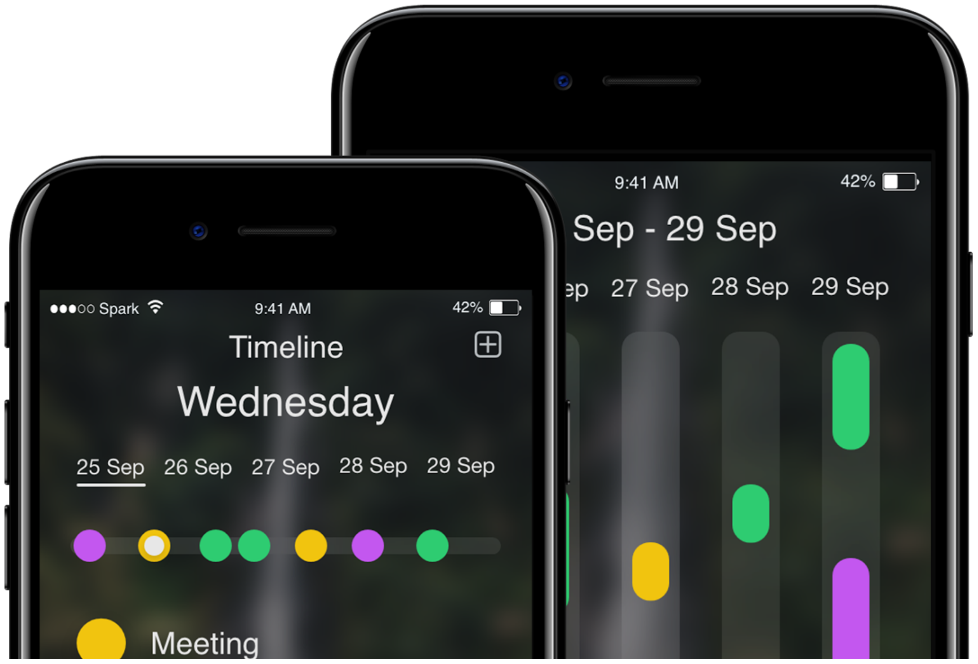
OCTAGON

BE ORGANISED



Charlie Rawstorn, Oliver Reid, Thomas Youngson

SCOPE

Developing an application is a multistage process. If it were as easy as sitting down and writing a few thousand lines of code, everyone would do it. Unfortunately, that's not the case. Feedback is one of the most important stages of development. What does the user like? More importantly, what don't they like? Why don't they like it? How development teams respond to these critiques can be the difference between a good application and a great application. Only when we recognize what is wrong with our app, can we make it better.

In this report, we break down and analyze the weaknesses in Octagon. By doing that, we are able to reflect on what needs to be changed and why. We also focus on how we are going to remedy each issue that we or our users found. This gives a clear sense of what to work on next and helps to set our priorities straight. We also rate the importance of each issue that we've found to create a roadmap which tells us in which order we should work on fixing existing issues. Finally, the issues outlined in the this report will be our primary focus for the remainder of COSC345.

Our team fully believes that our application fills a need. What follows are some of the important steps we need to take before we can be confident we are fulfilling that need correctly.

PROBLEMS AND SOLUTIONS

- - - - -

Android functionality - 10

P -> Currently our application doesn't work on Android. We think this is due to some of Android's security policies. Consequently, our buttons do not respond to click events. This prevents Android users from using our application. Obviously, this is a big problem as we intend on distributing Octagon to all mobile platforms.

S -> As this problem is Android specific and not replicated on iOS, we need to do further research on the Android side of Ionic to see if we have made a simple mistake.

