

National College of Ireland
BSc in Computing
2015/2016

Stephen Doolan
X12304221
x12304221@student.ncirl.ie

AcademyRS - <https://gentle-lake-22126.herokuapp.com>

Technical Report



Table of Contents

1	<i>Executive Summary</i>	4
2	<i>Introduction</i>	5
2.1	Background	5
2.2	Aims	5
2.3	Technologies	6
2.4	Structure	7
2.5	System	7
2.6	Use Case Diagram	8
2.7	Requirement 1 <Registration>	9
2.8	Requirement 2 <Login>	11
2.9	Requirement 3 <Reporting issues>	13
2.10	Requirement 4 <Storing Resolved Issues>	15
2.11	Requirement 5 <Generating KPI Charts>	17
2.12	Requirement 6 <View KPI Charts>	19
2.13	Requirement 7 <System Maintenance>	21
2.14	Non-Functional Requirements	23
3	<i>Design and Architecture</i>	25
3.1	System Architecture	25
3.2	UML	26
3.3	Implementation	29
4	<i>Testing</i>	40
5	<i>Graphical User Interface (GUI) Layout</i>	45
6	<i>Customer testing</i>	50

7	<i>Evaluation</i>	51
8	<i>Conclusions</i>	52
9	<i>Further development or research</i>	53
10	<i>References</i>	54
11	<i>Appendix</i>	55
11.1	Project Proposal	55
11.2	Monthly Journals	85

1 Executive Summary

The main objective of AcademyRS project is to create an online IT Help Desk for users of The Academy LMS (Learning Management System) that will make it simpler to report issues they are encountering with the system while also providing detailed information to the support team regarding the issue which can be used to identify a resolution. AcademyRS will keep records of issue details and IT staff members responsible for solving each issue. These records will then be used to create automated key performance indicator (KPI) charts. The objectives of KPI's are to:

- Improve efficiency within the IT support team.
- Allow staff to be recognised for their contributions to the organisation.
- Provide the organisation leaders with information so they can make educated decisions.

This program aims to create an environment that is user friendly and allows them to communicate their issues effectively with our support team thus leading to a speedy resolution and resulting in greater customer satisfaction.

2 Introduction

2.1 Background

As part of my work experience module in 3rd year I spent six months working in the IT department of Olive, a health and safety company that bring their courses to an online platform with a learning management system. My duties during my time there were dealing with customers issues and conducting reports regarding IT team performance. Customers could communicate with the IT support team via email, phone or intercom (live chat on the LMS). Customers would contact me through one of those mediums and state they are encountering issues on the system; I would then have to ask them a series of technical questions in order to find the root to the problem which some customers can get frustrated with. By providing users with an easy-to-use GUI I believe we can utilise a superior and more resourceful method of gaining information from them that will improve IT team's productivity and customer satisfaction.

Olive required that every issue was documented on a live spread sheet for future reference, they would state various things like, users name, issue type, issue, IT member resolving the issue and whether it was solved or not. Every Friday I would use this information to conduct a report on IT member's performance for the previous seven days. These Key Performance Indicators were then shared within the company. This task was very time consuming and I believe that I can create a system that will log information regarding issues solved and present them in a graph or chart that will keep company leaders informed on team performance.

2.2 Aims

AcademyRS will provide users with an easy-to-use GUI for reporting their issues to our support staff. This GUI will be carefully designed to gain all the necessary information our support team requires to resolve the issue promptly thus resulting in customer satisfaction. Through KPI charts we will provide managers with detailed information regarding the support teams performance which will allow staff members to be recognised for their contributions to the organisation and encourage friendly competition resulting in improved productivity.

2.3 Technologies

The approach I will be utilising in order to complete this project is an agile software development. Agile software development is a group of software development methods in which solutions evolve through collaboration between self-organizing, cross-functional teams. It promotes adaptive planning, evolutionary development, early delivery, continuous improvement, and encourages rapid and flexible response to change. This approach is an alternative to waterfall, or traditional sequential development.

I feel this approach will benefit my project as there will be a continuous development of useful software where as in the waterfall method there is no working software produced until late during the life cycle. I will be able to adapt to change to circumstances and even late changes in requirements are welcomed unlike in the waterfall method. In this method, once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.

In order to help me adapt the Agile approach I will be using Taiga which is an open source and free agile developer management tool that has been designed with students and others in mind who are adopting a tool for the first time.

I will be using Git as a version control system. Contrasting to most client-server systems, every Git working directory is a comprehensive repository with a complete history of files which permits full version-tracking, allowing you to revert to older versions of files. I will be using Github to host my repository.

To build my project I will using Ruby on Rails which is an open source web application framework, it is a full-stack framework. Ruby on Rails uses a well-known software engineering patterns and principles such as active record pattern, convention over configuration (COC), don't repeat yourself (DRY) and model-view-controller (MVC). It encourages and facilitates the use of web standards such as JSON or XML for data transfer, and HTML, CSS and JavaScript for display and user interfacing.

I plan to utilise MySQL to create a database, which is an open-source relational database management system that many business around the world use such as Facebook, Paypal and Github.

I will be using the Platform as a Service provider Heroku to host my web application as it is the platform I am most familiar with having used it for previous projects.

2.4 Structure

System - defines the project requirements, Design, the engineering of the software involved, testing plans, GUI design, customer testing and evaluation.

Conclusions - details the result of the project and a summary of what I learned during the process of the project and the current status of the system.

Further Development or research - relates to where I feel the future route of system development is, like developing a mobile app.

References - is a section where I list all my resources of learning, mainly separate to the college course that I required to develop the application.

Appendix - is used for all other information.

2.5 System

Requirements

User Requirements Definition

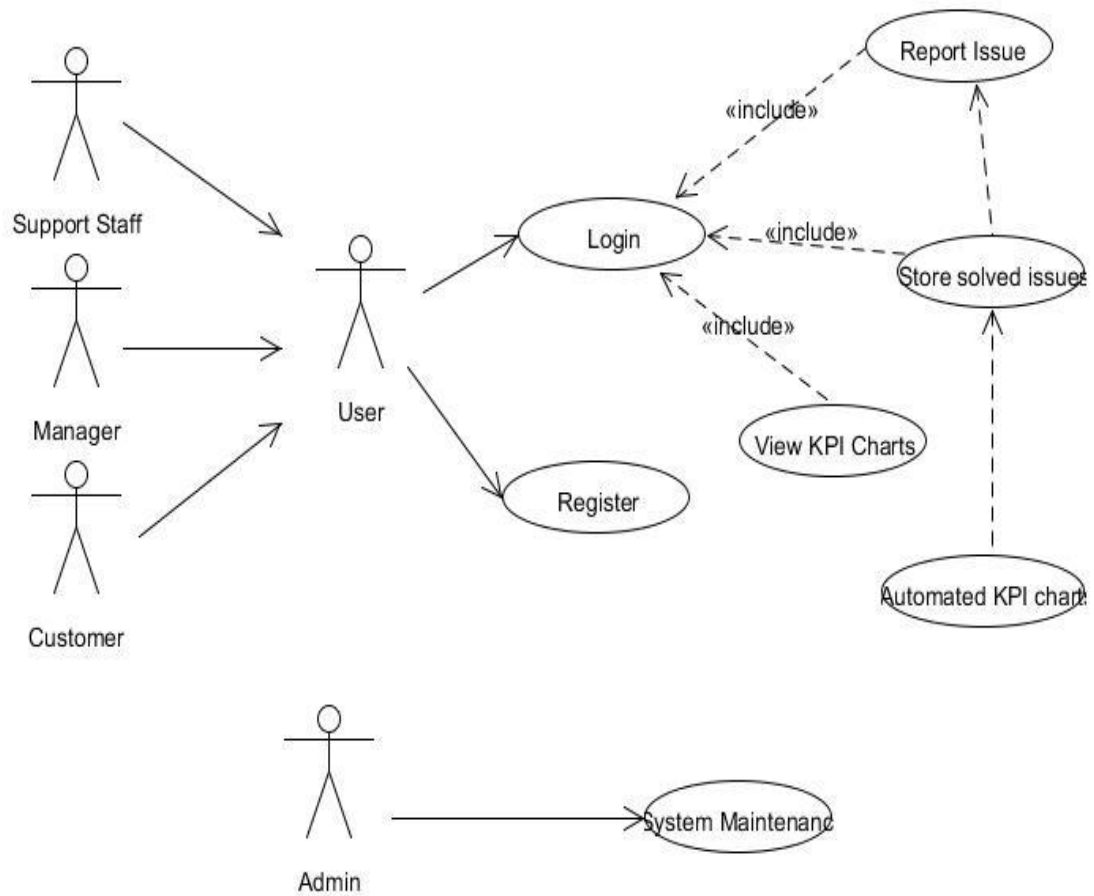
- The user must be a manager, support team member or customer of The Academy.
- The user must have a valid email address.

Functional requirements

- 1) Register
- 2) Login
- 3) Issue Reporting
- 4) Store solved issues
- 5) Automated KPI's
- 6) View KPI's
- 7) System Maintenance

2.6 Use Case Diagram

The Use Case Diagram provides an overview of all functional requirements.



2.7 Requirement 1 <Registration>

Description:

This is a webpage on The Academy RS website that will allow users to sign up to gain access to our service. Users will be required to provide us with profile information. We will use a HTML form to submit users profile information, which will then be used to create a new user and save its attributes to our database.

Priority:

High

Technical issues:

Users must fill in key criteria in order to complete the registration.

Use Case:

Register

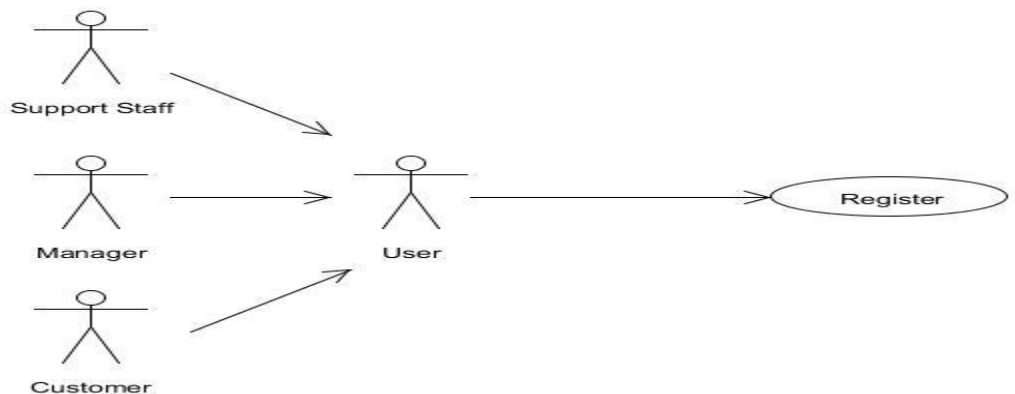
Scope:

The scope of this use case is a registration system for the user.

Description:

This use case shows the user registering to use The Academy RS service.

Use Case Diagram:



Flow Description

Precondition

None

Activation

This use case starts when a user want to register to use The Academy RS service.

Main flow

1. User opens The Academy RS home page.
2. User clicks "Register" from home page.
3. The actor fills out the form [A1 Form Validation]
4. The system stores the information provided by the user in the database.

Alternate flow

- A1 - Insufficient Form Data Provided
 - If in the Main Flow the user enters invalid profile information or does not fill in the required fields, the system displays an error message. The actor can then choose to rectify the errors or cancel the registration, at which point the use case ends.

Exceptional flow

None

Termination

The system stores all the registration data in the Academy RS database as a new record.

Post condition

If the use case was successful, the actor is now registered as a member in the system. If use case was unsuccessful, system is unchanged.

2.8 Requirement 2 <Login>

Description:

Allows a user to securely login to the Academy RS service.

Priority:

High

Technical Issues:

Users will be prompted to login using email/password combination.

Dependencies:

Requires user to have existing account with Academy RS

Use Case:

Login

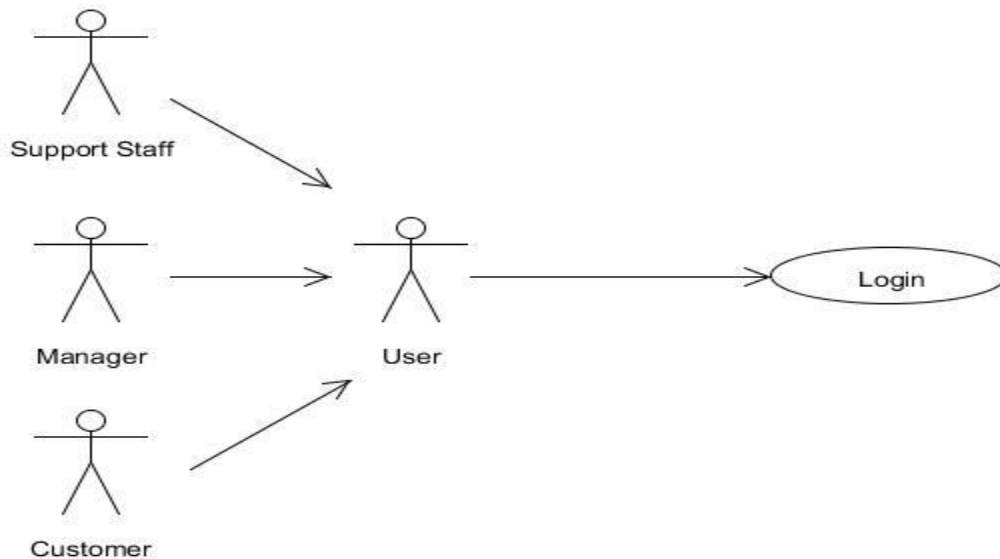
Scope:

The scope of this use case is a login system for the user.

Description

This use case describes how the user logs in to the Academy RS website.

Use Case Diagram



Flow Description

Precondition

The user is a registered member on the Academy RS system.

Activation

This use case starts when an actor wishes to log in to the Academy RS website.

Main flow

1. The system requests that the actor enters their email and password
2. The actor enters their email and password. [A1 invalid email/password]
3. The system validates the provided email and password and completes the login function allowing the actor access to our service.

