

Enterprise Development

Software Design Document(s) for Distinction (D) Software

Prepared by: Nguyen Tran Nguyen (658600)

[Optional Feedback, timeline and schedule]

Submission for Feedback: 6 June 2016 (Monday of Week 14) 9:00am

Tutor's feedback: 8 June 2016 (Wednesday of Week 14) 5:00pm

[Final Submission]

Submission for Portfolio: 13 June 2016 (Monday of Week 15) 9:00am

Instructions - This document is for students aiming to achieve Distinction (D) or above.

Intended Learning Outcomes (extracted from Unit Outline)

- ILO1. Use a range of APIs and technologies to build enterprise applications and explain why the APIs and technologies were selected to develop the applications
- ILO2. Perform independently research into a range of APIs and technologies so as to select appropriate technologies (with justification) to build enterprise applications; during the research process you should be able to present your findings and reasoning why such decisions are made
- ILO3. Design (with justifications) and describe an enterprise architecture for a software solution to a given business scenario. The justification should ideally include at least the following topics:
 - a. the choice of any APIs and technologies
 - b. the selection of architectural patterns and the use of any best practices
 - c. any security issues and concerns and how to mitigate the potential threats
- ILO4. Develop end-to-end features of enterprise applications to given business scenarios. Ideally, you should demonstrate your understanding of at least the following topics
 - a. The choice of any APIs and technologies
 - b. The selection of architectural patterns and the use of any best practices
 - c. The choice of enterprise technologies to mitigate any potential threat raised by security issues and concerns

Software Title: SAllocate+ – A staff scheduling system

Introduction

Staff work load and staff scheduling are essential parts of internal processes in service industry such as Taxi service, Trader service like plumbing, asbestos removal, electrical installations, stocktaking. Ranging from large enterprise to medium and small businesses, well prepared staff management and scheduling process is important to improve services quality & customer satisfaction. The SAllocate+ is a software system that provides a hassle-free and automation process of job staff allocation and staff management. This system is built specifically to support staff allocation on stocktaking service.

Business Scenario

RGIS is large provider of stocktaking service for Australia retailers. They have a large number of employees working as stocktakers and supervisors in all states of Australia. As the number of stocktake jobs are growing rapidly to a point that the existing management system of staff allocation by using phone call, SMS & email starts showing sign of clunky, shorted-sign vision, inefficient, and time consuming. There is a need of an automated system that provides better tool of managing staff allocation to make sure that all stocktake jobs have adequate number of staff with correct skill sets.

This system will be developed in multiple components:

- An administration website to manage staff, and provide a central work space to allocate staff into jobs.
- A mobile app for staff to receive job notifications, set choosing job preferences, and receive job allocation confirmation.

A full workflow of the job staff booking and allocation process is as follow:

Step 1. Send job alert process

1. The job data & staff data is managed by a different department and already made available in the database
2. A district manager (DM) request to view a list of current and future jobs of his district.
3. He select unallocated jobs from the list and mark them ready to send email message.
4. After the DM select jobs, he then select staffs from a staff list filtered by his district. He can choose to select staff from other district if he needs to.
5. After choosing the staffs, a message composer box will show up with the detail of jobs he selected before. He can customize the message if he needs to.
6. He finish this process by click to Send button. The email message will send to the chosen staffs, the jobs will be marked as Email Sent, the list of those selected staff can be viewed in the job detail page.

Step 2. Staff choose their preference for a job send to them

1. Once a staff receive a job alert email. He open the SAllocate app.
2. He need to login using his/her username & password
3. After login, the staff can see a list of all the jobs that he can work on.
4. He open a job, review the job detail and confirm if he is available or not available for this job.

5. Once he confirm his availability, the job will be removed from the list.
6. This process is finished when there is no more jobs in the job list.

Step 3. Review staff preference on a job.

1. Back to the DM, he open a job detail that is marked as Email Sent.
2. He can review job detail information and the staff availability responses in a table.
3. He then confirm which staff will be working on the job based on this availability preferences.

This process is completed when the number of staff required for the job is fulfilled.

Table of Contents

1. Software Requirements.....	5
1.1. Requirements Justification	5
1.2. Functionality and Technology Matrix.....	5
2. Software Design.....	6
2.1. Overall architecture of SAllocate+.....	6
2.1.1. Data Access Layer (DAL):	6
2.1.2. Business Logic Server (BLS):.....	7
2.1.3. Application Layer (AL):.....	7
3. Sample Coding.....	8
3.1. Sample code of Job entity defined by using EntityFramework Code First	8
3.2. Sample code of Job Data Access Object (JobDAO).....	8
3.3. Sample code of Job Service as a business object with the WCF interface and configuration	10
3.4. Sample code of the Account Service when processing sensitive data.....	12
3.5. Sample code of the EmailService	13
3.6. Sample code of the EncryptionService.....	13
3.7. Sample code of the JobController api in WebAdminUI project	13
3.8. Sample code of angular js template and controller of job listing page.....	14
4. Software Testing Results	16
4.1. Test send job alert process.....	16
4.2. Test job available selection from staff.....	22
4.3. Test review staff availability for a job.....	24
5. References	24

1. Software Requirements

1.1. Requirements Justification

Based on the above workflow process, the following functionalities is required for the administration website:

- F1. Allow admin to review jobs information, viewing staff profiles.
- F2. Allow admin to send out job notifications for a set of period, normally a weekly job schedule of the upcoming week.
- F3. Allow admin to review the job choice preferences from staff and confirm which staff should be allocated on the jobs.
- F4. Sending out the job schedule confirmation message.

Functionalities and features of the SAllocate+ mobile app:

- F5. Login / logout staff
- F6. Receive job notification alerts and review job detail.
- F7. Set their availability preference if they want to work on a job.
- F8. Receive job schedule confirmation.

1.2. Functionality and Technology Matrix

The following table shows the relevant technologies discussed in this subject that could be used to implement the functionalities as suggested in the Functionalities section above.

Functionality	Related Technology discussed in this subject
F1	Using ASP.NET MVC for web layer, Ajax for communication with the server. The database access layer will be done using EntityFramework 6.0. A Business Logic Server will be implemented and expose their logic via WCF (Session Façade) and Web Services (Web Services Façade).
F2	Using a messaging services to send job schedule alerts to multiple staff's email address at once.
F3	Using a Business Logic server to process the availability preferences of staff
F4	Using a messaging services to send confirmations to multiple staff at once
F5	Using a security layer to authenticate & authorize staff login information.
F6	Using a messaging services to receive job alerts using pull technique
F7	This function will be done using a REST api which connects to the same Business Logic Server in F1 to handle the availability selection on mobile app. The REST api is developed using WebAPI 2.0 from the ASP.NET MVC framework. The communication protocol use JSON data format to transfer data
F8	Using a messaging services to receive notifications using push or pull technique

