**Project Brief**

**Inventory and Pricing Manager**

Release: Final

Date: 16 Apr. 15

Authors: Shreyas Bedekar

# Report History

## Revision History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Revision date** | **Author** | **Version** | **Summary of Changes** | **Changes marked** |
| 14/04/2015 | Shreyas Bedekar | 1.0 | 1st Draft |  |
| 16/02/2015 | Shreyas Bedekar | 1.1 | Final |  |

**Table of Contents** Page

1 Report History 2

1.1 Revision History 2

2 Project Brief Purpose 4

3 Project Definition 4

3.1 Objectives 4

3.2 Requirements 4

3.3 Deliverables 4

3.4 Constraints 4

4 Technical Design 5

4.1 Front-End 5

4.2 Back-End 6

4.3 Data 6

5 Installation 6

5.1 With Node and Mongo 6

5.2 Local File System without a web server 8

6 Navigation 9

6.1 Home 9

6.2 Edit (Add new record) 11

6.3 All Data 12

6.4 Stats 13

# Project Brief Purpose

UI Assignment to assess following

A. Optimal solution

B. Code efficiency and modularity

C. Coding best practices

# Project Definition

## Objectives

The overall objective of this project is to build a single page HTML5 web application for Inventory and Pricing Management on Angular JS

## Requirements

### Home page should have a search box where user can search for product names. User should be shown an Auto Suggestion based on what he types in the search box.

### When the user hits enter show the correct product details below. User should be able to edit the product name and selling price only

### Product details will be ID, product name, cost per unit, selling price per unit , no of units available

### User will be able to edit product name, and selling! price only rest of the fields are un editable. On Hitting enter in the input field data should be saved

### Validation : Product name should be alpha numeric character !and selling price is numeric and cannot be less than cost price

### There should be another tab on the home screen called “EDIT”. User should be able to add new products !from this screen.

## Deliverables

* Submit project solution by Zip or checking into github as a repo.

## Constraints

### Browser Compatibility

* Works only on HTML5 supported modern browsers due to Angular JS’ compatibility
  + Supported Browsers
    - IE 10+ (For all features)
    - Chrome (from web server only)
    - Safari
    - Firefox
* Using solution statically (from file system) on Chrome and Opera will not work due to file XHR restrictions applied by default by Chrome and Opera
  + On Mac:
    - open /Applications/Google\ Chrome.app/ --args --disable-web-security
  + On Windows:
    - chrome.exe --disable-web-security

# Technical Design

## Front-End

### Javascript Libraries

1. Angular JS (v1.3)
2. jQuery (v1.10)
3. HighCharts(v1.2)

### CSS

1. Twitter bootstrap for device compatibility
2. font-awesome for vector icons

**4.1.3 App Structure**

* **static**– Contains all static js/css/html
  + **components** – all resusable components
    - **buttons** – round shaped buttons (nav buttons)
    - **form** – Add/Edit forms
    - **grid** – All data view
    - **search** – auto complete remote/local search
    - **stats** – stats view
  + **data** – data to be consumed by app (pushed in mongo db by Node on first run)
  + **resources** – style resources for app
    - **css** 
      * app.css – application style sheet
      * bootstrap.css – open source twitter bootstrap css for device support
      * font-awesome.css – open source vector icons css
    - **fonts** – fonts used by font-awesome stylesheet
  + **app**.**js** – Angular Module definition
  + **index**.**html** – app entry point
  + **services**.**js** Angular factory/services
  + **utilities**.**js** – Non angular utility methods

## Back-End

* **Node.js** 
  + **Node Modules:**
    - **body-parser** – REST request parsing
    - **express** - REST
    - **method-override** - REST
    - **mongodb** – Mongo DB connection
    - **morgan** – Request logging

