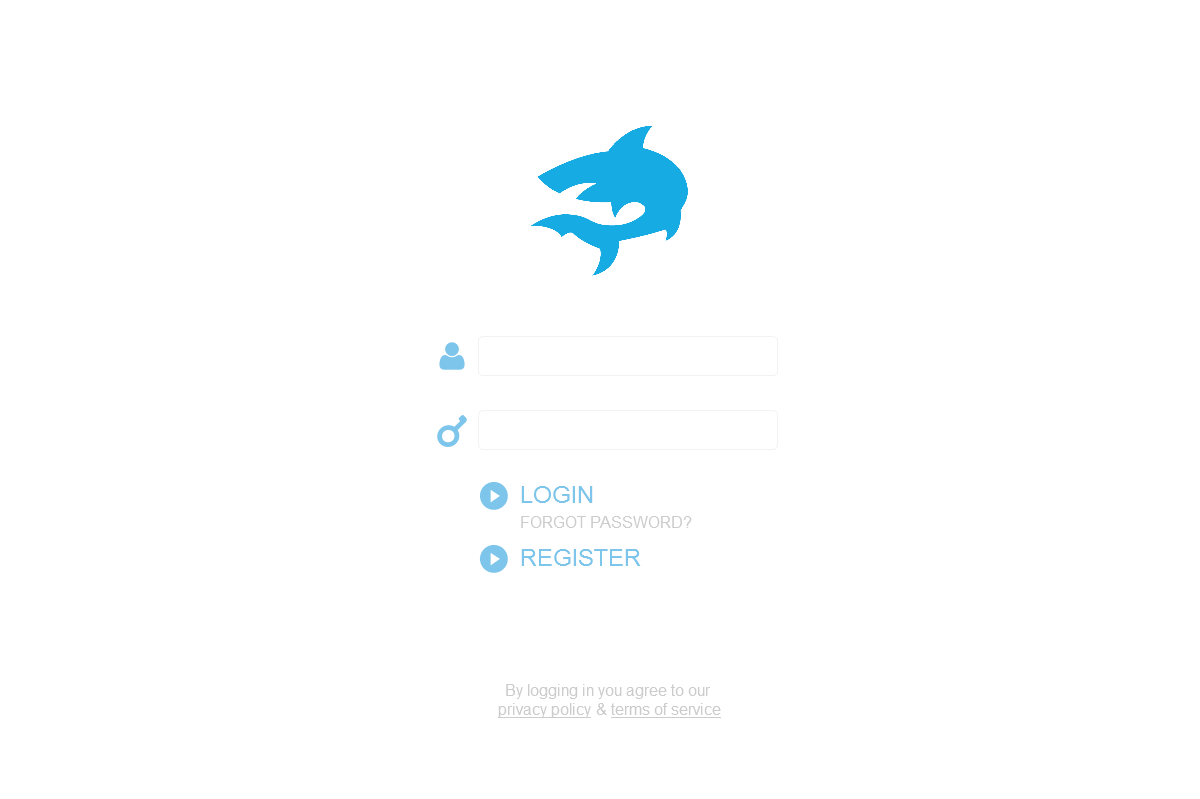
**- User Interface -**

1. **Login/Registration**

****

**Description:**

Our Login/Registration page will be the index page to our website. In order to access our site you MUST BE REGISTERED, there is nothing about this site that should be public to people who are not employed.

**Stub-Calls:**

//Register Function(s)

// 1.) Accepts Registration associative array:

// a.) Package up all the information we need from our forms

// into organized data to be inserted into our database to register a new user

// 2.) Registration **Employee table** requires:

// a.) Valid Employee First Name (Separate Function Validates)

// b.) Valid Employee Last Name (Same Function Validates as First Name)

// c.) Valid Employee birthday (Separate Function Validates)

// 3.) Registration **Contact\_Information table** requires:

// a.) Valid Phone Number (Separate Function Validates)

// b.) Valid Email Address (Separate Function Validates)

// c.) Valid Zip-Code (Separate Function Validates)

// 4.) Registration **Position table** requires:

// a.) Valid Registration Code Generated by Admin

// i.) That registration code will give access to position\_id

// 5.) Registration **Availability\_table** requires:

// a.) Valid Days Available (Will be Options to Select, No Validation Required)

// b.) Valid START TIMES for each DAY AVAILABLE

//1.) Accepts initial Associate array

//2.) Calls Validation checks listed above on each index of associative array that needs validation

