1. A Practicum Project Team Report for the VIP Parking Web Application
2. Submitted on December 11, 2016 to the
3. Department of Data Science of the
4. College of Computer & Information Sciences
5. Regis University
6. in partial fulfillment of the requirements for a
7. Master of Science in
8. Software Engineering and Database Technology
9. by
10. Acharya, Hemanta, Jason Kirby, Mathew Boyd, and Johnny Edward Tomblin Jr.

Database Practicum Project Advisor

Bob Mason, Ph.D.

Table of Contents

[List of Figures iii](#_Toc469241577)

[Chapter 1 –Introduction 1](#_Toc469241578)

[Overview 1](#_Toc469241579)

[Project Purpose 1](#_Toc469241580)

[Deliverables 2](#_Toc469241581)

[Stake holders 3](#_Toc469241582)

[Repository Link 3](#_Toc469241583)

[Chapter 2 – Methodology 4](#_Toc469241584)

[Functional Requirements 5](#_Toc469241585)

[Application Specific Requirements 5](#_Toc469241586)

[Software Specific Requirements 6](#_Toc469241587)

[Fully Dressed Use Cases 6](#_Toc469241588)

[Attempts to Authenticate 6](#_Toc469241589)

[Submits a Request for a Parking Spot 8](#_Toc469241590)

[Parking Requested Reviewed 9](#_Toc469241591)

[Generating Report 11](#_Toc469241592)

[Admin Administers Events 12](#_Toc469241593)

[User views map 13](#_Toc469241594)

[admin Bulk loads data 13](#_Toc469241595)

[Use Case Diagram 14](#_Toc469241596)

[Flow Diagram 16](#_Toc469241597)

[Entity Relation-ship Diagram: 17](#_Toc469241598)

[Mock-ups 18](#_Toc469241599)

[Chapter 3 –Results 23](#_Toc469241600)

[Chapter 4 – Conclusions 35](#_Toc469241601)

[References 37](#_Toc469241602)

# List of Figures

[Figure 1: VIP Parking Use Case Diagram Depicting all Actors and Application Interactions. 15](#_Toc469244945)

[Figure 2: Flow Diagram for Regis VIP Parking Application. 16](#_Toc469244946)

[Figure 3: Entity Relationship Diagram for the VIP Parking Application. 17](#_Toc469244947)

[Figure 4: Login Page Wireframe. 18](#_Toc469244948)

[Figure 5: Home Page Wireframe. 18](#_Toc469244949)

[Figure 6: Depicts the New Reservation Page and associated Feedback Pages. 19](#_Toc469244950)

[Figure 7: View Reservations Page Wireframe. 20](#_Toc469244951)

[Figure 8: Administration Home Page Wireframe. 20](#_Toc469244952)

[Figure 9: Administration Reservation Page Wireframe. 21](#_Toc469244953)

[Figure 10: Event Administration Pages Wireframes. 22](#_Toc469244954)

[Figure 11: Login screen allows faculty and staff to authenticate against Active Directory. Administrators can also log in through this form. This screen will notify the user if the login credentials are invalid or if the account is locked. 23](#_Toc469244955)

[Figure 12: Reservations screen. Once a user logs in, this page is displayed, which lists the user’s current and future reservations. Any reservations that have expired will not appear on this list. The user has the opportunity to create new reservations or view the reservations. 24](#_Toc469244956)

[Figure 13: When a user needs to create a reservation, this form is available to complete. Once submitted, all fields are validated to make sure proper information is sent to the system. If the form validates, the information is stored to the database and an email. 24](#_Toc469244957)

[Figure 14: This is the confirmation screen the user receives once a reservation is requested. It serves to notify the user of the request that it was received, and a message indicating that the reservation needs reviewed. 25](#_Toc469244958)

[Figure 15: This is the email the administrators receive when a request for a reservation has been submitted. 25](#_Toc469244959)

[Figure 16: The new reservation request shows up on the user’s dashboard with a status of processing. 25](#_Toc469244960)

[Figure 17: The user can click on the reservation link in order to view the reservation details. The status is included as well as all of the information the user filled out. Once the reservation is submitted, only the administrator can edit it. The user can only view it. 26](#_Toc469244961)

[Figure 18: This is the administrator’s dashboard. It looks similar to the user’s dashboard but contains more pieces of information. There is a search bar to allow him or her to search for reservations by name, as the list may become very long. The header menu also contains links specific to the administrator that the normal user will not be able to access. There is a button to access the waiting list. Just like normal users, the administrator can create, view, and cancel reservations. 27](#_Toc469244962)

[Figure 19: If the administrator clicks on a reservation, an edit form shows up where he or she may make changes to it. The Gate Code field is automatically populated with the data in the database based on the date of the reservation but can be changed for special circumstances. The Lots Allowed can also be checked to notify the requester which lots where vehicles can be parked. If the reservation is approved, the requester and administrators will receive a confirmation email with parking permits attached to it, as well as the gate code and allowed lots. 28](#_Toc469244963)

[Figure 20: The administrator will receive a confirmation message after the reservation has been approved or declined. 28](#_Toc469244964)

[Figure 21: The requester and administrators will receive this email when a reservation is approved. It contains information about the reservation, which in Microsoft Outlook, will allow the user the option to click the information to store it on their calendar. It also contains the gate code for that time slot, as well as the parking lots that are available. Two permits are attached to this email because two parking slots were requested. 29](#_Toc469244965)

[Figure 22: This is the parking permit that is produced upon reservation approval. This can be scan be a parking attendant using any standard smart-phone QR code reader. The QR code contains the direct link to the reservation that will redirect the mobile browser to the reservation record. Only administrators will be able to access this record. 29](#_Toc469244966)

[Figure 23: If a reservation is already approved, the administrator has the options to resend the permits, decline the reservation, or cancel it. Cancelling the reservation removes the reservation from the system. In all cases, a pop-up window will alert the user of the action he or she is about to make, giving the option to avoid the transaction. 30](#_Toc469244967)

[Figure 24: The user administration screen shows a list of all users that have signed into the application. 30](#_Toc469244968)

[Figure 25: If an administrator clicks on a user, the user’s information shows up as well as his or her reservations. Unlike the main reservation screen, all reservations for the user show up here: past, present, and future. This gives the administrator the ability to remove past reservations from the system. The administrator also has the ability to lock a user out of the system to prevent him or her from continuing to use it. 31](#_Toc469244969)

[Figure 26: The departments administration screen allows the administrator to create, view, update, and delete departments from the database. The departments appear in the dropdown inputs of the reservation screen. 31](#_Toc469244970)

[Figure 27: The VIP Categories administration area gives the administrator the ability to create, view, update, and delete VIP categories that will appear in the Category dropdown menu in the reservations request screen. 32](#_Toc469244971)

[Figure 28: The administrator can produce reports under the reports link. Initially, he or she will be presented a form to determine the criteria of the data shown on the report, including the date range. 32](#_Toc469244972)

[Figure 29: This is a report for the month of December. Two tables appear, one for the number of requests per department and one for the number of requests for each VIP category. The administrator can opt to copy the data to the clipboard, which can then be pasted into an Excel spreadsheet or any other kind of document. 33](#_Toc469244973)

[Figure 30: The waiting list is very similar to the standard reservation screen. The only exception is that reservations that are exclusively on the waiting list appear here. 33](#_Toc469244974)

[Figure 31: Administrators can manage gate codes. They can either create them manually or bulk upload gate codes in an Excel spreadsheet. 34](#_Toc469244975)

[Figure 32: The administrator can easily manage the Lots. Like many other aspects of the application, Lots can be added, edited, viewed, and deleted. These lots appear on the Edit screen for a reservation so that administrators can assign lots to the reservation. 34](#_Toc469244976)

# Chapter 1 –Introduction

## Overview

The MSCD696 - Database Practicum II course satisfies the Graduate Final Project requirement for graduation from the Master of Science in Database Technologies program. A total of 36 credit hours of course work is required to graduate. The MSCD696 course is taken as one of the last courses within the program and the student is required to participate in the Database Practicum for six months. The course facilitator enrolls the practicum students in the MSCD692 course automatically for the third eight-week term of the six-month practicum. The MSCD692 Database Practicum I course is optional and the student may or may not take the course depending upon whether the student needs an additional 3 credit hours to graduate. However, the student is still required to work through the practicum exercises from the Practicum I that include Linux, OEM, Oracle Apex, Microsoft SQL Server, and ASP.Net programming. The course facilitator may require proof that each student is proficient in these areas prior to enrolling the student in the MSCD696 course.

The purpose and objective of the Database Practicum project is give the students experience with development that takes place in an average software development company. The students much work together to achieve a common goal and objectives set forth from the stakeholders. The group will leverage the Software Development Life Cycle (SDLC) to complete the development objectives and deliver the product which the stakeholder requests, requirements modeling, analysis, design, software construction, software integration and system testing. The Database Practicum Project allows students to demonstrate the database knowledge that they have learned throughout their course work within the Database Technologies program. The fall 2016 DB Practicum will work on requirements modeling, analysis and design documentation for the project.

## Project Purpose

The purpose for this project was to aid the VIP parking administrators in better capturing reservations and organize all the data around the VIP parking. We will meet with the stakeholders to ensure that we are able to capture the necessary requirements to ensure that we are successful in completing the project and capturing the necessary data and allowing for the right levels of administration.

## Deliverables

The main deliverable for the 2016 Fall Database Practicum was to develop two different websites to meet the needs of different groups. The first of those sites was a Scholarly Database which housed research information which was taking place at the university. The second was a VIP parking database which was to house parking reservations for VIPs the visited the campus.

Each DB Practicum participant will:     

* Provide documentation (using the methodology outline below) at various lifecycle steps depending upon which phase of the project they are assigned. For example, the Fall Practicum 2016 team will document functional requirements, create analysis & design documentation, create Web pages, test and then implement the application.
* Provide two types of final reports that will document the accomplishments of the team. One consolidated team report and a series of individual reports from each team member that has only the accomplishments of the individual. It is okay to make the comprehensive report first and then let each individual customize the report for an individual report or create the individual reports and then consolidate into a team report (whatever works).

Web Development

* Create master style and template
* Create home page + Active Directory Authentication
* Requestor's Main Screen / Cancel Reservation
* Create/Edit Reservation
* View Reservation Screen
* Administrator's Main Screen + Create Reservation
* Admin Reviews a Reservation
* Department Administration
* Gate Code Bulk Upload
* User Administration
* Add to waiting list
* Reports
* Permit Lookup
* Waiting List Main Screen
* VIP Categories Administration
* Misc. Bug Fixes
* Planning the Project
* Meet with team to discuss ERD
* Create Use Cases and Review with Team
* Review Use Cases with Fiorella

## Stake holders

The following are the stake holders of our VIP parking project.