Table 1. Functionality and Technology Matrix

2. Software Design

2.1. Overall architecture of SAllocate+:

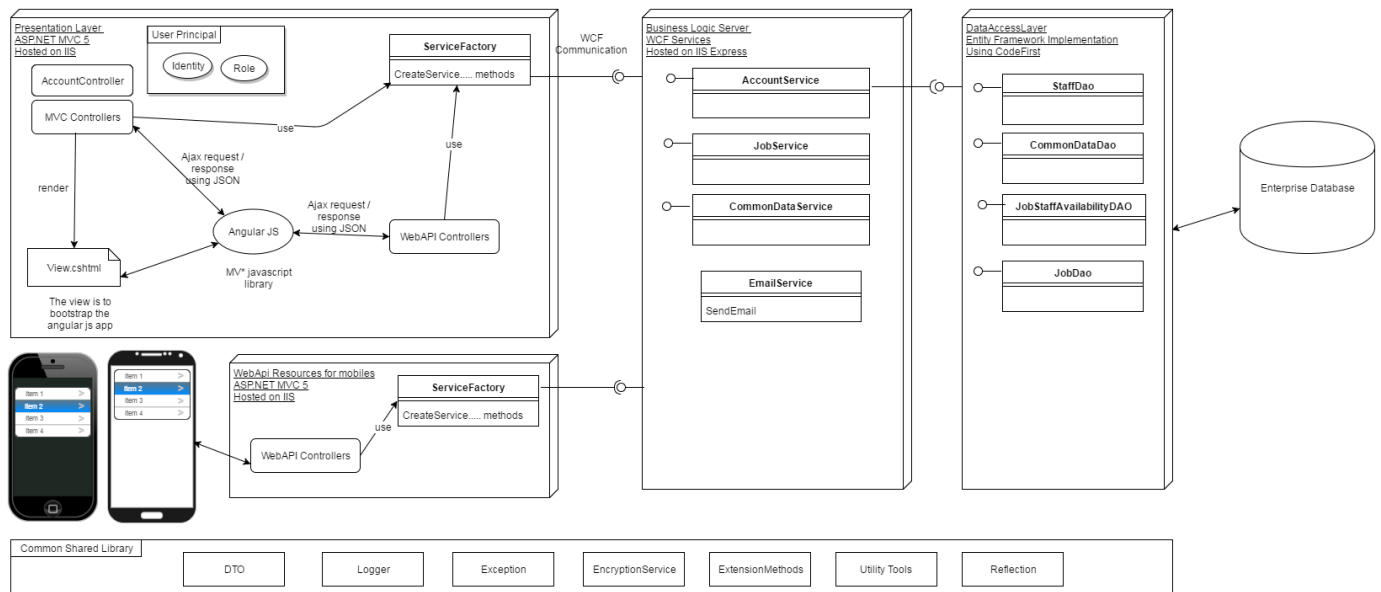


Figure 1. Overall architecture of SAllocate+

This system utilizes 3-tier architecture design:

2.1.1. Data Access Layer (DAL):

This layer is responsible for all data access & data manipulation in the whole application. It contains a number of data access objects (DAOs) that contains CRUD operation methods required by the application. This layer is defined in 2 .Net project libraries:

- a. **Tna.SAllocatePlus.DataAccessLayer**: This project define the main structure of how data should be accessed by the upper level by using a number of DAO interfaces. For example, the IJobDao is a DAO interface that defines all the methods related to accessing and manipulating job data. Entity objects using for ORM is defined in this project.

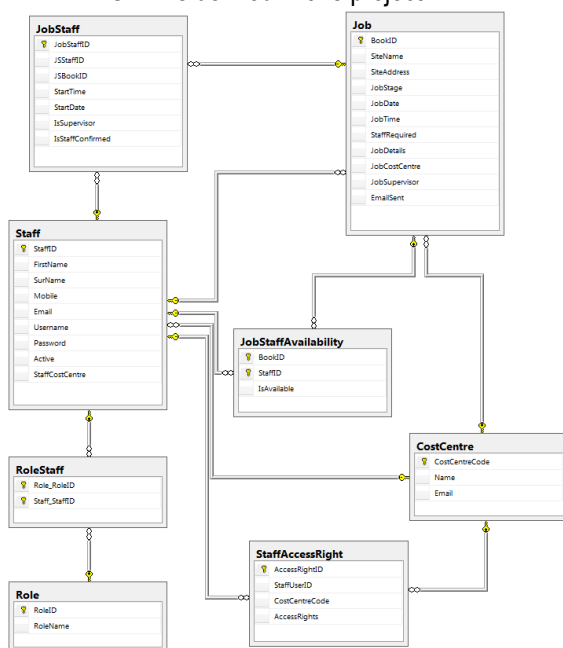


Figure 2. Database table schema generated by EF CodeFirst

- b. **Tna.SAllocatePlus.DataAccessLayer.EF**: This project is an implementation of the DataAccessLayer using Entity Framework as the Object Relation Mapping from the physical DataTable in Database layer to the Entity object defined in the above project. In this project, Entity Framework is implemented in Code First strategy to minimize the deployment process. This project contains a database migration scripts that can be run on Visual Studio to automatically create database or update the database to latest version.

By creating a level of abstraction on top of the implementation of database access using Entity Framework, this design provides a flexibility that the implementation of the DataAccessLayer can be replaced from Entity Framework to a different ORM technology and implementation if required. For example, in an event where the company is required to change their database storage to Oracle or a Cloud database, this will be done easily by creating a separated implementation of the DataAccessLayer for that required database layer.

2.1.2. Business Logic Server (BLS):

This layer is a middleware server that is responsible for implementing all the business logics, business rules, validations and workflow processes. It contains a number of business logic objects that exposing their services to the network using Windows Communication Foundation (WCF) technology. Again, in this layer, each of the services are defined using interfaces. WCF uses these interfaces along with the configuration to create and publish the services using WSHttpBinding and HTTP protocol. Other software project with proper network access authority and valid credential can access those services. The business logic objects are defined as the implementation of the interfaces are also defined in this project. This layer is implemented in a single .Net project that includes all the interfaces, configuration and the business objects. This project also contains a messaging helper library that allows sending email.

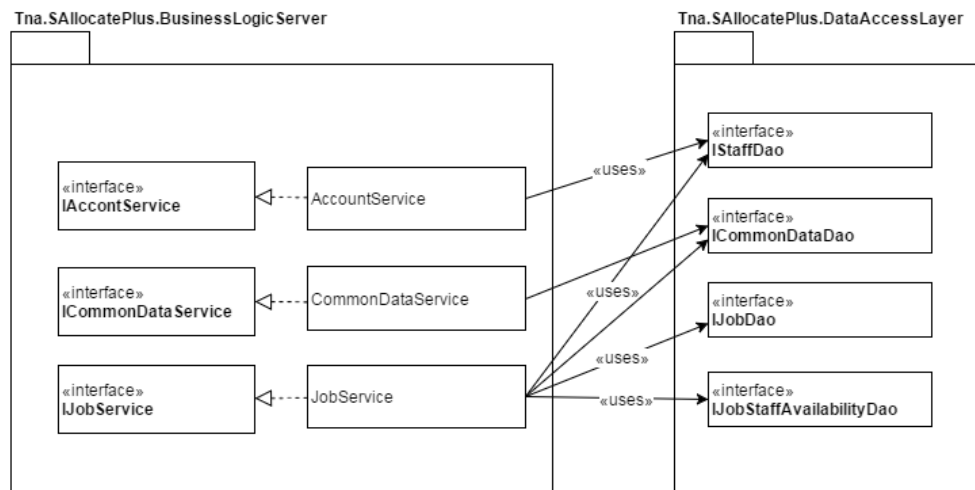


Figure 3. Business logic server component model

Using a Business Logic Server has many benefits. First, by using WCF to implement this layer, the business logic server can be hosted in many ways: inside IIS, self-hosted in a Windows console application or can be run as a Windows service. Another benefit of this layer is that, it provides a centre places to put business logic separately from the presentations and database access technology and allow these business logic be reusable by multiple applications. For example, in this project, the business logic is utilized by 2 separated projects in the Application Layer.

2.1.3. Application Layer (AL):

Application layer takes responsibility to render views, capturing user interaction and direct them to the proper business objects. In this design, this layer also include a public API in Restful format to support data accessing to job and staff account from mobile devices.

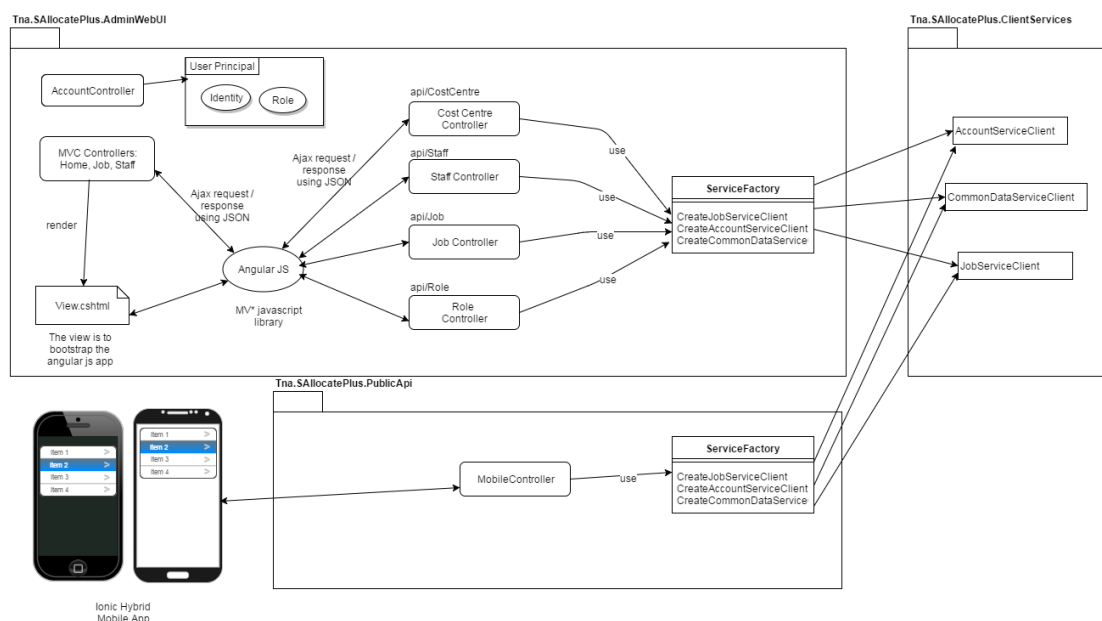


Figure 4. Component diagram of Application Layer

In more detail, this layer consists of 4 separated projects:

- c. **Tna.SALlocatePlus.ClientServices:** This project contains all the client proxies that are created from the WCF services in the BLS. These proxies are shared for the AdminWebUI and PublicApi projects.
- d. **Tna.SALlocatePlus.AdminWebUI:** This project is an MVC web application hosted on an IIS server and works as a web administration backend. This project contains a number of MVC controllers and WebApi controllers to render admin pages, capture user interactions, navigation and collect data requests. Client side view, and interaction are implemented using AngularJS framework to provide a better user experience, quick response and performance optimization. Almost all communications between client web browsers and the IIS Server are done via Ajax requests in JSON format. Account profile feature that is sensitive and require higher security is implemented by using traditional MVC controller with Razor template engine for page rendering. This project implement a Web Security layer by using FormsAuthenticationProvider to identify a user using login form method with a user name and password. The resources provided by this project is granted based on the user role. There are 2 roles currently setup in this system: Administrator and Employee. Administrator can manage jobs and staffs profile information while a user with Employee role can only maintain their own profile.
- e. **Tna.SALlocatePlus.PublicApi:** This project provides an application programming interface (API) that expose selected services for accessing job data and staff account from mobile platform. This project is implemented as a single MVC web application that contains only WebApi controller.
- f. **SALlocateApp:** This project is the mobile application interface of the SALlocate+ system that targeting to employee users. It is a hybrid mobile app developed using Ionic framework that can be deployed cross platform.

3. Sample Coding

3.1. Sample code of Job entity defined by using EntityFramework Code First

```
public class Job
{
    [Key]
    [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    public int BookID { get; set; }
    [Required]
    public string SiteName { get; set; }
    [Required]
    public string SiteAddress { get; set; }

    [Required]
    public JobStageEnum JobStage { get; set; }
    public DateTime? JobDate { get; set; }
    public TimeSpan? JobTime { get; set; }
    public int StaffRequired { get; set; }
    public string JobDetails { get; set; }
    [ForeignKey("CostCentre")]
    public string JobCostCentre { get; set; }
    [ForeignKey("Supervisor")]
    public int? JobSupervisor { get; set; }
    public bool? EmailSent { get; set; }
    public virtual Staff Supervisor { get; set; }
    public virtual CostCentre CostCentre { get; set; }
    public virtual List<JobStaff> JobStaffList { get; set; }
}
```

3.2. Sample code of Job Data Access Object (JobDAO)

```
public interface IJobDao
{
    List<Job> GetAll();
    List<Job> FindByRegion(CostCentre region);
    Job Get(int bookID);
    void Create(JobDto job);
    void Update(JobDto job);
    void Delete(Job job);
    void SetEmailSent(List<int> list);

    void CreateJobStaffPreference(List<int> jobList, List<int> staffList);
}
```



```

}

public class JobDao : IJobDao
{
    TnaContext _context;
    public JobDao()
    {
        _context = new TnaContext();
    }
    public List<Job> GetAll()
    {
        return _context.JobSet.ToList();
    }

    public List<Job> FindByRegion(CostCentre costCentre)
    {
        return _context.JobSet.Where(j => j.JobCostCentre == costCentre.CostCentreCode).ToList();
    }

    public Job Get(int bookID)
    {
        return _context.JobSet.FirstOrDefault(j => j.BookID == bookID);
    }

    public void Create(JobDto dto)
    {
        var job = new Job()
        {
            CostCentre = _context.CostCentreSet.FirstOrDefault(cc => cc.CostCentreCode == dto.JobCostCentre),
            EmailSent = dto.EmailSent,
            JobDate = dto.JobDate,
            JobDetails = dto.JobDetails,
            JobStage = dto.JobStage,
            JobSupervisor = dto.SupervisorStaffID,
            JobTime = dto.JobTime,
            SiteAddress = dto.SiteAddress,
            SiteName = dto.SiteName,
            StaffRequired = dto.StaffRequired,
            Supervisor = _context.StaffSet.FirstOrDefault(s => s.StaffID == dto.SupervisorStaffID)
        };

        _context.JobSet.Add(job);
        _context.SaveChanges();
    }

    public void Update(JobDto dto)
    {
        var existingJob = _context.JobSet.FirstOrDefault(j => j.BookID == dto.BookID);
        if (existingJob == null) throw new EntityNotFoundException();

        dto.MergeTo(existingJob, "BookID", "JobStaffList", "CostCentre", "Supervisor");

        existingJob.CostCentre = _context.CostCentreSet.FirstOrDefault(cc => cc.CostCentreCode == dto.JobCostCentre);
        existingJob.Supervisor = _context.StaffSet.FirstOrDefault(s => s.StaffID == dto.SupervisorStaffID);

        _context.SaveChanges();
    }

    public void Delete(Job job)
    {
        _context.JobSet.RemoveRange(_context.JobSet.Where(j => j.BookID == job.BookID));
    }

    public void SetEmailSent(List<int> list)
    {
        var jobList = _context.JobSet.Where(j => list.Contains(j.BookID));
        foreach (var j in jobList)
        {
            j.EmailSent = true;
        }
        _context.SaveChanges();
    }
}

```

```

public void CreateJobStaffPreference(List<int> jobList, List<int> staffList)
{
    var jobData = _context.JobSet.Where(j => jobList.Contains(j.BookID)).ToList();
    var staffData = _context.StaffSet.Where(s => staffList.Contains(s.StaffID)).ToList();
    var existingJobAvailability = _context.JobStaffAvailabilitySet.Where(a => jobList.Contains(a.BookID) || staffList.Contains(a.StaffID)).ToList();

    foreach(var staff in staffData)
        foreach (var job in jobData)
        {
            // skip existing
            if (existingJobAvailability.Any(a => a.StaffID == staff.StaffID && a.BookID == job.BookID)) continue;

            var jobStaff = new JobStaffAvailability()
            {
                Job = job,
                Staff= staff,
                IsAvailable = null
            };
            _context.JobStaffAvailabilitySet.Add(jobStaff);
        }

    _context.SaveChanges();
}
}

```

3.3. Sample code of Job Service as a business object with the WCF interface and configuration

```

[ServiceContract(Namespace = "http://Tna.SAllocatePlus.BusinessLogicServer")]
public interface IJobService
{
    [OperationContract]
    List<JobDto> GetJobsByCostCentre(string regionName);

    [OperationContract]
    List<JobDto> GetJobsByStaff(int staffId, bool hasAvailability);

    [OperationContract]
    void SetAvailabilityForJob(int bookId, int staffId, bool available);

    [OperationContract]
    List<JobStaffDto> GetStaffListForJob(int bookID);
    [OperationContract]
    void SendJobEmail(SendEmailRequestDto request);

    [OperationContract]
    JobDto GetJobById(int bookId);
    [OperationContract]
    List<JobAvailabilityDto> GetJobAvailabilityById(int bookId);
    [OperationContract]
    void UpdateJob(JobDto job);
}

public class JobService : IJobService
{
    IJobDao _jobDao = null;
    IStaffDao _staffDao = null;
    ICommonDataDao _commonDataDao = null;
    IJobStaffAvailabilityDao _jobStaffAvailabilityDao = null;

    public JobService()
    {
        _jobDao = new JobDao();
        _staffDao = new StaffDao();
        _commonDataDao = new CommonDataDao();
        _jobStaffAvailabilityDao = new JobStaffAvailabilityDao();
    }
    .....

    public void SendJobEmail(SendEmailRequestDto request)
    {
        var staffList = _staffDao.GetStaffsByIdList(request.StaffList).ToList();
    }
}

