

# **A WEB BASED OFFICE MARKET**

CS 297 Project Report  
Presented to  
Dr. Christopher Pollett  
San José State University

By  
Manodivya Kathiravan  
May 2016

# INTRODUCTION

This report describes preliminary work toward my CS297 project concerning a web based office market. People who work in an office often have different pools of resources that they want to exchange. They want to trade their resources/work(seller) with a person who wants that particular resource(buyer) and in return get another resource the buyer offers. For example, a person having an on call on a holiday might want to exchange the on call with another person who is willing to work on that day. In return that person can take up his technical support for some other day. As another example, professors willing to trade their course credits and travel allowances. In cases like this we need an open market place where we can trade with other people the resources we have without meeting in person.

I was working to solve this problem using a web-based solution. My primary focus was on using auctioning systems. Auctioning systems can be used in this kind of scenario to bid for a particular resource or item and then select a winner based on either the highest offer or the one with the lowest offer (reverse auctioning system). The following are the deliverables that I have done this semester to understand the essence and the working of auctioning systems. I will now discuss the organization of the rest of the paper.

The next section describes my deliverable one. For the deliverable one I had created a simple application using Angular JS, PHP and MySQL. The goal of this deliverable is to understand the basic Angular JS working architecture and how it communicates to the database using PHP as the server side language.

In the third section I discussed about my deliverable two. I had created a PowerPoint presentation of my literature survey on auctioning system. This deliverable helped me understand the different types of auctioning system and mathematical logic behind how auctioning systems function.

In the fourth section I have described what I did for my third deliverable. In this deliverable, I developed the UML design for the web application using StarUML and determined the primary use cases. This gave me an overall view on how any auctioning system must function.

The fifth section contains deliverable four, in which I implemented a simple auction system where the user can bid for an item and the system selects the winner after the closing time for the auction. The system contains login, SignUp and logout functionality. This deliverable helped me to code well in AngularJS for a simple auction system. The last section contains the conclusion and my goals towards CS298. This report includes a detailed analysis of my findings.

## **DELIVERABLE 1: A Simple form using Angular JS with database connectivity**

For deliverable 1, I designed a simple Sign Up form with fields username, password and email address using AngularJS. The main motivation of this deliverable was to understand the architecture and working logic behind angular. I used MySQL for my database and PHP as my server side language. When you click on Sign Up button after entering the username, password and email, the angular form uses post method to pass the user entered details to PHP program.

The PHP program sets up the database connectivity and checks if the values are already present in the database. In case if user does not exist, it creates a new user. If user is already present it shows an error message "User already exists".

### **1.1 An Overview On AngularJS**

**Angular JS** (commonly referred to as "**Angular**" or "**Angular.js**") is an open-source web application framework mainly maintained by Google and by a community of individuals and corporations to address many of the challenges encountered in developing single-page applications. It aims to simplify both the development and the testing of such applications by providing a framework for client-side model–view–controller (MVC) and model–view–view model (MVVM) architectures, along with components commonly used in rich Internet applications.

AngularJS directives allow the developer to specify custom and reusable HTML-like elements and attributes that define data bindings and the behavior of presentation components. Some of the directives that I used for are:

#### **ng-app**

Declares the root element of an AngularJS application, under which directives can be used to declare bindings and define behavior.

#### **ng-bind**

Sets the text of a DOM element to the value of an expression. For example, `<span ng-bind="name"></span>` displays the value of 'name' inside the span element. Any change to the variable 'name' in the application's scope reflect instantly in the DOM.

#### **ng-model**

Similar to ng-bind, but establishes a two-way data binding between the view and the scope.

#### **ng-controller**

Specifies a JavaScript controller class that evaluates HTML expressions.

## ng-repeat

Instantiate an element once per item from a collection.

## ng-show & ng-hide

Conditionally show or hide an element, depending on the value of a Boolean expression. Show and hide is achieved by setting the CSS display style.

