

[Description](#)

[Intended User](#)

[Features](#)

[User Interface Mocks](#)

[Screen 1](#)

[Screen 2](#)

[Key Considerations](#)

[How will your app handle data persistence?](#)

[Describe any corner cases in the UX.](#)

[Describe any libraries you'll be using and share your reasoning for including them.](#)

[Describe how you will implement Google Play Services.](#)

[Next Steps: Required Tasks](#)

GitHub Username: [RahulHP](#)

Daily Journal

Description

This is a daily journal app which tries to make journalling easier by limiting each entry to a length of 100 characters since this forces the user to use keywords and short phrases to summarise his day. It also adds a social element by adding the option to share the user's entries with friends.

This app is based on a concept found in a Medium blog post by Sean Yang - <https://medium.com/@xsvfat/daily-journal-app-design-case-study-2b0826d10fa2>

Intended User

This app is intended for any users who want a quick way to summarise their day and possible share it with a group of close friends.

Features

- Saves text only journal entries with a limit of 100 words
- Allows user to share specific entries with selected friends
- Syncs all journal entries to Firebase Real Time Database

User Interface Mocks

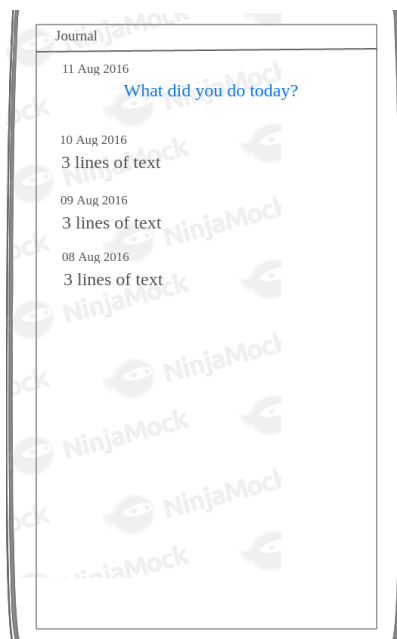
A preview of the app can be found on this link - <http://www.ninjamock.com/s/K93ND>

Sign in Screen



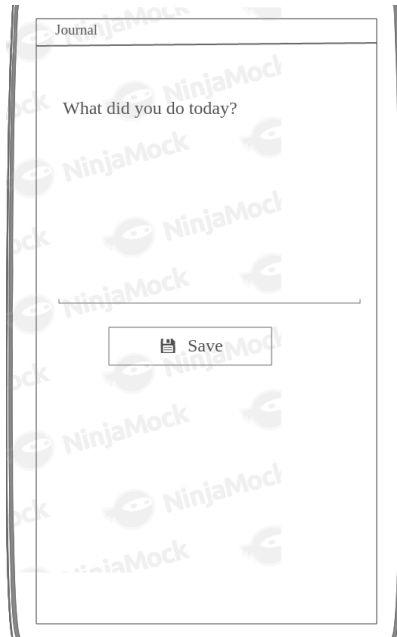
This screen will use Firebase Authentication to allow the user to sign in with his Google account.

List Entries



This homepage will show the entries for the past month and the “What did you do today” button will lead the user to create a new entry for the current day.

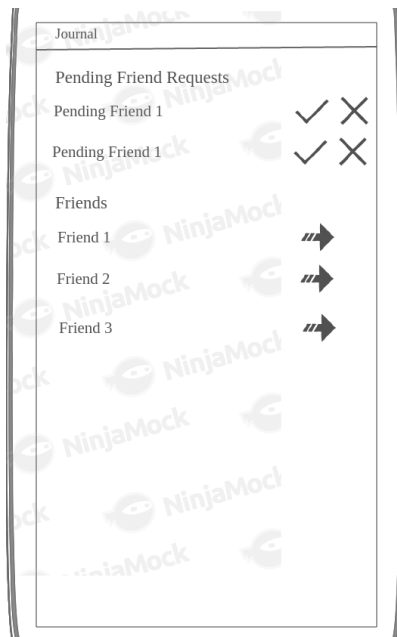
New Entry Screen



A mobile app mockup for a 'Journal' screen. At the top, the title 'Journal' is in a light blue bar. Below it, the text 'What did you do today?' is followed by a large, empty text input field. At the bottom, there is a 'Save' button with a document icon.

