Sequence Diagram Document

Advisely, An Education Support Application

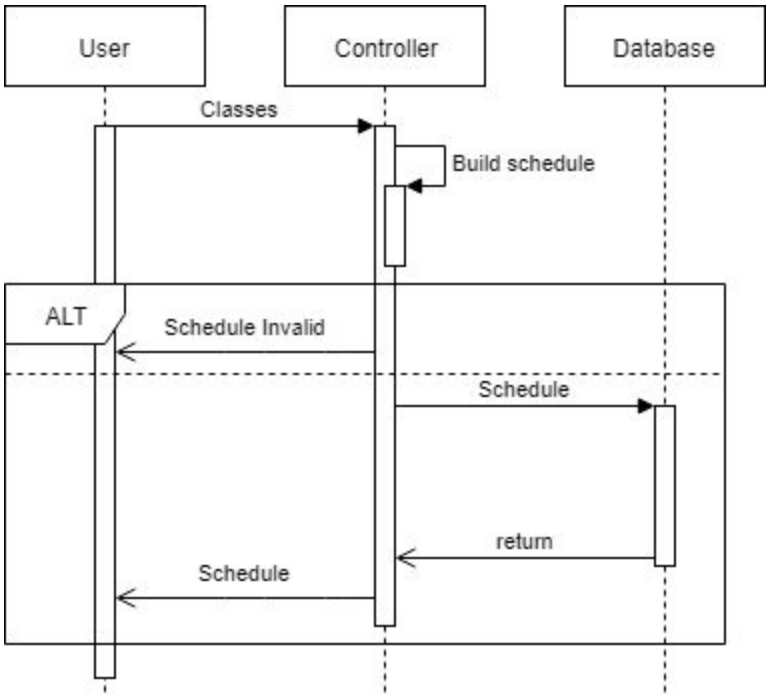
Big Muscle Boys

Sam Elliott, Jens Hansen, Connor Langlois, Eli Legere, Eric Schessler, Spencer Ward

