AroundTown : Project Design Document

Project name	AroundTown	
Project leader	Karen Zhang	kz7@princeton.edu
Group members	Hrishikesh Khandeparkar	hrk@princeton.edu
	Timothy Lou	tlou@princeton.edu

Section 1: Overview

Inspired by products and services like Facebook and Waze whose valuable content is created by the users, AroundTown is a mobile application for Princeton students and staff that integrates live events or alerts and their related locations onto an interactive campus map and provides users with live notifications of nearby events. These events or alerts are reported by users by dropping a "pin" and a short description onto a map of the area, through a simple interface similar to that of Waze, a successful traffic and road condition crowdsourcing app. Other users of the app can see these events appear on the map and receive push notifications about events popping up at locations near them, or types of events they are interested in. AroundTown is location-aware, meaning that event-related push notifications can be specific to the user's location.

Some of the general types of events and alerts that AroundTown supports are:

- "Free food": a common announcement that interests many students
  - AroundTown offers a better alternative to the freefood listserv, one of the most popularly subscribed-to listservs at Princeton
- "Study break": most often organized by student organizations around campus
- "Red alert": for reporting dangerous situations, such as robberies or accidents
- "Recruiting": for networking events, receptions, information sessions
- "Facilities status": for reporting broken printers, laundry machines, toilets, etc.

Section 2: Requirements and Target Audiences

The problem we are solving is two-fold. Every day, an enormous amount of information related to on-campus events is generated by student organizations, individual students, USG members, departments, pre-professional societies, etc. This not only includes emails and other reminders about events, but also simpler alerts like where a road might be blocked or where a printer is broken. All of this information can have enormous relevance to a student's day-to-day, but there is no current solution to managing all of this data, presenting it in an organized fashion, or providing only events that interest a particular user. The second problem is that while many things students do are planned, a lot of their other activities like stopping by a study break for a snack are spur-of-the-moment decisions, often because of convenience. Oftentimes, students find

themselves unaware of events of interest, even if they were at the right place at the right time.

AroundTown is a platform that provides solutions to both problems, allowing for a seamless, digestible stream of event/alarm-related data and a notification system that helps students be more efficient, productive, and aware of campus events and resources. The primary users of AroundTown are Princeton students, while secondary users may include faculty and event organizers. With the app just running in the background, relevant notifications will be sent directly to the user for a more convenient and connected lifestyle. Users can additionally explore the app's interactive map to check conditions in other parts of campus, giving them easy access to all the information they want, whenever they want.

Section 3: Functionality

The following is a list of typical use scenarios.

On the search for free food

Tim opens the app on his phone. He sees a view of the map centered at his current location, with a few pins within his view. He clicks on the each of the pins to see if any of them denote free food. Every time he touches a pin, a text description appears with details about what is going on at that location. He does not find any free food within his map view, so he zooms out to see a larger view that includes several buildings on campus. He sees a few pins at Frist campus center, so he moves the map's field of view so that Frist is centered on his screen, and zooms in. He clicks on a "free food" pin and reads the description, which includes the time that has passed since the pin was dropped and the description of the food. Also, based on several upvotes by other users, he decides that the chance of getting free food from Frist is pretty high and starts walking towards Frist.

Looking to print a document

Karen, sitting in her room in Blair hall, has just sent a document to the cluster printers. She checks her phone and notices a "facilities status" push notification from the AroundTown app. She opens to the app to make sure the printer she usually uses is working. When she opens the app, the map view is centered on Blair, and there is a pin on the location of the Blair printer room. When she taps on the pin, the description says that the printer is out of ink and will not properly print documents. She drags the map view to focus on nearby Joline hall, where there is no "facilities status" pin. She goes to the Joline basement to print her documents and avoids the frustration of dealing with a broken printer. Whew!

Late to a food study break

Hrishi is hungry, and he happens to receive a notification from the AroundTown app about a Mathey pizza study break from thirty minutes ago. He realizes that he still wants food, and opens the app. He zooms in on the Mathey common room, and sees that no one has commented "all gone" yet on the original pin. He decides to check for himself if there is any food left, and walks over to the Mathey common room where he finds one slice of pizza left. He happily eats the pizza slice, and to prevent other people from being disappointed by the food being gone, he opens the app again, clicks on the study break pin, and comments that the food is all gone.

In search of a job at a career fair

Jasmine is attending a career fair in Dillon gym. Between talking to Google and Facebook, she checks her phone and sees a notification for free food in a nearby building, which reminds her that she should put a pin down to notify people about the career fair. She opens the app and sees that no one has created a pin for the event yet, so she drops a "recruitment event" pin on Dillon gym. The app prompts her to type in a short description of the career fair, so she types a description and lists a few of the main companies that are present. After she is done, the app sends out a notification to the appropriate groups of people (i.e. people in the location, people who have chosen to always receive notifications from the Dillon gym area).

Section 4: Design

Since our app is primarily mobile based we will try to leverage many open source app development tools to attempt to build a native app with a visual UI. For developing our app we will primarily use React Native (RN). This will enable us to have a codebase that's shared across platforms. We highlight some of the main components of our code base and our plans on how to implement them.

