Dream Up

Mobile App Development II

McCormack, Owen
Owen_McCormack@student.uml.edu

Phillips, Jacob
jacob_Phillips@student.uml.edu

Spring 2019

Abstract

Our app is a dream journal. The user can enter information about their dreams when they wake up, and our app will save it to a database. The app can send a reminder every morning to tell the user to enter any dreams they may have had from the night before. The user enters detailed information about their dream through a reaction, drawing, fragments, tags, and a description. We give the users a large color palette to choose from so they can draw what they remember from their dream. Once the user has set everything and drawn their picture the dream is saved to a local database. The number of fragments each dream has during a week is shown on a graph. Hopefully in the end we have created a useful app for keeping a personal dream journal. Since it sends out notifications every morning and has lots of information about a dream it should be very effective when helping users remember dreams that they otherwise would not be able to.

1 Project Summary

Our app is a minimal and locally stored dream journal. The major features we were looking for when we originally got started included: a touch canvas where the user could create drawings, repeatable reminders that would remind the user to enter their dream, and a clean interface to enter and view all the detailed information they want to about their dream before it was stored into the database. We were able to successfully implement each of these core features, not without their own individual challenges along the way.

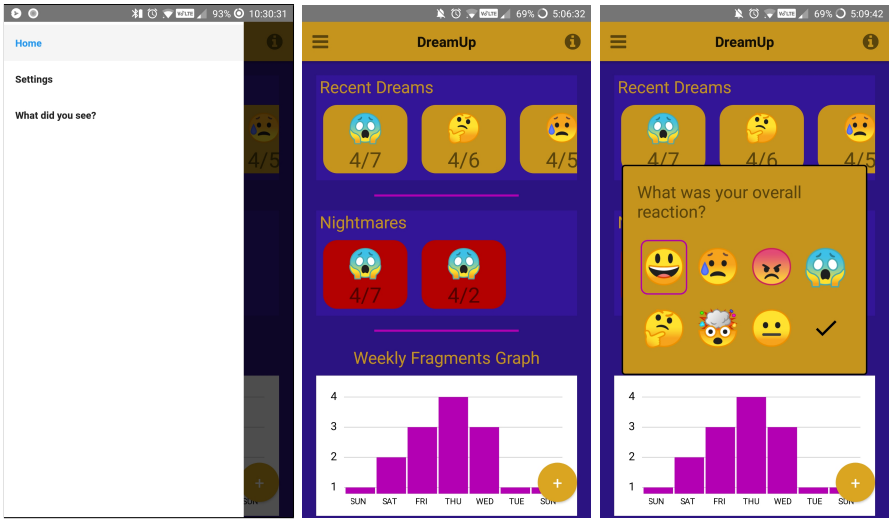
In its final form, our app includes 5 different screens that will be broken down in detail. Before diving into the screens though, it's important to identify what information the user must enter for us to store their Dream as a javascript object. A dream has 7 fields connected to it:

- **createDate** - date user adds the dream
- **visionPath** - path to the image the user drew
- **fragments** - list of one sentence descriptions from the dream
- **reaction** - emoji face representing overall reaction to the dream
- **tags** - optional list of keywords relating to the dream
- **description** - optional full text description
- **id** - unique int representation of createDate
- **nightmare** - bool category hidden from user

1.1 Dream Dashboard

The main screen the users sees when they enter the app is the "DreamDashboard". This screen presents a dashboard for the user to get an overview of the Dreams they have entered into the app. Two "Card Containers" take up most the dashboard, the "Recent Dreams" CardContainer shows the user a scrollable list of all the dreams they have entered in chronological order. The "Nightmares" CardContainer shows a chronological list of nightmares, which are dreams where the user picked the "Afraid" emoji reaction when they were created. Tapping on any of these "DreamCards" links the user to the "DreamScreen" for that particular dream which will be explained

more later. Long pressing any of these dream cards brings up a dialog asking about removing the dream permanently. Choosing Yes will remove the Dream from the DB and update the list. The last component on the Dashboard is the "Weekly Fragments Graph". This component shows a dynamic bar graph of how many fragments each dream has had over the last 7 days. The graph will show stats for the last 7 days, whether they were entered on consecutive days or not. For example, a user could have a dream on 4/6 and then not record another until 4/8 but the bars would be right next to each other on the graph. This is why you might see two of the same weekday on the graph. The user has two options if they want to leave the DreamDashboard: they can start the process of adding a new dream by tapping the FAB in the lower right, or they can access the drawer menu on the top left to either look at settings or start adding a dream from there.