//3.) Returns TRUE on a successful registration and adding a new Employee to our Database

**Bool Register (char RegisterInfo[]);**

//1.) Accepts either first or last name

//2.) Checks each character to ensure it is a valid character

//3.) Checks length of string to ensure it’s less than or equal to 25

//4.) Returns TRUE on Successful validation, FALSE on error

**Bool NameValidate(String name)**

//1.) Accepts a Date (Users Birthday)

//2.) Checks to make sure that the user’s age matches requirements for position

// a.) Servers must be ATLEAST 18

// b.) Bartenders must be ATLEAST 21

// c.) All others must be ATLEAST 16

//3.) return TRUE on Success, FALSE on Failure

**Bool BirthdayValidate(Date bday)**

//1.) Accepts Phone Number

//2.) Ensures the number of digits is EXACTLY 10

//3.) Ensures all the values are digits

//4.) return TRUE on Success, FALSE on Failure

**Bool PhonNumberValidate(double phoneNumber)**

//1.) Accepts Email Address

//2.) Ensures that the email has proper format

// a.) Either lookup API for this

// b.) Or check for “@” and “.com” to exists in the string

//3.)Check to make sure email length is less than or equal to Max email length

//4.) return TRUE on success, FALSE on failure

**Bool EmailValidiate(String emailAdress)**

//1.)Accepts a Zipcode

//2.)Checks to make sure zip code has 5 digits exactly

//3.) returns TRUE on Success, FALSE on failure

**Bool ZipValidate(int zipcode)**

//1.) A code is generated by our Manager that allows an employee to register AND links the employee to a position ID

//2.) We check that the code to make sure it is accurate (Check against codes in Database)

//3.) returns position ID if the code is valid, returns -1 if invalid code

**int ValidationCode()**

//1.) Accepts email and password (both strings)

//2.) Runs Query against database that receives the hashed password associated with the email

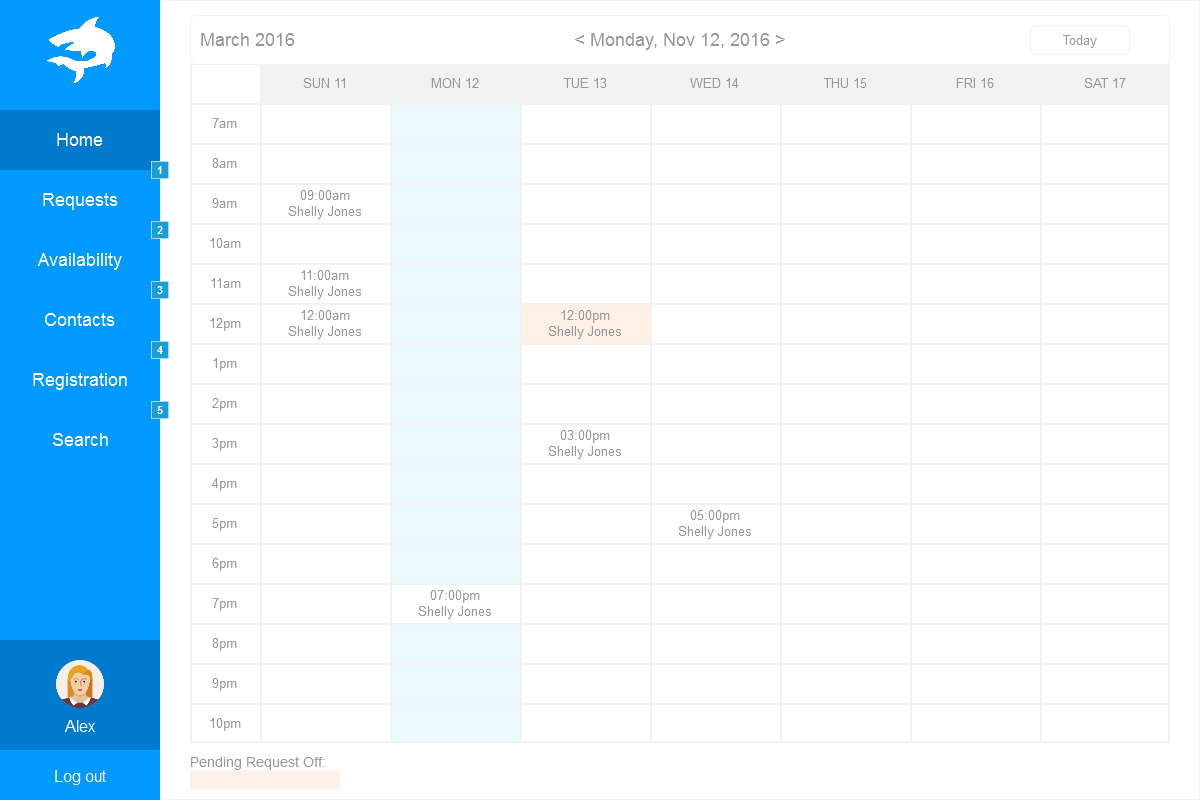
//3.) We will hash the password that was inputted and compare it to the one from the Database