## Data

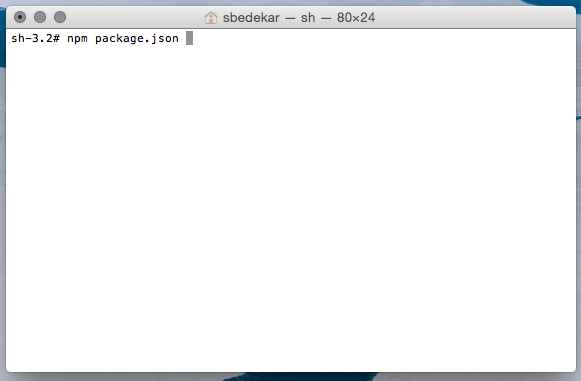
* **Mongo DB**
  + Collection: Products
  + Document Structure
    - ID – Unique record identifier
    - ProductName – Name
    - Price – Selling Price of product
    - Cost – Cost of product
    - Quantity – Quantity in stock
* **Data**
  + **Data is assumed and created only for this assignment**
  + **Not based on any real data**

# Installation

## With Node and Mongo

### Navigate to app folder

**RUN** npm package.json to install all node dependencies

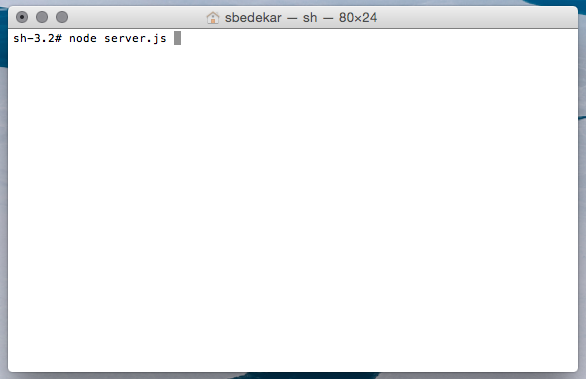


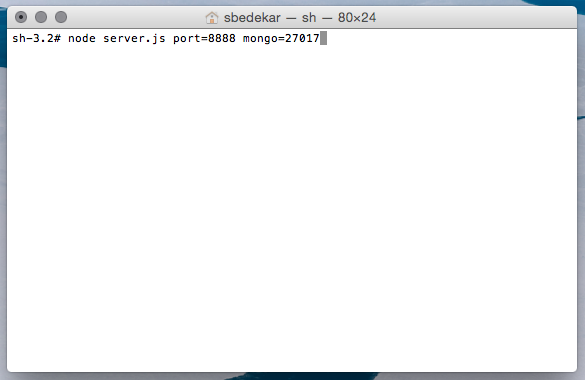
### Make sure Mongo DB is installed and running

**On the first run Node will push data.json in Mongo**

IF Mongo is not running on default port **27017,** please note the port number

### RUN node server.js



* Server will start with default port no 8888 and try to connect to Mongo on port 27017
* IF it is required to give different port numbers for server and mongo use command line arguments
  + port= (for server port)
  + mongo = (for mongo port)
* 
* Open browser and go to localhost:8888
* Home page of app should load

