Project Artifacts - MunchEase

Team: Chandra’s Angels

<table>
<thead>
<tr>
<th>Member</th>
<th>Email</th>
<th>Github</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jeffrey Tellew - Lead</td>
<td><a href="mailto:jefftellew@gmail.com">jefftellew@gmail.com</a></td>
<td>jefftellew</td>
</tr>
<tr>
<td>Austin Quinn</td>
<td><a href="mailto:austinquinn@ucsb.edu">austinquinn@ucsb.edu</a></td>
<td>austinquinn</td>
</tr>
<tr>
<td>Anthony Hall</td>
<td><a href="mailto:anthonylla.hall@gmail.com">anthonylla.hall@gmail.com</a></td>
<td>ahl99</td>
</tr>
<tr>
<td>Suds Kannan</td>
<td><a href="mailto:sudarshankannan99@gmail.com">sudarshankannan99@gmail.com</a></td>
<td>sudarshankannan</td>
</tr>
<tr>
<td>Yagnya Patel</td>
<td><a href="mailto:yagnyapatel@gmail.com">yagnyapatel@gmail.com</a></td>
<td>yagnyapatel</td>
</tr>
</tbody>
</table>

Relevant Links

GitHub: https://github.com/yagnyaPatel/Chandras-Angels
Trello: https://trello.com/b/AXdpdLP6/chandras-angels
Travis-CI: https://travis-ci.org/yagnyaPatel/Chandras-Angels
Slack: https://cs48s19.slack.com

Presentation: Chandras-Angels-presentation.pdf
Presentation skit: https://vimeo.com/338586420 (password: cs48)
Demonstration video: https://vimeo.com/339914219 (password: cs48)
Burndown Chart

SPRINT 1: To Do, In Progress, Ready for Testing and Done

SPRINT 2: To Do, In Progress, Ready for Testing and Done
About our project

What problem are we solving?
Oftentimes when spending time with friends, we find ourselves at a standstill, unable to determine where exactly to eat. One friend might suggest finding a Chinese place, another strongly advocates going to In-N-Out, and three other folks might be spitting out wildly differing alternatives. The point is, by the time you get everyone to come together and determine the group’s dietary course of action, you’ve spent about 15 or more minutes arguing uselessly. Furthermore, some of the more outspoken individuals may get more of a say and some of the quieter ones may be left voiceless. Regardless, the whole process is inefficient and oftentimes unfair.

Why is the problem important?
When people gather together to spend time with each other, they don’t want to waste time bickering over where to eat. The importance of this issue is as significant as the importance of time itself. As our society becomes increasingly efficient, people are constantly looking for new ways to make their lives more efficient. Our app aims to cut down on the amount of time spent
deciding where to eat, thus allowing people to spend more of their time enjoying each others’ company. Additionally, it is important to give everyone an equal playing field in terms of the value of their opinion. Something as simple as contributing to the decision of where to eat should not have to rely on someone’s social skills or persuasive abilities.

How is the problem solved today?

Currently there are a number of solutions on the market to solve the question of where to eat (Yelp, Zomato, OpenTable to name a few). Most of these are for individuals and do nothing to solve the actual problem of group decision making. There is an app, Ness, that is similar to these but has additional features that multiple users can use to get group restaurant recommendations based on a combination of their individual preferences in their profiles. However, this does not solve any issues regarding conflicts where different people are set on different places to eat. There is also a more general purpose app called VoteUp that allows groups to be created where users submit questions and answers that all members can vote on. This can tackle group decision making, but since all options must be inputted manually a pre-existing knowledge of what the options are is required. It would be much more convenient to be able to look up local restaurants and conveniently have links to provide further information about them (like ratings, menus, price range, etc.). We were not able to find any existing solutions that tackle both sides of this issue.

Outcome

Our project outcome is an Android application that allows a user to create a party that other users can join with a unique access code. Users can then search for restaurants with a search bar that displays restaurants pulled from the Yelp API. Restaurants can then be added to the party list, where party members can vote for their favorites. Links to restaurants’ Yelp pages are also available in the party list, so that further information is available to the party members to help in the voting process.

Our app simplifies the entire process of deciding where to eat in a group and equalizes the issue by having a democratic voting system. Thus, we save users both time and headache by creating an easy-to-use app that offers our solution to this common problem.

Creating Our Solution

Implementation Platform and Technologies

The Android application frontend will be built using Java and various associated libraries on Android studio. To obtain data (customer reviews, star ratings, price, etc.) about suggested
establishments, we will use the Yelp fusion API. Firebase will be used as the backend to store group, voting, and idea suggestion information, as well as to relay information between devices.

Project Milestones

First Sprint (4/15 - 4/26)

Milestone Goal: Users in a party can search for and remotely add restaurants to a list
  ● Apply the Yelp Fusion API to the Android frontend to search for and select songs, create a list of preferences stored in the Firebase backend.