COS 420 UMaine Spring 2019

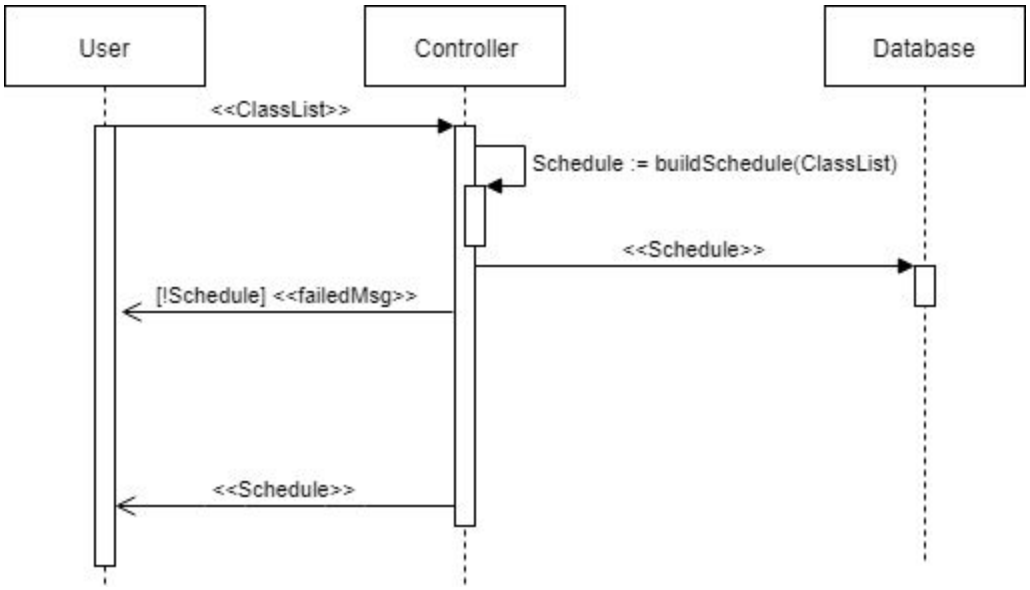
Professor: Sepideh Ghanavati

Build Schedule Analysis Sequence Diagram



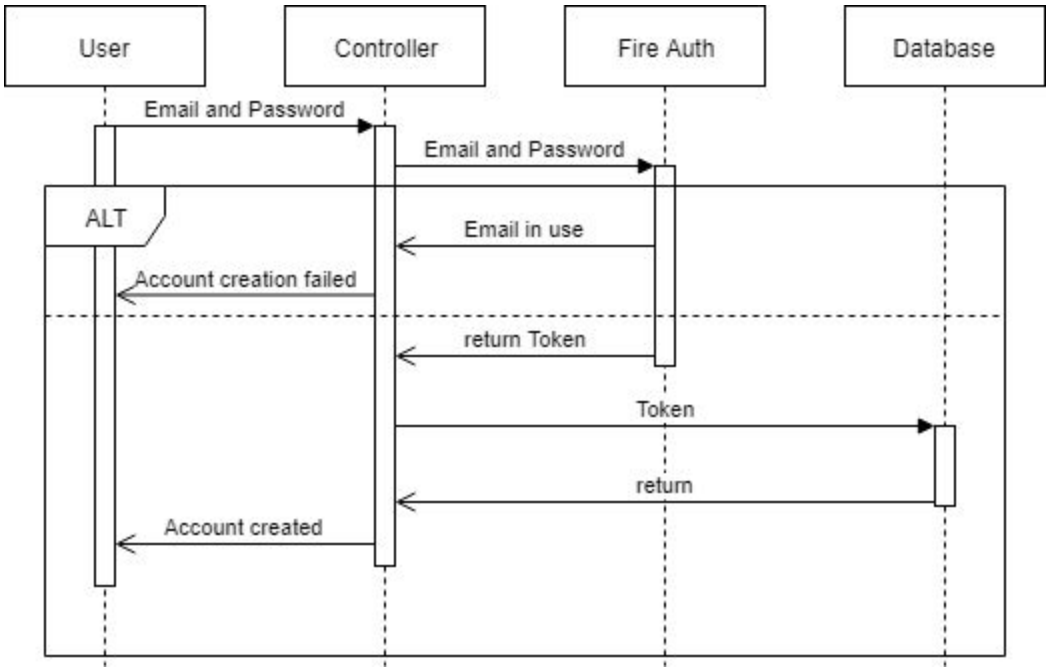
The build schedule sequence diagram describes how a user will create a schedule of their liking, and have it stored for later use. They will enter the classes they wish to take into Advisely, which will create a schedule object, if the schedule has no conflicts. If there are no conflicts in building the schedule, it is then stored in the database and returned to the user.

Build Schedule Design Sequence Diagram



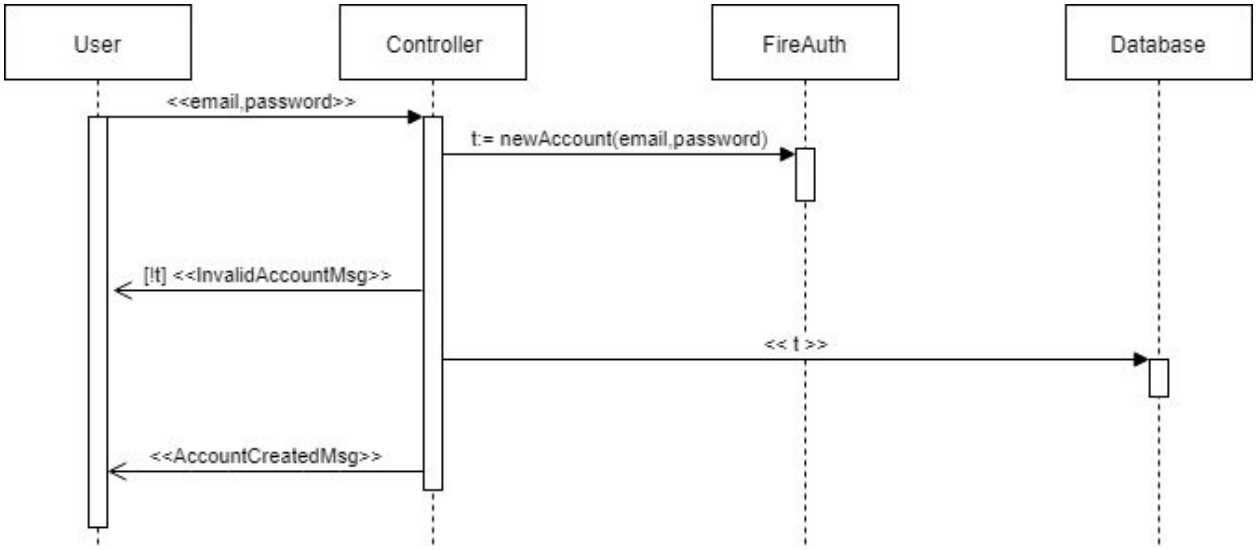
As with the analysis design document, this describes the process of creating a schedule given classes from the user. The user pasts a list of classes, classList. The application controller then builds the schedule, stored in the Schedule variable. If the schedule is invalid, it returns a failed schedule building message, failedMsg. Otherwise, it stores the schedule in the database and returns it to the user.

