Software Engineering

Final Document

California State University

SAN MARCOS

11/29/2019

Milestone Project

Jordan Mower

Jordan Nienaber

Jonathan Hua

Nick Luckey

Table of Contents

Introduction

The goal of the “Milestone” application is to provide a personal planner for students of CSUSM. It is intended to allow users to store courses and tasks so that they have an idea of upcoming events. This application also allows users to subscribe to each others courses and consequently their tasks. These are all presented to users in an easy and intuitive way.

System Functions

3.2.1.1 Functional Requirement 1 - Registration

The system shall allow users to register using a username, password, school name, grade/year and date of birth. A username will be 8-20 alphanumeric characters. A password will be 8-20 alphanumeric characters. A school name will be at most 50 alphabetical characters. Grade/year will be a range from 1-12. Date of birth will be a range from January 1st 1920 to present. All users MUST be registered to use the application.

Use Case Name	3.2.1.1 Registration
Goal	Allow an unregistered user to create an account
Primary Actor	User
Preconditions	The user must have a smartphone and the application downloaded.
Postconditions	The user will be enrolled in the system with the username and information provided.
MSS	1. User enters username, password, password confirmation, DoB, School, Grade/year 2. User presses submit/login 3. User is enrolled in database 4. User is logged in
Extension 1	1a. User incorrectly matches password and password

	confirmation. 1a1. User is alerted to error with password matching and asked to reconfirm. 1b. User uses invalid characters in one of the provided fields. 1b1. The user is alerted to the invalid entry and asked to reenter.
Extension 3	2a. User fails to fill in all fields 2a. User is alerted to missing fields and asked to fill them in. 2b. Username is taken 2b1. User is alerted that username is taken and asked to change to something else.

3.2.1.2 Functional Requirement 2 - Log In

The system shall allow users to log in using their registered credentials.

Use Case Name	3.2.1.2 Log In
Goal	Log a user into the application using their registered credentials.
Primary Actor	User
Preconditions	3.2.1.1
Postconditions	The user will be logged in to the application
MSS	1. User enters username 2. User enters password 3. User presses Log In 4. User is logged in to the application
Extension 1	3a. User enters an incorrect username or password 3a1. User is asked to reenter their username or password. 3a2. Start back at MSS 3.

3.2.1.3. Functional Requirement 3 - Add Course

The system shall allow users to add courses that include a course name, instructor name, meeting days, and course color. A course name will be 1-20 alphanumeric characters. An instructor name shall be 1-25 alphabetical characters. Meeting days will consist of only Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, and Saturday. Course colors shall be in the range of 10-30 different color variations. The addition of a course name that already exists within your planner will result in an error.

Use Case Name	3.2.1.3 Add Course
Goal	Allow user to add a new course to their planner.
Primary Actor	User
Preconditions	3.2.1.1, 3.2.1.2
Postconditions	The user has added the course to their planner.
MSS	1. User enters course name, instructor name, meeting days, and selected a course color 2. The add course button is pressed. 3. The users course is added to their planner.
Extension 1	1a. User uses invalid characters in the course name or instructors name 1a1. The system alerts the user there is incorrect information 1a2. The user fixes error
Extension 2	2a. The user already has a course with the same course name. 2a1. The user is prompted to change the course name 2a2. Begin at MSS 2.

3.2.1.4 Functional Requirement 4 - Delete Course

The system shall allow users the ability to delete a course that is in their planner. Deletion of a course will result in unsubscribing to the selected course and removal of all data from their planner related to that course.

Use Case Name	3.2.1.4 Delete Course

Goal	User deletes selected course
Primary Actor	User
Preconditions	User has navigated to course menu. 3.2.1.3, Or user has subscribed to another users course
Postconditions	The selected course is deleted/unsubscribed from the users planner.
MSS	1. User selects a course 2. User presses delete course button 3. System deletes course information from users planner and/or unsubscribes them from other users course.

3.2.1.5 Functional Requirement 5 - Edit Course

The system shall allow users to edit courses that have been added to their planner. This includes changing the instructor name, course name, meeting days, and course color. Editing shall adhere to validity checks mentioned in functional requirements 3.2.1.2. A user will only be able to edit a course that they have created.

Use Case Name	3.2.1.5 Edit Course
Goal	User edits course information
Primary Actor	User
Preconditions	3.2.1.3, and course being edited was created by the user.
Postconditions	Course information is successfully edited and saved.
MSS	1. User selects a course from the course menu. 2. User selects edit course from course menu. 3. System populates the course fields 4. User edits a field from the add course screen. 5. User presses add course 6. The edited course is saved and uploaded.
Extension 1	2a. The selected course from the course menu was not created by the user and can not be edited.

