Study Buddy

App Design Document

Ryan Adams
Caitlin Ard
Elena Davidson
Angela Kearns

December 3, 2017

Table of Contents

1. Introduction

1.1. Purpose

Study Buddy provides a space for University students to organize a meetup to study. After all, “Professor Q is absolutely killing me with these gravitational acceleration questions, and I have no one to work through this with.” To clarify the problem for our imaginary student is not that there are not resources at the university to help solve this problem. She could contact the TA, pay for a tutor, and the University has a big building with plenty of space to work the problem. What this student lacks is what Universities have traditionally provided that are now seemingly absent , a community for degree seeking students who are going through similar problems in similar classes. University message boards spontaneous meetups all used to dominate the college sphere. A similar student fifteen years ago might have said “ It's Thursday and the Physics 214 people always meet in the sub to do work through Professor Q’s problems maybe I can check that out.” Google and search engine contain a great deal of knowledge, however community and context is ultimately how the human machine is programmed.

1.2. Scope

The scope of this project is that our product will match the view of group based on the student’s selected campus. At this time, the scope of this project will not match based on location, just the student’s declared campus. The student will be able to additionally match with groups within their campus based on classes they are in. Groups can be created and joined by users for specific classes and users will have the ability to join existing groups. The user will also be able to look to see which other users/groups are in their classes by selecting drop downs that query the database.

1.3. Acronyms, Abbreviations, Definitions

SB - Study Buddy
UI - User Interface
FE - Front end {The Android App / What the Customer See}
DB - Database {SQLAlchemy}

1.4. References

Trello - This is the scrum like board we use to coordinate tasks among team members.
https://trello.com/

2. User Stories

2.1. User Logins

2.1.1. As a user i want to have a unique username and password that is saved

2.1.2. As a user i want to be able to use my own unique username and password to login.

2.1.3. As a user I want to make sure that I am signing up for a website that uses some validation that only people with a .edu address can sign up.

2.2. User Profile

2.2.1. As a user I want to be able to view my profile information

2.2. As a user I want to be able to see the study groups I have signed up for

2.3. As a user I want to be able to log out

2.3. User Boards

2.3.1. As a user I want to be able see what study groups are going happening

2.4. Create and Join Groups

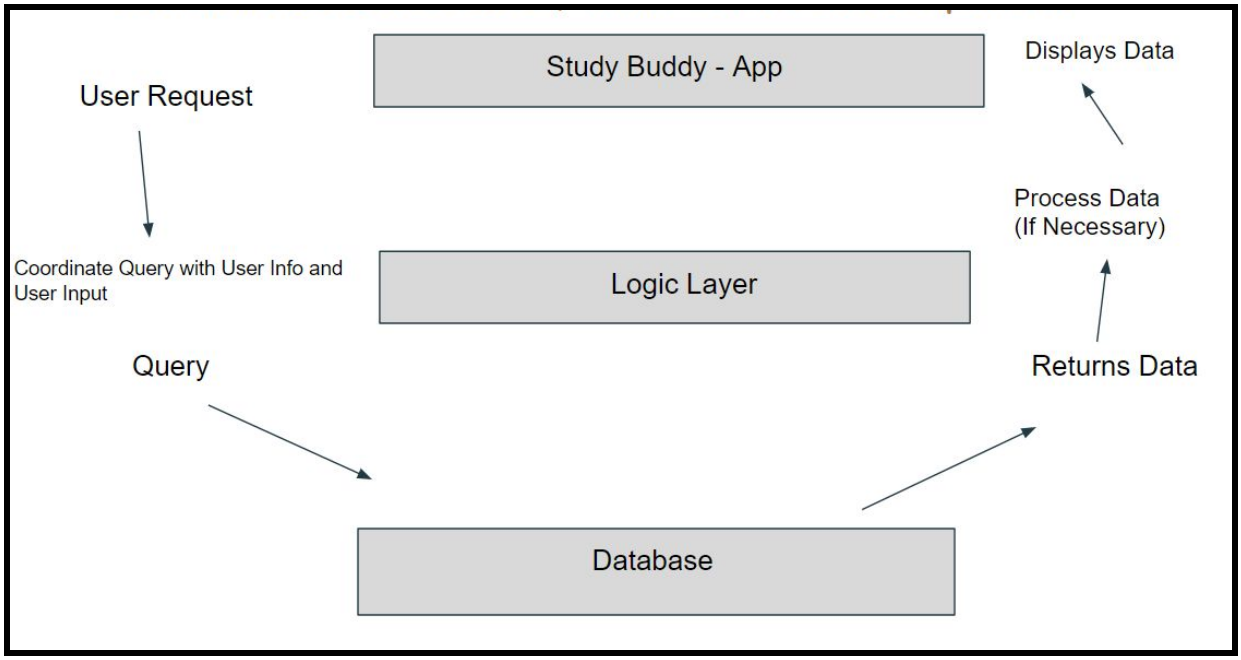
2.4.1. As a user I want to be able to create a group

