# Laboratory 9

Title of the Laboratory Exercise: Implementation and Testing

1. Introduction and Purpose of Experiment

Students will use JavaDoc and similar standards to ensure good documentation and maintainability of code

1. Aim and Objectives

Aim

* To implement a given design with appropriate coding standards

Objectives

At the end of this lab, the student will be able to

* + Apply industry standard coding standards
  + Use automatic documentation tools
  + Create maintainable code

1. Experimental Procedure

* Work in teams of 7 students
* Each team should read the class diagram and identify objects, interactions and states of objects
* Each team will then split workload and develop classes individually.
* Each individual will then write their lab manual, documenting their observations

1. Calculations/Computations/Algorithms

**Shop functionality**

**Shop Front Page**

* Featured Products with link to corresponding product detail page
* Features for «New Arrivals», «On Sale» and «Best Rated» Products

**Products**

* Products fetched from FireBase backend and cached for future requests
* Sorting: Products can be sorted by date created, price and name
* Products can be viewed in a grid or a list view
* Products are shown paged via a PagingService
* When logged in as a user with adminstrative rights, additional buttons are shown for product CRUD operations

**Product CRUD**

* Add a new product
* Edit existing product
* Delete existing product
* Images handled with Firebase Storage

**Cart**

* Products can be added from the list/grid View or product detail view
* Cart is handled via a CartService
* Adding the same product multiple times, increases the amount in cart
* Cart has a dedicated cart page and is visible as a dropdown widget as well
* Quantity of each cart item can be adjusted via cart page
* Cart can be cleared at once
* Single products can be removed from cart
* Subtotal and Totals will be calculated on the fly
* Link to Checkout is available from both carts

**Checkout**

* Prefill fields, if user is already logged in
* Enter Address, Shipping Method and Payment Data with Validation
* Show review of the order before final submit
* When submitting a order, OrderService creates a new Order linked to the user
* Anonymous Orders are possible too, in that case OrderService creates a new anonymous order
* Order summary is shown in the sidebar during the checkout process

**Authentication**

* Checkout: As registered user / guest
* Sign up: Create user account
* Log in: General login or during checkout
* Role based authentication

**Account**

* Create new shop user accounts
* Login with existing user account
* User Profile, Email, Password, Firstname, Lastname are updateable via account page
* Order history is visible to logged in users
* Role base authentication via Firebase, roles can be assigned to users like isAdmin

**Orders**

* Checkout process generates Order for registered user or guest
* Order / Confirmation Email for Shop/User/Guest
* Orders can be viewed by logged in user

1. Presentation of Results

Application Screenshots:

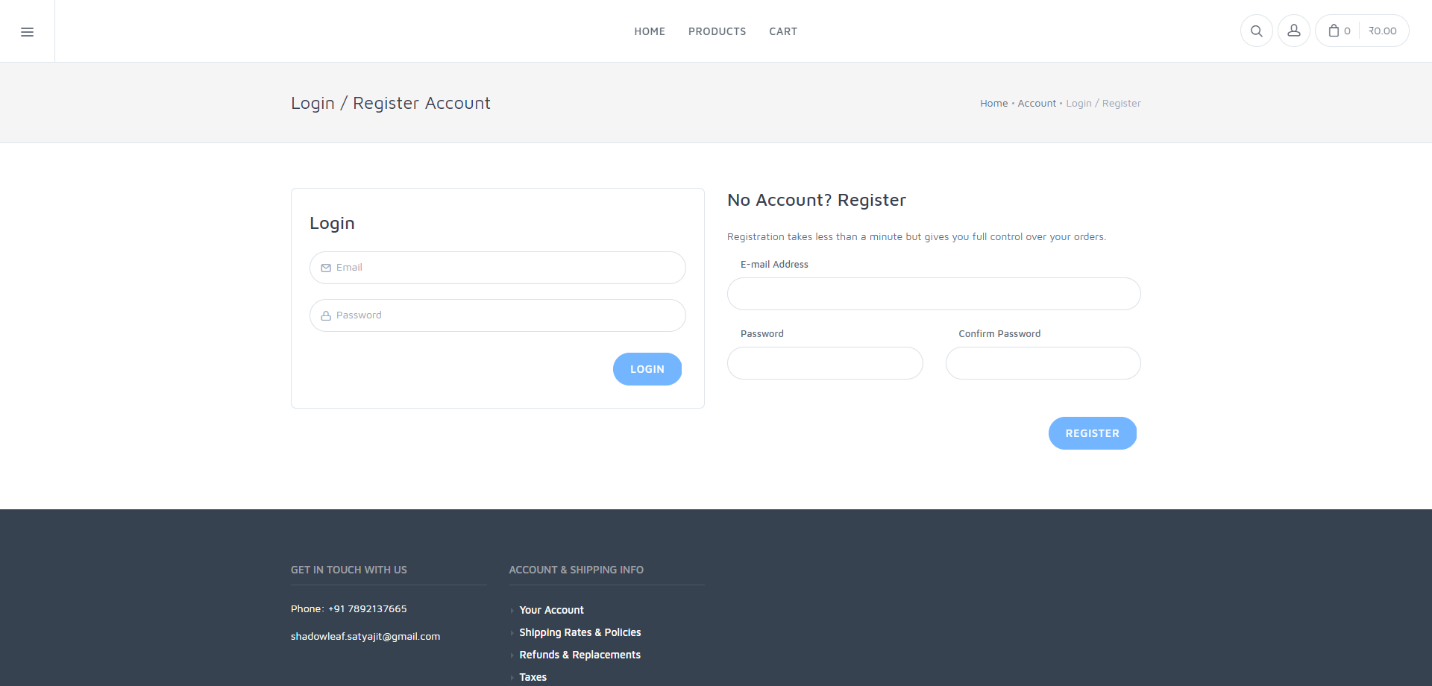
