Software Design Document
Team: Learning Center 152
Version: 2.0

Product Owner	Romeo Vanegas
Frontend	Joseph Buskirk
QA Tester	Jonathan Wong
Scrum Master	Aakash Sharma
Backend Developer	Eric Yang
Backend Developer	Gurdev Singh

Prepared by: Romeo Vanegas

Table of Contents

1. Introduction

1.1 Purpose

This document will detail the design of The Learning Center mobile application as specified in the Software Requirements Specification document. This document will include each of the applications interfaces, architectures, database schemas and motivations behind those choices. High level designs are included in this document.

1.2 Scope

This Software Design Document will provide a sufficient high-level description for the underlying software and its systems to allow for the creation of a base level of functionality. The focus of this document will use this base of functionality to describe the system and its main features.

1.3 Overview

This Software Design Document will comprise of 5 sections:

1. Introduction
2. Use Cases
3. Design Overview
4. Dynamic Model
5. Non-Functional requirements

1.4 References Definitions, Acronyms and Abbreviations

Learning Center Application – the subject of this document but will also be referred to as

- "The LC App, Application, Mobile Application or App"

Definitions and Acronyms:

API: Application Programming Interface.

SI: Supplemental Instruction

ASC: Academic Success Coaching

LC: Learning Center (Team)

The following references are cited in this document:

- **Learning Center SRS (Click to view):**
  - Final_srs_LC152.docx
- **Learning Center Application Business Proposal (Click to view):**
  - Learning Center Application Proposal.pdf

2. Use Cases

2.1 Actors

2.1.1 **Current Students:** Any registered and currently enrolled student will have complete access to all Application functionalities.

2.1.2 **Administrators:** These actors will consist of Learning Center Tutors/Staff/Faculty

2.2 List of Use cases

2.2.1 Registered Users:

2.2.1.1 *Academic Success Coaching:* Accessible by button press.

2.2.1.2 *Supplemental Instruction:* Accessible by button press.

2.2.1.3 *Tutoring:* Accessible by button press.

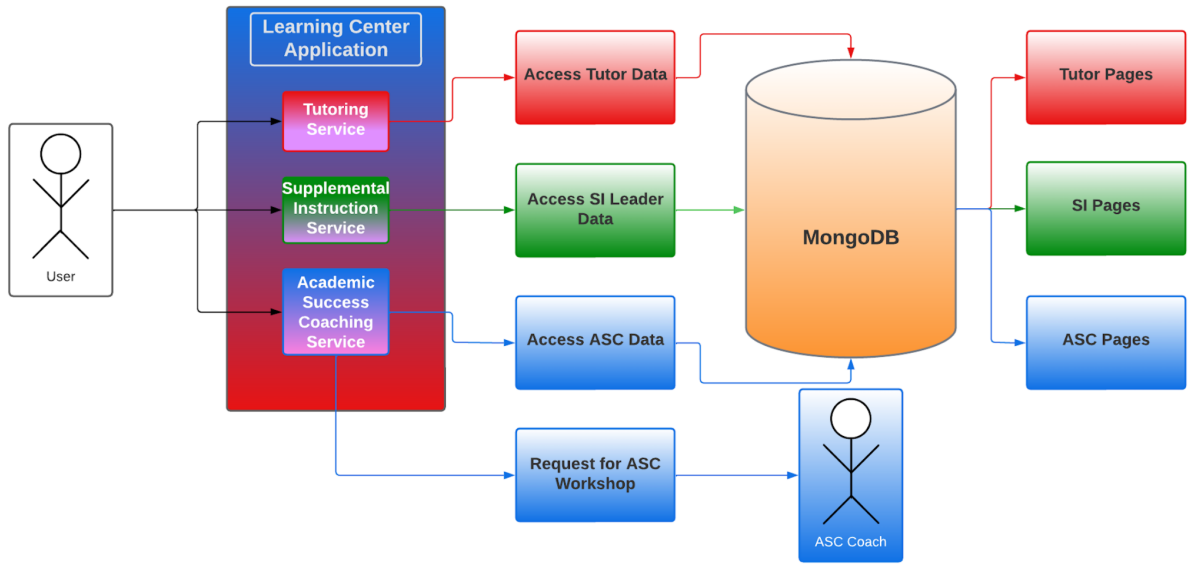
2.2.2 App maintainers:

2.2.2.1 *Tutors:* Will have access to their availability, subjects, and profile information.