Alternate flow

- A1-Invalid Email/Password
 - If in the main flow the actor enters an invalid email and/or password, the system will display an error message. The actor can choose to return to the start of the main flow or cancel the login, at which point the use case ends.

Termination

The system stores all login information.

Post condition

If the use case was successful, the actor is now logged into Academy RS. If unsuccessful, the system is unchanged.

2.9 Requirement 3 <Reporting issues>

Description:

This use case describes the user reporting an issue.

Priority:

High

Use Case:

Reporting issue

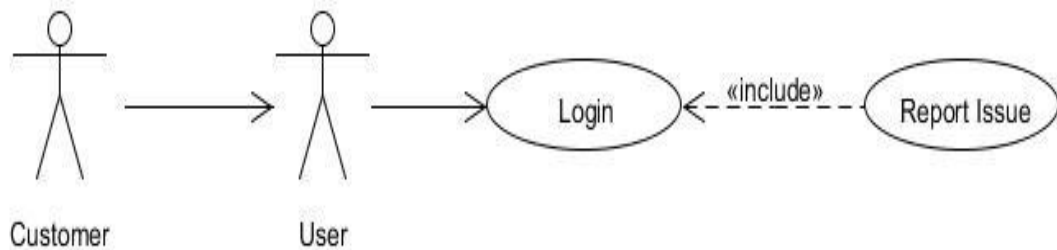
Scope:

The scope of this use case is to describe the process of how a user logs an issue report to the system.

Description

This use case describes the process of how a user logs an issue report to the system so it can be reviewed and resolved by support staff.

Use Case Diagram



Flow Description

Precondition

The system is in a ready state mode.

Activation

This use case starts when a user logs in to the Academy RS system.

Main flow

1. The user will be directed to the register page
2. The user enters their full name
3. The user enters their email address
4. The user submits a password for their account
5. **[A1 Skip Registration]**
6. The user logs into the system
7. The system loads and displays the user information
8. The user logs issue report into the system
9. The system saves the data into a database

Post condition

Issue is reported and stored in database.

2.10 Requirement 4 <Storing Resolved Issues>

Description:

This use case describes the support staff submitting solved issues to the database.

Priority:

High

Dependencies:

User must report an issue.

Support staff must resolve the issue.

Use Case:

Storing solved issues

Actors:

Support Staff

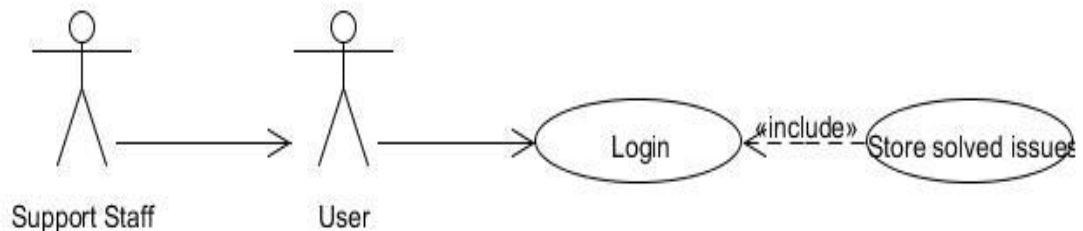
Scope:

The scope of this use case is to describe the process of storing solved issues.

Description

This use case describes support staff submitting details of solved issues to the Academy RS database so that it can be used to create KPI charts.

Use Case Diagram



Flow Description

Precondition

The system must be on

Activation

This use case starts when support staff has logged on to the system.

Main flow

1. The actor logs into the system. [A1 Invalid Username/Password]
2. The actor views reported issue.
3. The actor solves user issue.
4. The actor submits solved issue details to the database.

Alternate flow

- A1-Invalid Email/Password
 - If in the main flow the actor enters an invalid email and/or password, the system will display an error message. The actor can choose to return to the start of the main flow or cancel the login, at which point the use case ends.

Post Condition

Solved issues are stored in the database.

2.11 Requirement 5 <Generating KPI Charts>

Description:

This use case describes the system generating automated KPI charts using the solved issues stored in the database.

Priority:

High

Dependencies:

User must report an issue.

Support staff store solved issues.

Use Case:

Automated KPI Charts

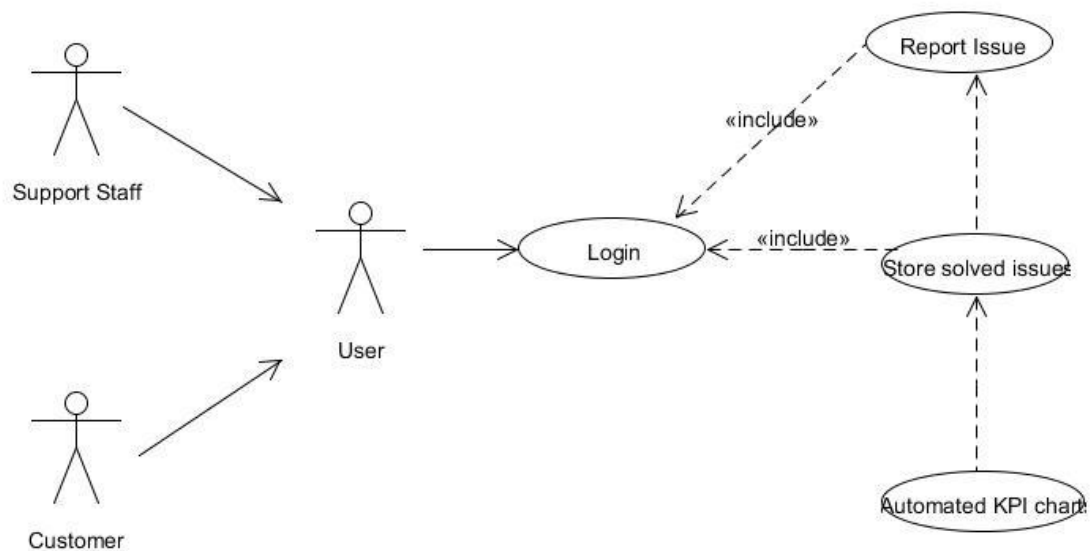
Scope:

The scope of this use case is to describe the process of automating KPI Charts.

Description

This use case describes the system gathering solved issues from the database and displaying them in a chart format.

Use Case Diagram



Flow Description

Precondition

The system must be on

Activation

This use case starts when support staff has submitted a solved issue.

Main flow

1. The customer logs into the system. [A1 Invalid Username/Password]
2. The customer reports issue.
3. System stores reported issue in database.
4. The support staff views reported issue.
5. The support team member solves the users' issue.
6. The support team member submits solved issue details to the database.
7. System pull solved issue details from database and displays them in KPI chart.

Alternate flow

- A1-Invalid Email/Password
 - If in the main flow the actor enters an invalid email and/or password, the system will display an error message. The actor can choose to return to the start of the main flow or cancel the login, at which point the use case ends.

Post Condition

KPI charts can be displayed to user.

2.12 Requirement 6 <View KPI Charts>

Description:

This use case describes the Manager viewing KPI charts.

Priority:

High

Use Case:

Viewing KPI's

Actors:

Manager

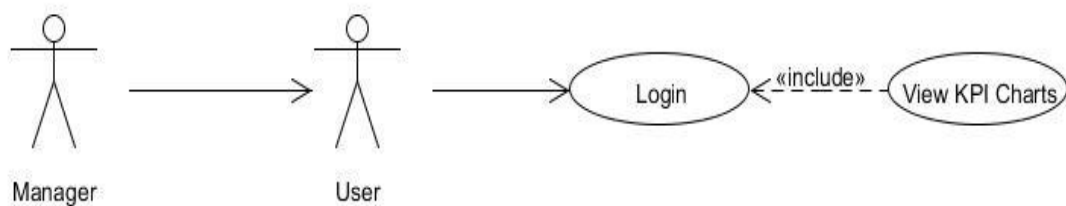
Scope:

The scope of this use case is to describe the process of a manager viewing KPI's.

Description

This use case describes Manager logging in and viewing automated KPI's

Use Case Diagram



Flow Description

Precondition

The system must be on

Activation

This use case starts when a manager has logged on to the system.

Main flow

1. The actor logs into the system. [A1 Invalid Username/Password]
2. The actor views KPI charts.

Alternate flow

- A1-Invalid Email/Password
 - If in the main flow the actor enters an invalid email and/or password, the system will display an error message. The actor can choose to return to the start of the main flow or cancel the login, at which point the use case ends.

2.13 Requirement 7 <System Maintenance>

Description:

This use case describes the tasks that are carried out in order to optimize the systems performance and security checks.

Priority:

High

Use Case:

System Maintenance

Actors:

Admin

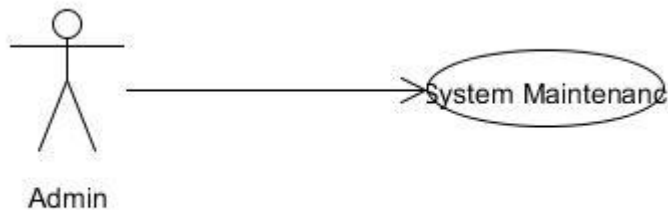
Scope:

The scope of this use case is to describe the process of maintaining the Academy RS system.

Description

This use case describes the admin operations on the system.

Use Case Diagram



Flow Description

Precondition

The system must be on

Activation

This use case starts when a admin has logged on to the system.

Main flow

1. The actor logs into the system. [A1 Invalid Username/Password]
2. The actor enters begins maintenance/diagnostic checks and operations on the system and or database
[A2 Security management]

Alternate flow

- A1 Invalid Name/Password
 - If the actor enters incorrect Username and/or password, the system will show an error message. The actor can choose to either return to the start of the Basic Flow or abandon the login, at which point the use case ends.
- A2 Security management
 - The actor starts up the security manager application to perform any security checks required.

Exceptional flow

In the case of a system crash the system will restore to the most recent updates when rebooting

Termination

The use case ends when all the maintenance operations have been completed and the actor logs out of the system.

2.14 Non-Functional Requirements

Physical requirement

This section describes the environment where the equipment is to function and if there are any environmental restrictions, such as temperature, humidity or magnetic interference.

Requirement 1 “Cloud Server”

The system will require a server on which to operate .The server will be hosted on a free cloud service called Heroku. Heroku has a full REST API and integrates with third party services instantly. Heroku's large catalog of add-on services is constantly growing, and includes services like databases, caching, monitoring, performance management, email, transcoding, search, billing, and many more.

Requirement 2 “Client Machines”

The system will require client machines to use the website. Users can log on using a desktop/laptop.

Interface requirement

My application will run using a Graphical User Interface (GUI). The operations will be carried out using keyboard and mouse.

My application will have a different GUI for each type of user.

User & Human requirement

• Who will use the system?

Managers, Support, and Customers will be using the system

• Will there be several types of users?

Managers, Support, and Customers will be have different accounts

• What is the skill level of each type of user?

There will be a minimal skill level of each type of user, except for the admin who may require further training.

• How easy will it be for a user to understand and use the system?

There written tutorial for customers on how to use the system.

Security requirement

Users will need a username and a password to log in to Academy RS.

The system will be backed up regularly and the backed up files will be stored on a local server.

Maintainability requirement

The system shall provide the capability to back-up the Data such as the database.

The database shall keep a log of all the errors, so the admin can fix them as soon as they can.

Resource Utilization requirement

What materials, personnel, or other resources are required to build, use and maintain the system?

The system needs an admin to maintain and manage the system.

What skills must the developer have?

The developer should have knowledge of Ruby on Rails, HTML, CSS, JAVASCRIPT and MySQL.

How much physical space will be taken up by the system?

The system does not require physical space as it is on the cloud.

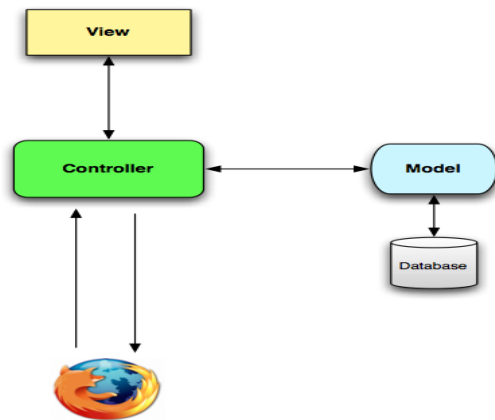
Availability requirement

The system must be available 24/7 as the academy LMS is used globally.

3 Design and Architecture

3.1 System Architecture

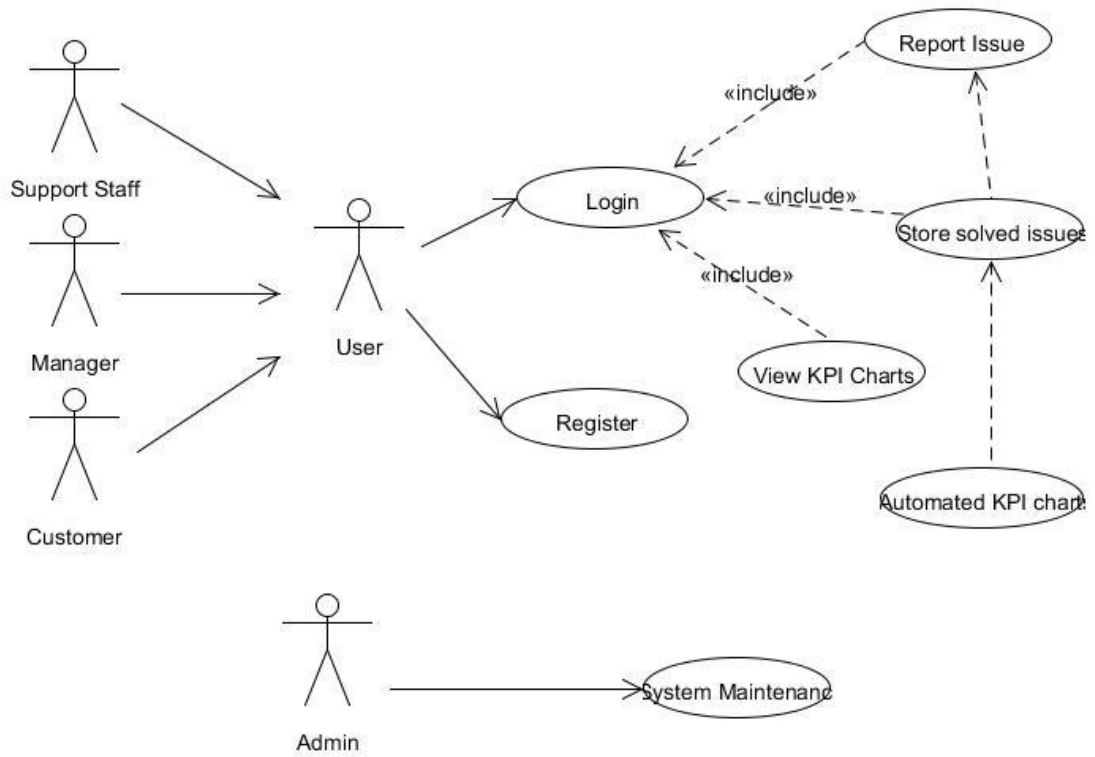
Here is a high level overview of Academy RS.MY application will use a Model-View-Controller architecture. MVC separates the “domain logic” from the input and presentation logic associated with a GUI.