## Local File System without a web server

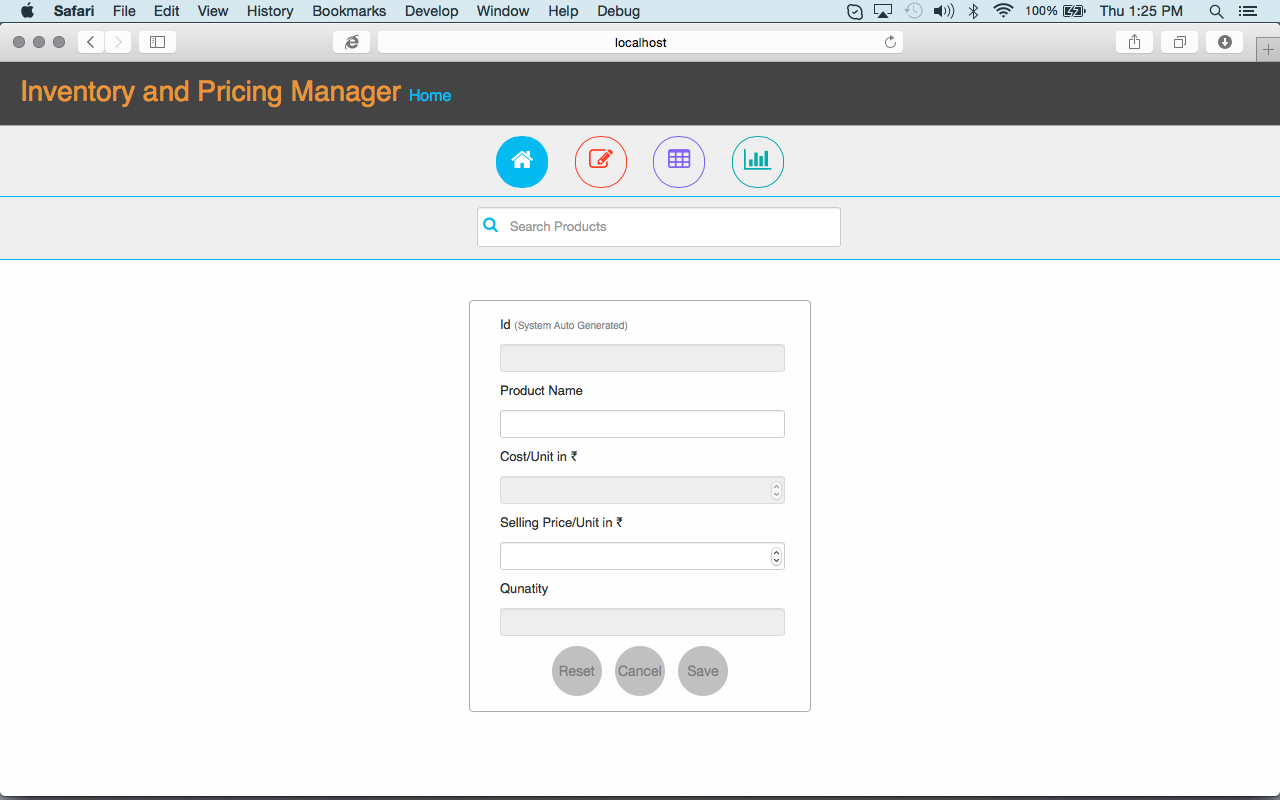
## Open index.html to run locally without any web serve

In case the app is running locally then it will use localstorage (on available browsers) OR local data

If Browser supports local storage then changes would be retained until cache is cleared or localstorage is cleared