//4.) On success, fill out all necessary Session Variables and allow login, return true

//5.) On failure, return false

**Bool Login(String email, String Password)**

1. **Employee Home**

****

**Description:**

Upon login, our users will be able to view their schedule as their home page. This allows for a quick login, check schedule, and logout.

**Stub-Calls:**

//1.) Accepts nothing, returns nothing

//2.) Runs a strategic query on our Schedule table to print the information out in an organized

// fashion viewable for the employee

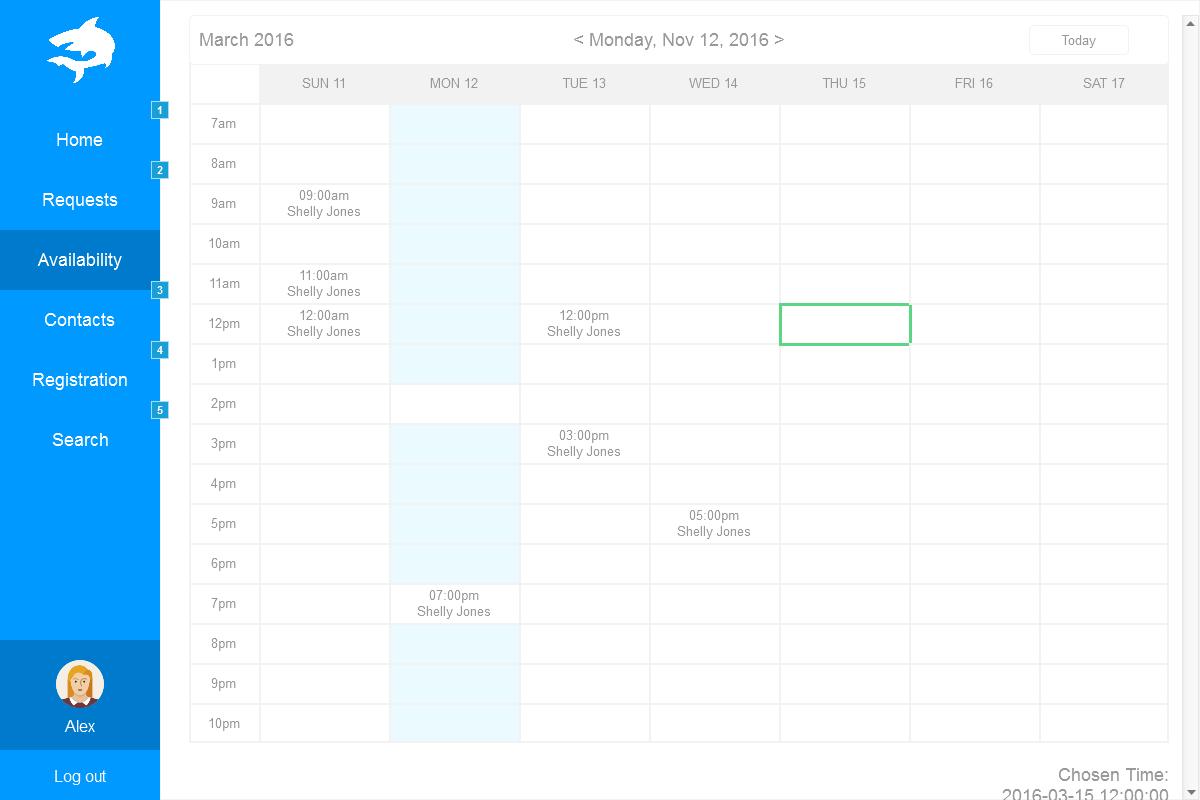
**Void LoadSchedule()**

//1.) Logout Accepts nothing, and returns nothing

//2.) This function will remove all SESSION variables we currently have present

**Void Logout()**

1. **Employee Availability**

****

**Description:**

This page will allow our users to update their availability (Upon approval by a manager). It will be a simple form to fill out that will be extremely similar to the form that we use to get availability during registration.

