CPSC 473 - Web Programming and Data Management

Spring 2014

Project 1

- Presentations: week of **March 23**
- Source code and documentation due: week of **April 6**

Use client-side JavaScript and Node.js to build a web application. You may use any available third-party Web Service APIs, libraries, or modules.

Server-side data storage

Use of a server-side data store such as an RDBMS or NoSQL database is not required for this project. You may use one if you like, but storing data as variables in the memory of your Node.js process is perfectly fine for this project.

If you would like to persist data server-side without tackling Chapter 7 of the textbook, consider using a simpler server-side framework such as Hoodie or Meteor.

Functionality

Note that the following project descriptions are brief and deliberately under-specified. Start with a set of features, plan according to the available time, and build something interesting. Think of your project as a proof-of-concept or prototype: the initial version of a site that you might build in order to attract customers.

Section 1, Team 1 - “Hangdog Happenstance”

- Andrade, Mario Alberto
- Castrillon, Jhonny Alexander
- De Almeida Cruz, Bruna Angelica
- Donaldson, Eric Alan
- Jangid, Sumit Subhash
- Saad, Kathy Bassem

Section 2, Team 1 - “Reviled Curator”

- Ahmad, Maira
- Huebert, Brandon James
- Khairatkar, Anay Uday
- Murmann, William Anthony

- Takale, Aakash Mahesh
- Thakkar, Meet Sudhirkumar

Section 2, Team 2 - “Lugubrious Propinquity”

- Barroso Silva, Erick Bhrener
- Lujan, Calvin Hector
- Mubarki, Mussa M
- Murphy, James Francis
- Phan, Tommy
- Yaghoobi Saray, Saba

Build a social site where people can meet other people with similar interests. Allow each user to maintain one or more “favorite” lists for different topics (e.g., books, movies, television shows, games). Users should be able to find and contact other users whose lists contain similar items.

Section 1, Team 2 - “Diacritical Crackerjack”

- Kiriakos, Frida s
- Movius, John Preiss
- Rojas, Eduardo
- Ruiz Gutierrez, Annette De Jesus
- Valbuena, Denice Ron Murillo

Section 1, Team 3 - “Inestimable Circularity”

- Chern, Pete Ming
- Gomez, Alberto
- Hackemack, Robert Derrick
- Sok, Kourun
- Yadav, Vivek

Section 2, Team 3 - “Unsubstantial Harbinger”

- Baflah, Somayah Omar
- Chheda, Yash Kirti
- Patel, Dhaval Kamleshbhai
- Rafiee, Mehrdad
- Vo, Johnny Dieu
- Westerman, Andrew John

Build a community-based Twitter gateway, allowing groups of users to post to Twitter as a group. Individual users can submit updates, but they will not be posted to Twitter until other members of the group agree that they should be published. Users can submit new updates, edit existing unpublished updates, or vote existing unpublished updates up or down. When an

update reaches a given number of votes, it will be published.

Section 1, Team 4 - “Reproachful Kookaburra”

- Chau, Hung Ngoc
- Choudhari, Akhil Angad
- Cobb, Nathan Robert
- Danan, Christopher Dancarlo Roque
- Yenter, Alec Matthew

Section 2, Team 4 - “Representational Lateness”

- Anand, Chaitra
- Desai, Chandni Dashrath
- Mandang, Michael Jordan
- Patel, Vishal Chandrakant
- Watson, Brian Michael
- Wypych, Michael Sean

Section 2, Team 5 - “Untitled Equalizer”

- Choi, Ah Som
- Gandhi, Utkarsh Pankaj
- Petersen, Mikkel Vester
- Simas De Souza Neto, Felix
- Truong, Brandon Thien

Build a web application for community decision-making. Users can post an issue to be decided and several possible responses. Other members of the community can vote existing responses up or down, or submit new possible alternative responses. For example, a group of friends might use a site like this to decide where to go to lunch, or a family might use it to decide on a name for a new baby or pet.

Section 1, Team 5 - “Cutthroat Cohort”

- Anil Kumar, Karan
- Huynh, Albert
- Le, Henry Hong
- Lee, Sarah
- Van Steenburg, Yuriko

Section 1, Team 6 - “Breathtaking Tetrahedron”

- Cao, Linh Hoang

- Chau, Jenny
- Crane, Tyler Scott
- Gupta, Kartikeya
- Meyerhardt, Kyle
- Pham, Truong Quang

Section 2, Team 6 - “Morphological Inelegance”

- Ater, Timothy Phillip
- Mullangi, Venkata Naga Sathya
- Rivas, Francisco
- Shah, Dhaval Vardhaman
- Tahir, Umair

Build a site that will allow users to post ephemeral questions and answers. Users can post questions and solicit answers from the community, but only for a short period of time such as 10 minutes. When users visit the site, they can see and respond to questions that are currently active, or post new questions of their own. When a question expires, it should no longer be active. (If you prefer, it may also disappear completely, a la SnapChat).

Submission

On the day of the presentation, give a short demonstration of your application to the class. Include both functionality and implementation details. Your entire team must be present, but everyone does not need to present.

On the due date, do the following:

1. Document your work in a PDF file with screenshots demonstrating your application’s functionality.
2. Create a `.ZIP` or `.tar.gz` file containing your `.PDF`, `.js`, and other assets.
3. Send your `.ZIP` or `.tar.gz` file as an attachment in an e-mail to csuf.kenytt.net@gmail.com
4. Alternatively, submit your project as links to a GitHub repository rather than directly as attachments.
5. Include your section, team number, team name, and the names of all members of your team in your e-mail.
6. Set the `Subject:` line of your e-mail to
   `[CPSC 473 - Section 1] Project 1`
   or

`[CPSC 473 - Section 2] Project 1`
as appropriate (Wednesday night is Section 1; Monday night is Section 2).

Grading

Each of the following will be graded on a 2-point scale (0, 1, or 2 points):

- Functionality - Does the application work as advertised? Is it a reasonable approximation to the project description?
- Error handling - Is there any attempt to handle, report, or recover from errors? Is there any validation of input?
- Documentation - Is there any? Does it tell me how to get started? Is there any additional information that users will need?
- Source code hygiene - Does JSHint pass? Are lines indented appropriately? Are there large chunks of unexplained commented-out code floating around?
- Source code maintainability - Are there reasonable variable names? Are there modules? If you needed to add some functionality or fix a bug, would it be easy to find the appropriate spot?

The following factors are each worth an additional point, up to a maximum of 10:

- Aesthetics - Does it look nice?
- Creativity - Does it do anything special?

Goofy team names courtesy of the Random Phrase Generator.