Devoir

Software Design Document

February14, 2015

Project Team

Candice Davis
Alan Dayton
Brady Kelley
Cory Kirkland
Skyler Mitchell
Adam Nelson
Brent Roberts
Seunghee Yang

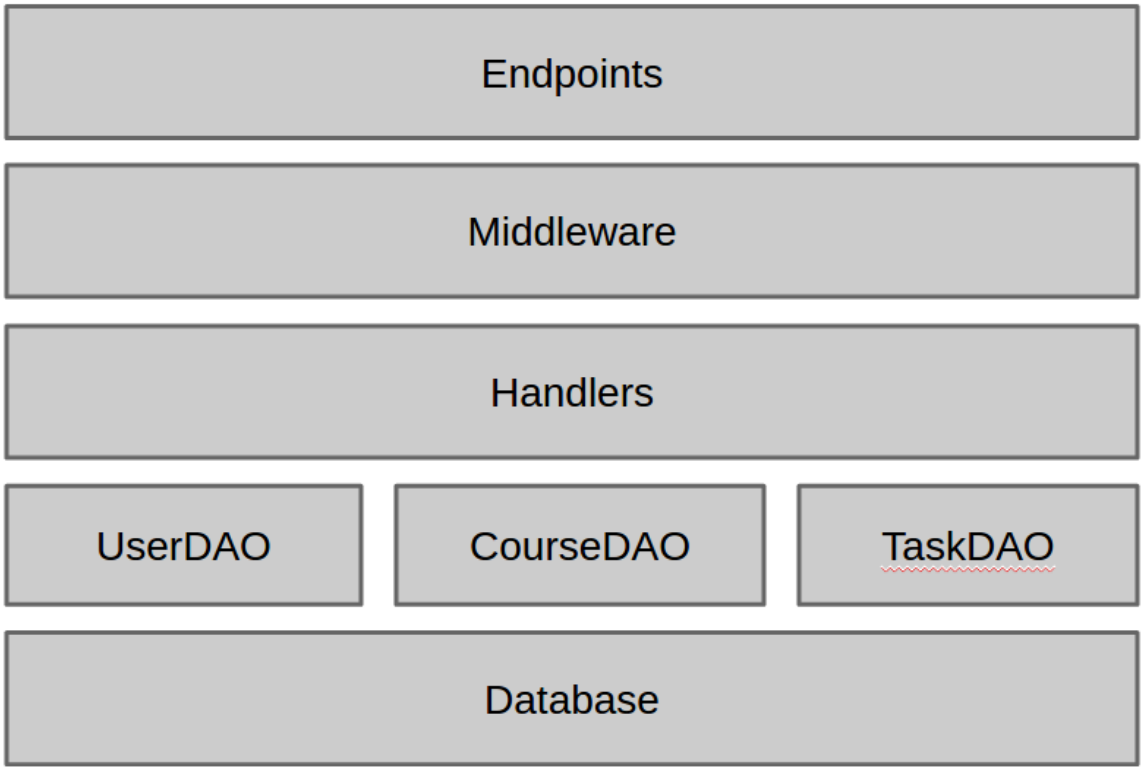
Introduction

This software design document describes the architecture and system design of Devoir, a multi platform homework tracking app. Devoir will be available on 3 platforms: iOS, Android, and web. The following outlines the interactions between each of the platforms and the central server.

Server

Overview

The server consists of several parts. The web and mobile apps will interact with the server through the public endpoints. Then requests are authenticated through the use of middleware. From there the requests are handled normally. If the requests involve any data stored in the database, the database is accessed through database access classes.



Endpoints

The web and mobile apps will interact with the server through the following API:

Users

GET /api/users
Parameters: NONE
Returns a list of all users. For use by admin only.