3.2.1.6 Functional Requirement 6 - Add Task

The system shall allow users to add tasks to the selected course. Task titles shall be in the range of 8-20 alphanumeric characters. Due dates shall be in the range of present day to any day in the future. A priority level shall be in the range of 0-3. A task percentage shall be in the range of 0-100%. Additional comments shall be alphanumeric characters in the range of 0-255. A user will only be able to add tasks to courses that they have created.

Use Case Name	3.2.1.6 Add Task
Goal	Add a task to the selected course
Primary Actor	User
Preconditions	3.2.1.3, user is the owner of selected course.
Postconditions	The task is added to the selected course and displayed in planner.
MSS	1. User navigates to Task Menu 2. User selects the course that they wish to add a task to. 3. User fills in task information. 4. Task is added to course and planner.
Extension 1	2a. User selects a course that they are not the creator of. 2a1. User is alerted that they cannot add tasks to courses they didn't create and are asked to select another course. 2a2. Begin at MSS 2
Extension 2	2a. User enters invalid characters into one of the fields.

3.2.1.7 Functional Requirement 7 - Edit Task

The system shall allow users to edit tasks that have been previously added. Editing tasks shall adhere to validity checks proposed in Functional Requirement 3.2.1.5. The system shall not allow users to edit tasks that are past due. A user will only be able to edit tasks for courses that they have created.

Use Case Name	3.2.1.7 Edit Task
Goal	User edits a previously added task
Primary Actors	User
Preconditions	3.2.1.6, Course that task exists in belongs to user
Postconditions	The edited task is saved and updated in planner.
MSS	1. User navigates to view tasks from calendar view 2. User selects 'E' on task they wish to edit. 3. System populates add task view 4. User edits information 5. User presses add task 6. Task is saved and planner is updated
Extension 1	2a. User selects a task they do not own 2a1. User is told they cannot edit tasks they do not own
Extension 2	4a. User enters invalid characters into one of the task fields 4a1 User is told to correct the invalid field.

3.2.1.8 Functional Requirement 8 - View Task

The system shall allow users to view a task by selection of the day from the home page by clicking on a date. If a date is selected that contains multiple tasks they will be vertically stacked on the view day screen.

Use Case Name	3.2.1.8 View Task
Goal	User views tasks for a specific date
Primary Actor	User
Preconditions	3.2.1.6 or 3.2.1.10

Postconditions	User is given a view of tasks for the selected date
MSS	1. User selects a date from the calendar view. 2. System provides a view of all tasks for the selected day if any exist.

3.2.1.9 Functional Requirement 9 - Delete Task

The system shall allow users to delete tasks that have been previously added to courses. A user will only be able to delete a task for a course that they created.

Use Case Name	3.2.1.9 Delete Task
Goal	Delete a task from the course
Primary Actor	User
Preconditions	3.2.1.6 or 3.2.1.10, and user is owner of selected course, task is in the future
Postconditions	User successfully deleted a task from their course.
MSS	1. User selects a date from the calendar view. 2. System populates task view 3. User selects “X” on the task they wish to delete 4. System deletes the task from their course and updates planner.
Extension 1	2a. No tasks to display for selected date. 2a1. User is shown an empty task view.
Extension 2	3a. User selects a task they do not own. 3a1. System alerts the user that the task cannot be deleted because they don’t own it.

3.2.1.10 Functional Requirement 10 - Complete a task

Use Case Name	3.2.1.10 Complete Task
Goal	Mark a task as completed
Primary Actor	User
Preconditions	3.2.1.8, task is in the future, user is the owner of the course that

	the task belongs to
Postconditions	User successfully marked a task as completed or finished
MSS	1. User selects a date from the calendar view. 2. System populates the task view. 3. User selects the green check on the task he/she wishes to mark as completed. 4. System will mark the specified task as completed and it is greyed out on the calendar view and task view.
Extension 1	2a. No task for the date. 2a1. User is shown an empty view.
Extension 2	3a. User selects a task they do not own. 3a1. System alerts the user that the task cannot be marked with completion because they don't own it.

3.2.1.11 Functional Requirement 11 - Subscribe to Course

The system shall allow users to subscribe to a course created by another user. This will populate their calendar with the selected courses information. Users subscribed to a course cannot alter or append any information to a subscribed course.