## ng-view

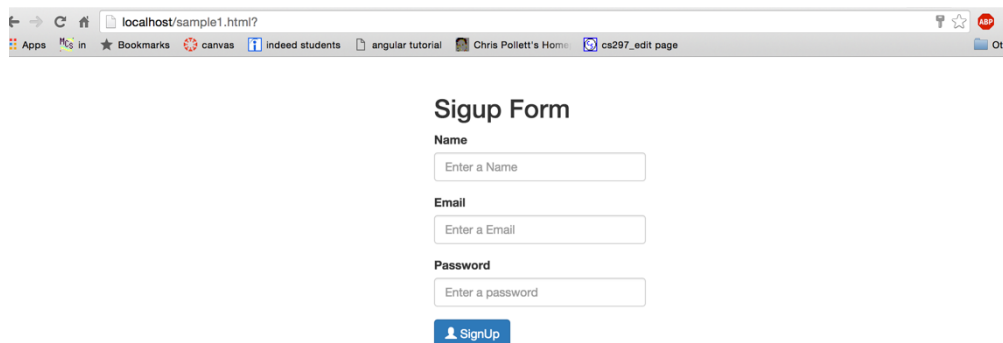
The base directive responsible for handling routes that resolve JSON before rendering templates driven by specified controllers.

AngularJS' two-way data binding is achieved by the \$scope service that detects changes to the model section and modifies HTML expressions in the view via a controller. Likewise, any alterations to the view are reflected in the model. AngularJS detects changes in models by comparing the current values with values stored earlier in a process of dirty-checking.

## 1.2 Screenshots

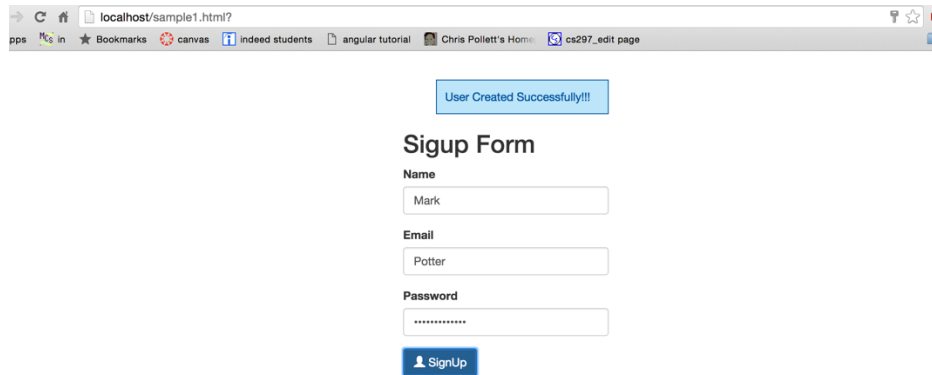
Below are few screenshots from my deliverable one.

Figure 1.2.1: Sign Up form



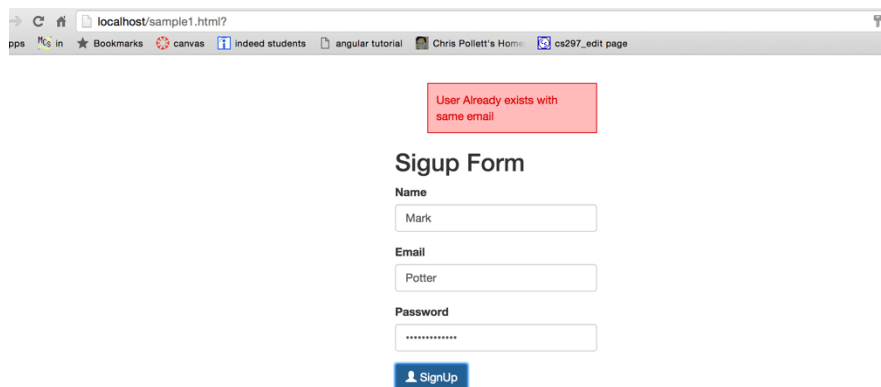
The screenshot shows a web browser window with the address bar displaying 'localhost/sample1.html?'. The browser's bookmark bar includes 'Apps', 'in', 'Bookmarks', 'canvas', 'indeed students', 'angular tutorial', 'Chris Pollett's Home', 'ca297\_edit page', and 'Other'. The main content area features a form titled 'Sigup Form'. The form contains three input fields: 'Name' with the placeholder text 'Enter a Name', 'Email' with the placeholder text 'Enter a Email', and 'Password' with the placeholder text 'Enter a password'. Below these fields is a blue button with a white user icon and the text 'SignUp'.

Figure 1.2.2: After user enters valid credentials



A screenshot of a web browser window. The address bar shows 'localhost/sample1.html?'. The browser's bookmark bar includes 'pps', 'in', 'Bookmarks', 'canvas', 'indeed students', 'angular tutorial', 'Chris Pollett's Home', and 'cs297\_edit page'. A blue message box at the top center says 'User Created Successfully!!!'. Below it is the 'Sigup Form' with three input fields: 'Name' (containing 'Mark'), 'Email' (containing 'Potter'), and 'Password' (containing '\*\*\*\*\*'). A blue 'SignUp' button is at the bottom of the form.

