Deliverable 3

What tools will you use for your task board?

Trello. It's easy to use and comes with an intuitive user-interface.

What tools will you use for your burn-down chart?

Online Sprint Burndown Chart Generator offered by Visual Paradigm:
https://online.visual-paradigm.com/diagrams/templates/line-chart/sprint-burndown-chart/

Who will maintain the burn-down chart? How?

Jesse Francispillai.
He will take snapshots from the current sprint backlog after every workday to update the burndown chart. Data will be inputted into the online generator manually. Committed every 48 hours.

What is every team member's role?

- **Full-Stack Developers:** Everyone
  - Front-end: HTML, CSS, JavaScript, Bootstrap
  - Back-end: NodeJS, ExpressJS, MongoDB, Mongoose
- **Scrum Master:** Jesse
- **Client Liaison:** Husni, Maduvan
- **Note taker:** Ahmad
- **Burndown Chart Updater:** Jesse, Joseph

What tools, if any, will you use for communication?

1. Facebook Messenger Group Chat
   - Primary method of communication. Used for generic messaging as well as group calls.
2. Trello Task Board
   - Used to add notes/comments to specific development tasks
3. CSCC10 Github Group
   - Used for any formal communication that needs to be shared with the prof/TA.

When do you plan to meet in person?

Due to the current nature of the global pandemic, we do not plan to meet in person for the duration of the course. In accordance with our Team Expectations, we will continue to hold weekly online meetings every Friday.

How will you use your repository on GitHub?

We plan to use the following categories of branches:

- Master Branch
  - Will contain code for release, updated at the end of every sprint

- Develop Branch
  - Will contain completed features that have yet to be pushed to release
- Feature Branches
  - Will contain code for features still in development
- Bugfix Branches
  - Will serve as temporary branches to fix bugs in completed code

Naming Convention for branches:

- **Story**<User Story Number>**Task**<Task Number>
  - E.g “Story1Task1” would be the branch for user story 1, task 1

Styleguide:

Code Style: ESLint

Documentation: Using the following format for backend operations:

- One section for each of the CRUD operations
- The request URI with the HTTP operation
- The format of the body of the request
- The response body for various responses

Example:

~~~
### Create

- description: Upload a new image
- request: `POST /gallery.html/images/`
   - content-type: `application/json`
   - body: object
     - title: (string) the title of the image
     - file: (file) the image file
- response: 200
   - content-type: `application/json`
   - body: object
     - file : (object) the image object in JSON format
     - _id: (string) the image id
     - title: (string) the title of the image
     - author: (string) the image author's username
- response: 403
   - Forbidden