1. Bob Mason, Ph.D.

Associate Professor

Regis University - College of Computer & Information Sciences

M.S. in Database Technologies (DBT) Program Coordinator and

M.S. in Software Engineering and Database Technologies (SED) Program Coordinator

1. Lisa Lanford

Parking & Transportation Manager  
Office of Parking & Transportation Services | Division of Auxiliary & Business Services

1. Fiorella Perez  
   Parking Enforcement | Auxiliary & Business Services

## Repository Link

All contributions can be found in the project’s Gitlab repository at:

<https://gitlab.com/RU-CCIS-DB-Practicum-Projects/RU-CCIS-DB-Practicum-VIP-Parking-Project-Fall-2016>

# Chapter 2 – Methodology

While working in the VIP Parking project we didn’t have special adherence to the particular model of the Software Development Life Cycle but closer look to the process of development reveals that our working model quite resembles to Waterfall model which is one of the flavor of predictive modeling (Stephens, 2015). Various steps of this model and their implementation on our project can be expressed as

1. Requirements: Requirements to our project such as Business rules and requirements as well as development platform and its specifications were defined clearly by our stakeholders which will be discussed on later sections.
2. Design: We started with high level sketch of our user interface and navigation to the different methods by writing fully dressed code which then was used to generate of the mockup of our design. Required database was designed as entities and their relations were figured out.
3. Implementation: Design was broken down to the smaller units which were labeled as milestones in Gitlab. Each unit were developed and integrated in order to implement full project.
4. Verification: Each unit were tested in order to figure out if these are producing desired result and are efficient in taking care of exception occurred on process.
5. Deploying and Maintenance: Once we are done with the implementation and Verification, product is then handed over to the ITS Department. They will review the product, make necessary changes as desired and then deploy. Maintenance task will also be handled by ITS of Regis University.

## Functional Requirements

The following are the requirement for VIP Parking Project as intended by our stakeholder:

* An online, user friendly form that can be accessed through our webpage with the following fields:
  + Guest name.
  + (Possibly guest email address).
  + VIP Category (see current procedure document).
  + Date of visit.
  + Time of visit (allowing for parking time, etc.).
  + Name of requestor.
  + Requestor’s email address.
  + Sponsoring department.
* A way to funnel these request to our customer service area for approval (Approved, Declined/Processed) and processing.
* A way to assign location where the guest will be parking (location = VIP Lot –or- Any Lot).
* The system should be able to:
  + Generate a permit,
    - Name of Guest.
    - Date and time of visit.
    - Parking location.
  + Send the permit to requestor (and guest???).
  + Provide a gate code specific to the date of visit (we have a spreadsheet with random, codes we generated).
  + Provide a map to the VIP Lot or link to map at: <http://www.regis.edu/~/media/9B3B59E289BD495FA3D64210BF9774B7.ashx>.
* A way to generate reports by date, VIP category, spaces used, and department.
* Ability to enter large events into the system, without them having to go through the website.
* A way to limit requests to Regis staff and faculty.
* A way for the online user to know if the lot is already full or close to full (ex., red “light” means lot is at capacity, yellow “light” means the lot is close to full, green “light” means space is available). Maybe an incorporated calendar of reservations?

## Application Specific Requirements

Some of the specific requirement for application can be express as follows:

1. Access to reservation will only be limited to staff and faculty of Regis University. Authentications will be performed by application calling to the active directory group.
2. Microsoft SQL-Server database will primarily be used to store the data.

## Software Specific Requirements

Specific requirements for software development can be documented as:

1. Development environment will be Community edition of Visual Studio 2015.
2. Asp.net (4.5.2 or higher) will be used as the template.
3. MVC4 (or higher) with razor syntax should be used as presentation layer.

Fully Dressed Use Cases:

### Attempts to Authenticate

**Primary Actors:** Staff, Faculty

**Stakeholders and Interests:**

* *Requestor (Staff or Faculty):* wants to authenticate into the system in order to obtain a parking permit for a parking slot. He or she would prefer to use his existing Regis login.

**Preconditions:**

* Authentication system is configured by the application

**Success Guarantee (Post conditions):**

* User is authenticated and has authorization to request parking spots.
* User sees a screen that allows him to see any existing reservation with the ability to create a new slot reservation.
* User has the ability to log off
* User session is set

**Main Success Scenario**

1. User goes to main URL where he is presented with a login screen consisting of a username and password field, along with a submit button.
2. User enters invalid credentials [Alt 1: Invalid Credentials]
3. User leaves a field blank and submits form [Alt2: Missing Required Fields]
4. Unauthorized user tries to login [Alt 3: Unauthorized User Tries to Login]
5. User tries to login 3 times with valid credentials but fails [Alt 4: User is Locked Out]
6. User enters in valid credentials and submits the form
7. User information is checked against the user table
8. System logs the user logon and sets user session
9. User is brought to the main screen for authenticated users which consists of a list of existing reservations, the ability to create a reservation, and the ability to log off. [Use Case Ends]
10. User clicks link to log out
11. User is brought back to login form
12. User’s session information is cleared

**Alternative Flows**

*Alt 1:* Invalid Credentials

1. Flow resumes at Step 1 with a message indicating that either the username or password is incorrect.
2. System logs the invalid login attempt

*Alt 2:* Missing Required Fields

1. Flow resumes at Step 1 with a message indicating that all fields must be filled out.

*Alt 3:* Unauthorized User Tries to Log In

1. Flow resumes to Step 1 with a message indicating that the user does not have sufficient privileges to access the system
2. System logs the invalid login attempt

*Alt 4:* User is locked out

1. User receives a notification that his account has been locked with an email to contact to unlock the account
2. User emails the administrator
3. The administrator logs into the portal and goes to the user’s menu
4. The administrator clicks the link that belongs to the user
5. The administrator clicks an ‘unlock account’ button
6. The database is updated to indicate that the account has been unlocked
7. The unlock button is replaced by “Account has been unlocked” text
8. An email is sent to the user indicating that his or her account has been unlocked

**Exceptions**

If the system cannot reach the authentication system or the authentication system throws an exception, the exception is logged and the user is notified that there’s an issue with the system. The system administrator should be notified.

**Special Requirements**

Authentication form should have reasonable sized input fields and labels.

**Open Issues**

* Do we have access to the authentication system?
* If so, do we need configuration information to interact with the authentication system?
* Is there a maximum number of login attempts to avoid hacking? -
* Should a captcha be used for security purposes?

### Submits a Request for a Parking Spot

**Primary Actors:** Staff, Faculty, Administrator

**Stakeholders and Interests:**

* *Requestor (Staff or Faculty):* wants to obtain a parking permit for a specific event
* *Administrator:* wants only staff and faculty to obtaining parking permit, wants to be notified whenever a reservation is made, wants a record of who is submitting a request for a permit, wants to know which spots are taken up. The administrator can also perform the same roles as the requestor and make/update/cancel reservations

**Preconditions:**

* User is authenticated by system

**Success Guarantee (Post conditions):**

* User receives a confirmation that his request is being reviewed.
* Administrator receives an email mentioning that a request needs reviewing
* A record has been inserted into the database with request information

**Main Success Scenario**

1. User clicks the create request button
2. All slots are filled up [Alt2: No slots Left]
3. User is presented a form with fields appropriate for acquiring a parking reservation including a dropdown with the events to choose from
4. User fills out form, which includes the number of slots they’d like to request and the lot they want, and hits the submit button
5. Some required fields are not filled in [*Alt 1:* Required Fields Need Filling Out]
6. Form is successfully submitted
7. The slots may fill up between the actions of the user making his decision and hitting submit. [*Alt 2:* No Slots Left]
8. User receives a confirmation that his request is being reviewed
9. Administrator(s) and requestor receives an email that a request needs reviewing [*Use Case Ends*]
10. User clicks the cancel button next to his or her reservation
11. A popup shows up asking the user if he is sure he wants to cancel the reservation
12. User clicks OK in the popup
13. Screen is updated with parking reservation removed
14. Information is about the reservation is cleared from the database
15. The administrators receive an email notifying them of the cancellation.

**Alternative Flows**

*Alt 1:* Required Fields Needs Filling Out

1. Flow resumes at step 1 with an error message next to each empty required field and existing filled out fields pre-populated

*Alt 2:* No slots left

1. User is notified that there are no available slots and to check back later to see if a slot has opened up.
2. User is given an option to be put on a waiting list
3. If user chooses to be put on waiting list, take user to a form to add information to waiting list. This form will be pre-populated with any information the user may have already entered.
4. User submits waiting list form and information is added to the database. User is notified that he has been added to waiting list and administrator is emailed about waiting list request.

**Exceptions**

If the system is not able to save the reservation, log the error and send an email to the system administrator, and let the user know that there is a system issue. Use case ends.

**Special Requirements**

Form should have reasonable sized input fields and labels. Required fields should be clearly marked with asterisks on the labels.

**Open Issues**

* None

### Parking Requested Reviewed

**Primary Actors:** Administrator, Requestor (Staff or Faculty)

**Stakeholders and Interests:**

* *Administrator:* Reviews information about a parking requests and approves or declines it.
* *Requestor:*Submitted a request for a parking slot and is looking to obtain a permit

**Preconditions:**

* Administrator has logged in with an administrator account
* Staff or faculty has submitted a request for a parking slot

**Success Guarantee (Post conditions):**

* Staff or faculty given a confirmation screen that the user has been granted or denied a permit
* Permit information, such as a unique number, has been generated and put in the database
* Staff or faculty receives an email with an attached permit

**Main Success Scenario**

1. Administrator receives an email saying that someone has requested a parking permit
2. Administrator logs into portal using his or her Regis credentials
3. Administrator sees a list of open requests and clicks one
4. Administrator sees information about the request taken from the form the requestor filled out
5. Administrator hits decline button [*Alt 1:* Administrator Declines Permit]
6. Administrator reviews information, which also includes an auto-filled gate code that the administrator specified in a gate code screen (can be changed in the form), and hits an approve button
7. Administrator receives a popup asking if he’s sure he wants to approve the request
8. Administrator hits cancel on the popup [*Alt2:* Administrator Cancels Approval]
9. Administrator hits OK on the popup
10. Administrator receives a confirmation screen saying that the user has been approved for his or her parking slot with permit number(s)
11. Approval is stored in the database
12. An email is sent out by the system to the requestor and administrator with parking permit information and gate code. [*End Use Case*]
13. If the parking lot is determined to be full at this point, let the administrator know and send an email to admin notifying that parking lot is full.

**Alternative Flows**

*Alt 1:* Admin Declines Permit

1. Administrator receives a popup asking if he’s sure he wants to decline the permit
2. Administrator hits cancel on the popup. [*Alt 3:* Administrator Cancels Decline]
3. Administrator hits OK on the popup
4. Administrator receives a confirmation screen saying that the user has been declined for a parking slot
5. An email is sent out by the system to the requestor (not the guest) with a message saying the parking request has been declined [*End Use Case*]

*Alt 2:* Administrator Cancels Approval

1. Flow resumes at step 4

*Alt 3:* Administrator Cancels Decline

1. Flow resumes at step 4

**Exceptions**

* If the system is not able to save the status of the request (accepted/declined), log the error and send an email to the system administrator, and let the administrator know that there is a system issue. Use case ends.
* If the email system is offline, log the error into the database and let the administrator know so that he can resubmit the email or send it manually

**Special Requirements**

* None

**Open Issues**

* Should the permit be sent as an attachment or as text in the email? – Attachment preferred
* Should the permit even be sent in the email or should a link be sent to the requestor instructing him to log into his or her account to obtain the permit (more secure but more steps to implement) – Send in an email
* Does the administrator choose a parking slot from a list of available slots or just enter a slot number into a freeform text field?