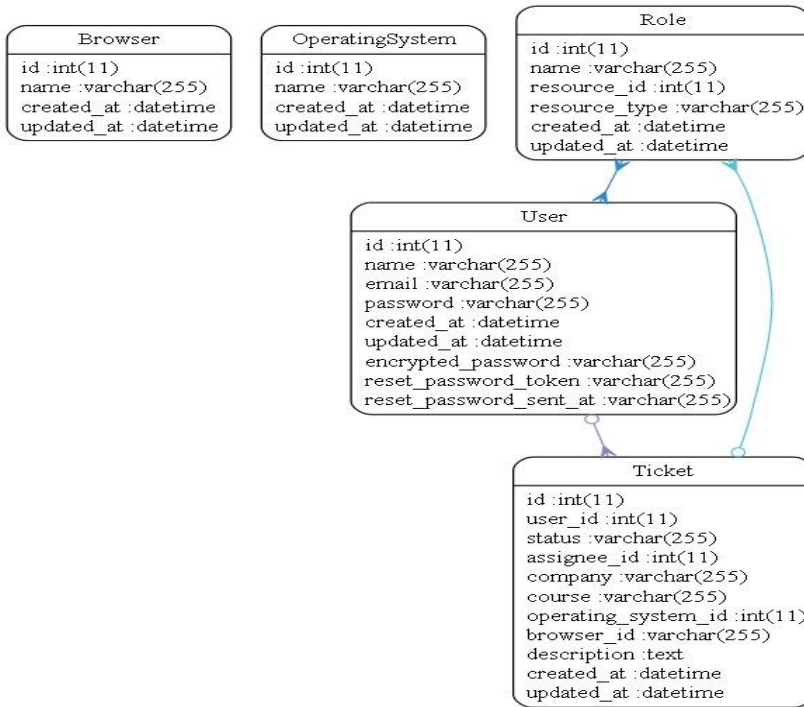
When interacting with a Rails application, a browser sends a request, which is received by a web server and passed on to a Rails controller, the controller interacts with a model, which is a Ruby object that represents an element of the site and is in charge of communicating with the database.

3.2 UML

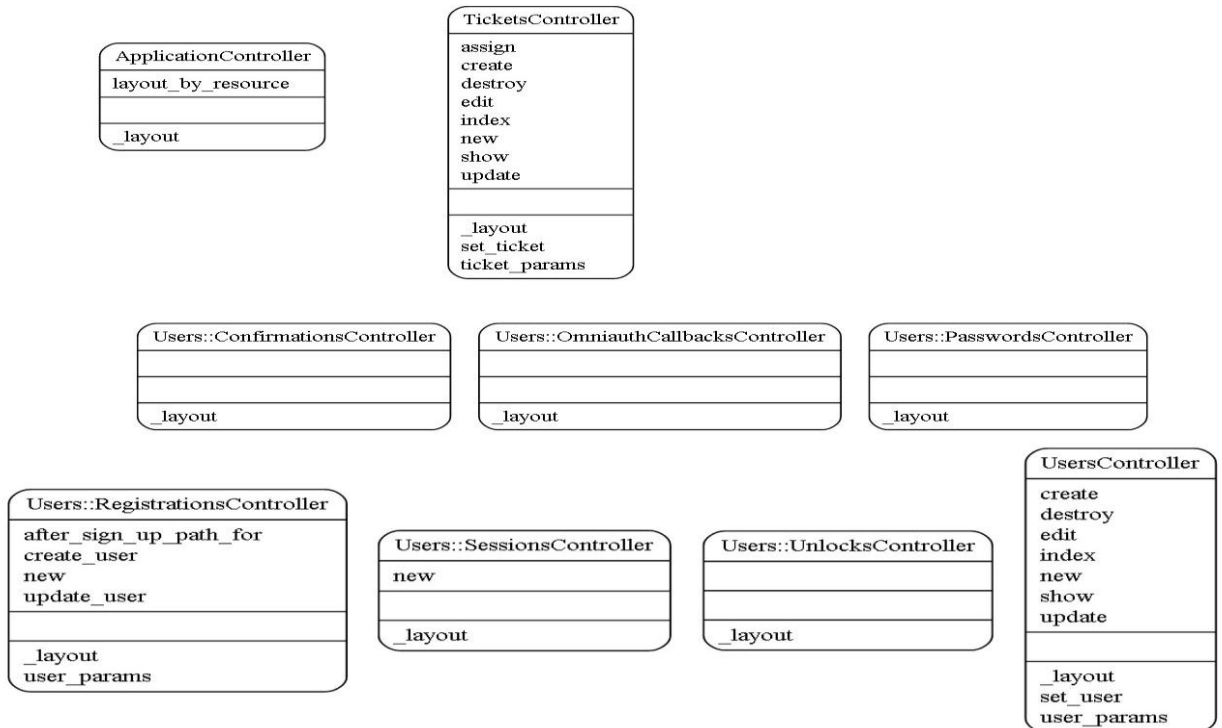
System Use Case Diagram



Model Class Diagram

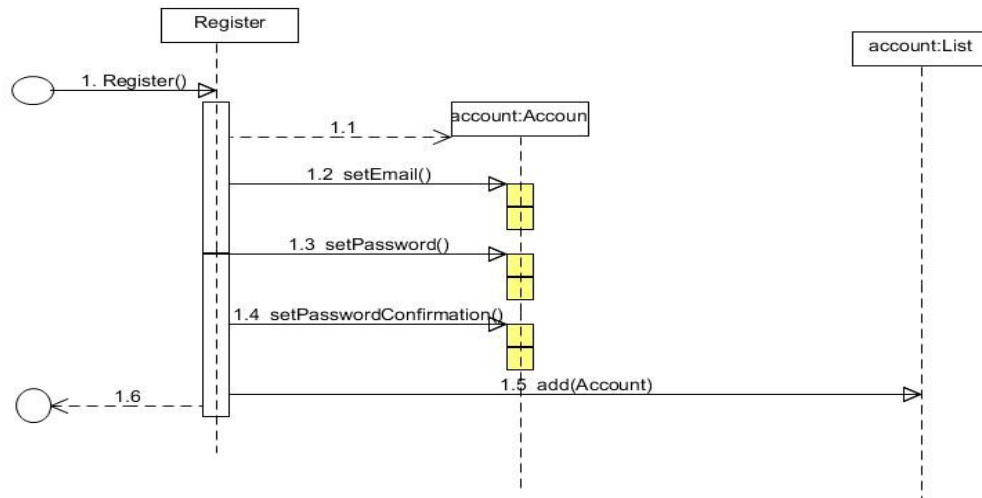


Controller Class Diagram

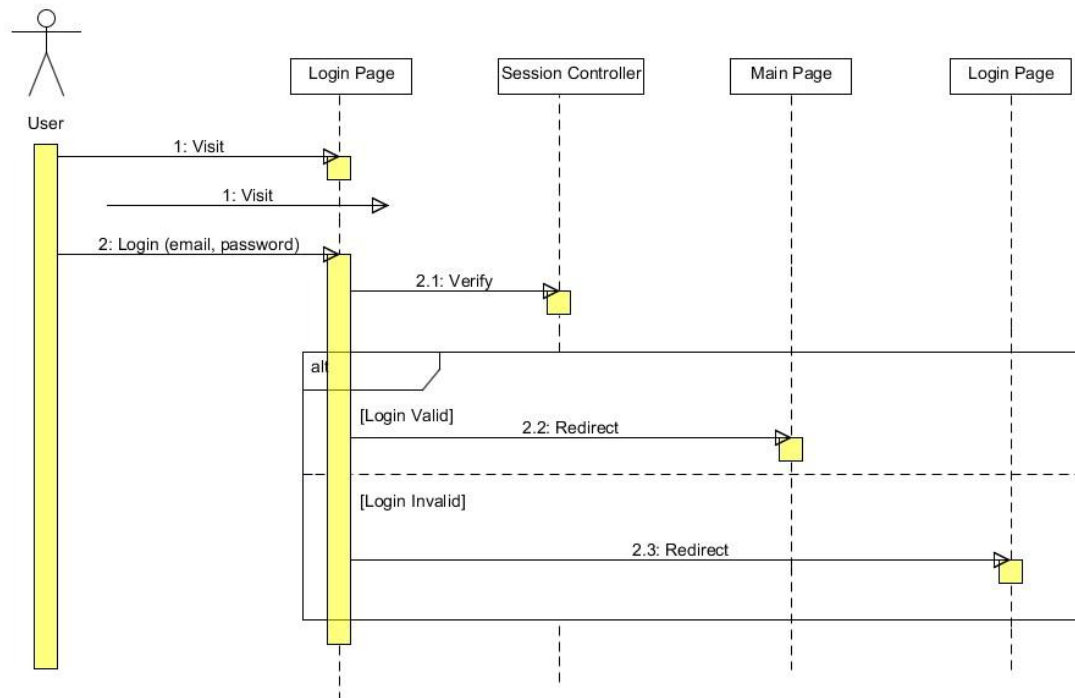


Sequence Diagram

Register User



Login User



3.3 Implementation

The reason for this section is to elaborate on the technologies utilised during the implementation of the AcademyRS application while also covering key methods used.

Technologies

Ruby

Ruby is an open source object-oriented programming language. It was created in 1993 by Yukihiro Matsumoto of Japan. The Ruby language is a combination of popular languages such as Smalltalk, Perl and Python. Yukihiro Matsumoto's goal was to create a language that was more powerful than Perl and more object-orientated than Python. Ruby allows simple and fast creation of web applications.

Rails

Rails is an full stack framework that is perfect for creating database-driven web applications, using the Model-View-controller pattern. Being a full stack framework means all layers are built to work seamlessly together. Rails requires less lines of code than the majority of other frameworks. Rails was created in Ruby by David Heinemeier Hanson.

Rails strength:

- Metaprogramming
- Active Record
- Convention over configuration
- Scaffolding
- Built-in testing

Git

It is paramount to a project that you utilise a version control in order to keep records of changes to the project and have the ability to rollback if needed. I have decided that Git would be perfectly suitable to use as it is very popular in the development world. I will be using Github to host my repository.

Configuration

I had to tell Git my *name* so your commits will be properly labeled

```
$ git config --global user.name "Stephen Doolan"
```

Secondly, you must enter your email address so it is associated with your Git commits.

```
$ git config --global user.email doolan12@hotmail.com
```

Initialising local repository

Once in the root directory do the following:

```
$ git init
```

Adding and Committing

I used the below commands to add files to my AcademyRS app and then committed them.

```
$ git add .  
$ git commit -m "initial commit"
```

Pushing code to Github

```
$ git remote add origin https://github.com/doolan12/The-Academy-RS.git  
$ git push origin master
```

Procedures

Project creation

I will go through the creation process step-by-step.

1. Creating the new application

```
rails new AcademyRS
```

2. Changing directory

```
cd AcademyRS
```

3. Installing Gems

```
bundle install
```

4. Creating the database

```
rake db:create
```

5. Generating models, controllers and migrations.

```
rails g scaffold User name:string email:string password:string  
rails g model OperatingSystem name:string  
rails g model Browser name:string  
rails g scaffold Ticket user_id:integer status:string assignee_id:integer company:string  
course:string operating_system_id:integer browser_id:integer description:text
```

User Authorisation, Authentication and Roles

- 1) Adding gems to Gemfile.

```
#authentication  
gem 'devise'  
#roles  
gem 'rolify'  
#authorization  
gem 'cancancan', '~> 1.10'
```

- 2) Installing gems

```
bundle install
```

3) Run Devise generator

```
rails generate devise:install
```

4) Create the User model from Devise

```
rails generate devise User
```

5) Create the Ability class from CanCanCan

```
rails generate cancan:ability
```

6) Create the Role class from rolify

```
rails generate rolify Role User
```

7) Run migrations

```
rake db:migrate
```

8) Configure ability.rb

```
if user.has_role? :manager
  #can :manage , :all
  can :manage , User
  can :manage , Ticket
elsif user.has_role? :support_staff
  can :manage , Ticket
  can :manage , User
elsif user.has_role? :customer
  can :create , Ticket
  can :read , Ticket
  can :update , Ticket
end
```

9) Use the resourcify method in the ticket model

```
class Ticket < ActiveRecord::Base
  resourcify
end
```

10) Adding roles in rails console

```
user.add_role "manager"
user.add_role "support_staff"
user.add_role "customer"
```


11) Assigning Roles

```
<div class="field">
  <%= f.input :role , collection: [ ["manager" , "manager" ] , ["customer","customer" ] ,
  ["support_staff" , "support_staff" ] ] %>
</div>
```

Ticket Assigning and status updating

Configuring index.html.erb within the Views/Tickets folder

Ticket Status

```
<td>
  <% if can? :manage, Ticket %>
    <%= simple_form_for(ticket) do |f| %>
      <%= f.input :status , :collection => [ ["Resolved" , "Resolved" ] , ["Unresolved" ,
  "Unresolved" ] ] , label: false %>
      <%= f.submit %>
    <% end %>
  <% else %>
    <%= ticket.status %>
  <% end %>
</td>
```