Use Case Name	3.2.1.11 Subscribe to Course
Goal	User successfully subscribes to another users course.
Primary Actor	User
Preconditions	User is not already subscribed to this course and 3.2.1.12
Postconditions	User will be subscribed to the new course and the planner will be updated.
MSS	1. User presses subscribe button on the course that they wish to subscribe to. 2. User is asked to confirm subscription 3. Subscription is confirmed and users calendar is updated

	with subscribed courses information.
Extension 1	2a. User already has a course with the subscribed courses name. 2a1. System asks user if they wish to overwrite the already existing courses information. 2a2. User confirms and data is overwritten or user cancels and subscription is aborted.

3.2.1.12 Functional Requirement 12 - Search for Courses

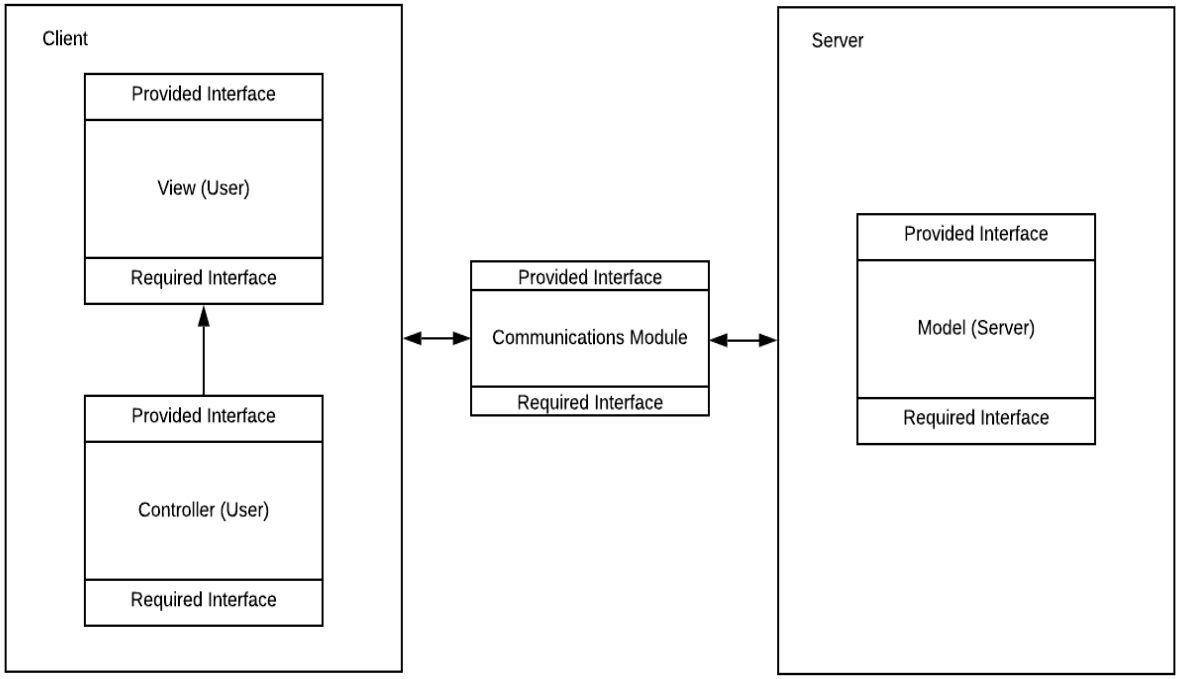
The system shall allow users to search for courses based on keywords. These will be alphanumeric and no longer than 8-20 characters.

Use Case Name	3.2.1.12 Search for Courses
Goal	User is able to search for courses that they specify by keyword.
Primary Actor	User
Preconditions	User has navigated to search function. Other users have added courses.
Postconditions	User will be presented a list of courses to subscribe to.
MSS	1. User navigates to search menu 2. User types in a keyword or phrase to be searched. 3. System populates the window with relevant results
Extension 1	2a. User enters invalid characters into search bar. 2a1. System will return no results.
Extension 2	3a. System finds no results matching search criteria 3a1. System reports that no search results were found.

Architecture

The “Milestone” application will make use of Model-View-Controller architecture. This suits our application well because of the three elements that are consistent with our needs. Our clients will be interacting with their view on the mobile devices which in turn will update the model or be updated by the model. The controller will provide the necessary ability to update the model and view based on the interactions with data components. The model provides us with the capabilities to store, access, and manipulate the necessary data for display and control.