### Generating Report

**Primary Actors:** Administrator

**Stakeholders and Interests:**

* *Administrator wants to generate report from the system.*

**Preconditions:**

* Application is connected to the database.
* Administrator is authenticated by system.

**Success Guarantee (Post conditions):**

* Admin is able to generate specified report in desired format.

**Main Success Scenario**

1. Admin clicks Reporting button.
2. Reporting page containing input fields Date, VIP Category, Department is displayed to admin.
3. Admin doesn’t provide any input and hits submit. (Alt .1 Missing field).
4. Admin doesn’t select date (start date, end date) from calendar. (Alt 2. Date error)
5. Admin selects date (start date and end date) from calendar.
6. Admin selects value from VIP category or Department or both or none.
7. Admin hits submit.
8. No data is displayed. (Alt. 3 NO Data Found)
9. Required report is generated on screen.
10. Admin hits save and selects desired format (pdf, Excel spreadsheet, etc.).

**Alternative Flows**

*Alt 1:* Missing Fields

1. If none of the field is specified then system displays error message stating “ At least one field required”.
2. System directs to step 2 of Main success scenario.

*Alt 2:* Date Error

1. Admin gets error message stating both fields for date needs to be filled up.
2. 2 System directs admin to step 2 of Main success scenario.

*Alt 3:* No Data Found

1. If no data is found from the query then system directs admin back to step 2 of main success scenario while presenting error message, “No data found please select different criteria”.

**Exceptions**

* If application failed to connect database, then logs error and displays “failed to connect database” message.

**Special Requirements**

* Mandatory input field such as date range should contain asterisk on the label so that it can’t be missed easily.

**Open Issues**

None

### Admin Administers Events

**Primary Actors**: Administrator

**Stakeholders and Interests:**

Administrator will be able to load new events, alter existing events and delete events as needed.

**Preconditions:**

Administrator has successfully logged into the system.

**Success Guarantee (post conditions):**

Administrator is able to perform any of the add/delete/change functions to the events within the application.

**Main Success Scenario:**

1. Administrator is able to add a new event.
2. Administrator is able to delete and existing event.
3. Administrator is able to change an existing event.
4. Any of the above options result in the change requested on the event calendar along with a success message about the change made to the event.

**Exception:**

Incorrect value is attempted to be loaded and or changed from the events calendar.

**Open issues**

Number of parking spots open for this event?

If we limit the spots for the event, what is the default value or should the field be left blank?

Check for dupe events before inserting into the calendar.

### User views map

**Primary Actors**: Faculty, Staff

**Stakeholders and Interests:**

Faculty: Wants to know the parking locations of Regis University.

Staff: Wants to know the parking locations of Regis University.

**Preconditions:**

User has successfully logged into the system.

**Success Guarantee (post conditions):**

User is presented with a map showing different parking lots available in campus.

**Main Success Scenario:**

1. User clicks on Map link.
2. User is presented with map of Regis parking lots.

**Exception:**

If system is unable to display the map, then automatically error log is created and email sent to the administrator.

**Open issues**

Is it necessary to be logged in order to see the map?

Can we make map interactive so that user can click to required lots in order to view any available parking spots?

### admin Bulk loads data

**Primary Actors**: Administrator

**Stakeholders and Interests:**

Administrator will be able to take a CSV or Excel file with the correct values mapped to the database and load them.

**Preconditions:**

Administrator has successfully logged into the system.

**Success Guarantee (post conditions):**

Administrator is presented with a dialog box confirming the loading of records. A record count of the rows loaded.

**Main Success Scenario:**

1. Administrator is able click an upload button which prompts to select a file.
2. File is uploaded and processed and success or failure dialog box is presented

**Exception:**

Incorrect value is attempted to be loaded into a field.

**Open issues**

A preview of the load be presented to the user to ensure the data has mapped and loaded correct should no data validation errors be thrown?

Partial load of the data up to the exception line in the data file?

Data Validation to prevent duplicates from being loaded?

## Use Case Diagram

Following used case diagram is created based on used cases of above which is also an excellent illustration of structure and feature of our project.

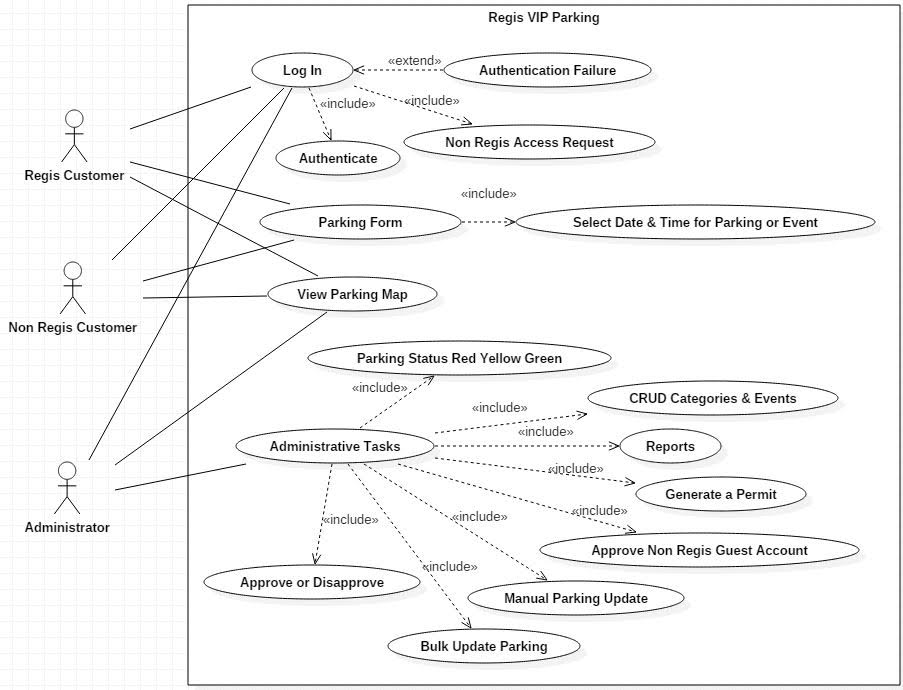


Figure : VIP Parking Use Case Diagram Depicting all Actors and Application Interactions.

## Flow Diagram

Flow diagram for overall application can be depicted as:

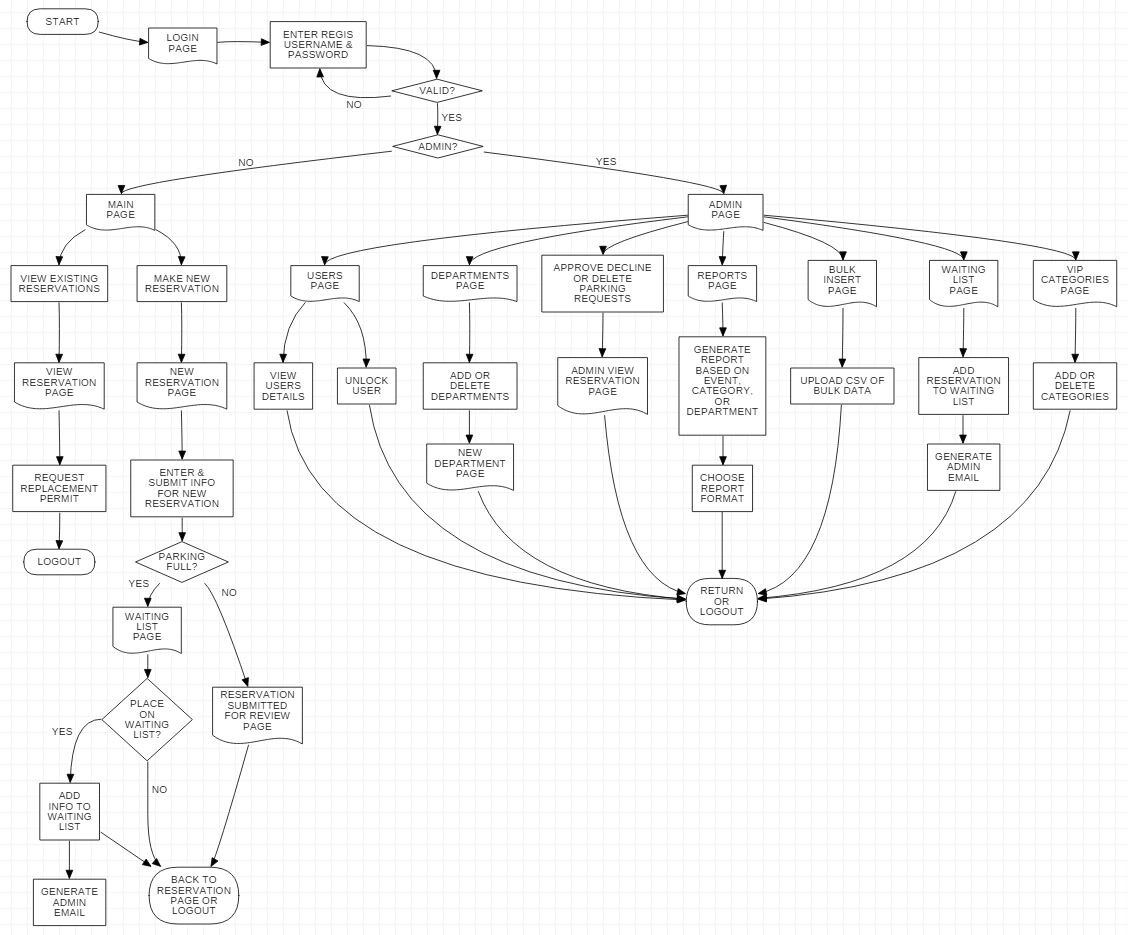


Figure : Flow Diagram for Regis VIP Parking Application.

## Entity Relation-ship Diagram:

After discussing about the requirements and the possible entities our final draft of ERD was created in SQL-Server which then serves as the base for the implementation.

