**Case Study using below UI technologies**

* HTML5
* CSS3
* JavaScript
* jQuery
* jQuery UI
* JavaScript OOP

**Application Name: Web Admin System**

**Tasks: UI**

**Create a Standard Layout which can be shared across the pages.**

**Layout includes header, footer, sidebar and content section.**

**Include navigation section in the header, so that the user can navigate across the pages using hyperlinks.**

**You can use HTML 5 semantic elements and CSS3 styles for better UX.**

**Include: Home, Contacts, about links in the navigation section.**

**When the start page (index.html) page is loaded, you must display**

**[Welcome, Guest!][Login/Register] on the top right of the page.**

**So that, you use can Login with the existing account. Or the user can create a new Account.**

**When the user clicks on the Login link, login.html page must be loaded.**

**When the user clicks on the Register, Register.html page must be loaded.**

**If the user login to the system using existing account, you must display**

**[Welcome Username!] [Logout] on the top right of the page.**

**And you must enable some links like View Details, Edit/Update Details, and Reset Password links in the sidebar section.**

**For View Details, you must display the UserInfo for readonly.**

**For Edit/Update, you must able to edit and update the user info.**

**For Reset Password, you must able to reset the current password with new password.**

**Pages:**

**Index.html (single page /main page)**

**The below are partials :**

**About.html**

**Contacts.html**

**Login.html**

**Register.html**

**UserInfo.html**

**EditUserInfo.html**

**ResetPassword.html**

**Signout.html**

**Create User UI:**

**You need the below fields:**

**Username, Password, Email, Mobile**

**Include the validations**

**Business Logic:**

**Use JavaScript OOP for the complete application**

**Steps: app.js**

var webAdminApp = {};

webAdminApp.appName = 'Web administration Application';

webAdminApp.version = 1.0;

webAdminApp.UserModule = {

data: [{ id: 1, username: 'admin', password: 'admin', email: 'admin@marlabs.com', mobile: '12345' }],

addUser: function (user) {

//add user here...

