MIAMI

CSC 431

Acacia

System Architecture Specification (SAS)

Team Number :

Chase Morell	Team Leader/Back End Developer
Christopher Duarte	App Designer/Developer
Kevin Padron	App Designer/Developer
Enrique Grijalva	Developer
Alison Cohen	Front End Designer/Developer
Emily Silvershein	Front End Designer
Ava Grossman	Project Manager

Version History

Version	Date	Author(s)	Change Comments
.5	3/23	Acacia Team	
1.0	3/28	Chase Morell	
2.0	3/31	Ava Grossman	description in various diagrams

Table of Contents

1. 11
1.1 12
1.2 12
1.3 12
1.4 12
1.4.1 13
1.4.2 13
1.4.3 13
2. 13
2.1 **Error! Bookmark not defined.**
3. 16
4. **Error! Bookmark not defined.**

Table of Tables

Table of Figures

System Diagram – Page 6

Sequence Diagram – Page 8

Structural Diagram – Page 9

51. System Analysis

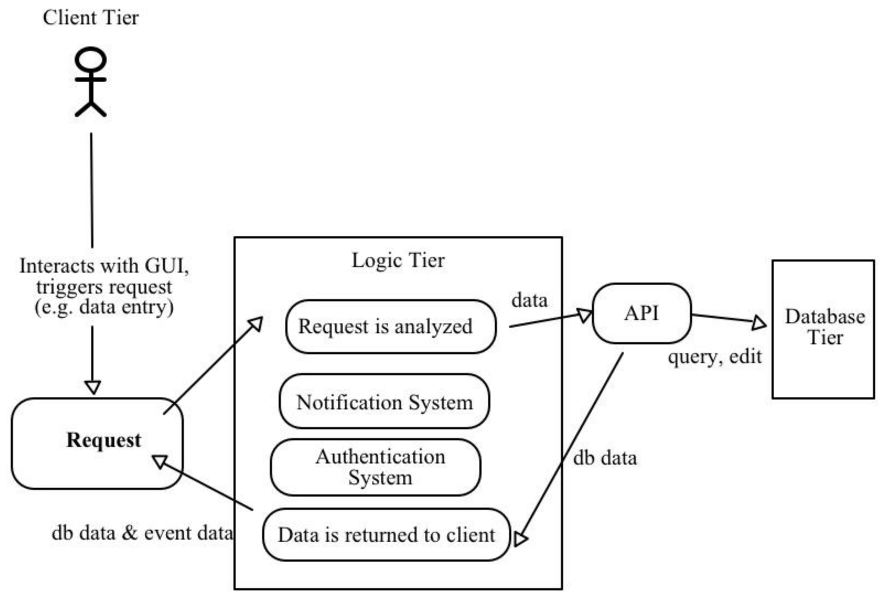
51.1 System Overview

The system will be compromised of 4 main parts: a mobile front-end client, a user authentication system, a database, and a notification system. The front-end GUI will connect to the authentication system, database, and notification system.

The three-tier architecture will be used for this project. The client tier will be the mobile app. This presentation tier will have a rich user interface with the Acacia brand and styling. The logic tier will run on Firebase, which allows for cloud functions. When the client tier submits a request, it is handled by the logic tier. The logic tier analyzes the request and interacts with the third tier which is the storage model to 1) retrieve data from the database or 2) make changes to the database. The logic tier passes the data received from the storage model back to the client tier. The client tier takes the information received and represents it in a graphical user interface.

51.2 System Diagram

Figure 1



51.3 Actor Identification

There are 2 types of human actors: unauthorized users and authorized users. Unauthorized users can download the app and can create an account but are not allowed past the login page of the app. To access any of the functionality of the app, a user must be authorized. To become authorized, a human user must create an Acacia account. Authorized users can access app functionality that queries the database, makes changes to the database, schedules notifications, as well as a number of client-side features, such as data visualization.

51.4 Design Rationale

1.4.1 Architectural Style

The Acacia project naturally fits a 3-tier architectural style. The first tier is the client tier. Human actors interact directly only with the client tier, which not only receives user input, but also presents data and information graphically based on user requests. The logic tier handles requests made by the client tier by communicating between the storage model tier and the client tier. The logic tier is located in the cloud on a server. Client tier requests are handled as events and analyzed. The logic tier will fetch data from the database located in the storage model. Information is sent back to the logic tier and then back to the client.

1.4.2 Design Pattern(s)

Flutter is a powerful Dart framework that provides a robust set of features. We predict that the following design patterns will be primarily used in the following ways:

- Façade: To provide a simpler, unified interface to the often complex underlying code of both the logic and database tiers.
- Singleton: To maintain a single, consistent reference, such as a database reference
- Adapter: To allow 2 components with incompatible interfaces to talk to each other without modifying the source code in either component (ex. Let the client tier talk to the notification system).