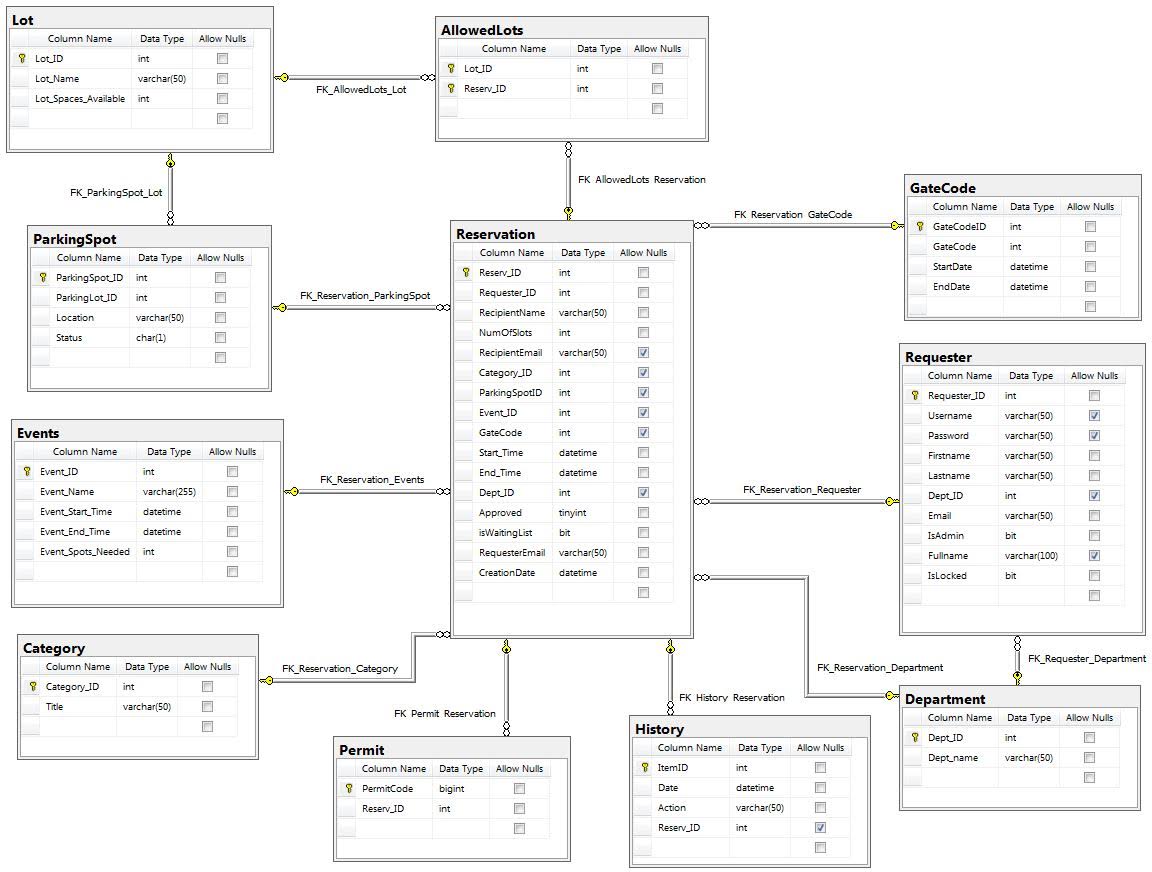


Figure : Entity Relationship Diagram for the VIP Parking Application.

## Mock-ups

After creation of used cases, flow chart and ERD then it’s time to think about designing end user interfaces. In order to streamline our Implementation process, mockups were created which helps as visual guide while coding.

1. **Log-in Screen**: Following is the screen when any one visits parking website will be presented with.

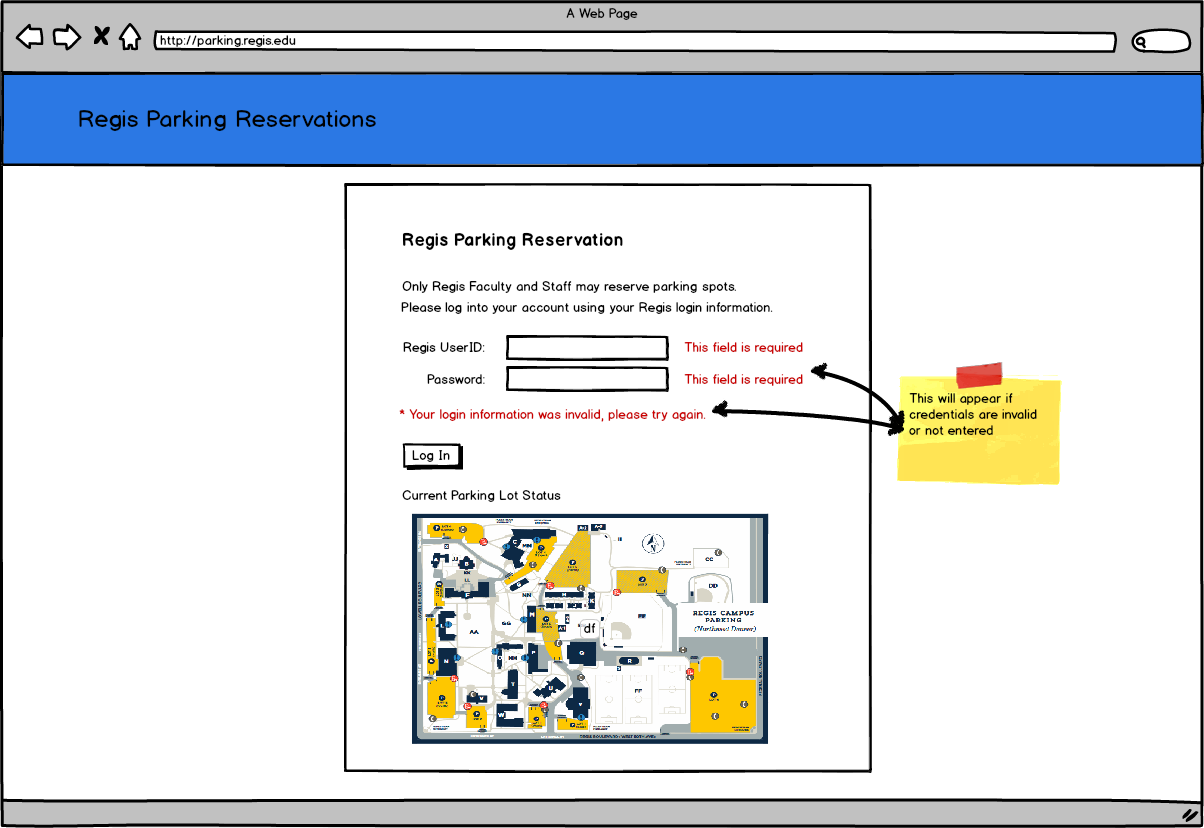


Figure 4: Login Page Wireframe.

1. **Home Screen:** As user log in successfully then s/he will be presented with the second screen which will show existing reservations if any. It will also have a radio button in order to create a new reservation.

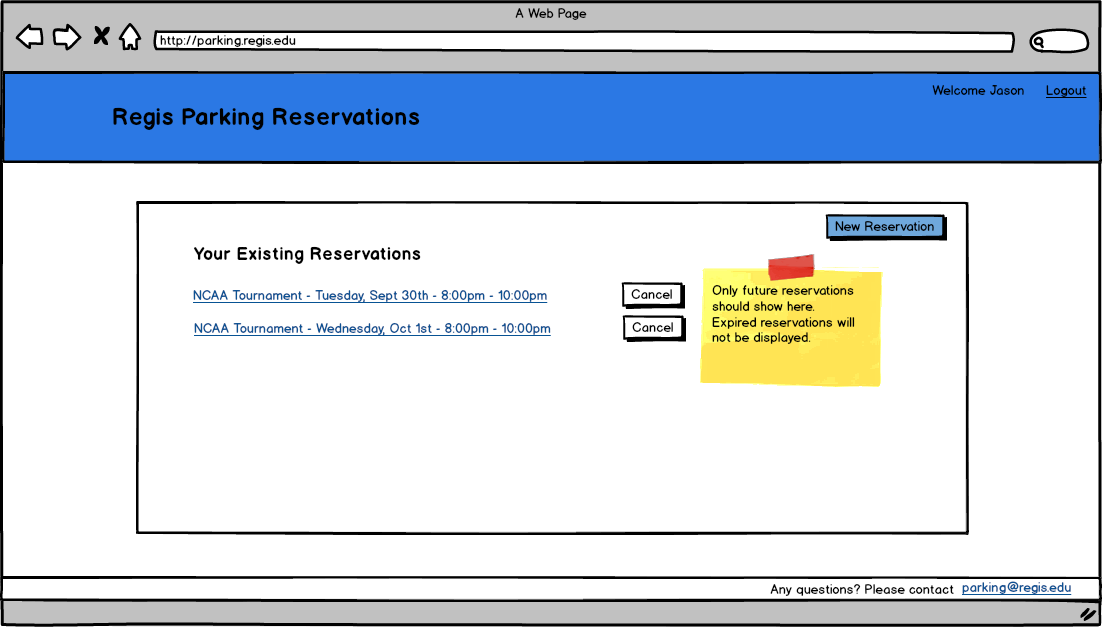
****

Figure 5: Home Page Wireframe.

1. **New reservation:** Following figure shows mockup for creating a new reservation.

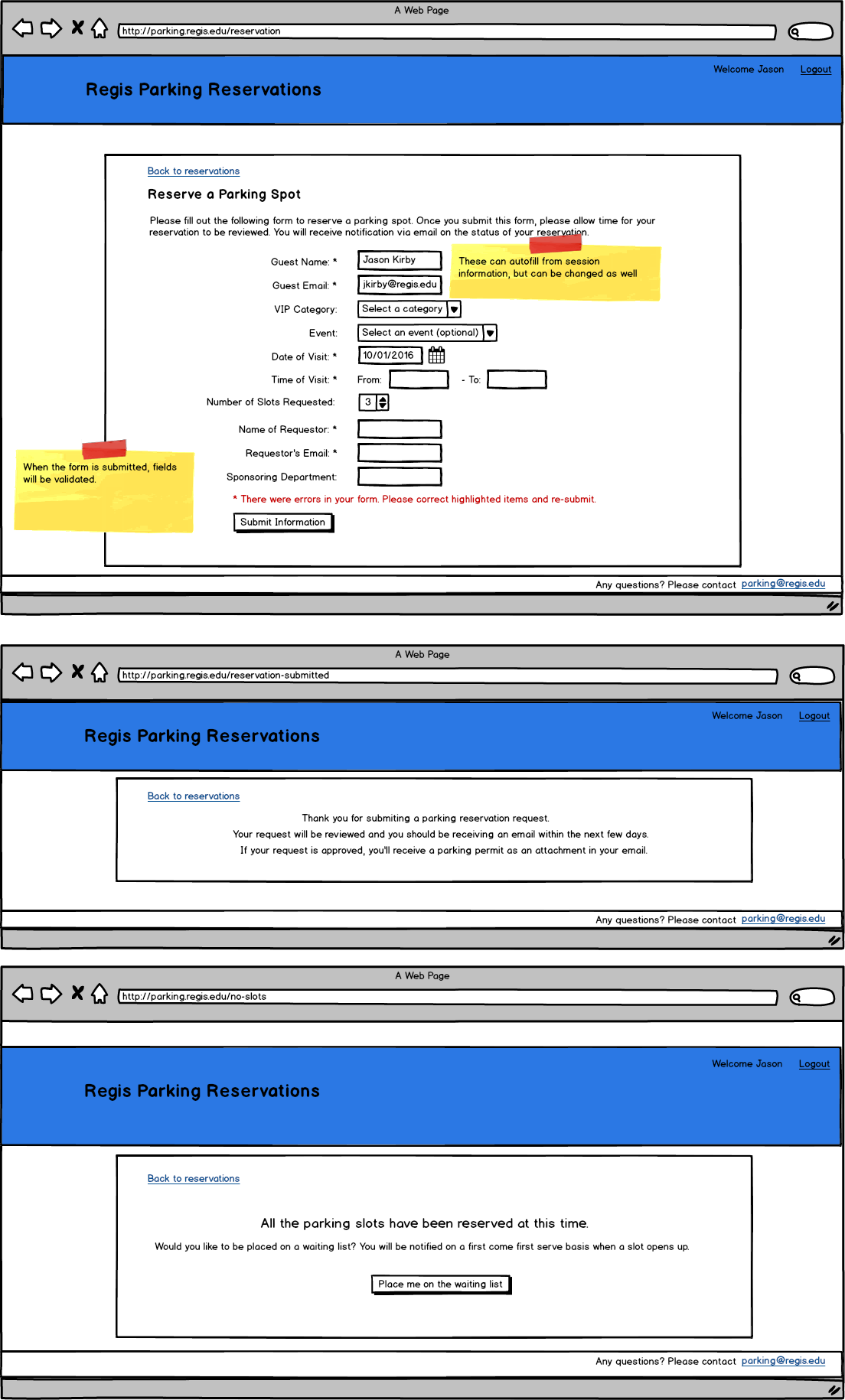


Figure 6: Depicts the New Reservation Page and associated Feedback Pages.

