Coupon Now

Final Report

Fall 2018/Winter 2019

By:
Zain Mustafa
Rafae Saeed
Dethanone Oudomsouk
Robert Selman

Contents

Overview

CouponNow is a way for local businesses to advertise themselves through digital couponing. Using the website, business owners can register as many businesses or locations as they want and create advertising campaigns with coupons that remain valid from a defined start to end date. Each campaign also has an “interest tag” attached to it so customers can filter what they would like to see.

Customers just have to sign up and download the app, then to find coupons they can choose their interests or check for all coupons within a preferred range. If the user sees something they like, they can save the coupon to be used before it expires. And as the customer begins to use more coupons, the system will learn their preferences and make sure they see the most relevant coupons first. CouponNow makes it easy for people to discover local restaurants, bars, stores, etc. and also find deals.

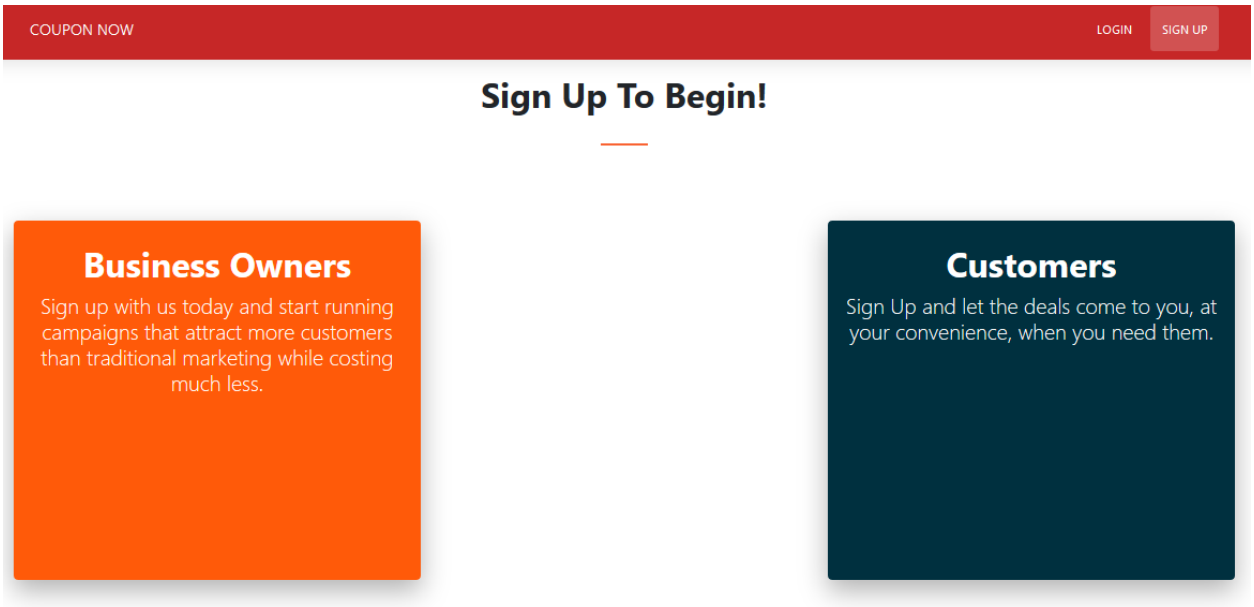


Figure 1: Sign up page for CouponNow

Requirements and Specifications

Functional requirements

Business Owners

- System allows business owners to register and sign in to their dashboards where they will be able to add their own businesses.
- Businesses can be added based on location(s) of the store, the type of the business as well as business licenses ID.
- System validates these new entries based on the given business license. Once verified the business owner will be allowed to start a campaign.
- Each campaign will allow the business owner to set start and end dates for each coupon while keeping track of how many coupons sold in each campaign.
- Business owners will have the option to view analytics regarding each campaign and/or business as well as generate reports.
- Based on the coupon count used for each campaign, our system will set a percentage fees that the owner must pay.
- A payment tab will be set with the amount the owners owes as payable amount.

