Introduction and Purpose

SOCIAL CAMPERS

Social Campers application is designed to provide information about parks where there are camping facilities available. It also gives information about facilities in a campground and reviews of other users. It lets a user write a review, invite friends, or send private messages to a friend who is already using the app. It also has option to post user's review on their timeline for friends to view and comment.
URL to publish through facebook: https://apps.facebook.com/social-campers

Code description

➢ Java classes

1. **Park.java class**

   This is the datastore park entity model. It gets park entity property values from CreateParkServlet which in-turn gets the values from create_park.jsp and sets it.

2. **Campground.java class**

   This is the datastore campground entity model. It gets campground entity property values from CreateCampgroundServlet which in-turn gets the values from create_campground.jsp and sets it.

3. **Review.java class**

   This is the datastore review entity model. It gets review entity property values from CreateReviewServlet which in-turn gets the values from create_review.jsp and sets it.

4. **PMF.java**

   This is a Java API that lets us interact with the Google Datastore. Set the namespace to "Nandini.ye5386" (This is the namespace in Google datastore). This has jdoHelper imports and PersistenceManagerFactory imports from javax.jdo. This Java API is referred from https://cloud.google.com/appengine/docs/java/datastore/jdo/overview.

5. **StoreOperation.java**

   This class contains helper methods to store data entity into Google App Engine Datastore. This has methods to create new campground, park or review entity. Stores the values of each property into the datastore entities.

6. **DataRetriever.java**

   This class contains helper methods to retrieve information from Google App Engine Datastore. It has method to get list of parks from datastore, display park information and associated campgrounds for the park from datastore, gets campground information and associated reviews from the datastore. The methods make use of fetchCampground and fetchReviews to get the list of campgrounds and reviews. Gets property values from the datastore for particular entity.

7. **FacebookUtils.java:**

   This contains helper methods to access information of the user from facebook. It uses RESTFB API for java. This class is used to get the name of

logged in user by looking up access_token from request, get lists of friends who also use the app and allows the app to post message on user's timeline.

➢ Servlets

1. **CreateCampgroundServlet:**
   Fetches the user-entered data from the web page for campground and create a datastore entity.
2. **CreateParkServlet:**
   Fetches the user entered data from the web page for park and creates a datastore entity.
3. **CreateReviewServlet:**
   Fetches the user-entered data from the web page for review and creates a datastore entity.

➢ Html and jsp pages

1. **Index.jsp:**
   It uses Javascript API. This page imports FacebookUtils.java class. The index page checks whether the user has logged into facebook. If no, it sends the user to facebook login page. If the user has logged in and not given permissions then it shows the dialog box to get permissions. Once a user gives permissions, it takes the 'access_token' and passes it on to each page a user navigates.
2. **Create_park.jsp:**
   This page is currently visible only to the app admin. This page is used to add enities into the datastore, this is mapped to CreatePark servlet and this is mapped to StoreOperation.java class. The intention is to add only correct information from reliable sources.
3. **Create_campground.jsp:**
   This page is currently visible only to the app admin. This page is used to add enities into the datastore, this is mapped to CreateCampground servlet and this is mapped to StoreOperation.java class. The intention is to add only correct information from reliable sources.
4. **Create_review.jsp:**
   Lets the user enter review information for a campground. This page is used to add enities into the datastore, this is mapped to CreateReview servlet and this is mapped to StoreOperation.java class. It also has a 'Post on timeline' check box. This will work only for developers because the app is not yet reviewed by facebook. That is a restriction from facebook. When a user clicks on 'Submit Review' button it takes the text, rating, username and date from the form and displays them on 'Display Campground' page.
5. **List_parks.jsp:**
   Retrieves data from DataRetriever.listParks() and prints the list of park names. It has park name as link, when a user clicks on the link it takes to the 'Display Park' page.
6. **Display_park.jsp:**
   Gets parkId from previous page that is List_parks.jsp and prints all the property values of the selected park. List the campgrounds associated with the park with a link to click on the campground name.

7. **Display_campground.jsp:**

   Gets the campgroundId from previous page that is Display_park.jsp and displays all the campground property values and also the review lists about the campground with user, rating, review text and date. When a user clicks on 'Write a review' button it takes to the Create_review.jsp page and it also passes the access_token as a hidden text in-order to fetch the user name. The 'Recommend to a friend who uses the app' button takes the user to 'Recommend Campground' page. This also carries the access_token as hidden text.