1. **View reservations:**  Below screen shot shows sample details of a reservation.

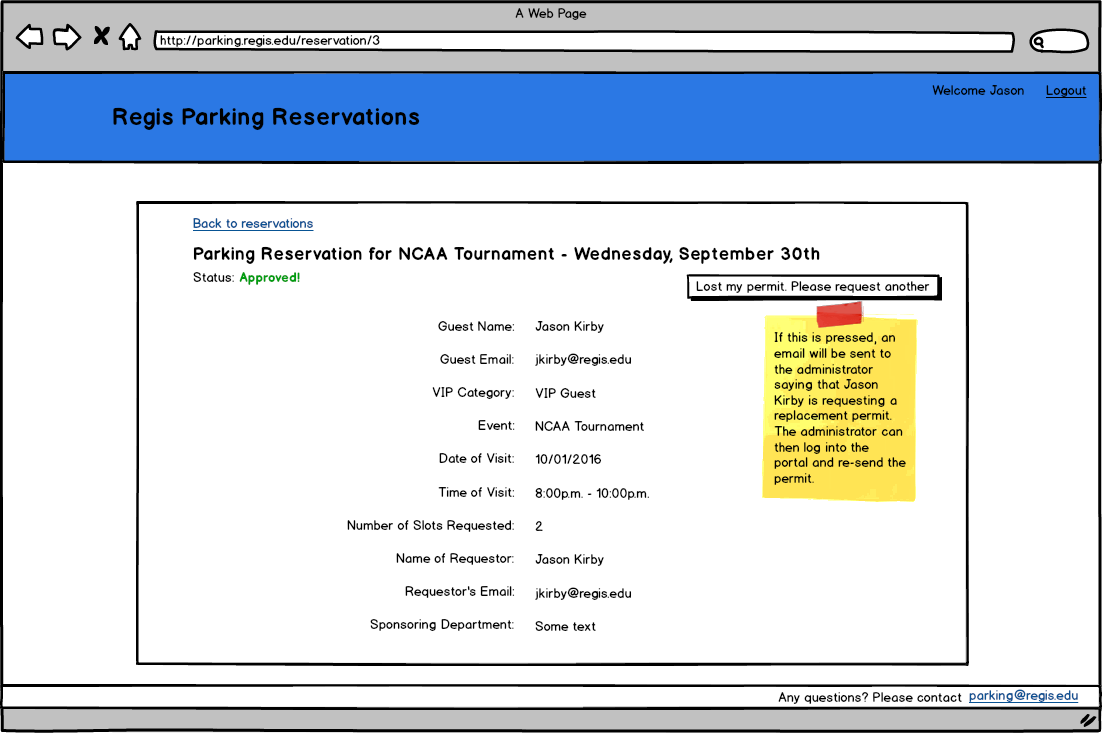
****

Figure 7: View Reservations Page Wireframe.

1. **Admin Home page:**  It is the screen when someone logs with admin privilege.

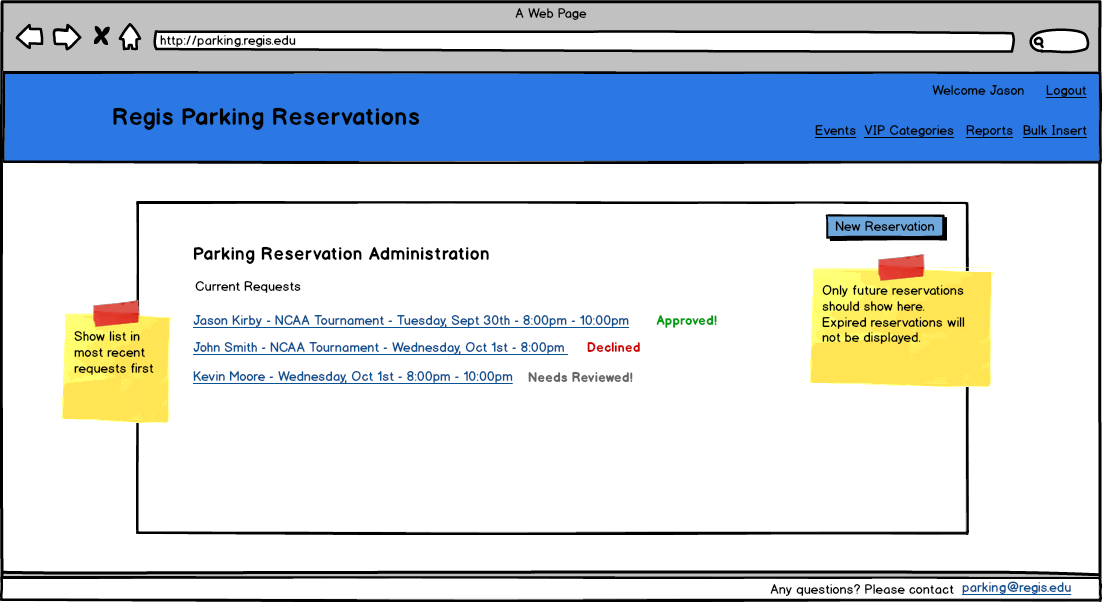
****

Figure 8: Administration Home Page Wireframe.

1. **Admin reservation page:** This page entails how each reservation is seen from admin side.

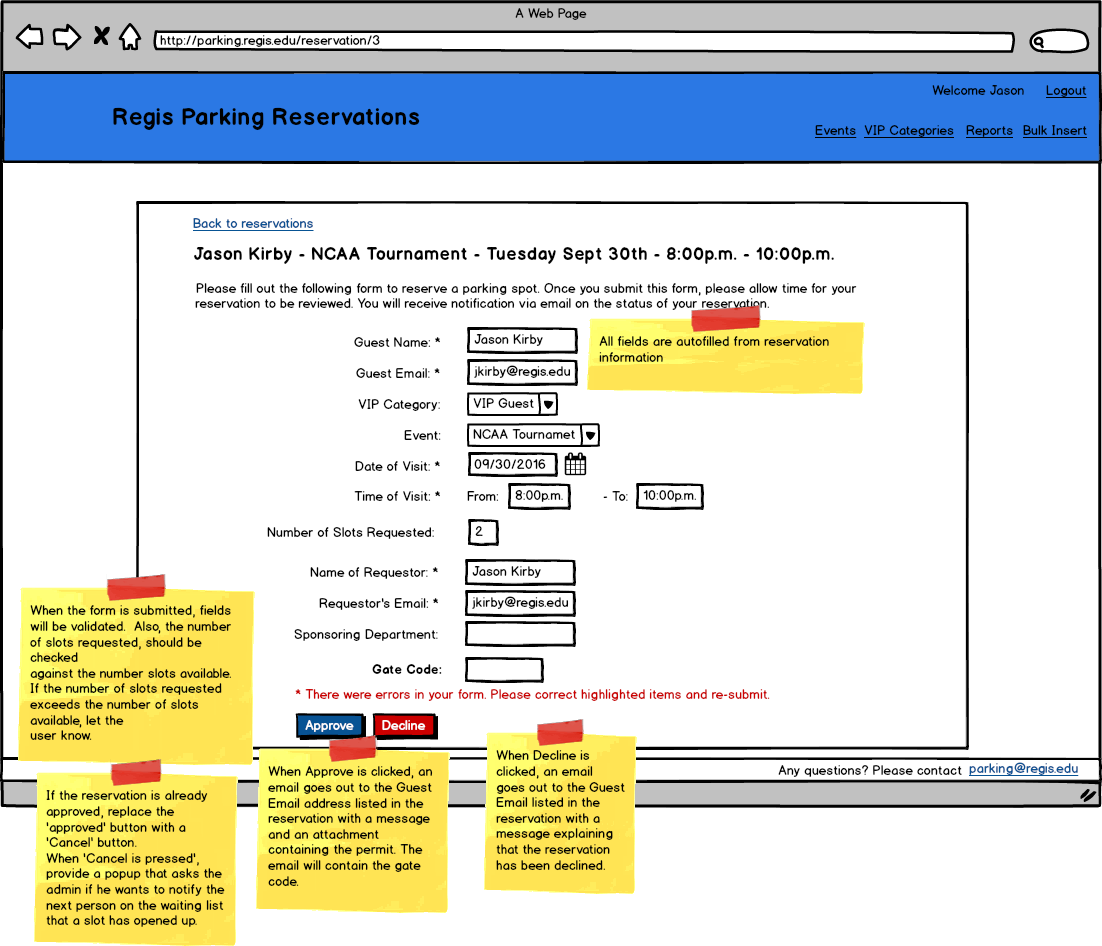
****

Figure 9: Administration Reservation Page Wireframe.

1. **Event Admin page:**  From this page Admin will be able to create, update and delete events as necessary.