**Stub-Calls:**

//1.) This will accept the information from the update availability form

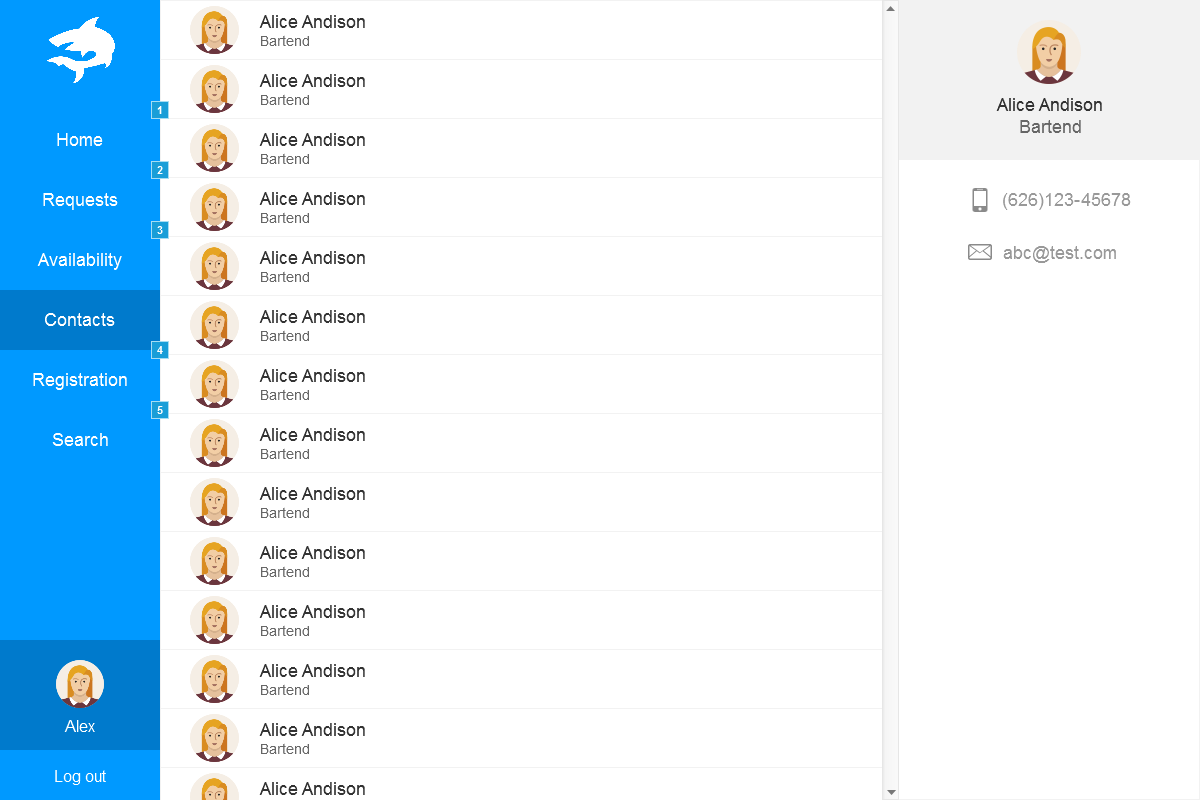
//2.) Insert this information into our Availability table (UNNAPROVED)

//3.) The manager will have to approve availability before changes take effect

//4.) Return TRUE on success, FALSE on failure

**Bool UpdateAvailibility()**

1. **Employee Contact**

****

**Description:**

This page will allow our users to check on contact information for other employees. The information given will be limited to employees of a similar position ID. There is no reason to contact other employees for non-professional purposes, therefor we will not be providing unnecessary information.