Consumers

- Register and sign in to the application
- Create a profile which contains information on user interests
- Can set their range for receiving coupons
- Receive notifications for available coupons in the set range
- Manually browse coupons
- Accept/Reject coupons to save them to a list or ignore them
- View accepted coupons list
- Select and use coupon from accepted coupons list
- View (transaction) history for information, e.g. current savings
- Change notification settings

Non-Functional Requirements

- Usability - interface should be simple and easy to navigate
- Security - payments must be secure
- Scalability - the system must be robust so that it can handle large amounts of data

- Availability - availability of coupons should coincide with business hours

Technology Stack

The web application was developed using the MEAN stack:

- M: MongoDB, a NoSQL database that uses JSON-like documents.
- E: Express.js, a backend web framework running on Node.js.
- A: Angular, a frontend JavaScript MVC framework that runs on the browser JavaScript engine.
- N: Node.js, an execution environment for backend JavaScript.

The mobile application was developed using NativeScript:

- NativeScript was used because it works well with Angular Framework and allowed us to integrate our already developed code into the mobile application.

MongoDB was a good choice for us because it allows for flexible development. For a project in its early stage with a short development timeline and schemas that are subject to change, NoSQL gives the developers a lot of power. The lack of traditional relational constraints allowed many changes to be made to schemas as different requirements were discovered. Additionally for future development, NoSQL is considered to be relatively simple to scale due to being able to distribute the data across multiple servers.

Express.js and Node.js were chosen because using JavaScript on the front- and backend simplified development. Also, CouponNow is primarily API-driven with little to no CPU-heavy tasks in its current state which allowed us to take advantage of the asynchronous event-driven style of Node.js for quick development of features.

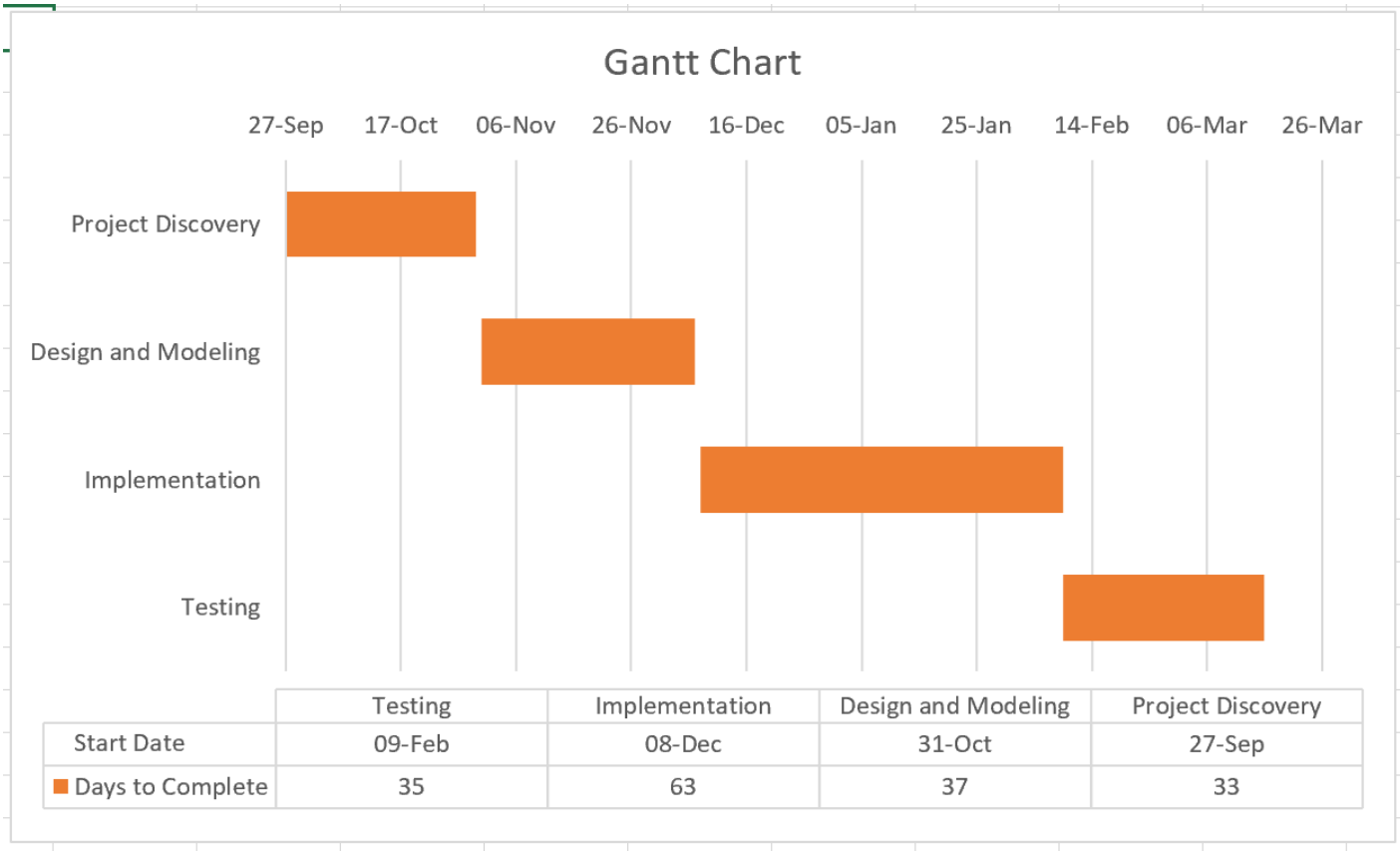
Angular is a widely-used frontend framework with a lot of support and it was a good learning opportunity for our team which did not have much prior experience with frontend frameworks. Its modularity and opinionated nature helped create a well-structured codebase.

