System and Software Architecture Description (SSAD)

City of Los Angeles Personnel Department Mobile Applications

Team 02

Anushree Sridhar - Software Architect

Shreya Kamani - Project Manager

Divya Reddy - Requirements Engineer

Pattra Thongprasert - Operation Concept Engineer

Abhishek Trigunayat - Prototyper

Travis Jones - Feasibility Analyst

William Everton - IIV and V

USC-CSSE

Version History

Date	Author	Version	Changes made	Rationale
10/16/13	Anushree	1.0	Added: • System context diagram • Use case diagram and descriptions • Artifacts and information diagram	Initial technology independent draft of SSAD for FC package
10/23/13	Anushree	1.1	• Modified diagrams	• Modifications made based on the FC ARB
10/26/13	Anushree	1.2	• Modified use case descriptions (post and pre conditions, flow)	• Modifications made on the basis of “SSAD in DC” lecture to the use case descriptions
12/02/13	Anushree	1.3	Added: • Hardware component diagram • Software component diagram • Deployment diagram • Class diagrams • Sequence diagrams	• Completed SSAD according to the draft DC package specifications
12/09/13	Anushree	1.4	Use cases were updated	• Updated use cases with feedback from FC package

Table of Contents

Table of Tables

Note: Process Description tables can be found in pages: 4-12

Table of Figures

1. Introduction

1.1. Purpose of the SSAD

The purpose of the System and Software Architecture Description
is to describe what the proposed system will do, how it will do those functions, who will be using it, what the flow of events is and where it will be deployed and operate. It shows how the system interacts with the external environment and how the various components of the system interact. It contains a complete description of the design of the proposed system and its working.

1.2. Status of the SSAD

The SSAD is being submitted for the Development Commitment Review (FCR). It includes the technology specific design of the system being developed.

2.System Analysis

2.1. System Analysis Overview

The primary purpose of the City of Los Angles Personnel Department Mobile Application is to provide job seekers a means of subscribing for notifications about openings in jobs in the city of Los Angeles, which they are interested in. The application enables job seekers to view a list of currently open jobs, search for a job they are interested in and subscribe to receive notifications. It then sends a notification to the job seeker when a job they have subscribed to opens up.

2.1.1. System Context

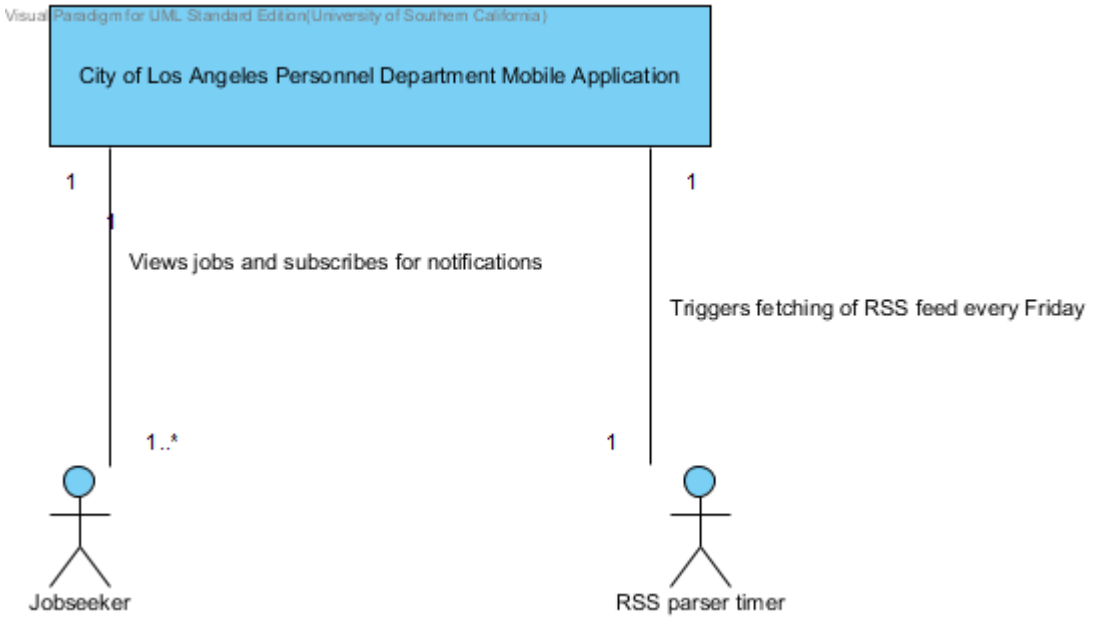

Figure 1: System Context Diagram

ID	Name	Related Use Cases
AC01	Jobseeker	Search all existing jobs Create / update user profile View job descriptions View open jobs Subscribe for notifications Receive notifications when job opens
AC02	RSS Parser timer	Process Neogov RSS feed