2.2.2.2 *Staff:* Will have the ability to add new subjects, tutors or other to be determined items to the learning center application.

2.2.2.3 *Faculty:* Same description/duties as Staff.

2.3 Use Case Diagram



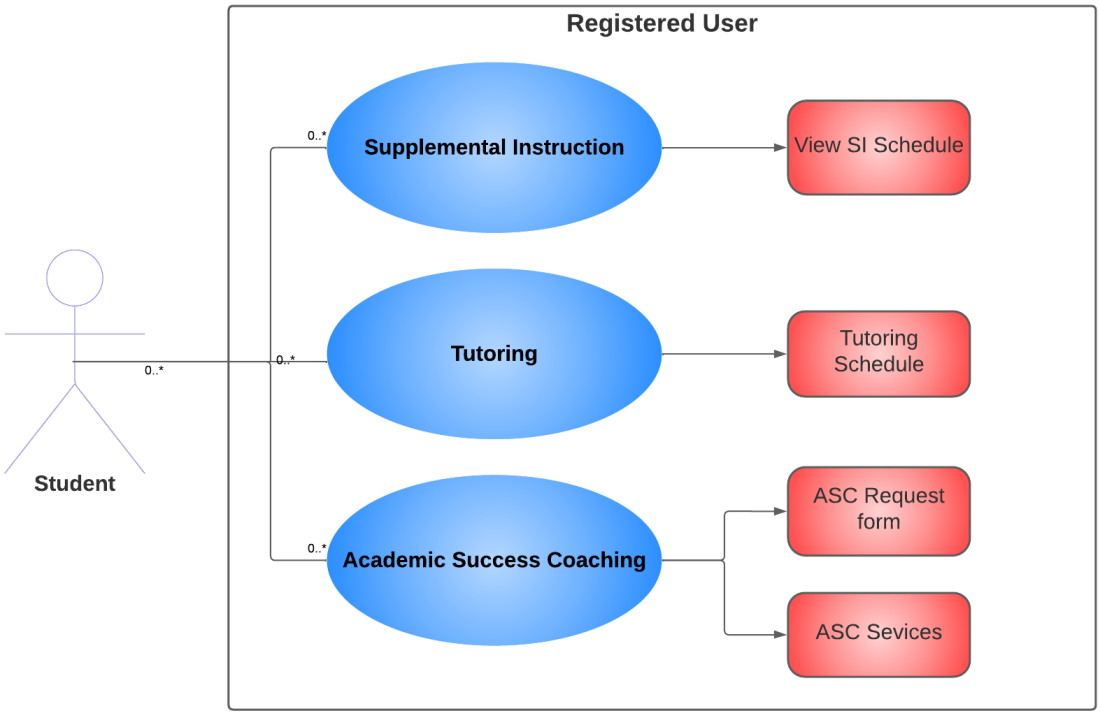
2.3.1 Registered Users (students)

2.3.1.1 Select Supplemental instruction via a touch interaction.

2.3.1.2 Select Tutoring via a touch interaction.

2.3.1.3 Select Academic Success Coaching via a touch interaction.

2.3.1.4 Students



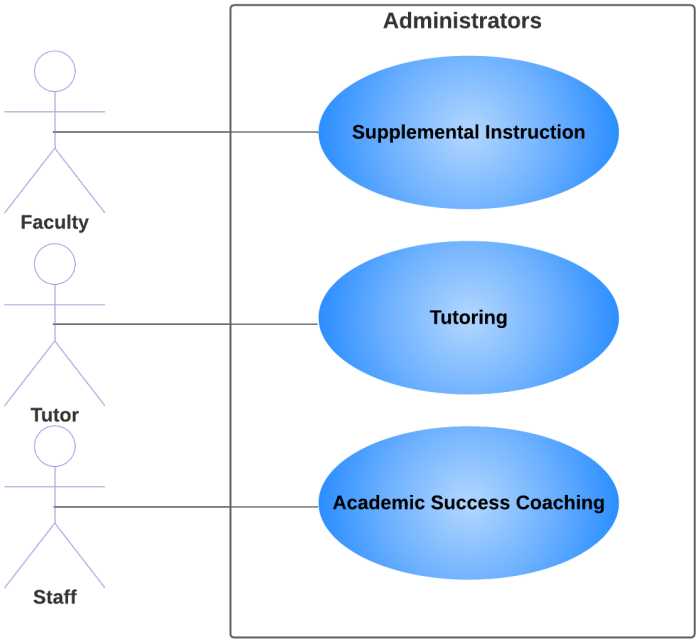
2.3.2 Administrators (Staff/Faculty/Tutors)

