Admin Panel Customization in Book-a-Nook

Google Summer of Code 2016

Organization: Berkman Center for Internet and Society

Applicant: Shubham Patel (shubhpatel108@gmail.com)

Mentors:
Jessica Yurkofsky (jessica@metalab.harvard.edu)
Justin Clark (jclark@cyber.law.harvard.edu)

Abstract

Book-a-Nook is an online platform where a user can look up for available community spaces like libraries and make reservation requests to the admin. The admin checks for the conflicts and sends the corresponding confirmations. This project aims to ease the process of confirming such requests. Another major aim is to allow admin to gain useful insights on such demands through graph visualizations and report generation. The project will also aid in notifying a requestor of his/her requests' status and favoriting locations as well as nooks.

Contact Information

Name	:	Shubham Patel
Country	:	India
Degree	:	B.Tech ICT (Information and Communication Technology)
School	:	Dhirubhai Ambani Institute of Information and Communication Technology (DA-IICT)
Email	:	shubhpatel108@gmail.com
GitHub and IRC	:	shubhpatel108
Phone	:	+91-7622086123
Preferable method of communication	:	Email, video conference, chat
Berkman project of interest	:	Book-a-nook

About Me

1. **If you have a link to a resume/CV/LinkedIn profile, include it here.**

   Website: http://shubhpatel108.github.io/

   Resume: http://shubhpatel108.github.io/ShubhamPatel_Resume.pdf

   Linkedin: https://in.linkedin.com/in/shubham-patel-425a1155

2. **Please describe yourself, including your development background and specific expertise.**

   I am a final year student at Dhirubhai Ambani Institute of Information and Communication Technology (DA-IICT), Gandhinagar pursuing B.Tech ICT. I'm presently interning at Practo Technologies Pvt. Ltd, Bangalore as a Platform engineer, spending most of the time with Amazon Web services and writing scripts in Python.

   In the past, I have been associated with Amazon (India) as Software Development Engineer - Intern and worked on Java, Spring, Hibernate and JSP during this 2 months period.

I have worked with 2 startups to solve real life problems. One of them being TheCollegeStore.in where I was vested with the responsibility of leading a team of 8 members in developing a Rails-based platform.

Organizing tech events and conferences to spread technology is another hobby of mine. I lead a Google Developer Group chapter (chapter link) for fulfilling this purpose.

Most of the projects I have built or worked on are web-based and use Ruby on Rails. I would not consider myself an expert but can claim that it is the interest I have very well pursued. During this pursuit, I have also gained significant experience with HTML, CSS, JavaScript, jQuery, AJAX and Sinatra.

3. **What do you hope to gain through the process of participating in GSoC, and specifically by contributing to a Berkman project?**
    - Hands-on experience with Administrate, an admin panel developed by thoughtbot: Customizing its views to suit the project requirements is a big challenge. Furthermore, I have only worked with ActiveAdmin till now. So, it will be a new experience for me.
    - Experience with Slim template engine: Some of the features proposed in the application are additions to User's UI. This will help me to get started as well as go deep into the templating engine used in the project.
    - Better and advanced testing techniques using Rspec.
    - Analysing and improving the code with respect to automated code reviews.
    - Lessons on collaborative coding according to industrial standards and practices.
    - Improvement in documentation skills: Along with the development, I will update the project wiki for the benefit of fellow developers.
    - Work closely with law students at Harvard: Discussing great ideas for building great products can help me contribute in better ways to many other projects too.

4. **Why are you interested in the Berkman coding project(s) you stated above?**

   For the following reasons:

   - The aim of the project is to identify and utilize available spaces, especially libraries for a better purpose. Moreover, it simplifies the reservation process and reduces the admin's efforts in making a decision. So, the project is a certain way to help others through technology and a perfect example of sustainable development. It would be self-inspiring to dedicate months for such a great cause.
   - The broad scope of improvement: The project is in its initial stage and many features can be added. This will help me contribute more to the major features and push it to the first release.
   - My affection for Ruby *and* Rails: I have worked on many major projects based on Ruby and Rails. I am always looking for opportunities where I can learn more about the language as well as the framework and take a step towards perfecting this skill.

5. **Have you reviewed the important dates and times for GSoC 2016?**

   Yes.

6. **Do you have any significant conflicts with the listed schedule? If so, please list them here.**

   No, there are no significant conflicts with the listed schedule.