Table 1: Actors Summary

2.1.2. Artifacts and information

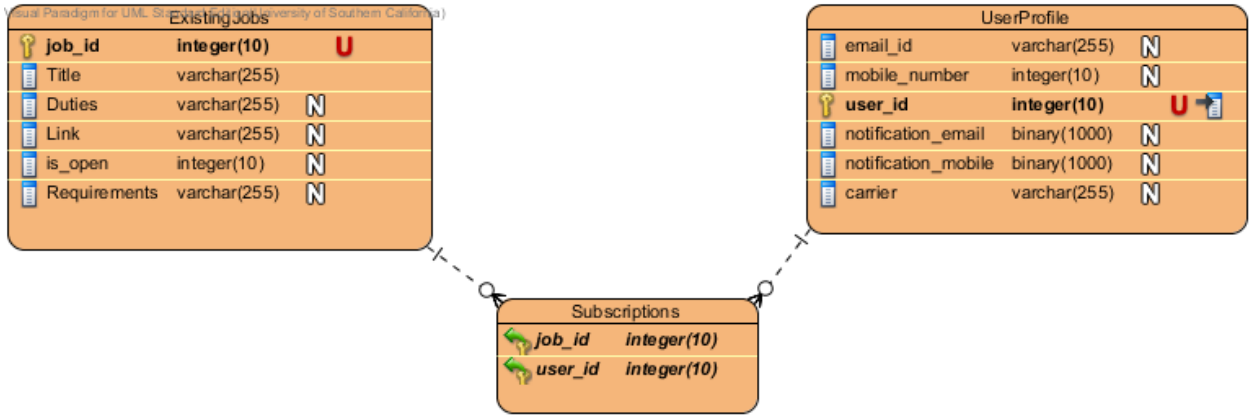

Figure 2: Artifacts and Information Diagram

Artifact	Purpose
ExistingJobs	Table of all the currently existing jobs in the City of LA. Will be mirrored from the existing database.
UserProfile	Table containing the user profile as saved by the user.
Subscriptions	This table has a list of users and jobs which the user has subscribed for notifications to.