```

```

var costCentre = _commonDataDao.GetAllCostCentres()
    .FirstOrDefault(cc=>cc.CostCentreCode == request.CostCentre);
var fromEmail = costCentre.Email;

EmailService emailService = new EmailService();
bool isSentSuccess = true;
foreach (var staff in staffList)
{
    try
    {
        emailService.SendMail(fromEmail, staff.Email, "Job schedule for " + request.CostCentre,
request.Content.Replace("{StaffName}", string.Format("{0} {1}", staff.FirstName, staff.SurName)));
    }
    catch (Exception ex)
    {
        isSentSuccess = false;
        break;
    }
}
// update jobs
if (isSentSuccess)
{
    _jobDao.SetEmailSent(request.JobList);

    // create job staff data
    _jobDao.CreateJobStaffPreference(request.JobList, request.StaffList);
}
}

public List<JobDto> GetJobsByStaff(int staffId, bool hasAvailability)
{
    return Map(_jobStaffAvailabilityDao.GetAllByStaff(staffId, hasAvailability).Select(a => a.Job));
}

public void SetAvailabilityForJob(int bookId, int staffId, bool available)
{
    var ja = _jobStaffAvailabilityDao.Get(bookId, staffId);
    if (ja == null)
        throw new Exception("Availability is not found");

    ja.IsAvailable = available;
    _jobStaffAvailabilityDao.Update(ja);
}

public JobDto GetJobById(int bookId)
{
    return Map(_jobDao.Get(bookId));
}

public List<JobAvailabilityDto> GetJobAvailabilityById(int bookId)
{
    var jobAvailability = _jobStaffAvailabilityDao.GetAllByJob(bookId);
    return Map(jobAvailability);
}

.....

public void UpdateJob(JobDto job)
{
    // business rule validation
    if(string.IsNullOrEmpty(job.JobCostCentre))
        throw new Exception("Cost centre must not be empty");
    if(string.IsNullOrEmpty(job.SiteName))
        throw new Exception("Site name must not be empty");

    // if valid, pass on to DAO to do update
    _jobDao.Update(job);
}
}

```

WCF configuration in web.config for all business objects in BLS

```

<system.serviceModel>
  <protocolMapping>

```

```

    <add scheme="http" binding="wsHttpBinding" />
  </protocolMapping>
  <!-- binding configuration - configures WSHttp binding for reliable sessions -->
  <bindings>
    <wsHttpBinding>
      <binding>
        <reliableSession enabled="true" />
      </binding>
    </wsHttpBinding>
  </bindings>
  <!--For debugging purposes set the includeExceptionDetailInFaults attribute to true-->
  <behaviors>
    <serviceBehaviors>
      <behavior>
        <serviceMetadata httpGetEnabled="True"/>
        <serviceDebug includeExceptionDetailInFaults="False" />
      </behavior>
    </serviceBehaviors>
  </behaviors>
  <serviceHostingEnvironment aspNetCompatibilityEnabled="true"
    multipleSiteBindingsEnabled="true" />

</system.serviceModel>

```

3.4. Sample code of the Account Service when processing sensitive data.

```

public class AccountService : IAccountService
{
    IStaffDao _staffDao;

    public AccountService()
    {
        _staffDao = new StaffDao();
    }

    public List<string> ResetPassword(CommonShared.Dto.ResetPasswordRequestDto resetPasswordRequest)
    {
        List<string> errors = new List<string>();

        var user = _staffDao.GetStaffByID(resetPasswordRequest.StaffID);
        if (user == null)
            errors.Add("Staff is not found");

        if (Encoding.UTF8.GetString(EncryptionService.EncryptPassword(user.Password))
!= Encoding.UTF8.GetString(resetPasswordRequest.OldPassword))
            errors.Add("Your password doesn't match");

        if (errors.Count > 0)
            return errors;

        user.Password = resetPasswordRequest.NewPassword;
        _staffDao.Update(user);

        return errors;
    }

    public StaffAccountDto ValidateLogin(string username, byte[] encryptedPassword)
    {
        var user= _staffDao.GetStaffByUserName(username);

        // check if user is found
        if (user == null)
            return null;

        // check if user is active
        if (!user.Active)
            return null;
    }
}

```

```

        // check for valid password
        var userPassword = EncryptionService.EncryptPassword(user.Password);
        if (Encoding.UTF8.GetString(encryptedPassword) != Encoding.UTF8.GetString(userPassword))
        {
            return null;
        }

        return Map(user);
    }
    .....
}

```

3.5. Sample code of the EmailService

```

public class EmailService
{
    public void SendMail(string from, string to, string subject, string body)
    {
        MailMessage mail = new MailMessage(from, to);

        SmtpClient client = new SmtpClient();
        client.Port = 587;
        client.EnableSsl = false;
        client.DeliveryMethod = SmtpDeliveryMethod.Network;
        client.UseDefaultCredentials = false;
        client.Timeout = 10000;
        client.Host = "mail.wnext.net.au";
        client.Credentials = new NetworkCredential("contact@wnext.net.au", "*****");

        mail.Subject = subject;
        mail.Body = body;

        client.Send(mail);
    }
}

```

3.6. Sample code of the EncryptionService

```

public class EncryptionService
{
    public static byte[] EncryptPassword(string password)
    {
        byte[] hashBytes = Encoding.UTF8.GetBytes(password);

        SHA1 sha1 = new SHA1CryptoServiceProvider();
        return sha1.ComputeHash(hashBytes);
    }

    public static string MD5Hash(string plainText)
    {
        byte[] hash = MD5.Create().ComputeHash(Encoding.UTF8.GetBytes(plainText));
        return Encoding.UTF8.GetString(hash);
    }
}

```

3.7. Sample code of the JobController api in WebAdminUI project

```

[Authorize]
[Route("api/Job")]
public class JobController : ApiController
{
    JobServiceClient client;

    public JobController()
    {
        client = ServiceFactory.CreateJobServiceClient();
    }
}

```

```

[Route("api/Job/{id}")]
public IHttpActionResult Get(int id)
{
    //return Ok();
    return Ok(client.GetJobById(id));
}

public IHttpActionResult Get([FromUri]string costCentre)
{
    return Ok(client.GetJobsByCostCentre(costCentre).ToList());
}

[HttpPost]
[Route("api/Job/{id}")]
public IHttpActionResult UpdateJob(JobDto job, int id)
{
    try
    {
        client.UpdateJob(job);
        // return a fresh entity
        return Ok(client.GetJobById(id));
    }
    catch (Exception ex)
    {
        return InternalServerError(ex);
    }
}

public IHttpActionResult Post([FromUri]int id,[FromBody]JobDto job)
{
    return Ok(job);
}

[HttpGet]
[Route("api/Job/{id}/Availability")]
public IHttpActionResult GetJobAvailability(int id)
{
    return Ok(client.GetJobAvailabilityById(id));
}
}

```

3.8. Sample code of angular js template and controller of job listing page

```

<h2>
  Job Listing <a href="#/create" class="btn btn-default">Create new</a>
</h2>

<div class="well well-sm">
  <div class="form form-inline">
    <select class="form-control"
      ng-model="listFilter.JobCostCentre"
      ng-options="cc as cc for cc in costCentreList">
      <option value="">Please select</option>
    </select>
    <a class="btn btn-primary" href="#/{{listFilter.JobCostCentre}}">Show job
      <i class="glyphicon glyphicon-refresh spin-animate"
        ng-show="FindWatcher('getJobsForCostCentre').loading"></i></a>

    <div class="pull-right">
      <button class="btn btn-primary" ng-click="sendEmail()">Send emails</button>
    </div>
  </div>
</div>

<table class="table wn-table table-clickable">
  <thead>
    <tr>
      <th style="width:50px"></th>
      <th>Cost centre</th>
      <th>BookID</th>
      <th>Site name</th>
      <th>Address</th>
      <th>Job date</th>
      <th>Job time</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td></td>
      <td></td>
      <td></td>
      <td></td>
      <td></td>
      <td></td>
      <td></td>
    </tr>
  </tbody>
</table>