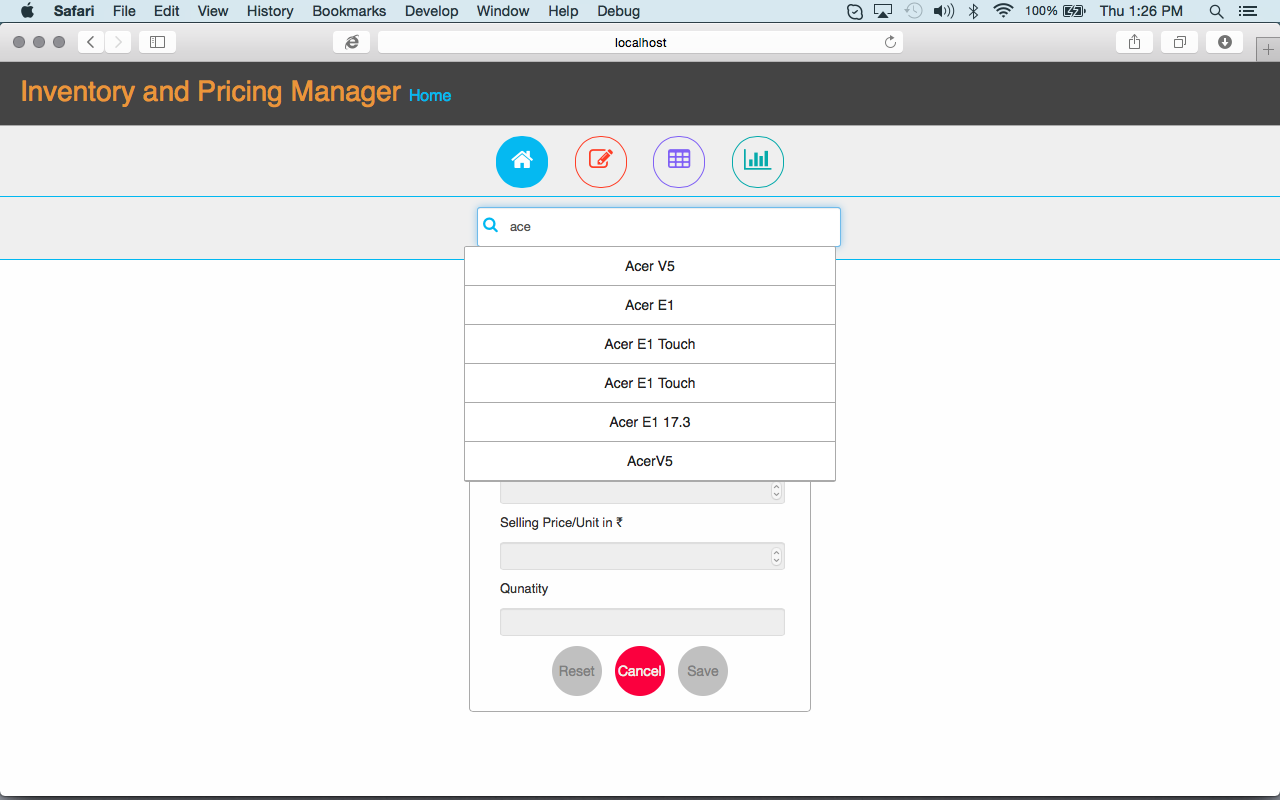
# Navigation

## Home

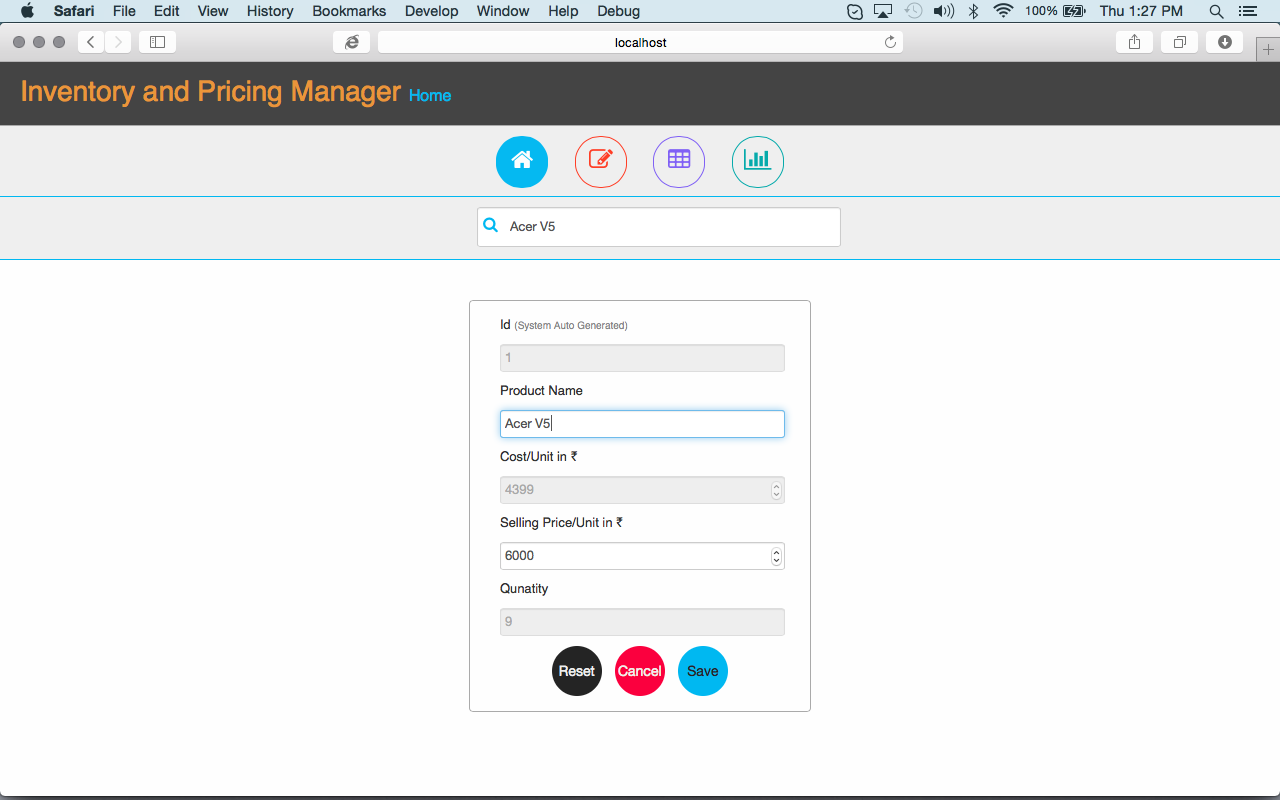


### Search

Type in the Search Products box to search for a product



### Select required product from the Options to edit



### Make required