Table 2: Artifacts and Information Summary

2.1.3. Behavior

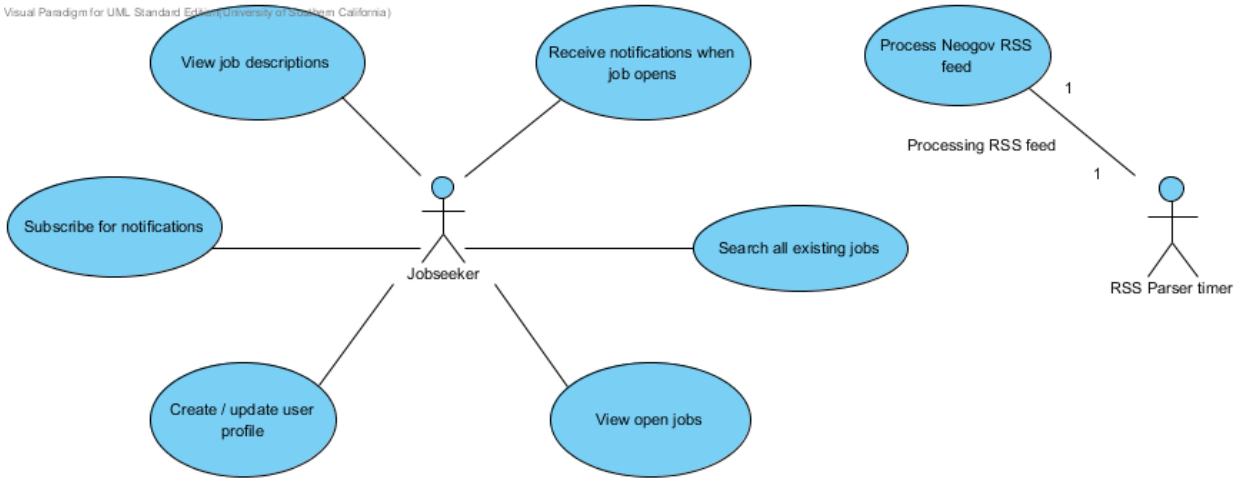

Figure 3: Process Diagram

View open jobs

Use Case Descriptions

Main			
Super Use Case	None		
Brief Description	The user can view a listing of currently open jobs.		
Preconditions	UC07 is performed.		
Post-conditions	User will view a listing of all currently open jobs.		
Flow of Events		**Actor Input**	**System Response**
	1	User clicks "View open jobs" on the home screen.	
	2		UI sends request to the server.
	3		Server parses RSS feed.
	4		Server sends a list of the open jobs to the UI.
	5		UI displays the open jobs titles on a separate screen.

Alternative			
Brief Description	If UI is unable to fetch the details from the server.		
Preconditions	UC08 is performed.		
Post-conditions	None		
Flow of Events		**Actor Input**	**System Response**
	1	User clicks "View open jobs" on the home screen.	
	2		UI sends request to the server.
	3		UI does not receive a reply after a suitable time out.
	4		UI resends the request (3 times).
	5		If the UI still doesn't receive a reply it displays an error message and encourages the user to retry at a later time.

Process Neogov RSS feed

Use Case Descriptions

Main			
Super Use Case	None		
Brief Description	The server processes the Neogov RSS feed for the open jobs listing every friday.		
Preconditions	None		
Post-conditions	Open jobs get updated, notifications are sent accordingly		
Flow of Events		**Actor Input**	**System Response**
	1	Function to retrieve RSS feed from Neogov is triggered every Friday at 0001 in the application server.	
	2		Neogov RSS feed is fetched and stored as an XML file on the server.
	3		The XML file is parsed and the open job ids are saved.
	4		The system triggers the notification function to send required notifications.

Exceptional flow			
Brief Description	Server is unable to retrieve the RSS feed.		
Preconditions	It is friday 0001 PST and server triggers RSS feed retrieval		
Post-conditions	The server either retrieves the feed or keeps trying every 30 minutes.		
Flow of Events		**Actor Input**	**System Response**
	1	Function to retrieve RSS feed from Neogov is triggered.	
	2		Server tries to retrieve the RSS feed from the Neogov server.
	3		Server is unable to retrieve the feed. Retries 3 times.
	4		Server is still unable to retrieve the feed. It retries

			again every 30 minutes until it is able to retrieve the feed.

Receive notifications when job opens

Use Case Descriptions

Main			
Super Use Case	None		
Brief Description	The user will receive a notification either through email or text message when a job he has subscribed for opens.		
Preconditions	UC07 and UC05		
Post-conditions	User receives a notification through text or email.		
Flow of Events		**Actor Input**	**System Response**
	1	Job seeker subscribed for notification to a job that just opened on the View Job Description page.	
	2		RSS feed is downloaded and parsed. A list of open jobs is obtained.
	3		Server searches through the user profile database for users who have requested to be updated about this job.
	4		If such users are found, server checks the method of notification selected.
	5		Server sends an email or text message, whichever is requested, with job id, title and a link to apply for the job.

View job descriptions

Use Case Descriptions

Main			
Super Use Case	None		
Brief Description	The user should be able to view a brief job description and use a link to connect to the job description page on the web site.		
Preconditions	The user performs a search and selects a given job from the search results. (UC03 and UC04)		
Post-conditions	The user can view a brief description of the job he is interested in.		
Flow of Events For a currently open job		**Actor Input**	**System Response**
	1	User clicks on job from the jobs search result.	
	2		UI sends request to the server.
	3		Server responds with the requested job details from the XML saved from the RSS feed.
	4		UI displays these details in the job description page.
	5		Job description page contains job id, title, salary details, location of job, last date of application, category of job, a brief description of the job role and a link to apply for the job. These details are available in the Neogov RSS feed.
	6	User clicks "Apply now" link.	
	7		Redirects to the existing web site from where the candidate can apply for the post.
Flow of Events For unopen jobs		**Actor Input**	**System Response**
	1	User clicks on job from the jobs search result.	
	2		UI sends request to the server.
	3		Server responds with the

			requested job details from the job listing database.
	4		UI displays these details in the job description page.
	5		Job description page contains job id, title, a brief description of the job role and a link to the job details in the existing web site. It also contains a subscribe button.
	6	User clicks the web site link.	
	7		Redirects to the existing web site from where the candidate can view further details about the post.