```

```

        <th>Staff required</th>
        <th>Email sent</th>
        <th></th>
    </tr>
</thead>
<tbody>
    <tr ng-repeat="job in jobList" ng-class="{ 'selected': job.selected}">
        <td>
            <button class="btn" ng-class="job.selected?'btn-primary':'btn-default'"
                ng-click="job.selected=!job.selected"><i class="fa fa-check"
                    ng-class="{ 'invisible': !job.selected}"></i></button>

            </td>
            <td ng-click="job.selected=!job.selected" ng-bind="job.JobCostCentre"></td>
            <td ng-click="job.selected=!job.selected" ng-bind="job.BookID"></td>
            <td ng-click="job.selected=!job.selected" ng-bind="job.SiteName"></td>
            <td ng-click="job.selected=!job.selected" ng-bind="job.SiteAddress"></td>
            <td ng-click="job.selected=!job.selected" ng-bind="job.JobDate | date"></td>
            <td ng-click="job.selected=!job.selected" ng-bind="job.JobTime | time"></td>
            <td ng-click="job.selected=!job.selected"><span class="badge"
                ng-bind="job.StaffRequired"></span></td>
            <td>
                <i class="fa fa-check text-blue" ng-show="job.EmailSent"></i>
                <i class="fa fa-remove text-red" ng-show="!job.EmailSent"></i>
            </td>
            <td>
                <a class="btn btn-action btn-default"
                    href="#/edit/{{listFilter.JobCostCentre}}/{{job.BookID}}">Edit</a>
                <button class="btn btn-action">Send message</button>
            </td>
        </tr>
    </tbody>
</table>

```

jobListCtrl.js

```

(function () {
    'use strict';

    angular
        .module('tna.sap.controllers')
        .controller('jobListCtrl', ['$scope', '$rootScope', '$routeParams',
            '$location', 'jobService', 'userService',
            function (s, $rootScope, $routeParams,
                $location, jobService, userService) {
                $rootScope.AppTitle = "Job list";

                s.listFilter = {
                    JobCostCentre: undefined,
                    Keyword: undefined
                };

                function initData() {
                    userService.getCostCentre(true)
                        .then(function (result) {
                            s.costCentreList = result;
                            console.log(result);
                        }, function (error) {
                            console.log(error);
                        });
                }

                s.showJob = function () {
                    jobService.getJobsForCostCentre(s.listFilter.JobCostCentre)
                        .then(function (result) {
                            s.jobList = result;
                            console.log(result);
                        }, function (error) {
                            console.log(error);
                        });
                }

                if ($routeParams != null && typeof ($routeParams.cc) != 'undefined') {
                    s.listFilter.JobCostCentre = $routeParams.cc;
                    s.showJob();
                }

                s.sendEmail = function () {

```

```

        var jobList = s.jobList.filter(function (j) {
            return j.selected;
        });
        if (jobList.length == 0) {
            alert('Please select a job first');
            return;
        }

        $rootScope.SelectedJobs = jobList;

        $location.url('/sendEmail/' + s.listFilter.JobCostCentre);
    }
    initData();
}
})();
})();

```

4. Software Testing Results

4.1. Test send job alert process.

ID	Test case name	Step	Expected Output	Result
101	View list of jobs filtered by cost centre district	1. Login as an Administrator account. 2. Open the job list page 3. Select a cost centre district	A list of the jobs in that district should be displayed	Pass
102	View job detail	1. From the job list, click on Edit a job.	A job detail screen should be displayed with all job detailed information.	Pass
103	Edit and save job detail	1. Edit the detail of a job and click to Save button. 2. Go back to the job listing page 3. Open the previous job	A success message is displayed. The job detailed information should be keep the same when reopen	Pass
104	View staff list	1. Using an Administrator account, go to the Staff page. 2. Select a cost centre district and click Show Staff	A list of staff in the selected district should be displayed	Pass
105	View staff detail	1. On the staff listing page, click on a staff row to view staff detail	A staff profile detail page should be displayed with the Role and Cost Centre access permissions. The Cost Centre access should only be visible for Administrator role	Pass
106	Compose job alert email	1. From Job Listing page, select jobs that has not been sent 2. Click to Send email button 3. Select staff from the staff list displayed. 4. Click preview message 5. Click Send	2=> a staff listing page should be displayed to select staff 4=>a pre composed message is displayed for the selected jobs. 5=>an email is sent to each staff. The staffs will also be listed on the job detail page. The jobs are marked as Email Sent	Pass

Cost centre	BookID	Site name	Address	Job date	Job time	Staff required	Email sent
AU-VIC	1	MARCO ENGINEERING	139 BOURKE ST, MELBOURNE, VIC, 3000	2016-06-13	08:30	1	✓
AU-VIC	2	UPPER PLENTY PRIMARY SCHOOL	TOWONG ROAD, CORRYONG, VIC, 3707	2016-06-13	15:00	9	✓
AU-VIC	8	HAMLYN BANKS PRIMARY SCHOOL	15 ALBERT STREET, BRUNSWICK EAST, VIC, 3057	2016-06-13	07:30	6	✓
AU-VIC	14	SOLECTRON	JANET STREET, BLACKBURN, VIC, 3130	2016-06-14	16:00	6	✓
AU-VIC	16	FARM IMP. TRACTOR & MOTOR CO	BOURCHIER STREET, SHEPPARTON, VIC, 3630	2016-06-14	16:00	5	✗
AU-VIC	20	HORSHAM COLLEGE	WILMOT RD, SHEPPARTON, VIC, 3630	2016-06-14	16:00	6	✗
AU-VIC	24	ACCUWEIGH QLD	17-23 BRYANT STREET, PADSTOW, NSW, 2211	2016-06-15	16:00	0	✗
AU-VIC	28	STOCKTON IGA	677 SPRINGVALE ROAD, MULGRAVE, VIC, 3170	2016-06-15	11:30	8	✗
AU-VIC	40	AUSTRALIAN AUTOMOTIVE AIR	WAMON AVENUE, LEETON, NSW, 2705	2016-06-16	08:30	3	✗
AU-VIC	42	ASSETS - TRAINING	707 SKOKIE BOULEVARD, NORTHBROOK, NULL, 60062	2016-06-17	14:00	5	✗
AU-VIC	43	SACRED HEART SENIOR COLLEGE	LAWRENCE ST, WOODONGA, VIC, 3690	2016-06-17	11:30	7	✗

102. Job detail page

Job Detail

Job detail

Job Status: Booked

Job cost centre: AU-VIC

Supervisor: JAMES BRAZIER

Site name: HORSHAM COLLEGE

Site address: WILMOT RD, SHEPPARTON, VIC, 3630

Job date: 2016-06-14T00:00

Staff required: 6

Job details: Please arrive 15 minutes before the schedule time

Availability

Search staff

Legend:

- Not yet answer
- Not available
- Available

© 2016 - Staff Allocate Plus - Nguyen Tran Nguyen (6586880)

103. Save success message

Job Detail

detail

Job Status: Booked

Job cost centre: AU-VIC

Supervisor: JAMES BRAZIER

Site name: HORSHAM COLLEGE

Site address: WILMOT RD, SHEPPARTON, VIC, 3630

Job date: 2016-06-14T00:00

Staff required: 6

Job details: Please arrive 15 minutes before the schedule time

Availability

Search staff

Legend:

- Not yet answer
- Not available
- Available

© 2016 - Staff Allocate Plus - Nguyen Tran Nguyen (6586880)

localhost:52617 says: Save success

OK

104. Staff listing page

Cost centre	Name	Email	Mobile	Active
AU-VIC	YANG HODGSON	6586880@student.swin.edu.au	0456 759 809	✓
AU-VIC	BARRY BARRERA	6586880@student.swin.edu.au	0456 751 004	✓
AU-VIC	ELISE LEGG	6586880@student.swin.edu.au	0456 281 684	✓
AU-VIC	ELIZABETH DAY	6586880@student.swin.edu.au	0456 791 708	✓
AU-VIC	ERI HARK	6586880@student.swin.edu.au	0456 242 070	✓
AU-VIC	JAMES SIMPSON	6586880@student.swin.edu.au	0456 734 021	✓
AU-VIC	JAMES BRAZIER	6586880@student.swin.edu.au	0456 926 813	✓
AU-VIC	MALCOLM LANGLEY	6586880@student.swin.edu.au	0456 296 242	✓
AU-VIC	DANIELLE WATTS	6586880@student.swin.edu.au	0456 748 586	✓
AU-VIC	TAMSIN HARK	6586880@student.swin.edu.au	0456 106 768	✓
AU-VIC	MARIA MERCECA	6586880@student.swin.edu.au	0456 089 889	✓

105. Staff detail page

YANG HODGSON

Send message

Staff profile

Staff cost centre: AU-VIC ☒ Active

Name: First name: YANG Surname: HODGSON

Mobile: 0456 759 809

Email: 6586880@student.swin.edu.au

Access Account: User Name: YH983396

Role access permission

Role name	Is in role	
Administrator	✗	<input type="button" value="Grant role"/>
Employee	✓	<input type="button" value="Revoke role"/>

© 2016 - Staff Allocate Plus - Nguyen Tran Nguyen (6586880)

Administrator profile

Nguyen Tran

Send message

Staff profile

Staff cost centre: AU-VIC ☒ Active

Name: First name: Nguyen Surname: Tran

Mobile: 0432431067

Email: nguyenn86@gmail.com

Access Account: User Name: nguyenn86

Role access permission

Role name	Is in role	
Administrator	✓	<input type="button" value="Revoke role"/>
Employee	✓	<input type="button" value="Revoke role"/>

Cost centre access permission

Cost centre	Can access	
AU	✗	<input type="button" value="Grant access"/>
AU-NSW	✓	<input type="button" value="Revoke access"/>
AU-QLD	✗	<input type="button" value="Grant access"/>
AU-SAT	✗	<input type="button" value="Grant access"/>
AU-VIC	✓	<input type="button" value="Revoke access"/>
AU-WA	✗	<input type="button" value="Grant access"/>

106. Compose job alert email

Select jobs

Cost centre	BookID	Site name	Address	Job date	Job time	Staff required	Email sent
AU-VIC	1	MARCO ENGINEERING	139 BOURKE ST, MELBOURNE, VIC, 3000	2016-06-13	08:30	1	✓
AU-VIC	2	UPPER PLENTY PRIMARY SCHOOL	TOWONG ROAD, CORRYONG, VIC, 3707	2016-06-13	15:00	1	✓
AU-VIC	8	HAMILYN BANKS PRIMARY SCHOOL	15 ALBERT STREET, BRUNSWICK EAST, VIC, 3057	2016-06-13	07:30	1	✓
AU-VIC	14	SOLELECTRON	JANET STREET, BLACKBURN, VIC, 3130	2016-06-14	16:00	1	✓
AU-VIC	16	FARU IMP. TRACTOR & MOTOR CO	BOURCHIER STREET, SHEPPARTON, VIC, 3630	2016-06-14	16:00	1	✗
AU-VIC	20	HORSHAM COLLEGE	WILMOT RD, SHEPPARTON, VIC, 3630	2016-06-14	16:00	1	✗
AU-VIC	24	ACCUWEIGH QLD	17-23 BRYANT STREET, PADSTOW, NSW, 2211	2016-06-15	16:00	1	✗
AU-VIC	28	STOCKTON IGA	677 SPRINGVALE ROAD, MULGRAVE, VIC, 3170	2016-06-15	11:30	1	✗
AU-VIC	40	AUSTRALIAN AUTOMOTIVE AIR	WAMOOON AVENUE, LEETON, NSW, 2705	2016-06-16	08:30	1	✗
AU-VIC	42	ASSETS - TRAINING	707 SKOKIE BOULEVARD, NORTHBROOK, NULL, 80062	2016-06-17	14:00	1	✗
AU-VIC	43	SACRED HEART SENIOR COLLEGE	LAWRENCE ST, WOODONGA, VIC, 3690	2016-06-17	11:30	1	✗

Administrator profile

Nguyen Tran

Send message

Staff profile

Staff cost centre: AU-VIC ☒ Active ☒

Name: First name: Nguyen Surname: Tran

Mobile: 0432431067

Email: nguyennb@gmail.com

Access Account: User Name: nguyennb Change password

Save

Role access permission

Role name	Is in role	Revoke role
Administrator	✓	Revoke role
Employee	✓	Revoke role

Cost centre access permission

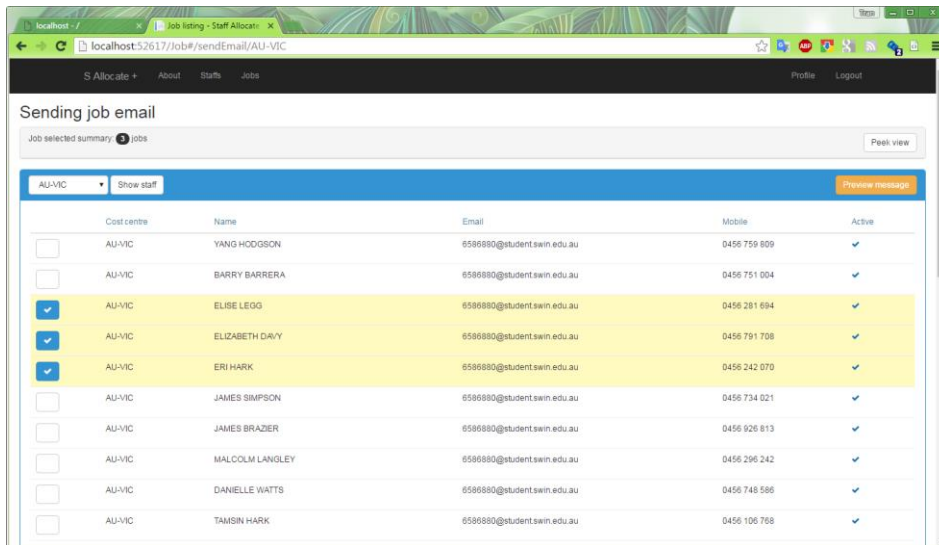
Cost centre	Can access	Grand access
AU	✗	Grand access
AU-NSW	✓	Revoke access
AU-QLD	✗	Grand access
AU-SAT	✗	Grand access
AU-VIC	✓	Revoke access
AU-WA	✗	Grand access