Work Breakdown Structure

Project Work Details

Program	Project Discovery	Brainstorm	Accumulate interest from each student in the group
			Brainstorm Ideas that develop interest amoung all the members
			Decide on what the team hopes the end result of the Project will be
		Requirement Discovery	Identify what type of user will the project be directed towards
			Identify what each type of user expects to get out of the application
			Research how the application will accomplish the users needs.
		Requirement Analysis (Consumer)	Identify the assumptions that will be made for the consumer base.
			Write down user stories for the consumers of the product.
		Requirement Analysis (Business Owner)	Identify the assumptions that will be made for the business owners using the product.
			Write down user stories for the business owners using the product.
		Functional Requirements	Identify the functional requirements that need to be accomplished.
			Prioritize the functional requirements.
			Check the requirements for dependencies upon one another.
			List how the functional requirements should be implemented.
	Design and Implementation	Website Design	Design Website Layout using Wireframes
			Design Login Interface
			Implement Login Interface
			Design Business Owner Dashboard Interface
			Implement Business Owner Dashboard Interface
			Design campaign information interface
			Implement campaign information interface
			Design and Implement displaying campaign statistics
			Tag content to be added to website
			Finalize website layout (Color Schemes, Font style, etc)
		Application Design	Design GUI layouts for application
			Implement GUI layouts
			Design and implement Login Interface
			Design and implement Main application screen to display coupons
			Design and implement notifications to display for alerts
			Incorporate GPS tracking of the Business Addresses
			Incorporate heirarchy for most used coupons
	Database Design and Implementation	Business Owner Database	Create a schema for the business owners
			Identify Primary keys needed for different table
			List attributes of each table
			Identify relationships between Business Owner tables and Consumer tables
			Normalize Tables
			Finalize Business Owner Database Design
		Consumer Database	Create a schema for the business owners
			Identify Primary keys needed for different table
			List attributes of each table
			Identify relationships between Business Owner tables and Consumer tables
			Normalize Tables
			Finalize Consumer Database Design
	Testing	Unit Testing	Identify individual units in the source code.
			Test each unit to identify if they can be used.
		Integration Testing	Identify the components in the web app and the mobile app.
			Test if components can communicate with in each other properly.
		System Testing	Identify the user stories that have been implemented in the system.
			Test all the user stories that have been implemented.
		Acceptance Testing	Select a potential user to test the application.
			Get feedback from the potential user.
		Bugs	List out the bugs found in the system.
			Iteratively fix the bugs found.

Gantt Chart



Implementation

The team decided that everyone should work full stack to prevent blocking during the short development time. Over the ~3 months, we divided the work as follows:

Work Distribution

Dethanone Oudomsouk

- Customer Setup
  - Created web interface for Customers to create an account, add and view their preferences
- API's for Customers
  - Implemented all API's facilitating the storage and retrieval of Customer data from MongoDB database
- Load and Bug testing

    - Conducted usability tests to ensure the application met the project requirements
    - Conducted load and stress testing using Locust

Robert Selman

- Business Owner Setup
    - Created web interface for Business Owners to create an account, add businesses and add locations for their businesses
- API's for Business Owners
    - Implemented all API's facilitating the storage and retrieval of Business Owner data from MongoDB database
- Load and bug testing
    - Conducted usability tests to ensure the application met the project requirements