Second Sprint (4/29 - 5/10)

Milestone Goal: Users can vote for their preferred restaurants
  ● Create a voting interface that updates the tally on the host device

Third Sprint (5/13 - 5/24)

Milestone Goal: A user can host a new group and other users can join via add code.
  ● Allow the creation of multiple independent “party” instances in the backend.
  ● Test multiple instances of “parties” and member devices.
Architecture Overview

- Yelp API
  - Restaurant queries
  - Restaurant information

- Database
  - Upvotes and downvotes
  - New restaurant suggestions
  - Joining and leaving party

- User
  - Current party information
### Detailed Design

#### UML Diagram

**Restaurant**

- alias: String
- Name: String
- url: String
- rating: String
- reviewCount: int
- displayAddress: String
- latitude: double
- longitude: double
- price: String
- schedule: RestaurantSchedule

- + accessors for all private members
- + parseFromJson()
- + convertToJson()
- + updateVoteCount()

**RestaurantSchedule**

- isOpen: boolean
- daySchedules: DaySchedule[7]

- + isOpen()
- + getOpeningTime() // of today
- + getClosingTime() // of today

**DaySchedule**

- day: string
- start: string
- end: string
- isOvernight: boolean

- + accessors for all private members
<table>
<thead>
<tr>
<th>Party</th>
</tr>
</thead>
</table>
| - ID: String  
- host: String  
- members: String[]  
- restaurants: Restaurant[] |
| - + addMember()  
- + addRestaurant()  
- + openMap() |

```
    Party
     ↓
     ↑ 0..*  1
  Restaurant Schedule  
     ↓
     ↑ 7
  Day Schedule
```

```
Restaurant Schedule
  ↓
  ↑ 1
Restaurant
```

```
Party
  ↓
  ↑ 0..*
Restaurant
```

```
Restaurant
  ↓
  ↑ 1
Restaurant Schedule
```

```
Day Schedule
  ↓
  ↑ 7
Restaurant Schedule
```

```
Restaurant Schedule
  ↓
  ↑ 1
Day Schedule
```

```
Restaurant
  ↓
  ↑ 1
Restaurant Schedule
```

```
Day Schedule
  ↓
  ↑ 7
Restaurant Schedule
```

```
Restaurant Schedule
  ↓
  ↑ 1
Day Schedule
```

```
Party
  ↓
  ↑ 0..*
Restaurant
```

```
Restaurant
  ↓
  ↑ 1
Restaurant Schedule
```

```
Restaurant Schedule
  ↓
  ↑ 1
Day Schedule
```

```
Day Schedule
  ↓
  ↑ 7
Restaurant Schedule
```

```
Restaurant Schedule
  ↓
  ↑ 1
Day Schedule
```

```
Party
  ↓
  ↑ 0..*
Restaurant
```

```
Restaurant
  ↓
  ↑ 1
Restaurant Schedule
```

```
Restaurant Schedule
  ↓
  ↑ 1
Day Schedule
```

```
Day Schedule
  ↓
  ↑ 7
Restaurant Schedule
```

```
Restaurant Schedule
  ↓
  ↑ 1
Day Schedule
```

```
Party
  ↓
  ↑ 0..*
Restaurant
```
Sequence Diagrams

p:Party

new(r)

r:Restaurant

getName()

name

upvote()

User

Opens app

Shows home screen

MunchEase

Taps “Create”

Creates new party, shows “Share” screen

Taps “Add Restaurant”

Opens SearchActivity
UI Mockups
Requirements

User Stories

1. **(1) As a User, I should be able to launch the app so that I can use it.**
2. **(2) As a Party Host, I can create a party so that people can join.**
3. **(3) As a User, I should be able to connect to a party so that I can be a party member.**
4. **(4) As a Party Member, I should be able to see the list of restaurants up for selection so that I can interact with them.**
5. **(5) As a Party Member, I should be able to search up restaurants to add in a search bar.**
6. **(6) As a Party Member, I can add restaurants to the list so that they can be voted on.**
7. **(7) As a Party Member, I should be able to vote on restaurants in the list so they are either more or less likely to be chosen.**
8. **(8) As a Party Host, I can specify a backup restaurant so that there will still be a default option if nobody suggests any other restaurants.**

Missing Features

1. **As a User, I should be able to search up nearby parties, see the restaurant list**
2. **As a User, I can upload my dietary preferences so that my search results will be more suitable for myself**
3. **As a Party Host, I can enable and disable downvoting to prevent toxic users from ruining the experience.**
4. **As a Party Host, I can set limits to how often a certain restaurant can be suggested in my parties.**
5. **As a Party Member, I can share the party via QR code so I can get others to join the party I'm in.**
6. Other missing features and ideas
   a. Use the phone’s actual GPS location for the center of the restaurant search, rather than the hard-coded coordinates on campus
   b. Display more restaurant info in-app, rather than navigating to the Yelp webpage
      i. There is already existing implementation for more restaurant info, particularly restaurant schedule information. However, it causes bugs in Firebase storage that we did not have time to further investigate.
   c. Map view for the restaurant list
   d. Profile stat tracking - could go lots of ways with this
The Development Experience