Create Account Analysis Sequence Diagram



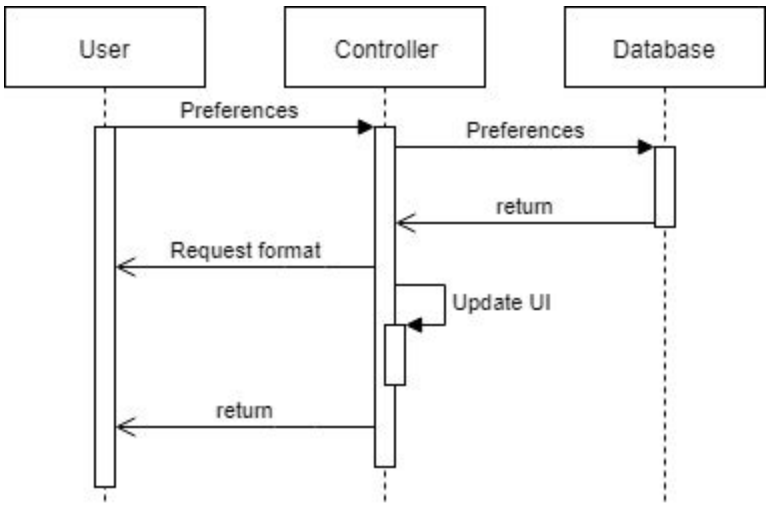
The user enters their password and email address. Firebase authentication then verifies the information given. If the email address is in use, an account creation failure message is returned to the user. Else, a successful token is return from firebase. This token is then stored in the Advisely database, and a success message is sent to the user.

Create Account Design Sequence Diagram



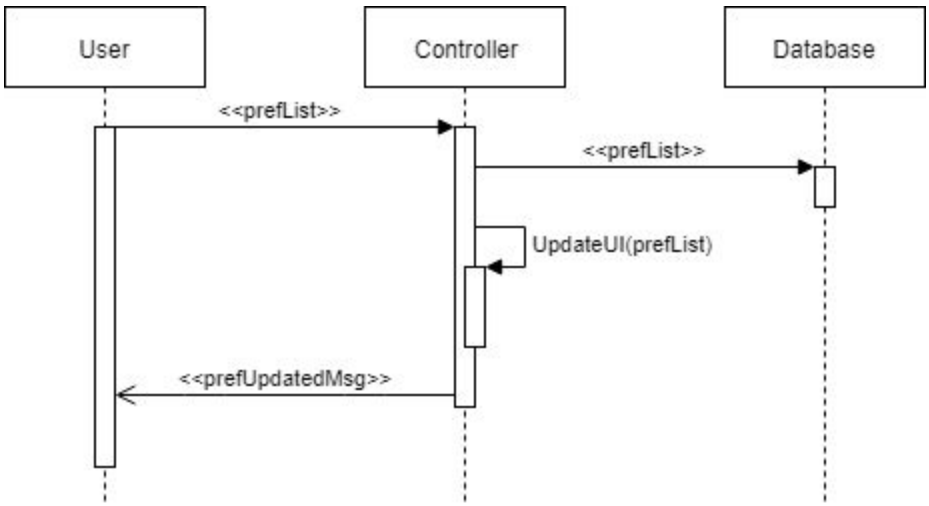
The user enters their email address and password. It then grabs a token from Firebase authentication, t.If this token is false, it returns an invalid account creation message. Otherwise, sends the token to the database and a success message to the user.

