Final Report, “TaskIT” Personal Task Manager App

R. Brian Redd, April 2, 2018

Honors Assignment 2; Multiplatform Mobile App Development with Web Technologies, Coursera; Instructor: Jogesh K. Muppala

Introduction

“TaskIt, the Personal Task Manager App I am creating in Ionic is a just that: a simple personal task manager. This app will allow users to add, review, edit, and complete tasks.

The initial plan was for tasks to will be stored in an online JSON data store, with each user’s tasks protected via login; however, for now the user data and tasks will be stored in the app’s local storage (services are employed so once the online data source is available, transitioning will be simple).

The UI will take advantage of the Ionic Framework, including Ionicons, Ion-Lists, Modals, Toasts, and gestures.

Design and Implementation

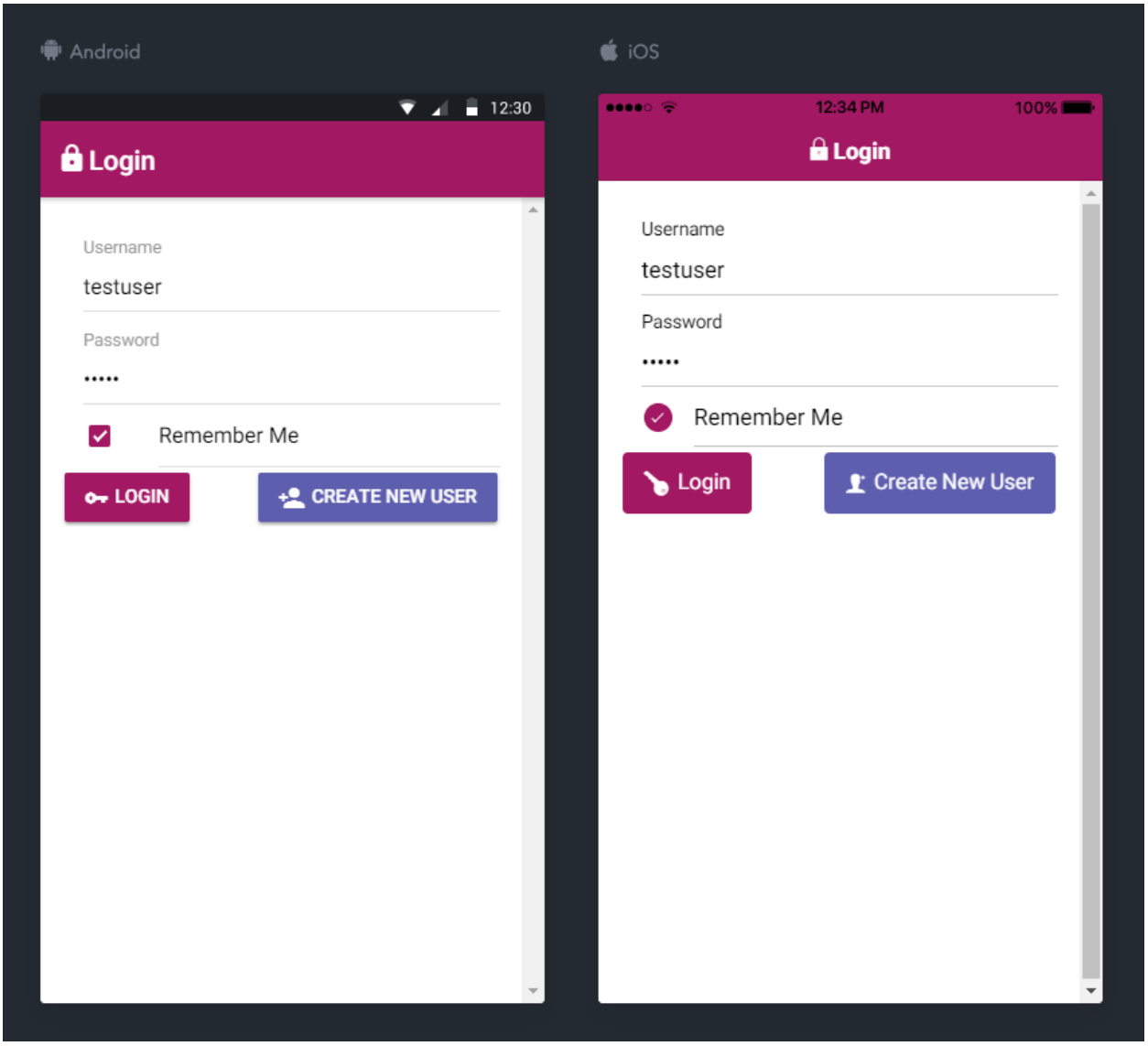
General

After my initial build and design, I decided to redesign it. The original “Tabs” template was interesting, but ultimately I decided to move to a left-rail menu instead. There are currently six primary navigation pages: Home, Tasks, New Task, Completed Task, User, and About, plus three additional pages: Login, New User, and Task Detail. Still in development are two additional pages: Edit Task and Task Display Settings (including Sort).

