Module Code: CS2PJ20

Assignment report Title: Spring Android Studio App coursework

Date (when the work completed): 25/03/2024

Actual hrs spent for the assignment: 16

# Overview

1. A brief overview of TwentyOneDays

The TwentyOneDays is a habit-tracking application designed to confront the modern challenge of procrastination, exacerbated by social media distractions. Targeted at individuals aiming to enhance their productivity and consistency in habit cultivation, TwentyOneDays is built to integrate effortlessly into daily life, facilitating the tracking and monitoring of habit progress. The app's name, "TwentyOneDays," is inspired by the popular belief that it takes 21 days to form a new habit, symbolizing the app's commitment to helping users embark on a journey of personal improvement and transformative change.

The name "**TwentyOneDays**" was meticulously chosen to reflect the app's essence and mission. It is based on the concept that consistent effort over a period of 21 days can significantly contribute to the formation of lasting habits. This name highlights the transformative journey users undertake with the app, emphasizing progress, commitment, and the potential for change.

1. Objectives

The development of TwentyOneDays centres around three key objectives:

**User-Friendly**: ensuring the app is accessible and straightforward, with a focus on a simple yet effective interface to encourage regular use.

**Adaptable**: catering to a wide array of habits and goals by supporting multiple input methods, allowing users to personalize their habit-tracking experience.

**Motivational**: utilizing visual indicators of progress and motivational elements to keep users engaged and motivated to maintain their habit streaks.

# Application Specifications

1. Describing of TwentyOneDays technical specifications:

TwentyOneDays features a modular application structure, with each screen represented as an activity that includes an XML layout and a Java class. These components work together to facilitate smooth user interactions and backend processes. Highlights include:

**Welcome Screen**: designed with interactive elements to guide users through the app’s features.

**User Authentication**: secured with Firestore for robust data management, employing dedicated classes for managing user interactions;

**Functionality**: tailored functions for showcasing recent activity, setting and tracking milestones, and exploring the potential for social connectivity to enhance the user experience.

1. User Interface Design

The app embraces a visually soothing theme, using colors and designs that evoke tranquility and trust, making users feel comfortable and secure as they log their activities and track their progress.

I will try to use rounded edges and a more modern font family as rounded edges can improve usability by reducing the risk of accidental taps or swipes on sharp corners, especially on mobile devices where precision can be challenging. This can lead to a smoother and more frustration-free user experience, encouraging users to interact with the app more frequently. Modern font families are often designed with legibility and readability in mind, making them suitable choices for digital interfaces. These fonts typically feature clean and well-defined letterforms, which can improve readability on various screen sizes and resolutions.

To be a successful application, I would have to have a list of features that need to be implemented:

Basic authentication where a user can enter a name email, username and password and after registering, are able to login with that information.

A Habit system, a user is able to choose a specific habit that they want to record and the settings around when they want to be reminded about it.

There are more aspirational features that I would want to implement that would extend upon the features mentioned prior:

a Social system that allows users to search to see if their friends are using the application and then add them, this will allow them to see their friends and if they are keeping on top of their habits.

# Application Implementation

1. Basic functionality 1 – User Authentication

The Login activity is responsible for authenticating users using email and password credentials. Here's a summary of its main functions and how it provides user authentication:

**Layout Setup:**

 In the onCreate() method, the activity layout is set using setContentView(). References to the email input field (editEmail), password input field (editPassword), login button (loginBtn), and sign-up redirect text (signUpRedirectTxt) are initialized.

**Firebase Authentication Initialization:**

An instance of FirebaseAuth (mAuth) is obtained using FirebaseAuth.getInstance().

**Sign-In Functionality:**

The signIn() method is called when the login button (loginBtn) is clicked.It retrieves the email and password entered by the user from the input field It uses the signInWithEmailAndPassword() method of mAuth to attempt to sign in the user with the provided credentials.A listener is added to handle the asynchronous result of the sign-in attempt.If the sign-in attempt is successful (task.isSuccessful()), the user is redirected to the main application activity (Drawer), and the user's unique ID (UID) is passed along with the intent.

If the sign-in attempt fails, an appropriate error message is displayed to the user via a toast.

**Sign-Up Redirection:**

The signUpRedirectTxt TextView is clickable and triggers an intent to navigate to the sign-up activity (SignUp) when clicked.

Overall, the Login activity integrates with Firebase Authentication to securely authenticate users, and upon successful authentication, it directs them to the main application interface. If authentication fails, appropriate error messages are displayed to the user.