1.4.3 Framework

The main framework used is Flutter. Flutter is a cross-platform UI software development kit that allows for full application development. Flutter uses the Dart programming language for both front and back-end development. Another framework used is the Firebase API. The Firebase API provides many features for the application, chiefly: authentication of user accounts and providing the database.

Links:

https://flutter.dev/

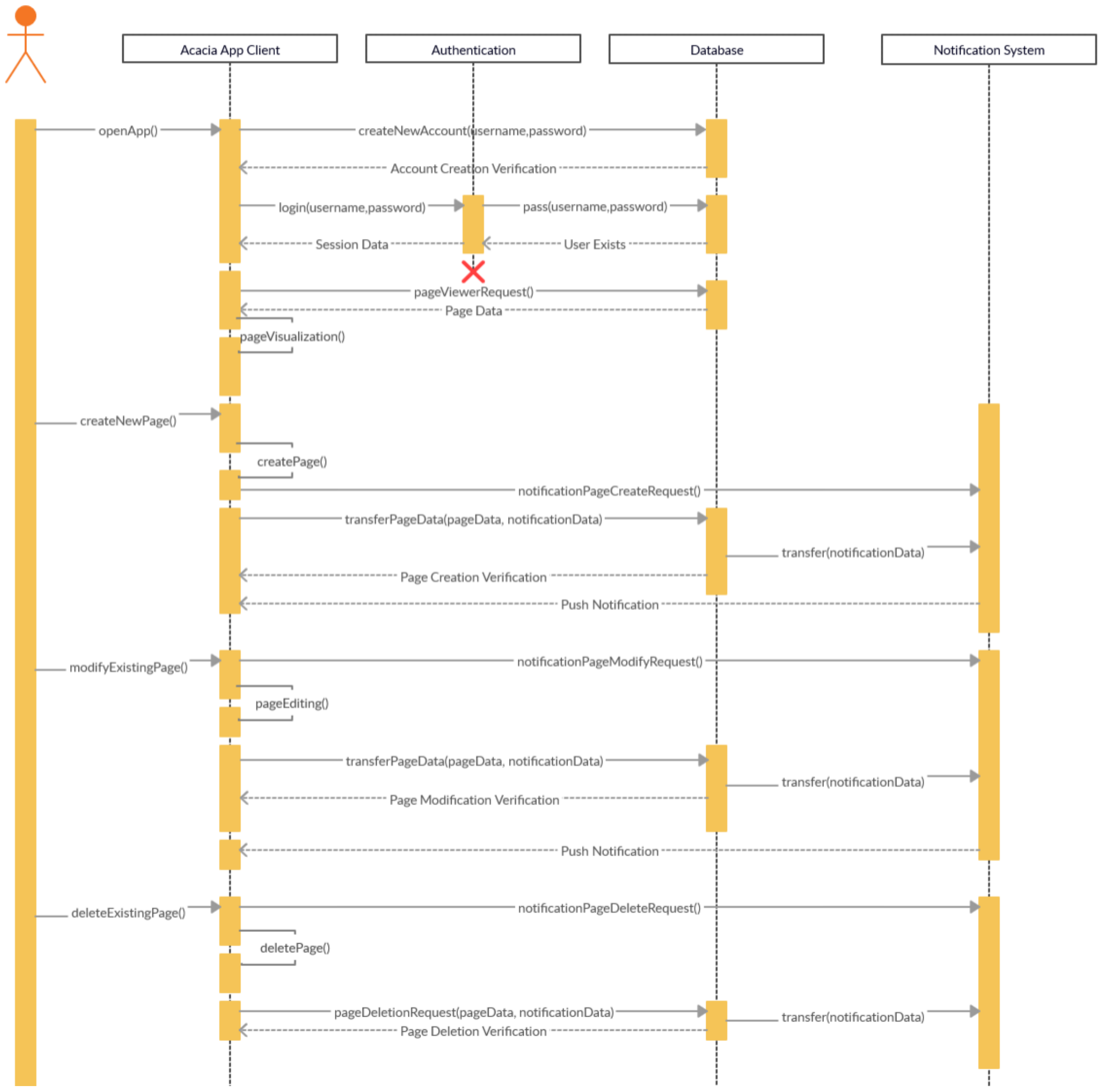
https://firebase.google.com

52. Functional Design

Here we present the sequence diagram for the Acacia app. This outlines the core functionality for this system.