8. **Recommend_campground.jsp:**

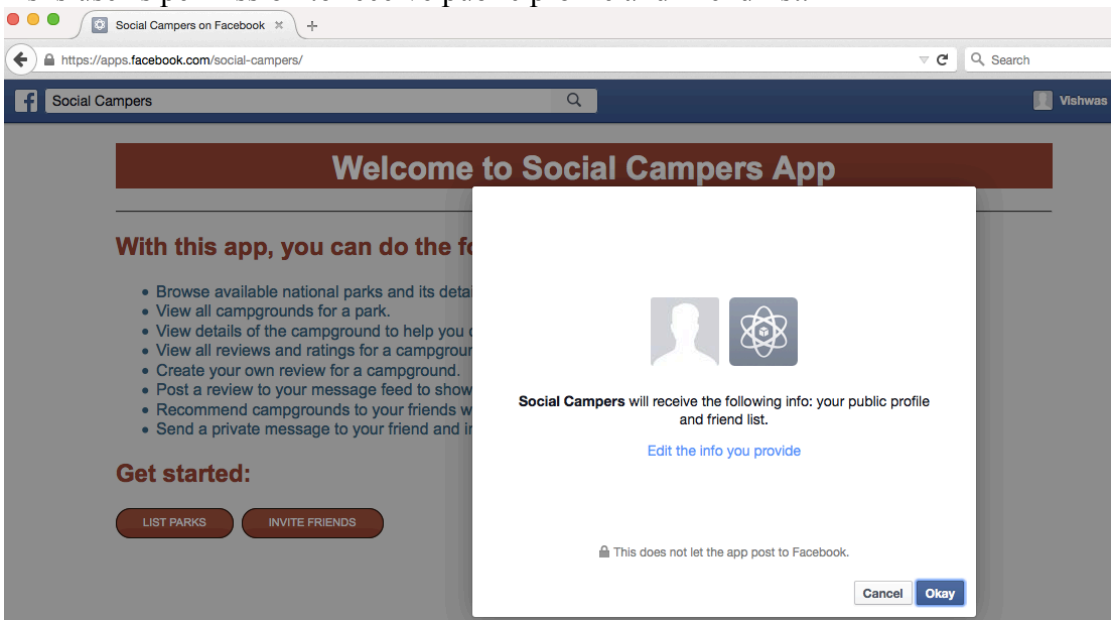
   This page displays list of user's friends who use the app with a checkbox in front of their name. When the user checks on the box and click on 'Recommend' the page invokes send method from FB.ui and it sends the URL of particular campground. If there are no friends who use this app then it says 'Sorry no friends are using this app' and shows 'Invite' button that invokes 'appRequest()' function which in-turn uses 'FB.ui 'apprequests'' method to send an app invite.

9. **Social_campers.css:**

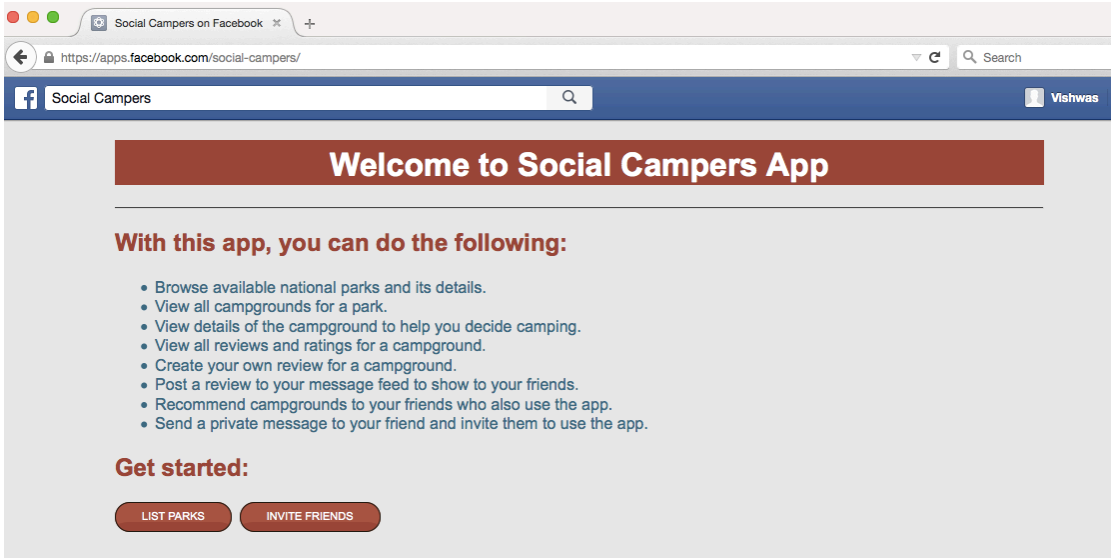
   This has styles for html tags like <body>, <h1>, <h2>, <h3>, <table>, <p>, , <textarea> and <button>/<submit>.

Demonstration of application working

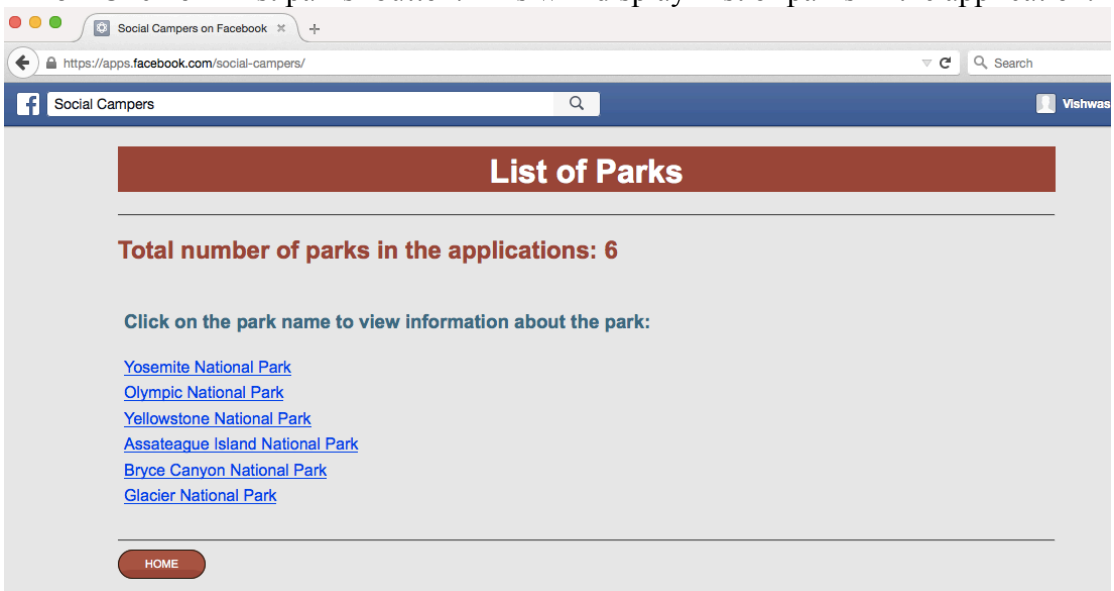
- 1st screen when the user logs in for the first time using the app URL. Asks user's permission to receive public profile and friend list.