Figure 1-1: High-Level Architecture View



Implementation

Implementation Technologies

The implementation of this project involved the use of the Java programming language, GraphQL Schema definitions, and XML. The framework involved in the implementation is the AWS Amplify Framework. The Amplify Framework uses the AWS DynamoDB to host the database in the cloud. DynamoDB can be used in a variety of configurations, however, for Milestone a GraphQL NoSQL type database was used. AWS Amplify also provides an authentication service via Cognito UserPools.

This was used to handle user login, authentication and sign-out through AWS AppSync. All of this was created and accessed inside Android Studio IDE.

Implementation Tasks

View

The implementation of the view component involved the creation of layout files in Android Studio. These are comprised of XML and Java and are the front-end aspect of the implementation. These house the various UI components that users interact with to alter the state of the model or send information to the controllers. They may also be updated by the model via GraphQL calls. This involved the implementation of the following XML and Java files:

- activity_add_course.xml
- activity_add_task.xml
- activity_authentication.xml
- activity_course_menu.xml
- activity_edit_course.xml
- activity_edit_task.xml
- activity_main.xml
- activity_profile.xml
- activity_subscriptions.xml
- activity_task_view.xml
- content_main.xml
- recyclerview_row.xml
- recyclerview_rowmain.xml
- Recyclerview_rowsubs.xml
- TaskViewActivity.java
- TaskViewAdapter.java
- UpcomingTaskAdapter.java
- CourseMenuActivity.java
- EditCourseActivity.java
- EditTaskActivity.java
- MainActivity.java
- ProfileActivity.java
- SubscriptionsActivity.java
- SubscriptionsAdapter.java
- AddCourseActivity.java

- AddTaskActivity.java
- AuthenticationActivity.java

Controller

The implementation of the controllers are Java files. These are tasked with creating, querying, or updating the model or updating the view with new information. These also hold information that is related to the state of the application. These establish connections to the model through GraphQL Callback methods. Some of the controllers lay within a designated controller class, while others reside in Activities where they have access to various functions of the underlying Android OS logic.The following classes were created to represent the controllers:

- ClientFactory.java
- CourseController.java
- UserDataController.java
- TaskController.java

Database/Model

The implementation of the database was done using AWS DynamoDB using GraphQL which is NoSQL. This represents the model and had schemas for course,task,and userdata. Within each of these were the necessary variables related to each data point that was required for the implementation to work. Because of AWS Amplify connecting to the database was simply done through callback methods under a standard structure provided to us by the framework. This drastically simplified querying,mutating, deleting, and updating information to/from the client.

Consistency

Consistency in terms of functionality was essentially at parity with the Software Specification Requirements. However, some controls and/or requirements were loosened in the interest of allowing users more freedom that previously was unnecessarily regulated.Searching worked but was not done by keyword, but by matching course name. The following changes to SRS have been made:

3.2.1.1 Functional Requirement 1 - Registration: The user now only needs a username, given name, email, password, and phone number. The school defaults to CSUSM, and the rest of the required information is optional and accessed from the Profile page.

3.2.1.3. Functional Requirement 3 - Add Course: The user is now allowed to enter whatever they would like as a course name, but a max character limit of 30 still exists. Repetitive course names are no longer

checked and instead courses are assigned ID's in the database.

3.2.1.6 Functional Requirement 6 - Add Task: Task titles have undergone the same change as above. Task Priorities are no longer 0-3, but None, Low, Medium, and High.

3.2.1.12 Functional Requirement 12 - Search for Courses: Since course names can be any length up to 30, searching for a course had to be altered as well. In addition, and unfortunately, courses are not searched by keyword but by a matching course name. I was unable to get AWS ElastiSearch to behave in the way I expected and I was nearing the allotted free time of its usage.

System Availability

GitHub: Milestone

New User Demonstration: Video #1

Existing User Demonstration: Video #2

Project Management

Group Member Name	Worked On
Jordan Mower	SRS, Architectural Document, Presentation #1, Presentation #2, Implementation, Final Presentation, Final Report
Jordan Nienaber	SRS, Presentation #1, Final Presentation
Jonathan Hua	SRS, Architectural Document, Presentation #1, Final Presentation

Nick Luckey	SRS, Presentation #1, Final Presentation

Working on this project presented a few problems but none that caused an incomplete or ineffectual product. My team didn't or couldn't help out in certain aspects and so rather than dragging my feet, hoping for equal effort I did what needed to be done to finish it.

Conclusion

In conclusion, this project was an incredible learning experience. There were many things that previously wouldn't have been considered when thinking about how a large software project comes together. After this project though, it is pretty clear that precise and detailed documentation is required for any substantial project to succeed through all the stages of software development.