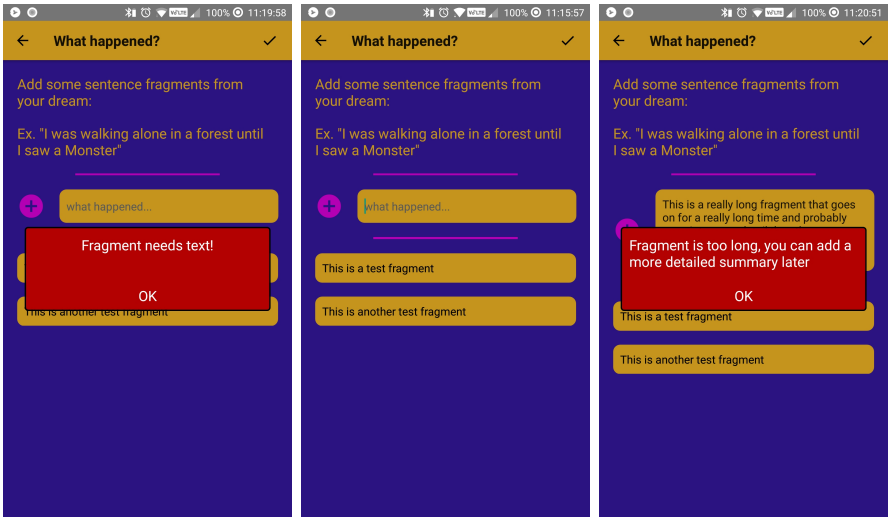
1.2 Vision Canvas

Once the user has chosen a reaction for their new dream they are moved to the "VisionCanvas" screen. On this screen they can draw whatever they want for the dream using various colors, erasers, and stroke widths. The canvas we integrated was an open source npm package called react-native-sketch-canvas, which will be discussed more in the Technology Used section. Once integrated we restyled the canvas to fit the colors and design of the rest of the app and we added a save callback that saved the path to the image as visionPath in the DreamObject. Drawn images are stored in the user's pictures folder so that the users themselves may manage them. The

new dream object is not stored in the database yet, all we have from the user so far is the reaction and the vision(drawing), we still need the fragments. Because of this, the path and reaction are passed as params through ReactNavigation to the next screen, the DreamFragmentScreen.

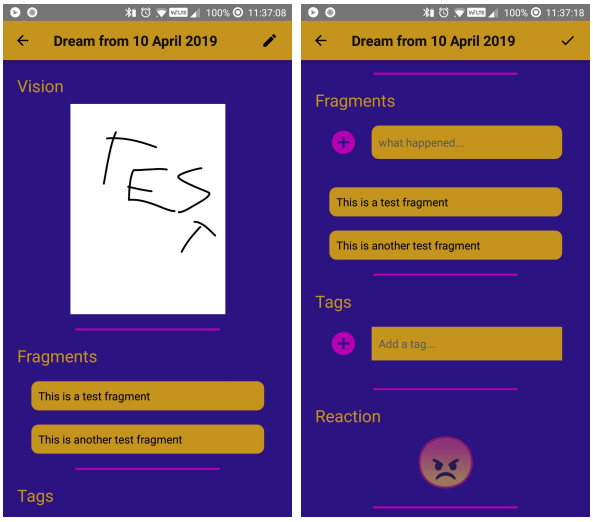
1.3 Dream Fragment Screen

This is the final screen before the new dream gets stored in the database. If the user decides to back out of the new dream process at this point, the only thing saved is their drawing in the phone's pictures folder. This screen encourages the user to describe what happened in their dream in small "fragments". The idea was that people have great variety in their dreams, sometimes we have a very linear story that seems to encompass the entire night, while other times, dreams seem like a set of loosely connected events. Fragments allow the user to define their dreams structure in a concise, yet memorable, manner. To enter a fragment the user simply has to enter their sentence in the textInput and tap the plus button. Fragments that are empty or too long will not be able to be submitted and an error message is provided. Once a fragment has been added it can be seen below the input and tapping it brings up a dialog for deleting. In order to move into the next screen by tapping the done icon in the top right, the user must have at least one fragment.