Figure 1.2.3: After user enters invalid credentials



A screenshot of a web browser window, similar to the one in Figure 1.2.2. The address bar and bookmark bar are the same. A red message box at the top center says 'User Already exists with same email'. Below it is the 'Sigup Form' with the same three input fields: 'Name' (containing 'Mark'), 'Email' (containing 'Potter'), and 'Password' (containing '\*\*\*\*\*'). A blue 'SignUp' button is at the bottom of the form.

## **DELIVERABLE 2: A Power Point Presentation On the Literature Review**

This deliverable was to understand the mathematics behind auction systems, study different auctions systems and popular auctioning systems online like eBay. I presented all my understandings as a slide presentation. I have summarized my literature findings on auction theory book below.

### **2.1 Types of Auction System**

An auction is a process of buying and selling goods or services by offering them up for bid, taking bids, and then selling the item to the highest bidder. The most popular types of auction systems are:

**An English Auction:** The sale is conducted by an auctioneer who begins by calling out a low price and raises it, typically in small increments, as long as there are at least two interested bidders. The auction stops when there is only one interested bidder.

**A Dutch Auction:** The Dutch auction is the open descending price counterpart of the English auction. It is not commonly used in practice but is of some conceptual interest. the auctioneer begins by calling out a price high enough so that presumably no bidder is interested in buying the object at that price. This price is gradually lowered until some bidder indicates her interest. The object is then sold to this bidder at the given price.

**A Sealed-bid First Price:** In this auction type the bidders submit bids in sealed envelopes; the person submitting the highest bid wins the object and pays what he bid.

**A Sealed-bid Second Price:** In this type of auction the bidders submit bids in sealed envelopes; the person submitting the highest bid wins the object but pays not what he bid but the second-highest bid.

### **2.2 Valuations**

Auctions are used precisely because the seller is unsure about the values that bidders attach to the object being sold. There are few important valuations that are used in auctions. If the seller knew the values precisely, he could just offer the object to the bidder with the highest value at or just below what this bidder is willing to pay. This uncertainty is an inherent feature of the auctions.

**Private Values:** If each bidder knows the value of the object to himself at the time of bidding, the situation is called one of privately known values or private values. For example: consumption good (paintings).

**Interdependent Values:** If we take assets an expert might give an estimate of the asset whereas other bidders might possess information -say, additional estimates or test results.

Values are unknown at the time of the auction and may be affected by information available to other bidders these values are called interdependent values.

**Common Values:** A common value model is most appropriate when the value of the object being auctioned is derived from a market price that is unknown at the time of the auction.

Finally, we can conclude that the open descending price (or Dutch) auction is strategically equivalent to the first price sealed-bid auction. When values are private, the open ascending price (or English) auction is also equivalent to the second-price sealed-bid auction, albeit in a weaker sense.

## 2.3 The Symmetric Model

This is the valuation structure for the bidders. single object for sale,  $N$  potential buyers are bidding for the object. Bidder  $i$  assigns a value of  $X_i$  to the object i.e. the maximum amount a bidder is willing to pay for the object. Each  $X_i$  is independently and identically distributed on some interval  $[0, \omega]$  according to the increasing distribution function  $F$  and each bidder is both willing and able to pay up to his or her value. The emphasize was on the fact that distribution of values is the same for all bidders, and this situation is referred as one involving symmetric bidders.

Also in this framework two major auction formats were examined:

- A first-price sealed-bid auction, where the highest bidder gets the object and pays the amount he bid
- A second-price sealed-bid auction, where the highest bidder gets the object and pays the second highest bid

## 2.4 Bidding On eBay

As a part of the literature review I studied how bidding on popular bidding sites like eBay work. When you bid on eBay, you aren't really placing a single bid instead you are instructing eBay to automatically bid on your behalf up to the maximum you are prepared to bid. Your maximum bid is kept secret. No buyers or sellers can see your maximum bid while you are winning an auction. Your maximum bid is only revealed if you are outbid and no longer in the lead.

For example: You are first to bid on one of our auctions with a \$14.99 start price. You want to pay no more than \$30, so place a bid for \$30. You now lead the auction with a bid of \$14.99 showing. Your bid will only increase if someone else places a bid. Below is a sample table of how bid increments are considered.