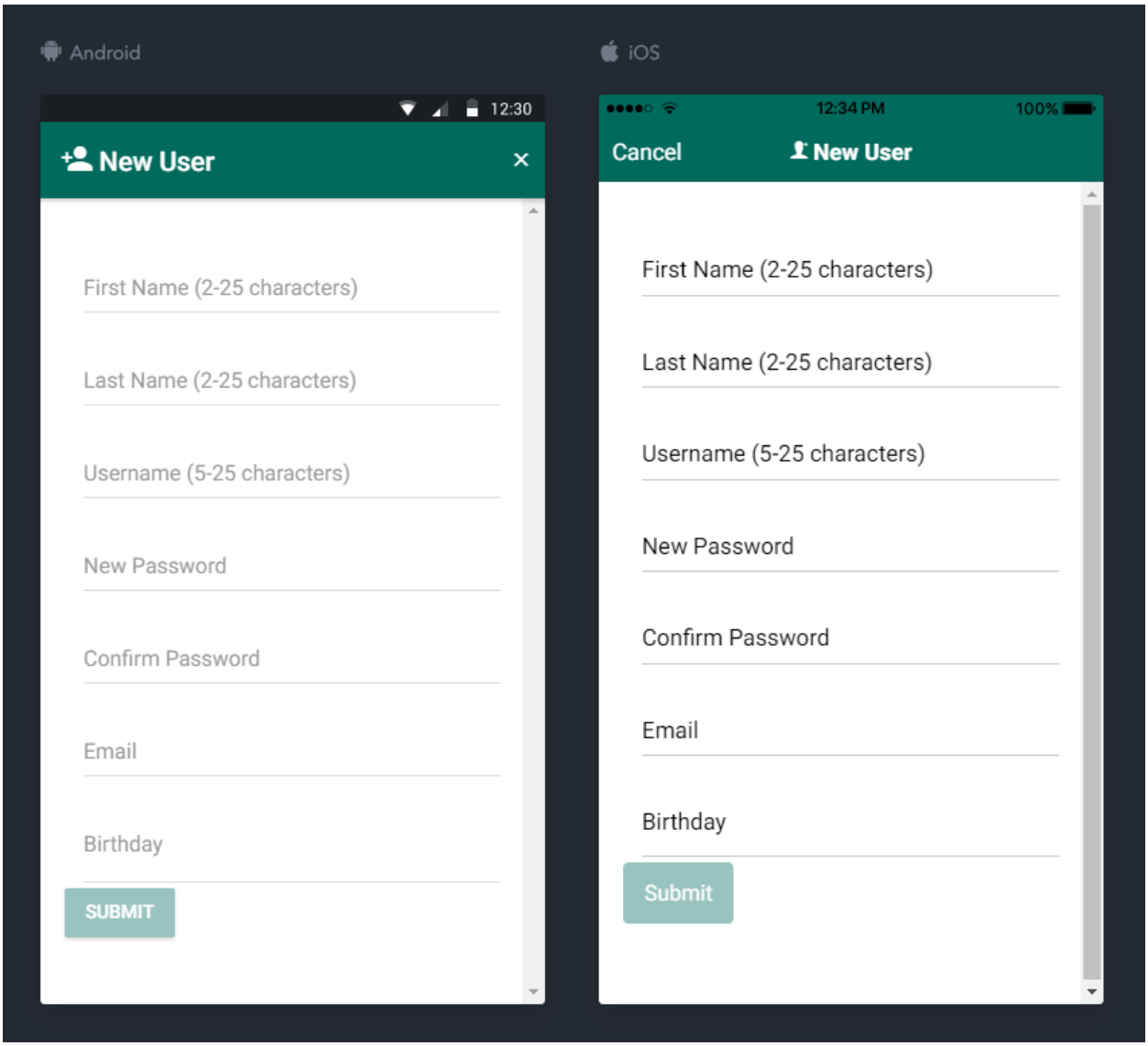
Initial Application Flow and Navigation

While eventually user details will be stored on a server, so users can login from any device with the same app, for now all information is stored locally. However, a user is still required.

When a user opens the app, the first page they’ll see is the Login Page:



If this is a user’s first time to the app (if no stored users are on the device), they will immediately be taken to “Create New User”.



Here, the user enters their information. The only field that is not required is “Birthday”.

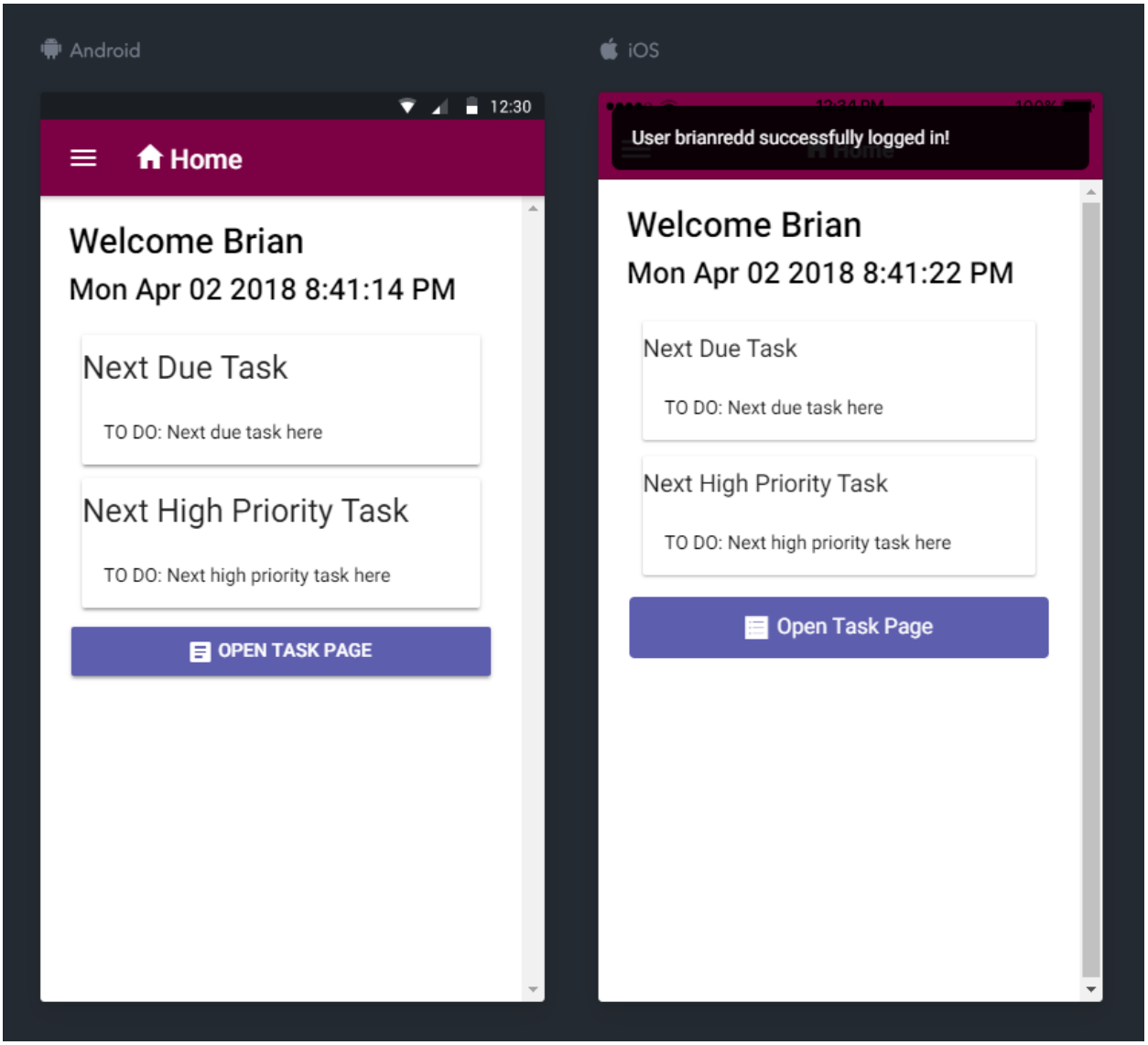
When the user Submits, validation takes place, including verifying that Password and Confirm fields match, and the Username is unique among saved users (locally initially, but this will also verify against a server once the server is in place).