```
$ curl -X POST
      -H "Content-Type: `application/json`"
~~~

```
     -d '{"content":"Test image"}
     http://localhost:3000/gallery.html/images/'
```

What are your rules for what's being committed (eg. autogenerated files?)

- The file is used for the project
- The file will be used by anyone else in the team
- The file was not generated by another process
-

What are your rules for what's being included in commit messages

- Explanation of what task was being completed and how much of it has been done
- Small detailed explanation
- Each task is written in sentence form

In addition, we plan to have mandatory peer-reviews of at least 1 other teammate for commits to the Develop and Master branches.

Which machines will be used for development by each team member?

- Maduvan: Macbook Pro Laptop
- Joseph: Windows Home Computer
- Jesse: Macbook Pro Laptop
- Ahmad: Macbook Pro Laptop
- Husni: Windows Home Laptop

What is your DoD (definition of done)?

- All user stories have been pushed through the task board and are completed
    - All user stories have been verified (part of task board)
- All commits to master branch have been code-reviewed (enforced on Github repo)
- Deployed on server
- Units tests achieve at least 85% code coverage over the entire system
- Deployed product has been sanity tested
- Client has seen and approved final release

Product Backlog:

Trello Board: https://trello.com/b/V8Ty9yno/agile-sprint-board

Reasoning Behind Story Points and Priorities:
One story point will be equal to 30 minutes. This means a story that will take one hour will need 2 story points. The reason we made the interval 30 minutes was so that we can be more specific in the work/time required. Stories are prioritized according to the general guidelines set

by Scarborough Dining. The first priorities are also to get the structure of a minimum viable product.

Static Table:
🟨_: Restaurant Search Engine 🟪_: Restaurant Profile Page ⬜_: Online Ordering
🟩_: Simple Social Network 🟦_: Analytics Dashboard 🟥_: Other

Priority	User Story	Estimated Cost (in hours)
1	As Steve (Restaurant Owner), I want a page to feature my restaurant's address and contact information, so that my customers can get in touch with me directly.	6
2	As Steve (Restaurant Owner), I want to showcase personal stories and anecdotes about my dishes, so that customers can understand the history behind them.	5
3	As Javier (Customer), I want to search for restaurants based on rating, so that I know which places would draw in the most views to my blog.	6
4	As Steve (Restaurant Owner), I want to upload simple photos of my dishes and restaurant, so that I can promote exactly how I did in traditional media.	4
5	As Fiona (Customer), I want to see videos of how restaurants make their food, so that I can feel safe ordering it for me and my children.	4
6	As Fiona (Customer), I want to search for restaurants based on their cuisine, so that I can make sure the food I order meets my dietary needs.	5
7	As Bonny (Customer), I want to search for restaurants by price, so that I can stay within my tight budget.	4
8	As Bonny (Customer), I want to order food for pickup, so that I can have it ready when I arrive and save time.	8
9	As Fiona (Customer), I want to order food for delivery, so that I can order on my commute and save time.	4
10	As Steve (Restaurant Owner), I want a simple ticket system to see online orders, so that my kitchen doesn't waste valuable time learning the new system.	4
11	As Diane (Restaurant Owner), I want to see how often my	6

	restaurant appears in various searches, so that I can see what sets me apart from my competitors	
12	As Steve (Restaurant Owner), I want a daily record of what was ordered, so that I can easily cross-reference with my physical inventory	4
13	As Diane (Restaurant Owner), I want to see how often my various menu items are ordered, so that I can decide where to expand and where to cut back	4
14	As Diane (Restaurant Owner), I want to see a distribution of when (time/date) my orders come in, so that I can choose effective promotional periods	5
15	As Fiona (Customer), I want to see the nutrition labels of any food I order, so that I can ensure it pertains to my dietary needs.	8
16	As Diane (Restaurant Owner), I want to show where my ingredients are sourced from, so that I can differentiate myself from the chain franchises.	6
17	As Javier (Customer), I want a personal profile page, so that I can link my other social media accounts, as well as my blog.	7
18	As Diane (Restaurant Owner), I want to see the demographics of the people that order from my restaurant, so that I can hone in on my target audience	6
19	As Javier (Customer), I want to share my thoughts on restaurants, so that I can grow my brand and public influence.	5
20	As Diane (Restaurant Owner), I want my customers to be able to share media that links to my restaurant page, so that they can direct their friends there.	4