Search all existing jobs

Use Case Descriptions

Main			
Super Use Case	None		
Brief Description	The user can search through a database of all existing (including open ones) jobs in the city of LA. He can use job ID or keywords to perform the search.		
Preconditions	Job listings database is populated.		
Post-conditions	User will view jobs matching his search criteria (if any).		
Flow of Events		**Actor Input**	**System Response**
	1	User clicks on "Search for Jobs" in the home page.	
	2		UI redirects to the search page.
	3	User enters the search term (job id or keyword) and presses enter.	
	4		UI validates if search term is non-empty.
	5		UI sends the search term to the server.
	6		The server searches the job_id and title of the job listings database.
	7		The server sends back a list of matching jobs.
	8		UI displays the jobs in the search result age. If no job is returned it displays "No matching jobs".

Create / update user profile

Use Case Descriptions

Main	
Super Use Case	None
Brief Description	The user can create a profile containing the email id and / or the

	mobile number he would like to use to receive notifications. Optional.
Preconditions	User is on home page
Post-conditions	If an email id or mobile number is added by the user, it should be automatically used to send notifications when the user subscribes for a job

Flow of Events

	Actor Input	System Response
1	Clicks "User Profile" on the home page.	
2		UI takes user to the Profile page.
3	User inputs email id and / or mobile number. He can also input notification preferences.	
4	User clicks save.	
5		UI checks if email id and number are valid.
6		If details are valid. UI sends a copy of the details entered to the server in an encrypted fashion.
7		Server checks if email id and mobile number are already present in the User database.
8		If details are not present it inputs them in the User database. If the same email id or mobile number is present then it updates the database accordingly.
9		Server sends a message saying details were saved.
10		UI saves a copy of the details on the phone.
11		UI displays a success message. Then redirects to the home page.

Alternative	
Brief Description	If invalid email or mobile numbers are entered
Preconditions	User inputs invalid details
Post-conditions	None

Flow of Events		Actor Input	System Response
	1	Clicks "User Profile" on the home page.	
	2		UI takes user to the Profile page.
	3	User inputs email id and / or mobile number. He can also input notification preferences.	
	4	User clicks save.	
	5		UI checks if email id and number are valid.
	6		If invalid, displays a message "The details inputted are invalid. Please re-enter valid detail."
	7		Displays the Profile page.

Subscribe for notifications

Use Case Descriptions

Main			
Super Use Case	None		
Brief Description	The user subscribes for a notification when a particular job opens either through email or text messages.		
Preconditions	User is viewing the job description page of an existing but unopen job.		
Post-conditions	User will receive a notification when the job opens.		
Flow of Events		**Actor Input**	**System Response**
	1	User clicks subscribe on the "Job Description" page.	
	2		UI redirects to the subscription page.
	3		If user profile exists, UI inputs the email id and the mobile number in the form. Else, it provides a form to input these values.
	4	User chooses whether they	

		want email or text message notification.	
	5	User enters profile details if needed and clicks on subscribe.	
	6		UI sends job id, subscription method and user details to the server in an encrypted fashion.
	7		Server saves details in the user profile table and sends success message.
	8		UI displays success message. Redirects to job description page.

Alternative			
Brief Description	UI receives a failure message or no message at all.		
Preconditions	UI sends profile and subscription details to the system.		
Post-conditions			
Flow of Events		**Actor Input**	**System Response**
	1	Steps 1-5 as in main flow.	
	2		UI sends job id, subscription method and user details to the server in an encrypted fashion.
	3		UI doesn't receive a message till time-out or receives an error message from the server.
	4		UI displays error message and suggests that the user retry the operation at a later time.

2.1.4 Modes of Operation

The application has a single mode of operation. The only user is the job-seeker

2.2 System Analysis Rationale

The proposed system considers two types of jobs:

Existing Jobs: These are all the jobs that already exist in the City of Los Angeles. It consists of both the jobs for which openings are currently open as well as jobs which do not have any openings at present. A job-seeker can search through this list of jobs using keywords and subscribe for jobs which are not currently open.