2.3.2.1 Staff would be allowed to make updates to scheduling.

2.3.2.2 Faculty would be in control in regards to the availability of Supplemental Instruction.

2.3.2.3 Tutors would have access to availability and information regarding tutoring services.

2.3.2.4 Administrators

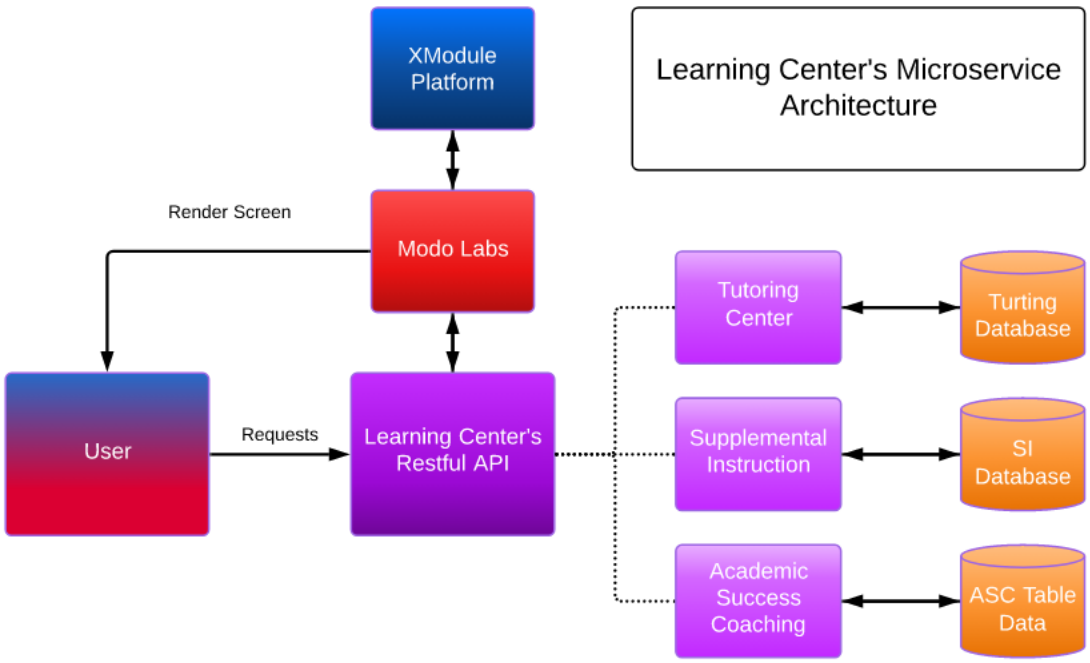


3. Design Overview

3.1 Introduction:

The applications system architecture will comprise various microservices. Each module/component will be its own microservice that will run each application process as a service. These services will communicate using responsive and lightweight API's. Each microservice will perform a single function; since each microservice will run independently, each service can be updated, deployed, and scaled to meet demand for specific functions of an application.

3.2 System Architecture:



3.3 System Interfaces

3.3.1 User Interface

3.3.1.1 The user interface will allow users to navigate through the application quickly with little to no complexity. The navigational functions will allow users easy access to all data according to what inputs were given. When navigating through the application, the application will communicate with the database through a simple API created by **LC**.

3.3.2 Software Interface

3.3.2.1 The API provided and created by **LC** will pull data from the database and present it to the users when the user interacts with the application.

3.4 Constraints and Assumptions

3.4.1 Assumptions

3.4.1.1 Users of the application are able to access the internet.
3.4.1.2 Users of the application are able to access California State University, Fresno's mobile application.
3.4.1.3 All supporting information about each subject is accurate and true.
3.4.1.4 The application's API and database are up-to-date and working.
3.4.1.5 The application will be designed and developed in accordance with the requirements specified in the SRS.

3.4.1.6 The application will be developed using the programming languages, frameworks, libraries, and tools specified in the project plan or by the client.
3.4.1.7 The application will be designed to be modular and scalable, with consideration given to future enhancements and upgrades.
3.4.1.8 The application will be designed to be compatible with existing systems and infrastructure, when applicable.
3.4.1.9 The application will be designed to be secure and protect user data and confidential information.
3.4.1.10 The application will be designed to be maintainable, with clear and well-documented code and modular components.
3.4.1.11 The application will be thoroughly tested with consideration given to both functional and non-functional requirements.