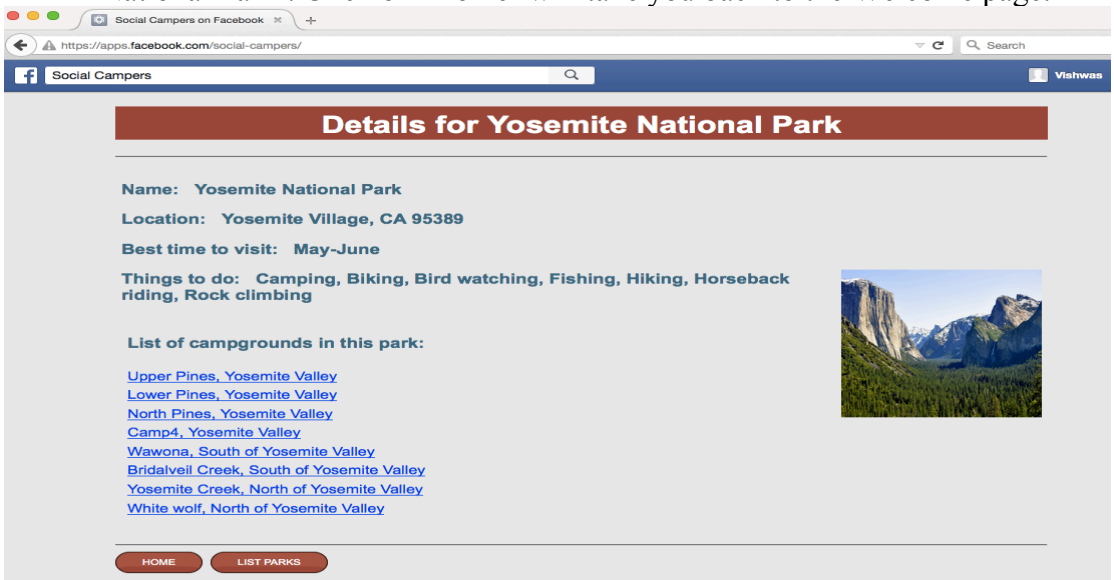
- Click on 'Okay' and if you are a developer of the app then it asks for permission to post on timeline. After these 2 boxes it will display the Welcome page.



- Click on 'List parks' button. This will display List of parks in the application.

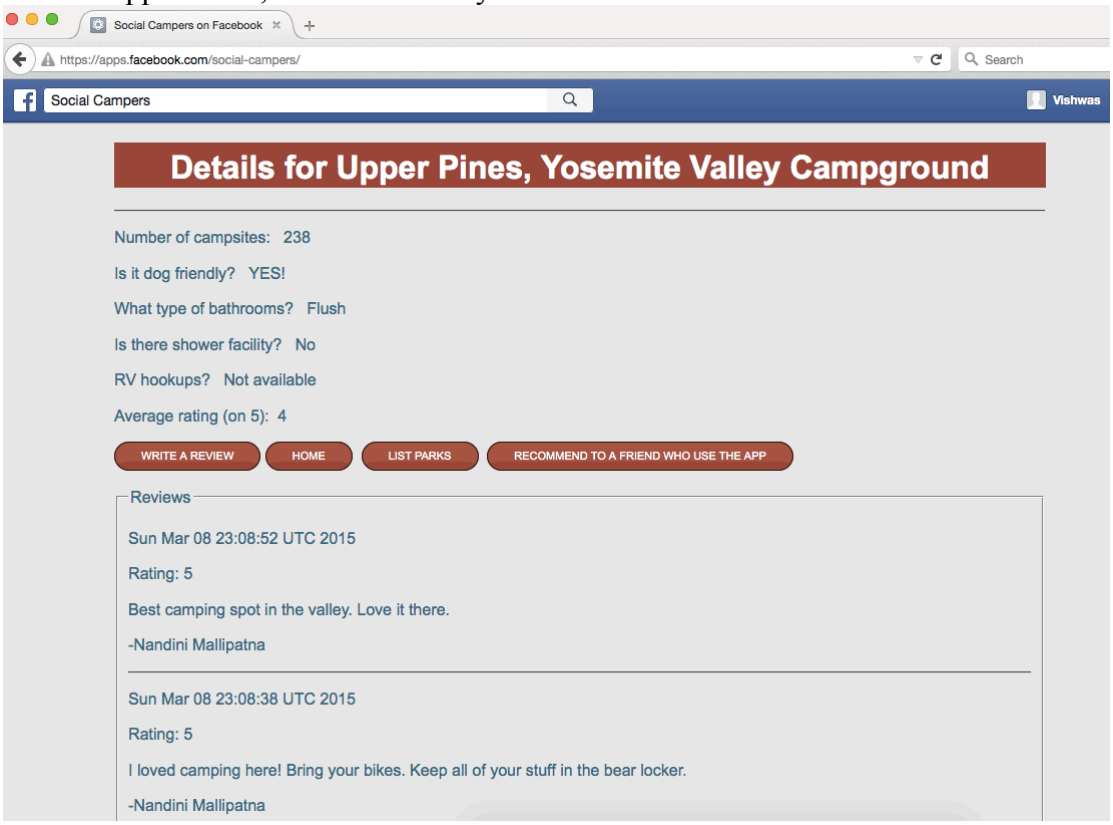


- Click on any one of the link. For demonstration I clicked on 'Yosemite National Park'. Click on 'Home' will take you back to the Welcome page.