Open Jobs: This is a subset of the Existing Jobs. It includes jobs which currently have a vacancy and for which the City of Los Angeles is accepting applications.

3.Technology Independent Model

This section is left blank as the technology specific design of the system contains the relevant details. Please refer to section 4 below.

4. Technology-Specific System Design

4.1. Design Overview

4.1.1. System Structure

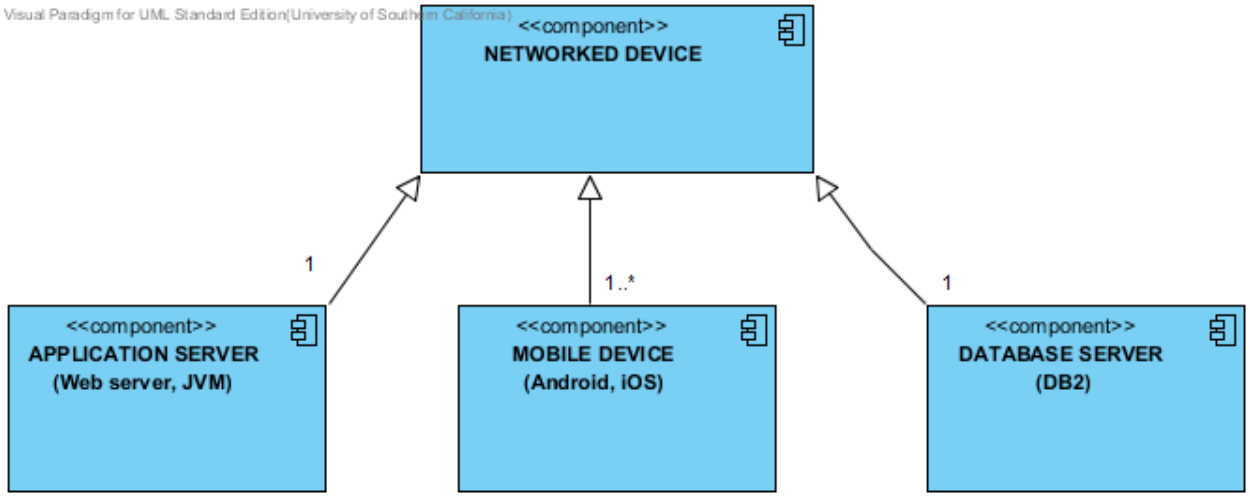

Figure 4: Hardware Component Diagram

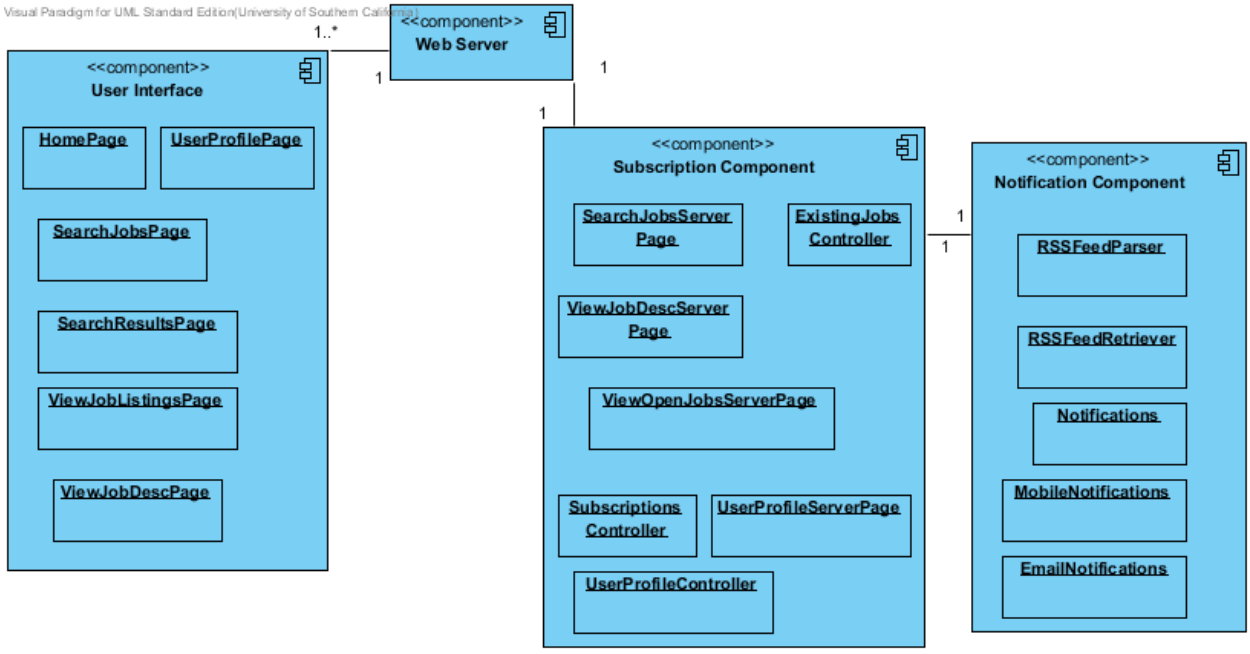

Figure 5: Software Component Class Diagram

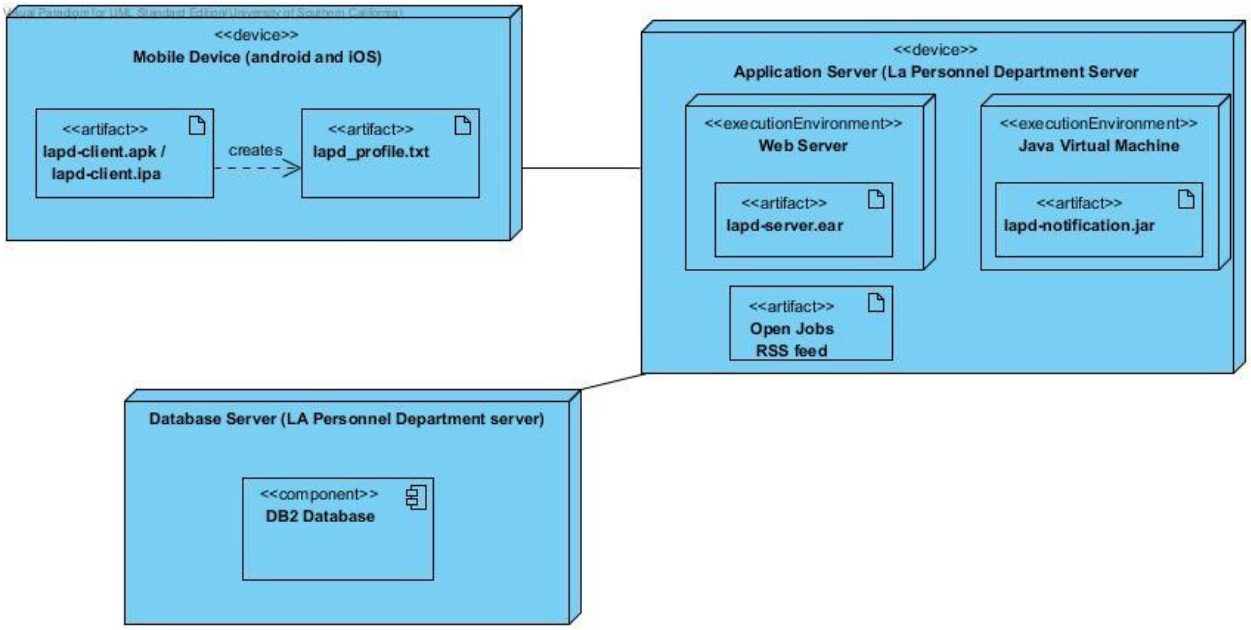

Figure 6: Deployment Diagram

Hardware Component	Description
Application Server	Web server with JVM which will contain the server side code for our system. Code will be in Java and JSP.
Mobile Device	Android or iOS based mobile device which will contain the client side native application for our system. Code in HTML, CSS and Javascript.
Database Server	DB2 server provided by the client containing the tables used by our system.

Table 3: Hardware Component Description

Software Component	Description
User Interface	The client side application with which the user will interact to view and subscribe for notifications.
Subscription Component	The server side component which will handle saving the user profile, searching for jobs and subscription for notifications.
Notification Component	The server side component which will handle fetching and parsing the open jobs feed and sending the required notifications.

