases we will be using will be the following: UC-1: Monitoring the Network Status UC-2: Detecting Faults in the Network UC-8: Displaying information UC-10: User Authentication
Quality Attribute Scenarios	

ID	**Importance**	**Implementation Difficulty**
QA-1	High	High
QA-3	High	Medium
QA-4	High	Low
QA-5	Medium	High
QA-6	Medium	High

STEP 2: Establish iteration goal by selecting drivers

QA-1: Performance
QA-3: Modifiability
QA-4: Security
QA-5: Interoperability
QA-6: Availability
CON-2: Application must be accessible to the public. This cannot interfere with existing users' security information.
CON-3: Decreased bandwidth for user convenience without affecting overall performance
CRN-2: App suitable for customers across all devices

STEP 3: Choose one or more elements of the system to refine



STEP 4: Choose one or more design concepts that satisfy the selected drivers

Design Decisions and Locations	**Rationale**	**Discarded alternatives:**
Logically using the SmartShop Request using the Mobile Application reference architecture	The reference architecture (see Section A.1.4) supports the development of the SmartShop application being installed on customer and store devices. This allows the user to have an enjoyable UI experience and have adequate performance (UC-1) . These capabilities are also helpful in achieving QA-2. This application is not suited for a web browser, as explained in CON-1. They would need to be installed via the app store.	Web Applications: The reference architecture is oriented towards the development of applications that are accessed from a web browser.
Physically structure the application using the three-tier deployment program	Because the application must be accessed from a mobile application (CON-1), and the existing database (CON-4) will be here to support the SmartShop system, so a three tier	

	system can be used to be deployed.	
Logically structure the Request Server using the Service Application reference architecture	Service applications won't be providing the user interface any benefits, but will allow the app to function whilst being closed so orders won't get cancelled even if you close the app.	
Build the user interface of the client using Swift	The standard framework for building applications using Swift ensures portability (CON-3) and the developers are most familiar with. (CRN-2)	

STEP 5: Instantiate architectural elements, allocate responsibilities, and define interfaces

Design Decisions and Locations	**Rationale**
Remove local data sources in the Mobile Application reference architecture	With the network connection being reliable, SmartShop no longer needs to store data locally. Main communication with the server is controlled in the data layer and any other communication will be managed through local method calls
Deploy a data, application and user tier for the mobile application system	In the SmartShop system, the three-tier architecture used are data, application and user. The application and user tier is where the customer's data is processed, and the data tier is where the data is stored and managed.
Implement a module dedicated to deploying data in the Service application reference architecture	Ensures user's will still be receiving information/notifications regarding their orders even if they closed the app. This will further facilitate the achievement of QA-3.