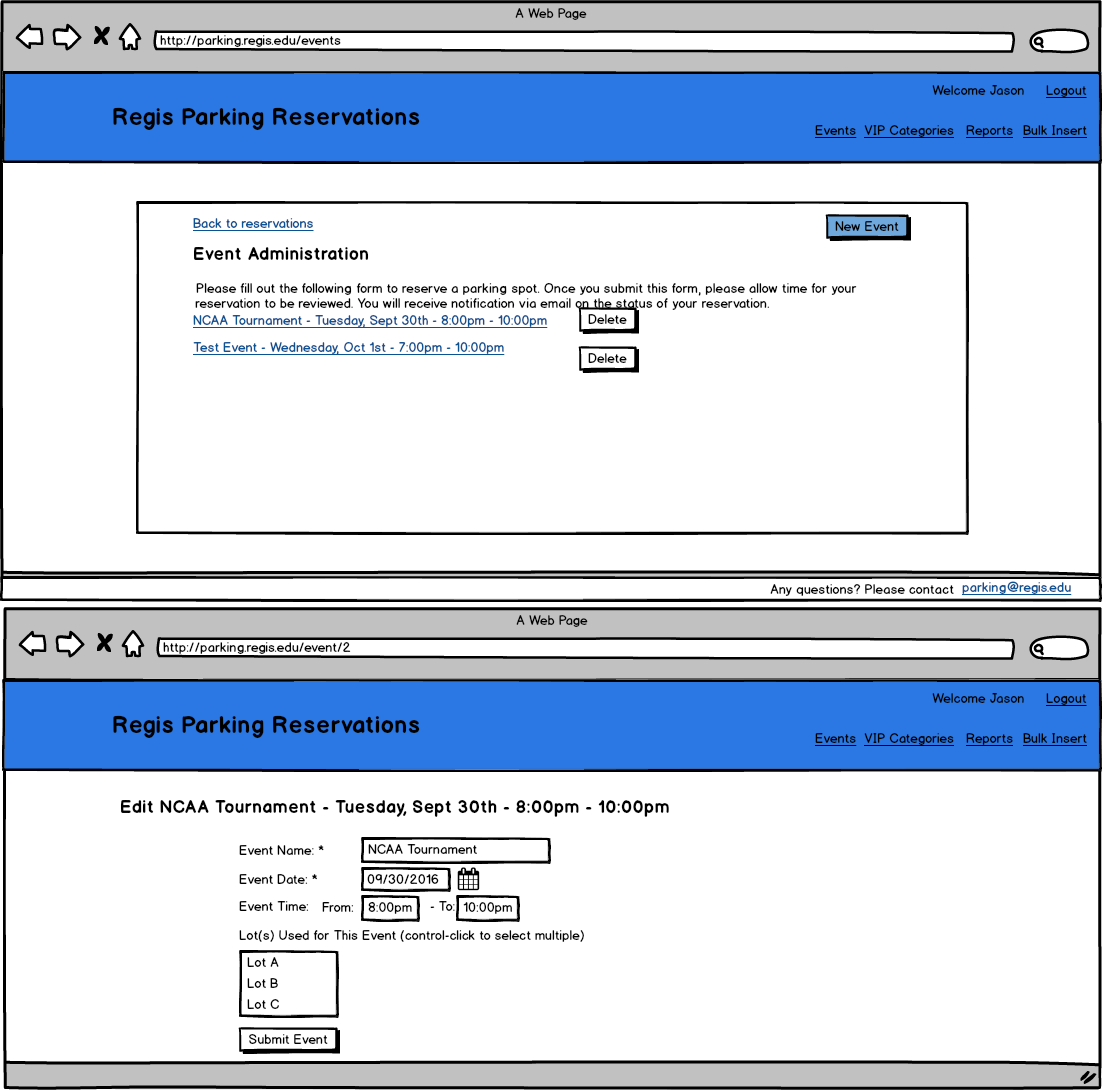
****

Figure 10: Event Administration Pages Wireframes.

# Chapter 3 –Results

This section demonstrates and describes various actions that are performed by the VIP Parking web application. At various points throughout the development, some of these tests had failed and have since been corrected. The screenshots below exhibit assorted pieces of functionality the application administers.

The majority of testing on this application was user acceptance testing contributed by the team. A testing specification was developed before and during the implementation phases. At distinct points, old tests required retesting because refactoring occurred throughout the lifetime of development. The comprehensive testing suite attempts examining every possible branch of code, but some repeated code bits, such as administrative authorization to methods, were tested on just some of the functions. It was assumed that if a bit of code worked of a sample of situations, it would work for all situations. Subsequently, code that was strictly produced through Visual Studio scaffolding, and unmodified, was minimally confirmed as it was considered that it had been previously analyzed by Visual Studio developers. Due to time constraints, unit and integration tests are not incorporated into the code. However, the user acceptance tests can be translated into these styles of tests for future development.

**Screenshot Demonstrations**

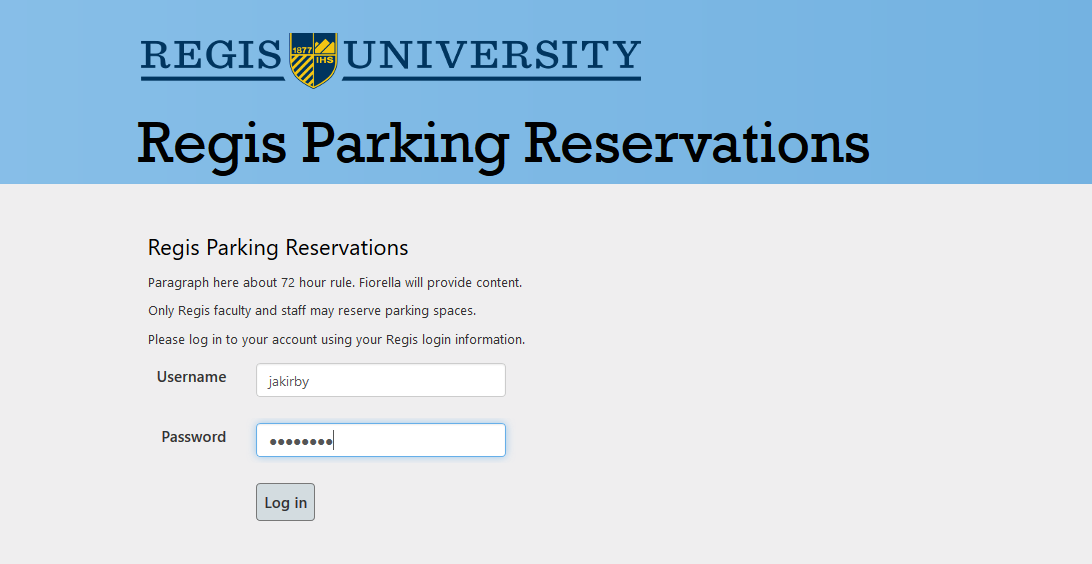


Figure : Login screen allows faculty and staff to authenticate against Active Directory. Administrators can also log in through this form. This screen will notify the user if the login credentials are invalid or if the account is locked.

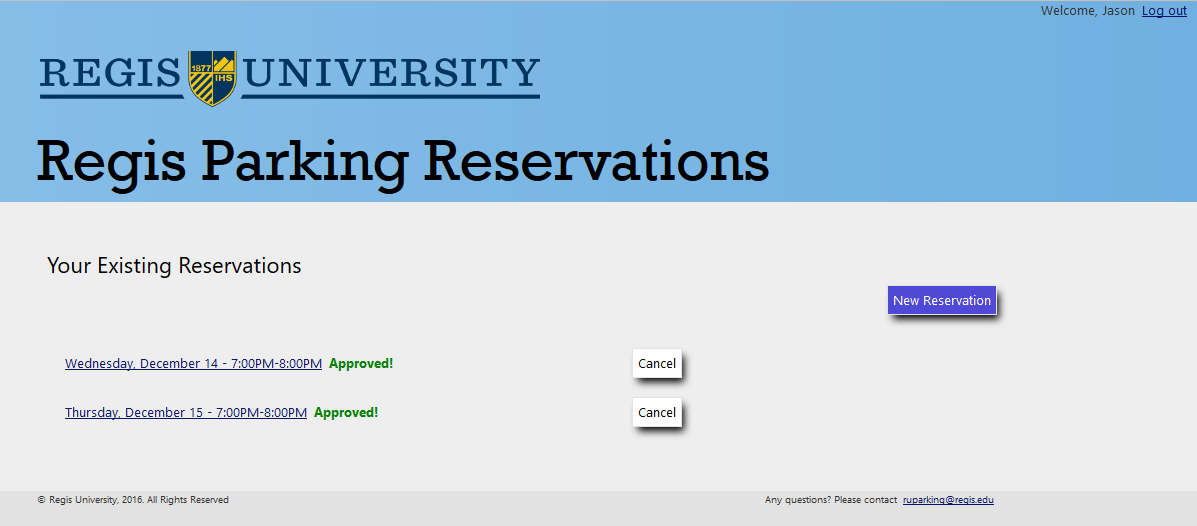


Figure : Reservations screen. Once a user logs in, this page is displayed, which lists the user’s current and future reservations. Any reservations that have expired will not appear on this list. The user has the opportunity to create new reservations or view the reservations.

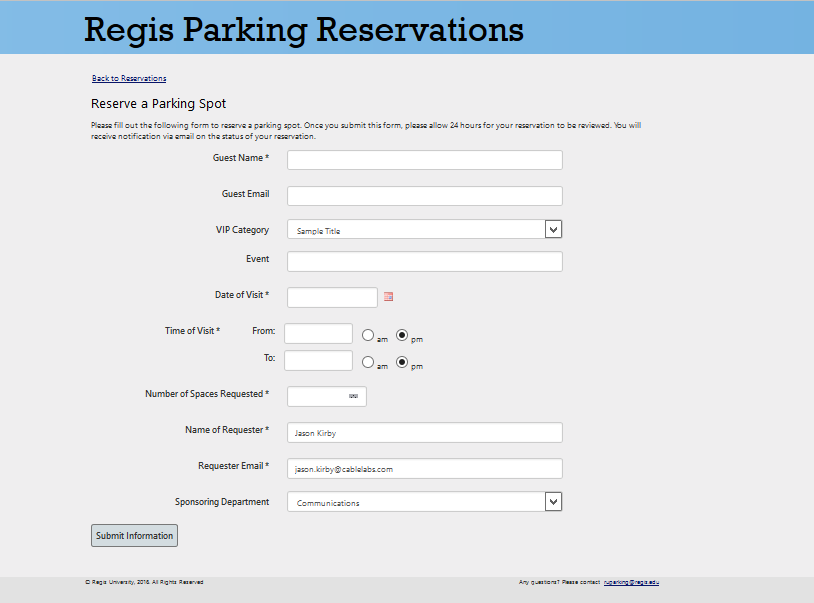


Figure : When a user needs to create a reservation, this form is available to complete. Once submitted, all fields are validated to make sure proper information is sent to the system. If the form validates, the information is stored to the database and an email.

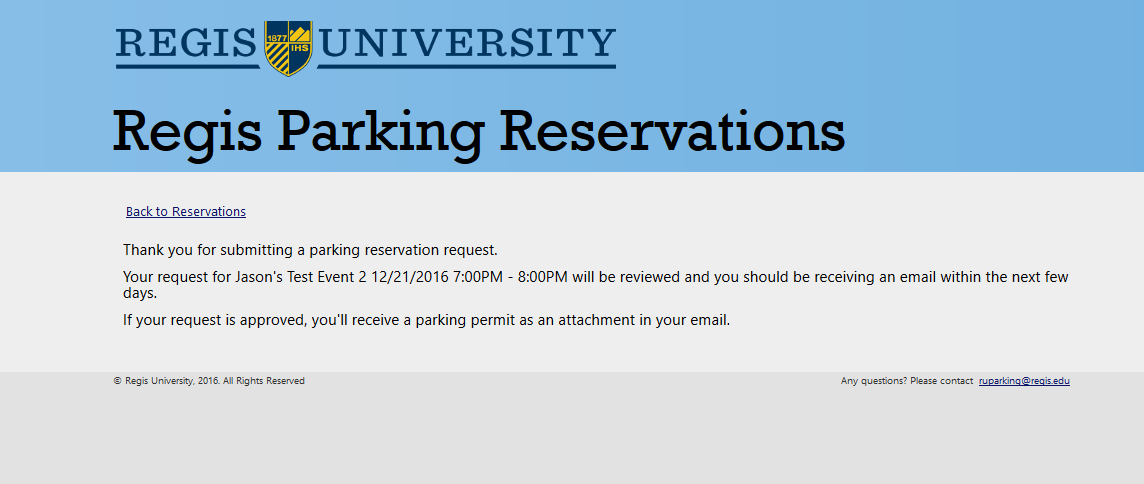


Figure : This is the confirmation screen the user receives once a reservation is requested. It serves to notify the user of the request that it was received, and a message indicating that the reservation needs reviewed.