The Sign-Up activity is responsible for registering new users by collecting their email, password, and username credentials. Below is a breakdown of its main functions and how it facilitates the user registration process:

**Layout Setup**:

In the onCreate() method, the activity layout is set using setContentView().

References to the username input field (editUsername), email input field (editEmail), password input field (editPassword), sign-up button (signupButton), and login redirect text (loginRedirectText) are initialized.

**Firebase Authentication Initialization:**

An instance of FirebaseAuth (mAuth) is obtained using FirebaseAuth.getInstance().

**Sign-Up Functionality:**

The onClick listener is attached to the sign-up button (signupButton). When the button is clicked, the input fields for email, password, and username are retrieved. Input field validation is performed to ensure all required fields are filled. The createUserWithEmailAndPassword() method of mAuth is used to attempt to create a new user account with the provided email and password.

A listener is added to handle the asynchronous result of the sign-up attempt, if the sign-up attempt is successful (task.isSuccessful()), the user's data, including their username, is stored in the Firebase Realtime Database under the "users" collection. A success message is displayed to the user, and they are redirected to the login activity (Login.class) to sign in with their newly created account. If the sign-up attempt fails, an error message containing details of the failure is displayed to the user via a toast.

**Login Redirection:**

The login redirect text (loginRedirectText) is clickable and triggers an intent to navigate back to the login activity (Login.class) when clicked.

Logout Button:

Although not explicitly mentioned in the code provided, it's common in authentication flows to include a logout button. This button would log out the current user from their account and bring them back to the start screen or login page, depending on the application's design.

