Host-Server Route:

Implemented by Mira & Cheyan

1. The *verb*, *target*, and *body* of the request.
   POST / parties [partySchema], where partySchema is a JSON object describing a Party.
2. What does the HTTP request do, and what mock server method does it replace?
   The HTTP request is returning the party object which has all the information about the party and storing it into the database. The mock server method it replaces is the Host method.
3. What resources the HTTP request creates/modifies/etc.
   Creates a new /parties.
4. Who is authorized to use the request
   POST /parties[partySchema]. The body of the HTTP request contains a “Party Name” field, which contains the party host ID. The requester must have the same party host ID.

PartyInfo-Server Route:

Implemented by Alex and Cheyan

1. The *verb*, *target*, and *body* of the request.
   GET/ parties / :partyId where partyId is the ID of the party
2. What does the HTTP request do, and what mock server method does it replace?
   The HTTP request is returning a list of users attending, invited to, and who have declined to join the party, as well as a list of supplies requested for the party and information about the host of the party. This function replaces three calls to getInvitedData(), one call to getAuthorData() and one call to get PartyData().
3. What resources the HTTP request creates/modifies/etc.
   Does not modify anything
4. Who is authorized to use the request
   Any users who have partysmart accounts

Profile-Server Route:

Implemented by Alex

1. The *verb*, *target*, and *body* of the request.
   GET/ users / :id / profile
2. What does the HTTP request do, and what mock server method does it replace?
   The HTTP request is returning a list of party objects ordered by whether they happened previously or will happen in the future, and by whether the user is invited to, attending, or has declined the invitation. It also returns information about the friends of the user. This function replaces one call to getAuthorData() and an other call to getProfileParties().
3. What resources the HTTP request creates/modifies/etc.
   Does not modify anything
4. Who is authorized to use the request
   The user who makes the request must have the user id, id.

Add Friend-Server Route:

Implemented by Alex

1. The *verb*, *target*, and *body* of the request.
   POST/ users/ :id/ addfriend / :friendid
2. What does the HTTP request do, and what mock server method does it replace?
   The HTTP request is returning an updated user object representing the user with the friend whose id is friendid. This function is new - does not replace a mock server function.
3. What resources the HTTP request creates/modifies/etc.
   This modifies the user with the id, id. It adds :friendid to the user's friend list.
4. Who is authorized to use the request
   The user who makes the request must have the user id, id.

Remove Friend-Server Route:

Implemented by Alex

1. The *verb*, *target*, and *body* of the request.
   POST/ users/ :id/ removefriend/ :friendid
2. What does the HTTP request do, and what mock server method does it replace?
   The HTTP request is returning an updated user object representing the user with the friend whose id is id, without the user object in the friends list with the id friendid. This function is new - does not replace a mock server function.
3. What resources the HTTP request creates/modifies/etc.
   This modifies the user with the id, id. It removes :friendid from the user's friend list.
4. Who is authorized to use the request
   The user who makes the request must have the user id, id.

Search Users-Server Route:

Implemented by Alex

1. The *verb*, *target*, and *body* of the request.
   POST/ search/ :userid / users with a body which is a string.
2. What does the HTTP request do, and what mock server method does it replace?
   The HTTP request is returning an object which contains two lists, one with friends whose full name contain the string body and one with all users whose full name contains the string body. This function is new - does not replace a mock server function.
3. What resources the HTTP request creates/modifies/etc.
   Does not modify resources.
4. Who is authorized to use the request
   The user who makes the request must have the user id, id, and the body must be a string.

Update Invite List-Server Route:

Implemented by Alex

2. The *verb*, *target*, and *body* of the request.

   PUT / parties / :id / invited/ users - with a body which is a single list of all users invited to a party regardless of attendance status.

2. What does the HTTP request do, and what mock server method does it replace?

   The HTTP request is returning an object which contains updated attending, declined and invited lists for the party with id id. This function is new - does not replace a mock server function.

3. What resources the HTTP request creates/modifies/etc.

   Modifies the three lists for the party, removes all users not contained in the body list.

4. Who is authorized to use the request

   The user who makes the request must have the user id which is the same as the party host's id, and the body must be a list of users.

Complaint-Server-Route:

Implemented by Vlady(Vladislava) and Cheyan

3. The *verb*, *target*, and *body* of the request.

   "POST", "/nearby_parties" - with a body which sends the coordinates, longitude and latitude, of the requestor.Complaint schema is a JSON object that describes a complaint.

2. What does the HTTP request do, and what mock server method does it replace?

   The HTTP request takes in the coordinates that were sent and checks for nearby parties that could potentially cause the issue. This function replaces getComplaint().