This screen will allow the user to write a new post for the current day and limit it to 100 characters. Clicking the save button will save the post to the Database.

View Friends



A mobile app mockup for a 'Journal' screen showing friend management. The title 'Journal' is in a light blue bar. Below it, the section 'Pending Friend Requests' is followed by two rows, each with 'Pending Friend 1' and a checkmark and an 'X' icon. Below this, the section 'Friends' is followed by three rows, each with 'Friend 1', 'Friend 2', and 'Friend 3' respectively, and a right-pointing arrow icon.

This screen shows the user's pending friend requests and current friends. Clicking on any of his accepted friends will lead to a screen showing his friends' posts.

View friends' posts

A mobile app mockup for the 'View friends' posts screen. The screen is titled 'Journal' at the top. Below the title, there is a section for 'Friend 2'. Under this section, there are two 'Single-Line List' items. The background of the screen is covered with a repeating 'NinjaMock' watermark.

This screen displays the shared entries of the user's friend.

Add friends

A mobile app mockup for the 'Add friends' screen. The screen is titled 'Journal' at the top. Below the title, there is a section for 'Add Friend'. Under this section, there is a 'Search' input field with a magnifying glass icon. Below the search field, there are two rows of text: 'Friend Name' and 'Friend ID' on the first row, and 'Friend name' and 'Friend ID' on the second row. Each row has a small person icon with a plus sign next to it. The background of the screen is covered with a repeating 'NinjaMock' watermark.

The user searches for his friends and sends a friend request to the person.

Settings

Journal	
Profile Information	
Name	XYZ
User ID	@IamXYZ
Notifications	
Enabled?	<input checked="" type="checkbox"/> ON
Notification Time (Daily)	9:00 PM

In this screen, the user can set his daily notifications to come at a specified time if he wants to.

Key Considerations

How will your app handle data persistence?

To store the journal entries, I'll use the Firebase real time storage. I'll have to decide the exact structure of my data first.

To store the friend list, I'll be using a content provider (to satisfy the Content Provider requirement)

Describe any corner cases in the UX.

I couldn't think of any corner cases in the UX as of now. I'll update this doc as I progress

Describe any libraries you'll be using and share your reasoning for including them.

I'll be using:

- Glide - To show the user's profile picture
- Firebase - For the complete backend of my app
 - Authentication - To enable Google Sign in

- Realtime Database - To store user entries
- AdMob - To display ads
- Google Play Services - Related to Firebase services above

Describe how you will implement Google Play Services.

- Authentication - To enable Google Sign in
- Realtime Database - To store user entries
- AdMob - To display ads

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and decompose them into tangible technical tasks that you can complete incrementally until you have a finished app.

Task 1: Project Setup

- Install required firebase libraries
- Glide
- Google play services

Task 2: New Entry Page

- Add textbox and save button
- Implement a character limit on the text box
- Wire things up so that the save button pushes the entry to the database
- Add an admob ad to the bottom of the page

Task 3: List Entries Page

- Add a FirebaseListAdapter to get entries for the current month from the database
- Add a box at the top to lead the user to the 'New Entry' page

Task 4: Sign In Page

- Follow the steps in the [firebase codelab](#) to enable users to sign in
- Add basic information about the app

Task 5: Search+Add friends Page

- Take input from user via search bar
- Use the input to query the database to get a list of possible results. Use an Async task for this if possible (To satisfy rubric requirements)g
- Send friend request to user

Task 5: Friend request method (may change in future)

The method is a very round about way but I found it the best way to incorporate Content Providers in my application.

- When User A sends friend request to User B:
 - Add User A to database/UserB/pending_requests
- When User B opens app on his phone:
 - Check pending requests by querying above path
 - If accepted:
 - Add user A to user B's content provider
 - Add user B to database/UserA/accepted_requets
 - Remove User A from database/UserB/pending_requests
- When user A opens app:
 - Check accepted requests from database/UserA/accepted_requests
 - Add user B to User A's content provider
 - Remove user B from database/UserA/accepted_requests

Task 5: View friend's Entries

I'll have to work on the database representation of this in detail. I'll subset the friends' posts and only display the public shared posts in a listView

Task 5: Settings

I'll need to implement notifications at the given time.

Task 6: Widget

This will be a small 1x1 widget which will take the user to the 'New Entry' page on clicking it.