The user will also be given a user ID (“0” for the first user, and incrementing from there). This ID will be used to separate out user tasks. Only the user’s tasks will be downloaded to the device (or recalled from local storage); this will not pull all tasks and then just filter out those belonging to the users.

Even when a server is in place, data will still be primarily stored locally, and will only update to the server as needed (initial load, changes to the data, etc.) There will also be indications of online status, with the user option to go offline even if online is available.

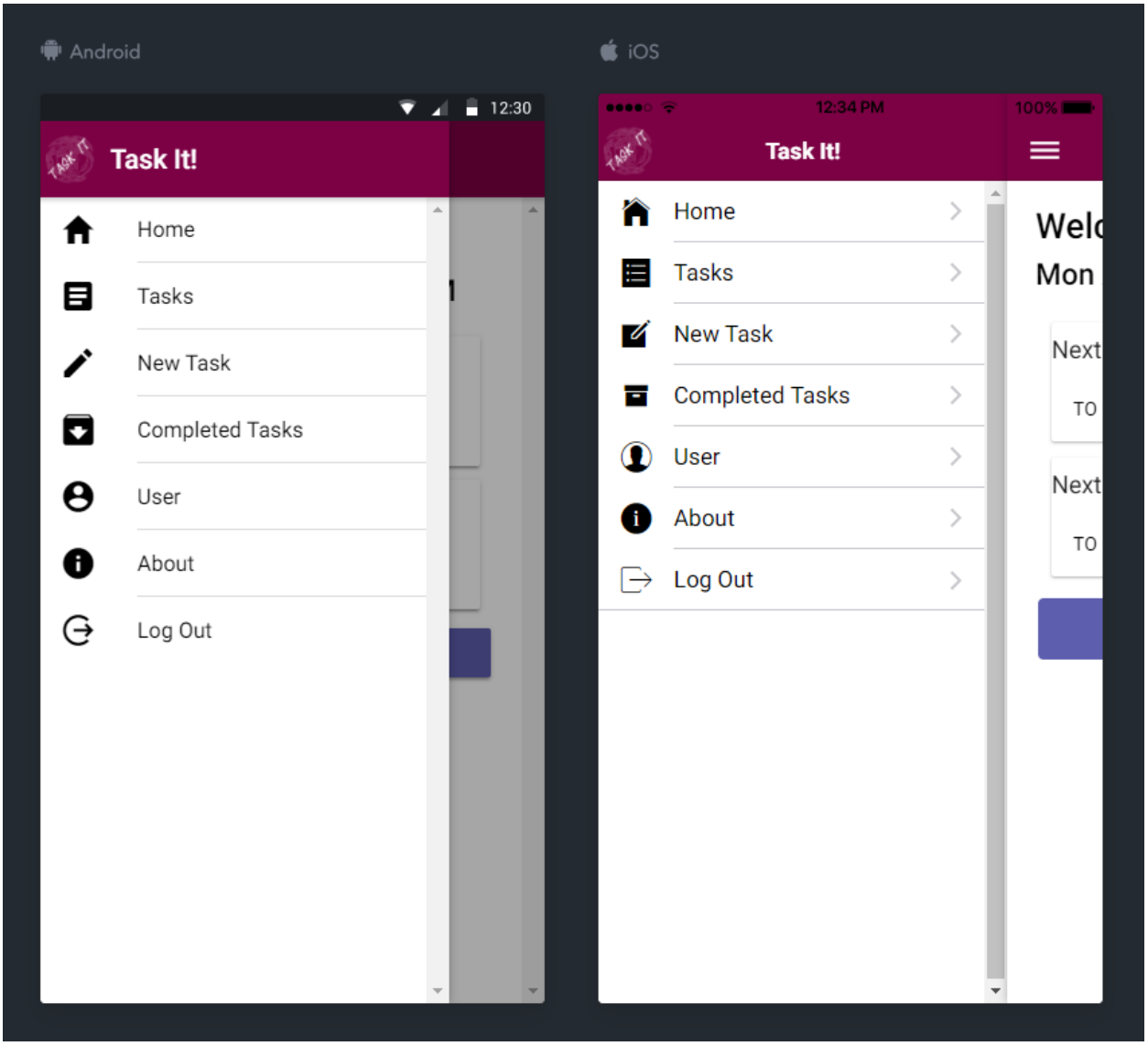
Remembered users (or new users) will have the username and password fields populated automatically. Signing in with “Remember Me” unchecked will delete the current user details from local storage.