Customize Appearance Analysis Sequence Diagram



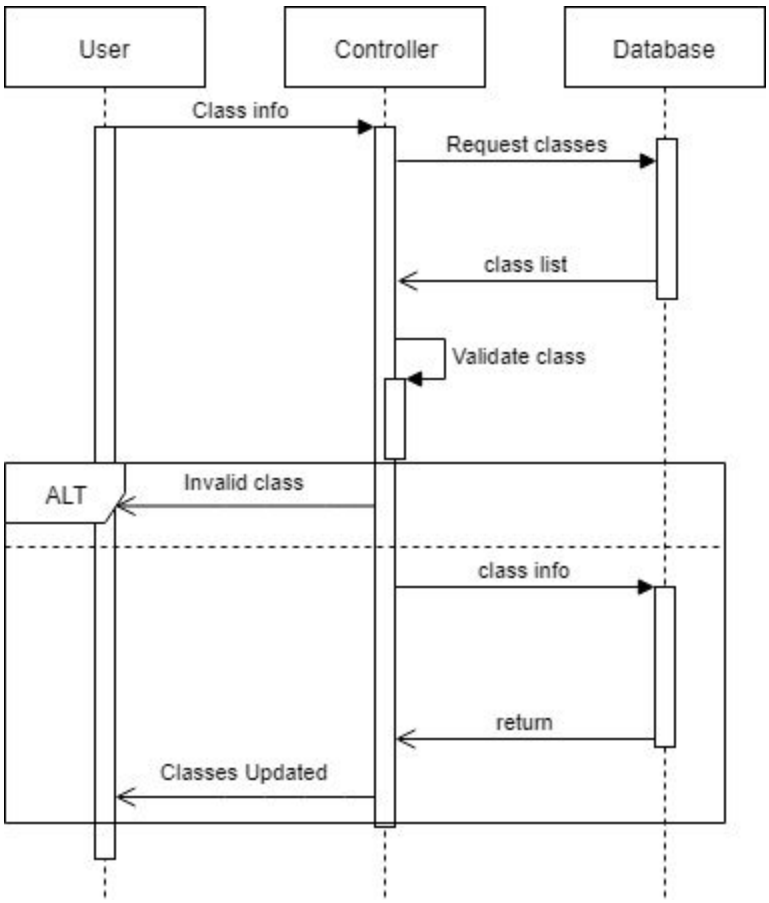
The customize appearance diagram displays the manner in which the user alters the appearance of the application. They enter their preferences. These are then sent to the database to be referenced later. The system then updates the UI and terminates the process.

Customize Appearance Design Sequence Diagram



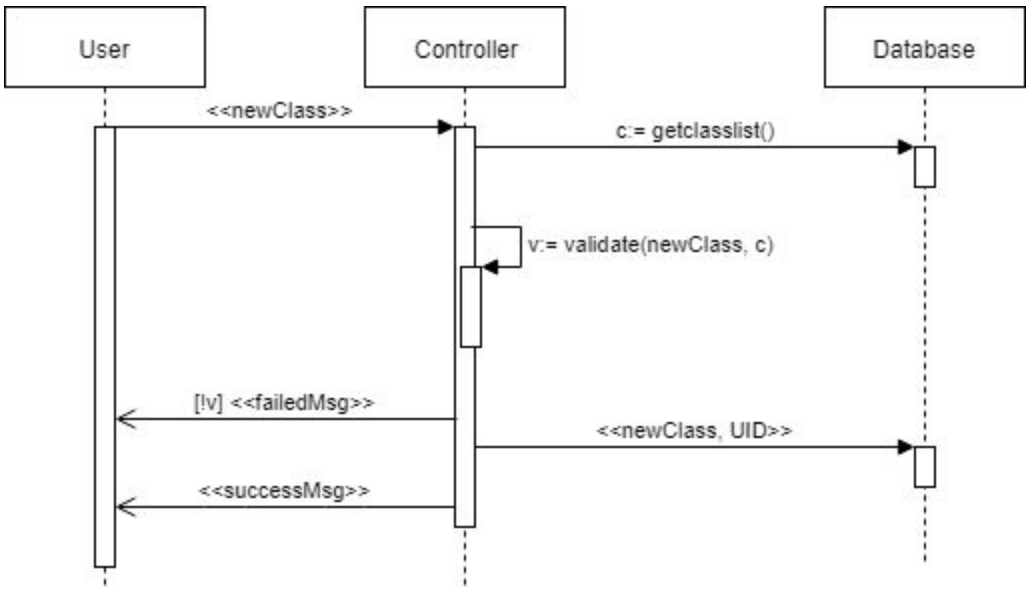
The user sends their preferences as prefList. The controller then updates this information within the database. The controller then updates the user interface based on prefList. It returns an update preferences message.