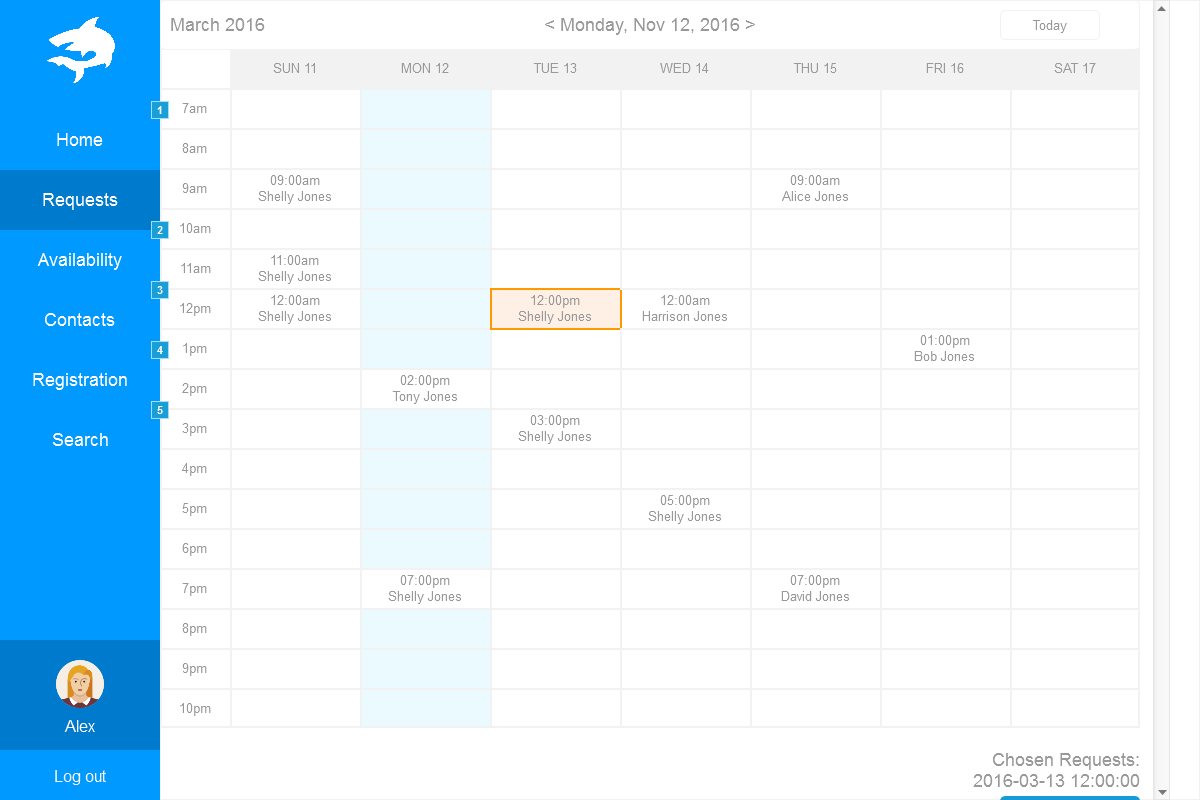
**Stub-Calls:**

//1.) Accepts employees positionID (will be stored as a session variable for easy access)

//2.) Displays the common PositionID contact information in a structured manor

**Void DisplayContactInfo(int positionID)**

1. **Employee Request Off**

****

**Description:**

This page will allow our users to submit request off requests. A simple form will be present to allow the user give us information on the start date, and the end date of their request off. A text box will also be available for the employee to give a description as to why they are requesting off.

**Stub-Calls:**

//1.) Accepts an associative array that stores the information with regards to request off