Rafae Saeed

- Team Lead
    - Responsible for code integration and maintenance of master branch in repository
- Campaign Setup
    - Created web interface for Business Owners to create campaigns
- API's for Campaigns
    - Implemented all API's facilitating the storage and retrieval of Campaign data from MongoDB database
- Resolving Bugs
    - Responsible for debugging and resolving bugs found during testing

Zain Mustafa

- Initial Setup for Web Application
    - Designed project skeleton and structure of web application
- API's for Signup/Login
    - Implemented all API's facilitating the storage and retrieval of Login and Signup of web
    - Implemented all mobile API calls.
- Developed the Mobile Application

Development and Testing

We prioritized the development of the web application first. This was the largest chunk of the development time because everyone had at least one new technology to learn, whether it was the language itself (e.g. JavaScript) or the stack.

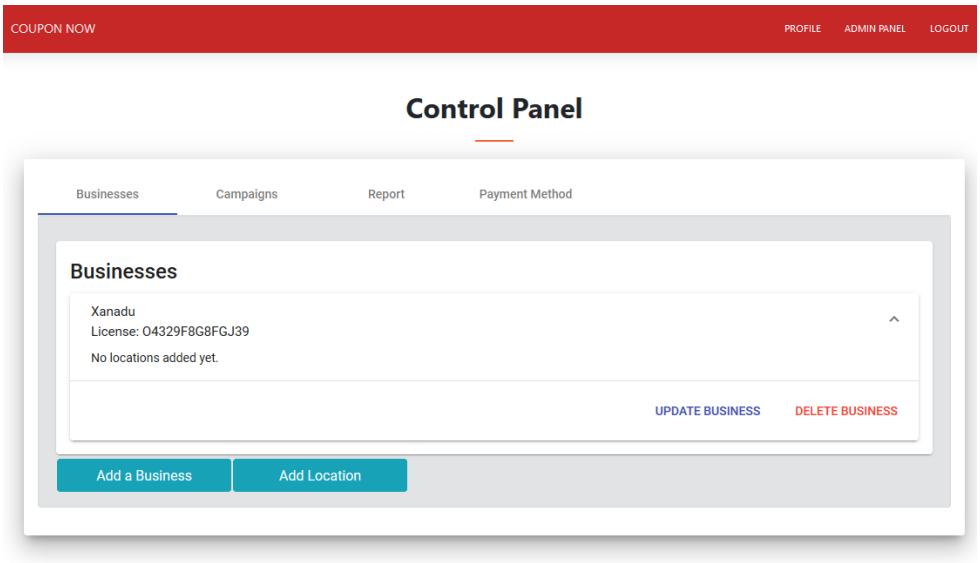


Figure 2: Business owner control panel showing businesses list

Following the website, the mobile application was developed. Many of the API calls developed for the web app were also reusable on the mobile app, so this greatly shortened this phase of development.

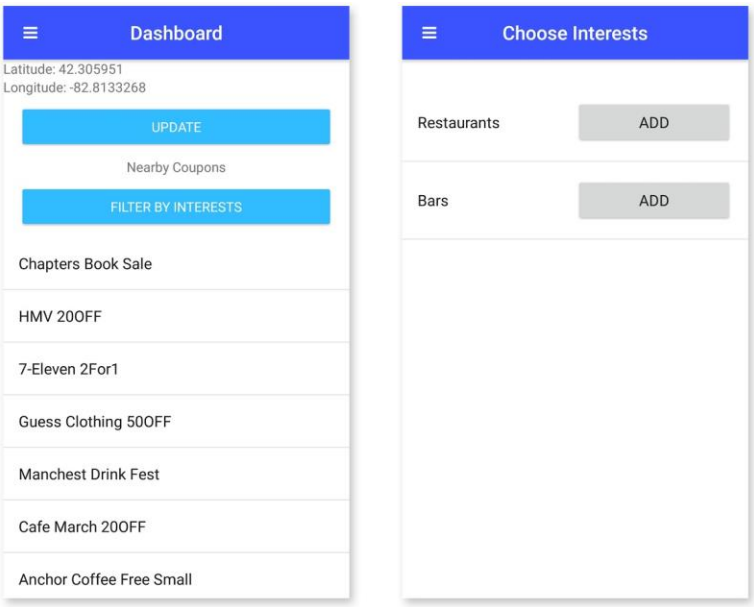


Figure 3: Dashboard and Add Interests page for the mobile application

We also decided to do load testing after development, to find potential areas for future improvement. The following are the results of approximately 5 minutes of load testing using Locust. There were 1000 users in this simulation, spawned at a rate of 20 per second and attempting a login upon spawning. *List Businesses* was called every 5-20 seconds and *Get Coupons* was called every minute.

Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)
POST	List Businesses	1533	50	54	1559	31.586170196533203	52523.8835811615
POST	Business Owner Login	91	13	44000	42646	358.80112648010254	72639.80674743652
POST	Customer Login	909	119	48000	42946	169.51513290405273	72814.59259986877
GET	Get Coupons	3831	364	230	3426	51.969051361083984	44688.4183883667
	Total	**6364**	**546**	**220**	**9182**	**31.586170196533203**	**72814.59259986877**

Figure 4: Load testing chart

Failure Rates

- List Businesses ≅ 3.3%
- Business Owner Login ≅ 14.3%
- Customer Login ≅ 13.1%
- Get Coupons (Customer) ≅ 9.5%
- ***Overall ≅ 8.6%***

Response Time:

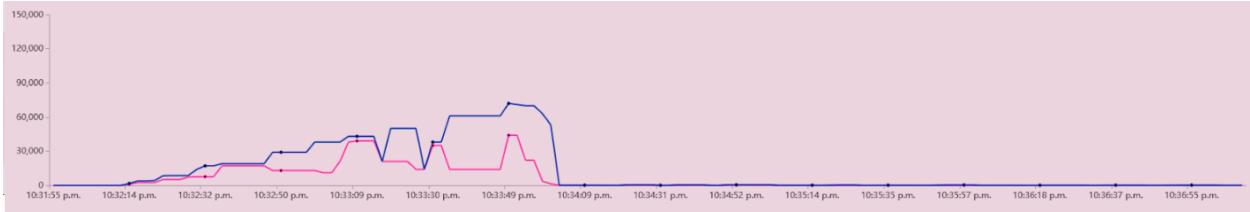


Figure 5: Response time graph

Number of Users:

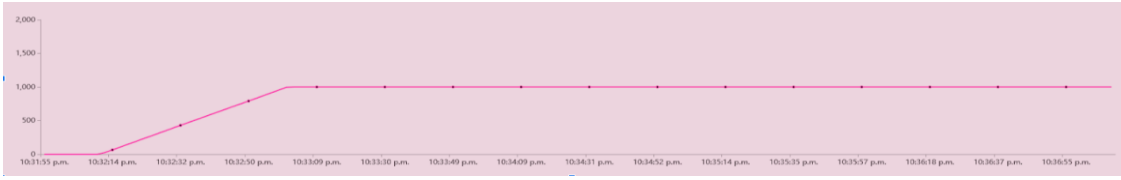


Figure 6: Number of users graph

We can see that the logins had the highest failure rates and once the logins are complete, the response times of the other requests stabilize significantly. The AWS machine we used for hosting is relatively low-powered, but the load testing suggests

that there may be room for optimization in the login code if this app were to be scaled up in any way.

Challenges

During the implementation of the project our team faced several challenges. The biggest challenge the team faced was time restrictions. CouponNow is a functional prototype that implements most of our initial requirements but there are still features that we did not have time to fully implement. We would have also liked for more time to more thoroughly test our application and use those results to optimize performance. Due to our time constraints we focused on usability, load and stress testing to ensure we would have a usable application at the end of our project.

Another challenge the team faced was working with unfamiliar technology. This was intentional because we decided at the beginning of the project to learn a new technology stack. The difficulties of this challenge were mitigated by planning time in advance for team members to learn new technologies and strong communication between team members meant that obstacles could be solved collaboratively as they appeared.

An example of this was finding a way to efficiently query business locations given a mobile users location. An unoptimized query would need to check every business location in the database and this would have a large impact on the performance of our application, especially as the number of users and businesses in our database increased. Our solution was to store business locations as GeoJSON objects and to implement a Geospatial Index on the location field. This allows for much more efficient location queries and will improve the scalability of our application.