Import Previous Classes Analysis Sequence Diagram



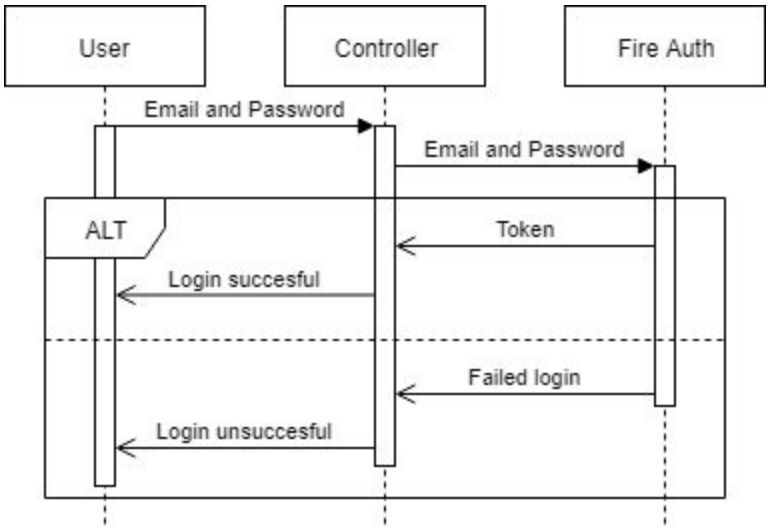
The user enters information about courses they have previously taken. The controller then attempts to pull those classes from the Advisely database. If a given class does not exist within the database, it sends an invalid class message to the user. Otherwise, it appends this course to the users completed courses list and sends a confirmation message to the user.

Import Previous Classes Design Sequence Diagram



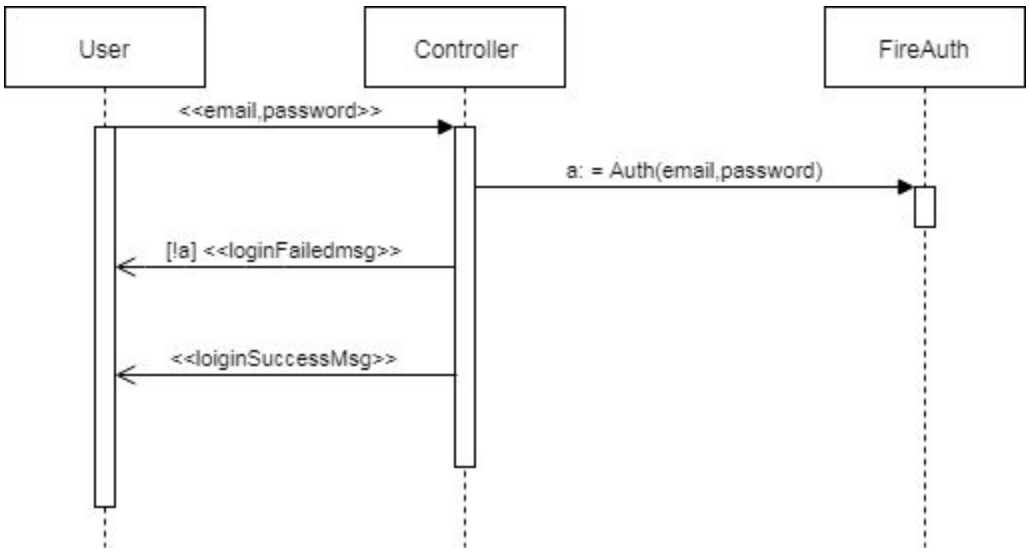
The user enters the classes they took previously as newClasses, a list of courses. The controller then grabs these classes from the database, as a list c. It then validates that the courses the user entered are valid. If not, the controller sends failedMsg, to let the user know that the information entered is invalid. Otherwise, it sends the newClass list as well as the account identifier to the database to update the list of courses the given user has taken. On termination, sends a succesful addition message.

Login Analysis Design Document



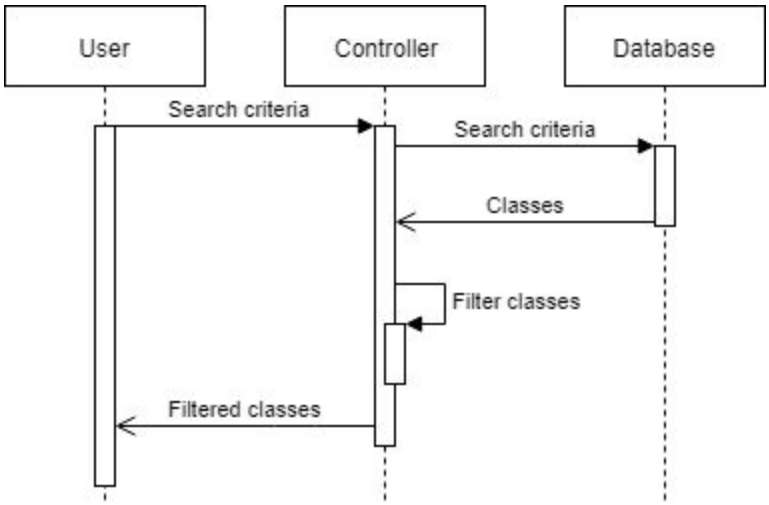
The user enters their email and password to the controller. It then validates this information with the firebase authenticator. If the login is successful, log the user in. Otherwise, send a login failed message to the user.