Once signed in, the user will be taken to the Home screen:



The user will be greeted by name, with their next due task and next high priority task (if any) displayed in medium detail. A button will provide the user easy access to the main Tasks page.

The Menu button in the upper right opens the left rail navigation menu:



From here users have access to the six primary pages, as well as the option to Log Out (which deletes the current “UserModel” from memory and returns the user to the Login Page).

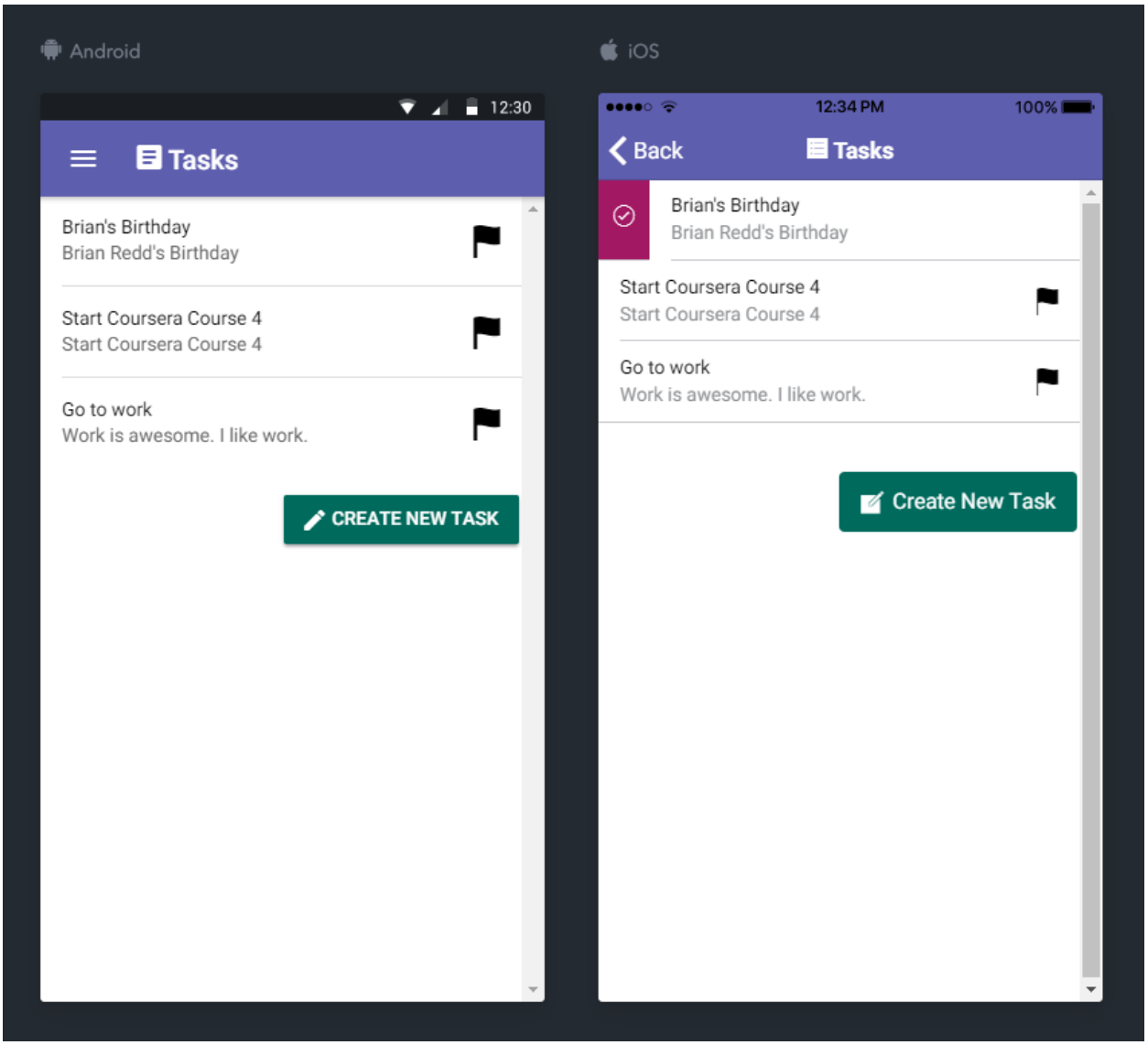
Tasks

The purpose of this application is to allow users to view, sort, create, edit, and complete tasks. This is done by displaying and manipulating the user-specific Task Object.

As mentioned above, on the Home Page the next due task and the next high priority task are displayed (with priority going to overdue tasks). If the next due task is a high priority task, then the next high priority due task will be displayed in the high priority card.

Each new user will be automatically provided with a Task: if they entered their Birthday during New User setup, then their Birthday will be set as a recurring Task; otherwise, a “Sample Task” will be created instead, with a “due date” of the next day.