Current Price	Bid Increment
\$0.01 - \$0.99	\$0.05
\$1.00 - \$4.99	\$0.25
\$5.00 - \$24.99	\$0.50
\$25.00 - \$99.99	\$1.00
\$100.00 - \$249.99	\$2.50

I also came across two interesting terms called snipping and reserve prices. A reserve price allows you to set a low starting price to generate interest and bidding, but protects you from having to sell your item at a price that you feel is too low. Snipping is bidding in the last few seconds of an auction so that it ends before another bidder gets a chance to outbid you. The idea is that this will get you a lower price.



## **DELIVERABLE 3: UML Design for The Application**

In this deliverable I tried to find out the possible use cases and scenarios in an auction system. I also drew a class diagram based on these findings. I used Star UML as my design tool. These are list of possible steps that can happen in the auction system.

### **1. User tries to login to the application.**

In the event of successful login:

### **2. The user can be a seller or a buyer.**

#### **2.1 In the event of being a seller:**

**2.1.1** The user can add items for auction.

**2.1.2** He/ She can see the maximum bid in the on going auction.

**2.1.3** After the auction closing time is reached he decides on whether to accept the bid or reject it.

**2.1.4** He/she can view the history of items that they have sold

#### **2.2 In the event of being a buyer:**

**2.2.1** The user can see the list of active auctions.

**2.2.2** She/he can bid for an item

**2.2.3** They can see the list of items they have bid and won

In the event of unsuccessful login:

### **3. The user can sign Up for an account incase if they don't have one.**

The following actors and use cases have been identified.

#### **Actors:**

- Seller - The person who sells the items.
- Bidder - The person who wants to bid for an item and purchase it.
- Administrator: The person who administrates all the accounts and makes sure there is no dispute between the buyer and the seller.

#### **Use Cases:**

- Login – The user logs in with his credentials.
- Validate User – Check whether the login credentials are correct and the user exists.
- Add Items – Sellers can add items for auction.
- Bid on Item – Buyers can bid on an item that is in auction.
- Select a winner – After the auction closing time is reached the system picks a winner.
- History – Can see the history of previous items bought / sold.
- Edit Item – Can edit the item added for auction.
- Delete Item – Can delete an item added for auction.
- Accept/Reject Bid – The seller after the system picks up a winner can either accept/reject the bid on the item.
- Logout – user logouts from the current session.
- Sign Up – User can signup for an account.