Assigning Ticket to support staff only

```
<% if can? :manage , Ticket %>

  <%= simple_form_for(ticket , :url => assign_ticket_path(ticket) , :method => :post) do |f|
%>
  <%= f.input :assignee_id , :collection => User.with_all_roles(:support_staff) , label: false
%>
  <%= f.submit %>
<% end %>
  <%else %>
  <%= ticket.assignee.try(:name) %>
<% end %>
</td>
```

Datatables and Date Picker

1. Adding Gems

```
gem 'jquery-datatables-rails', '~> 3.3.0'  
gem 'bootstrap-datepicker-rails'
```

2. Installing Gems

```
bundle install
```

3. Run the install generator:

```
rails generate jquery:datatables:install
```

4. Configure application.js and application.css.scss files

Application.css.scss:

```
*= require dataTables/jquery.dataTables  
*= require bootstrap-datepicker
```

Application.js:

```
//= require dataTables/jquery.dataTables  
//= require bootstrap-datepicker
```

5. Initialise datatable and date picker

Datatable:

```
<script type="text/javascript">  
$(document).ready(function () {  
  $('table').dataTable({  
    destroy: true  
  });  
});  
</script>
```

Date picker:

```
<script type="text/javascript">
  $(document).ready(function () {
    $('.date_picker').datepicker({
      format: 'dd/mm/yyyy'
    });
    $('.date_picker').datepicker({dateFormat: 'mm-dd-yyyy'});
    $('.date_picker').datepicker({dateFormat: 'mm-dd-yyyy'});
  });
</script>
```

Key Performance Indicator Charts

1. Adding Gem:

```
gem "highcharts-rails"
```

2. Installing Gem:

```
bundle install
```

3. Configuring application.js file

```
//= require highcharts
//= require highcharts/highcharts-more
```

4. Implementing charts method in tickets_controller

```
def charts
  @chart_data = []
  @ticket_type_data = []
  tickets_count = Ticket.all.count
  ["General", "Payment", "Technical", "Others"].each do |p|
    @ticket_type_data << { :name => p , :y => ("%2f" % (Ticket.where(:ticket_type => p).count.to_f/tickets_count.to_f)).to_f , :tickets => Ticket.where(:ticket_type => p).count }
  end
  User.with_role(:support_staff).each do |user|
    @chart_data << { :y => Ticket.where(:status => "Resolved" , :assignee_id => user.id).count , :name => user.name }
  end
  @support_staff = User.with_role(:support_staff).pluck(:name)
  @filter = Ticket.new( :to_date => Time.now.strftime("%m/%d/%Y") )
end
```

5. Create charts.html.erb file

```
<style>
  .highcharts-tooltip span {
    width: 200px !important;
    overflow: auto;
    white-space: normal !important;
  }
</style>

<script type="text/javascript">
  $(document).ready(function () {
    $('#datepicker').datepicker({
      format: 'dd/mm/yyyy'
    });
  });
</script>

<script>
  $(function () {

    $(document).ready(function () {

      var chart_json = JSON.parse('<%= raw @chart_data.to_json %>');
      var staff_json = JSON.parse('<%= raw @support_staff.to_json %>');

      // Build the chart
      $('#tickets_chart_container').highcharts({
        chart: {
          type: 'column',
          zoomType: 'x'
        },
        title: {
          text: 'Issues'
        },
        credits: {
          enabled: false
        },
        tooltip: {
          useHTML: true,
          formatter: function () {
            return '<b>' + this.point.name + ': ' + this.point.y + '</b><br><span style="font-size: 10px;"></span>';
          }
        },
        plotOptions: {
          column: {
            pointPadding: 0.2,
            borderWidth: 0
          }
        },
        xAxis: {
          type: 'Staff Members',
          categories: staff_json,
          labels: {
            rotation: -90,
            style: {
              fontSize: '10px',
              fontFamily: 'Verdana, sans-serif'
            }
          }
        }
      });
    });
  });
</script>
```

```

yAxis: {
  min: 0,
  title: {
    text: 'Number of Issues'
  }
},
series: [{
  name: 'Issues',
  colorByPoint: true,
  data: chart_json
}]
});

var chart_json = JSON.parse('<%= raw @ticket_type_data.to_json %>');
console.log(chart_json);
$('#tickets_type_chart_container').highcharts({
  chart: {
    plotBackgroundColor: null,
    plotBorderWidth: null,
    plotShadow: false,
    backgroundColor: 'transparent',
    type: 'pie'
  },
  title: {
    text: 'Tickets per ticket type',
    style: {color: '#ffffff'}
  },
  tooltip: {
    useHTML: true,
    formatter: function() {
      return '<b>'+ this.point.name + ' : '+ this.point.y * 100 + '%</b><br><span style="font-size: 10px;">'+
this.point.tickets + ' tickets</span>';
    }
  },
  plotOptions: {
    pie: {
      allowPointSelect: true,
      cursor: 'pointer',
      dataLabels: {
        enabled: true,
        format: '<b>{point.name}</b>: {point.percentage:.1f} %',
        style: {
          color: (Highcharts.theme && Highcharts.theme.contrastTextColor) || '#666'
        }
      }
    }
  },
  series: [{
    name: 'Issue Type',
    colorByPoint: true,
    data: chart_json
  }]
});
});
</script>

```

```
<div class="row">
  <div class="col-xs-offset-2 col-xs-3">
    <%= simple_form_for(@filter, :url => filter_charts_tickets_path, :method => :post) do |f| %>
      <%= f.input :from_date, :input_html => {:class => "datepicker"} %><br>
      <%= f.input :to_date, :input_html => {:class => "datepicker"} %><br>
      <div class="text-center">
        <%= f.submit "Filter" , :class => "btn btn-primary" %><br>

        <% end %>
      </div>
    </div>
  </div>
</div>

<div id="tickets_chart_container" style="min-width: 310px; height: 400px; margin: 0 auto"> </div>
<div id="tickets_type_chart_container" style="min-width: 310px; height: 400px; margin: 0 auto"> </div>

</div>
```

Browser and Operating System Detection

1. Adding Gem

```
gem 'browser'
```

2. Installing Gem

```
bundle install
```

3. Adding to New method in Tickets Controller

```
require "browser"

if browser.firefox?
  browser_id = "Mozilla Firefox"
elsif browser.chrome?
  browser_id = "Google Chrome"
elsif browser.ie?
  browser_id = "Internet Explorer"
elsif browser.safari?
  browser_id = "Safari"
end

if browser.platform.linux?
  platform = OperatingSystem.find_by_name("Linux")
elsif browser.platform.windows?
  platform = OperatingSystem.find_by_name("Windows")
elsif browser.platform.mac?
  platform = OperatingSystem.find_by_name("Mac")
end
browser.platform.mac?
@ticket = Ticket.new(:browser_id => browser_id, :operating_system => platform)
@support_staff = User.with_role(:support_staff )
end
```

4 Testing

Throughout the process of this project I carried out a lot of Black box testing to ensure that functionality of the project worked as expected by the end-user.

Black Box Testing:

Is a method of software **testing** that examines the functionality of an application without peering into its internal structures or workings.

User Sign up - successful:

Expected outcome: user fills in required fields correctly and submits them. Once submitted the user will be redirected to the tickets index page.

Actual outcome: User filled in required fields and submitted. The user was then redirected to the tickets index page.

User Sign up - unsuccessful:

Expected outcome: user fills in required fields incorrectly and submits them. Once submitted the user will be prompted to correctly fill in the required fields.

Actual outcome: User is prompted to correctly fill in the the required fields.

User Login - successful:

Expected outcome: User enters their details that are matched with the details they provided when signing up and are allowed to access the application.

Actual outcome: User is allowed to access the application and is redirected to the tickets index page.

User Login - unsuccessful:

Expected outcome: user provides details that do not match any within the database and is not allowed access the application.

Actual outcome: User is required to try again.

User Forgotten Password:

Expected outcome: User enters the email address they signed up with and system will send instructions on how to set new password.

```
Actual outcome: application error.  
"Missing host to link to! Please provide the :host parameter, set  
default_url_options[:host], or set :only_path to true"
```

Customer interface :

Expected outcome: user with the role customer will be directed to the tickets index page once successfully logged in. they will only be able to see issues they have reported previously or report a new issue.

Actual outcome: User is only able to see issues they have previously reported or create report new issue.

Customer interface :

Expected outcome: user with the role "customer" will be directed to the tickets index page once successfully logged in. they will only be able to see issues they have reported previously or report a new issue.

Actual outcome: User is only able to see issues they have previously reported or create report new issue.

Support Staff Interface :

Expected outcome: user with the role "support_staff" will be directed to the tickets index page once successfully logged in. Support staff will have the issues and users links within the navbar. They can view all issues reported.

Actual outcome: User can view all issues reported. User can navigate between issues and users page.

Manager Interface :

Expected outcome: user with the role “Manager” will be directed to the tickets index page once successfully logged in. Managers have all the capabilities of the support staff but can also view the the KPI charts.

Actual outcome: User can view all issues reported. User can navigate between issues, charts and users pages.

Report New Issue :

Expected outcome: users fills in the the issue form and submits the information. This information is then displayed on the issues page.

Actual outcome: User successfully submitted issue and it was displayed as expected.

Edit Existing Issue :

Expected outcome: user feels it is necessary to edit their issue and submits their changes. Changes are then displayed to the user.

Actual outcome: user was successfully able to edit an existing issue and changes were displayed once submitted.

Browser and operating system detection :

Expected outcome: when filling in the issue report form the application automatically detects what web browser and operating system the user is utilising.

Actual outcome: Application was able to successfully detect that I was using google chrome and windows.

Filter Issues by date :

Expected outcome: User selects a date range and application only displays issues reported between thos dates.

Actual outcome: Application was accurate when filtering issues.

Updating issue status :

Expected outcome: support staff and managers can update the status of an issue. Issue will update the status once submitted.

Actual outcome: issue status was updated correctly and displayed new status.

Assigning support staff to an issue :

Expected outcome: support staff and managers can assign support staff to particular issues. The assignee will then be displayed.

Actual outcome: assigned support staff to issue as expected.

KPI Charts:

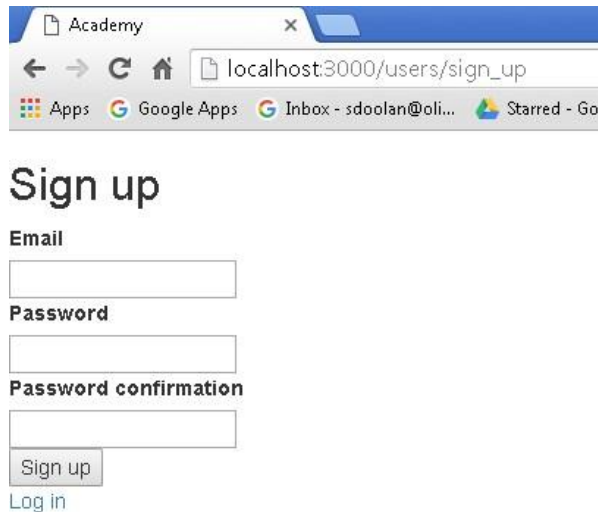
Expected outcome: chart displays the quantity of issues and who is responsible for solving them. 2nd chart displays the type of issue and the percentage in which it occurs.

Actual outcome: charts displayed accurately the number of issue and what staff was responsible for solving the issue. They also correctly displayed the type of issue and the percentage.

5 Graphical User Interface (GUI) Layout

Becoming a member

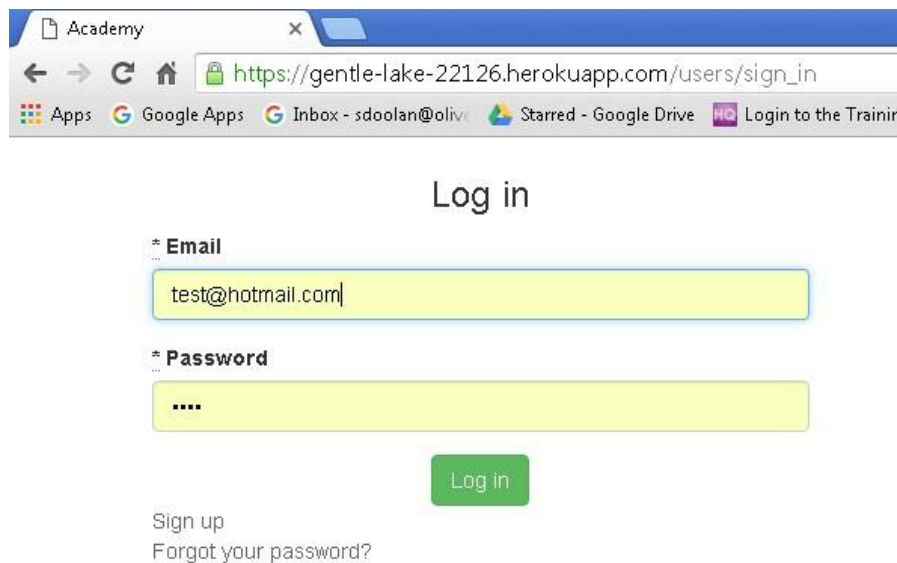
User registration – user signs up by submitting their email and password, then continues on to the login page.



The screenshot shows a web browser window with the title 'Academy' and the URL 'localhost:3000/users/sign_up'. The browser's address bar shows the URL, and the tabs below show 'Apps', 'Google Apps', 'Inbox - sdoolan@oli...', and 'Starred - Go'. The main content area displays a 'Sign up' form with the following fields and buttons:

- Email**: A text input field.
- Password**: A text input field.
- Password confirmation**: A text input field.
- Sign up**: A button.
- Log in**: A link.

User Login – Users enter the details they provided in the registration stage.



The screenshot shows a web browser window with the title 'Academy' and the URL 'https://gentle-lake-22126.herokuapp.com/users/sign_in'. The browser's address bar shows the URL, and the tabs below show 'Apps', 'Google Apps', 'Inbox - sdoolan@olivi...', 'Starred - Google Drive', and 'Login to the Trainin...'. The main content area displays a 'Log in' form with the following fields and buttons:

- * Email**: A text input field containing 'test@hotmail.com'.
- * Password**: A text input field with masked characters '....'.
- Log in**: A green button.
- Sign up**: A link.
- Forgot your password?**: A link.