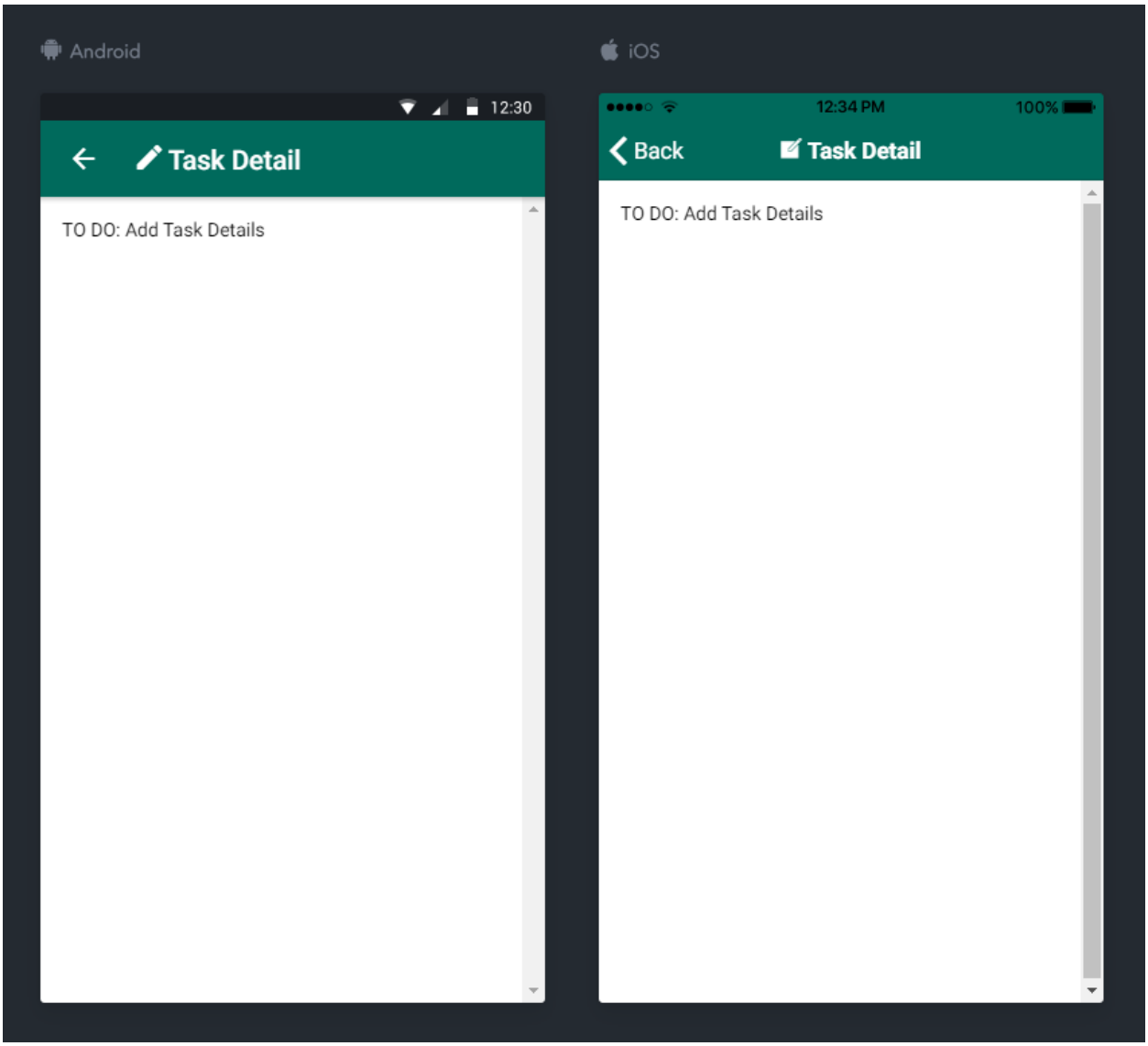
To view all non-complete tasks, user will go to the Task Page:



Styling and Ionicons will be used to designate tasks by priority, near due date, overdue, etc.

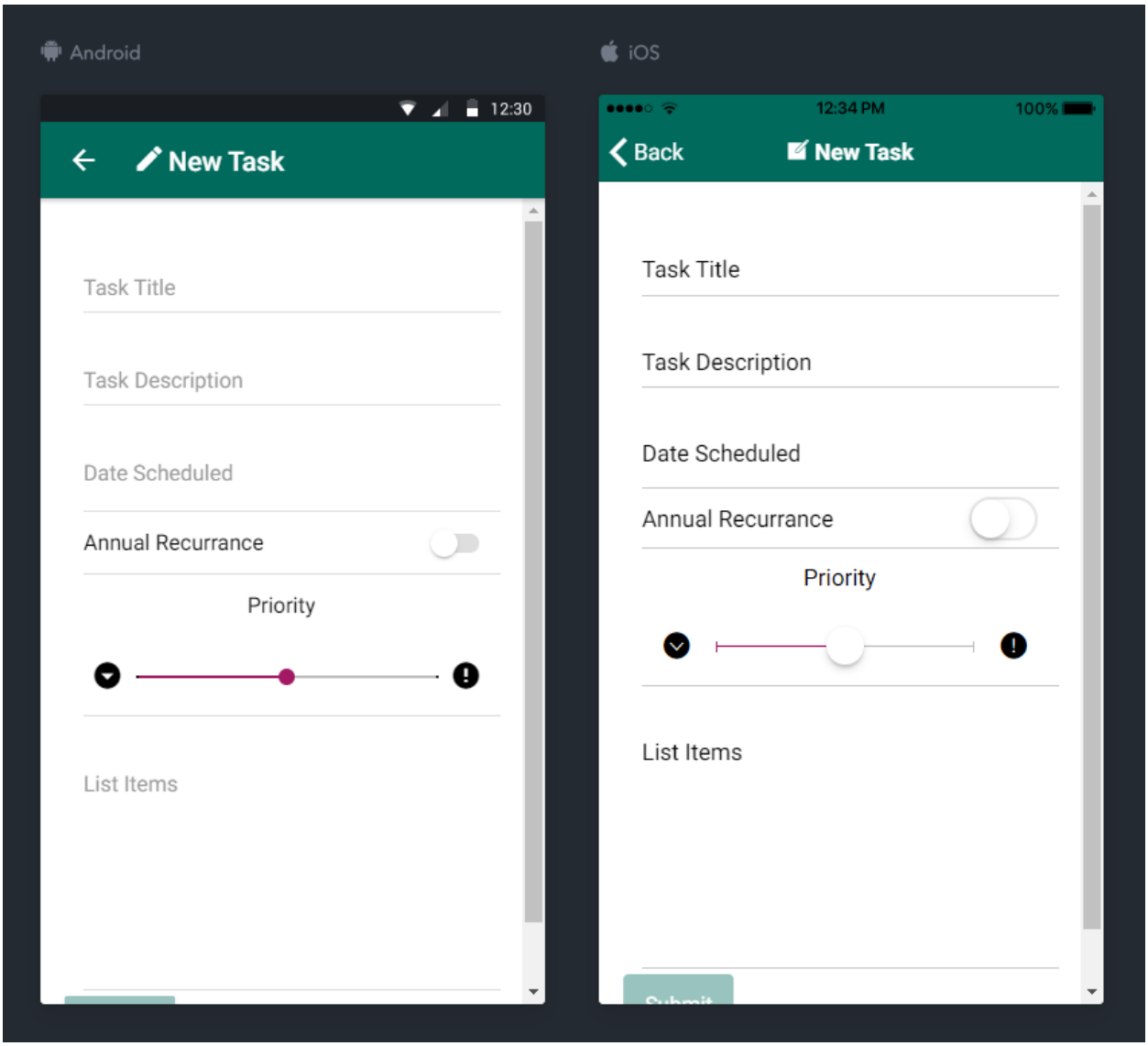
Marking a task as complete is as easy as swiping the task to the side and clicking the “Done” icon.