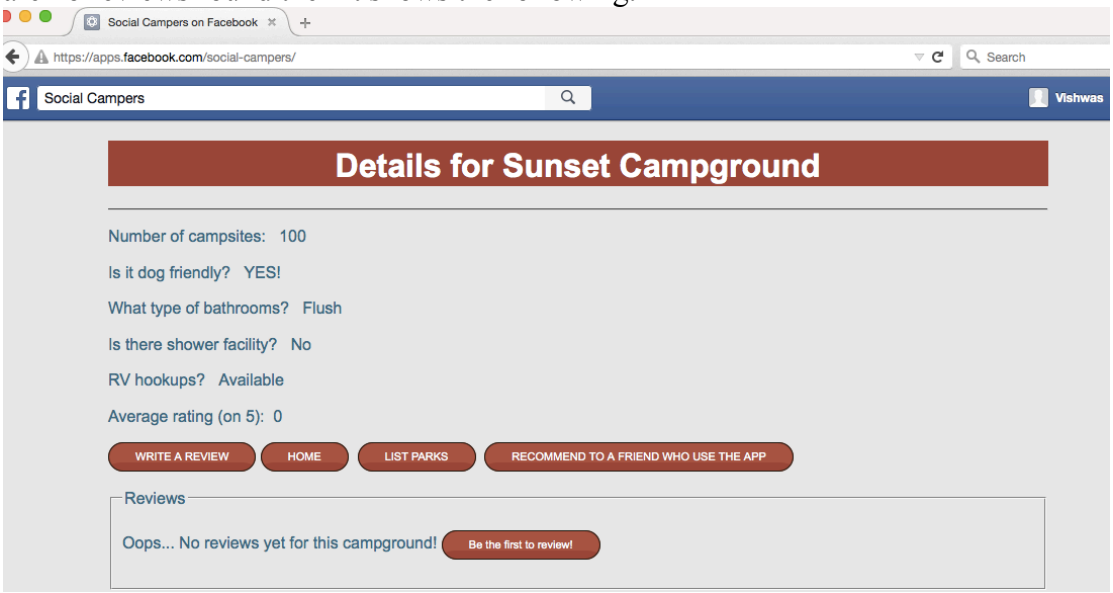


This above page shows the details for Yosemite National Park and also the list of campgrounds available in the park. Click on any one of the campground names will take to the 'Campground Details' page, click on 'Home' will take you to the 'Welcome' page and click on 'List Parks' takes you to 'Parks list' page.

- For demonstration purpose I clicked on the first campground name, which is 'upper Pines, Yosemite Valley'.

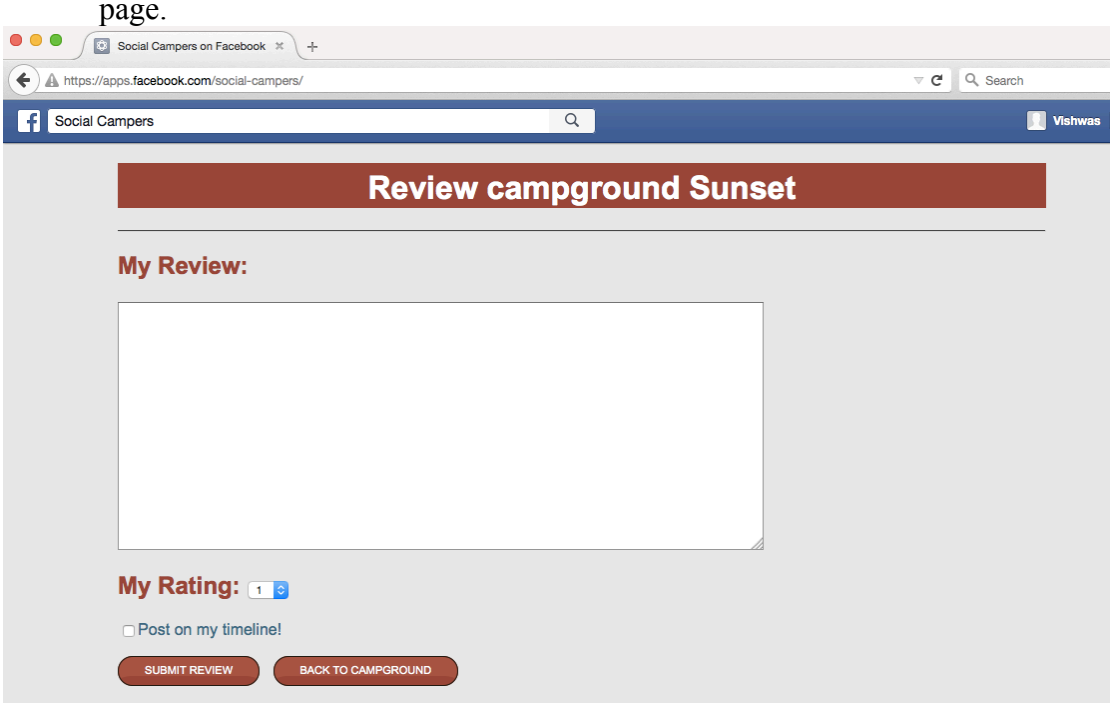


This will display details of campground and also the list of reviews with the reviewer's name, date etc. The review is sorted based on time (newest first). If there are no reviews found then it shows the following.