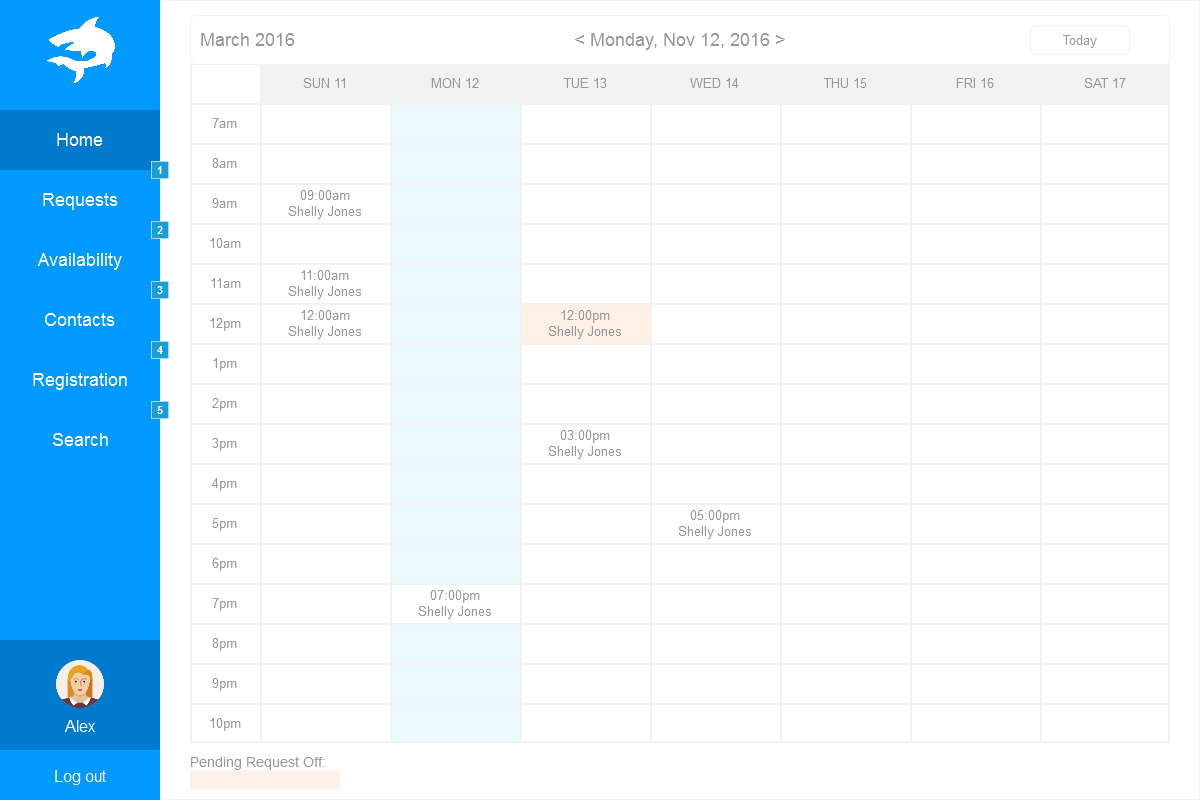
//2.) Does minor error checking on dates to ensure that date is in the future

//3.) On success, request off is stored in the Database to be approved/denied by the

// manager

**Bool submitRequestOff(String requestOffInfo[])**

1. **Manager Home**

****

**Description:**

Upon login, our managers will be able to view the schedules for each type of employee in an organized manor.

**Stub-Calls:**

//1.) Accepts positionID, returns nothing

//2.) Runs a strategic query on our Schedule table to print the information out in an organized

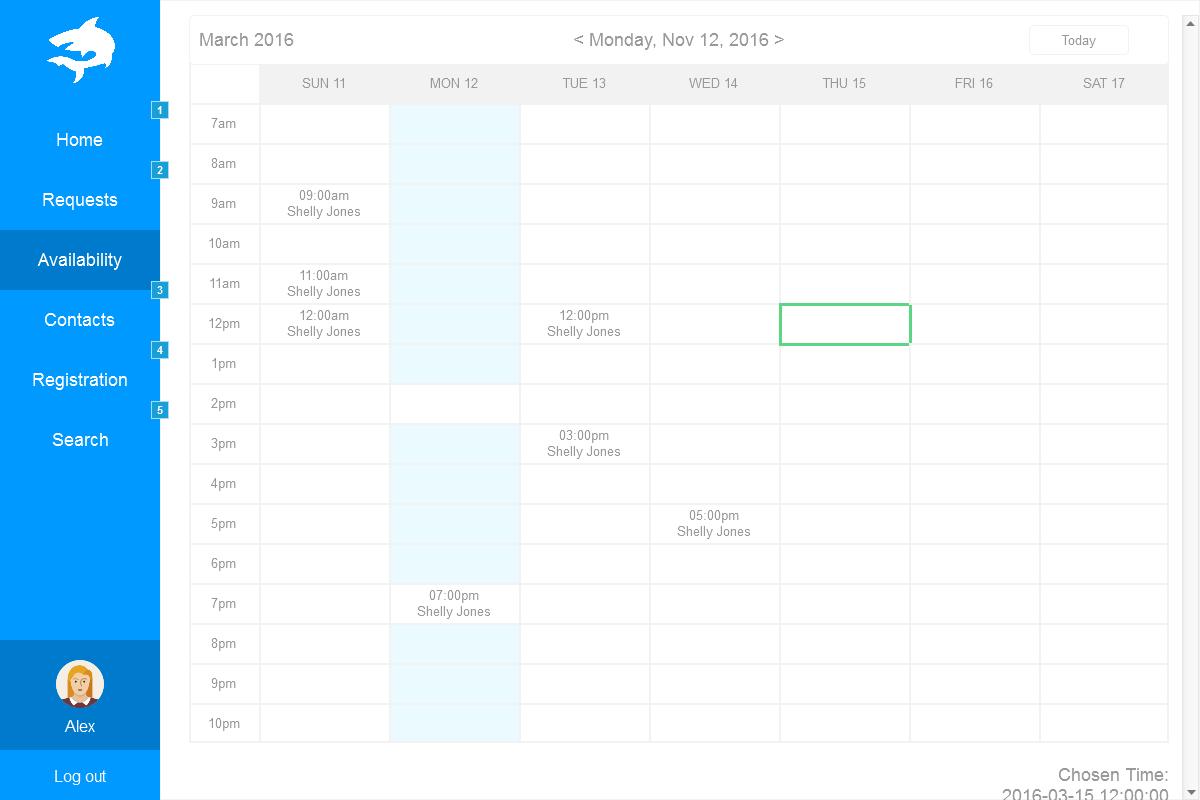
// fashion viewable for the manager (ALL SCHEDULES ARE PRINTED)