Login Design Sequence Diagram



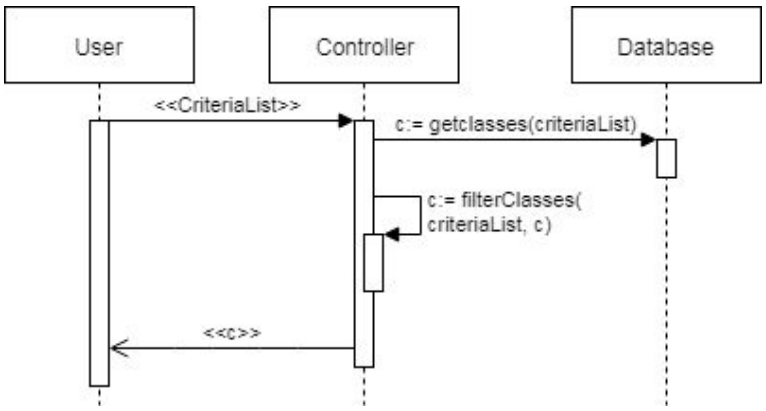
The user enters their email and password, named as such. It then sends this information to the Firebase authenticator, storing the resultant token as a. If the token is false, or the login failed, return a failed login message to the user. Otherwise, the user has logged in and return a successful login message.

Search Classes Analysis Sequence Diagram



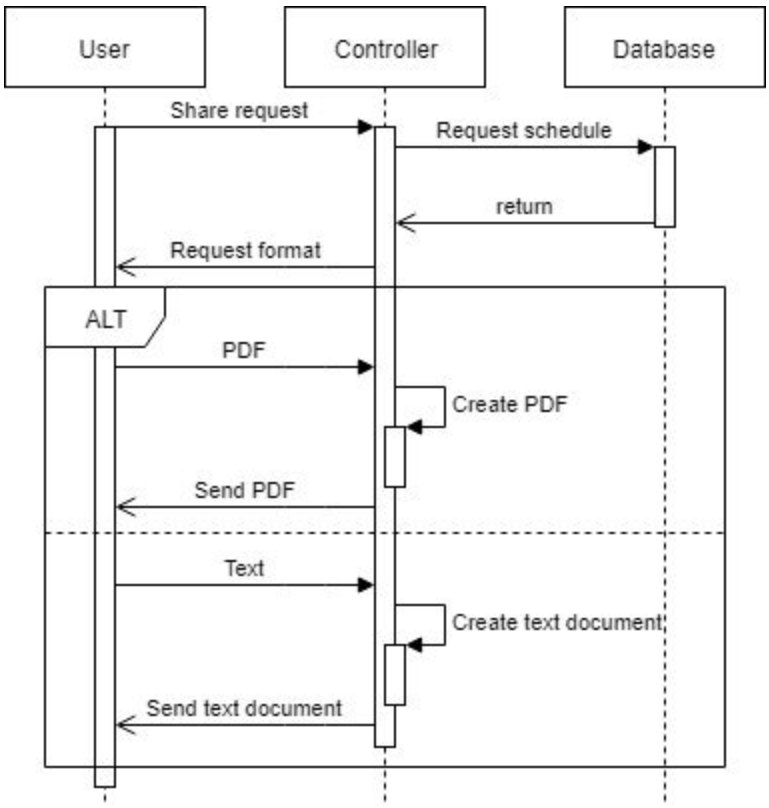
The user enters their search criteria. The controller then pulls classes from the database. The controller then filters the courses based on the criteria, and returns these filtered classes to the user.