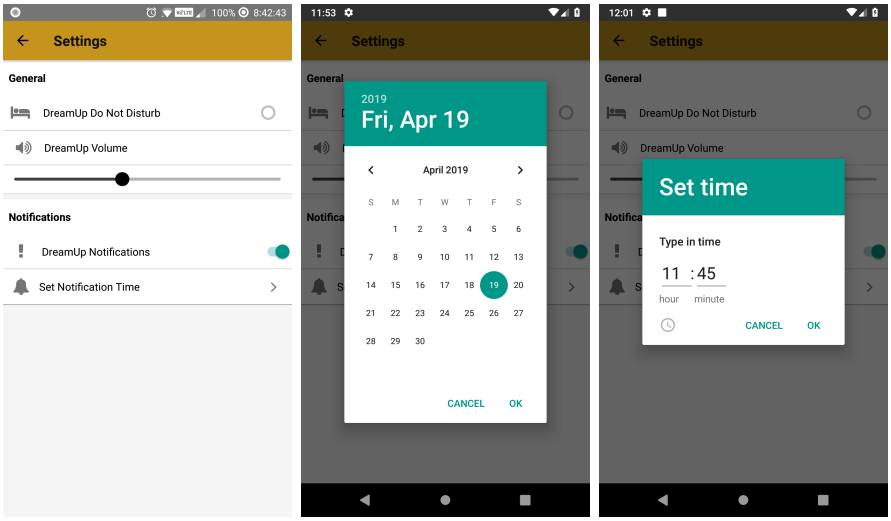
1.4 Dream Screen

Once the user has reached this screen the dream object has been stored in the database. This screen serves as an overview of all the information relating to the user's dream in the order vision, fragments, tags, reaction, and description. Initially after creating a dream the Tags and Full Description sections will be blank since they are optional fields. The title at the top shows when the Dream was created however that's not something the user can change. This screen has two states, readOnly and edit mode that are toggled by tapping the icon in the top right. ReadOnly mode is exactly how it sounds, none of the fields are interactive however in edit mode components for adding new fragments and tags appear and the description becomes editable. Once the DreamScreen is put back into readOnly mode, these changes will be updated in the DB.