Tapping a task will open the Task Details Page:



This page will show details of the selected task, including Title, Description, Creation Date, Due Date, Edited Date, Completion status, Completion Date, Recurring Status (does task repeat), Priority, Category, and a List of sub-tasks. There will also be buttons to Complete Task, Edit Task, Copy Task (creating a new task based on the current task), and Sort Tasks.

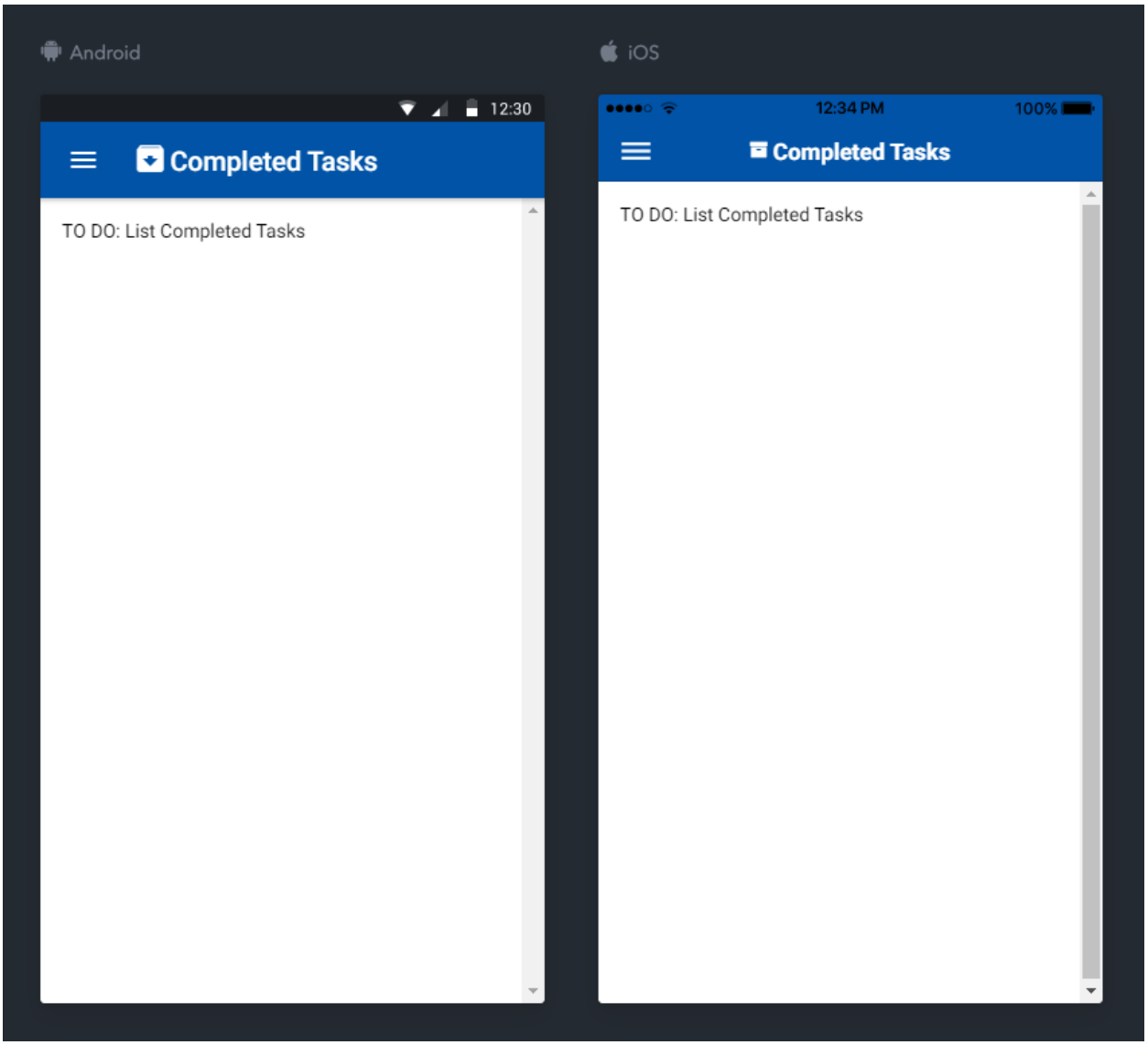
The List field is a child array of items, each with its own ID, Name, and Completion Status. This would be used for Tasks such as “Go Grocery Shopping”, with each item in the list a separate item to be bought. From the Task Detail page, users can swipe each item to complete it.



