To Do by Friday 12/2
Revised design doc
- Updates to data model (Val)
- Updates to security (James)
- More explanation about design decisions and thought process (Denis?)
- disclaimer about plagiarism terms and conditions (Karleigh)
- Address other comments Vinay had (Karleigh)

To Do by Wednesday 12/7

~~- User routes (need some routings) (Val will finish)~~ (I think I'm done-- let me know if there are any other routes you'd like implemented!)
~~- User schema (Val completed already)~~
~~- Validators (Karleigh will make sure all schemas have validation)~~
- Passport stuff ~~(Val has done-- needs to cite sources)~~ ~~*include crypto if missing (karleigh if needed)~~
- ~~login/signup pages (James)~~
- ~~Comments~~/likes/parents//~~NSFW~~ -- UI buttons/icons (James)
- Tags and tag search (implementing routing for tags and search bar for UI, Denis)
- UI for User profile need follow button, icons, etc (Denis)
~~- Make the UI a little nicer. Make sure to change the size of the containers for the news feed so that they are uniform (Karleigh)~~
~~- Need authentication for POST/DELETE/GET requests (make sure that user is logged in and if necessary, that it is a specific user logged in) (Karleigh will do)~~
- Code review (all!)
- Implement Nav bar
Added more tasks below based on the design doc
- Implement deleting a chart
- Make description and tags mutable
- Creating a link to parent chart (if one exists) on the chart_page (probably chart.template.html) actually
- Reporting system: NSFW and copyright infringement
- (Advanced feature, but I feel like we should also implement pencil down for working on the chart)
~~- Comment schema (Karleigh) (pushed to Master)~~
~~- like routes (Karleigh has done some of this-- needs help)~~
- Code clean up (all)

I can't figure out how to make a remix chart by clicking on the chart, going to the chart page, and then clicking remix. I get a 400 status in console but no redirect

1. Implement like feature
    1. server adds and removes the like (note that it also has to send some other response when a user likes twice)

    2. client-side implements button (and UI) and requests for handling like and unlike
2. Implement commenting
    1. server adds and removes comment
    2. client-side implements button and textbox for commenting. Also handles the requests necessary for posting/deleting (do we even have delete feature for this?)
3. Implement parent display
    1. server side needs to handle get request for a single :id (actually, I'm not sure, we might be able to just use the route which gets all the charts and pass in the right parameters)
    2. client-side need to add a link to the parent chart
4. Implement deleting a chart
    1. server side needs to change the models accordingly. Also, it should remove all Like and Comment documents that contains the chart
    2.
    3. client-side just needs to implement such a button on the chart-page and the request for deleting
    4.
5. Change chart_page to handle deleted charts.
    1. Purely client-side. Use Mustache templates
    2.
6. Implement user-profile
    1. Clientside: User profile simply displays charts that the user has created.
    2. Serverside: I don't know there is work for this part yet.
    3.
7. Add feature to display liked-charts on user-profile
    1. clientside: add a button for displayed the charts that the user has liked
    2. serverside: mongodb magic
8. Implement search
    1. mongodb magic
    2. implement all the various widgets and design the request
    3. This is actually fairly complicated. Some parts are already done, but the remaining parts are not easy to handle either
9. Implement nav-bar
    1. Should be decently easy? Just add some buttons put some links and make sure that certain variables are stored in window.sessionStorage
10. Implement reporting system
    1. Put in a button that sends and email to national-badger.
    2.
11. Implement authentication (csurf, etc). We need to put tokens into our forms and make sure they are checked (for CSRF)

we should update things so that it redirects when a user is not logged in to prompt them to do so and also maybe not even show the "remix" or "make chart" button if not logged in. also change the UI in a way that when they are logged in they have feedback (like a log out button appears and the login/signup goes away)

Stuff to do before final turn in
- Testing

- CSS niceness
- Deploy
- Write documentation/readme

Due Mon 11/14
Pitch (google slides) (Karleigh will do slides, everyone will contribute content)
- Name (each member come up with 3 names by Sat 11/12 1pm)
- Design risks (Denis) by Sunday afternoon at latest
- Concepts (each member will write some concepts to inform Val’s data model) by Sat 11/12 1pm

Due Tuesday 11/15 12:30pm (internal) (official Wed 11/16)
Design doc
- Data models (Val by Friday 11/11 afternoon)
- Operational principle (each member will write operational principles to inform Val’s data model) Sat 11/12 1pm
- Security concerns (James) due Monday
- UI and wireframes (Karleigh) Monday (rough) Tuesday (final)

By Friday 11/18
- James and Denis will give Karleigh and Val a list of the functions/routes/info they need for their views. Also decide whether or not you are using react.js.

Due Wed 11/23
MVP
- Create account and login (passport.js?)
- Make patterns and save them (similar to game of life?) (support deleting?)
- See everyone else’s patterns (similar to Fritter)
- Splashpage
- Remix (Idea for how to do this: button to remix <similar to retweet> which loads up a clickable chart with the original pattern’s colors preset. Then becomes like a normal chart object. All chart objects should have “parent field” and if original chart parent=none)

Post-MVP
- Comments on patterns
- Share others’ patterns and remix
- Individual user page
- Following others
- Star ratings
- Tags
- DOB
- Blocking user from viewing NSFW charts

- Flagging/reporting system

Due Mon 11/28
Peer reviews

Due Fri 12/2
Revised Design

Due Wed 12/7
Feature freeze

Due Mon 12/12
Final Presentation and Code Hand In

Notes from 12/7/16:
-we took a stab at things, lots of bugs
-commenting,liking, parent find, nsfw, etc→ priority
-secondary: polishing (margins, standardized colors/fonts/spacing)
-maybe remove about page? Unless we have nothing to do
START UNIT TESTING
-testing! Test every public function on server (edge cases!)
    -could also think about testing routes instead (using route testing framework-- may be slightly harder)
-if front end has significant logic, test it
DOCUMENT PUBLIC FUNCTIONS
- error handling, authentication (make sure that people can't edit desc/tags of charts not theirs)
    - Also. If these fields are editable, there should be some sort of indicator to the user that editing is possible!
-fix non-RESTful routes
 - /charts/43943282
 - req.params.id ←use in route
/GET users/
POST users/:myid/following   to follow (pass in data about user to be followed)

- Instance + static methods on mongo
    - Avoid including logic in routes
    - Put logic in model
    - this.model("freet") (see static and instance methods in mongo documentation
    - Ensure request is valid in route, if so: pass into model
    - In model callback function, send response to user
    - If all logic is in models, we only have to test our models