Definitions of ‘Done’ for ChoreDash

**Tasks:** Tasks that make up user stories vary greatly due to their specificity, and often small tasks may not even add testable functionality to the website from the user’s perspective. To simplify agreement on the completion of tasks, any task of any user story for the project is said to be complete once it is reviewed by one or more teammates and push to the appropriate branch on github.

**User Stories:** User stories’ completion definitions are defined as a team during sprint planning when tasks are created and assigned for each user story. Generally, a user story is completed once it is reviewed and tested by all teammates and the working branch for the user story is merged with the master branch.

\*definitions of done for learning related sprint 1 user stories are omitted

**Story 1:** As a developer, I want an encrypted login system

**Done:** This story is completed when users can register for an account and their password is

stored as a hash in the database. When the user tries to login, the password they enter

while logging in is hashed and compared to the existing hash in the database.

**Story 2:** As a user, I want a usable web page with member login

**Done:** This story is considered implemented when a user can register for an

account and then use the same credentials to log into the website to access functionality

restricted to members.

**Story 3:** As a developer, I want to restrict users to UCSC emails

**Done:** Done when users are not able to register with non @ucsc.edu emails and they

are warned when their username does not meet these requirements when signing up.

**Story 4:** As a user, I want to be able to make a profile page

**Done:** This is finished when a user can navigate to their profile page and see the information that they entered into the website about themselves displayed in an organized manner.

**Story 5:** As a user, I want to be able to post and manage a job

**Done:** Done when a user can fill out a form to post a job with a title, data, location, and some

basic information about the job. Jobs appear in a feed on the dashboard and user see

‘edit’ and ‘delete’ buttons on posts which they created which allow them to edit and

delete posts respectively.

**Story 6:** As a developer, I want to implement a database that manages posts

**Done:** Posts database management is completed once job data is stored in a ‘post’ model in

the mongo database with appropriate backend express.js functions in place to get, add,

edit, and delete posts to/from the database.

**Story 7:** As a user, I want to be able to contact other users by email

**Done:** Completed when users can see the email of job posters on the posts so that they can

email the poster manually.

**Story 8:** As a user, I want a user friendly dashboard/user interface

**Done:** Text is legible, there is ample spacing between elements, styling within components is

consistent, and buttons/links follow a logical flow/layout in order to navigate through the

site.

**Story 9:** As a user, I want to see where the jobs are located

**Done:** Story is complete when users can include / see a location field on posts. Optionally, a

google maps API can be used to suggest addresses/locations

**Story 10:** As a user, I want to able to rate the people I hire

**Done:** This story is done when a rating appears on all profiles that is an average of all ratings

given to a user by other users who have interacted with the user through a job.

**Story 11:** As a developer, I want to fully test and debug the website

**Done:** The website is considered sufficiently tested and debugged when all glaring flaws that

severely limit the functionality of the website or stop the website from functioning entirely

have been removed.

**Story 12:** As a user, I want to see a visually appealing website

**Done:** Done when most of the styling on the website is consistent and rounded-looking without

too many basic HTML elements and similar ‘raw’ or ‘unpolished’ looking components

**Story 13:** As a user, I want to be able to message/chat with relevant users

**Done:** Messaging is done when users can message other users and start a conversation

thread similar to email and the incoming messages are stored in the database for each

User. Users can see all messages addressed to them, grouped by the users they originate from, on a /messages route and can reply / compose a new message on this route.

**Story 14:** As a user, I want directions to the job

**Done:** The job directions story is complete when the user can click on the location on a job

posting and be directed to a google maps link that directs them to the listed job posting

location.

**Story 15:** As a user, I want reminders for upcoming jobs

**Done:** Done when users get a browser notification and/or an email 24 hours before a job is set

to begin, if a job time is provided as part of the post the user is responsible for.

**Product Backlog**

**Story 16:** As a user, I want to be able to pay and receive payment for jobs

**Done:** done when there is a link to paypal payment form with ‘amount’ and ‘payment to’ field

pre-populated with relevant user’s paypal account and post’s pay rate. The link should

be in the ‘manage jobs’ component and should appear on jobs that are past the time at which they were set to start.

**Story 17:** As a user, I want an intuitive search function

**Done:** Story is done when we limit jobs that appear in the feed to a set number that load at a

time as user scrolls down and add a search box at the top with which users can enter

text which will be matched against job posting titles, poster username/names, and post

bodies to return all matching posts.