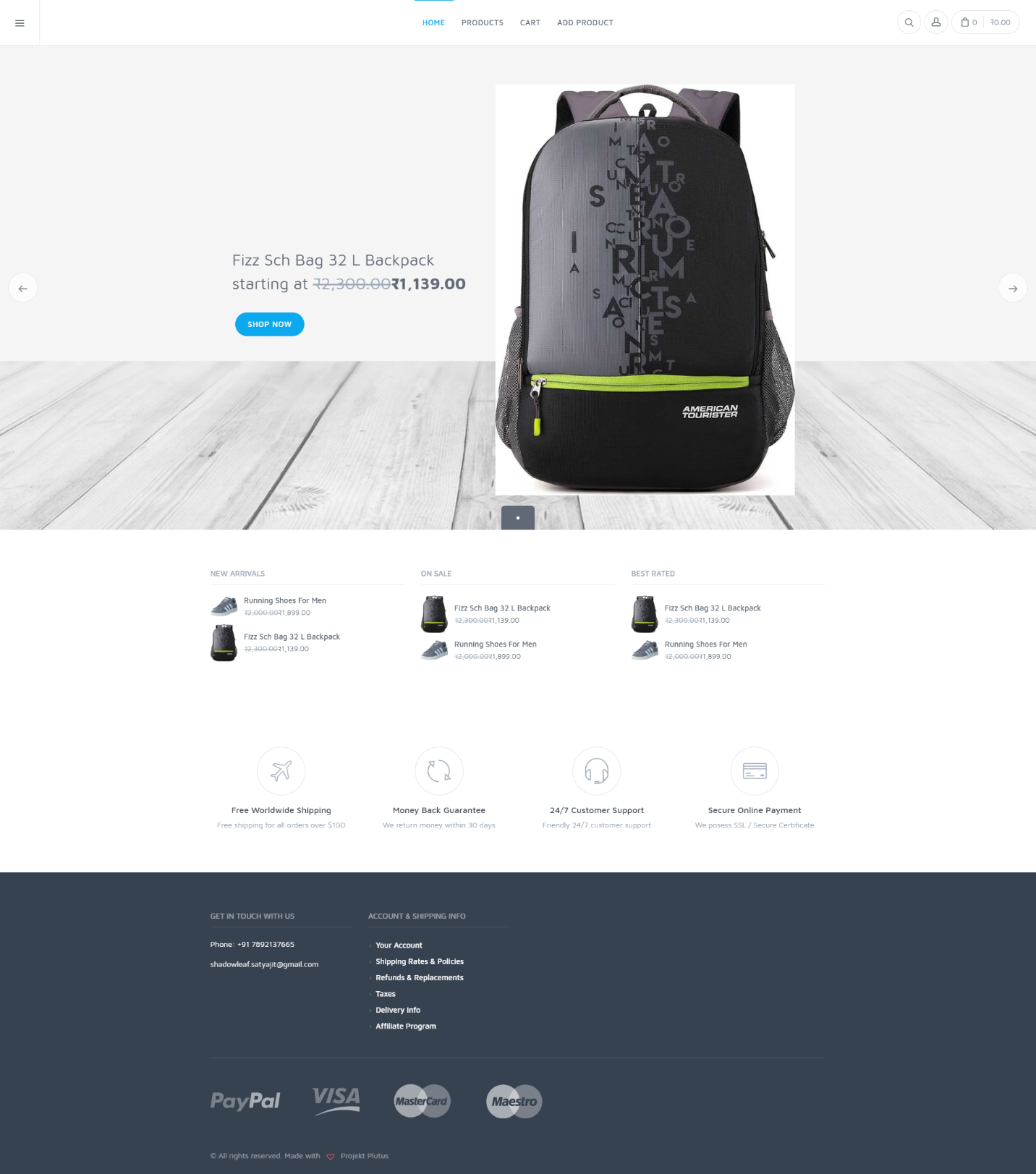
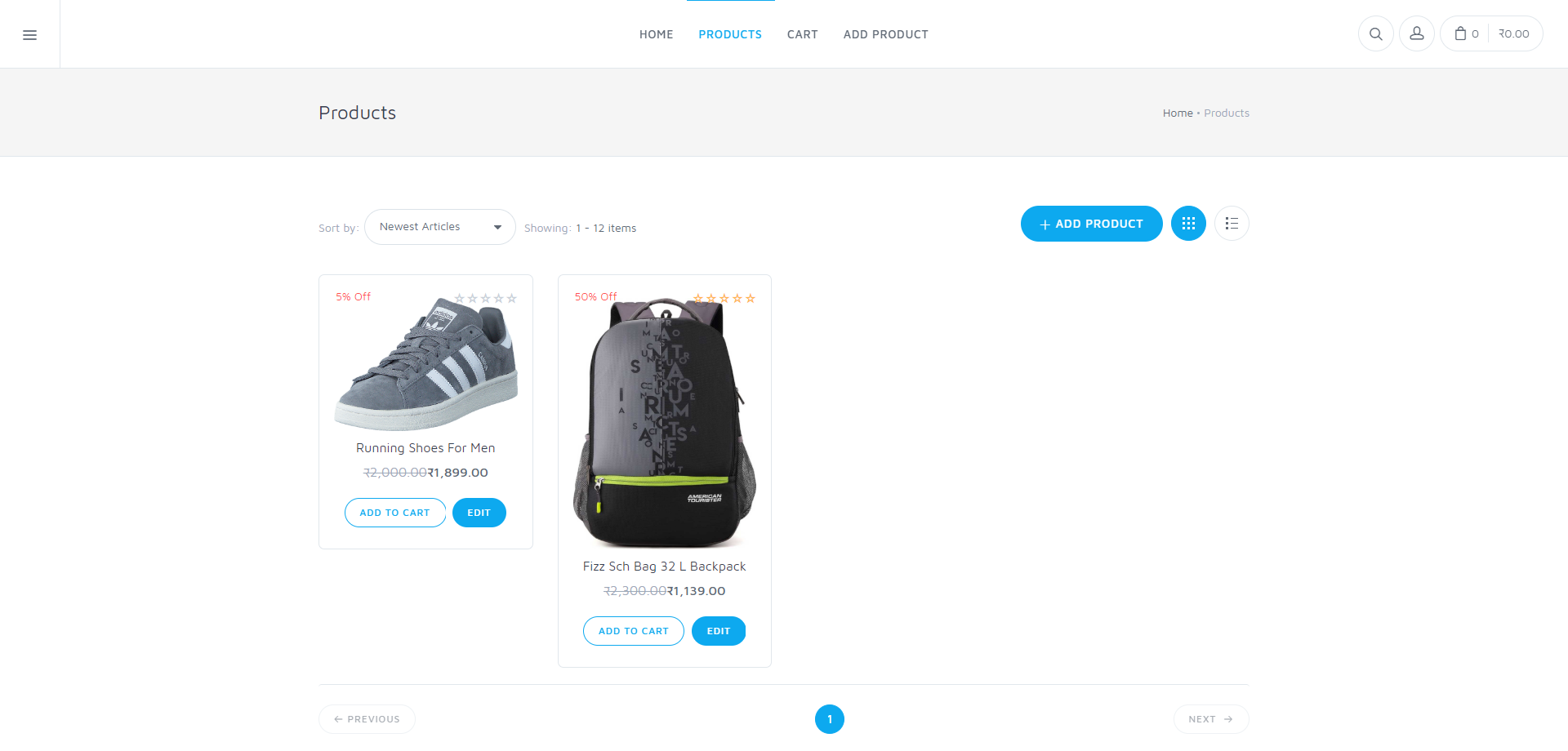