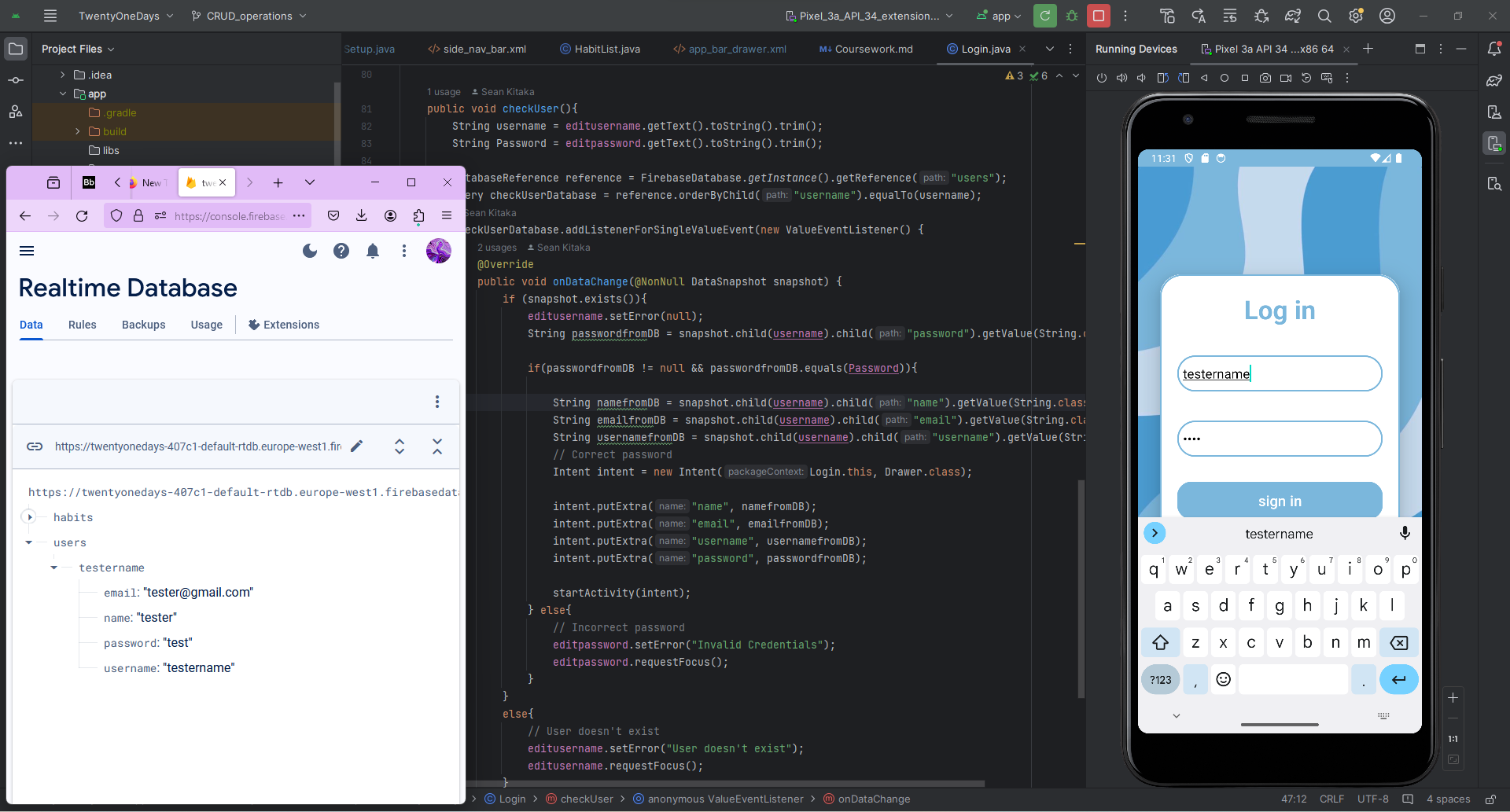


Figure 1 – Screenshot displaying login page and Firebase Authentication

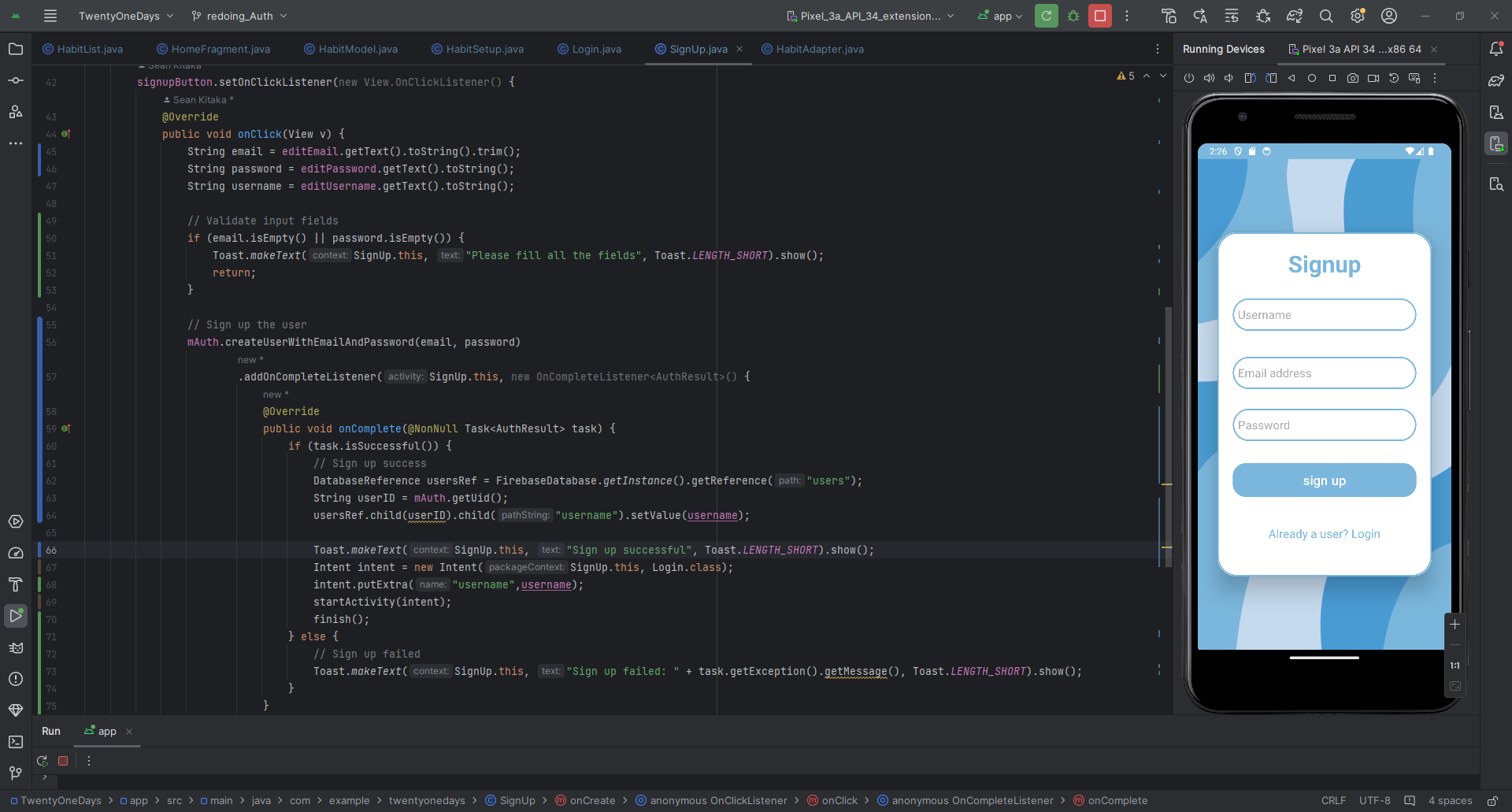


Figure 2 Screenshot displaying signup page and Firebase Authentication

1. Basic Functionality 2 – Habit Tracker

This feature isn’t simple as it requires multiple parts in order to work properly and is the prerequisite to other One of the key components of the Habit Tracker is the implementation of a RecyclerView. This decision was made to optimize memory usage and enhance performance, particularly when dealing with large datasets. Unlike traditional ListView, RecyclerView only renders the items that are currently visible on the screen, leading to improved efficiency.

To implement the RecyclerView efficiently, several classes were developed:

(Class) HabitModel: This class encapsulates the essential data fields related to a habit, including its name, type, frequency, and other pertinent details. By utilizing a dedicated model class, we ensure a structured approach to handling habit data, enhancing code organization and readability.

(Class) HabitAdapter: The HabitAdapter class is responsible for binding the HabitModel objects to the RecyclerView and inflating the corresponding view. This adapter facilitates the seamless display of habit data within the RecyclerView, enabling users to interact with their habits effortlessly.

(Class) HomeFragment: As part of the application's user interface, the HomeFragment plays a crucial role in collecting data from Firebase and passing it to the HabitAdapter for display within the RecyclerView. Additionally, the HomeFragment incorporates an OnClickListener for each habit item, enabling users to transition to detailed views for individual habits.

(Interface) RecyclerInterface: This interface defines the onItemClick function, allowing for customized actions to be executed for each item in the RecyclerView. By implementing this interface, we ensure flexibility and modularity in handling user interactions within the Habit Tracker.

1. Additional feature 1 – reviewing habit

After a habit is habit has been made, I want to be able to have the user add any media to mark progress along their journey of building habits, so I needed the habits to be editable, and I wanted the user to be able to see the accumulation of all of the times that they were recording their habit,

Additional Feature 2 - Friends