106. Compose job alert email

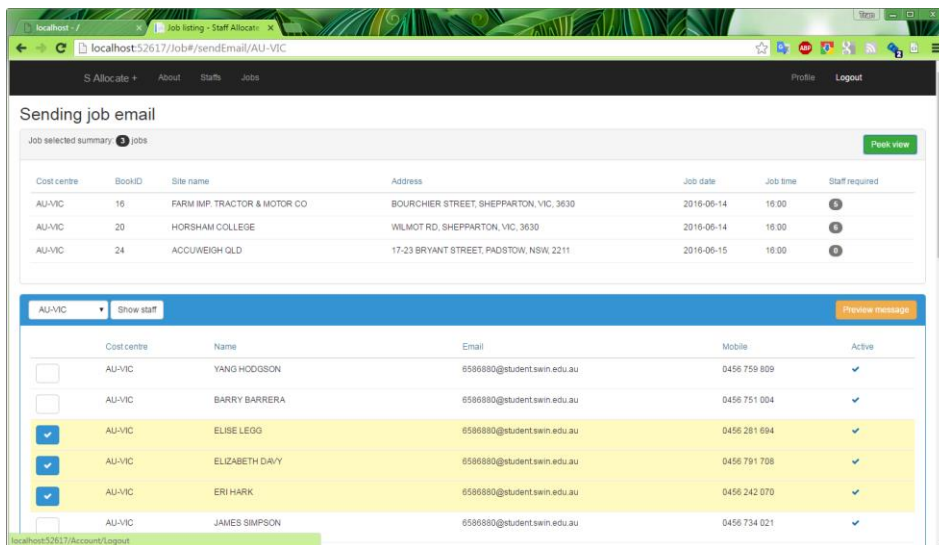
Select jobs

Cost centre	BookID	Site name	Address	Job date	Job time	Staff required	Email sent
AU-VIC	1	MARCO ENGINEERING	139 BOURKE ST, MELBOURNE, VIC, 3000	2016-06-13	08:30	1	✓
AU-VIC	2	UPPER PLENTY PRIMARY SCHOOL	TOWONG ROAD, CORRYONG, VIC, 3707	2016-06-13	15:00	1	✓
AU-VIC	8	HAMILYN BANKS PRIMARY SCHOOL	15 ALBERT STREET, BRUNSWICK EAST, VIC, 3057	2016-06-13	07:30	1	✓
AU-VIC	14	SOLELECTRON	JANET STREET, BLACKBURN, VIC, 3130	2016-06-14	16:00	1	✓
AU-VIC	16	FARU IMP. TRACTOR & MOTOR CO	BOURCHIER STREET, SHEPPARTON, VIC, 3630	2016-06-14	16:00	1	✗
AU-VIC	20	HORSHAM COLLEGE	WILMOT RD, SHEPPARTON, VIC, 3630	2016-06-14	16:00	1	✗
AU-VIC	24	ACCUWEIGH QLD	17-23 BRYANT STREET, PADSTOW, NSW, 2211	2016-06-15	16:00	1	✗
AU-VIC	28	STOCKTON IGA	677 SPRINGVALE ROAD, MULGRAVE, VIC, 3170	2016-06-15	11:30	1	✗
AU-VIC	40	AUSTRALIAN AUTOMOTIVE AIR	WAMOOON AVENUE, LEETON, NSW, 2705	2016-06-16	08:30	1	✗
AU-VIC	42	ASSETS - TRAINING	707 SKOKIE BOULEVARD, NORTHBROOK, NULL, 80062	2016-06-17	14:00	1	✗
AU-VIC	43	SACRED HEART SENIOR COLLEGE	LAWRENCE ST, WOODONGA, VIC, 3690	2016-06-17	11:30	1	✗

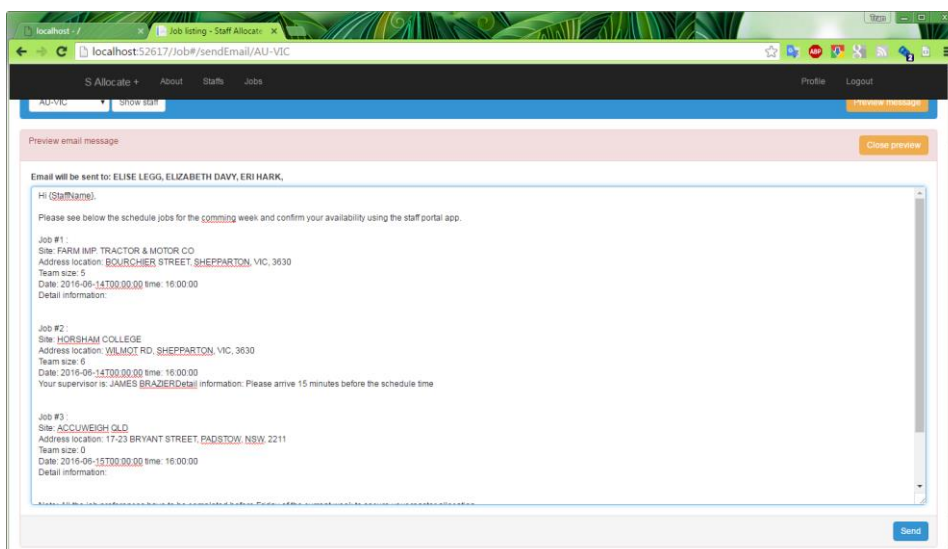
Select staffs



Job peek view



Compose job alert email



Jobs are marked as email sent

Cost centre	BookID	Site name	Address	Job date	Job time	Staff required	Email sent
AU-VIC	1	MARCO ENGINEERING	139 BOURKE ST, MELBOURNE, VIC, 3000	2016-06-13	08:30	1	✓
AU-VIC	2	UPPER PLENTY PRIMARY SCHOOL	TOWONG ROAD, CORRYONG, VIC, 3707	2016-06-13	15:00	3	✓
AU-VIC	8	HAMILYN BANKS PRIMARY SCHOOL	15 ALBERT STREET, BRUNSWICK EAST, VIC, 3057	2016-06-13	07:30	3	✓
AU-VIC	14	SOLETRON	JANET STREET, BLACKBURN, VIC, 3130	2016-06-14	16:00	3	✓
AU-VIC	16	FARM IMP. TRACTOR & MOTOR CO	BOURCHIER STREET, SHEPPARTON, VIC, 3630	2016-06-14	16:00	3	✓
AU-VIC	20	HORSHAM COLLEGE	WILMOT RD, SHEPPARTON, VIC, 3630	2016-06-14	16:00	3	✓
AU-VIC	24	ACCUWEIGH QLD	17-23 BRYANT STREET, PADSTOW, NSW, 2211	2016-06-15	16:00	3	✓
AU-VIC	28	STOCKTON ISA	677 SPRINGVALE ROAD, MULGRAVE, VIC, 3179	2016-06-15	11:30	3	✗
AU-VIC	40	AUSTRALIAN AUTOMOTIVE AIR	WAMOOON AVENUE, LEETON, NSW, 2795	2016-06-16	08:30	3	✗
AU-VIC	42	ASSETS - TRAINING	707 SKOKIE BOULEVARD, NORTHBROOK, NULL, 60062	2016-06-17	14:00	3	✗
AU-VIC	43	SACRED HEART SENIOR COLLEGE	LAWRENCE ST, WOODONGA, VIC, 3690	2016-06-17	11:30	7	✗

Staff allocations are displayed on the job detail page

Job Detail

Job detail

Job Status: Booked

Job cost centre: AU-VIC

Supervisor: JAMES BRAZIER

Site name: HORSHAM COLLEGE

Site address: WILMOT RD, SHEPPARTON, VIC, 3630

Job date: 2016-06-14T00:00:00 to 16:00:00

Staff required: 3

Job details: Please arrive 15 minutes before the schedule time

Availability

All Search staff

Staff name	Is available	Is confirmed
ELISE LEGG	✓	
ELIZABETH DAIVY	✗	
ERI HARK	✗	

Legend:

- Not yet answer
- ✗ Not available
- ✓ Available

© 2016 - Staff Allocate Plus - Nguyen Tran Nguyen (6586860)

Staff receives email to their email address:

Job schedule for AU-VIC

contact@wnext.net.au via student.swin.edu.au 13:41 (3 minutes ago)

to TRAN

Hi ELISE LEGG,

Please see below the schedule jobs for the coming week and confirm your availability using the staff portal app.

Job #1 :
 Site: FARM IMP. TRACTOR & MOTOR CO
 Address location: BOURCHIER STREET, SHEPPARTON, VIC, 3630
 Team size: 5
 Date: 2016-06-14T00:00:00 time: 16:00:00
 Detail information:

Job #2 :
 Site: HORSHAM COLLEGE
 Address location: WILMOT RD, SHEPPARTON, VIC, 3630
 Team size: 6
 Date: 2016-06-14T00:00:00 time: 16:00:00
 Your supervisor is: JAMES BRAZIER
 Detail information: Please arrive 15 minutes before the schedule time

Job #3 :
 Site: ACCUWEIGH QLD
 Address location: 17-23 BRYANT STREET, PADSTOW, NSW, 2211
 Team size: 0
 Date: 2016-06-15T00:00:00 time: 16:00:00
 Detail information:

Note: All the job preferences have to be completed before Friday of the current week to secure your roster allocation.