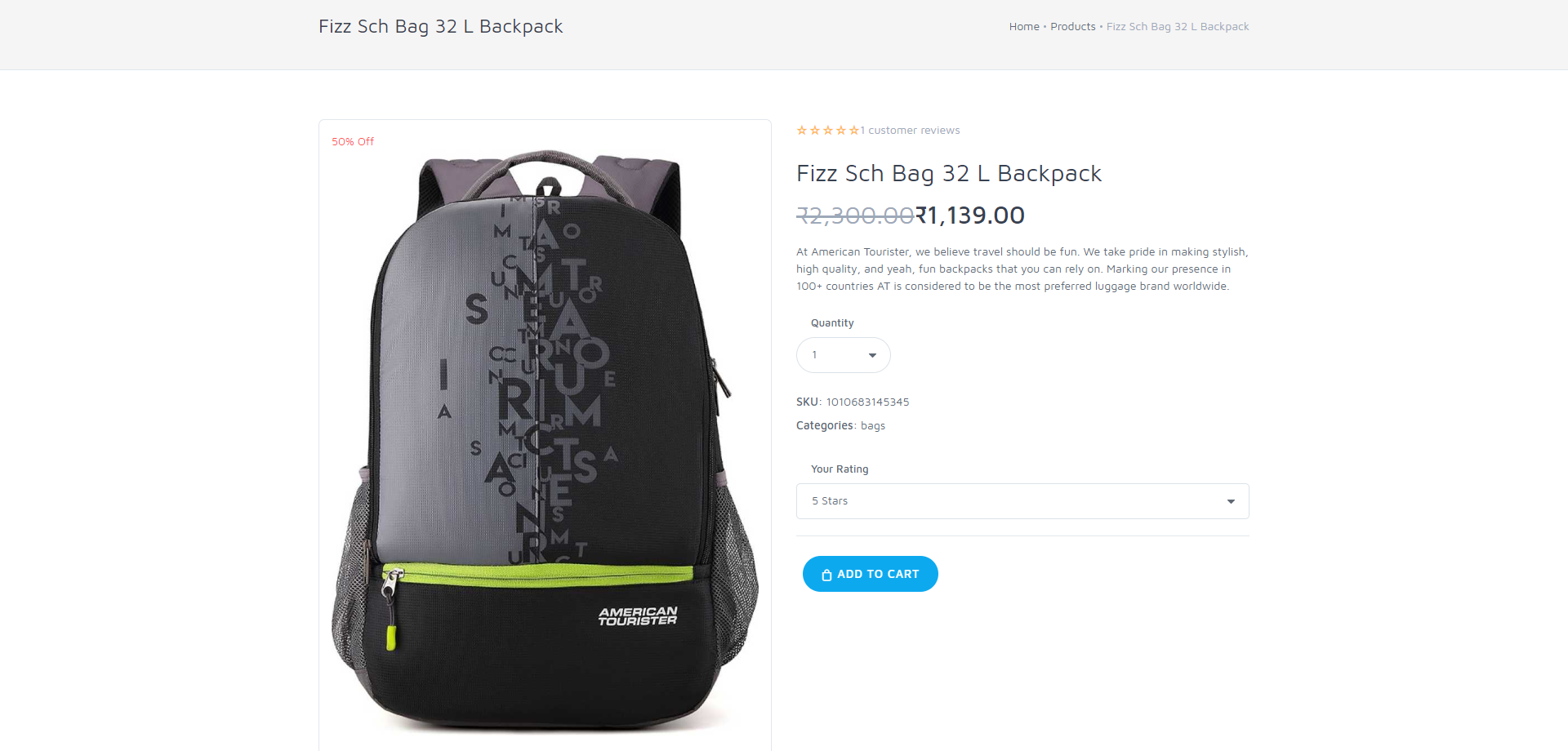