Data management - 10

P -> When a user loads a page, our application does the following:

- Sends a request to the API for a certain type of data associated with the user
  - Currently, we support the following types:
    - Event Data
    - Settings Data
    - User Profile Data
- If the user is online and can connect to the API
  - API responds with the user's requested data
    - The data is sent to local storage then used to populate the relevant page
- If the user isn't online or can't connect to the API
  - The app uses the data stored in local storage to populate the relevant page

Sending a request and waiting for a response before loading a page is very expensive, especially when we consider New Zealand mobile networks. On the other hand, using local storage to store data is a bad idea as it's volatile, and not application specific. That means data saved in local storage can be overwritten by other applications on a user's device. We need to find somewhere in between. Currently our naive solution is to only send requests to the API when a user has an internet connection. Consequently, when a user reconnects to the API after making offline changes, all offline changes are overwritten by the current state of the API.

S -> The obvious solution is to make database calls less expensive. We need to setup an application specific local database. Users can then make calls to their local database instead of waiting for our API to respond. When a user loads a page the process becomes:

- Request data from local database
- Load pages with data generated from local database
- Check to see if the API's information matches the local database
    - Send information to the API from the local database

This way we only send information from the local database to the API, updating the user's data held remotely. This can be done at any time and won't slow the user down. The user is no longer concerned with pulling information from an online API and can make persistent changes regardless of internet connectivity. Since we are still using an online API, users will still be able to log in from other devices and view their current timetable.

Testing - 10

P -> All testing so far has been performed manually. Each time our applications functionality changes, we manually go through and test that nothing is broken. Manual testing to this degree is a huge waste of development time, and relies on the developers finding anomalies and detecting all the bugs within the app. Many problems are often missed due to human error.

S -> We need to setup automated tests to check our applications functionality each time we implement a new feature. Angular provides many testing frameworks that we can use.

Notifications - 9

P -> If a user has an upcoming event, they need to load up Octagon and check the daily view page. This process relies on the users not only being conscious of the current time, but remembering to check their schedule and the app.

S -> We need to implement push notifications. The user will then be reminded that they have an event, without conscious consideration.

Edit events - 9

P -> When a user creates an event there is no way to make edits. If the user makes was to make a typo, they'd need to delete the entire event, and recreate it; hopefully without the typo. Frustrating!

S -> We need to add an edit function that allows the user to fix simple mistakes. This will remove the headache of deleting and recreating entire events.

Stretching out bubbles - 9

P -> Currently timeline bubbles are displayed as dots representing each event's start time. This only tells the user when an event is starting, but doesn't indicate when it finishes. This means it's easy for users to create conflicting events as it's unclear when an event finishes.

S -> Stretching bubbles out to their finish time will allow users to see if they have any clashes during a given day. To stretch bubbles on the timeline, we will take the user's phone display size and then stretch the bubbles height and width along the timeline, according to how much of the day the event takes up. This will give users a better understanding of how their day is laid out and enable them to gain important insight about their day at just a glance.

Grey out submit buttons when a form is invalid - 8

P -> Currently our form submit buttons (Save and Add) are clickable by user's regardless of whether the user has entered valid form data. Although, we have used client and server side validation for checking that the user has entered valid data, by allowing users to click the submit button with invalid data we encourage them to try and submit invalid data. We'd much prefer working in a prevention rather than cure paradigm and believe that by only enabling the submit button when a form has valid form data, we will further help users fill out forms with valid data.

S -> Most common applications only display the submit button when a form contains valid information. Although this approach is good, it alone doesn't help users understand what breaks form validation. We will change our form submit buttons to be disabled while the form is invalid as well as display error messages.

This will help users understand that they've failed to fill the form in correctly and provide immediate feedback about what violates the form validation. As users edit their input they will get an immediate sense of whether they are addressing the issue or not.

Reorder daily events - 8

P -> When a user adds an event to the current day, and they are not connected to the internet the event is added to the bottom of the daily view page, regardless of its start or end time. This confuses users and could cause them to miss out on important events.

S -> This should be solved when we implement a local database. When a user loads or updates a page, we can simply get the required data from the user's local database and re-render the page.