POST /api/users
Parameters: username, email
Creates a new user

GET /api/users/{userID}
Parameters: userID, token
Returns the specified user

PUT /api/users/{userID}
Parameters: userID, token, username, password, email
Updates the specified user

GET /auth/google
Parameters: none
The request will be redirected to Google for authentication

POST /api/logout
Logs out the user

Courses

GET /api/courses
Parameters: userID
Returns a list of the user's calendars

POST /api/courses
Parameters: userID, name, color, visible
Imports the course if there is an ical feed. Otherwise, creates a course from information provided

GET /api/courses/{courseID}
Parameters: userID
Returns the specified calendar

PUT /api/courses/{courseID}
Parameters: userID, name, color, visible, icalFeed
Updates the specified calendar

DELETE /api/courses/{courseID}
Parameters: userID
Deletes the specified calendar

Tasks

GET /api/courses/{courseID}/tasks
Parameters: NONE
Returns all the tasks for a particular calendar

POST /api/courses/{courseID}/tasks
Parameters: userID, name, description, startDate, endDate, completed, visible (optional)
Creates a new task belonging to the specified calendar

POST /api/courses/{courseID}/tasks/import
Parameters: icalFeed
Imports events from icalendar into course

GET /api/tasks/{taskID}
Parameters: NONE
Returns the specified task

PUT /api/tasks/{taskID}
Parameters: userID, name, description, startDate, endDate, completed, visible (optional)
Updates the specified task

DELETE /api/tasks/{taskID}
Parameters: userID
Deletes the specified task

Middleware

All requests will be authenticated by the use of middleware. This will involve checking user credentials (the user ID and a token).

Handlers

Every server endpoint has a handler function that will service the request. If the request requires interaction with the database, the handler function will call a method on the appropriate database access class.

Database Access Classes

Every access to the database will go through a database access class. This is more secure and easier to maintain. This layer will interact directly with the database through a pool of connections. The database access classes will be as follows:

UserDAO
+ getUserById + getAllUsers + createUser + updateUser + deleteUser + loginUser + logoutUser

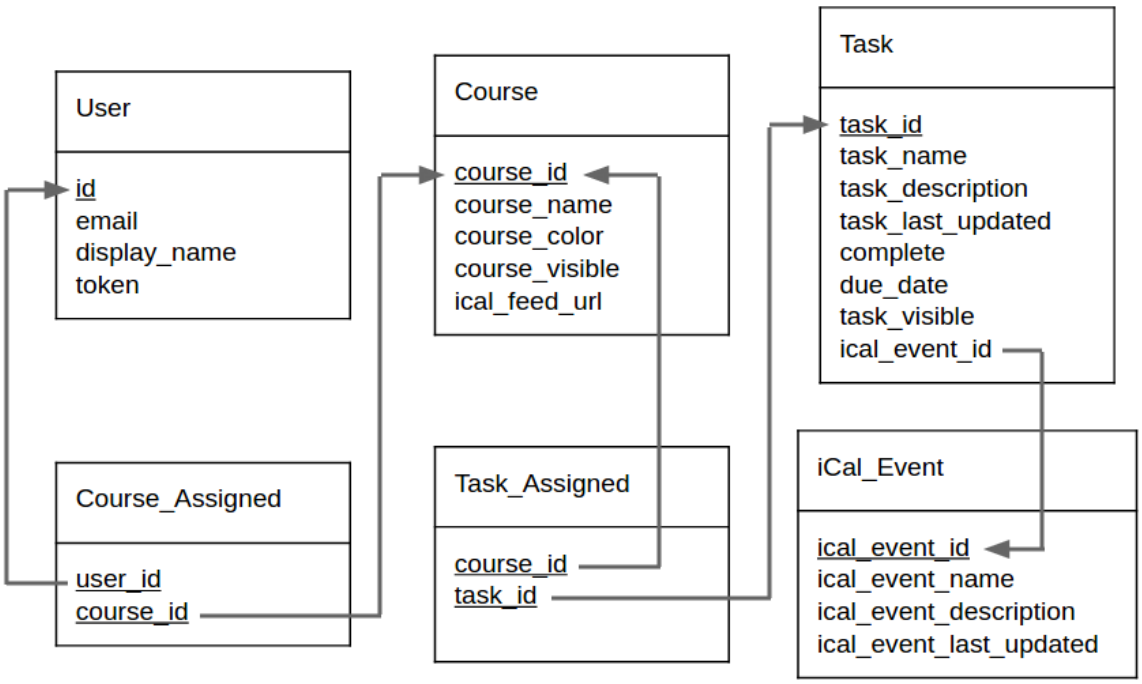
CourseDAO
+ getCourseById + getCoursesByUserId + getAllCourses + createIcalCourse + createCourse + updateCourse + deleteCourse