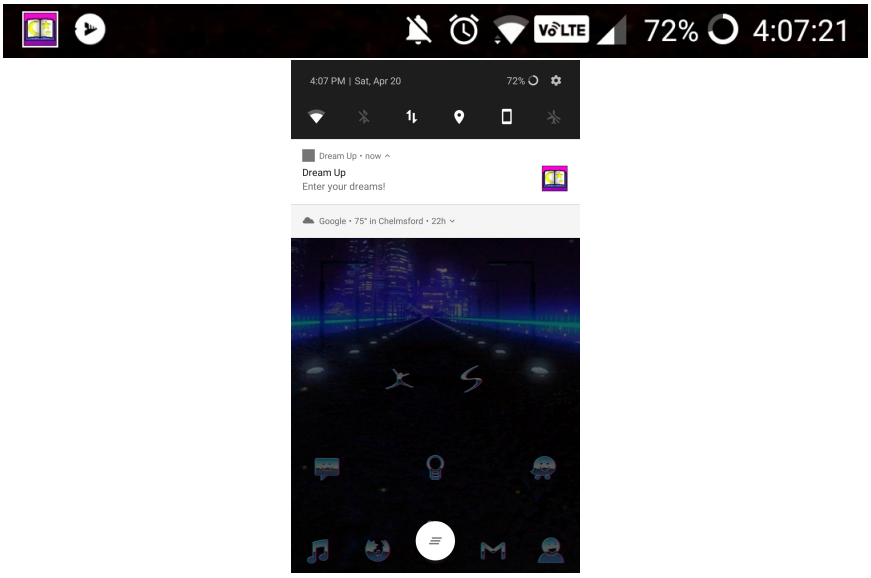
1.5 Settings Screen

The settings screen shown below can be accessed from the menu discussed in Section 1.1. It provides the user with all the option listed: do not disturb, volume, notifications and set notification time. Do not disturb will silence and disable all notifications for DreamUp when turned on. Our app only uses sound for notifications, so the volume controls that and the notifications will not have sound when turned all the way down. To set the notification time select the option and a date and time picker will open. The notifications will repeat daily at the time you enter starting on the day selected.



1.6 Notifications

Notifications will be sent out at the time selected in the date and time picker shown in Section 1.5. They are sent using firebase and react-native-push-notifications on a daily schedule. Our hope is the notifications will remind users to enter their dreams when they wake up since dreams are only stored in your brain's short term memory. Hopefully with notifications users can easily fill up their Dream Up journals.



2 Technology Used

The main technology we used for our project was React Native. React Native allowed us to break up our UI into reusable components that saved a lot of code and time. For example, The error dialog and Fragment textinput were both custom components that got used more than once, keeping the code clean and organized. Because of its popularity, React Native also had many open source npm packages that were simple to integrate into our projects. Stuff like the canvas, graphs, and DB were not as much of a time sink as they could have been, allowing us to focus more on design. Here's a full list of the npm packages used with links:

- react-native-sketch-canvas
  `https://www.npmjs.com/package/@terrylinla/react-native-sketch-canvas`

- react-native-local-mongoDB
  `https://www.npmjs.com/package/react-native-local-mongodb`
- react-native-vector-icons
  `https://github.com/oblador/react-native-vector-icons`
- react-navigation
  `https://www.npmjs.com/package/react-navigation`
- react-native-fab
  `https://www.npmjs.com/package/react-native-fab`
- react-native-firebase
  `https://www.npmjs.com/package/react-native-firebase`
- react-native-push-notifications
  `https://www.npmjs.com/package/react-native-push-notification`
- react-native-svg-charts
  `https://www.npmjs.com/package/react-native-svg-charts`

This is the main list of packages, however there are a few other small ones for things like header spacing and such. The full list can be found in the package.json file of the project.

As mentioned earlier, our database used an implementation of MongoDB implemented for React Native. Originally we were going to use SQLite as a database, but there wasn't a very good package for working with it in React Native and the Javascript objects we wanted to store fit the querying better for a noSQL database like Mongo. We also used firebase for all things related to sending out and scheduling the notificaitons.

3 Future Improvments/Lessons Learned

If given more time there are a few things we wouldn't mind changing about our project. The most visible change we would want want to make is to add more graphs and analysis to the dashboard. Setting up the graph we had was somewhat difficult, and took up a lot of time creating the dashboard. Behind the scenes, I think some of the components that we used could be abstracted more for cleaner code. Other than those things, we feel that our app fits what we set out to do initially very well.

We both learned many things working on this project, neither of us had any experience working with React or React Native before hand so that

was a lot to learn initially. We both also become familiar with creating a multi-platform app, trying to correct our code to work well with iphone and android was always a struggle. In the end, we ended up leaning a bit more in the Android direction since neither of us had the environment to be able to test everything for iphone.

4 Individual Contributions

4.1 Owen's Portion

My work was mainly based around creating react components and building the UI for the front end as well as the navigation between screens. The UI for the VisionCanvas, DreamFragmentScreen, and DreamScreen took up most of my development time. I think the hardest challenge was probably working on integrating the canvas into our app smoothly.

I would say overall I contributed around 50% to the project.

4.2 Jake's Portion

My work was mainly based around creating the navigation from the sidebar menu along with setting up the Settings UI and functionality. I spent a lot of my time on the backend integrating firebase and working with is to get notifications working properly.

Overall I believe I contributed around 50% to the project