Search Classes Design Sequence Diagram



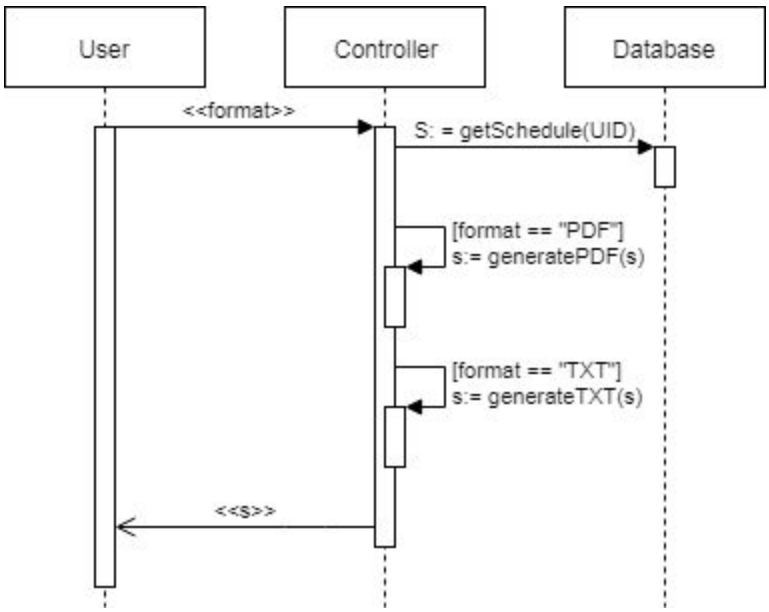
The user enters their search preferences, denoted CriteriaList. It then pulls courses from the database, based on the CriteriaList, storing the resultant courses as c. These courses are then filtered based on time constraints and other factors. The remaining courses are stored as c, and this list is the returned to the user for selection.

Share Schedule Analysis Design Document



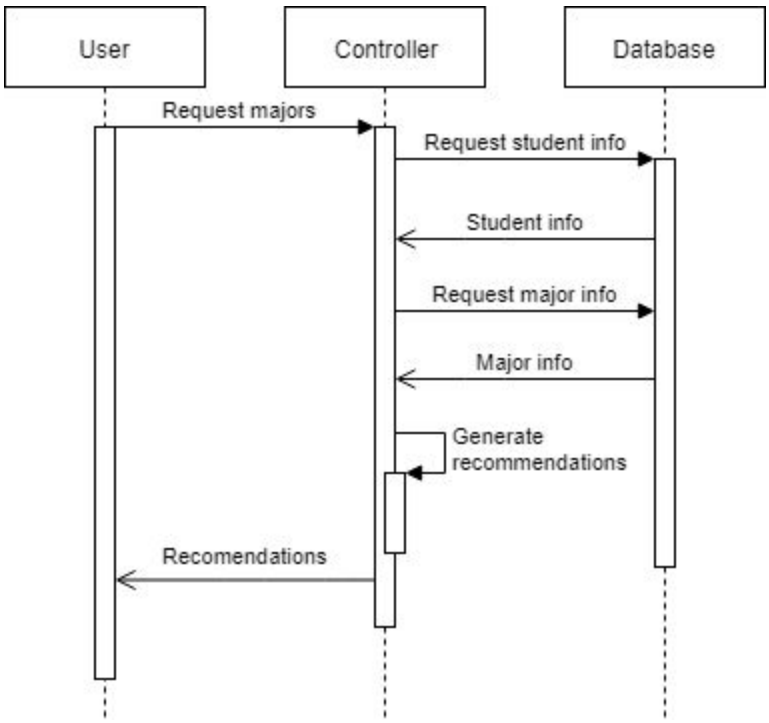
The user sends a request to the controller to share their schedule. The controller grabs the schedule from the database. The database then requests a format from the user. If the user requests a pdf, the controller creates a pdf document of the schedule and sends this to the user. Otherwise, the user requests text and the controller sends a text document to the user.

Share Schedule Design Sequence Diagram



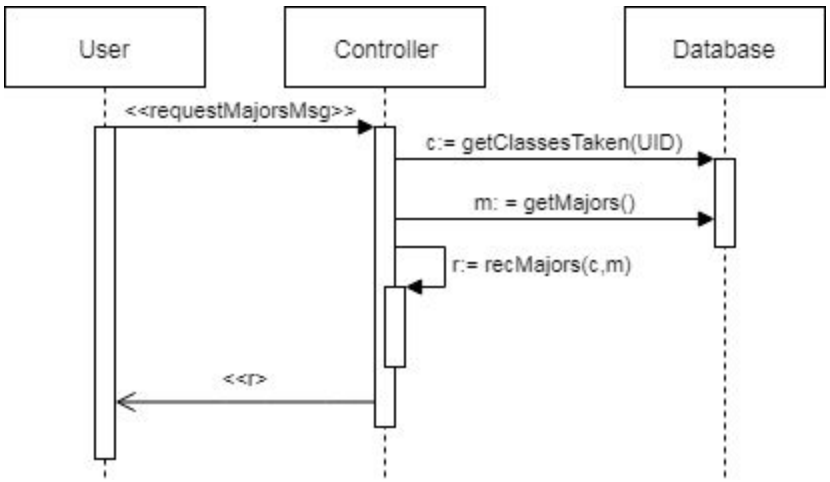
The user enters the format they want to output their schedule. The controller then pings the database for the users schedule, stored as S. A pdf or text document is then create and stored in S, based on the input format. This document is then sent to the user.