TaskDAO

+ getTaskById
+ getTasksByCourseId
+ getTasksByUserId
+ getAllTasks
+ createTask
+ updateTask
+ deleteTask

Database

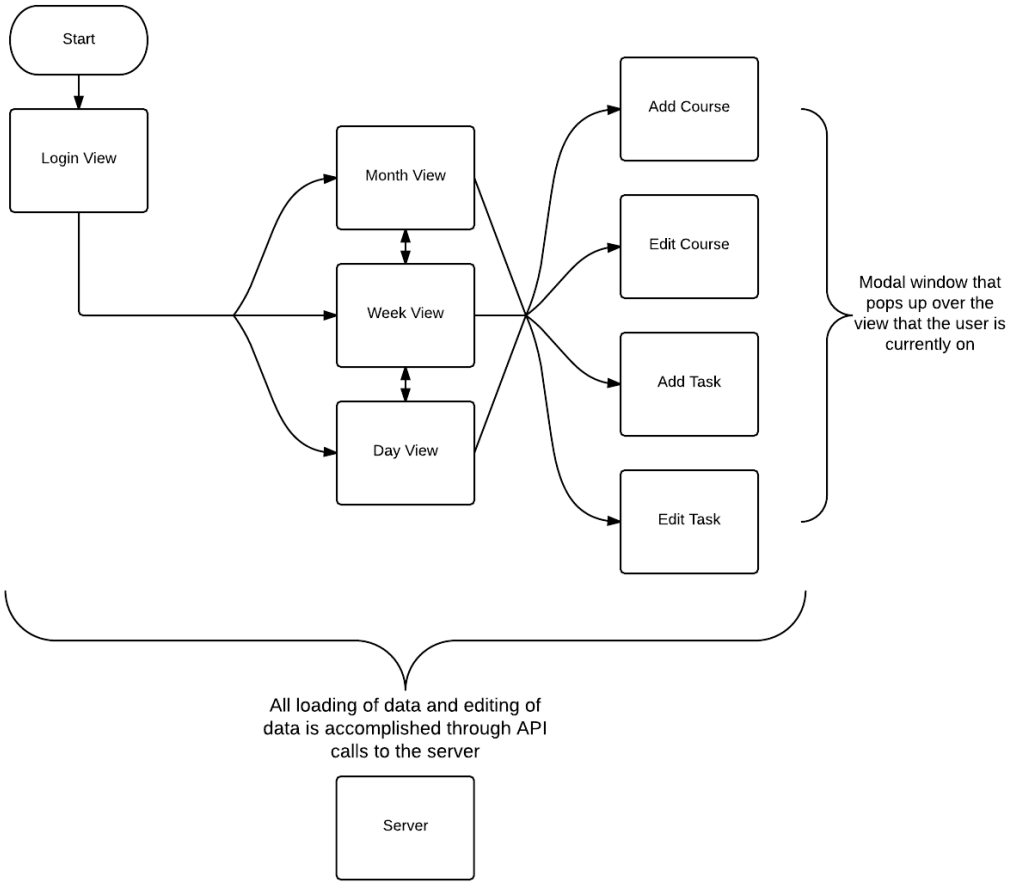
A PostgreSQL database will be used to store all information about users, courses, and tasks. The schema will be as follows:



Web

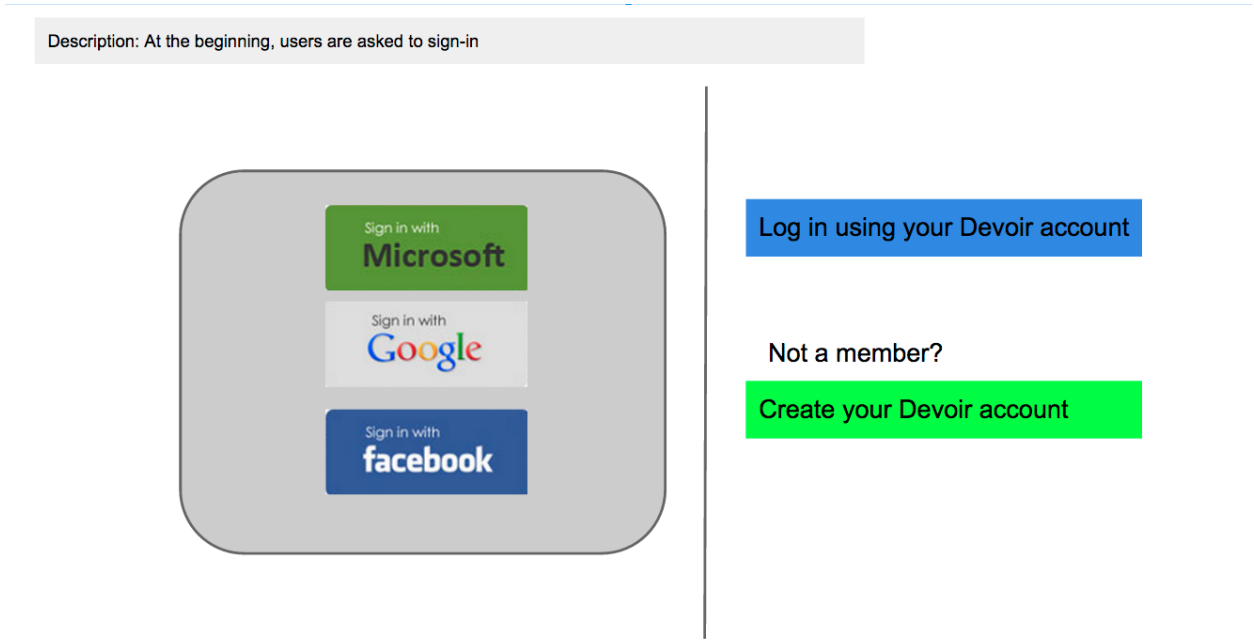
User Flow

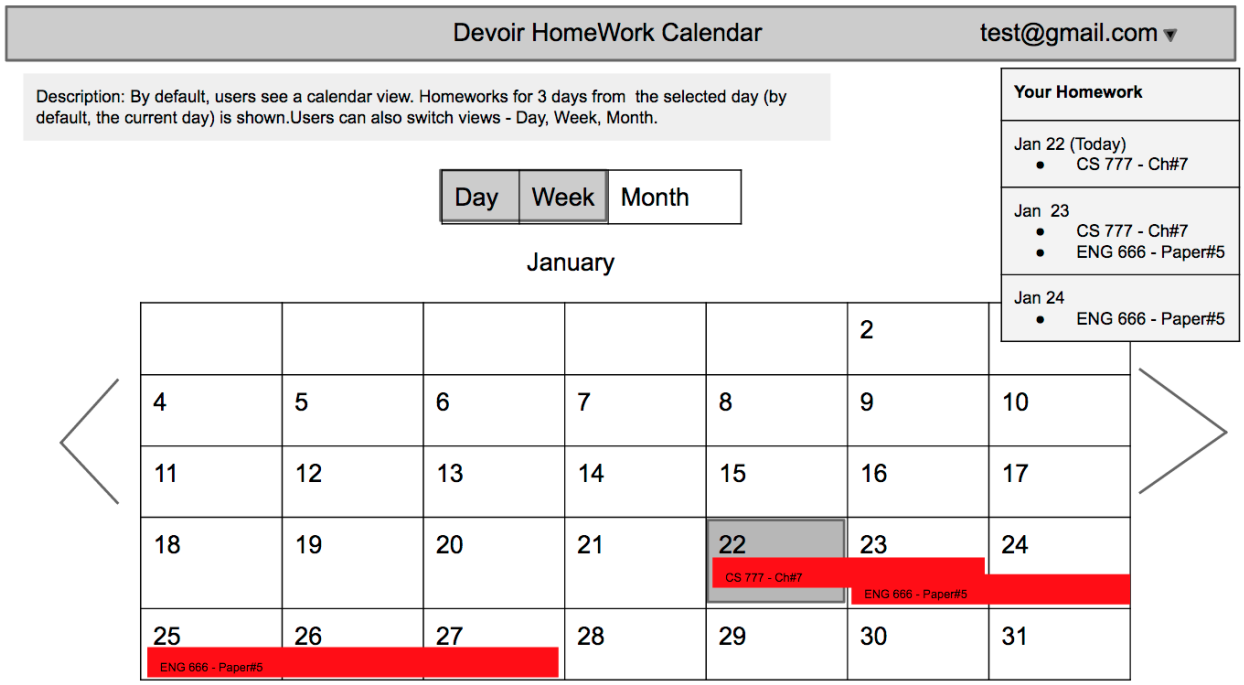
When a user lands on our website they will first see a login page. After they have successfully logged in they will be taken to the main page of the site. The main page shows a calendar that has all of the users assignments on it. There are tabs so that the user can switch between the calendar view, a week view and a single day view. When a user wants to add a new iCal feed or an event to an existing feed a modal window will pop up so they can enter information. After they have entered in all the needed information the modal window will be dismissed.