New Tasks and Task Edit pages will be very similar, and allow users to edit fields for each new or existing Task. Priority will be provided with a range-slide; once implemented Categories will be a drop down and Recurrence can be specified as Weekly, Monthly, or Annually. List Items can be added, copied, and deleted while in Edit Task.

Eventually, the device’s native camera will also be hooked into the app, so pictures can be taken and added to Tasks. For instance, if you need to buy a specific brand of something, you can take a picture of the label and attach it to your shopping list Task. Or, if a friend borrows a DVD, you can take a picture of your friend holding the DVD and create a Task reminding you to ask for it back.

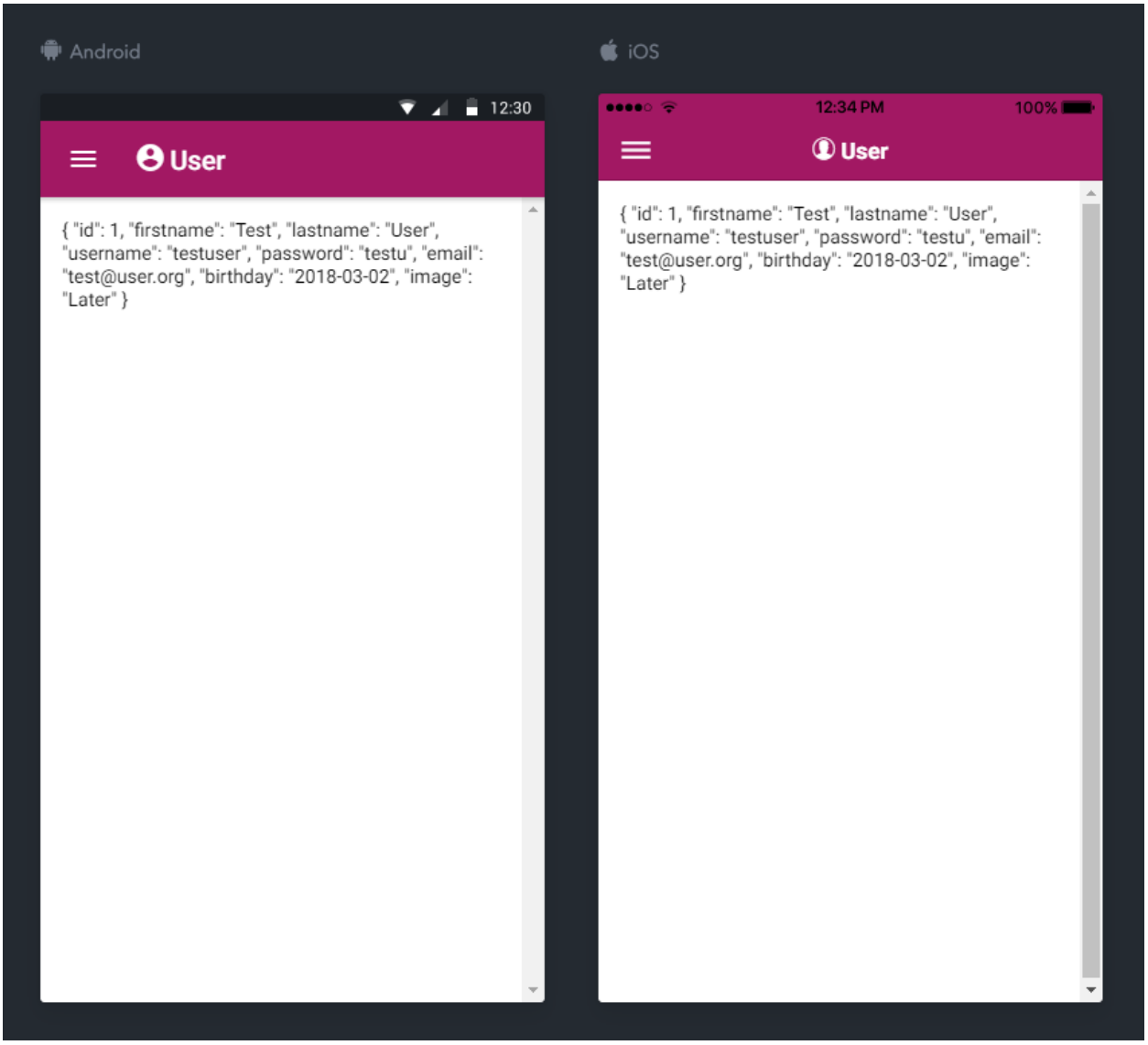
Finally, Completed Tasks, while not displayed by default, are available in the Completed Task Page:



This will list all completed tasks in descending order of completion.

User Details and About

In addition to Tasks, the app also grants the user access to their User details, which not only lists their user information, but also will allow the user to edit First and Last Name, Password, Email address, and Birthday.