   I'm presently involved in an internship which will end on 29th April 2016. I'll join for full-time employment around mid-August, which would be concluding time for this programme and hence will not be a hindrance. The summer between these dates will be fully dedicated to this project.

7. **Do you understand this is a serious commitment, equivalent to a full-time paid summer internship or summer job?**

   Yes. I will dedicate 45 hours per week to this project. If necessary, I can utilise the weekends too.

8. **We strongly prefer students that provide code samples, ideally in a "social coding" site like GitHub. Please provide a link to code you've written, whether it's a zip file / tarball you host on your own or your Github profile.**

   These are some of the projects I have built and worked on:

   - TheCollegeStore - An online platform for selling and buying used books built using Ruby on Rails and PHP.
   - Prologg [code] - An integrated developer profile which uses REST APIs of prominent development sites like GitHub, TopCoder and Codeforces to give insights on the “programmer” personality of a person. It is built using Ruby on Rails.
   - ExternDisk [code] - A Linux based platform built to share the files within a LAN. It is built using Ruby for the back-end, Shoes Framework for the front-end and Sockets for connectivity.
   - There are other projects too and can be found on my GitHub account.

PROJECT PROPOSAL

Project Objectives and respective Solutions

- To allow an admin to easily confirm a reservation request with having to put minimal efforts in checking the conflicts by:
  - Adding Welcome dashboard in admin panel for showing the most important information like unreviewed reservation requests and recently cancelled reservations.
  - Allowing to check Nook's availability on the click of a button
  - Adding Calendar view to show all reservation requests segregated on the basis of their status.
- To allow an admin to gain useful insights on reservation demands
  - Adding Report dashboard in admin panel depicting location as well as nook based statistics in the form of a variety of graphs.
  - Adding a flexible filter to adjust which entities' data should be present in the visualizations.
- To ease the reservation process for a user by
  - Adding calendar view for showing all public reservations to save him/her from the frequent embarrassment of getting his/her requests rejected.
  - Sending quick notifications on the dashboard as well as mails, to provide updates on the reservation status.
- To improve the platform's reliability and user experience by
  - Mapping Libraries on geo-maps and showing them in search results
  - Providing personalized view to each user in his/her respective time zone
  - reducing cancellation frequency
  - Providing emergency contact information of a location.

The majority of the time will be spent on developing the admin panel. This is because it is the first point of entry for most of the data.

Detailed Plan with Timeline

I have clustered all the tasks into various code sprints. A typical sprint will last for 1 week, except for implementing larger features which could take an additional week. Each sprint is relevantly named so they can be added to GitHub tasks tracker easily.
For most of the tasks, corresponding issues were created at the GitHub repo to brainstorm on the ideas. Look for the ⇪ icon for the link to the issue related to that particular task.

Community Bonding Period

- Read and analyse the code closely. Understand how each existing feature is implemented.
- Make necessary preparations after discussing with the mentors and make necessary changes in the plan (if any).
- Add tasks to Github milestone trackers
- Add a **`ReadMe`** file to the project repo to help other developers to contribute.
- Learn more about Administrate and Slim template engine

Week 1 - 23rd to 28th May : Authentication Sprint

- Travis Integration
    - For building and testing code before it is merged into the main branch.
    - A **`travis.yml`** file will be added to the root directory of the project. It will contain the commands to test each commit before it is merged with the main branch. GitHub has webhooks to trigger this tests, whose logs will appear at travis.org.
- Integration with CodeClimate
    - For checking the code against certain rules for duplicity, security, complexity, etc. As the project is in the initial stage, this will help us maintain certain coding standards and will be beneficial in long run.
- Add Google+ authentication
    - Users will have an *alternate* option to signup with Google+.
    - It will be useful for Google Calendar and Places APIs.
    - **`Omniauth`** gem will be used to provide this authentication.
- Enable Admin Registration
    - Restrict admin access to users.
    - Make a super admin which can access all other Users and give admins rights. Such super admin will be seeded into the application during initial setup.
    - **`Is_admin`** can be replaced by **`role_id`** to distinguish between super-admin, admin and a simply user.