},

getUser: function (username) {

//get user by name

},

updateUser: function (user) {

//update the existing user with current user

},

deleteUser: function (username) {

//delete the existing user with current user

},

login: function (username, password) {

//check whether user exists or not with given username and password

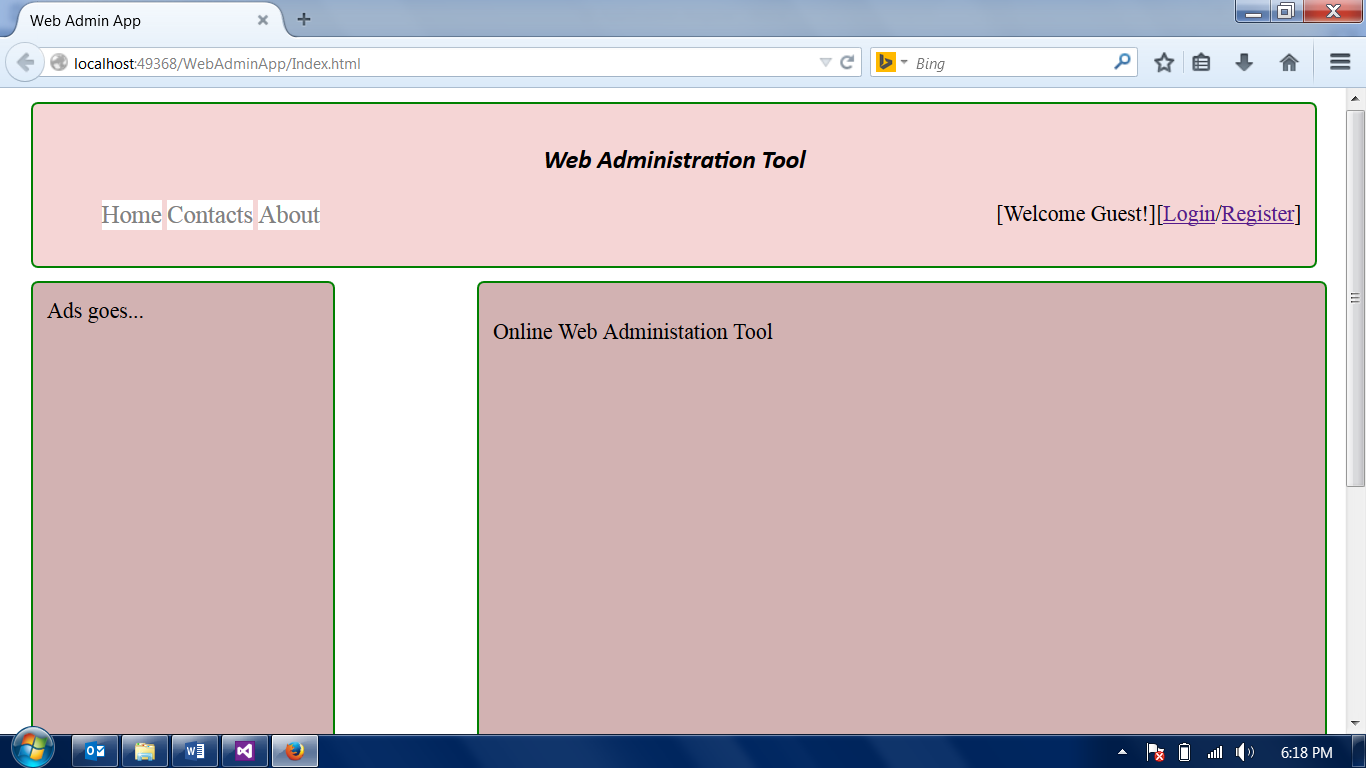
}

};

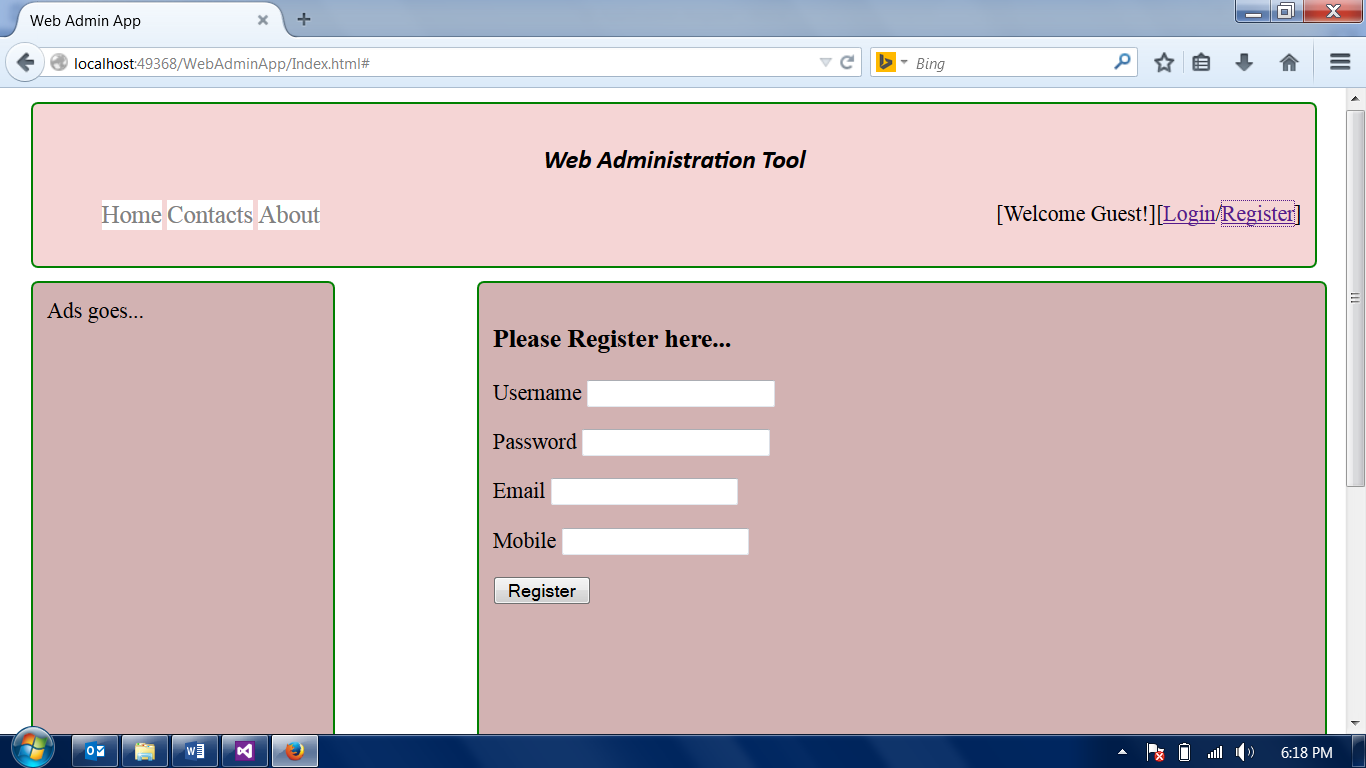
Sample Screens:

Sample Screens:

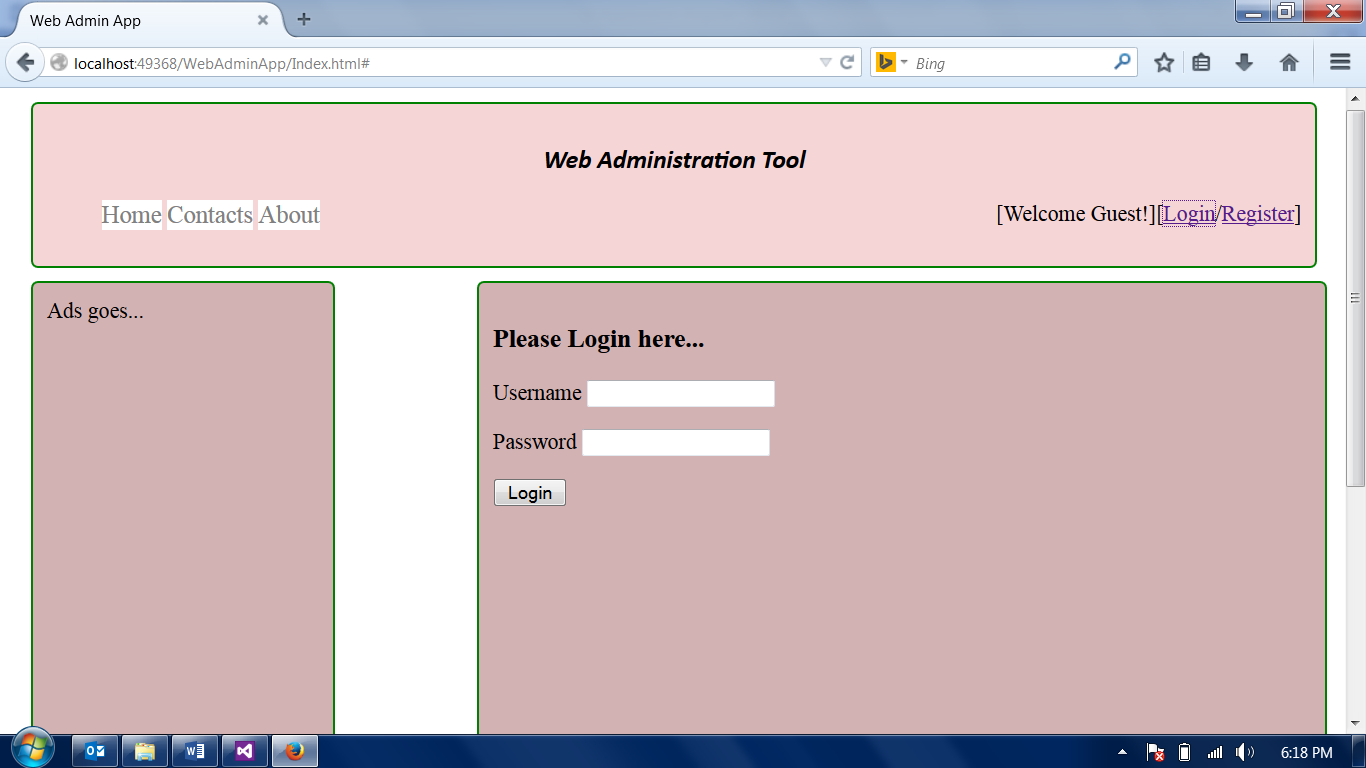
index.html