STEP 6: Sketch views and record design decisions

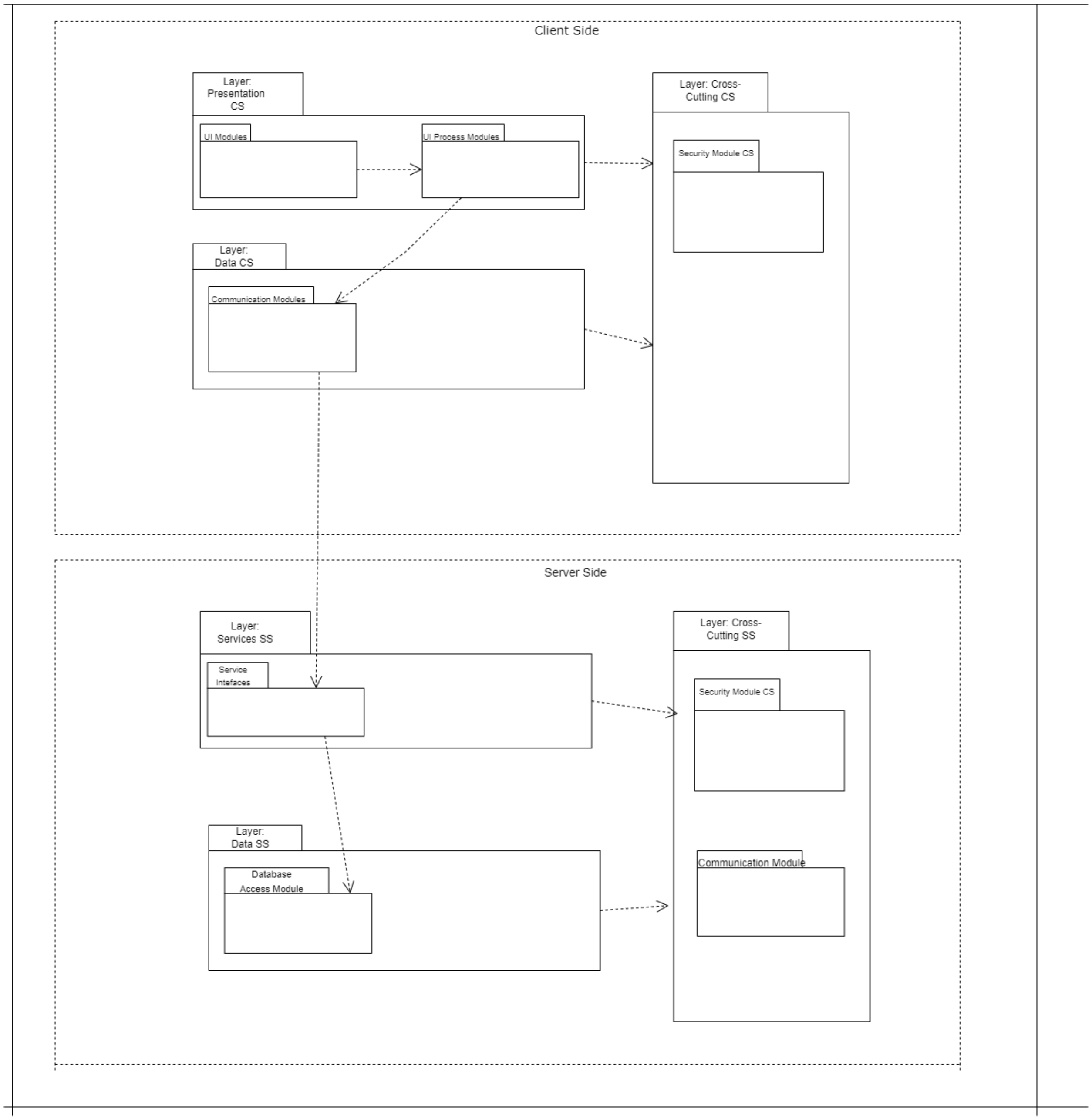


Figure 1: Module View

Element	**Responsibility**
Presentation CS	This layer contains modules that control user interaction and use case control flow
Data CS	This layer contains modules that are responsible for communication with the

	server
Cross-cutting CS	This “layer” includes modules with functionality that goes across different layers, such as security
UI modules	These modules render the user interface and receive user inputs
UI process modules	These modules are responsible for control flow of all the system use cases
Communication modules CS	These modules consume the services provided by the application running on the server side
Services SS	This layer contains modules that expose services that are consumed by the clients
Data SS	This layer contains modules that are responsible for data persistence
Cross-cutting SS	These modules expose services that are consumed by the clients
Services interfaces SS	These modules expose services that are consumed by the clients
DB access module	This module is responsible for persistence of business entities (objects) into the relational database.

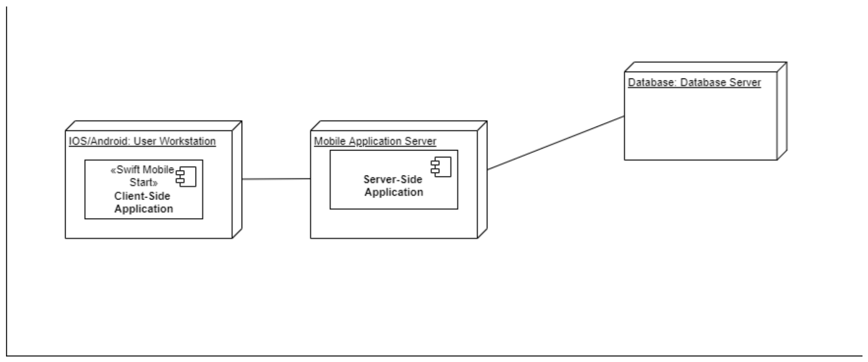

IOS/Android: User Workstation
«Swift Mobile Start»
Client-Side Application
Mobile Application Server
Server-Side Application
Database: Database Server

Figure 2: Initial deployment diagram

Element	**Responsibility**
User workstation	The user's PC, which hosts the client side logic of the application
Application server	The server that hosts server side logic of the application
Database server	The server that hosts the relational database

Relationship	**Description**
Between application server and database server	Communication with the database will be done using SQLite

STEP 7: Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

Not Addressed	**Partially Addressed**	**Completely Addressed**	**Design Decisions Made During the Iteration**
		UC-1	Diagrams have been created to support this use case in this iteration.
QA-2			Not addressed in this iteration

		CON-1	Identified modules relating to the issue being investigated
	CON-3		The standard framework for building applications using Swift ensures portability
CON-4			No relevant design decisions have been made so far in this iteration.
	CRN-2		The standard framework for building applications using Swift ensures portability
	QA-3		Introduction of a module dedicated to deploying data on the client application that ensures users will be able to close the app without any disruptions to their order status. The details of this component and its interfaces have not yet been defined