Customer GUI

Customers can only view issues they have reported.

Academy Issues

Issues

From date

To date

Show entries

Search:

Description	Date	Customer	Status	Company	Course	Operating system	Web Browser	Assignee	Issue Type
course not playing	09/05/16	customer	Unresolved	BAM	BAM introduction	Windows	Google Chrome		<input type="checkbox"/>

Showing 1 to 1 of 1 entries

Previous Next

Creating new ticket – customers fills in all the required fields and submits the issue report.

New Ticket

Assigned to -

* **Company**

* **Course**

Operating system

Browser

* **Description**

Support staff and Managers capabilities.

Tickets GUI - Both these user types can create, read, edit and delete tickets. They can update the status of issues as well as the ability to assign support staff to specific issues.

Signed in successfully.✕

From date

To date

Show entriesSearch:

Description	Date	Customer	Status	Company	Course	Operating system	Web Browser	Assignee	Issue Type	
course keeps freezing	09/05/16	test	<input type="text" value="Resolv"/> <input type="button" value="Submit"/>	BAM	BAM introduction	Windows	Google Chrome	<input type="text" value="test"/> <input type="button" value="Submit"/>	General	<input type="checkbox"/>
course not playing	21/03/16	stephen doolan	<input type="text" value="Resolv"/> <input type="button" value="Submit"/>	BAM	BAM introduction		Internet Explorer	<input type="text" value="test"/> <input type="button" value="Submit"/>	Technical	<input type="checkbox"/>
course not playing	09/05/16	customer	<input type="text" value="Unresc"/> <input type="button" value="Submit"/>	BAM	BAM introduction	Windows	Google Chrome	<input type="text"/> <input type="button" value="Submit"/>		<input type="checkbox"/>

Showing 1 to 3 of 3 entriesPrevious Next

Users GUI – support staff and managers have the ability to create, edit and delete users and assign roles to each user.

Listing Users

Show **10** entries Search:

Name	Email	Role				
	testing@hotmail.com	customer				
customer	customer@hotmail.com	customer				
customer1	customer1@hotmail.com	customer				
customer2	customer2@hotmail.com	customer				
customer3	customer3@hotmail.com	customer				
customer4	customer4@hotmail.com	customer				
stephen doolan	doolan12@hotmail.com	manager				
test	test@hotmail.com	support_staff				
test1	test1@hotmail.com	support_staff				
test2	test2@hotmail.com	support_staff				

Showing 1 to 10 of 12 entries Previous **1** 2 Next

New User

Name

*** Email**

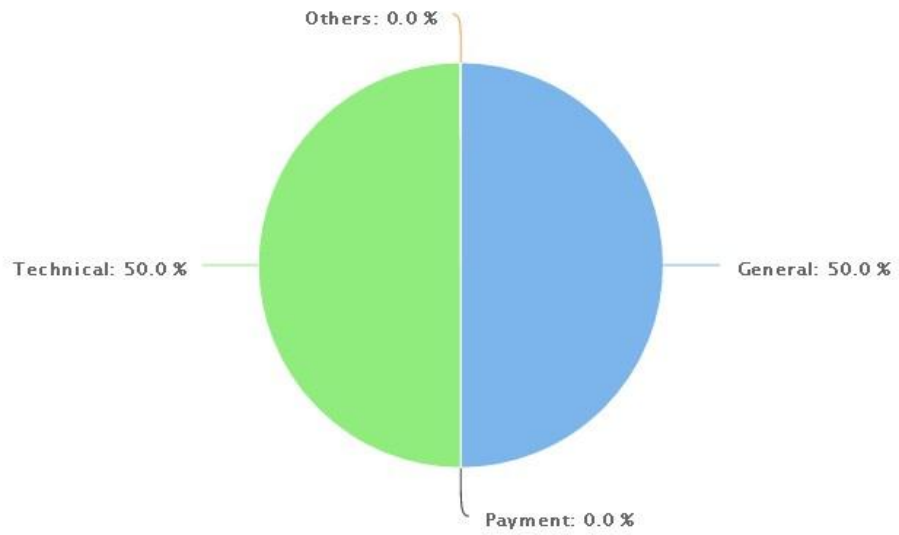
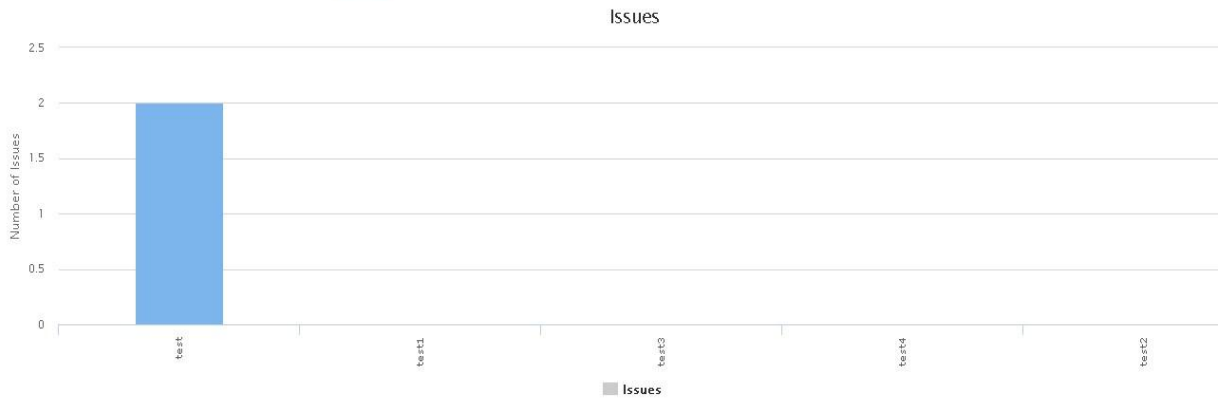
Role

*** Password**

Charts GUI

From date:

To date:



6 Customer testing

One usability concern that came to my attention during testing was that although application automatically detects the web browser and operating system the current user has, it does not make them aware that it is detected automatically which can lead to them trying to figure this out when it is unnecessary. This issue can be fixed by simply alerting the user that the application has detected their browser and operating system.

Stephen Greene, the Academy's support team manager had a look over the AcademyRS application and recommended that the key performance indicator charts should have an export function so they can be presented at the organisations meetings. I felt that this was a good observation and tried to add this functionality to the application. It was simply done by including the following code with my application.js file: `//= require highcharts/modules/exporting`

7 Evaluation

Please note that results of testing are outlined in the testing section of this document and the usability findings are detailed within the customer testing section of this report.

8 Conclusions

AcademyRS has many advantages such as the browser detection that can make it easier for people who are not tech savvy to still be able to provide the support staff with the necessary information. Another reason that AcademyRS can be advantageous to staff is that KPI charts are automatically produced rather than manually which leaves support staff with more time available to solve users issues. These KPI charts also provide the organisation with quality information that can be used to make informed decisions regarding performance.

This document outlines what motivated me to develop this project. My goals are a well-defined. The report details all that is required from the application. I discuss the technologies and procedures used to create the AcademyRS application. The designed and architecture of the application is deliberated and use case diagrams are provided to elaborate further.

9 Further development or research

I believe that many organisations would find the automated Key Performance Charts regarding their support staff very useful. Thus I believe that AcademyRS could be easily adapted to many organisations by simply redesigning the issues form so that it suits that organisation's needs.

10 References

Bibliography: RolifyCommunity (2016) *RolifyCommunity/rolify*. Available at: <https://github.com/RolifyCommunity/rolify/wiki/Devise---CanCanCan---rolify-Tutorial> (Accessed: 9 May 2016). In-line Citation: (RolifyCommunity, 2016)

Bibliography: PerfectlyNormal (2016) *PerfectlyNormal/highcharts-rails*. Available at: <https://github.com/PerfectlyNormal/highcharts-rails> (Accessed: 9 May 2016). In-line Citation: (PerfectlyNormal, 2016)

Bibliography: fnando (2016) *Fnando/browser*. Available at: <https://github.com/fnando/browser> (Accessed: 9 May 2016). In-line Citation: (fnando, 2016)

Bibliography: rweng (2016) *Rweng/jquery-datatables-rails*. Available at: <https://github.com/rweng/jquery-datatables-rails> (Accessed: 9 May 2016). In-line Citation: (rweng, 2016)

Bibliography: Nerian (2016) *Nerian/bootstrap-datepicker-rails*. Available at: <https://github.com/Nerian/bootstrap-datepicker-rails> (Accessed: 9 May 2016). In-line Citation: (Nerian, 2016)

Bibliography: Ocean, D. (2016) *#223 charts*. Available at: <http://railscasts.com/episodes/223-charts?autoplay=true> (Accessed: 9 May 2016). In-line Citation: (Ocean, 2016)

11 Appendix

11.1 Project Proposal

Project Proposal

The Academy RS

Stephen Doolan, x12304221, doolan12@hotmail.com

BSc (Hons) in Computing

Cloud Computing

21/09/15

1. Objectives

The main objective of THE ACADEMY RS project is to create an online reporting system that will make it simpler for users to communicate their problems while providing the IT support team with detailed information necessary to identify a resolution. The program will keep track of the types of problems, IT staff members responsible for solving issues and will be designed to share key findings within the organisation. This will be known as “key performance indicators”. The objective of KPI’s is to improve efficiency within the IT support team, allow staff to be recognised for their contributions to the organisation and provide the organisation leaders with information so they can make informed decisions.

2. Background

As part of my work experience module in 3rd year I spent six months working in the IT department of Olive, a health and safety company that bring their courses to an online platform with a learning management system. My duties during my time there were dealing with customers issues and conducting reports regarding IT team performance. Customers could communicate with the IT support team via email, phone or intercom (live chat on the LMS). Customers would contact me through one of those mediums and state they are encountering issues on the system; I would then have to ask them a series of technical questions in order to find the root to the problem which some customers can get frustrated with. By providing users with an easy-to-use GUI I believe we can utilise a superior and more resourceful way to gain information from them that will improve IT team’s productivity and customer satisfaction.

Olive required that every issue was documented on a live spread sheet for future reference, they would state various things like, users name, issue type, issue, IT member resolving the issue and whether it was solved or not. Every Friday I would use this information to conduct a report on IT member’s performance for the previous seven days. These Key Performance Indicators were then shared within the company. This task was very time consuming and I believe that I can

create a system that will log information regarding issues solved and present them in a graph or chart that will keep company leaders informed on team performance.

3. Technical Approach

The approach I will be utilising in order to complete this project is an agile software development. Agile software development is a group of software development methods in which solutions evolve through collaboration between self-organizing, cross-functional teams. It promotes adaptive planning, evolutionary development, early delivery, continuous improvement, and encourages rapid and flexible response to change. This approach is an alternative to waterfall, or traditional sequential development.

I feel this approach will benefit my project as there will be a continuous development of useful software where as in the waterfall method there is no working software produced until late during the life cycle. I will be able to adapt to change to circumstances and even late changes in requirements are welcomed unlike in the waterfall method. In this method, once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.

In order to help me adapt the Agile approach I will be using Taiga which is an open source and free agile developer management tool that has been designed with students and others in mind who are adopting a tool for the first time.

I will be using Git as a version control system. Contrasting to most client-server systems, every Git working directory is a comprehensive repository with a complete history of files which permits full version-tracking, allowing you to revert to older versions of files. I will be using Github to host my repository.

To build my project I will using Ruby on Rails which is an open source web application framework, it is a full-stack framework. Ruby on Rails uses a well-known software engineering patterns and principles such as active record

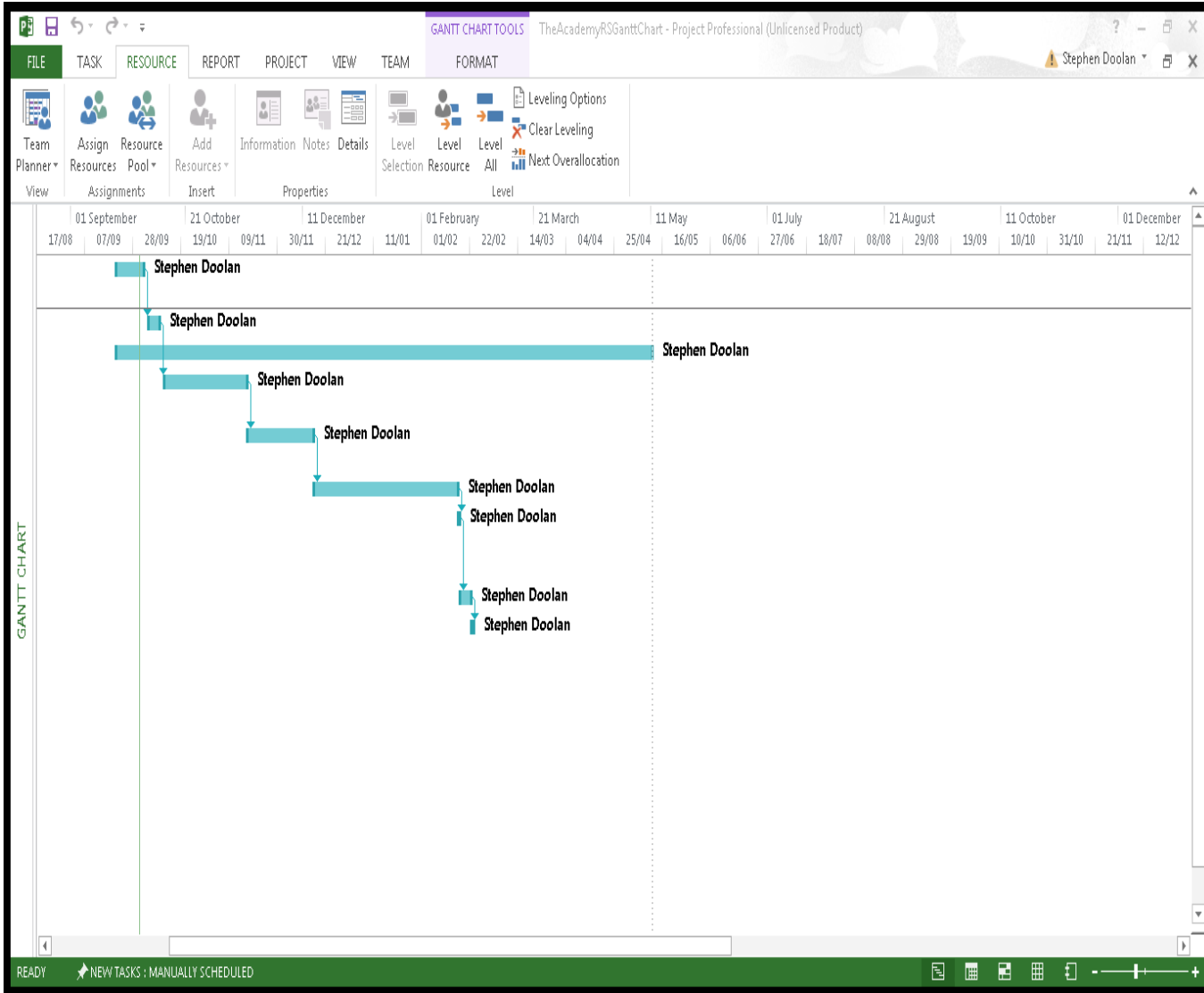
pattern, convention over configuration (COC), don't repeat yourself (DRY) and model-view-controller (MVC). It encourages and facilitates the use of web standards such as JSON or XML for data transfer, and HTML, CSS and JavaScript for display and user interfacing.