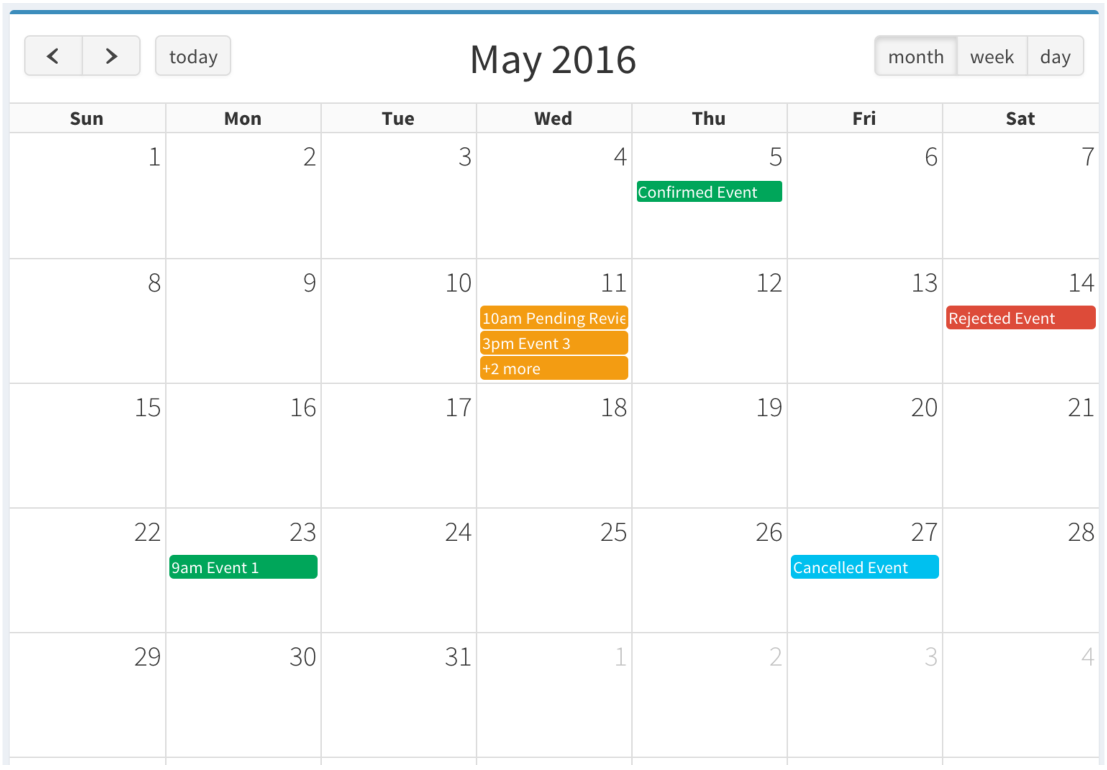
Week 2 - 30th May to 4th June : Association Sprint

- Establish Many to Many Relationship between **`Admin`** and **`Location.`** Presently admin is associated with a Nook instead of locations. Practically, an admin can be related to one or more locations and a location can have one or more admins.
    - A joint table will be created to persist the association data like **`admin_id`** and **`location_id`** per tuple.

    - Reservations shown on the admin panel should be scoped to the locations the admin is related to. This can be implemented by overriding `find_resource` method in Admin's reservation controller or adding custom scopes in the `User` model.
- Allow user to favourite nooks and locations ⇧
    - Two separate joint tables will be created for storing these favorites association.
    - A `My favourites` page will be accessible by the user from the settings' drop down at the top-right corner of the main page. The will have two partitions or two tabs listing his/her favourites nooks and locations.