3. What resources the HTTP request creates/modifies/etc.

   It sends back a list of all of the nearby parties.

4. Who is authorized to use the request

   It is anonymous so everybody is authorized to make a complaint.

Admin-Server-Route:

Implemented by Vlady(Vladislava) and Cheyan

4. The *verb*, *target*, and *body* of the request.

   "GET", "/admin" - with a body, which sends the token, we change the way the returned information is parsed.

2. What does the HTTP request do, and what mock server method does it replace?

   The HTTP request maps all of the information of an admin in the database and sends it back to the client. This function replaces getAdminInfo().

3. What resources the HTTP request creates/modifies/etc.

   It sends back admin information.

4. Who is authorized to use the request

   Only a host/admin can do this request because we want to keep the hosts information private.

Update Supply List-Server Route:

Implemented by Ed Fennessey

1. The *verb*, *target*, and *body* of the request.
   PUT, /parties/:id/supplies, body is a list of all registered supplies for the party as arrays which hold the supply ID and user ID of the person bringing the supply
2. What does the HTTP request do, and what mock server method does it replace?
   The request verifies the user is allowed to perform this action, determines what was deleted from the body sent to it, and returns an updated supply list. This was a missing feature from the last submission and does not replace any previous functions.
3. What resources the HTTP request creates/modifies/etc.?
   The request modifies the supplies attribute of PartyInfo.
4. Who is authorized to use the request?
   Right now only the host of a party can use this request.

Party-Status Route:

Implemented by Vlady(Vladislava) and Alex&Cheyan

5. The *verb*, *target*, and *body* of the request.
   "PUT", "/parties/:id/private_status", where we send if the party needs to be private or open i.e we pass down a String variable that holds the value of either “true” or “false”.
6. What does the HTTP request do, and what mock server method does it replace?
   The HTTP request returns the status of the specific party. The mock server methods it replaces are: setPartyPrivate(), setPartyOpen().
7. What resources the HTTP request creates/modifies/etc.
   It modifies the party object by changing its status.
8. Who is authorized to use the request
   Only admin/host users are allowed to change this information.

Individual User Route:

Implemented by Cheyan

9. The *verb*, *target*, and *body* of the request.
   GET/ users/:id
10. What does the HTTP request do, and what mock server method does it replace?
    The HTTP request is returning the database form of a user with basic information for friends. It replaces getAuthorData().
11. What resources the HTTP request creates/modifies/etc.
    It sends back user information.
12. Who is authorized to use the request
    The user themself

Individual User Parties Route:

Implemented by Cheyan

13. The *verb*, *target*, and *body* of the request.
    GET/ users/:id/parties
14. What does the HTTP request do, and what mock server method does it replace?
    The HTTP request is returning a list of parties for the user and their basic information
    What resources the HTTP request creates/modifies/etc.
    It sends back party information.
15. Who is authorized to use the request
    The user themself

Individual Parties Route:

Implemented by Cheyan

16. The *verb*, *target*, and *body* of the request.
    GET/ parties/:id
17. What does the HTTP request do, and what mock server method does it replace?
    The HTTP request is detailed information for a specific party
    What resources the HTTP request creates/modifies/etc.
    It sends back party information.
18. Who is authorized to use the request
    Any user who is related to the party, e.g. in invited, declined, attending, or host

Nearby Parties Route:

Implemented by Cheyan

19. The *verb*, *target*, and *body* of the request.
    GET/ nearby_parties
    What does the HTTP request do, and what mock server method does it replace?
    The HTTP request returns a list of parties that is within a certain radius of the user, I believe it's 0.25 miles, but the metric is a little messed up
    What resources the HTTP request creates/modifies/etc.
    It sends back basic party info
    Who is authorized to use the request
    Anyone

Lingering Bugs / Issues / Dropped Features

- We are no longer allowing users to request to join a party - they must be invited by the host
- Deleting a supply from a party does not delete the correct item
- Google map has not been implemented

Additional Individual Contributions

Cheyan:

- Functionality for a variety of pages
- The messaging service, I know it's ugly, I know it's bad practice, but it works, sometimes

Alex:

- Functionality / front end of add/remove friend
- Functionality / front end of search friends/all users
- Functionality / front end of remove user from invited list

Ed:

- Functionality of switching between tabs in a party-info page
- Content of Supplies tab in party-info page
- Content of Complaints tab in party-info page

Vlady:

- Helped implemented Admin Route

Mira:

- Implemented Host-Server Route