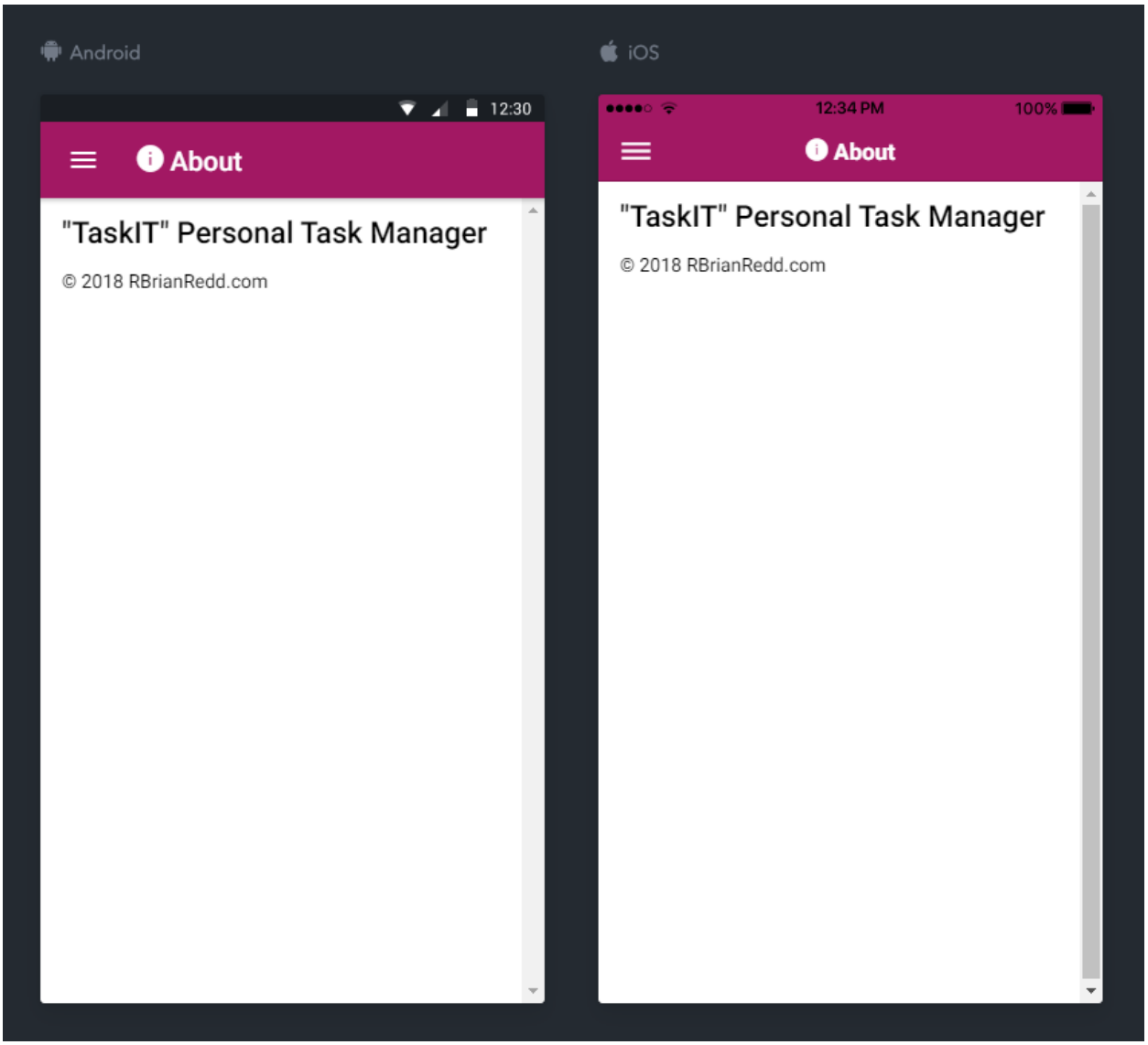


In addition, user will eventually be able to use the device’s camera to take a picture of themselves and add it to their User details.

The one visible field the user cannot modify is the Username; likewise, they’ll never be able to edit (or actually see) their User ID.

Once a server is in place to support this app, email verification will be used to reset forgotten passwords.

Finally, there is the About page, which will provide details about the App (version history, Help, etc) and myself.



Implementation Challenges

Setting up the Login component and necessary data services was more of a challenge than I had anticipated. I knew I needed services in place (in anticipation of data being provided by and to a server), but the Observable configuration for local storage is different than what we’ve learned for online data (in particular pulling the data via HTTP module from a JSON-server). However, I was eventually able to get the pieces working smoothly.

The advantage of a modular design like what Angular and Ionic require is that now that I’ve completed the Login component for my “TaskIT” app, I should be able to repurpose it for other Ionic applications that require login.

In addition to the challenges of the Login component, I had struggled with the Task dataset in my original incarnation of this app. For security reasons, it is important than when the Task data is put onto a JSON-server, that each user’s tasks be stored separately. This means that three “Value Objects” were needed: UserVO (containing the fields used in the User objects/UserModel), TaskVO (containing the fields used in each task) and a “UserTaskVO”, which contained simply an ID (to be mapped to a User ID) and the “TaskVO” object. This way the ID value could be used (much as we did for DishDetails in our Angular and Ionic class examples) to access the JSON object, and that ID-based object would then contain its own Task array containing all of that user’s Tasks.

Luckily, this is not as complicated when using just local storage; for local storage, I simply store each user Task object as “Tasks_*x*”, where *x* is the User ID.

Conclusions

While I know there are plenty of personal task manager apps available (and many that are much more robust that what I am creating) I am looking forward to completing this “Task” and creating a usable app.

References

- Houser, Jeff, *Learn With: Angular 4 and Bootstrap*, www.learn-with.com
- Angular: https://angular.io/
- Ionic Framework: https://ionicframework.com/
- Ionic Icons: https://ionicframework.com/docs/ionicons/