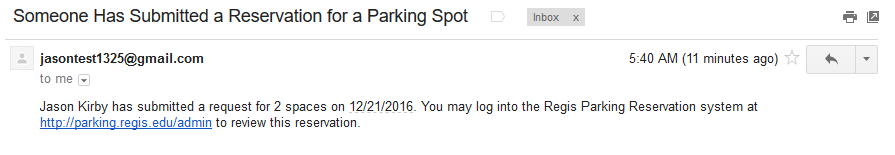


Figure : This is the email the administrators receive when a request for a reservation has been submitted.

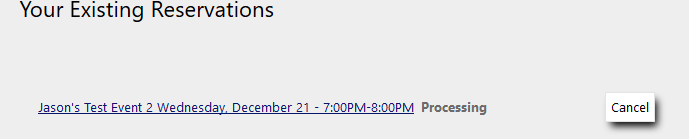


Figure : The new reservation request shows up on the user’s dashboard with a status of processing.



Figure : The user can click on the reservation link in order to view the reservation details. The status is included as well as all of the information the user filled out. Once the reservation is submitted, only the administrator can edit it. The user can only view it.

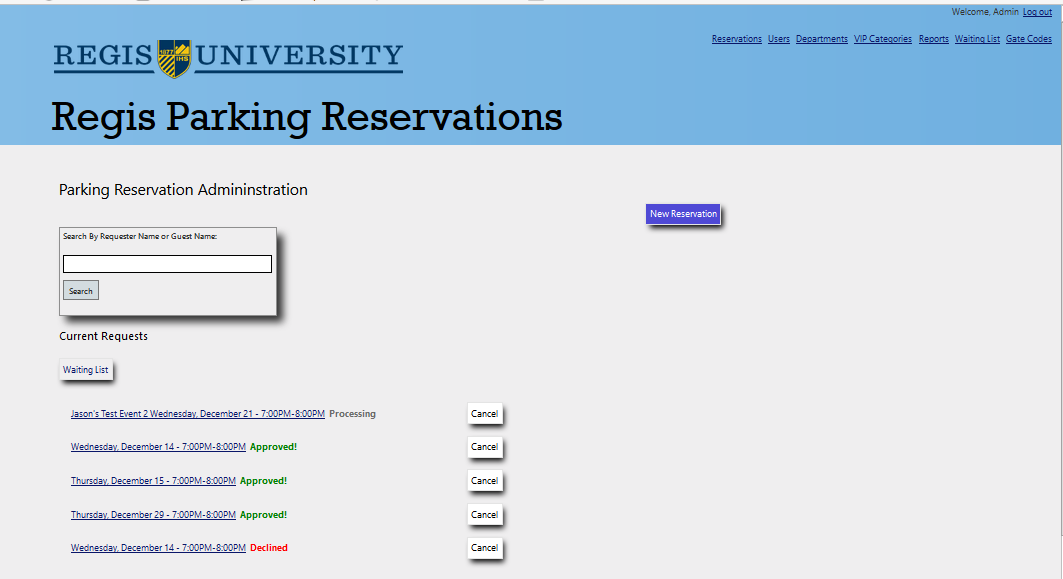


Figure : This is the administrator’s dashboard. It looks similar to the user’s dashboard but contains more pieces of information. There is a search bar to allow him or her to search for reservations by name, as the list may become very long. The header menu also contains links specific to the administrator that the normal user will not be able to access. There is a button to access the waiting list. Just like normal users, the administrator can create, view, and cancel reservations.

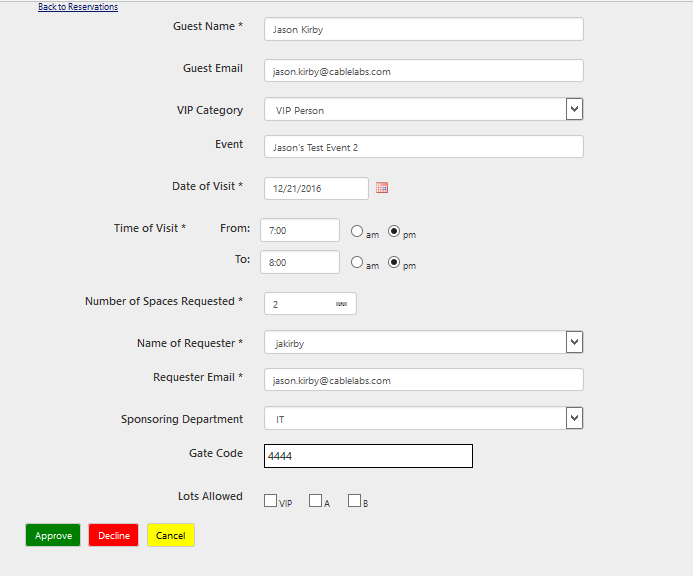


Figure : If the administrator clicks on a reservation, an edit form shows up where he or she may make changes to it. The Gate Code field is automatically populated with the data in the database based on the date of the reservation but can be changed for special circumstances. The Lots Allowed can also be checked to notify the requester which lots where vehicles can be parked. If the reservation is approved, the requester and administrators will receive a confirmation email with parking permits attached to it, as well as the gate code and allowed lots.

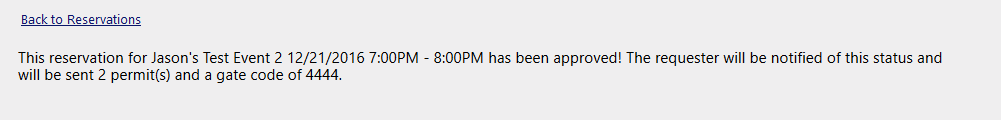


Figure : The administrator will receive a confirmation message after the reservation has been approved or declined.

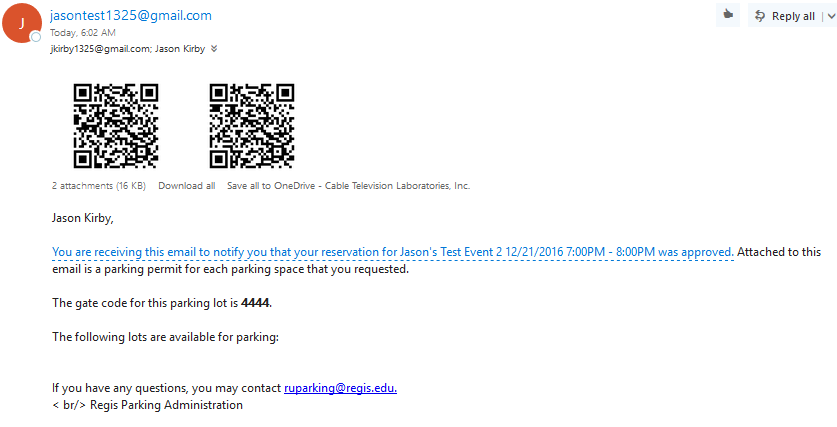


Figure : The requester and administrators will receive this email when a reservation is approved. It contains information about the reservation, which in Microsoft Outlook, will allow the user the option to click the information to store it on their calendar. It also contains the gate code for that time slot, as well as the parking lots that are available. Two permits are attached to this email because two parking slots were requested.



Figure : This is the parking permit that is produced upon reservation approval. This can be scan be a parking attendant using any standard smart-phone QR code reader. The QR code contains the direct link to the reservation that will redirect the mobile browser to the reservation record. Only administrators will be able to access this record.

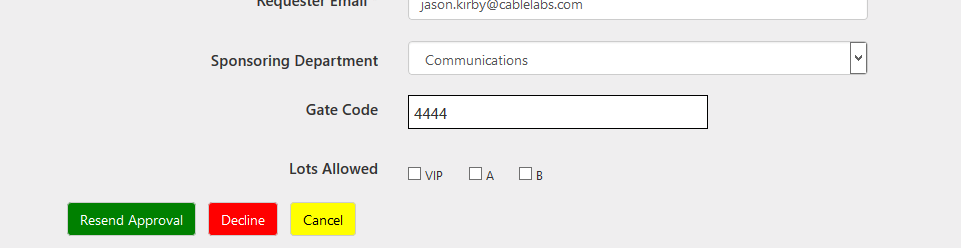


Figure : If a reservation is already approved, the administrator has the options to resend the permits, decline the reservation, or cancel it. Cancelling the reservation removes the reservation from the system. In all cases, a pop-up window will alert the user of the action he or she is about to make, giving the option to avoid the transaction.

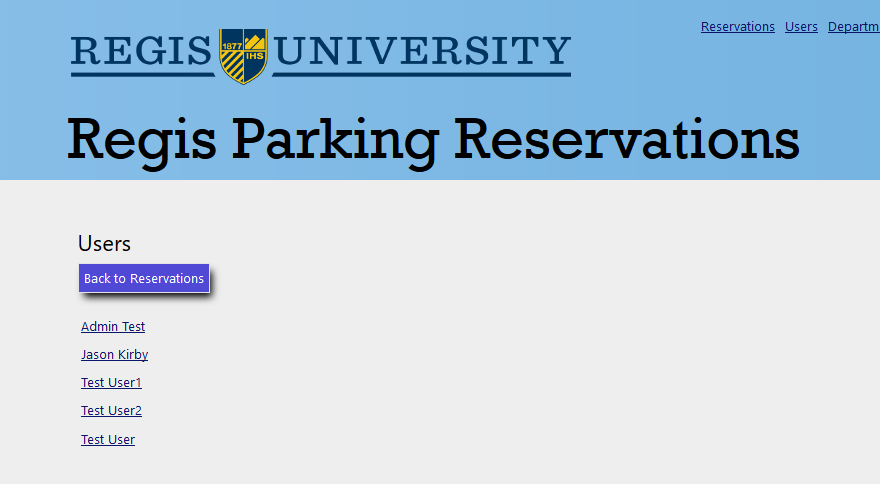


Figure : The user administration screen shows a list of all users that have signed into the application.

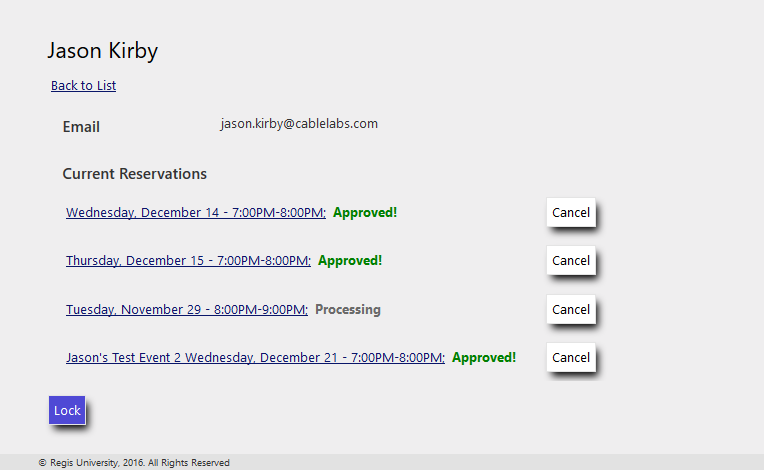


Figure : If an administrator clicks on a user, the user’s information shows up as well as his or her reservations. Unlike the main reservation screen, all reservations for the user show up here: past, present, and future. This gives the administrator the ability to remove past reservations from the system. The administrator also has the ability to lock a user out of the system to prevent him or her from continuing to use it.

