CSC 431

UPlan

System Architecture Specification (SAS)

Group 5

Dana Feeney	Developer
Mariana Baquero Degwitz	Scrum Master
Jonathan Raskauskas	Developer

Version History

Version	Date	Author(s)	Change Comments
1.0.0	4/1/2021	Dana Feeney Mariana Baquero Jonathan Raskauskas	Initial Draft

Table of Contents

Table of Tables

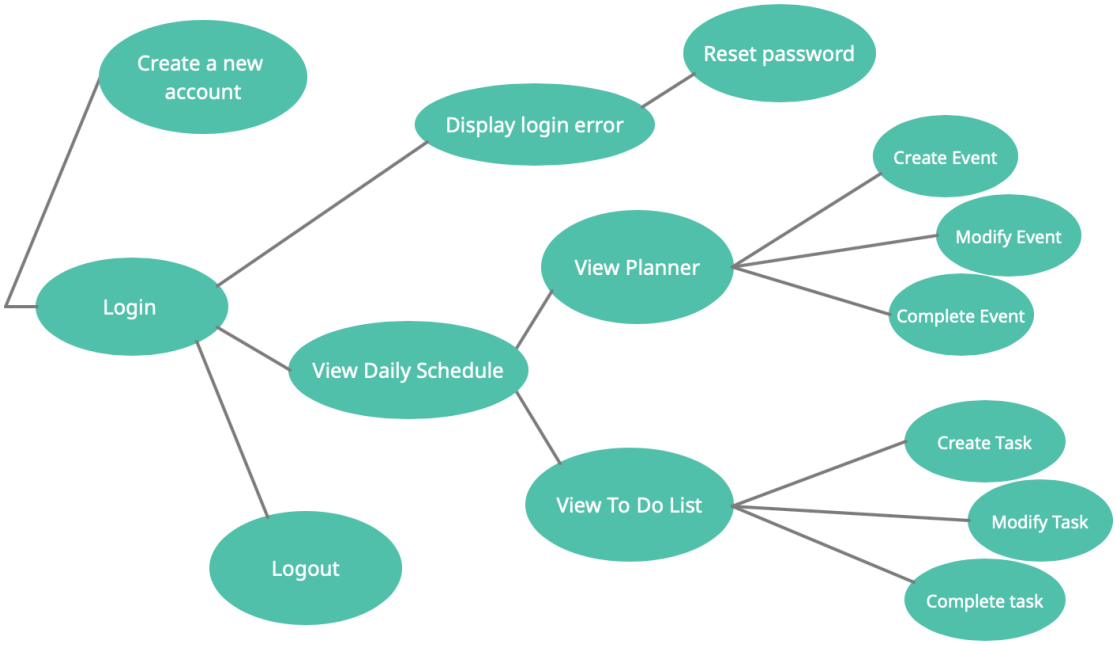
Table of Figures

1. System Analysis

1.1 System Overview

Our System will be composed of several components. Each component will interact with each other and have a general flow to them. The external factors that weigh on our system will be the users ability to use our user interface to create events (calendar events) and tasks (to-do events). The creation of these events will extend from a main class called External Event. These components once triggered will appear in the daily calendar component. Once an event is almost at deadline, the external event will trigger and the notification component will trigger.

1.2 System Diagram



1.3 Actor Identification

The only actor in our system is the user. The user will create the start of the external events that trigger the rest of our system.

1.4 Design Rationale

1.4.1 Architectural Style

Our architecture is event driven. We have multiple triggers as the events. If the user of the creates a calendar event or to-do task, that is the external event that is the trigger for our system to add the task to their daily view and to trigger the notification when the time is appropriate. We have based our system on this architecture choice.