I plan to utilise MySQL to create a database, which is an open-source relational database management system that many business around the world use such as Facebook, Paypal and Github.

I will be using the Platform as a Service provider Heroku to host my web application.

4. Project Plan

Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Names
Manually Scheduled	Project Proposal Document	10 days	Mon 21/09/15	Fri 02/10/15		Stephen Doolan
Manually Scheduled	Project Plan	5 days	Mon 05/10/15	Fri 09/10/15	1	Stephen Doolan
Manually Scheduled	Reflective Journal	168 days	Mon 21/09/15	Wed 11/05/16		Stephen Doolan
Manually Scheduled	Requirements Specification	26 days	Mon 12/10/15	Mon 16/11/15	2	Stephen Doolan
Manually Scheduled	Project Analysis & Design	21 days	Tue 17/11/15	Tue 15/12/15	4	Stephen Doolan
Manually Scheduled	Prototype	45 days	Wed 16/12/15	Tue 16/02/16	5	Stephen Doolan
Manually Scheduled	Project Documentation and code	1 day	Wed 17/02/16	Wed 17/02/16	6	Stephen Doolan
Manually Scheduled	Presentation	3 days	Thu 18/02/16	Mon 22/02/16	7	Stephen Doolan
Manually Scheduled	Showcase	1 day	Tue 23/02/16	Tue 23/02/16	8	Stephen Doolan



5. Technical Details

I will be using the Ruby on Rails framework and for a package manager I will be using RubyGems which provide Ruby programs and libraries. I will be using MySQL for my database. I will be using HTML and Twitter Bootstrap for my front end design.

6. Evaluation

I intend to carry out a number of tests that will be designed to discover if the system is working as expected. Any bugs that are found will be used as a measurement to gauge improvements over the life cycle of the project. I think it will important that I get users opinions on how the system looks and performs. In order to return quantifiable results, a questionnaire will be planned to guide testers.

__Stephen Doolan _01/10/2015_____

Signature of student and date

11.2 Requirements specification

National College of Ireland

Technical Report

Stephen Doolan – x12304221
10/8/2015

Table of Contents

Contents

<u>Purpose</u>	63
<u>Project Scope</u>	63
<u>Definitions, Acronyms, and Abbreviations</u>	63
<u>User Requirements Definition</u>	7
<u>Functional requirements</u>	7
Requirement1 <Registration>	5
<u>Requirement 2 <Login></u>	11
<u>Requirement 3 <Reporting issues></u>	13
<u>Requirement 4 <Storing Resolved Issues></u>	15
<u>Requirement 5 <Generating KPI Charts></u>	17
<u>Requirement 6 <View KPI Charts></u>	19
<u>Requirement 7 <System Maintenance></u>	21
<u>Non-Functional Requirements</u>	23
<u>GUI</u>	80
<u>Application Programming Interfaces (API)</u>	83
<u>System Architecture</u>	25
<u>System Evolution</u>	84

Purpose

The purpose of this document is to set out the requirements for the development of the Academy RS application. The Academy RS is a web application which allows users to report any issues they may be having with the LMS rather than emailing or calling. The Support Team will receive the users issue report with the necessary information and act accordingly to resolve the problem. Academy RS will store resolved issues in a database and will use that information to create key performance indicators that can be shared within the organisation allowing staff to be recognised for their contributions.

The intended customers are support staff and users of the Academy Learning management system ran by Olive Media. I chose the Academy LMS because I spent six months working there as part of the support team and I feel this application would benefit users and staff greatly.

Project Scope

The scope of the project is to develop a fully functional cloud based application that allows users to report issues they are having using the LMS. The system shall be fully responsive website so users can report issues on their mobiles if they need to. Users issue reports will be stored into a database and used to create KPI charts.

I was involved in discussions with many support team members from Olive Media to elicit the following requirements:

- If user is new, they must be able to create new account.
- Else, if user exists, users must be able to identify themselves.
- System must have different interfaces for each user type e.g. Managers, Support staff and Customers.
- All user reports must have detailed description of the issue.
- Staff members must store all solved issues in the database.
- The system must have automated KPI Charts function.

Definitions, Acronyms, and Abbreviations

LMS	Learning Management system
KPI	Key Performance indicator
Users	Users would be Managers, Support team and Customers of the Academy.
Participant	Any registered person that takes part in usability tests.

User Requirements Definition

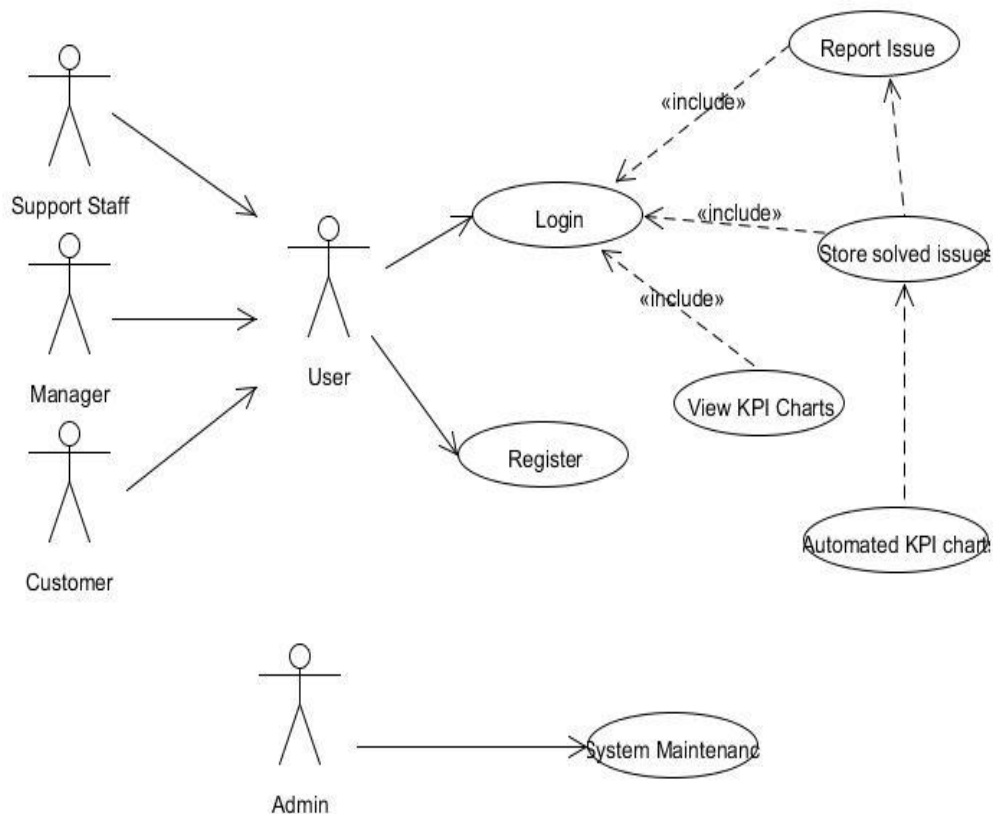
- The user must be a manager, support team member or customer of The Academy.
- The user must have a valid email address.

Functional requirements

1. Register
2. Login
3. Issue Reporting
4. Store solved issues
5. Automated KPI's
6. View KPI's
7. System Maintenance

Use Case Diagram

The Use Case Diagram provides an overview of all functional requirements.



Requirement 1 <Registration>

Description:

This is a webpage on The Academy RS website that will allow users to sign up to gain access to our service. Users will be required to provide us with profile information. We will use a HTML form to submit users profile information, which will then be used to create a new user and save its attributes to our database.

Priority:

High

Technical issues:

Users must fill in key criteria in order to complete the registration.

Use Case:

Register

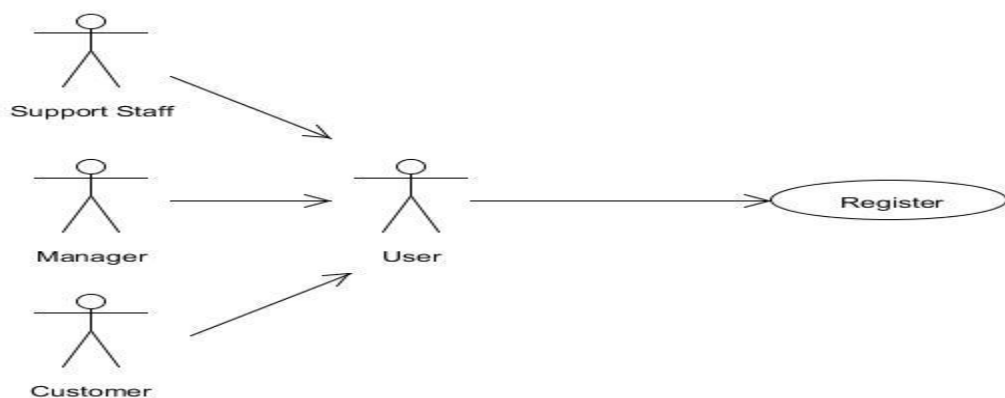
Scope:

The scope of this use case is a registration system for the user.

Description:

This use case shows the user registering to use The Academy RS service.

Use Case Diagram:



Flow Description

Precondition

None

Activation

This use case starts when a user want to register to use The Academy RS service.

Main flow

5. User opens The Academy RS home page.
6. User clicks “Register” from home page.
7. The actor fills out the form [A1 Form Validation]
8. The system stores the information provided by the user in the database.

Alternate flow

- A1 - Insufficient Form Data Provided
 - If in the Main Flow the user enters invalid profile information or does not fill in the required fields, the system displays an error message. The actor can then choose to rectify the errors or cancel the registration, at which point the use case ends.

Exceptional flow

None

Termination

The system stores all the registration data in the Academy RS database as a new record.

Post condition

If the use case was successful, the actor is now registered as a member in the system. If use case was unsuccessful, system is unchanged.

Requirement 2 <Login>

Description:

Allows a user to securely login to the Academy RS service.

Priority:

High

Technical Issues:

Users will be prompted to login using email/password combination.

Dependencies:

Requires user to have existing account with Academy RS

Use Case:

Login

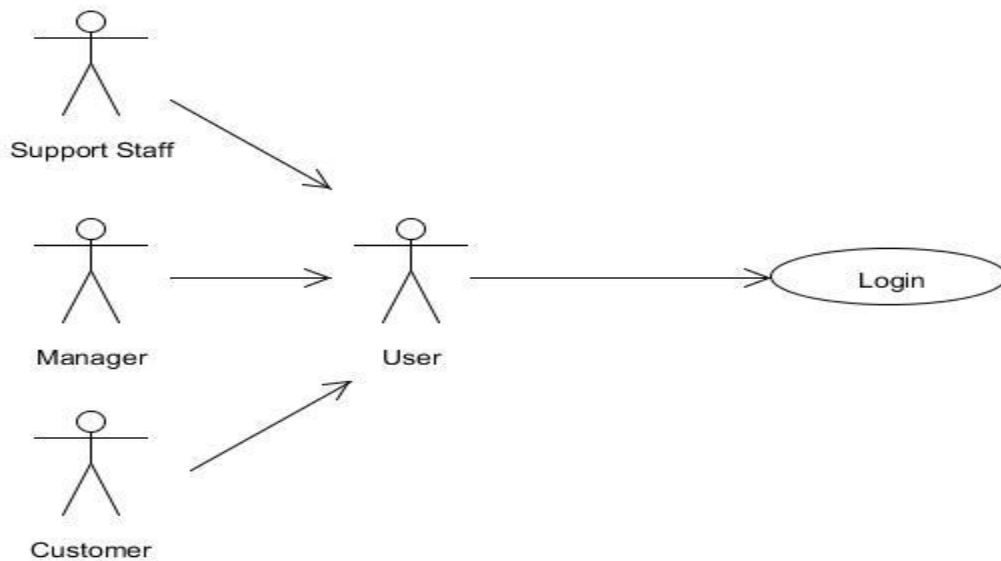
Scope:

The scope of this use case is a login system for the user.

Description

This use case describes how the user logs in to the Academy RS website.

Use Case Diagram



Flow Description

Precondition

The user is a registered member on the Academy RS system.

Activation

This use case starts when an actor wishes to log in to the Academy RS website.

Main flow

4. The system requests that the actor enters their email and password
5. The actor enters their email and password. [A1 invalid email/password]
6. The system validates the provided email and password and completes the login function allowing the actor access to our service.

Alternate flow

- A1-Invalid Email/Password
 - If in the main flow the actor enters an invalid email and/or password, the system will display an error message. The actor can choose to return to the start of the main flow or cancel the login, at which point the use case ends.

Termination

The system stores all login information.

Post condition

If the use case was successful, the actor is now logged into Academy RS. If unsuccessful, the system is unchanged.