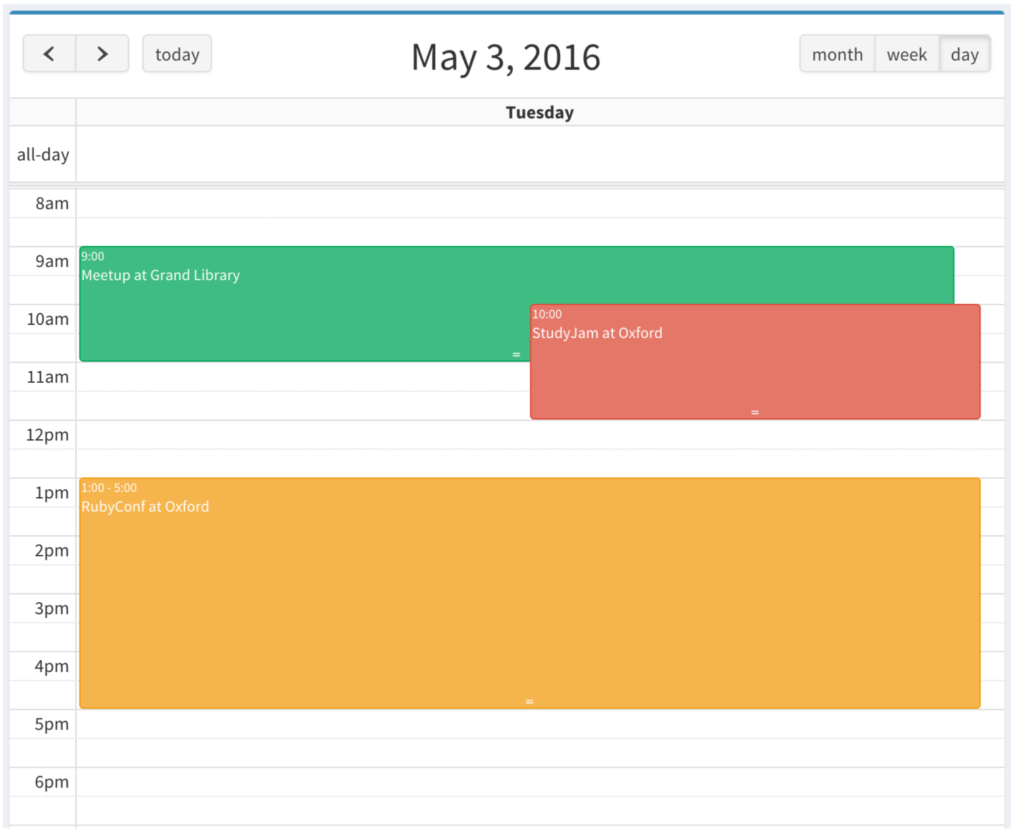
Week 3 - 6th to 11th June : Admin Calendar Sprint

- A separate Calendar dashboard will be added to admin panel ⇧
- To depict events on the calendar, we can have the following color scheme on the basis of their reservation status:
    - 🟧 Pending
    - 🟥 Rejected
    - 🟩 Confirmed
    - 🟦 Canceled by User
- The following is a mockup of Monthly view



Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5 Confirmed Event	6	7
8	9	10	11 10am Pending Revie 3pm Event 3 +2 more	12	13	14 Rejected Event
15	16	17	18	19	20	21
22	23 9am Event 1	24	25	26	27 Cancelled Event	28
29	30	31	1	2	3	4

- If a day has more than 3 events, a ‘x more events’ tab will appear instead of listing all of them. The admin can click on any day to get a closer look at that day's reservation. This will help him/her to make confirmation decision.
- This following is the mockup of the day view which appears on clicking a day on the monthly view:



- Clicking on any one of the events will trigger a modal with 2 choices in the form of buttons: Confirm or reject. This is because such updates are likely to be widely used by the admin.
- If there are no events, an admin can click at the time where a new reservation has to be made. A form for creating a reservation will appear with date and time auto filled. The event period might be assumed of a certain length for auto-filling.
- FullCanlendar will be used for the implementation. It is jQuery-based open source plugin which will allow adding a customized calendar in our app.

Week 4 - 13th to 18th June : User Calendar Sprint

- Allow a user to make his/her reservation request public. Only a subset of these attributes like status, time and places will be revealed to other people. ⇪
- To preserve the momentum, this sprint will add a calendar view for the user. It has two benefits:
    - The requester can plan and make a reservation request after checking the calendar and finding an available slot.
    - A user can attract people to join for an event by keeping it public.