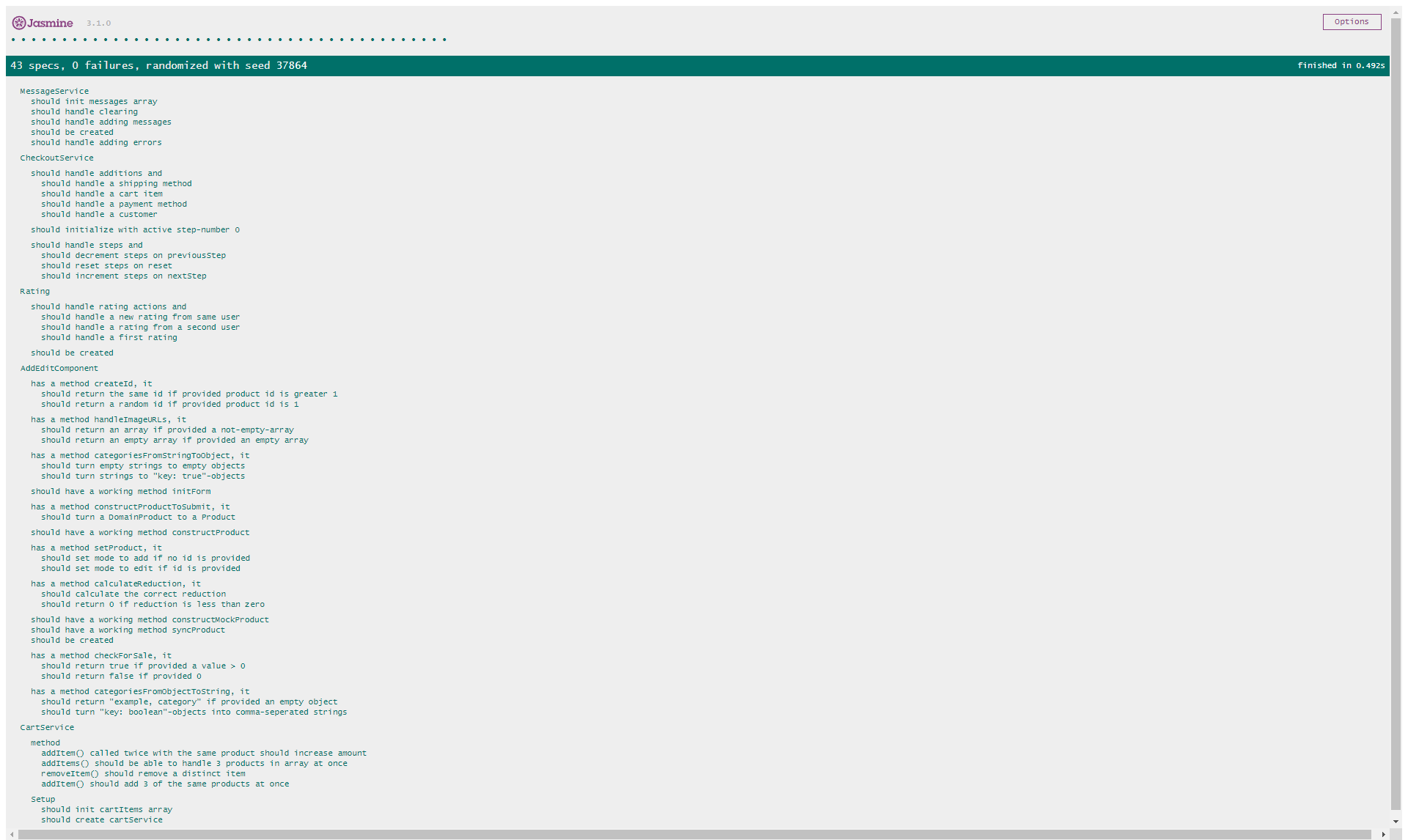
Figure 0‑1 Login and Register Testing







Karma Testing (Test Runner for JavaScript)



1. Analysis and Discussions

It is an essential practice to properly document the code, as well follow code conventions. Special attention is given to the developer’s documentation. It is helpful for development, maintenance and knowledge transfer to other developers. Description of code gives a clear idea of what is happening.

The technologies used here are Angular 7, TypeScript, JavaScript, Firebase, Karma (for testing), HTML, CSS and SCSS.

The code that was written follows the Model-View-Presenter pattern where the HTML/CSS provides the raw front end for it and TypeScript makes this front-end dynamic, the back end is Firebase and any database queries are made here, any changes made from the user to the view makes changes to the model, that alters the database. Although any design pattern of Model-View type can be used such as the Façade design pattern.

Testing for the same was done using Karma which is a framework to test JavaScript Code. The main goal for Karma is to bring a productive testing environment to developers. The environment being one where they don't have to set up loads of configurations, but rather a place where developers can just write the code and get instant feedback from their tests. Because getting quick feedback is what makes you productive and creative.

1. Conclusions

In this lab, we implemented our design and carried out testing of an independent and dependent function. The test cases were designed and carried out using black box testing. The test cases need to be extensive and thorough and testing is a very important activity in software development.

1. Comments

1. Limitations of Experiments

Large scale applications with several lines of code take time to document.

2. Limitations of Results

Documentation requires time and effort, but it saves time spent on reading and correcting code later on.

3. Learning happened

Implementation process, good coding and documentation practices and their need.

4. Recommendations

|  |  |  |
| --- | --- | --- |
| **Component** | **Max Marks** | **Marks Obtained** |
| **Viva** | **6** |  |
| **Results** | **7** |  |
| **Documentation** | **7** |  |
| **Total** | **20** |  |

None.