21	As Javier (Customer), I want to interact with other people’s posts, by liking and sharing them, so that I can connect with my friends.	5
22	As Diane (Restaurant Owner), I want to see how customers are interacting with posts about my restaurant, so that I can directly see what they generally like and dislike	4
23	As Diane (Restaurant Owner), I want to make posts that target people who frequent my restaurant, so that I strengthen the ‘local community’ feel.	7

24	As Javier (Customer), I want to see how often my friends have eaten at a restaurant, so that I can judge how popular and ‘trendy’ that restaurant is.	8
25	As Bonny (Customer), I want to search for restaurants by location, so that I can make sure I have the means to actually get there.	9
26	As Bonny (Customer), I want a way to see how long it would take me to reach a restaurant via public transit, so that I can ensure I can make it there and back within my schedule.	6
27	As Diane (Restaurant Owner), I want online orders to be sorted by expected pick-up time, so that my kitchen can be efficient with their cooking.	5
28	As Javier (Customer), I want a website that has a fully-featured and intuitive mobile interface, so that I can easily access it on-the-go.	4

Release Plan:

Sprints will be one week long, and will last the duration of the course. By opting to use one-week sprints, we can have more frequent retrospectives - we felt this was important since it allows us to more quickly bring up any unforeseen issues to the rest of the team. In addition, shorter sprint times necessitates more planning and task-allocation sessions, which results in individual sprint backlogs being pivotal to the project as a whole. This provides us with some flexibility when prioritizing tasks and assigning developers.

Sprint Plan:

Sprint 1 Backlog

Story 1: As Steve (Restaurant Owner), I want a page to feature my restaurant’s address and contact information, so that my customers can get in touch with me directly.

Acceptance Criteria:

- Given Steve wants to create a profile page for his restaurant, when Steve clicks on the “Create Restaurant” button, a web form with relevant fields should appear for him to fill out.
- Given Steve has completed filling out the form to create a restaurant, when Steve clicks on the “Complete” button, he should be directed to a profile page for his restaurant, with the relevant info published.

- Given Steve wants to edit his restaurant's profile page, when Steve clicks on the "Edit Profile Page" button, the fields on his page should become editable textboxes.
- Given Steve has finished editing his restaurant's profile page, when Steve clicks on the "Complete" button, the textboxes should become immutable once again.
- Given Steve is entering information into fields for his restaurant's profile page, when Steve tries to submit an inappropriate entry (e.g. alphabet characters in "phone number" field), he should be presented with a pop-up detailing the error(s).

Tasks:
- T1: Create Restaurant Schema (file that holds schema) to have a structure to follow for all restaurants
- T2: Create UI Form for restaurant owner to create restaurant info
- T3: Link submitted form to the database
- T4: Update front end to reflect changes to restaurant database
- T5: Write unit tests for the backend portion of the web form

Story 2: As Steve (Restaurant Owner), I want to showcase personal stories and anecdotes about my dishes, so that customers can understand the history behind them.

Acceptance Criteria:
- Given Steve wants to add new text to his restaurant's profile page, when Steve clicks on the "Update Content" button, a web form should appear for him to fill out.
- Given Steve has previously submitted content to his restaurant's profile page, when Steve clicks on the "Update Content" button, the web form that appears should be pre-filled with the current content.
- Given Steve has completed filling out the form to update content, when Steve clicks on the "Complete" button, he should be directed to his restaurant's profile page, with the updated info published.

Tasks:
- T1: Create UI Form for Additional Text Info in Restaurant Profile Page
- T2: Link submitted form to the database
- T3: Update front end to reflect changes to restaurant database
- T4: Write unit tests for the backend portion of the web form

Story	Task	Cost	Priority	Day 1	Day 2	Day 3	Day 4	Day 5	Assigned
1	T1	2	1	0	1	1	0	0	Husni
	T2	2	1	0	1	1	0	0	Jesse
	T3	4	2	0	0	4	0	0	Husni

	T4	2	2	0	0	2	0	0	Joseph
	T5	2	3	0	0	0	0	0	Joseph
2	T1	2	1	0	0	0	0	0	Jesse
	T2	4	2	0	2	1	0	1	Ahmad
	T3	2	2	0	0	1	0	0	Maduvan
	T4	2	3	0	0	0	0	0	Maduvan
Total		22		0	4	10	0	1	
Work Left				22	22	12	12	8	