Website User Interface

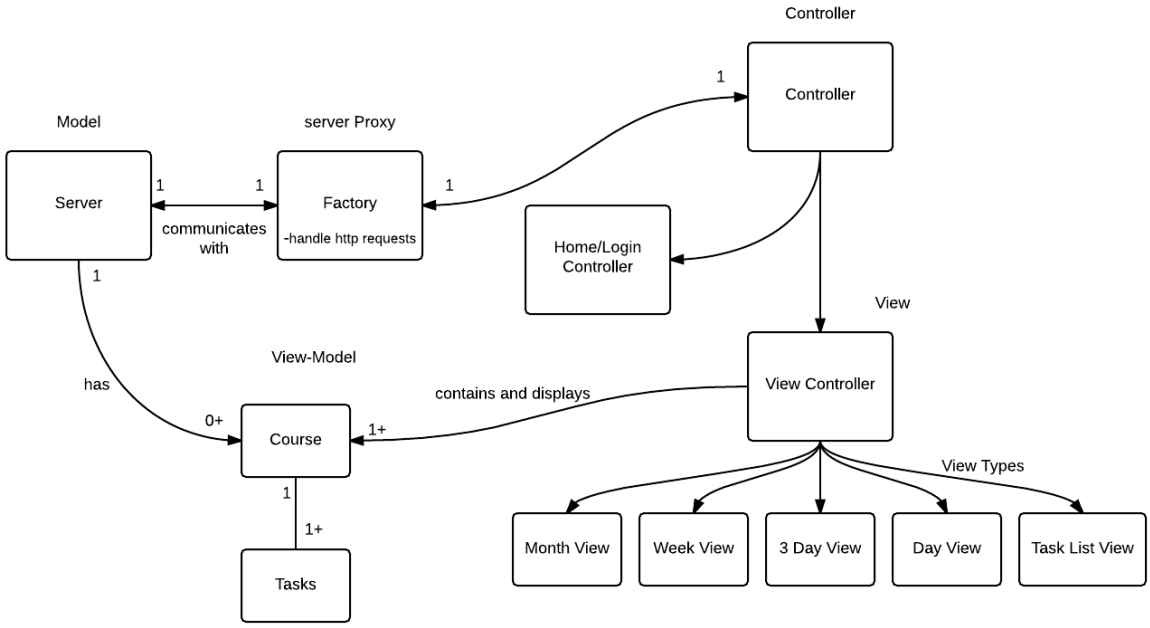
Shown below are a couple simple mock ups of what the website login and main page will look like.