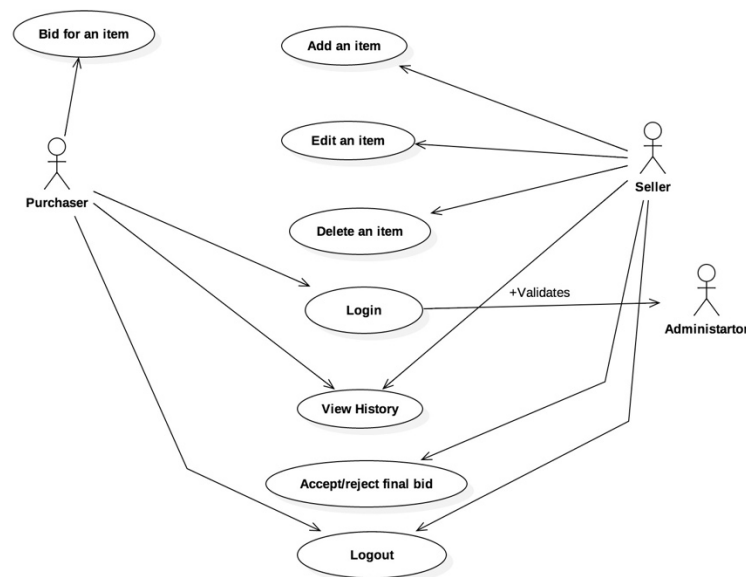


Figure 3.1 Use Case Diagram of an Auction System

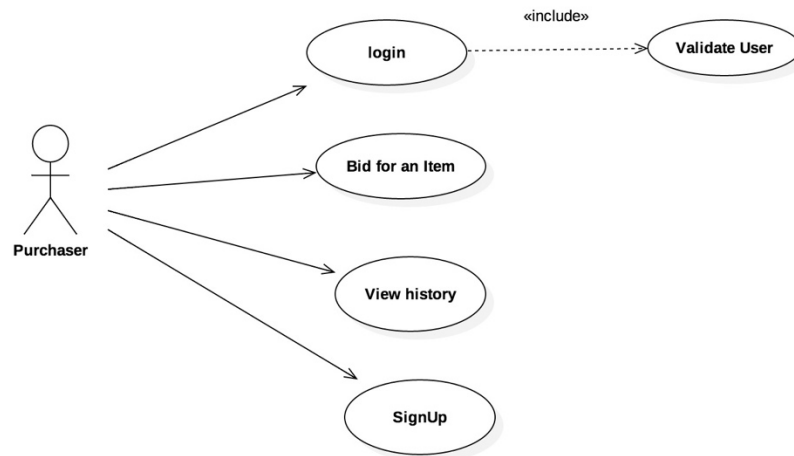


Figure 3.2 Use Case Diagram of a Purchaser

## DELIVERABLE 4: A Web application implementing the use cases

In this deliverable I developed a small web based auction system called “**Bitwise**” using AngularJS as my front-end, MySQL as my database and PHP for sever side. I created a ReSTful web service using PHPSlim framework as data provider. The data from the ReSTful service will be used to enable user authentication in our AngularJS application.

The slim framework provides a fast and powerful router that maps route callbacks to specific HTTP request methods and URIs. It supports parameters and pattern matching.

```
<?php
require 'Slim/Slim.php';

$app = new Slim();

$app->get('/Customers', 'getCustomers');
$app->get('/Winner', 'getWinner');
$app->get('/Customers/:id', 'getCustomer');
$app->post('/login', 'doLogin');
$app->post('/AddBid', 'addBid');
$app->post('/New_Customer', 'addCustomer');
$app->put('/Customers/:id', 'updateCustomer');
$app->put('/Customers/:user', 'itemSold');
$app->delete('/Customers/:id', 'deleteCustomer');

$app->run();
```

Figure 4.1

After the user enters the correct combination of email and password, he will be authenticated from the credentials stored at our MySQL database. On successful login it will store the credentials in the user session and redirect to the welcome page. All subsequent pages browsed after login will have access to the user information. The passwords are hashed before they are stored in the database.

uid	name	email	phone	password
169	Swadesh Behera	swadesh@gmail.com	1234567890	\$2a\$10\$251b3c3d020155f7553c1ugKfEH04BD6nbCbo78AIDV...
170	Ipsita Sahoo	ipsita@gmail.com	1111111111	\$2a\$10\$d84ffcf46967db4e1718buENHT7GVpcC7FfbSqCLUJD...
171	Trisha Tamanna Priyadarsini	trisha@gmail.com	2222222222	\$2a\$10\$c9b32f5baa3315554bffcuWfjiXNhO1Rn4hVxMXyJHJ...
172	Sai Rimsha	rimsha@gmail.com	3333333333	\$2a\$10\$477f7567571278c17ebdees5xCunwKISQaG8zkKhvfE...
173	Satwik Mohanty	satwik@gmail.com	4444444444	\$2a\$10\$2b957be577db7727fed13O2QmHMd9LoEUjioYe.zkXP...
174	Tapaswini Sahoo	linky@gmail.com	5555555555	\$2a\$10\$b2f3694f56fdb5b5c9ebeulMJTSx2lv6ayQR0GUACDs...
175	Manas Ranjan Subudhi	manas@gmail.com	6666666666	\$2a\$10\$03ab40438bbddb67d4f13Odrzs6Rwr92xKEYDbOO7IX...

Figure 4.2

The point of entry to our program will be index.html where we define our JavaScript, CSS, PHP dependencies. This will be single point where we define everything. This is where we define our **ng-view** directive where all our partial html views will get updated. The index.html is turned into a template what is called a "layout template". This is a template that is common for all views in our application. Other "partial templates" are then included into this layout template depending on the current "route" — the view that is currently displayed to the user.

Application routes in Angular are declared via the \$routeProvider, which is the provider of the \$route service. This service makes it easy to wire together controllers, view templates, and the current URL location in the browser. Using this feature, we can implement deep linking, which lets us utilize the browser's history (back and forward navigation) and bookmarks.