**Void LoadSchedule(int positionID)**

//1.) Logout Accepts nothing, and returns nothing

//2.) This function will remove all SESSION variables we currently have present

1. **Manager Availability**

****

**Description:**

This page will allow our managers to approve/deny any pending changes to employees availability.

**Stub-Calls:**

//1.) This function accepts the action (Approve or Deny) the employee ID to alter their

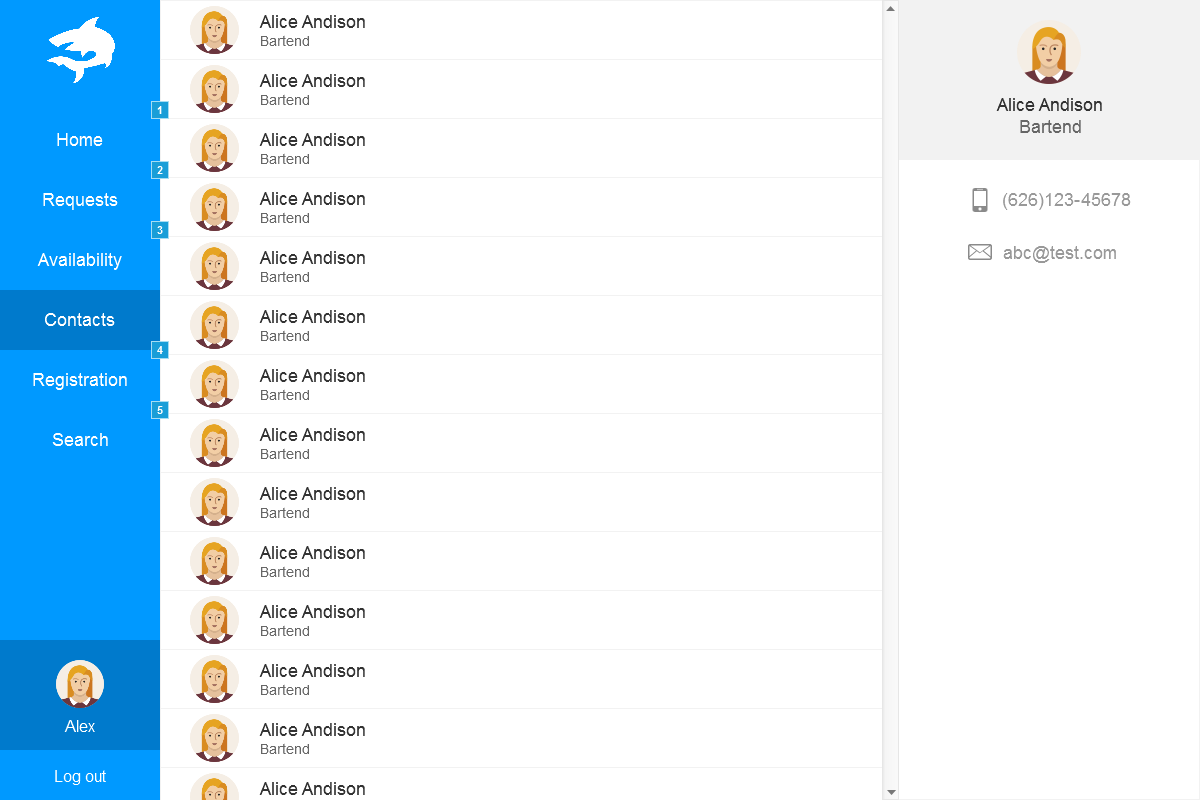
// availability, and an array storing the new availability