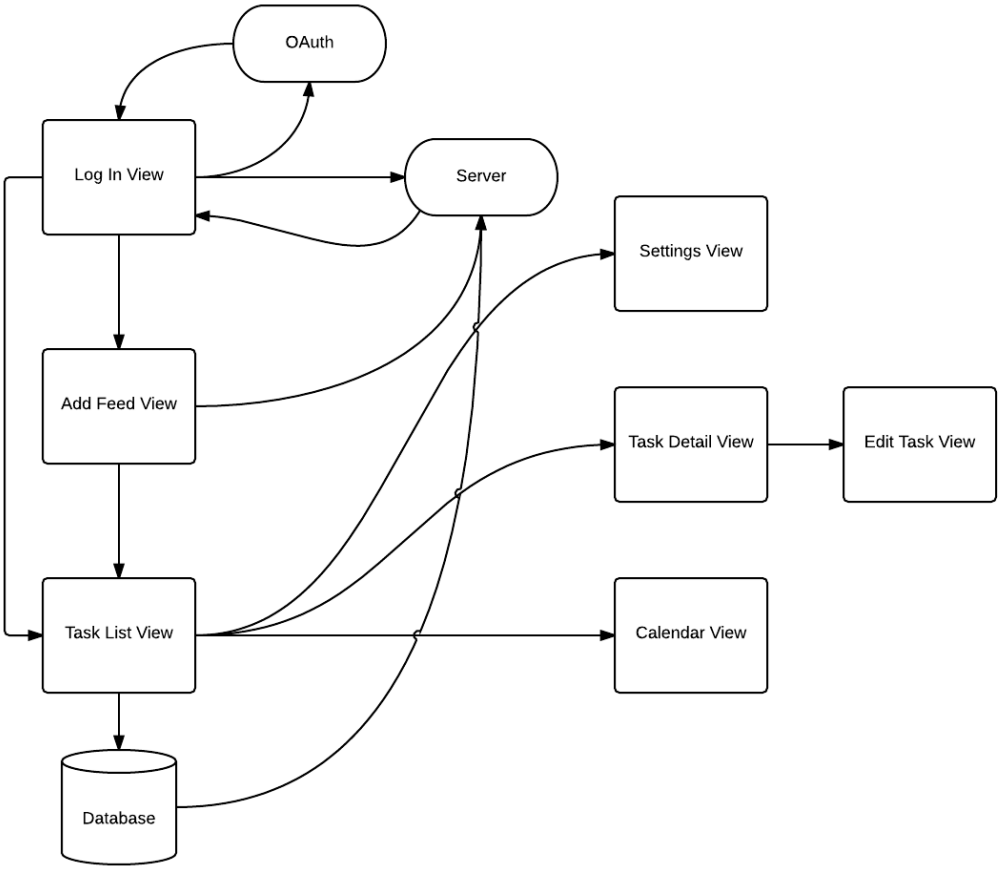
UML-like Chart of the Components of the Website

Below shows the basic components of the website front end and how all the parts interact with each other and the server.