Landscape view - 7

P -> Depending on the user's settings, Octagon will automatically rotate to landscape view, despite Octagon not being designed to be viewed in landscape mode.

S -> When building the application in Xcode we need to disable landscape mode.

Improve page navigation - 5

P -> Currently our login and register pages stack on one another. This makes the pages feel as if they are on different levels of the app hierarchy. Navigation of the app when the user is logged in also feels odd, as users are unable to move between pages with a swiping motion.

S -> We can fix the login/register navigation by changing the way pages are placed on the view stack. This should be a reasonably straightforward change. The swipe motion navigation can be implemented by enabling the setting for each page. This will allow users to explore the app faster.

Application icon - 5

P -> After compiling the application for use, we don't display an Octagon icon. Instead, Cordovas default application image will appear with our name underneath.

This will confuse users and miss a significant chance to instil a good sense of brand recognition.

S-> We need to create a logo we can attach to the application.

Email field auto caps - 3

P -> When a user enters their email address to login or register, clicking out of the email field will automatically capitalise the fields first character. This creates a barrier to joining or using Octagon, as some users may be unable to login.

S -> We need to override the default settings on the form to stop this behaviour.

Status bar - 3

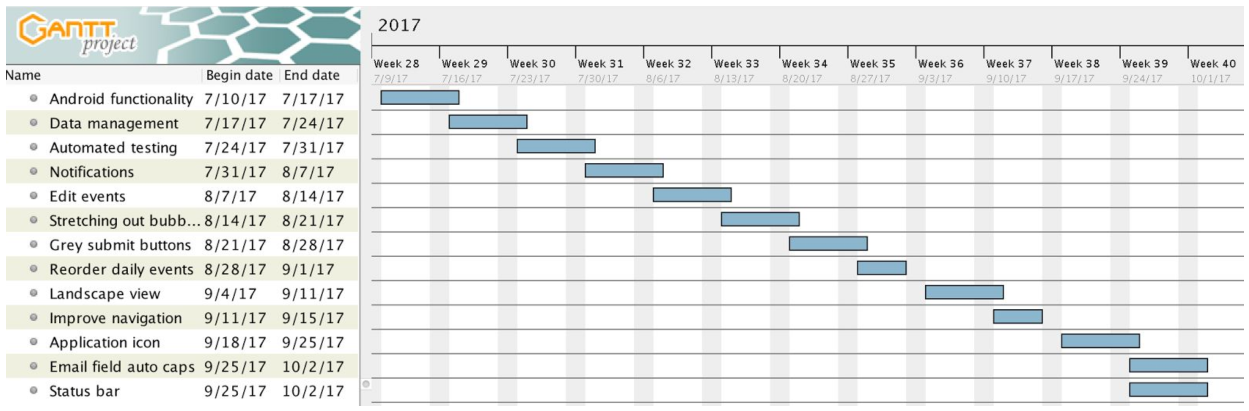
P -> When using the application on an iPhone, the status bar is white and doesn't match Octagons theme. This detracts from the smooth UI that we want users to experience.

S -> We need to add a transparent styling to the status bar.

FUTURE

- - - -

That pretty much sums it up. We've attached a Gantt chart to the back of this report to show you how we plan to address each of these issues moving forward.



Name	Begin date	End date
Android functionality	7/10/17	7/17/17
Data management	7/17/17	7/24/17
Automated testing	7/24/17	7/31/17
Notifications	7/31/17	8/7/17
Edit events	8/7/17	8/14/17
Stretching out bubb...	8/14/17	8/21/17
Grey submit buttons	8/21/17	8/28/17
Reorder daily events	8/28/17	9/1/17
Landscape view	9/4/17	9/11/17
Improve navigation	9/11/17	9/15/17
Application icon	9/18/17	9/25/17
Email field auto caps	9/25/17	10/2/17
Status bar	9/25/17	10/2/17

We look forward to touching base next time and talking about how we overcame each issue. Until next time.

Octagon Development Team