Team 15 Sprint 1 Planning Document
Delivery and Order Management System(DOMS)

Kunal Goyal, Hemil Desai, Shivank Tibrewal, Dinesh Gajwani

Sprint Overview

The goal for our first sprint is to get started with basic user stories of our project and to get familiar with the architectures and logic behind our software. At the end of the sprint, we plan to have a minimum viable product that can be represented with basic user functionality.

We have described the user stories, we aim to implement in this sprint. Along with each user story, are the key technical features associated with it. Each user story has been assigned to a team member with the number of hours expected to complete it.

Scrum Master: Hemil Desai
Scrum Meeting Time: 1:30pm MWF
Risks/Challenges: Getting familiar with new libraries and frameworks like AngularJS and bootstrap for web development and Node.js for the backend API.

Current Sprint Detail

User Story: As an Admin, I would want to manage store owners (Add, Remove, Modify) into the system

Task Description	Estimated Time	Owner
Construct a simple web page using HTML, CSS and Javascript form where the details of a store owner can be added and submitted through an HTML request	6	Shivank Tibrewal
Perform Validation to check if store owner already exists or user entered invalid data	2	Kunal Goyal

Construct a simple web page using HTML, CSS and Javascript form where the details of a store owner can be loaded from the database through an HTML request	5	Kunal Goyal
Construct a simple web page using HTML, CSS and Javascript form where all store owners in the database are listed through an HTML request	5	Shivank Tibrewal
Make changes to an already existing store owner account, after its details have been loaded.	2	Kunal Goyal

Acceptance Criteria: If this user story is implemented well, the tester should be able to add a store owner, view all store owners in the system, delete a store owner, load and modify the details of a store owner.

User Story: As an existing store owner, I need to log into the DOM system.

Task Description	Estimated Time	Owner
Research and construct a simple web page using HTML, CSS and Javascript where a store employee can type his or her username and password and submit the HTML request	6	Shivank Tibrewal
Research on passportJS and then receive the HTML request and authenticate the user using passportJS	8	Kunal Goyal
Redirect the user to a new logged-in-webpage using HTML,	5	Kunal Goyal

CSS and Javascript where he or she can view their own information.		

Acceptance Criteria: If this user story is implemented well, the tester should be able to log in to the system if he or she exists in the system and should not be able to log in if he or she doesn't exist.

User Story: As a store owner, I should be able to add new orders to the system.

Task Description	Estimated Time	Owner
Create a new route in the backend API to handle the orders placed and store them into the database. Use the same routes to get orders from the database. Implement CRUD functionality on orders. (Node.js)	6	Hemil Desai
Create a store owner dashboard using HTML, CSS, JS and angularJS. Create a new page on the web application using HTML, CSS and JS to handle form input/output.	6	Dinesh Gajwani
Use the newly created orders to display analytics to the stores. This analytics information will be displayed on the dashboard of the store owner.	4	Hemil Desai

Acceptance Criteria: If this user story is implemented well, the tester should be able to view the front end admin dashboard with analytics information via HTTP GET requests to the development server, add orders to a particular store using POST requests to the development server, and check the API routes through Postman.

User Story: As a store owner, I should be able to manage drivers.

Task Description	Estimated Time	Owner
Create routes to display the list of drivers for a particular store in the backend API. Data will be transmitted in JSON. Interact with the MySQL server to get the queries required.	5	Hemil Desai
Create a frontend web page to perform CRUD operations on the drivers in a particular store. This will be part of the store owner dashboard.	5	Dinesh Gajwani
Associate the drivers with previous orders of a particular store that they have delivered. Query the database and send the data in JSON format.	5	Dinesh Gajwani

Acceptance Criteria: If this user story is implemented well, the tester should be able to view the drivers for different stores in the store dashboard, manage the different drivers with CRUD operations on the API and see a list of past deliveries performed for a particular store.

User Story: As a store owner, I should be able to assign orders to available drivers - automatically and manually.

Task Description	Estimated Time	Owner
Create the data structure and algorithm for delivery scheduling - First Come First Serve. Explore different services like Redis for creating fast reliable queues.	5	Hemil Desai
Pop the first order and the first available driver, assign the order to the driver and update the database	4	Hemil Desai

and the frontend. Interact with the data store, update the database and send the data in JSON format.		
See the list of available drivers and pending orders - updated in real time. Use ajax requests to handle real time updates in driver and order status.	5	Shivank Tibrewal

Acceptance Criteria: If this user story is implemented well, the tester should be able to efficiently test the algorithm by creating dummy orders using faker and see the scheduling of deliveries in real time on the store dashboard along with the list of drivers and orders being updated in a timely manner.

User Story: As an Admin, I would want to manage drivers (Add, Remove, Modify) into the system

Task Description	Estimated Time	Owner
Construct a simple web page using HTML, CSS and Javascript form where the details of a driver can be added and submitted through a HTML request	6	Dinesh Gajwani
Perform Validation to check if driver already exists/invalid data	2	Dinesh Gajwani
Construct a simple web page using HTML, CSS and Javascript form where the details of a driver can be loaded from the database through a HTML request	6	Hemil Desai
Construct a simple web page using HTML, CSS and Javascript form where all driver in the database are listed through a HTML	6	Dinesh Gajwani

request		
Make changes to an already existing driver, after its details are loaded.	2	Kunal Goyal

Acceptance Criteria: If this user story is implemented well, the tester should be able to add a driver, view all drivers in the system, delete a driver, load and modify the details of a driver.

User Story: As a user(store owner/driver) of DOMS, I should be able to register/sign up for the service.

Task Description	Estimated Time	Owner
Construct a simple web page using HTML, CSS and Javascript form where the details of the user can be added and submitted through a HTML request	6	Shivank Tibrewal
Perform Validation to check if user already exists/invalid data	2	Shivank Tibrewal
Construct a simple web page using HTML, CSS and Javascript form for the admin to view and accept/reject these requests.	6	Kunal Goyal

Acceptance Criteria: If this user story is implemented well, the tester should be able to sign up as a driver/store owner successfully, and as an admin, be able to view and accept requests.

Remaining Backlog

Functional

1. As an administrator, I should be able to:
   a. Locate active drivers on a map
   b. View feedback by users
   c. Add other administrators
   d. View scheduled / completed delivery orders

2. As a driver, I should be able to:
   a. Accept and reject delivery
   b. Log into the DOM mobile application
   c. Update delivery status
   d. See the delivery address on the map
   e. Get the best routes available and an ETA.

3. As a store owner, I should be able to:
   a. Locate working drivers on a map.
   b. View feedback from users

4. As a customer (TBD), I should be able to:
   a. Provide feedback for the service.
   b. Pay tip.
   c. Get an ETA.

5. As a developer (TBD), I should be able to:
   a. Use the tools of the platform through an open API
   b. Use simple HTTP protocols to place orders via the API
   c. Integrate seamless payments.
   d. Get feedback based on the orders.

Non-Functional

1. The application should focus on being user friendly. The application should not have too many buttons on one screen as that can overwhelm the user. Instead, we should make it simple and understandable.
2. The system should have fast response times and efficient latency and bandwidth. The system should have an online response time of no more than 10 seconds.
3. The server must be able to handle concurrent users.

4. The app should aim to have minimum downtime.
5. The API and database should be scalable to handle occasional overloads and several users.
6. The server should be efficient. If deployed on cloud, it should integrate well with services like Amazon AWS or Microsoft Azure.
7. The platform should have a testing mode and a production mode.