Week 5 - 20th to 25th June : Mid-Term Evaluation

- Make MidTerm progress report
- Refactoring the code if necessary
- Documentation and updating the wiki

Week 6 - 27th June to 2nd July : Welcome Dashboard Sprint

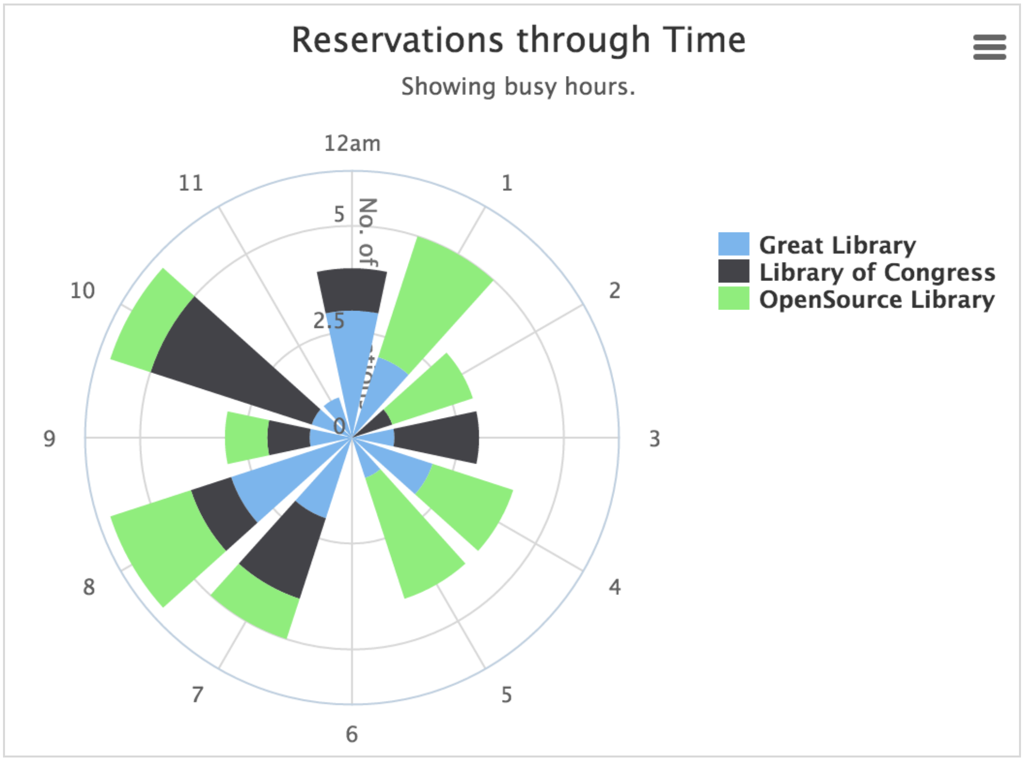
- Add One-click functionality which tells whether or not a nook is available for a reservation request. A pre-loaded status is not useful here as another admin could have booked the nook after the dashboard was loaded by the first admin. Another possibility is that the same admin accepts one of the many overlapping requests, but all of them are loaded on the same page with 'available' status.
- This will save a lot of time and manual efforts of going through the available slots. The request and response will be in AJAX and JSON form respectively. ⇪
- Add Welcome Dashboard to the admin panel. This prioritizes the display of the information for which the admin usually visit the application ⇪.
    - The page will have 3 counters horizontally arranged in a row. Clicking on each will respectively show:

        - Reservations requests which are not yet reviewed
        - Confirmed reservations recently canceled by the users. So, that the admin knows which slots are free again.
        - "Today's Reservations" or "Upcoming Reservations”
    - Each list will be loaded using AJAX.
    - Reservations will have actions like Check Availability and Confirm or reject, according to their status.