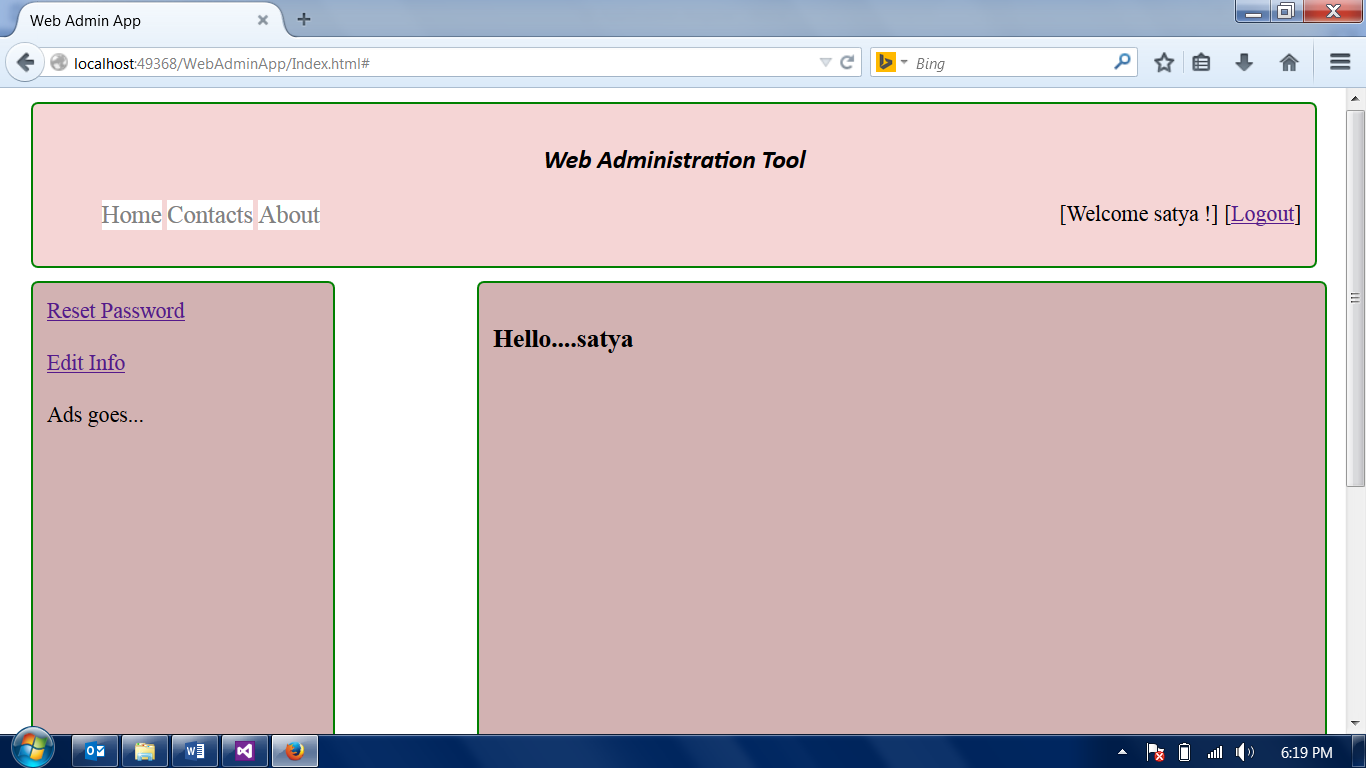
Register.html



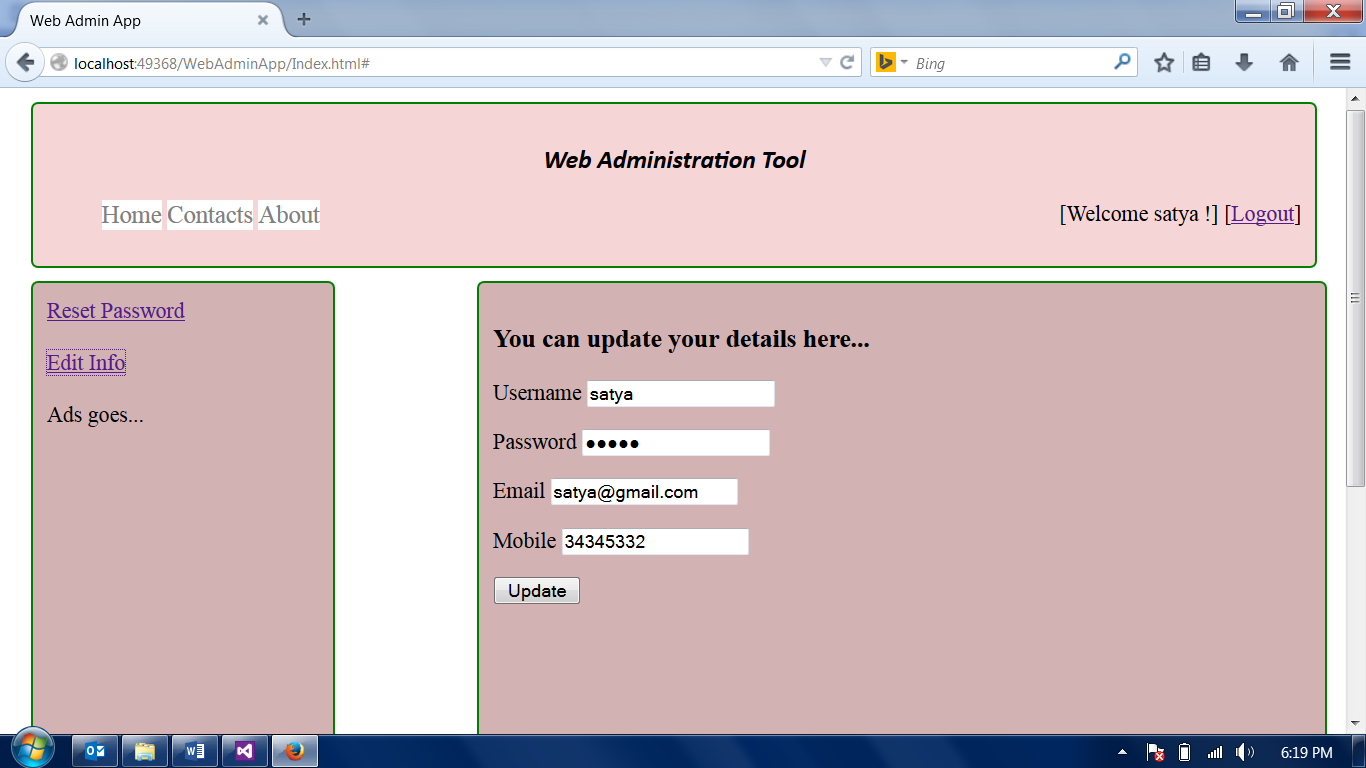
Login:



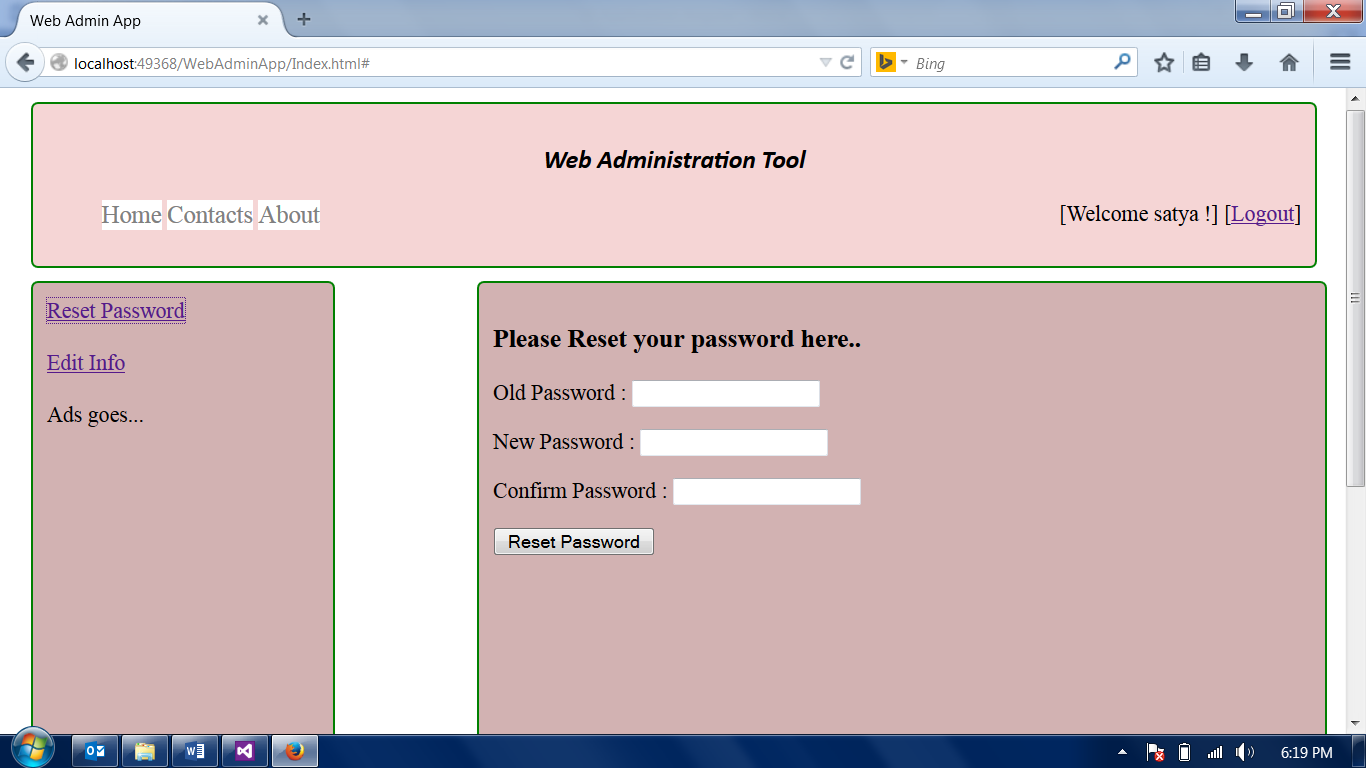
After login:



Edit Info:



Reset Password:



Enhancement to the above Application:

Phone Book

Once the user is login to the system, the user can create his/her phone book.

Properties in the Phonebook for each contact:

* Id (ContactId)
* FirstName
* LastName
* Email
* State
* City
* Gender
* Mobile
* UserId (Currently Logged In User)

You need to perform :

Add contact

View Contact

Delete Contact

Update Contact

Take one partial for this task: (Phonebook.html)

In this partial you must display all the existing contacts for the currently logged in user.

You must able to add new contact to the current logged in user phonebook.

You take one grid (table) and display all the contacts and add actions (edit, view, delete)

Behavior :

You can use the regular Javascript for the behavior.

You can use JavaScript array for phonebook storage.

e.g.

app.js

webAdminApp.Phonebook={

data:[{contactId:1,firstname:’satya’,lastname:’sat’,email:’dotentsatya@gmail.com’,’nj’,’edison’,mobile:’3434534’,userid:1}],

addContact:function(contact){

},

getContactById:function(contactId){  
},

getAllContacts:function(){

}  
};