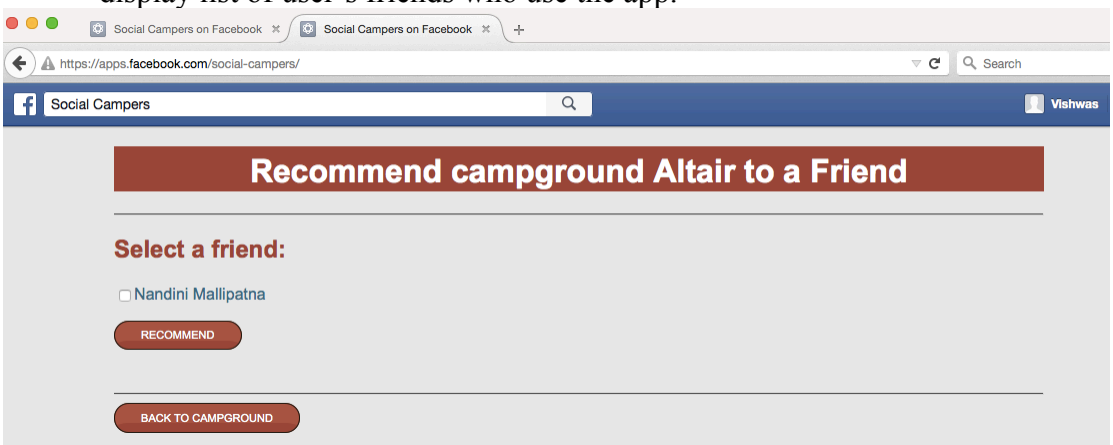


'Be the first to review!' button takes you to 'Write review' page. 'Recommend to a friend who use the app' takes you to 'Recommend to a friend' page.

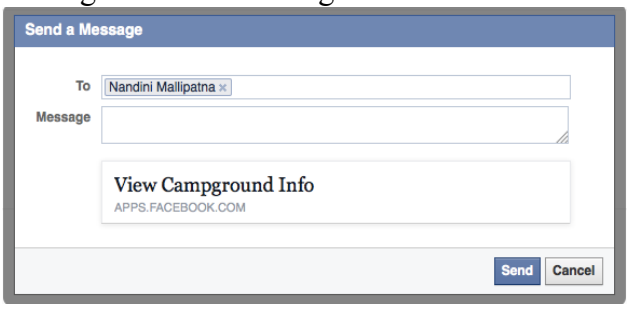
- When you click on ‘Write a review’ button it takes you to ‘Write review’ page.



- When you click on ‘Recommend to a friend who use the app’ button it will display list of user’s friends who use the app.

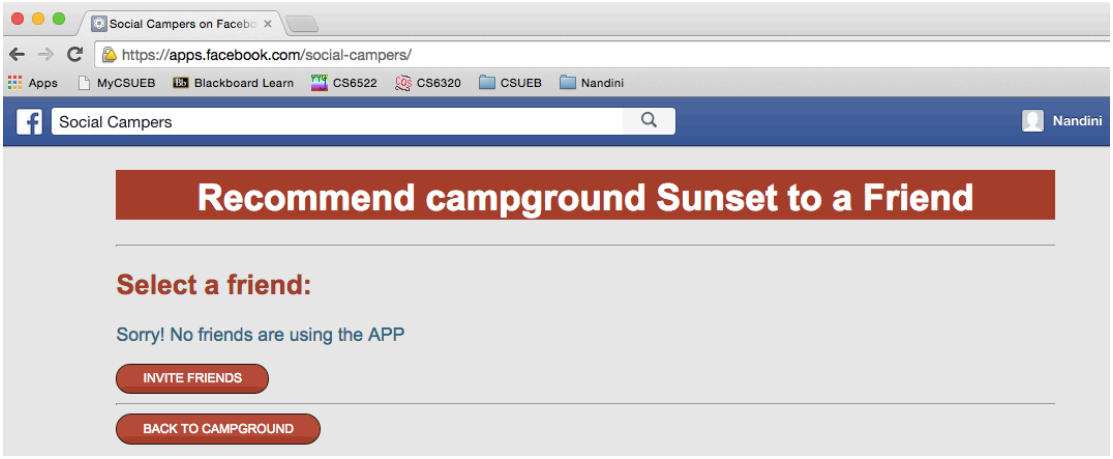


Check ‘Nandini Mallipatna’ and click on ‘Recommend’. This will open ‘Send a message’ facebook dialog box. You can also enter a message and click on ‘Send’.



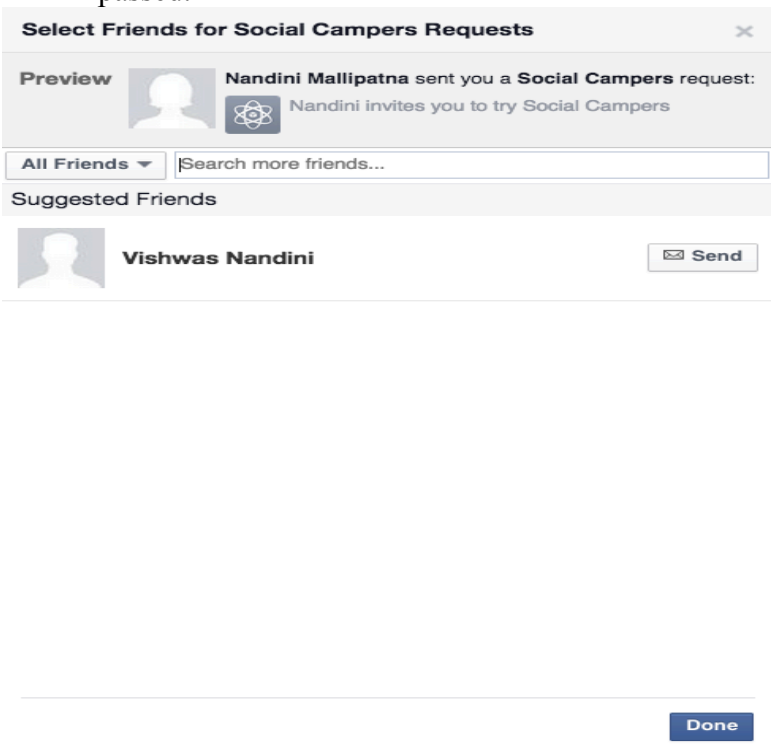
Nandini Mallipatna will receive a message with text containing URL of the campground. When clicked on that, it will take Nandini to the details of campground page.