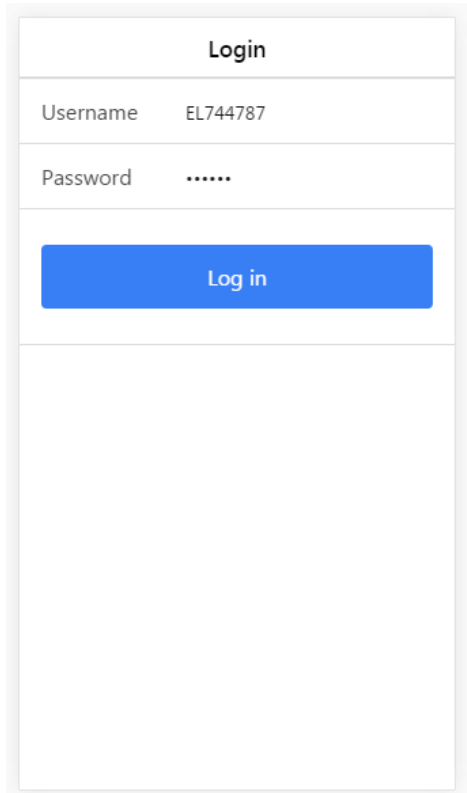
Thank you,

Human Resource VIC Team

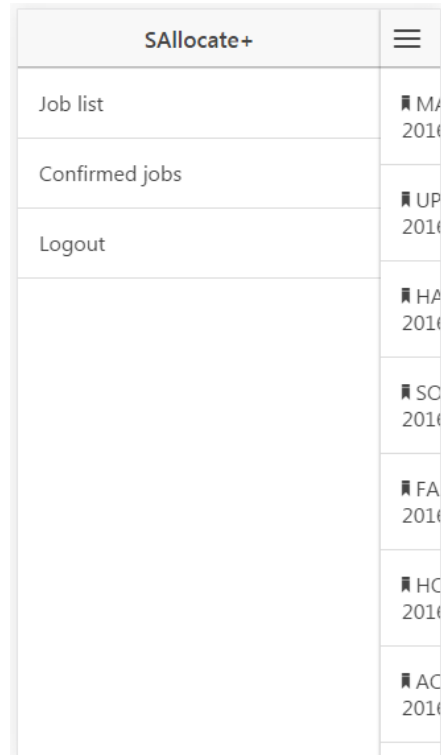
4.2. Test job available selection from staff

ID	Test case name	Step	Expected Output	Result
201	Login to the mobile app	1. Open the SAllocate+ mobile app 2. Enter the staff login information	The login form should be hidden and user can open the navigation bar	Pass
202	View current job alerts	1. Navigate to the Job List screen	A list of current job alerts is displayed	Pass
203	View job alert detail	1. Select a job from the job list above	The job detail should be displayed with.	Pass
204	Set job availability	1. From the job detail screen above, choose an option of "Available"	The screen navigate to job list page and the job selected before is removed from the list. From the web admin, the staff availability is displayed on the job detail page	Pass

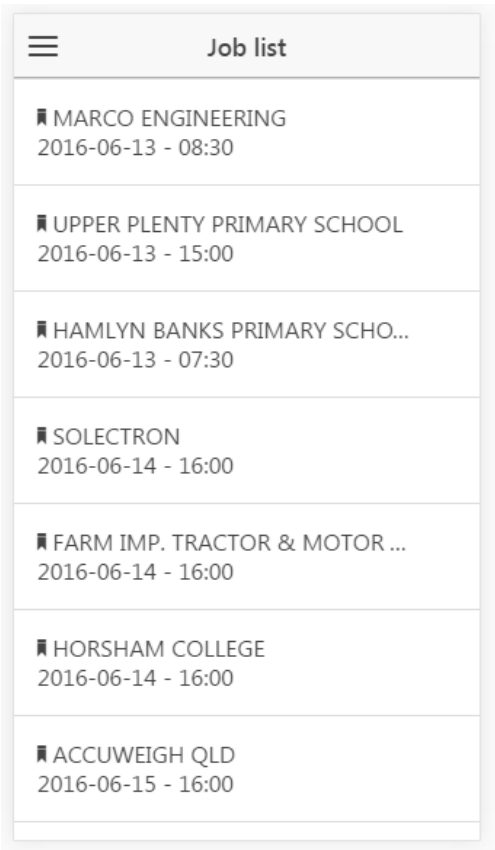
201. Login to the mobile app



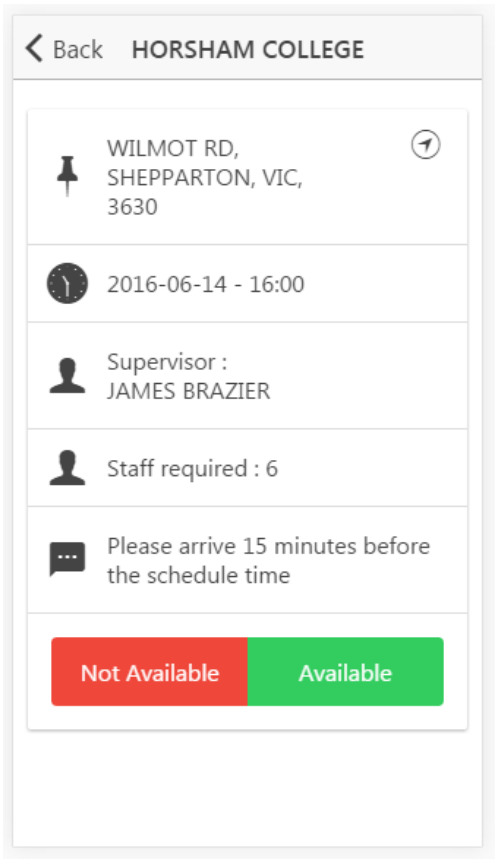
Login form



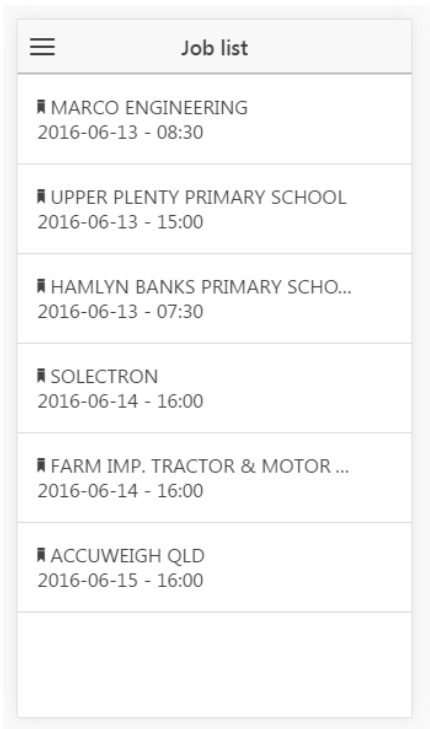
App main navigation menu:



202. Job list is displayed for the login staff



203. View a job detail



204. Job is removed from the job list after set as Available

4.3. Test review staff availability for a job

ID	Test case name	Step	Expected Output	Result
301	View staff availability on the job	1. After a staff selected his availability for a job, open this job detail on the Admin backend	The staff availability should be displayed on the Availability table	Pass

301. Staff availability should be displayed on the job detail page.

The screenshot shows the 'Job Detail' page. On the left, there are fields for Job Status (Booked), Job cost centre (AU-VIC), Supervisor (JAMES BRADIER), Site name (HORSHAM COLLEGE), Site address (WILMOT RD, SHEPPARTON, VIC, 3630), Job date (2016-06-14 00:00), and Staff required (5). On the right, the 'Availability' section shows a table with columns 'Staff name', 'Is available', and 'Is confirmed'. The table lists three staff members: ELISE LEGG (green checkmark in 'Is available'), ELIZABETH DAVY (grey dash), and ERI HARK (grey dash). A red arrow points to the green checkmark for ELISE LEGG. A legend at the bottom indicates: grey dash for 'Not yet answer', red X for 'Not available', and green checkmark for 'Available'.

5. References

- Dijkstra, T 2015, 'Getting Started with Entity Framework 6 Code First using MVC 5', Asp.Net Microsoft, viewed on 10 Jun 2016, <<https://www.asp.net/mvc/overview/getting-started/getting-started-with-ef-using-mvc/creating-an-entity-framework-data-model-for-an-asp-net-mvc-application>>
- 2016, 'Ionic framework', Ionic, viewed on 8 Jun 2016, <<http://ionicframework.com/>>
- 2016, 'ASP.NET Web Api', ASP.NET, Microsoft, viewed on 11 Jun 2016, <<http://www.asp.net/web-api>>
- BitBucket full Source code: <<https://bitbucket.org/werynguyen/swin-school/src/1f8bce60843b9c8d8095ce612aa4220cc6b8a98c/?at=DProject>>