2.1 Acacia App Sequence Diagram

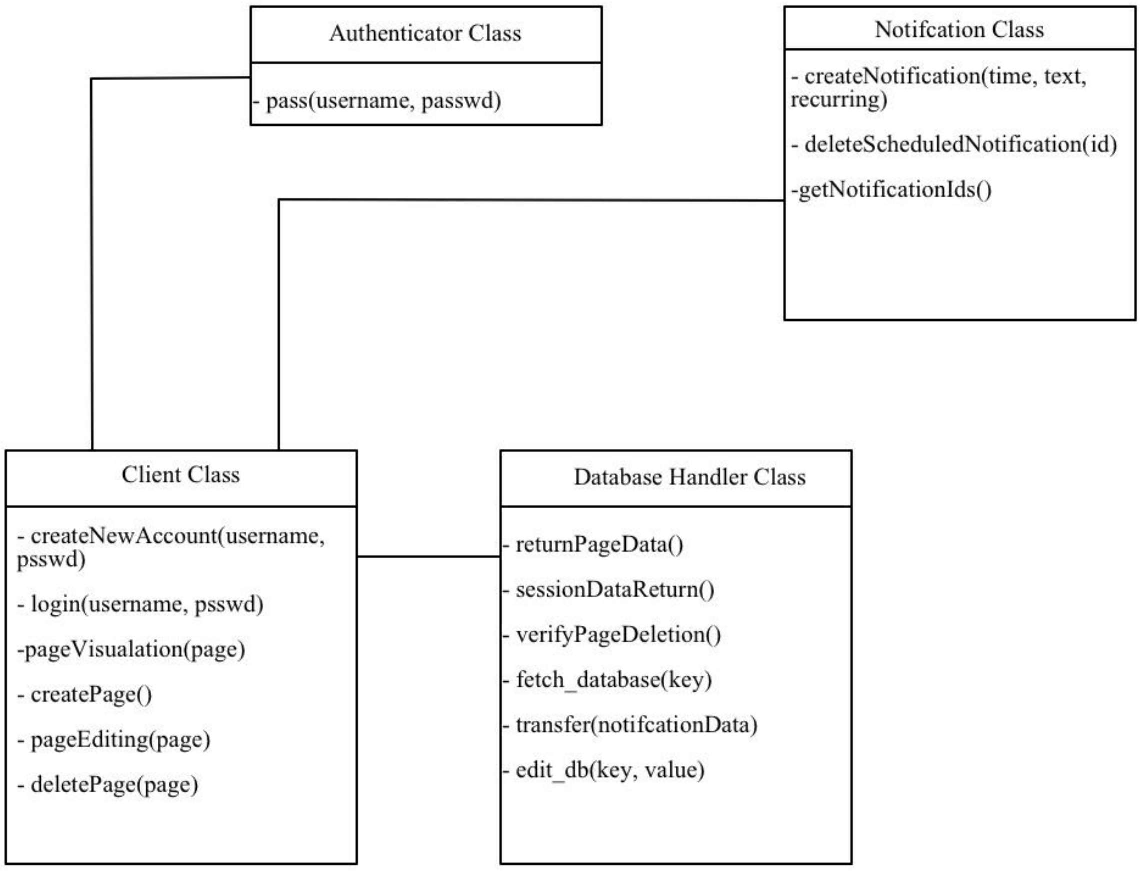
6A. Figure 2


Acacia App Client
Authentication
Database
Notification System
openApp()
createNewAccount(username,password)
Account Creation Verification
login(username,password)
pass(username,password)
Session Data
User Exists
pageViewerRequest()
Page Data
pageVisualization()
createNewPage()
createPage()
notificationPageCreateRequest()
transferPageData(pageData, notificationData)
transfer(notificationData)
Page Creation Verification
Push Notification
modifyExistingPage()
notificationPageModifyRequest()
pageEditing()
transferPageData(pageData, notificationData)
transfer(notificationData)
Page Modification Verification
Push Notification
deleteExistingPage()
notificationPageDeleteRequest()
deletePage()
pageDeletionRequest(pageData, notificationData)
transfer(notificationData)
Page Deletion Verification

61. Structural Design

The following class diagram represents an overview of the component breakdown in our system. A primary focus for this design is modularity.

Figure 3


Authenticator Class
- pass(username, passwd)
Notifcation Class
- createNotification(time, text, recurring)
- deleteScheduledNotification(id)
-getNotificationIds()
Client Class
- createNewAccount(username, psswd)
- login(username, psswd)
-pageVisualation(page)
- createPage()
- pageEditing(page)
- deletePage(page)
Database Handler Class
- returnPageData()
- sessionDataReturn()
- verifyPageDeletion()
- fetch_database(key)
- transfer(notifcationData)
- edit_db(key, value)