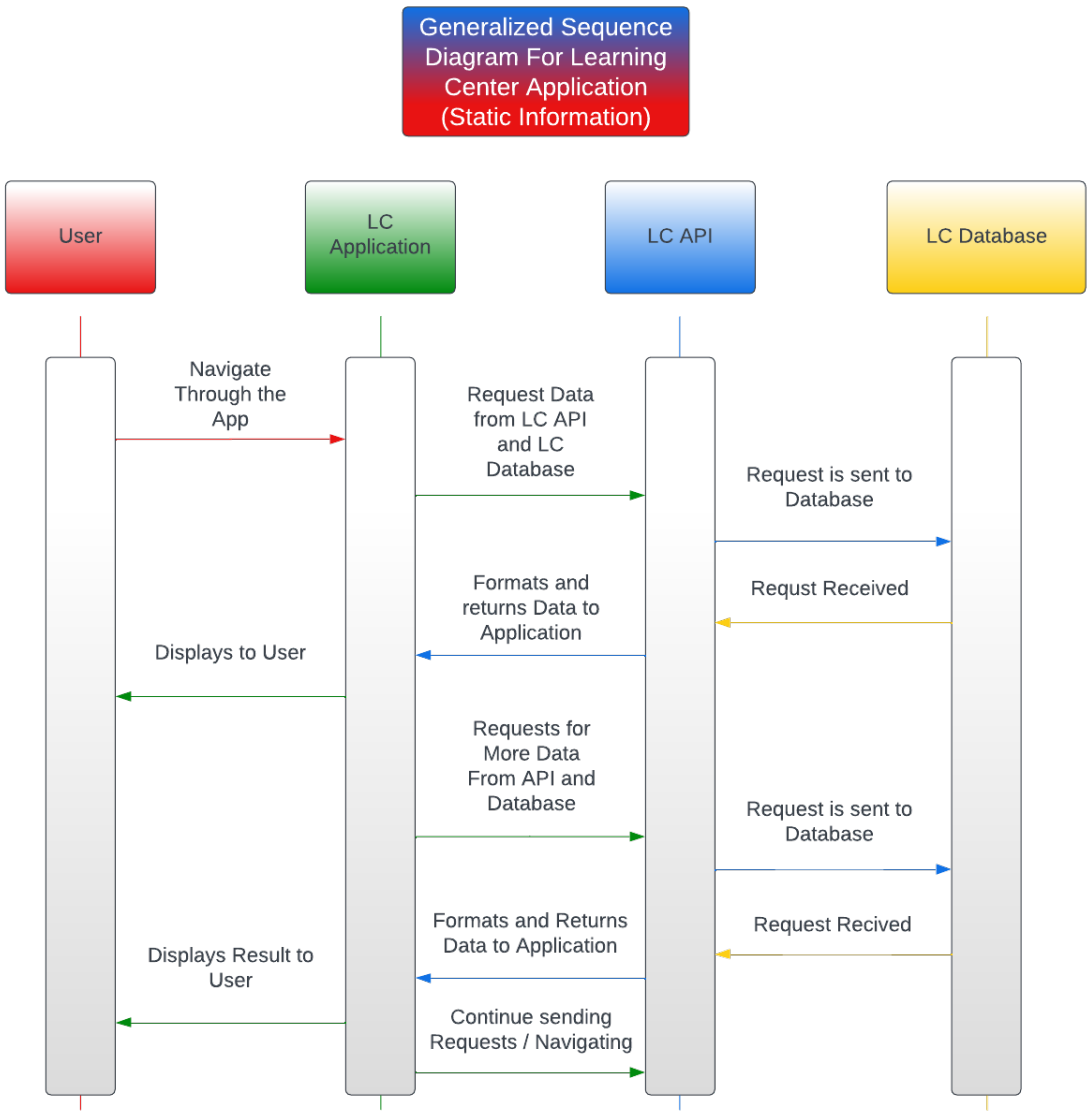
3.4.2Constraints

3.4.2.1 The application must comply with applicable laws and regulations, such as data privacy laws or industry-specific regulations.
3.4.2.2 The application must be developed within a specified timeline, and must be delivered on or before the agreed-upon deadline.