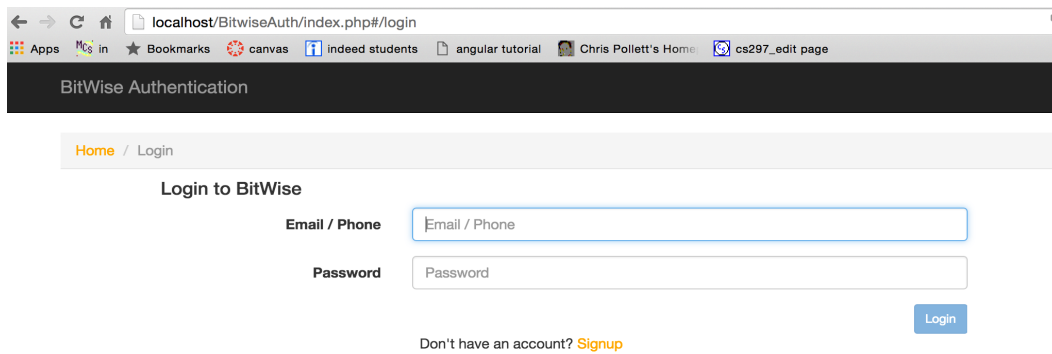
```
var app = angular.module('Application', ['ngRoute']);

app.config(['$routeProvider', function($routeProvider) {
  $routeProvider.
    when('/', {templateUrl: 'pages/login.html', controller: 'loginControler'}).
    when('/dashboard', {templateUrl: 'pages/lists.html', controller: 'AuctionListControler'}).
    when('/NewCustomer', {templateUrl: 'pages/new.html', controller: 'AuctionAddControler'}).
    when('/UpdateCustomer/:id', {templateUrl: 'pages/edit.html', controller: 'AuctionEditControler'}).
    otherwise({redirectTo: '/'});
}]);
```

Figure 4.3

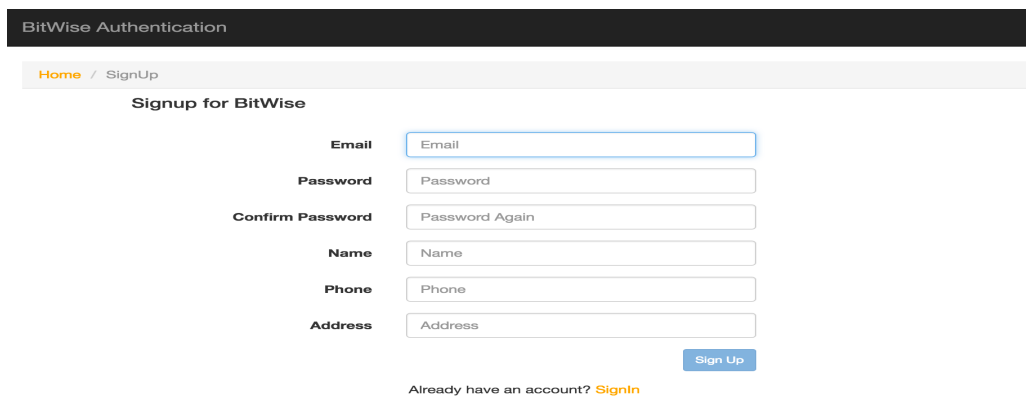
Upon unsuccessful login or logout, the app will redirect the user to the login page. I used AngularJS-Toaster plugin to communicate to the user easily. It is a non-blocking notification JavaScript library. The user upon successful login can see his/her dashboard page with a list of active items for auction. The user can add items for auction which other buyers can place a bid and buy. Clicking on add item opens up a form with a list of user input information's such as selling product name, description, minimum bid and closing time of the auction.

Once this information is entered the item automatically gets added to the list of active auctions. Buyers can now bid on this item. The buyer's bid value must be greater than the minimum value of the bid. After the closing time is reached. I user angular timer for my stop down timer.



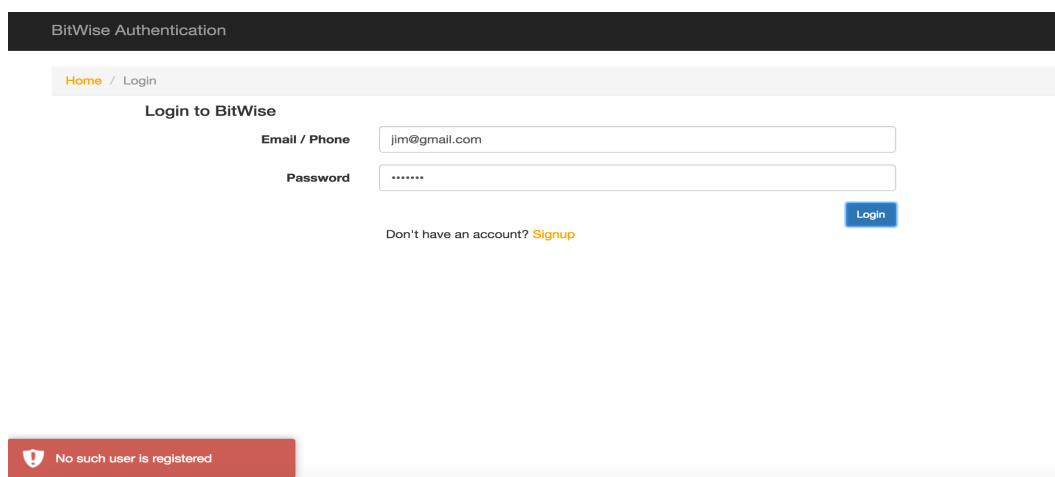
The screenshot shows a web browser window with the address bar displaying 'localhost/BitwiseAuth/index.php#/login'. The browser's bookmark bar includes 'Apps', 'in', 'Bookmarks', 'canvas', 'indeed students', 'angular tutorial', 'Chris Pollett's Home', and 'cs297\_edit page'. The page has a dark header with 'BitWise Authentication'. Below the header is a breadcrumb trail 'Home / Login'. The main heading is 'Login to BitWise'. There are two input fields: 'Email / Phone' and 'Password'. A blue 'Login' button is positioned to the right of the password field. Below the inputs, a link says 'Don't have an account? Signup'.