Week 7 and 8 - 4th to 16th July : Reports Dashboard Sprint

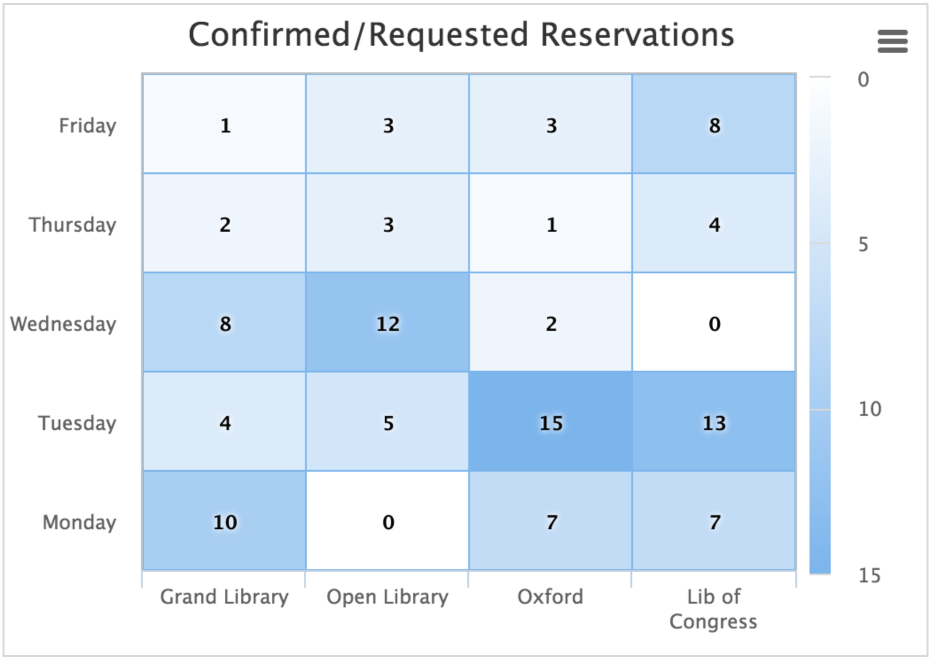
Some metrics and statistics can be very important for the admin. It can help admin gain insights on reservation demands and provide assistance to users on requesting reservations. Sometimes, he/she might also need to submit reports to the venue facilitator. A Reports dashboard will be added to the admin panel with the following visualizations ⇪:

- Reservations with different statuses between 2 dates: Admin will be able to generate a PDF report.
- Times for which majority of the reservations are requested for: A Wind Rose chart will be used show the reservations made for different locations against the time, imitating a clock. An example is given on next page.



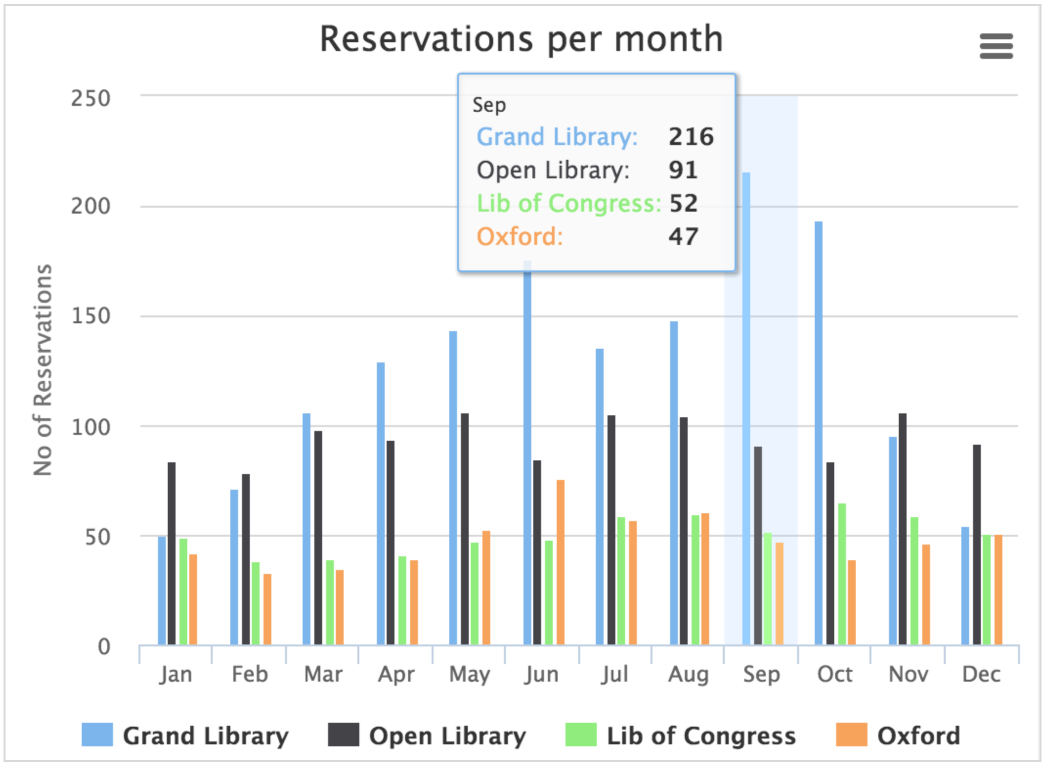
- Two such clock graphs can cover 24 hours of the day.

- Busy days in the week: A heat map can give us the comparison between no. of reservations on different days of the week.