Requirement 3 <Reporting issues>

Description:

This use case describes the user reporting an issue.

Priority:

High

Use Case:

Reporting issue

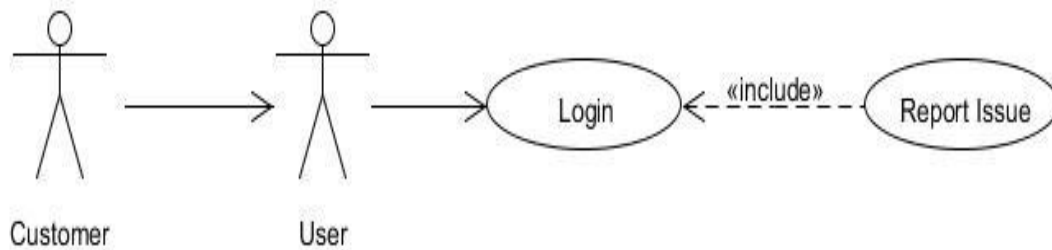
Scope:

The scope of this use case is to describe the process of how a user logs an issue report to the system.

Description

This use case describes the process of how a user logs an issue report to the system so it can be reviewed and resolved by support staff.

Use Case Diagram



Flow Description

Precondition

The system is in a ready state mode.

Activation

This use case starts when a user logs in to the Academy RS system.

Main flow

10. The user will be directed to the register page
11. The user enters their full name
12. The user enters their email address
13. The user submits a password for their account
- 14. [A1 Skip Registration]**
15. The user logs into the system
16. The system loads and displays the user information
17. The user logs issue report into the system
18. The system saves the data into a database

Post condition

Issue is reported and stored in database.

Requirement 4 <Storing Resolved Issues>

Description:

This use case describes the support staff submitting solved issues to the database.

Priority:

High

Dependencies:

User must report an issue.

Support staff must resolve the issue.

Use Case:

Storing solved issues

Actors:

Support Staff

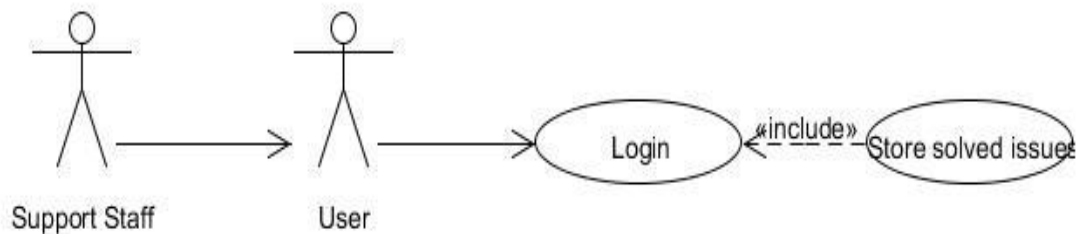
Scope:

The scope of this use case is to describe the process of storing solved issues.

Description

This use case describes support staff submitting details of solved issues to the Academy RS database so that it can be used to create KPI charts.

Use Case Diagram



Flow Description

Precondition

The system must be on

Activation

This use case starts when support staff has logged on to the system.

Main flow

5. The actor logs into the system. [A1 Invalid Username/Password]
6. The actor views reported issue.
7. The actor solves user issue.
8. The actor submits solved issue details to the database.

Alternate flow

- A1-Invalid Email/Password
 - If in the main flow the actor enters an invalid email and/or password, the system will display an error message. The actor can choose to return to the start of the main flow or cancel the login, at which point the use case ends.

Post Condition

Solved issues are stored in the database.

Requirement 5 <Generating KPI Charts>**Description:**

This use case describes the system generating automated KPI charts using the solved issues stored in the database.

Priority:

High

Dependencies:

User must report an issue.

Support staff store solved issues.

Use Case:

Automated KPI Charts

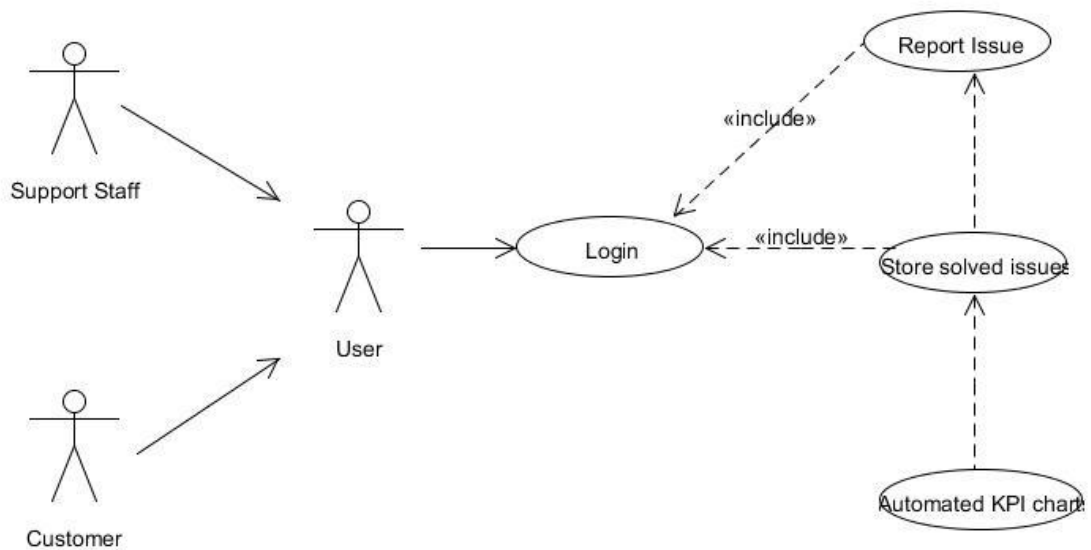
Scope:

The scope of this use case is to describe the process of automating KPI Charts.

Description

This use case describes the system gathering solved issues from the database and displaying them in a chart format.

Use Case Diagram



Flow Description

Precondition

The system must be on

Activation

This use case starts when support staff has submitted a solved issue.

Main flow

8. The customer logs into the system. [A1 Invalid Username/Password]
9. The customer reports issue.
10. System stores reported issue in database.
11. The support staff views reported issue.
12. The support team member solves the users' issue.
13. The support team member submits solved issue details to the database.
14. System pull solved issue details from database and displays them in KPI chart.

Alternate flow

- A1-Invalid Email/Password
 - If in the main flow the actor enters an invalid email and/or password, the system will display an error message. The actor can choose to return to the start of the main flow or cancel the login, at which point the use case ends.

Post Condition

KPI charts can be displayed to user.

Requirement 6 <View KPI Charts>

Description:

This use case describes the Manager viewing KPI charts.

Priority:

High

Use Case:

Viewing KPI's

Actors:

Manager

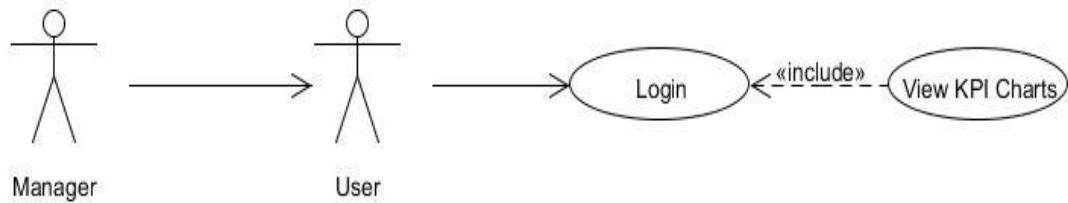
Scope:

The scope of this use case is to describe the process of a manager viewing KPI's.

Description

This use case describes Manager logging in and viewing automated KPI's

Use Case Diagram



Flow Description

Precondition

The system must be on

Activation

This use case starts when a manager has logged on to the system.

Main flow

3. The actor logs into the system. [A1 Invalid Username/Password]
4. The actor views KPI charts.

Alternate flow

- A1-Invalid Email/Password
 - If in the main flow the actor enters an invalid email and/or password, the system will display an error message. The actor can choose to return to the start of the main flow or cancel the login, at which point the use case ends.

Requirement 7 <System Maintenance>

Description:

This use case describes the tasks that are carried out in order to optimize the systems performance and security checks.

Priority:

High

Use Case:

System Maintenance

Actors:

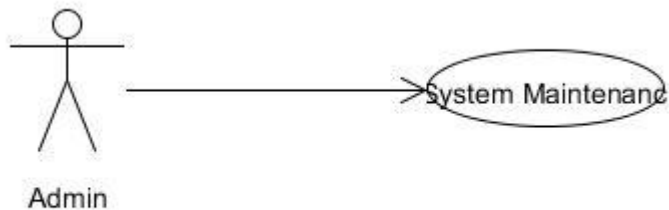
Admin

Scope:

The scope of this use case is to describe the process of maintaining the Academy RS system.

Description

This use case describes the admin operations on the system.

Use Case Diagram**Flow Description****Precondition**

The system must be on

Activation

This use case starts when a admin has logged on to the system.

Main flow

1. The actor logs into the system. [A1 Invalid Username/Password]
2. The actor enters begins maintenance/diagnostic checks and operations on the system and or database
[A2 Security management]

Alternate flow

- A1 Invalid Name/Password
 - If the actor enters incorrect Username and/or password, the system will show an error message. The actor can choose to either return to the start of the Basic Flow or abandon the login, at which point the use case ends.
- A2 Security management
 - The actor starts up the security manager application to perform any security checks required.

Exceptional flow

In the case of a system crash the system will restore to the most recent updates when rebooting

Termination

The use case ends when all the maintenance operations have been completed and the actor logs out of the system.

Non-Functional Requirements

Physical requirement

This section describes the environment where the equipment is to function and if there are any environmental restrictions, such as temperature, humidity or magnetic interference.

Requirement 1 “Cloud Server”

The system will require a server on which to operate .The server will be hosted on a free cloud service called Heroku. Heroku has a full REST API and integrates with third party services instantly. Heroku's large catalog of add-on services is constantly growing, and includes services like databases, caching, monitoring, performance management, email, transcoding, search, billing, and many more.

Requirement 2 “Client Machines”

The system will require client machines to use the website. Users can log on using a desktop/laptop.

Interface requirement

My application will run using a Graphical User Interface (GUI). The operations will be carried out using keyboard and mouse.

My application will have a different GUI for each type of user.

User & Human requirement

• Who will use the system?

Managers, Support, and Customers will be using the system

• Will there be several types of users?

Managers, Support, and Customers will be have different accounts

• What is the skill level of each type of user?

There will be a minimal skill level of each type of user, except for the admin who may require further training.

• How easy will it be for a user to understand and use the system?

There written tutorial for customers on how to use the system.

Security requirement

Users will need a username and a password to log in to Academy RS.

The system will be backed up regularly and the backed up files will be stored on a local server.

Maintainability requirement

The system shall provide the capability to back-up the Data such as the database. The database shall keep a log of all the errors, so the admin can fix them as soon as they can.

Resource Utilization requirement

What materials, personnel, or other resources are required to build, use and maintain the system?

The system needs an admin to maintain and manage the system.

What skills must the developer have?

The developer should have knowledge of Ruby on Rails, HTML, CSS, JAVASCRIPT and MySQL.

How much physical space will be taken up by the system?

The system does not require physical space as it is on the cloud.

Availability requirement

The system must be available 24/7 as the academy LMS is used globally.

GUI

This shows how the new member can use our registration form to set up a new account with Academy RS

Academy RS Registration

Name:

Email:

Password:

Password:

[Already a member? Click here to login!](#)

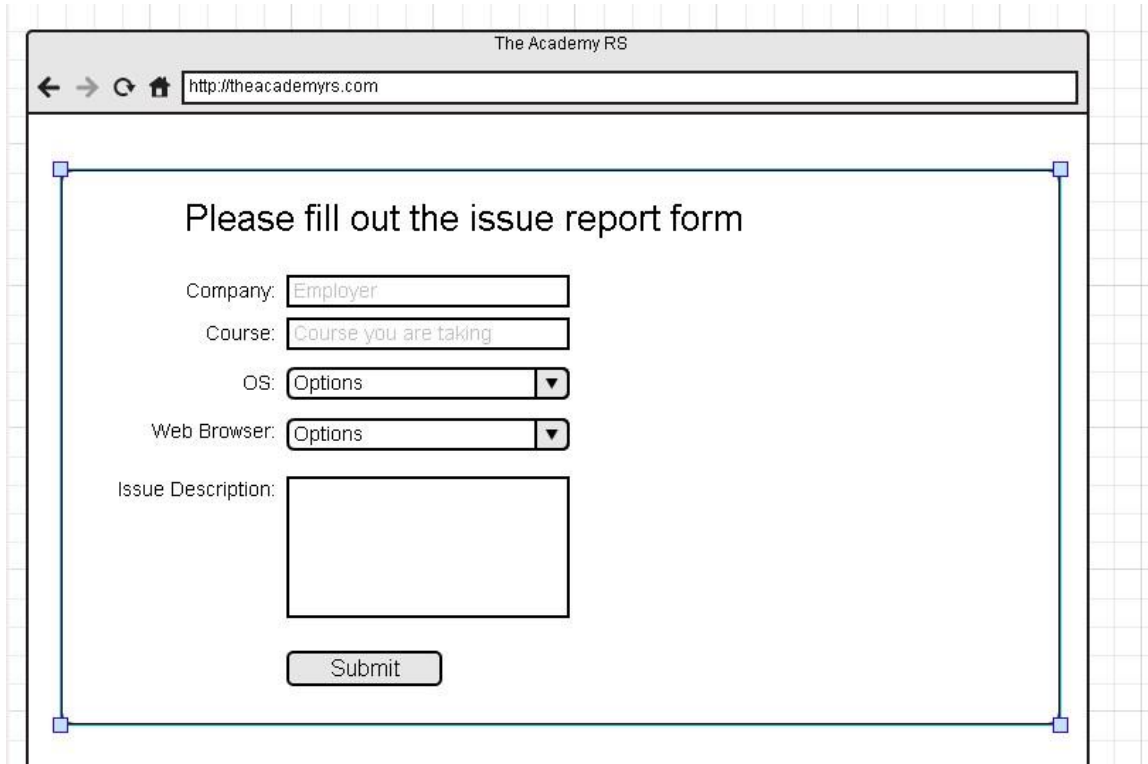
This is a mock-up of the Login form existing members will use to gain access to our web service.