If there are no friends using the app then it shows the page below and a button to invite friends.



- In 'Review campground' page if a user checks on 'Post on my timeline' and hits 'Submit Review' the review will be displayed on user's timeline as a post. (Only admin and developers can give permission to post on timeline. This is a restriction from facebook, as the application has not yet been reviewed from facebook.)

- Click on 'Invite friends' button accesses your friend's list and lets you add your friends and send them an invitation to use the app. The app URL will be passed.



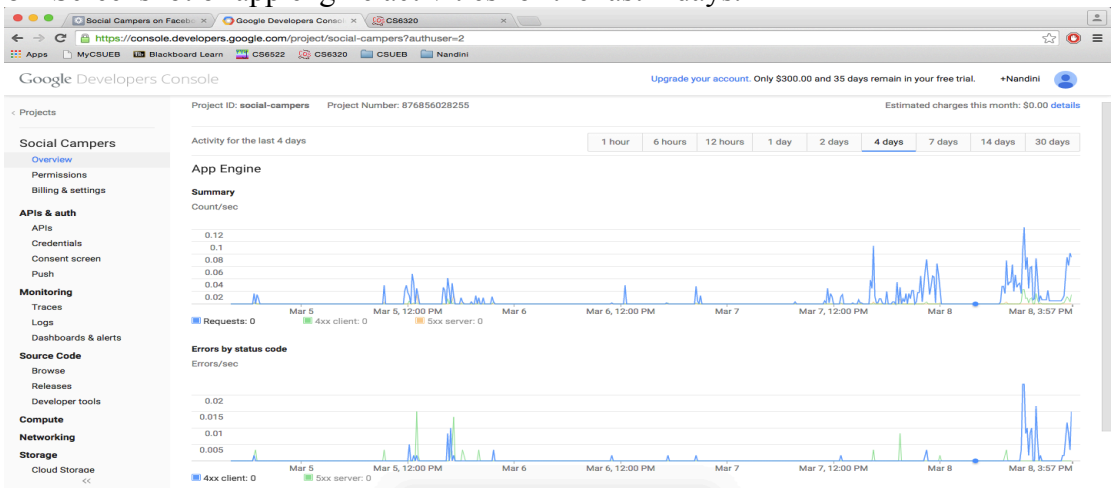
Click on the notification, and it will take you to the welcome page where it asks the user to give permission to access public profile and friend list.

Monetization

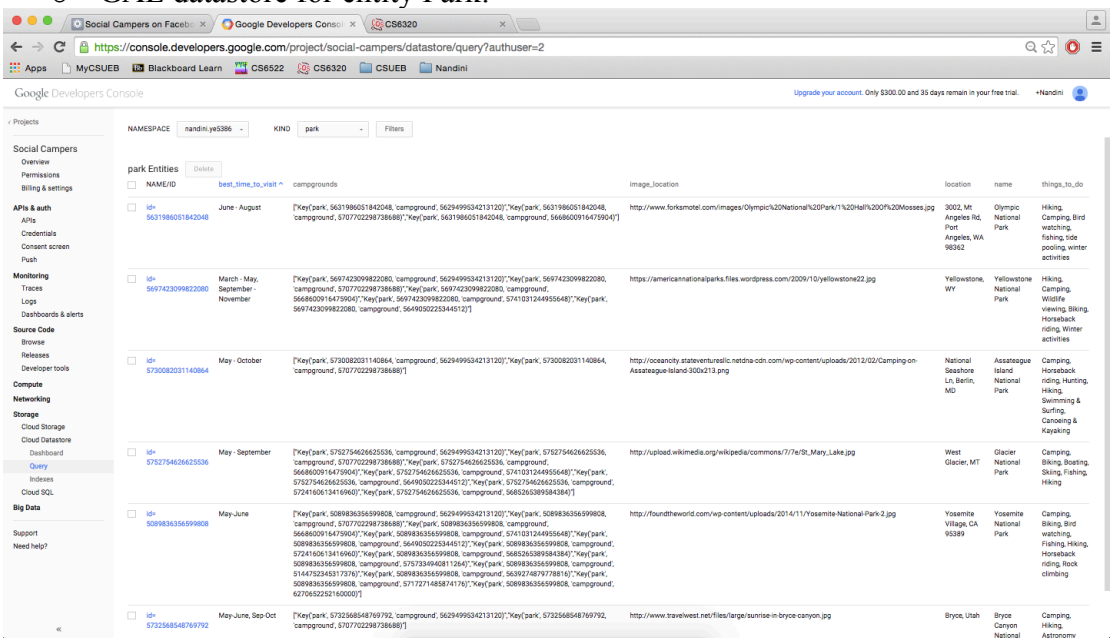
- The Social campers application can make money by selling advertisements.
- Further this application can be developed to make
  - Online reservations
  - Offers/deals
  - Sponsored search.
- List of companies that provide similar services.
  - Trip advisor - http://www.tripadvisor.com/
  - RV park reviews - http://www.rvparkreviews.com/
  - Yelp - http://www.yelp.com/
  - Camping - http://travel.camping.com/