	Grand Library	Open Library	Oxford	Lib of Congress
Friday	1	3	3	8
Thursday	2	3	1	4
Wednesday	8	12	2	0
Tuesday	4	5	15	13
Monday	10	0	7	7

- Most busy months for different libraries: A simple bar or line graph to show reservation during each month. It depicts the progress made during the year and highlights the months with high demand. For example:



- A filter on the top of this dashboard will allow admin to have a specific look at statistics based on one or more of:
    - Locations
    - Nooks
    - Reservation statuses
- The purpose of three small horizontal lines on right-top corner of every image is to facilitate the following:
    - Print the graph
    - Save as different formats like PNG, JPEG, PDF and SVG.
- Highcharts will be used to produce such visualizations.
- APIs will be set up to serve these statistics to the browser, so that the same APIs can be used for creating custom applications too.

Week 9 - 18th to 23th July : Location Sprint

- Integration with Google Map APIs for collecting library's location ⇪: When an admin fills up the form for adding a new location, he/she need to drag a pointer on the map to the library's geo-location.
- Two attributes will be added to the **`Location`** model for storing latitude and longitude. Alternatively, a key-value pair can be used and stored in one column. This is now supported by PostgreSQL.
- When searching nooks, the related locations of resultant nooks will be marked on a Google map. This will help the user to get an idea of the distance. ⇪

Week 10 - 25th to 30th July : Time Sprint

- The timestamps shown on the application will be personalised to the user’s timezone.
  - Two types of time zones are used in the applications:
    - Configured by default and can be accessed by **`config.time_zone`**. The data will be stored in the database in this timezone.
    - User’s timezone can be accessed by **`current_user.time_zone`** and be used in the UI. After the submission of the form, and before it is entered in the database it will be converted into the **`config.time_zone.`**
  - More information can be found here.
- Presently Nook can be open when it’s location is closed. Such erroneous scheduling will be restricted by modifying **`OpenSchedule`** model.
- As time is the trickiest element, this sprint can further be used to recover any lags in the schedule (if any).

Week 11 - 1st to 6th August : Notification Sprint

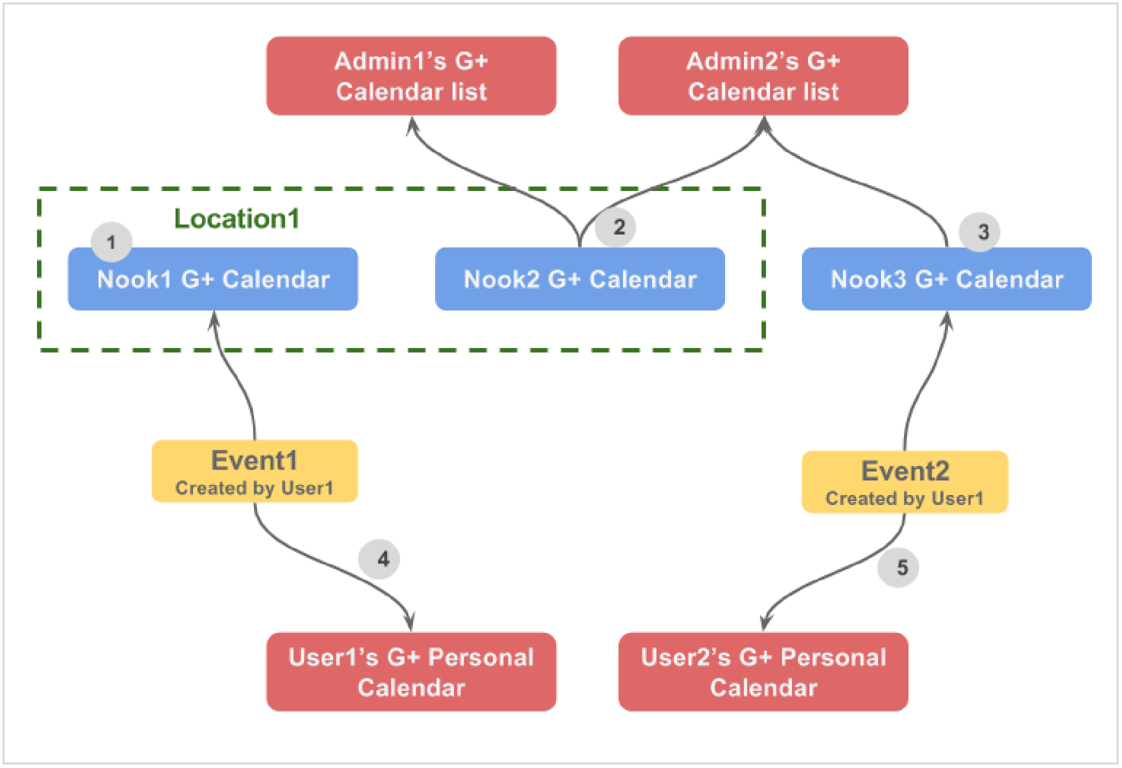
- A User notifications page will be added ⇪. The user will be notified when his/her reservation status is updated by admin.
- A **`Notification`** model will be created with join table for persisting many to one association between **`Notification`** and **`User.`**
    - All the notifications will be entered into the database and shown to the user in next login. A small counter at the right-top nav-bar will attract user's attention. Clicking on it will open a modal with the list of notification.
    - For every notification related to reservation status update, a mail will be sent to the user. The user will have the option to turn this off from the settings. An extra attribute in the User model will be used to check this subscription.
    - The Notification will have an **`action`** attribute signifying its cause.
- This will include designing mailing templates and setting up the mailers.