Mobile

User Flow



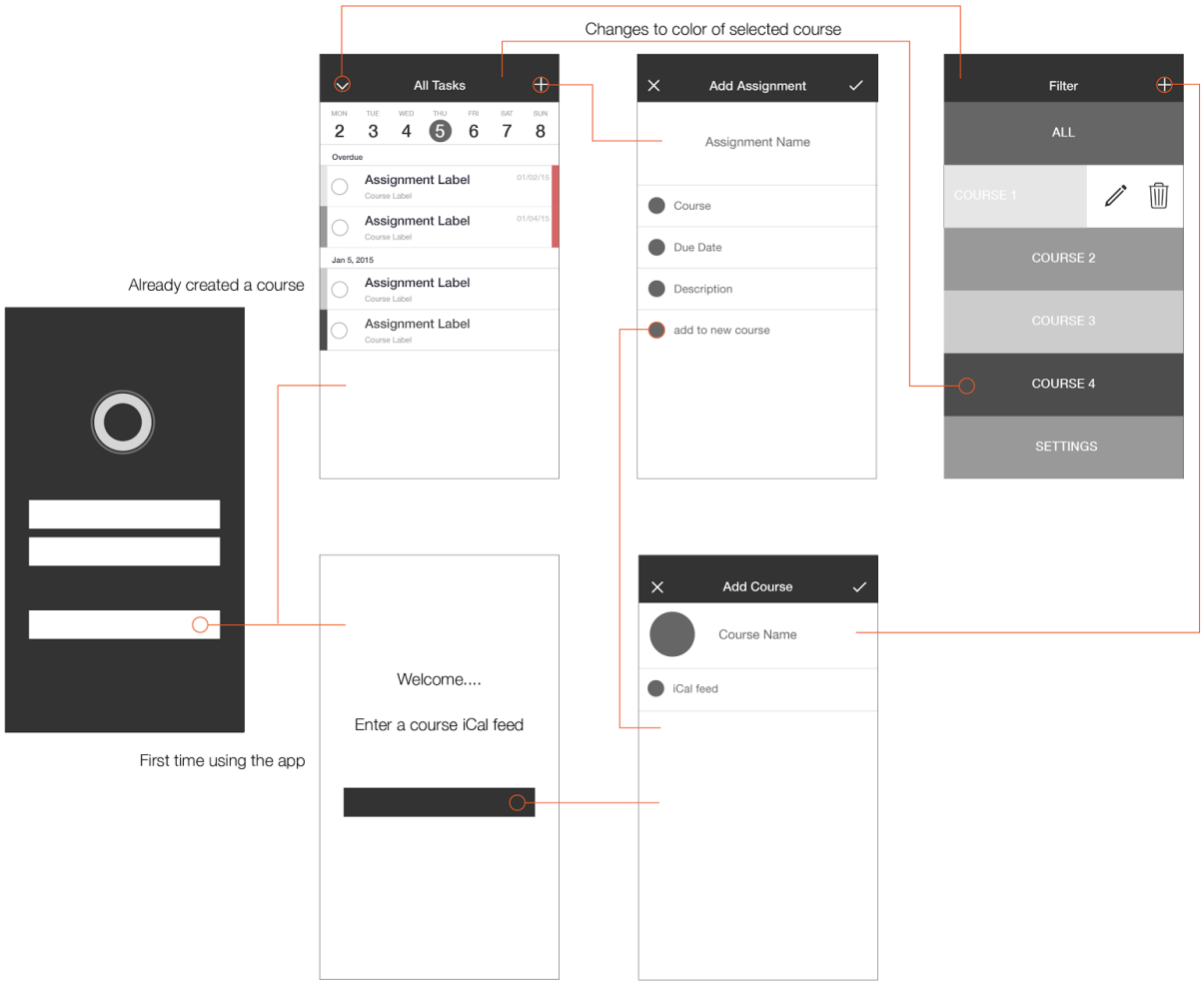
When a user opens the app, they will first go to the log in view. After authenticating with Google, we will send the user's token to the server, where it will be stored. If the user has already logged in before, then this process will be skipped. The user will then be sent to either (a) the add feed view, if and only if they have no iCal feeds set up for their account. Otherwise, the user will be sent to the task list view. From here the user has the option to go to the settings page, a page to display all details for a specific task, or to the calendar view, from which they can navigate to a list view for the specified day.

It is important to note that while in the task list view, the user has the ability to enter a new mode by long clicking a task item. In this mode the user will be able to hide or show any task. Also, the user