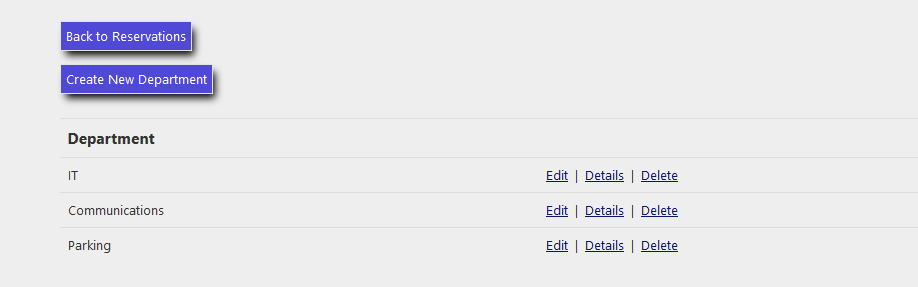


Figure : The departments administration screen allows the administrator to create, view, update, and delete departments from the database. The departments appear in the dropdown inputs of the reservation screen.

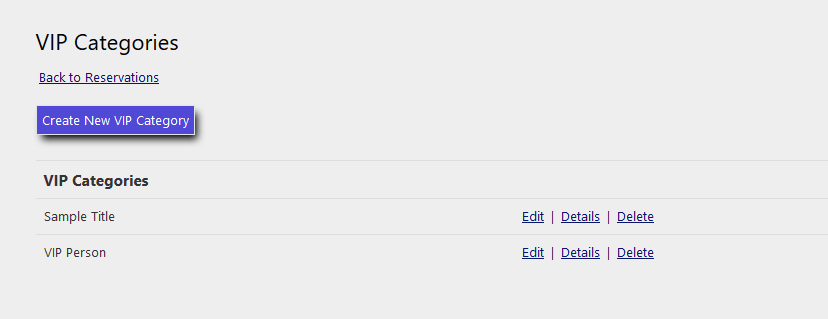


Figure : The VIP Categories administration area gives the administrator the ability to create, view, update, and delete VIP categories that will appear in the Category dropdown menu in the reservations request screen.

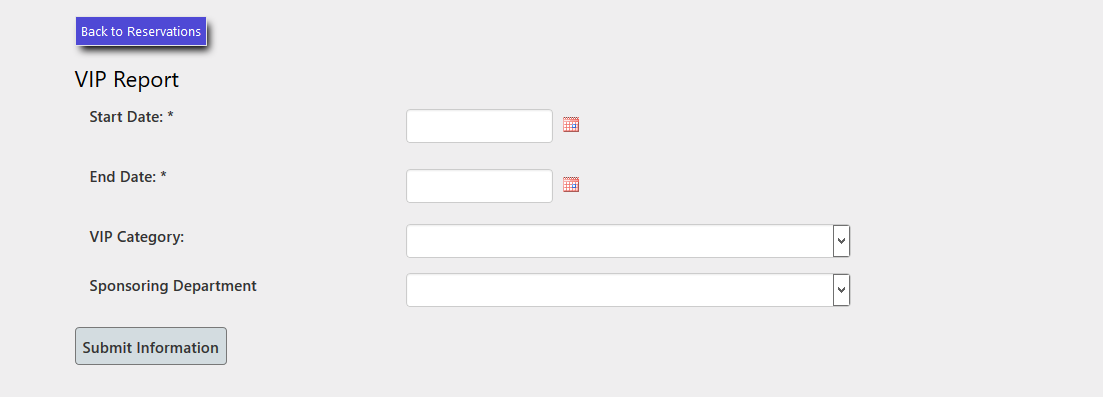


Figure : The administrator can produce reports under the reports link. Initially, he or she will be presented a form to determine the criteria of the data shown on the report, including the date range.

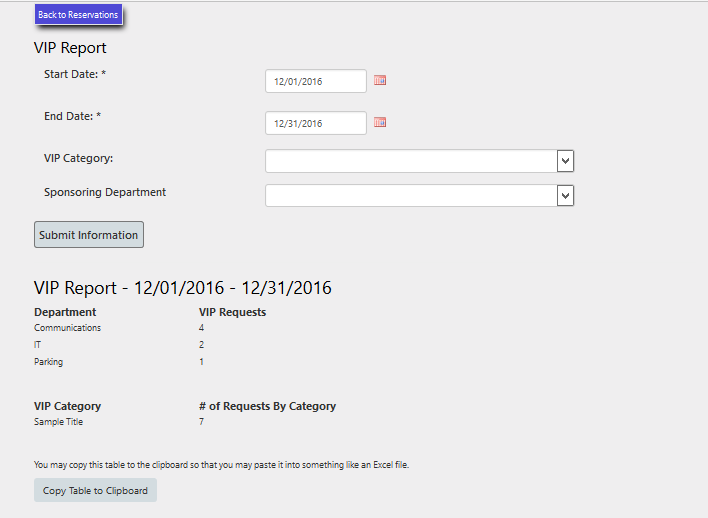


Figure : This is a report for the month of December. Two tables appear, one for the number of requests per department and one for the number of requests for each VIP category. The administrator can opt to copy the data to the clipboard, which can then be pasted into an Excel spreadsheet or any other kind of document.

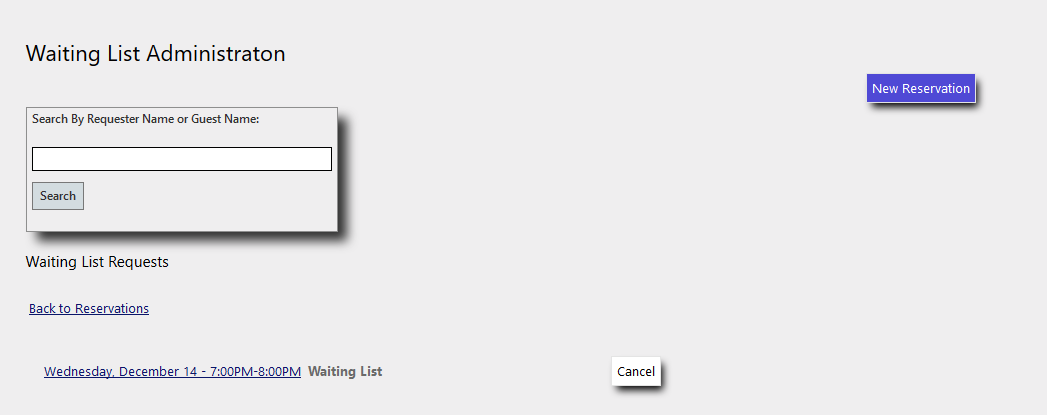


Figure : The waiting list is very similar to the standard reservation screen. The only exception is that reservations that are exclusively on the waiting list appear here.



Figure : Administrators can manage gate codes. They can either create them manually or bulk upload gate codes in an Excel spreadsheet.



Figure : The administrator can easily manage the Lots. Like many other aspects of the application, Lots can be added, edited, viewed, and deleted. These lots appear on the Edit screen for a reservation so that administrators can assign lots to the reservation.

# Chapter 4 – Conclusions

The purpose of the VIP Parking Project was to create a useful and user-friendly web interface which would allow Regis personnel to reserve and monitor VIP parking spaces. This goal was successfully completed and resulted in a usable web interface for VIP Parking control. The web application in its current form meets the majority of functional requirements provided from the Regis parking authority. Regis stakeholder feedback was actively sought for and implemented throughout the application development.

The following can be performed by the web application:

* Only Regis faculty and staff members can login using their Regis credentials in Active Directory.
* Reservations can be easily viewed, made, or canceled by users based on user, category, or department.
* The parking administration can view and manage reservations by approving, disapproving, or deleting.
* Emails are automatically generated and sent to parking administration personnel when a reservation is made.
* Once a reservation is approved, and email is auto generated and sent to the original requester.
* Gate codes can be generated, or if lost, regenerated by users.
* Batch parking uploads can be performed.
* Reporting can be quickly generated based on date, VIP category, spaces, or department.
* The site styling is similar to styling used by other Regis sites.
* The development team was mindful of and implemented Americans with Disabilities Act (ADA) requirements.
* Technologies that align with Regis ITS requirements were used to develop and deploy the site (Microsoft ASP.NET, Razor markup, and MVC architecture).

Unfortunately, time did not allow perfecting of the application. If time allowed, more testing should have been completed. Small, scale unit testing was completed during development, but a formal unified test plan was not executed on the final product. Parking administration personnel also requested that a map be made available, and a red, yellow, green status to quickly see if a lot is full, near full, or available. Refactoring, more thorough testing, maps, and red, yellow, green status could be features incorporated by the next practicum team.

Microsoft’s SQL Server was new to the team, but in many ways its implementation and use were similar to Oracle. Microsoft’s version is SQL is called Transact-SQL. SQL Server was used as the backend database for the VIP Parking application. To interact with SQL Server, Microsoft ASP.NET data objects where used. The site architecture followed the Model View Controller (MVC) paradigm. MVC allows for a complex project to be split into its requisite parts which lowers complexity and allows for more robust testing. In MVC, data is structured and stored in Models. Page views are rendered using Views, and Controllers are classes that aid in routing data and coordinating page renders. To allow for dynamic interaction, ASP.NET allows for full and partial page returns. Also, Razor technology allows for the imbedding of C# code snippets directly in the page HTML. This creates a seamless development environment where data, objects, functionality, and pleasing displays are all connected and organized.

The only frustration that surfaced with use of the Microsoft suite was the late discovery of the Code-First paradigm. The team started with Data-First by designing entity relationship diagrams, a DDL, then using the DDL to generate a database in our project using scaffolding. A Code-First approach structures data in classes using properties. The classes can then be bundled into a migration package that can be used to simultaneously update, synchronize, and document database changes. Code-First is an elegant approach to database design, and will be our choice for any new projects.

Overall, the practicum experience was very beneficial to our development as data professionals. Important lessons from previous classes were implemented in various ways. Examples include how to structure transactional databases using normalization, how to engineer software using UML diagrams, and how to structure an application using object oriented design and the MVC architecture. The understanding of software engineering and database technologies was greatly broadened by the practicum experience for all team members.

# References

ADA.gov. (2016). *Information and Technical Assistance on the Americans with Disabilities Act*. Retrieved from ADA.gov: https://www.ada.gov/

Microsoft. (2016). *Learn About ASP.NET MVC*. Retrieved from ASP.NET: <https://www.asp.net/mvc>

Stephens, R. (2015). Predictive models. In *Beginning software engineering (pp. 270)*. Indianapolis, IN: John Wiley & Sons.