1.4.2 Design Pattern(s)

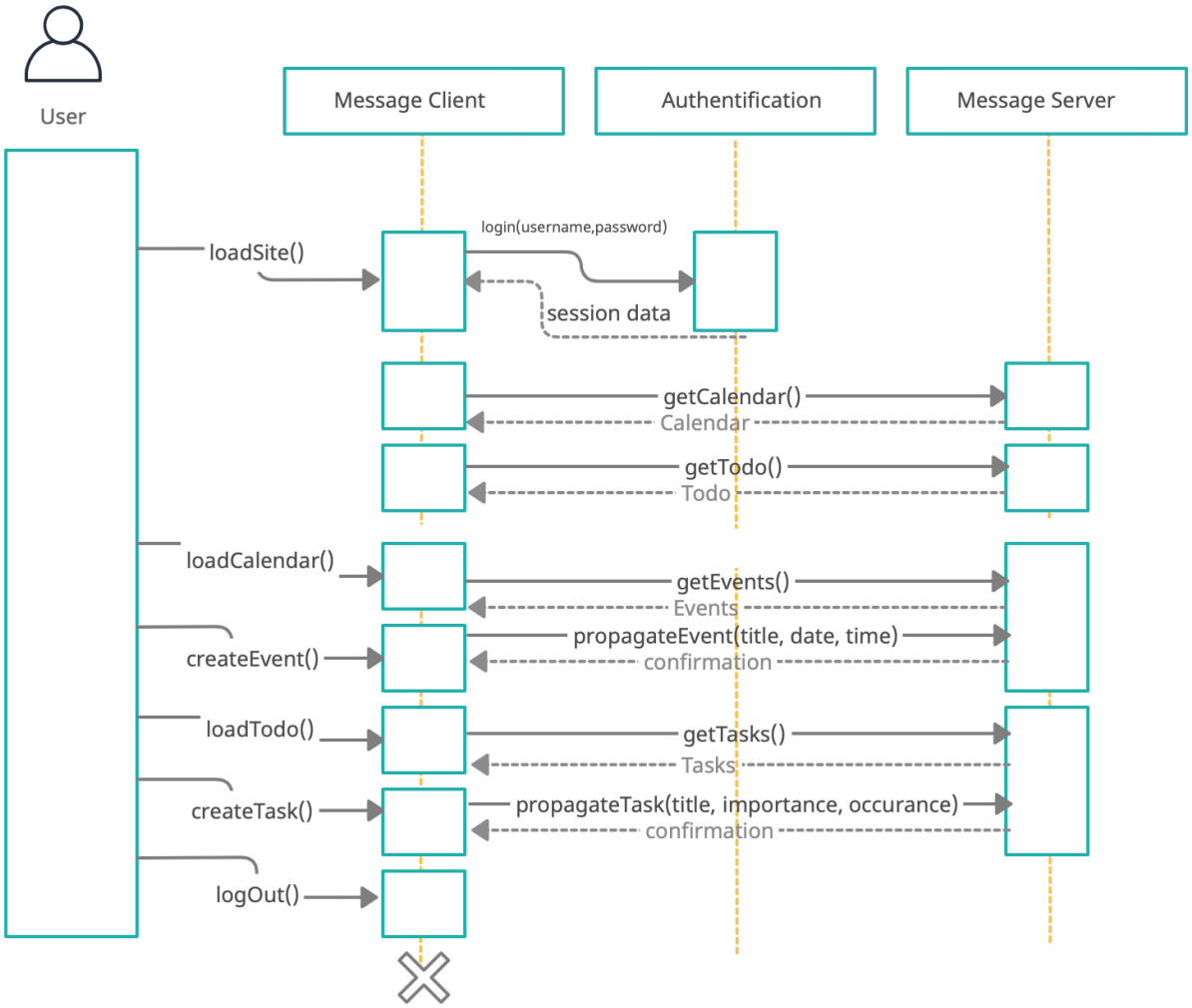
The design pattern most succinct with our chosen architecture is the Factory Method. This fits our overall design of the system because the event class that we are creating does not have to be known ahead of time, for the external factor (the user) is the one who actually chooses the class. Both of the two main events that the user can choose (event or task) fall under a central class, which we can call the external event.

1.4.3 Framework

We will be using the SOA framework. This framework fits our overall system and architecture choice because we want to have a focus on rapid development through the use of reusable components. We will also have broken down the framework into several layers, which will allow us to split our developers and allow them to focus on one section of the framework.

2. Functional Design

2.1 Functionality Diagram



3. Structural Design

3.1 Class Diagram