changes and Click – Save Or Hit Enter key to save

### Click Reset to undo any changes made

### Click Cancel to clear search

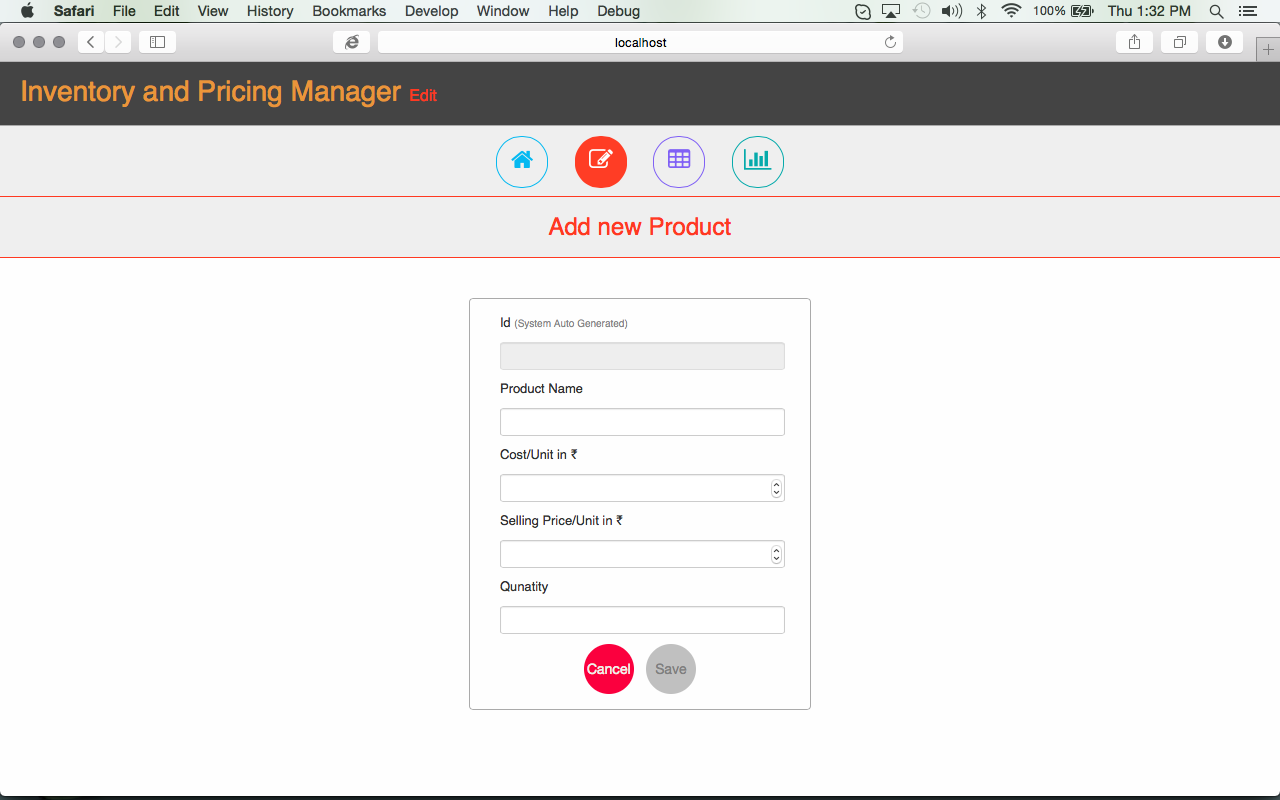
### Product Name will only accept Alpha Numeric values without any special characters