GAE issues and GAE datastore

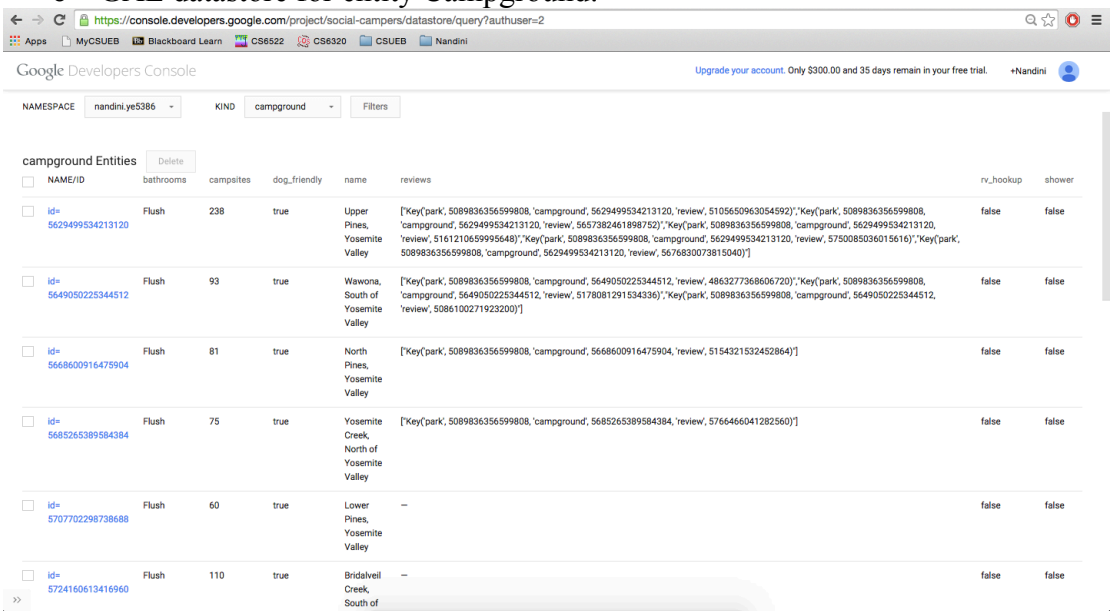
- Screenshots of GAE datastore (with all data in it).
  - Screenshot of app engine activities for the last 4 days.



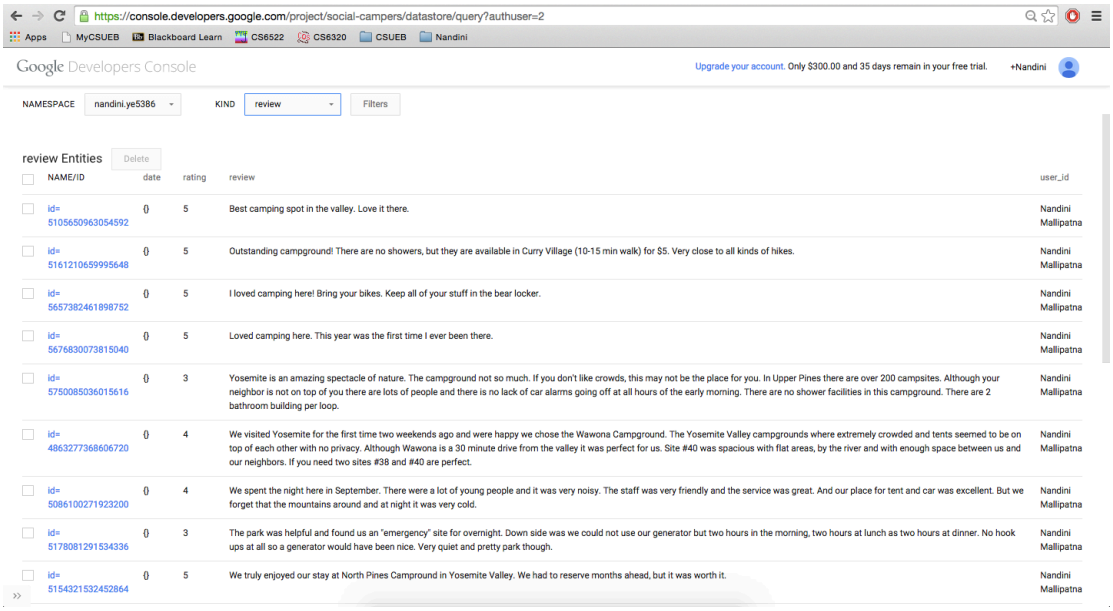
- GAE datastore for entity Park:



- GAE datastore for entity Campground:



- GAE datastore for entity Review:



- Data storage in GAE datastore:
  - Namespace: Nandini.ye5386
  - Entities:
    - Park: Park entity is used to store and fetch park information. It has the following properties with data type:

Property name	**Property value**	**Description**
ParkID	Key	This is a primary key that holds the parkId.
best_time_to_visit	String	This store month (from-to)
Campgrounds	Key	This holds the parkID and campground ID.
Image_location	String	This holds the location of image from where the image is to be fethed.
Location	String	Holds the address of the park.
Name	String	Holds the name of the park.
Things_to_do	String	Holds a list of things to do in the park.

    - Campground: Campground entity is used to store and fetch campground information. Each campground is associated with a park entity. It has the following properties:

Property name	**Property value**	**Description**
CampgroundID	Key	This is a reference key that holds the CampgroundID.
Bathrooms	String	Holds what type of bathroom (flush/pit).
Campsites	Number	Holds the number of campsites in a campground.
Dog_friendly	Boolean	Holds 'True/False' value.
Name	String	Holds the name string for the

		campground.
Reviews	Key	This holds the parkID, campground ID and reviewID.
Rv_hookup	Boolean	Holds 'True/False' value.
Shower	Boolean	Holds 'True/False' value.

- Review: Review entity is used to store and fetch data. Each review is associated with a campground which in-turn is associated with a park.

Property name	**Property value**	**Description**
ReviewID	Key	This is a reference key that holds the ReviewID.
Date	Date (blob)	Holds the date/time when a user clicks on 'Review' button.
Rating	Number	Holds number from (1-5).
Review	String	Holds review text.
User_id	String	Gets the user name from facebook and stores it.