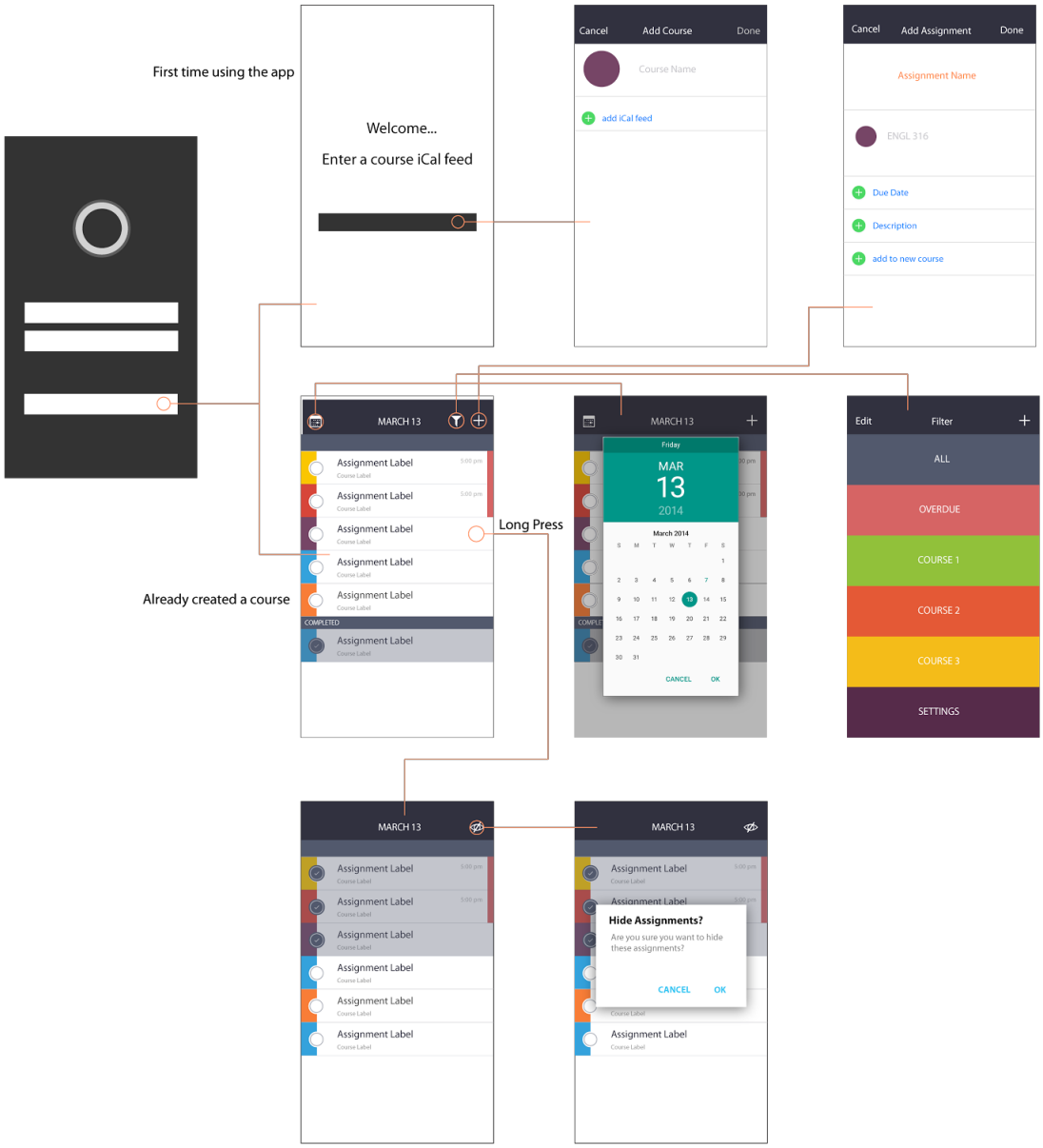
will be able to mark tasks as completed. This information will be stored in the local database, and eventually synced with the server. Further, after the user goes to the task detail view, they will have the ability to edit the task.

Data storage on the mobile devices with be done using SQLite. The schema on all mobile platforms will mirror that of the server. The purpose of using local data storage is to allow the user to perform actions with or without a network connection. If actions were taken without a network connection, the app will sync with the server upon acquiring a connection. Any conflicts in changed data will be resolved via the "last updated" property. That is, whatever platform has the latest "last updated" property will trump all others.

iOS User Interface



Android User Interface



Notable differences in the Android UI flow include the ability to long-press on an assignment to enter "edit" mode. This is a common pattern on Android devices and allows the user to hide irrelevant items that may clutter the task view. Also, a calendar button is included in the task view which allows the user to bring up a date picker to quickly jump to a different day. The user may also use a swiping gesture (left or right) on the task view, which will increment or decrement the date respectively.