2.4.2. I want to be made leader of the group

2.4.3. I want to see others who have joined my group.

3. System Architecture

3.1. Diagram

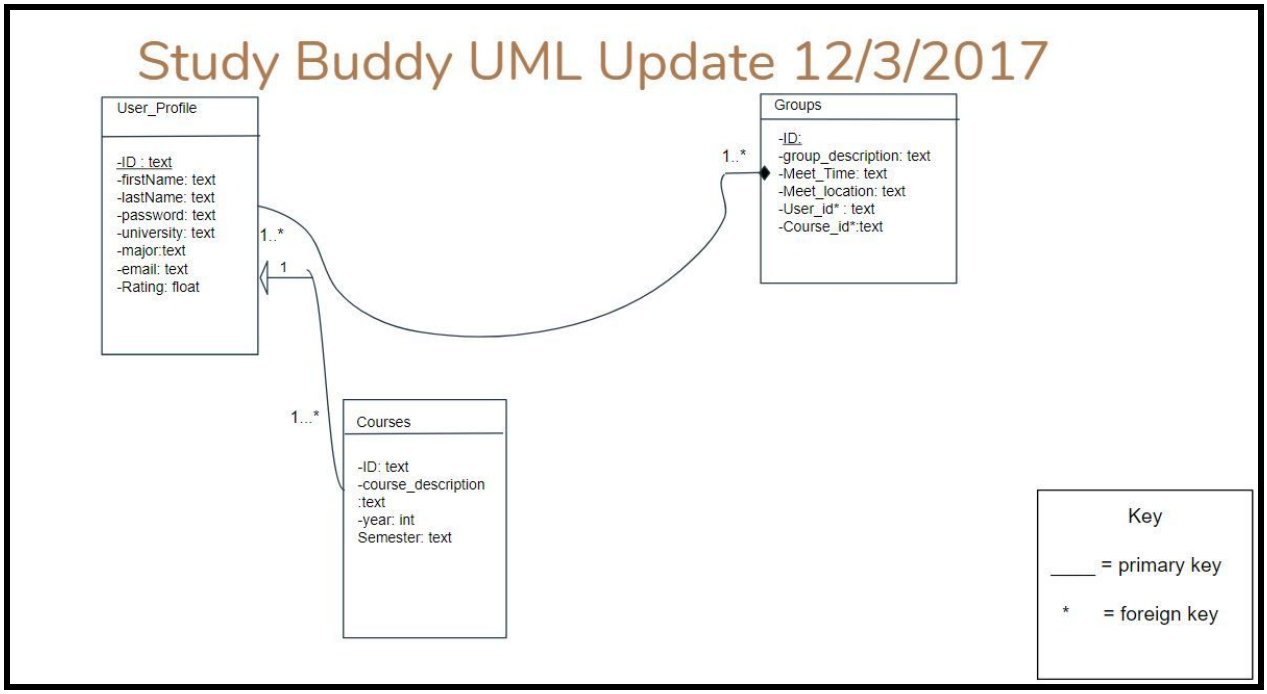


3.2. Explanation

We chose to build our app using a N-Tier (more specifically a 3-Tier) architecture design. The N-Tier architecture style was chosen because it allows the app to be broken up into three tiers: presentation/UI, logic, and database. The app will still have all the benefits of a typical Client-Server, including long terms data storage and allowing the user to interact with data in a controlled manner. However, the N-Tier architecture provides us with more control over the logic of the app than a typical Client-Server architecture. This extra tier for logic will help us better regulate the data that the users sees in order to create the most effective app experience. Update: We have pretty much have completed hosting the server for Study Buddy. Currently we are working on integrating each tier with each other. This could be a complicated process as we are still unsure how to exactly integrate the database, logic, and UI.

4. Class Level Design

4.1. UML Class Diagram



4.2. Class Description & Explanation

4.2.1. User should be able to login: and fill out required information with a 1:1 relationship with the User_Login

4.2.2. User_Profile will receive notifications from User_notification. Many notification as * to 1 user profile.

4.2.3. User_Ratings is a composition fo User_Profile as it extends User_profile by adding in the actual ratings. This is necessary because User_ratings will likely be a conglomeration of ratings, which may require more functions. Also this allows user ratings to be extensible for visualizations later in development.

4.2.4. Users will be able to add courses in the userprofile tab. But this information is inherited in to the course table. The implementation here is still under consideration: Relationship is 1 profile many courses.

4.2.4.1. Do users select from a drop down on courses?

4.2.4.2. Can they add a new course if they cannot find one?

4.2.5. User_Profile is used to aggregate a group. In fact multiple profiles are. Many to many relationship.

5. Requirement Specification

5.1. Functional Requirements

5.1.1. We want to have SQLite able to accept any inputs from the front end.

5.1.2. Generate app splash and login screens.

5.1.3. DB Table Users and front end & back end provides validation.

5.1.4. App provides user feedback to show errors or success.

5.1.5. User boards generated based on User Profile and Indicated Campus.

5.1.6. Object displayed as "cards" on board on front end of app.

5.1.7. Groups can be created by user.

5.1.8. Validation will be done on front/back end and then inserted into DB.

5.1.9. Groups are viewable my students in relevant campus.

5.1.10. Users can query DB from drop down lists to see users and groups in classes that user is in.

5.2. Nonfunctional Requirements

5.2.1. Our product can be viewed on devices and emulators able to run Nougat 7.0. Our emulator is a Nexus S running Nougat 7.0.

5.2.2. Our app will need to load pages at a reasonable speed (Less than 3 seconds).

5.2.3. Our app will need to run queries at a reasonable speed (Less than 10 seconds).

5.2.4. The UI of our app will be limited to the google design native to android for the sake of simplicity.

5.2.5. Users will expect to be given information at a reasonable speed (within 3 seconds or less of clicking a new tab).

5.2.6. Users will expect intuitive UI, which is why we are using the proven model from android.