The app has made significant strides in enhancing user engagement and facilitating personal growth.

Since the Users are within a collection on firebase, in order to implement a friend system the users will need to:

Search for a user name,

if they find the user,

then the capability to send a friend requestion,

see if someone has requested to be their friend and if they have,

be able to accept or decline the friend request**.**

# Conclusions and Future work

1. Concluding remarks (Summary of the TwentyOneDays outcomes).

TwentyOneDays sets out to tackle procrastination by empowering users to form and sustain new habits. Achieving its objectives of being user-friendly, adaptable, and motivational, the app has made significant strides in enhancing user engagement and facilitating personal growth.

1. Reflection on overall learning experience and achievements.

The journey of developing TwentyOneDays has been profoundly educational, offering insights into the complexities of app development, user experience design, and the psychology of habit formation. The project's challenges have fostered a spirit of creativity and resilience, culminating in the successful realization of the app's core features and the acquisition of positive initial feedback.

1. Future Work

The roadmap for TwentyOneDays includes exciting opportunities for expansion and enhancement:

**Enhanced Social Features**: Building on the app's social aspects to encourage community engagement and mutual support among users.

**Gamification Strategies**: Incorporating elements of gamification to reward progress and celebrate milestones, boosting user motivation.

**Personalized Insights**: Integrating AI to provide customized advice and insights, making the habit formation process even more tailored and effective.

**Platform Diversification**: Expanding the app's availability to web and desktop platforms, increasing its accessibility to a wider audience.

**Wellness Integration**: Adding features focused on mental and emotional well-being, such as stress management and mindfulness practices, to support users' overall health.

In conclusion, while TwentyOneDays has successfully met its initial goals, its journey towards facilitating meaningful habit change and personal development is just beginning. Future enhancements, guided by user feedback and technological advancements, will ensure that TwentyOneDays continues to be a pivotal tool for individuals seeking to make lasting changes in their lives.