//2.) If action is Deny, then we remove this request from the table, and return

//3.) On Accept, Update table and return

**Bool ManagerAvailability(String action, int employeeID, String availability[])**

1. **Manager Contact**

****

**Description:**

This page will allow our users to check on contact information for everyone. Managers will have the privilege to view ALL Employees contact information no matter what position they have.

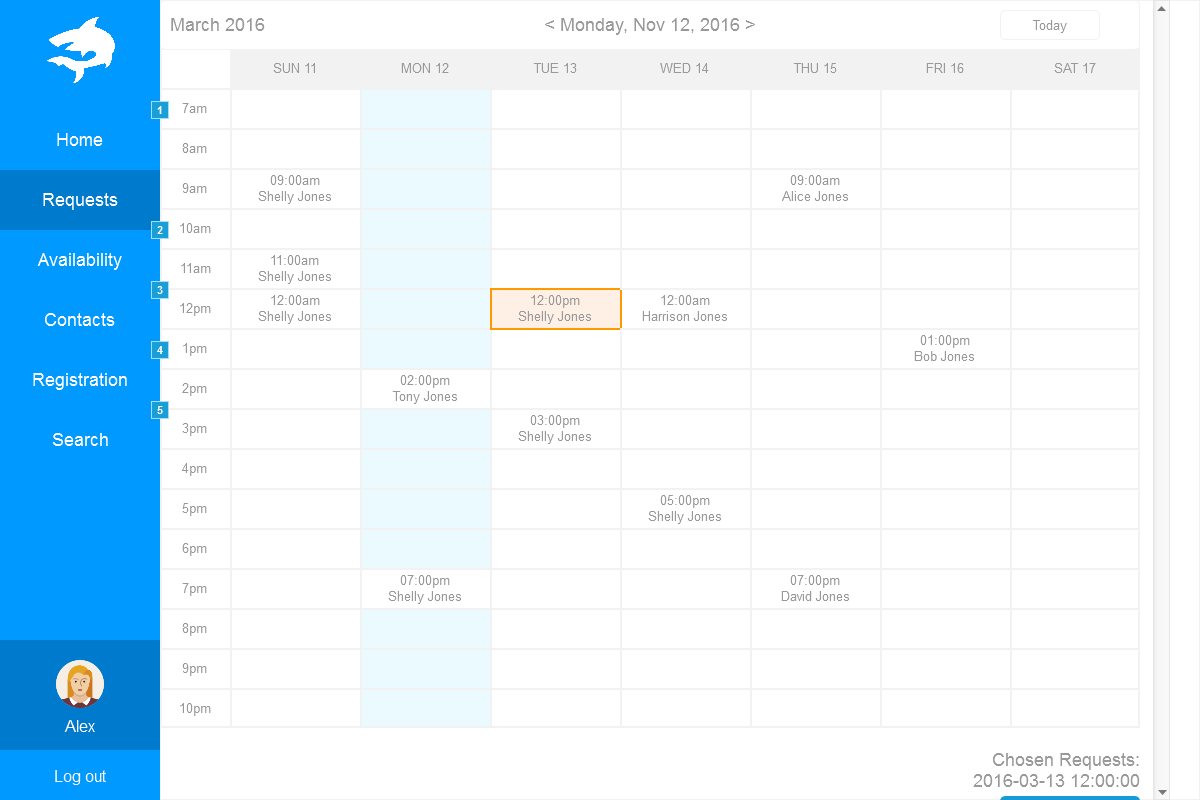
**Stub-Calls:**

//1.) Accepts employees positionID (will be stored as a session variable for easy access)

//2.) Displays the common PositionID contact information in a structured manor

**Void DisplayContactInfo(int positionID)**

1. **Manager Request off**

****

**Description:**

This page will allow our manager to approve or deny any request off

**Stub-Calls:**

//1.) Accepts associative array with request off info and approve or deny

//2.) Action is deny, we delete this request from our table

//3.) Action is accept, we update information in Database to not allow that employee to // be scheduled during the time given.

**Bool ActionRequestOff(String info[], String action)**