Figure 4.4 Login Page



The screenshot shows the 'Sign Up for BitWise' page. It features a dark header with 'BitWise Authentication' and a breadcrumb trail 'Home / SignUp'. The heading is 'Sign up for BitWise'. There are six input fields: 'Email', 'Password', 'Confirm Password' (labeled 'Password Again'), 'Name', 'Phone', and 'Address'. A blue 'Sign Up' button is to the right of the 'Address' field. Below the inputs, a link says 'Already have an account? SignIn'.

Figure 4.5 Sign Up Page



This screenshot shows the login page after an attempt with invalid credentials. The 'Email / Phone' field contains 'jim@gmail.com' and the 'Password' field contains seven dots. The blue 'Login' button is visible. Below the inputs, a link says 'Don't have an account? Signup'. At the bottom of the page, a red error message box with a shield icon states 'No such user is registered'.

Figure 4.6 User credentials invalid.

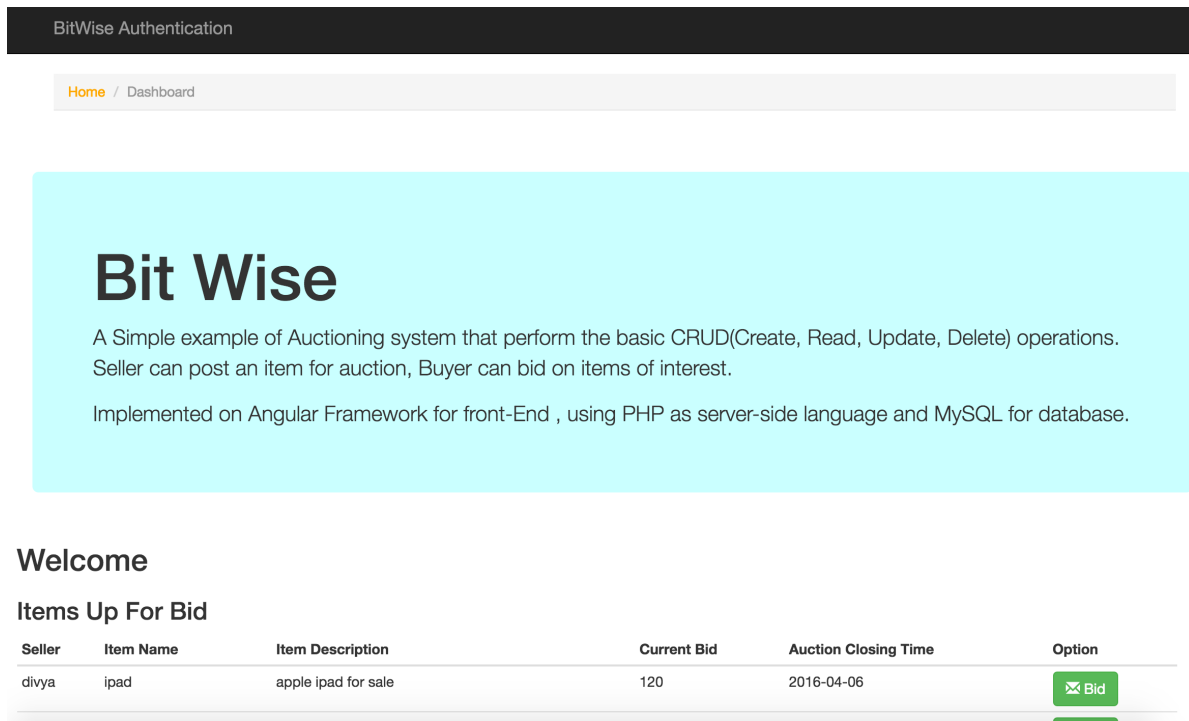


Figure 4.7 User Dashboard page - 1

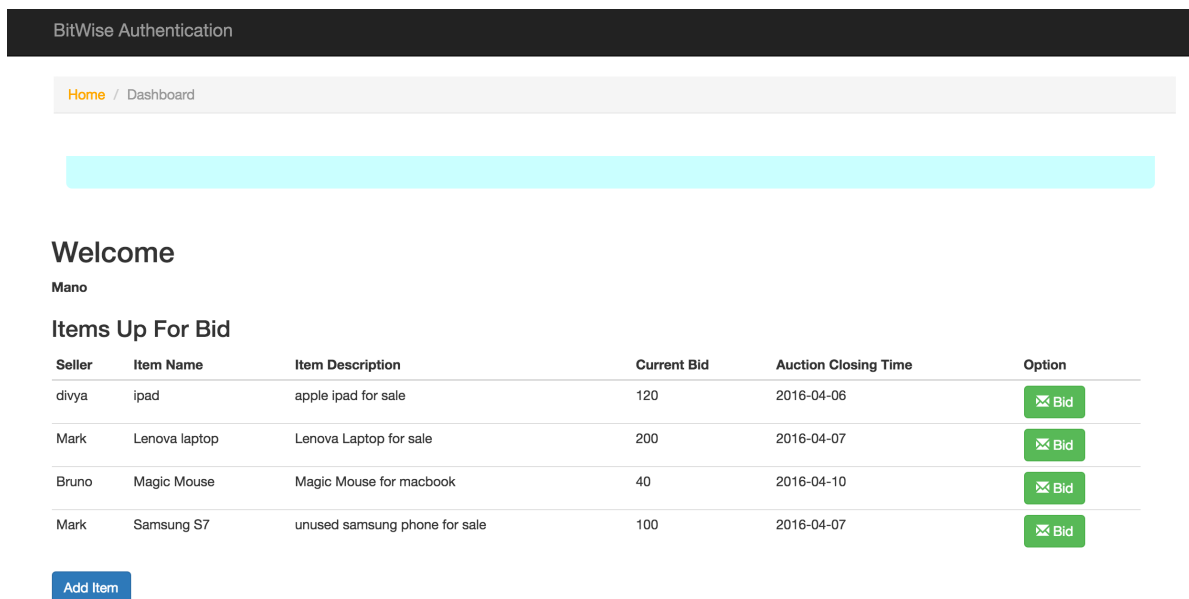


Figure 4.10 Dashboard page -2

BitWise Authentication

Home / Dashboard

## New Auction Item

**Seller**

**Item Name:**

**Item Description:**

**Min Bid**

**Auction Closing Time:**

[Save](#) [Cancel](#)

Figure 4.9 New Auction Item

BitWise Authentication

Home / Dashboard

## New Auction Item

**Seller**

**Item Name:**

**Item Description:**

**Min Bid**

**Auction Closing Time:**

[Save](#) [Cancel](#)

Figure 4.10 New auction item with values



BitWise Authentication

Home / Dashboard

