Teamerzer!

A Group Creation App

Design & Technical Document

Contents

Version History

Version	Date	Description	Author
1	11/16/2019	Initial Draft	John Pelliccia

1 Introduction

1.1 Purpose

The purpose of this document is to outline the technical and design of Teamerizer app and provide an overview of its implementation.

The Main purpose of our app is to:

- Provide information about the technical aspects of the application
- Provide a detailed account of the design of the application

1.2 Scope

To provide users with the design of our application based upon the requirements phases for each sprint.

1.3 Document Organization

This document is organized as follows

Introduction	This section provides information about the document
Design overview	This section describes architecture goals, principals and design patterns used in development
Topology Architecture	Describes our systems components and integration
Application Architecture	Describes our applications architecture and different screens and layers
Database Architecture	Describes the architecture of the database

2 Design Overview

2.1 Description of Problem

Traditional capstone project groups are created the first day of class when you know no one around you and are asked to find a group by the end of the class. What happens if you end up with students who may not be knowledgeable in your project? This could potentially ruin the whole project experience for

you. The Teamerizer app will provide students with a way to create groups where it's members have the necessary skills to complete the project.

2.2 Approach

Below describes the approach we used over the course of the project

- Requirements Phase – During the Requirements phase, the team created our application requirements and MVP goals.
- Application Design Phase – During the Application Design phase a hand drawn prototype and a digital design was created to establish a baseline for our system design,
- Development Phase – During the development phase we used the agile method to develop the application.

2.3 Architectural Goals

Our man architectural goal of the app is to provide users to pick group members who can contribute to a group project. Using firebase, the app will be highly available to users no matter the location.

The application can be used in the following ways

- Create a group and select the skills needed for that group.
- Review members who have the skills required to work on the project and invite them to the group
- View available groups as a member and request to be part of a group
- Manage group members in a group and remove if needed

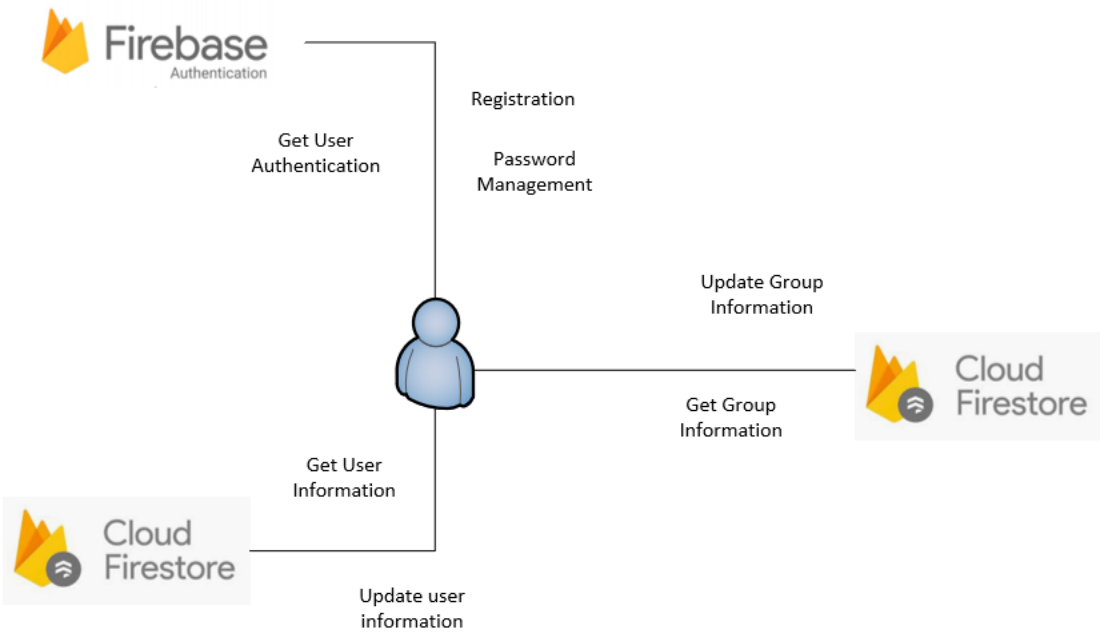
2.4 Design Principles

Best practices and design principles will be applied in two main areas

1) Application user interface
    a. Pages will need to be consistent for customer satisfaction and avoid confusion.
    b. This creates a familiar look and feel of the application
    c. Consistent ways of displaying user information to users
    d. Consistent method of user data entry
    e. CSS elements will be consistent across the application such as, backgrounds, layouts, fonts, spacing and ionic layout used
2) Database data handling
    a. Data will be collected from users and written to the database in a consistent manner
    b. Confidential data will be kept confidential using google firebase API
    c. Database reads will be carried out in a consistent manner for all data

3 Topology Diagram

This diagram displays our systems architecture in relation of the user interface and the database



Interactions Explained

Firebase Authentication – Manages user sign in and user registration through the firebase API, which handles encryption, storage and management of user credentials

Cloud Firestore – Cloud Firestore is Googles noSQL cloud database that stores information in application defined collections. Firestore will store out user information and group information as well as allow for searching on the data to find the right groups or group members.

User Personas (Audience)

4 Application Architecture

The application architecture of Teamerizer is critical to the use of the application as it is what bridges the gap between the database layer and the UI layer while providing users with the appropriate content.

Teamerizer has

Presentation Layer

Represents the user interface and how the primary user interacts with the application. The presentation layer is written in HTML and uses CSS for styling to bring a clean and user-friendly UI to the users. Angular events are used throughout the presentation layer as trigger events when user interact with button and submission of data. The presentation layer is written within ionic framework so that it is agnostic. Meaning it can be compiled into formats compatible with iOS, android or run on a web sever as a web page.