Challenges Faced

- Had to change projects two weeks into the quarter because another team was working on something very similar
- Had to learn databases (Firebase in particular) for the first time
- Had to learn how to implement Yelp API
- Many tasks relied on other tasks before they could be effectively worked on, resulting in a bit of a bottleneck at the beginning of the project
- Getting Android emulators to run on some old laptops was difficult
- Connecting the search bar input to the Yelp API “engine” was a challenge
- Learning how to effectively split tasks so everyone in the group is both comfortable working on things but learning at the same time
- Travis-CI was not very easy to learn how to work with
- Writing test cases in general was confusing until we really nailed down how we wanted the app to work behind the scenes

Retrospectives

1. Sprint 1
   a. The problem: Too many features rely on the app being set up in the first place, meaning that anyone working on the front-end had a ton of work to do and anyone working on other parts couldn’t do much.
   b. The solution: There is no good way around this, so we just cranked out the parts that needed to get done in order for progress to be made on the rest of the project.
   c. The result: This approach ended up working, and development proceeded after the first sprint with just a minor hiccup.

2. Sprint 2
   a. The problem: We spent a lot of time in Sprint 2 trying to get our heads wrapped around some of the new technologies we were using. Firebase, Yelp API, and Travis-CI were all challenging in their own ways.
   b. The solution: Again, we just had to power through this. There’s no way to get around learning the stuff that you need to learn, so we put in the time to get where we needed to be in order to really get the features we wanted done during the next sprint.
   c. The result: The challenges we faced in Sprint 2 show an obvious slowdown in the actual development of the app, but paid off later during Sprint 3 where we were able to make some incredible strides in our project.
3. **Sprint 3**

   a. **The problem:** In Sprint 3, we were trying to finish up some of the features we had slated in order to get our app where we wanted it to be for the demo. There was a lot that we wanted to do, but not all of it was possible to finish in time. We had to prioritize the features we wanted most.
   
   b. **The solution:** We had a few group meetings between the end of Sprint 2 and the start of Sprint 3 where we debated what we really wanted to have by the demo. Some of the features were chosen, and others weren’t -- many of those that we didn’t end up finishing can be found above in the “Missing Features” section.
   
   c. **The result:** During Sprint 3, we really did a lot of work to get the app where it needed to be. Since we had the knowledge from Sprint 2, we were able to get some great work done during Sprint 3. For example, we got the Yelp API working with a search bar, suggestions for the search bar, Yelp links from the restaurant list, separate parties with randomly generated codes, and a total UI overhaul.

**Commit History**

**Commits to Master:** [https://github.com/yagnyaPatel/Chandras-Angels/commits/master](https://github.com/yagnyaPatel/Chandras-Angels/commits/master)

**Contribution Page:** [https://github.com/yagnyaPatel/Chandras-Angels/graphs/contributors](https://github.com/yagnyaPatel/Chandras-Angels/graphs/contributors)

**Austin Quinn:** I didn’t have any commits to master because most of my work was done in other branches that were then implemented into other group members’ tasks. These tasks were then often committed by these other members in larger chunks to master, which included my code contributions, but didn’t have my ID attached to them. Additionally, I took on mostly larger tasks, like implementing the search bar and Yelp API, which were committed to the suds2 branch, and oftentimes only committed when we had made progress that didn’t cause major breaks, which reduced the frequency of commits. Lastly, I contributed considerably through pair programming with Jeff, Anthony, and Suds, and by debugging their code.

**Sudarshan Kannan:** Most of my commits were on the suds2 (yelp API and search bar) branch, rather than directly to master. In addition, we often utilised pair programming (I often partnered with Anthony), thus, we obviously pushed from that individual’s computer rather than our own. Thus oftentimes my ID was not attached to the commits. We also tended to commit larger chunks of code to master, thus resulting in infrequent pushes to master. In addition, most pushes tended to when we had code that functioned locally.

**Yagnya Patel:** My implementation of the MapView for Party lists was written but dependent on other components that weren’t completed until later in the project. This included setting up the Google Maps API and creating a window for the view based on longitude and latitude inputs I would get from the Yelp API returns. These contributions never made it to master due to the delay on the completion of the Yelp API implementation. To make sure I made a significant contribution, I dedicated my time to writing UI and unit test cases for the large backlog of tasks.
Test Suite

The Restaurant class consisted of accessors ("getters") and mutators ("setters") for its members. Several tests were written to test each of these methods. Along with the Restaurant class, the Party class was written a similar test suite. An additional class written for the Restaurant schedule was tested with an example output from Yelp since the API calls returned values dependent on time. Restaurant Schedule tests have not been tested for Yelp overnight hour listings. The Restaurant parser was also tested with this test output. UI elements were tested under ExampleInstrumentedTest.java.

All unit tests can be found under ./app/src/test/java/edu/ucsb/munchease(tests). The unit test files include:

- RestaurantUnitTests.java
- PartyUnitTests.java
- RestaurantParserTests.java
- RestaurantScheduleTests.java

The UI tests can be found under ./app/src/test/java/edu/ucsb/munchease(androidTest).

Appendix

1. App Interface
   a. Java
   b. Android Studio
2. Database
   a. Firebase Firestore
   b. Firebase Realtime Database
3. Restaurant Information
   a. Yelp API
   b. Google Maps
4. Development
   a. GitHub
   b. Travis-CI