Week 12 - 8th to 13th August : Integration Sprint

This sprint will aim towards Integrating Google Calendar (GC) and Contact APIs ⇪. This will help in leveraging its notification system. The admin will be able to have a look at the confirmed events along with his/her personal events. An add-to-my-calendar functionality will be available to patrons and admins who has authenticated with Google+.
Here is a scenario:

- Admin1 and Admin2 are associated with Location1. Admin2 is also associated with another Location2.
- Location1 has Nook1 and Nook2. Location2 has Nook3.
- There are two users: User1 uses our application. User2 does not.
- User1 creates 2 reservations. Both are confirmed.

Refer the diagram on the next page to get how GC can be integrated. For this part, calendar means Google Calendar.



1. Every Google user have a calendar list where he can add personal calendars as well as other public/permitted calendars. In our case, every Nook will have its own calendar, primarily attached to the Admin who created the nook, i.e. this calendar will be added to his/her GC list. Other admins may or may not add this calendar to their GC list. If possible, we can have the calendar detachable from the owner too.
2. Many admins can add a nook's calendar to their calendar list as shown in the diagram.
3. As an admin could be associated with many locations, he/she can add calendars of nooks related to other locations too.
4. For a confirmed Event1, the requestor User1 will able to add it to its personal Google calendar. This will be done on demand, for every reservation separately. We can have a common sync functionality too.
5. A confirmed Event2 is created by User1 but is not added to his own personal calendar. However when he invites User2 (using Google Contacts APIs), the latter

accept it and Event2 gets added to his/her calendar. This is an incremental goal and can be implemented only if time permits.

6. Delete the GC event if the status is changed back from confirmed to any other. This is because the GC events are always assumed to be confirmed.
7. A user can opt for pre-event alerts (emails, pop-ups and SMS) from Google.
8. Update the GC event information like title and description when an event is updated on our app.

15th August Onwards

- Write integration tests to check if all the features developed are orchestrating
- Refactor and modify the old code according to the results obtained through CodeClimate analysis

Additional tasks

There are some tasks which can be a sub part of the main tasks described above. The followings are Ad Hoc and additional/optional tasks I would like to take up:

- Ask for Library's emergency contact info from admin ⇧: A string attribute will be added to `Location` model. Admin will add the editable contact info during its creation and can update later on. This information will be available to all the users.
- Allow a user to cancel a confirmed reservation until x days of the event ⇧: An integer attribute will be added to `Location` model depicting the number of days before which the cancellation requests are closed. It will be set and updated by admin.
- Using glyph-icons for various action buttons in the admin panel. ⇧
- Preventing user to hit certain URLs endpoints which he/she is not permitted of. ⇧
- Client and server side validation of URL input fields. ⇧
- Providing a reason for rejection of a request. ⇧

Additional Information

- The summary of the timeline presented above is documented on this Trello board.
- The brainstorming of the ideas with the mentors happened on this Google Doc.
- The code of the project is hosted on this GitHub repo.
- The issues reported by me during the planning can be found here.