Content Layer

The content layer is driven from google Firestore. This layer presents users with their user information, group information, other member information and updates to that information

Database access Layer

The database layer is driven by our user server written in angular which handles all the user storage and retrievals from the database. This layer also includes a group service which serves the same purpose, storing and retrieval group information from the database. User authentication and update are also handled on this layer in a secure method.

Database Layer

At the database layer user authentication, user information and user groups are stored. User Authentication is encrypted by google so that even admins cannot see passwords, admins can how ever manage users including, deactivating, password resets and deletion. Stored user information will be used for our back-end algorithms for matching members to the correct groups based on their skill sets. Stored groups will store group information and members that are part of those groups, this will be presented to members on the main page.

5 Application Implementation

The application structure is included below

Application Structure	
Log in Page	Users can log into the application or sign up page
Sign-up page	New Users will need to fill out their email and create a password.
User Profile	All users must create a user profile after signup and list their skills
Main Page	The main page will present current groups and available groups to users
Group Creation	The group creation page will let users create a group and select the required skills
Matched Members	This page will display a list of matched members for the user to click on to review.

Group Details	When a group is selected from the main page the group details page presents details about the group and the current members
Member Details page	Description: This page presents details about the member

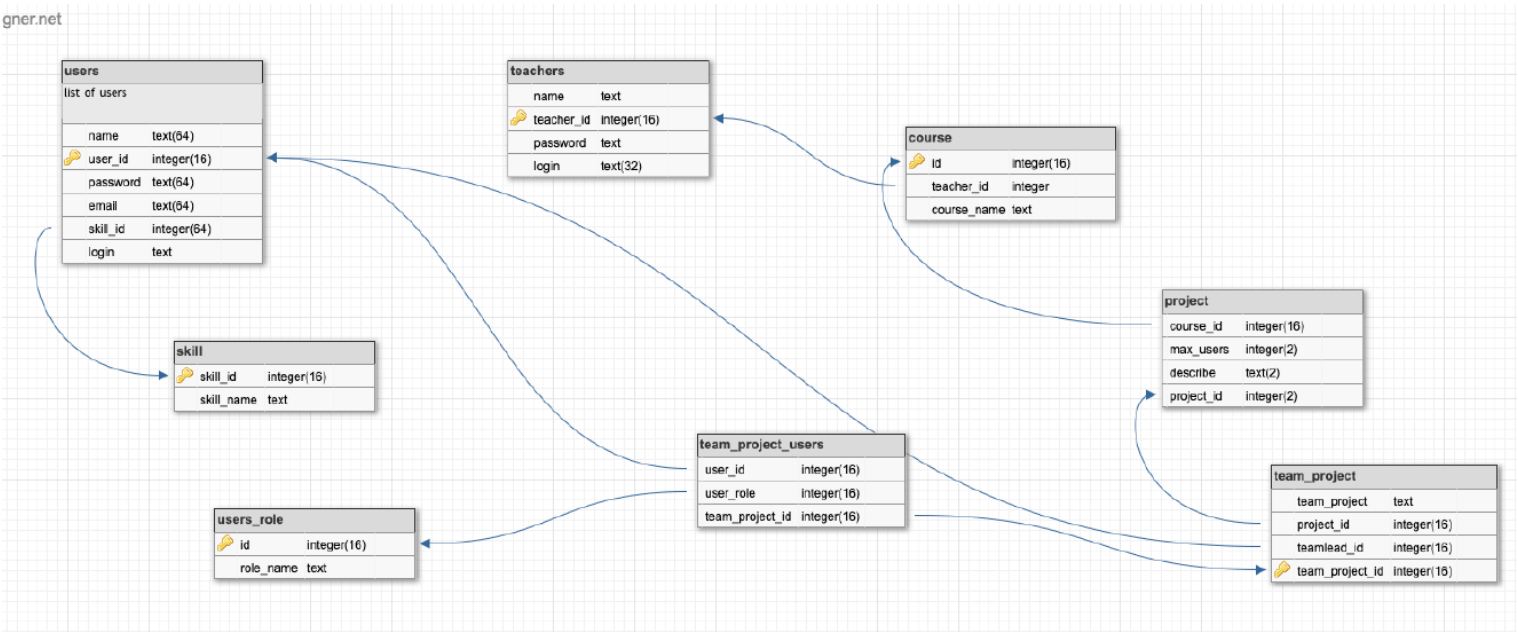
6 Database Architecture

Teamerizer utilizes google Firestore to store all out user and group data.

Information will be stored in a in collections in a noSQL database as a collection.

6.1 Data Model

Data in a noSQL database is stored as documents within those documents we have a collection of items. For example, users will be in a collection which will have a list of users. Below is the general scheme to our database. **Note** that this will be updated when we can generate a firebase schema once complete at the end of the semester.



6.2 Collections

Collections will mainly have users, groups, courses.

1. The Users collection will be used for storing all user information and have a reference id which will link it to googles user authentication. This collection will also store reference to the groups that a user is part of, if any. Other metadata included is, name, email, skill, skill level.
2. The Groups collection will be used for storing group information including the following: Name, description, max users, skills and a sub collection of current group members.
3. The Courses collection will have a list of courses. This collection will be used to link users and groups

7 Technologies Used

The Teamerizer app is built in Ionic, which is an “open source SDK for hybrid mobile app development”. It is well known for building high performance and progressive web apps that function well on any platform. It utilizes Angular scripting, HTML and CSS to model the pages and functionality of the application which is platform agnostic. Meaning it can be compiled to function with iOS, Android and Web based applications.

Teamerizer utilizes Google Firebase, specifically FireStore which is a highly available noSQL include cloud database. The database will be used for user Authentication and data storage through the application.