- Issues or modifications made while using datastore in the app:
  - To keep the association between Park→ Campground and Campground→Review entities I used ParkId as a property in campground entity and campgroundId as a property in review entity. This didn't work as desired. So, I had to the add properties viz. campgrounds, reviews in Park, Campground entities resp.

Java code classes

1. Park.java class

a. Class variables:
   i. private Key id;// indicates the id of park
   ii. private String name;// indicates the name of park
   iii. private String location;// indicates the location of park
   iv. private String bestTimeToVisit;// indicates best_time_to_visit the park.
   v. private String thingsToDo;// indicates things_to_do in the park.
   vi. private List<Campground> campgrounds;// indicates the list of campgrounds in the park.

b. Methods:
   i. It has getters and setters for all the class variables mentioned above.

2. Campground.java class

a. Class variables:
   i. private Key id;// indicates id of the campground.
   ii. private String name;// indicates name of the campground.
   iii. private int campsites;// indicates number of campsites in the campground.
   iv. private Boolean dogFriendly;// indicates whether the campground is dog_friendly or not.
   v. private int rating;// indicates the average rating of a campground.

vi. private Boolean shower;// indicates whether there is shower facility or not.
vii. private String bathrooms;// indicates what type of bathrooms are there.
viii. private Boolean rvHookup;// indicates whether there are rv_hookup or not.
ix. private List<Review> reviews;// indicates the list of reviews for a campground.

b. Methods:
i. Getters and setters for each class variable mentioned above.

3. **Review.java class**
a. Class variables:
i. private Key id;// indicates the id of a review entity.
ii. private String review;// indicates the review text.
iii. private int rating;// indicates the rating.
iv. private String userId;// indicates the name of the user from facebook.
v. private Date date;// indicates the date in which the review was submitted.

b. Methods:
i. Getters and setters for each class variable mentioned above.
ii. The int getRating() method checks for the number of reviews. If there are no reviews it sends 0, otherwise the total number of reviews [reviews.size()].

4. **PMF.java**
a. Class variables:
i. private static final PersistenceManagerFactory pmfInstance
// Creates an instance of type PersistenceManagerFactory.

b. Methods:
i. Empty constructor
ii. PersistenceManagerFactory get()
//Sets the namespace to nandini.ye5386

5. **StoreOperation.java**
a. Class variables: There are no class variables.
b. Methods:
i. public static void insertPark(Park park)
//Creates a new park entity in the datastore.
ii. public static void insertCampground(Campground campground, Long parkId)
// Creates a new campground entity in the datastore.
iii. public static void insertReview(Review review, String campgroundId)
// Creates a new review entity in the datastore.

6. **DataRetriever.java**
a. Class variables: There are no class variables.
b. Methods:
i. public static List<Park> listParks()
// Gets list of all parks from the datastore.
ii. public static Park displayPark(String parkId)

// Displays all park properties and it also shows campgrounds associated with the park. Uses fetchCampground to get all the campgrounds associated with the park.

iii. public static Campground displayCampground(String campgroundId)
// Display all campground properties also shows reviews if present. Reviews are associated with the campground. Uses fetchReviews to get all the reviews.

iv. private static void fetchCampgrounds(List<Campground> campgrounds)
// Get all the property values from the campground entity. Uses fetchReviews to get reviews.

v. private static void fetchReviews(List<Review> reviews)
// Get all the review property values from the review entity.

- **FacebookUtils.java**
  - Class variables:
    - private static final String *APP_ID="...";* //Indicates the app_id received from facebook.
    - private static final String *APP_SECRET="...";//* Indicates the app_secret received from facebook.
    - private static final String *SCOPE* = "user_friends,publish_streams, publish_actions";//Indicates the permissions to be asked and received from user.
  - Methods:
    - public static String getUserName(HttpServletRequest session) // Gets the user name of the current logged in user by looking up access_token from request.
    - public static List<User> getFriendsAsUser(HttpServletRequest session)// Gets lists of friends who also use the app.
    - public static List<String> getFriends(HttpServletRequest session)// Gets list of friend names of the logged in user. This gives all friends including ones who do not use the app. This is used for sending app invites.
    - public static void postMessageOnTimeline(HttpServletRequest session, String message)// Post a message on user's timeline. The user should have given publish_stream and publish_activity permissions. Note that if the user is not a developer, its not possible to post because Facebook restricts it unless the app is reviewed.
    - private static List<String> getFriendNames(Connection<User> friendConnections)//Gets a list of friends name who are using the app.
    - private static FacebookClient getClient(HttpServletRequest session)//Gets access_token from session and sets app_secret and version.
- **CreateParkServlet.java**
  - Class variables:
    - private static final long serialVersionUID = 1L;//Indicates universal version identifier for a Serializable class.
  - Methods:

        - public void doGet(HttpServletRequest req, HttpServletResponse res)// Fetches the user entered data from the web page for park and creates a datastore entity.
- **CreateCampgroundServlet.java**
    - Class variables:
        - private static final long serialVersionUID = 1L;//Indicates universal version identifier for a Serializable class.
    - Methods:
        - public void doGet(HttpServletRequest req, HttpServletResponse res)// Fetches the user entered data from the web page for campground and creates a datastore entity.
- **CreateReviewServlet.java**
    - Class variables:
        - private static final long serialVersionUID = 1L;//Indicates universal version identifier for a Serializable class.
    - Methods:
        - public void doGet(HttpServletRequest req, HttpServletResponse res)// Fetches the user entered data from the web page for review and creates a datastore entity.

Code zipped up

Code will be attached as a separate attachment in the post below.

References

[1] https://cloud.google.com/appengine/docs/java/datastore/
[2] https://cloud.google.com/appengine/docs/java/datastore/jdo/creatinggettinganddeletingdata
[3] http://restfb.com/
[4] https://developers.facebook.com/docs/graph-api/reference/
[5] http://www.w3schools.com/js/