4. Dynamic Model

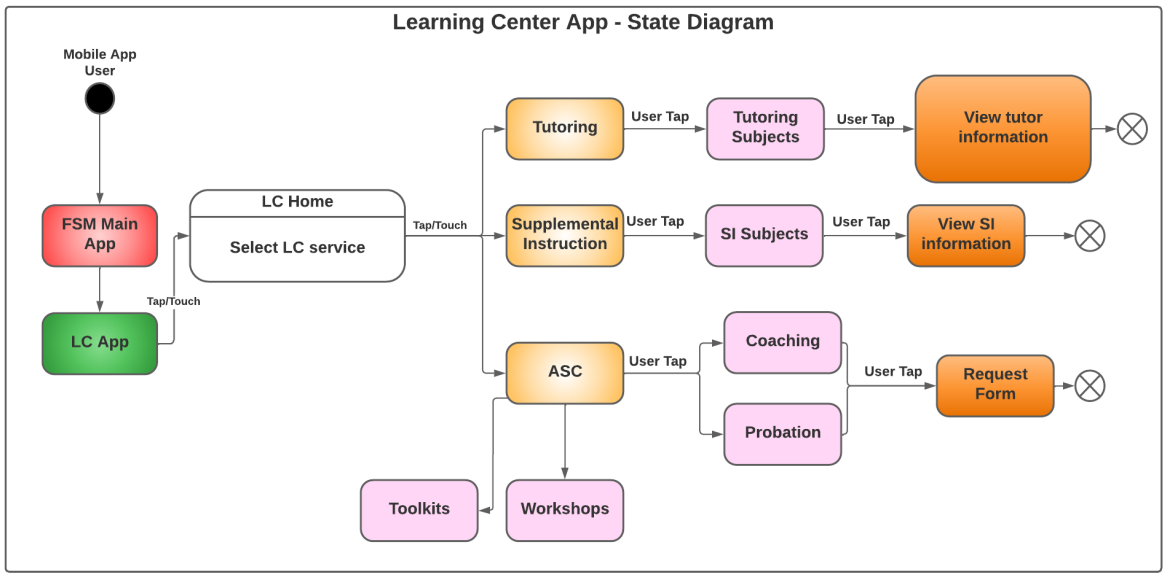
4.1 Sequence Diagrams

4.1.1 General Sequence Diagram for Static Information



4.2 State Diagram

4.2.1 Learning Center - All states


Learning Center App - State Diagram
Mobile App User
FSM Main App
LC App
Tap/Touch
LC Home
Select LC service
Tap/Touch
Tutoring
User Tap
Tutoring Subjects
User Tap
View tutor information
Supplemental Instruction
User Tap
SI Subjects
User Tap
View SI information
ASC
User Tap
Coaching
Probation
User Tap
Request Form
Toolkits
Workshops

5. Non-Functional Requirements

5.1 Performance Requirements

5.1.1 The application should be able to generate all data within a reasonable amount of time per user request.
5.1.2 The application should be able to have multiple users interact with the application without error.
5.1.3 The API and database provided by the application should be up-to-date and functioning at all times.
5.1.4 The application should be able to handle a large volume of user requests without experiencing significant delays or downtime.
5.1.5 The application should be able to handle peak loads during periods of high traffic without brimming unresponsive or experiencing degraded performance.
5.1.6 The application should be able to handle large amounts of data without slowing down or becoming unresponsive.
5.1.7 The application should be optimized for efficient data retrieval and processing to minimize the time required to generate data.
5.1.8 The LC API and database should have a high level of availability, with minimal downtime (99.999% availability) or not schedule maintenance windows during peak usage periods.
5.1.9 The application should have monitoring and logging in place to detect and diagnose performance issues.
5.1.10 The application should be designed with scalability in mind, to ensure that it can handle an increase in user traffic and data volume over time.
5.1.11 The application should be tested and optimized to ensure that it meets or exceeds the expected performance requirements, with regular performance testing and tuning to maintain optimal performance.

5.2 Design Constraints

5.2.1 The application must be designed with security in mind, and must adhere to the industry's best practices for data protection and cybersecurity.
5.2.2 The application must be designed to be maintainable and scalable, with consideration given to future upgrades and enhancements.
5.2.3 The application must be designed with usability in mind, with consideration given to the needs and preferences of end users.
5.2.4 The application must be developed using the programming languages, frameworks, libraries, and tools specified in the project plan or by the client.
5.2.5 The application must be tested thoroughly to ensure that it meets all requirements and functions correctly in all expected scenarios.

---------- End of Document ----------