Suggest Majors Analysis Design Document



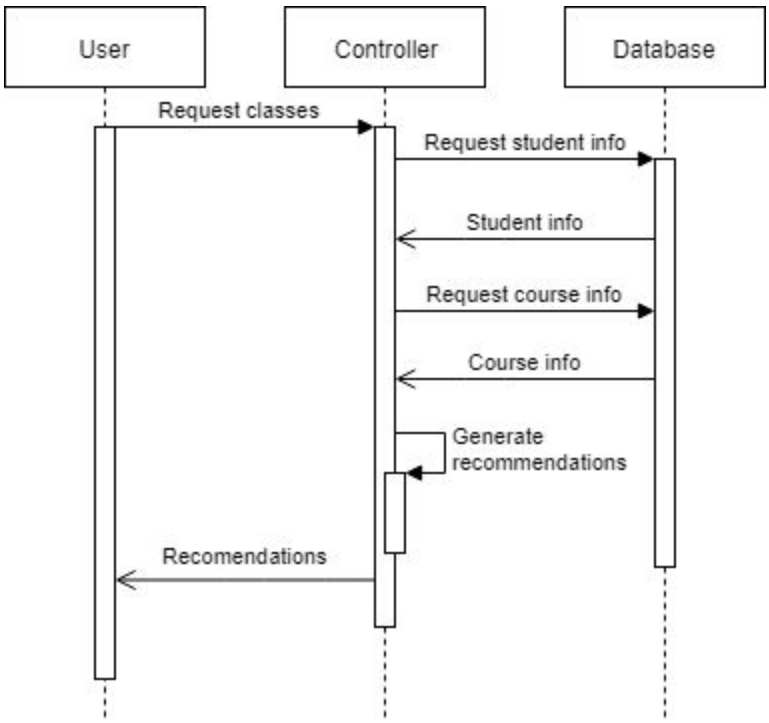
The user sends a request to the controller for major recommendations. The controller then pings the database for information about the student. Then the controller requests information about various majors that match the users information. The controller then generates recommendations and sends these to the user.

Suggest Majors Design Sequence Diagram



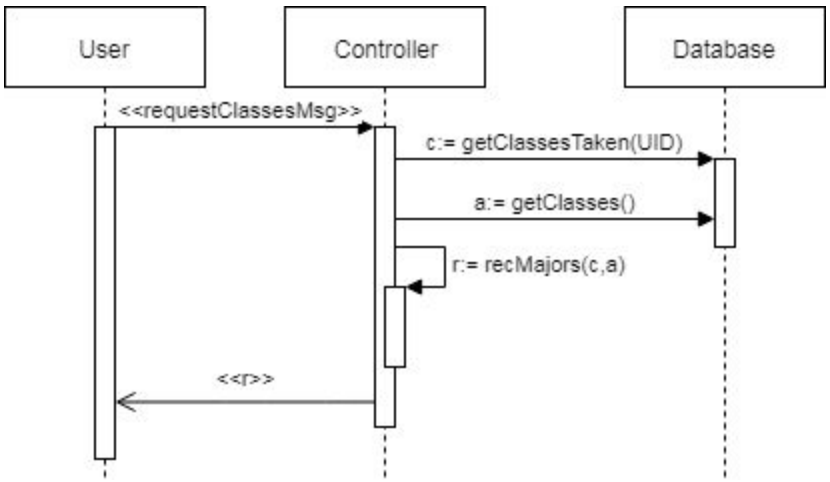
The user sends a request for majors to the controller. The controller then stores the classes the given user has taken as c. It then gets majors that relate to the user, stored as m. The controller then generates recommendations, r, and returns these recommendations.

Suggest Schedule Options Analysis Sequence Diagram



The user sends a request to the controller for classes. The controller then retrieves information from the database about the courses the user has taken. And about other optional courses the user may take. The controller then generates course recommendations, and returns those to the user.

Suggest Schedule Options Design Sequence Diagram



The user sends a request to the controller for classes. The controller then retrieves information from the database about the courses the user has taken, stored as c. And about other optional courses the user may take, stored a. The controller then generates course recommendations, r, and returns those to the user.