Brief overview of project from deliverable 2 to 3: In deliverable 2, we created all the user stories and grouped them by category. Leading to deliverable 3, we were able to prioritize all the user stories and create the product backlog on our Trello board. For our sprint backlog, we chose two user stories that were the highest priority to get a minimum viable working product. We were able to split these into tasks and we started working on them while starting daily scrum meetings and committing our tasks to the git repository.

Estimated project velocity: We created our sprint plan using 30-minute intervals as being equal to one story point. The total work for our sprint was equal to 22 story points. This was 2 stories and the relevant tasks. The estimated project velocity as shown in the burndown chart was 22 story points over 5 working days split between 5 team members.

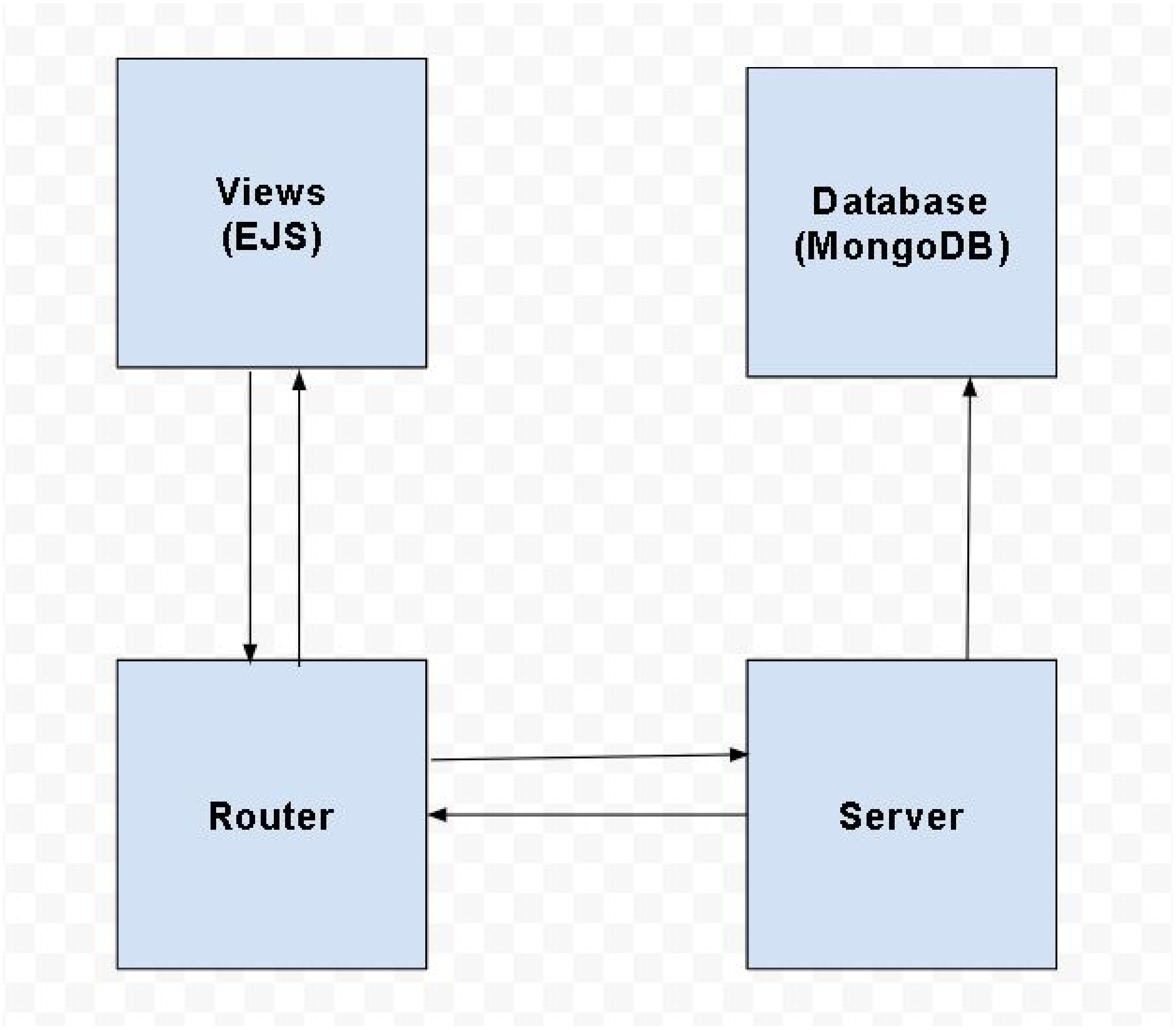
Actual project velocity: As seen through the sprint 1 backlog and the burndown chart, a lot of time was spent on planning the components and then many story points were completed on day 2 and 3. The actual project velocity did not keep up with the planned estimated velocity and these issues will be better estimated and resolved in the next sprint which is better planned.

Did you follow your plan(s) exactly: No. The tasks in our sprint backlog were not all able to be completed in the duration of the sprint. As such, some of these tasks will have to spill over to the next sprint.

What difficulties did you encounter: Prioritizing the sprint tasks proved to be a more pressing matter than initially expected. In particular, many tasks were unable to be completed because they were blocked by other tasks for a large duration of the sprint. Another factor in tasks taking longer to complete than they should have was that all the initial system components (DB, node template, etc) needed to be setup as boilerplate work, and all members needed to get up-to-speed with this system before starting their actual tasks.

Solutions: Our contingency plan did not initially incorporate daily standups for the sprints, which would have been very useful in hindsight. The lack of a scheduled standup resulted in no standup meetings for the first couple days of our first sprint. This, compounded with the lenient response times outlined in our contingency plan, served to exasperate the blocking issue with our tasks. As a solution going forward, we have much more clearly-outlined daily standup expectations, and are placing a lot more emphasis when prioritizing tasks.

Component Diagram



Views(EJS): Files contained within the views component will be responsible for holding the content to be displayed on the webpage. This component directly communicates with the router to both send and receive requests from the user. It utilizes the http protocol and sends post,get,delete, or update requests and will call the endpoints within the router.

Router: Component directs requests from the user to the server. It will also allow for direct communication from the server to the Views page by sending response's back. It allows dynamic rendering of the views page per request being sent if needed.

Server: The component which provides a request-response functionality for the router to be able to fulfill the end user's request to the server. The server will also have the role of querying collections in the database to be able to create/retrieve/update/delete Information as per requested from the end user.

Database (MongoDB): A storage for collections of data which allows for easy management. This data is organized with different schemas so that collections are separated as needed. Data within the database will have different purposes based on what is needed within the program such as authentication.

Component Dependencies:
Views dependency on Router
The views pages load by themselves however the content which is unique to the webpage and stored in the server cannot be accessed without the router. So the views pages are dependent on the router.

Router dependency with Views:
The router component is dependent on the views pages since it cannot receive any requests otherwise. Router's main purpose is to route requests from the user to the server.

Router dependency with Server:
It also is dependent on the server since it asks the server for a response based on the request that it is given. Without a response on the server the router will not be able to direct content from the database to the views component.

Server dependency with Router:
The server depends on the router because the server needs to know what to call from the Database. Without the router communicating to the server, the server is not able to receive the initial request from the end user.

Server dependency on Database:
The server is connected to the database because the server will be manipulating the data within it. The server receives requests from the router and needs to provide responses based on information in the database.