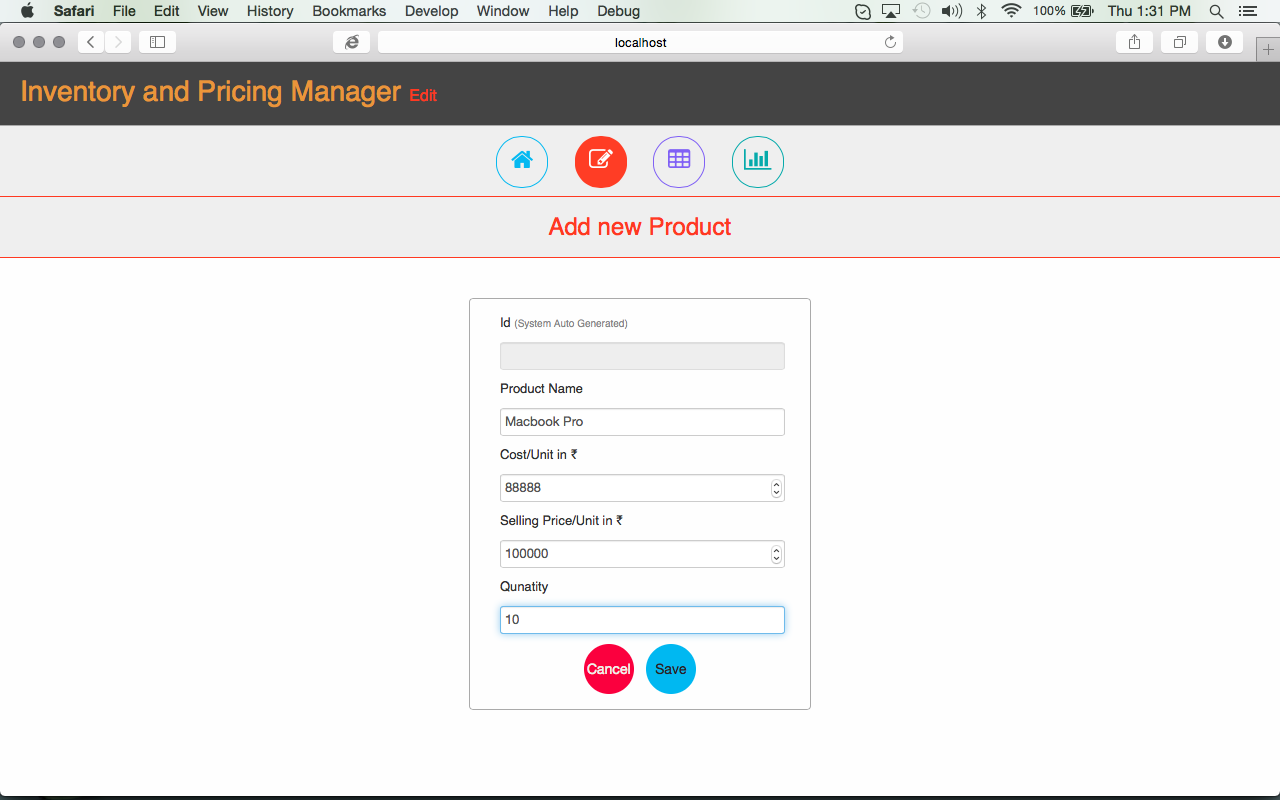
### Price and Cost and Quantity will only accept numeric values