Academy RS Login

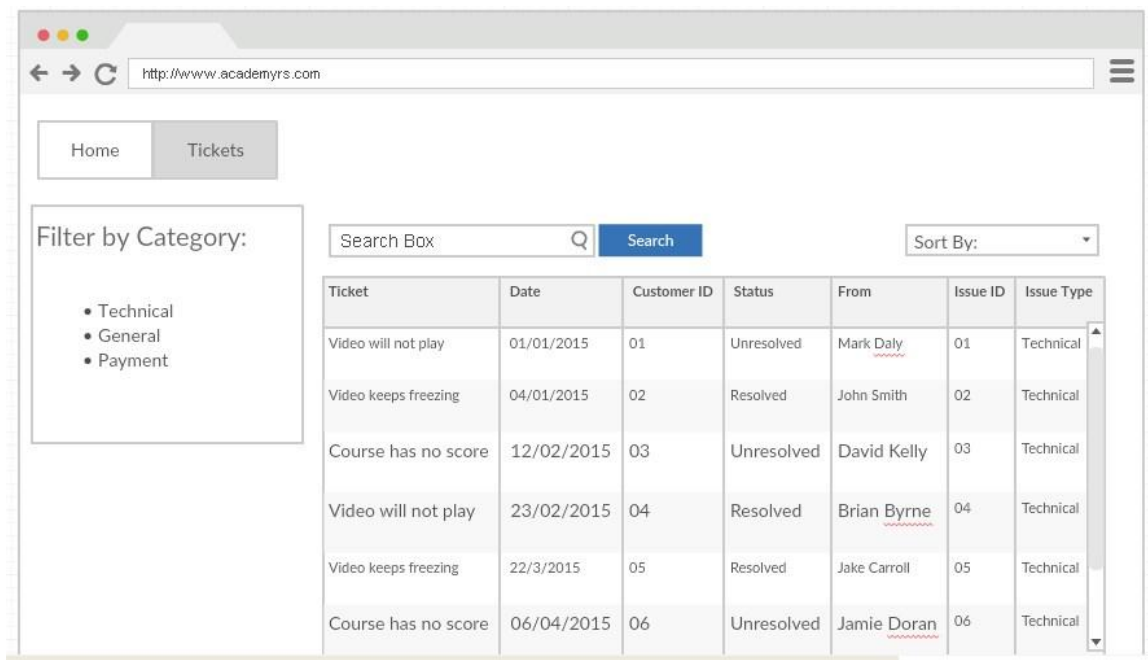
Email:

Password:

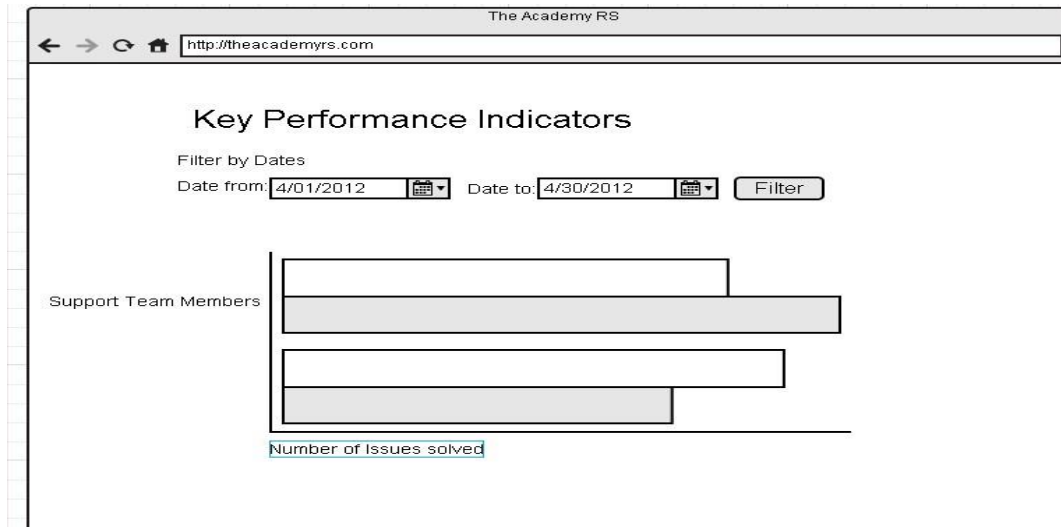
Customers can report issues to support staff like show below.



Support interface for Issue reports received.



KPI charts page

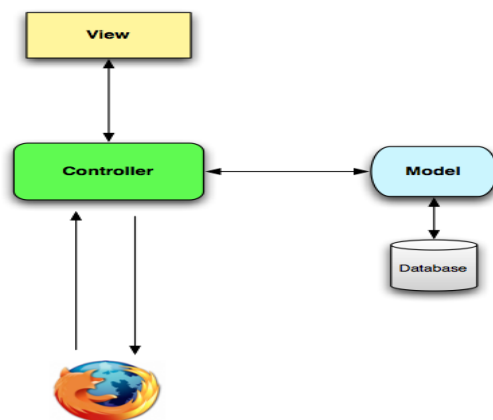


Application Programming Interfaces (API)

An API will be created.

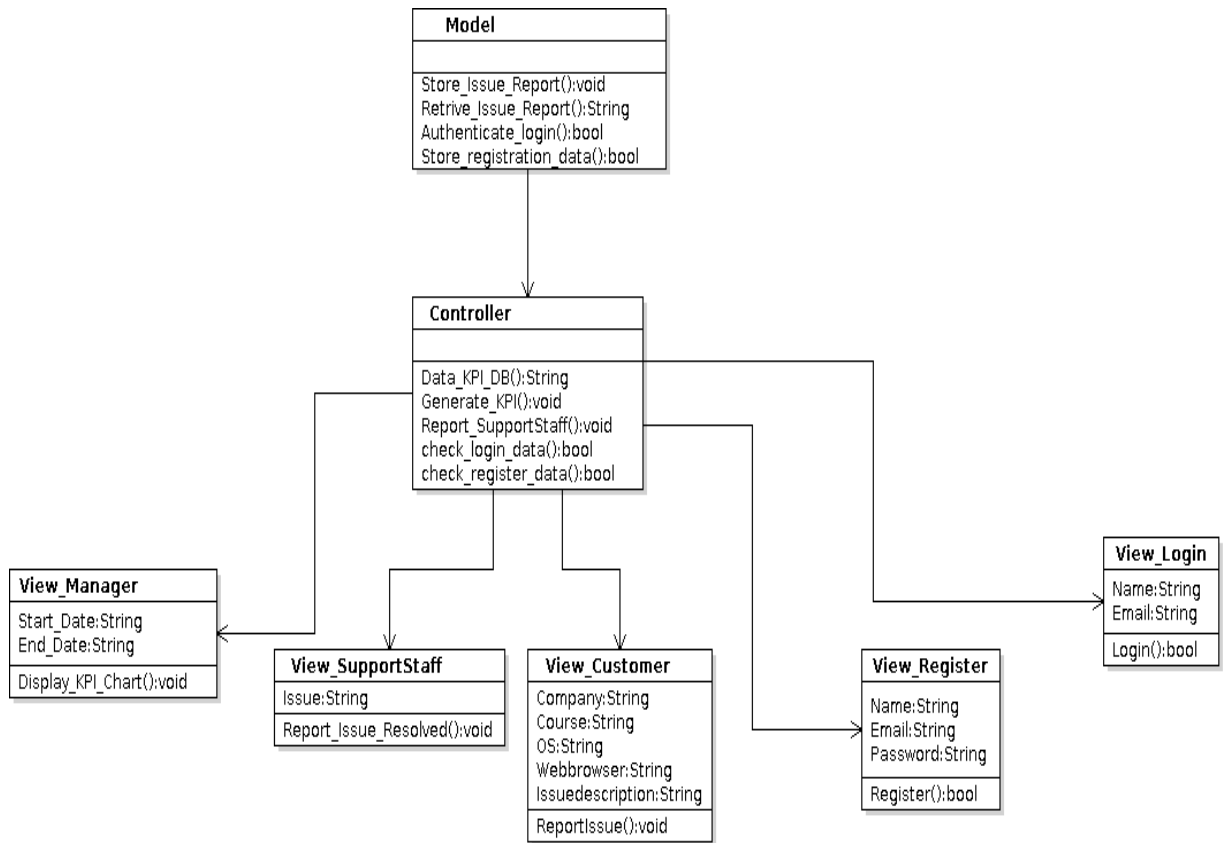
System Architecture

Here is a high level overview of Academy RS.MY application will use a Model-View-Controller architecture. MVC separates the “domain logic” from the input and presentation logic associated with a GUI.



When interacting with a Rails application, a browser sends a request, which is received by a web server and passed on to a Rails controller, the controller interacts with a model, which is a Ruby object that represents an element of the site and is in charge of communicating with the database.

UML



System Evolution

The system could evolve over time to include a chat function that will allow customers to instant message support staff to receive quicker feedback regarding queries.

11.3 Monthly Journals

Reflective Journal

Student name: Stephen Doolan
Programme (e.g., BSc in Computing):
Month: September

This month I actually committed to a project Idea. I had so many different ideas and could not decide on any, which has been very stressful. I finally decided on an IT Help Desk with a few extra features. After finally deciding on my project I had to go about setting up the tools I need to complete the project. I wanted to develop my project using ruby on rails so I went to <http://rubyonrails.org/download/> and downloaded ruby. I attempted to install ruby following their documents but couldn't so I followed a quick tutorial on youtube and got everything working.

I needed to do some research into IT help desks for my project proposal.

My Achievements

This month I successfully installed and configured ruby on rails. I downloaded and configured git and also setup a github repository for my project. I set up an account on Heroku also so I can host the application on the cloud. I completed my project proposal in time for the submission date.

My Reflection

I felt that adapting to using the command line for ruby on rails is a challenge for me considering I have mostly used GUI's but I am looking forward to the experience. By hosting my application with heroku I will learn how to configure application on the cloud.

Intended Changes

Next month, I plan on learning ruby on rails syntax and the rails framework by using books on the internet and tutorials.

Reflective Journal

Student name: Stephen Doolan

Programme (e.g., BSc in Computing):

Month: October

Once I had submitted my project proposal on the 6th of October, I turned my attentions to the Requirements specifications document. This document details all the information regarding my project. I thought this assignment was very stressful as I had learn how to do UML diagrams all over again in order to draw my use case diagrams and class diagram. I was very unsure about what I was doing was actually correct.

I learned ruby on rails syntax for the following:

- Assignment operation, Strings ,Conditionals(ifelse), Loops, Array , Hash and Classes

I also researched some of the gem files that would be helpful for my application at <https://www.ruby-toolbox.com/>

My Achievements

I have finished the 1st draft of my requirements document complete with use case and class diagrams using Umlet.

Learned ruby on rails syntax and learned about available gem files.

My Reflection

I feel that the requirement specification document was a really important process in preparing me to develop the project as it forced me to detail how I want the application to work. Now that I do not have to worry about completing this document I can focus on actually developing the project.

Intended Changes

Next month I want to get started on the development of the project. Create models and controllers.

Supervisor Meetings

Date of Meeting: 20th October 2015 and 21st October 2015

Items discussed: Project proposals and requirements specifications

Reflective Journal

Student name: Stephen Doolan

Programme (e.g., BSc in Computing):

Month: November

This month I will be focused on developing the project after submitting my Requirements document. I downloaded Rubymine IDE which is specifically for ruby on rails. I will create my academyRS project as well as controllers and models.

My Achievements

This month I was able to create my first project with ruby on rails and edit it using Rubymine. I have been learning ruby syntax from this book

<https://www.railstutorial.org/book>

- Created users and tickets controllers
- Created users, tickets, browser and operating system models.
- Ran project on the server
- Made my first commit to github repository

My Reflection

I finally feel like my project is moving in the right direction. It was great to see it running on the server although there was no functionality.

Intended Changes

Next month I would like to work on the associations between models.

Supervisor Meetings

Date of Meeting: 3rd November 2015

Items discussed: requirements specifications.

Frances gave me the all clear to submit my RS document after looking over it.

Reflective Journal

Student name: Stephen Doolan

Programme (e.g., BSc in Computing):

Month: December

This Month I will be focusing on User authentication, authorisation and roles using the Devise, Rolify and CanCan Gems.

My Achievements

This month I did the following:

- Installed Devise, Rolify and CanCan gems
- Created Manager, Support Staff and Customer user roles.
- Configured abilities.rb and Ticket to authorise users abilities within the application

My Reflection

I ran into a few errors when using these Gems but I feel it was a great learning experience solving them through trial and error.

Intended Changes

Next Month I would like to get to work on the Ticket assigning and updating ticket status.

Supervisor Meetings

Reflective Journal

Student name: Stephen Doolan

Programme (e.g., BSc in Computing):

Month: January

This Month I will be focusing mainly on my mid-point report and presentation while also attempting to add greater functionality to the application..

My Achievements

This month I did the following:

- Added status update functionality to the ticktes.
- Added functionality that allows support staff or managers assign supports staff to particular issues.
- Improved front end design.
- Completed first draft of my presentation.

My Reflection

I ran into a few errors when using these Gems but I feel it was a great learning experience solving them through trial and error.

Intended Changes

Next Month I will focus on search functionality.

Supervisor Meetings

Date of Meeting: 3rd february 2016

Items discussed: Mid-Point Presentations.

Frances gave us pointers on our slides for our presentation

Reflective Journal

Student name: Stephen Doolan

Programme (e.g., BSc in Computing):

Month: March

This month I will be focusing on on the KPI charts and browser detection

My Achievements

This month I did the following:

- Added key performance indicator charts.
- Added browser detection
- Added operating system detection
- Hosted AcademyRS on heroku

My Reflection

I feel that I have added the most vital parts of my project this month and it has been very successful. I think that hosting the project went smoothly.

Intended Changes

I intend to make subtle changes to the GUI.

Supervisor Meetings