Components

UI/UX

Since our primary UI/UX component is a interactive map, we will heavily rely on the react-native-maps package (credits: AirBnB) that integrates well into the RN framework. This package supports most of our essential operations. Some of them are

- Dropping pins on long click
- Looking at pin contents on click
- Most basic map based operations like zooming, panning, and possibly even searching.

If integrating this package becomes a lot more complicated that we foresee, we might choose to replace the UI by a more static version of the map of Princeton, and support the essential operations that we highlight in our MVP (minimum viable product) by coding them from scratch.

Another component of the UX is the push notifications - we will handle the creation of these notifications in the backend, and hope to able to use the react-native-push-notification package for simplifying the porting to iOS/Android. Again, if this becomes harder than we foresee, we might manually code push notifications solely for a native Android app as highlighted by our MVP.

We don't foresee having to use any fancier UI than the one that react-native-maps already offers, but if needed we may use React to make a slicker front end.

Backend

We will again heavily use React Native to code our backend - this backend will need to keep track of of location updates for all users, pin updates when pins are dropped/deleted, and also be able to send out (push) notifications to subsets of users.

The map based operations are handled well by the react-native-maps API, and to support push notifications we will use the react-native-push-notification package. In the MVP, if we choose to use SMS based notifications, we may use Twilio to handle that.

We are split between how we will update locations of users constantly with the backend - we may choose to update them periodically from the backend, or have individual user instances to send updates to the backend whenever a significant change in location is noticed. However, we think it will be very important to create a standardized API for our our backend before we worry about the specific implementation details. We expect a decent amount of moving parts in this whole location based interaction, and therefore will make specific API design decisions before making implementation decisions, since we can't anticipate which one of the above approaches will work better.

Database

Our database will store most of the information that is potentially needed by the app. Primarily, this will include the locations of all users, their preferences, and the locations and contents of all the pins present on the map. It is unclear about if using a powerful database like SQL/MongoDB based database might be necessary. A key aspect of our database will be the ease of selecting subsets of users to send push notifications to. In our MVP we plan on integrating this database into our backend, and later separating it from it as is needed.

Section 5: Timeline

Mar 19-25 (Spring break)	- Get familiarized with React Native, JavaScript - Learn the features of the react-native-maps and react-native-push-notifications API. Toy around with implementing basic apps created using these packages.
Mar 26-31 - Project website	- App displays a map - App handles basic map interactions (moving, zooming)
Apr 1-8	- Create a standard API for backend interactions - Create a standard API for selecting a subset of users for push notifications. - App can track location of user - Be able to place/display “pins” on locations on a map - Start implementing backend (store pin info)
Apr 9-15	- Refine backend API for unforeseen issues - Finish implementing backend (store pin info) - App can send information to user (SMS or push notifs)
Apr 16-22 - Prototype	- Minimum viable product complete - Users can place pins and receive notifications - Start working on location specific notifications for subsets of users
Apr 23-30 - Alpha test	- Add pin categories and filters - Users can comment/give feedback to existing pins - Implement user location updates
May 1-7 - Beta test	- Implement location-specific notifications - Add “home base” notification capabilities - No more adding features after the end of this week
May 8-12 - demo days	- Debug and look for ways to clean up
May 14 - Submission	

Section 6: Risks and Outcomes

One of the most immediate challenges to starting this project is understanding how different pieces of app development fit together because collectively, we have little experience in ground-up app design. However, once we make ourselves familiar with "how to build an app," the rest of the path will be clearer. In the same vein, learning JavaScript and its extensions like React Native will be an involving process.

Aside from these more general concerns, a potential barrier our team may run into is fully developing the notification system. We did some preliminary research on sending push notifications to the mobile device, where push notifications are defined as a notification that pops up on the device even when the user is not using the app. We found that with Android devices, the process of sending notifications is not too involved. However, for Apple devices, there may be more room for error or roadblocks that we cannot predict yet. The react-native-push-notifications API seems to simplify a lot of these issues, but it's unclear how complicated it might become once we start including location based notifications. We thus have considered an alternative to push notifications in our Design plan and our MVP.

Another potential problem is the fitness of the react-native-maps API in our project, since the kind of map we require is relatively lightweight and limited to a radius of at most 2 miles. While search by location is a feature we'd like to add, it is at the moment unclear how to restrict search results to just Princeton area through API. Furthermore, if we find no fitting maps library to use, we will resort to an image-based map where "locations" will be manually mapped out.

To address the absolute downside of all outcomes, we have drafted a minimum viable product (MVP) that details the minimum functionality of AroundTown:

Platform	Android
Notification Method	Text messages
Map	Static image (but still zoomable, scrollable) Will support refreshable map to view up-to-date pins.
Location Services	Searching not supported
Pin dropping	Supported
Pin veracity confirmation by users	Not supported
Subscriptions to certain notifications	Supported (by area and by topic)