Lenova laptop

**Item Description**

Lenova Laptop for sale

**Auction Closing Time**


2016-04-07

**Buyer Name**

Mano

**Min Bid**

160

Bid 

Cancel


0 hours ,2 minutes , 30 seconds

See Winner

Figure 4.11 Bid items

BitWise Authentication

Home / Dashboard



Cancel

0 hours ,0 minutes , 0 seconds

See Winner

Mano

Figure 4.12 Picked winner

The screenshots above give an overview on what happens in each step of the application. Once the user logouts the session information is lost. When user logs in again a new session is created.

## CONCLUSION

I have now implemented a simple auction system based on AngularJS, PHP and MySQL. I know how an auction system function and also the working logic behind AngularJS. I would like to use this knowledge to design a web application that uses some kind of points/ resources instead of money for auctions and develop a better platform for office markets.

This CS 297 project have me a good foundation for my future dissertation. As the next step, I would like to continue this work by designing a better web application performing auctioning in office markets. The application must let users pick what kind of resources they are interested to trade in. There must be restrictions on what type of users can trade what kind of money. I want to design a database figuring out the necessary tables, fields and the relationship between them. For my CS 298 I would like to develop the application using MEAN Stack (Mongo DB as my database, Express.js, Node.js for my server and Angular for my front-end).