Setup and Deployment

Web Application

1. In order to run the web application on http://localhost:4200, these steps can be followed:
2. Install NodeJS (v11.*): https://nodejs.org/en/
3. Install Angular CLI (v7.*) by running the command: **npm install -g @angular/cli** : https://cli.angular.io/
4. Install MongoDB (v4.*): https://www.mongodb.com/

5. Install Git: https://git-scm.com/
6. Clone the repository: https://bitbucket.org/saeed11b/couponnow/src/master/
7. Install nodemon from the command line: **npm install -g nodemon**
8. Install local dependencies in *couponnow/WebApp/*: **npm install**
9. Run MongoDB if it is not already automatically running
10. Run the frontend server in *couponnow/WebApp/*: **ng serve**
11. Run the backend server in *couponnow/WebApp/backend/*: **nodemon server.js**
12. To create a production instance of the project run the command : **ng build --prod**

Mobile Application

1. To run the mobile application on your android device, these steps can be followed:
2. Install NodeJS (v11.*): https://nodejs.org/en/
3. Install Angular CLI (v7.*): by running the command: **npm install -g @angular/cli** : https://cli.angular.io/
4. Install MongoDB (v4.*): https://www.mongodb.com/
5. Install NativeScript: by running the command: **npm install -g nativescript** : https://docs.nativescript.org/angular/start/quick-setup
6. Install Git: https://git-scm.com/
7. Clone the repository: https://bitbucket.org/zainmustafac/couponnow-mobile/src/master/
8. Install local dependencies in *couponnow/WebApp/*: **npm install**
9. Make sure the server is running from the web application repository (follow the instructions above for the web application).
10. Initiate debug mode from your android phones settings menu.
11. Connect your phone to your computer via a USB cable.
12. To start the application on your phone run the command : **tns run android**, from the terminal while in your root directory of where the code is.
13. To build a .apk file of the mobile application run the command: **tns build android**

AWS Setup

1. Create an Amazon Web Services account: https://aws.amazon.com/.
2. Through your console start a EC2 instance running ubuntu 18.04.
3. Connect to your EC2 instance using your local terminal.
4. Install NodeJS on your EC2 instance: **sudo apt install nodejs npm**
5. Install Angular on your EC2 instance: **npm install -g @angular/cli** *(optional)*
6. Install MongoDB on your EC2 instance: **sudo apt-get install -y mongodb-org**
7. Install http-server package from npm: **npm install http-server -g**

8. Install pm2 package from npm: **sudo npm install -g pm2**
9. Clone the backend folder to any folder in your EC2 instance from the git repository: https://bitbucket.org/saeed11b/couponnow/src/master/
10. Copy the build folder of your web application to the **/root/public/** directory in your EC2 instance.
11. Start the MongoDB in your EC2 instance: **sudo systemctl start mongodb**
12. Start the backend server instance by going into the directory you clones the backend folder in and run the command: **pm2 start "node server"**
13. Start your http-server by running the command: **pm2 start http-server -- -p 4200**

Summary

Overall, the team was able to complete the majority of the initial scope that was planned and the general concept of digital coupon delivery works. Some features were not able to be completed during the given time for this project, however. There is no payment system implemented yet nor is there a way for a profit to be made from the application. Also from the customer side, there is a primitive system in place to help users build up their preferences so that they see their most relevant interests first, but there is not a strong enough model to be able to collect data to generate reports and analyse consumer data/trends. This could be important in the future to make the application more profitable.

Additionally, at a lower and more development-focused level, the project could benefit from more rigorous testing and in general some more DevOps training and application. There are no end-to-end tests or unit tests, which could greatly reduce headaches for the developers. It can take very long to manually test features using the website and it can take even longer to discover bugs that are introduced into the product as a result of new features being improperly implemented. Then continuous integration system could be used to automate these tests and perform code analysis (e.g. additional linting for coding standards). Now, any code can be merged into the master with errors and other issues like unresolved merge conflicts.

Appendix

Project Repositories

Web Application: https://bitbucket.org/saeed11b/couponnow/src/master/
Mobile Application: https://bitbucket.org/zainmustafac/couponnow-mobile/src/master/

Relevant documentation

- MongoDB - https://docs.mongodb.com/
- AngularJS - https://angular.io/docs
- Express.js - https://expressjs.com/
- Node.js - https://nodejs.org/en/docs/
- Mongoose - https://mongoosejs.com/docs/guide.html
- NativeScript - https://docs.nativescript.org/
- Google Maps API - https://developers.google.com/maps/documentation/
- LocationIQ - https://locationiq.com/docs