## Edit (Add new record)

### Click Edit tab

### Fill all the required fields

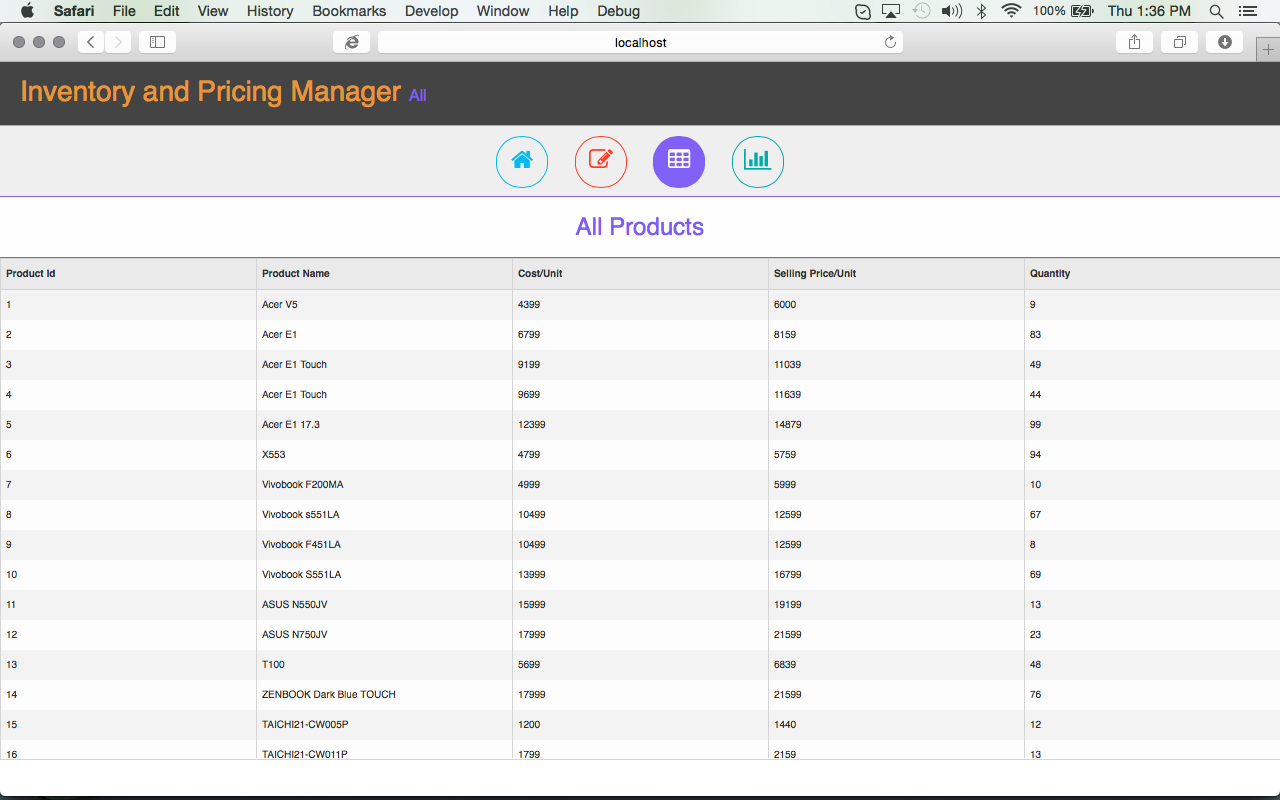




### Click Save OR Hit Enter to add new record

## All Data

Click Grid Tab to view all products



## Stats

Click Stats tab to view some product related stats

# 