Table 4: Software Component Description

4.1.2 Design Classes

Figure 7: Design Class Diagram

Class	Type	Description
SubscriptionsController	Control	This class updates and retrieves data from the subscriptions table.
UserProfileController	Control	This class updates and retrieves data from the UserProfile table.
UserProfileServerPage	Control	This class passes data from the UserProfilePage to the UserProfile
ExistingJobsController	Control	This class updates and retrieves data from the ExistingJobs table.
ViewJobDescServerPage	Control	This class interacts with ExistingJobsController and RSSParser to display job details as applicable.
SearchJobsServerPage	Control	This class interacts with the ExistingJobsController to retrieve a list of jobs meeting a particular search criteria.
ViewOpenJobsServerPage	Control	This class interacts with the RSSParser class to retrieve a list of currently open jobs to display to the user.
RSSFeedParser	Control	This class parses the RSS feed and checks which jobs are open. It then triggers notifications if required.
RSSFeedRetriever	Control	This class downloads and saves the RSS feed with information about the currently open jobs.
Notifications	Control	Abstract class to send notifications.
EmailNotifications	Control	Class to send notification be email.
MobileNotifications	Control	Class to send notification by mobile.

Table 5: Design Class Description

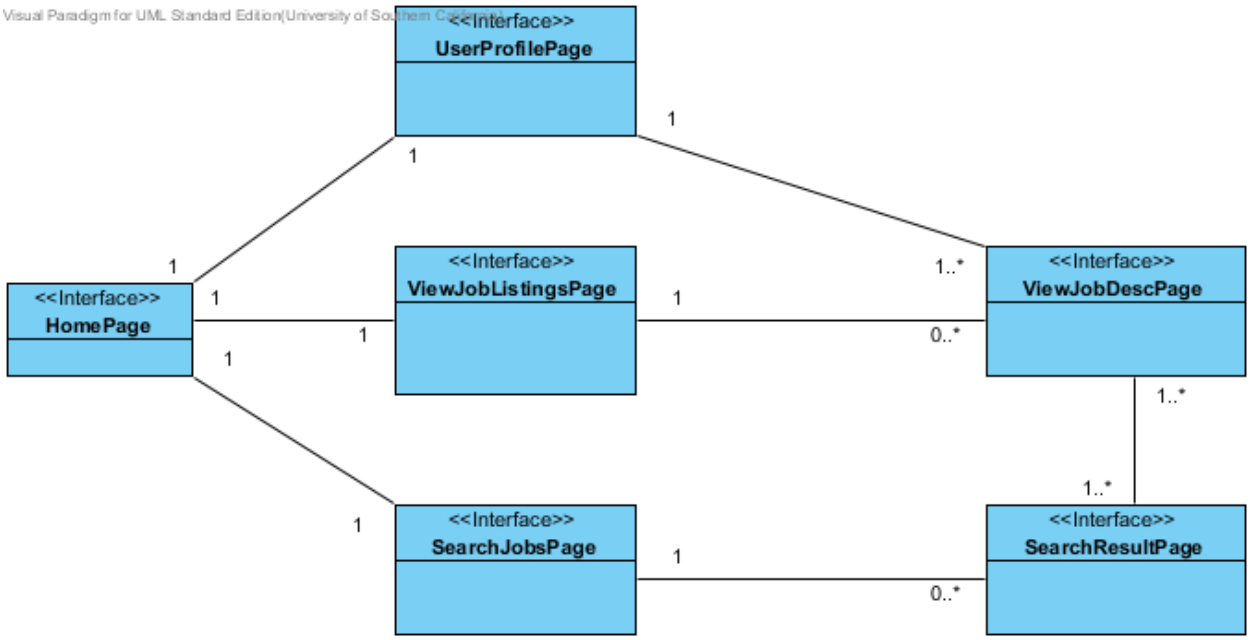

Figure 8: Interface Class Diagram

Class	Type	Description
HomePage	Boundary	The first page of the application.
UserProfilePage	Boundary	Allows the user to create and update their profile.
ViewJobListingsPage	Boundary	Allows user to view all the currently open job titles.
SearchJobsPage	Boundary	Has a search bar for the user to input keywords for searching the ExistingJobs database.
SearchResultPage	Boundary	Allows user to view all the job titles returned by the search query.
ViewJobDescPage	Boundary	Contains a description of the job. Allows users to subscribe for a notification if the job is not open.

Table 6: Interface Class Description

4.1.3 Process Realisation

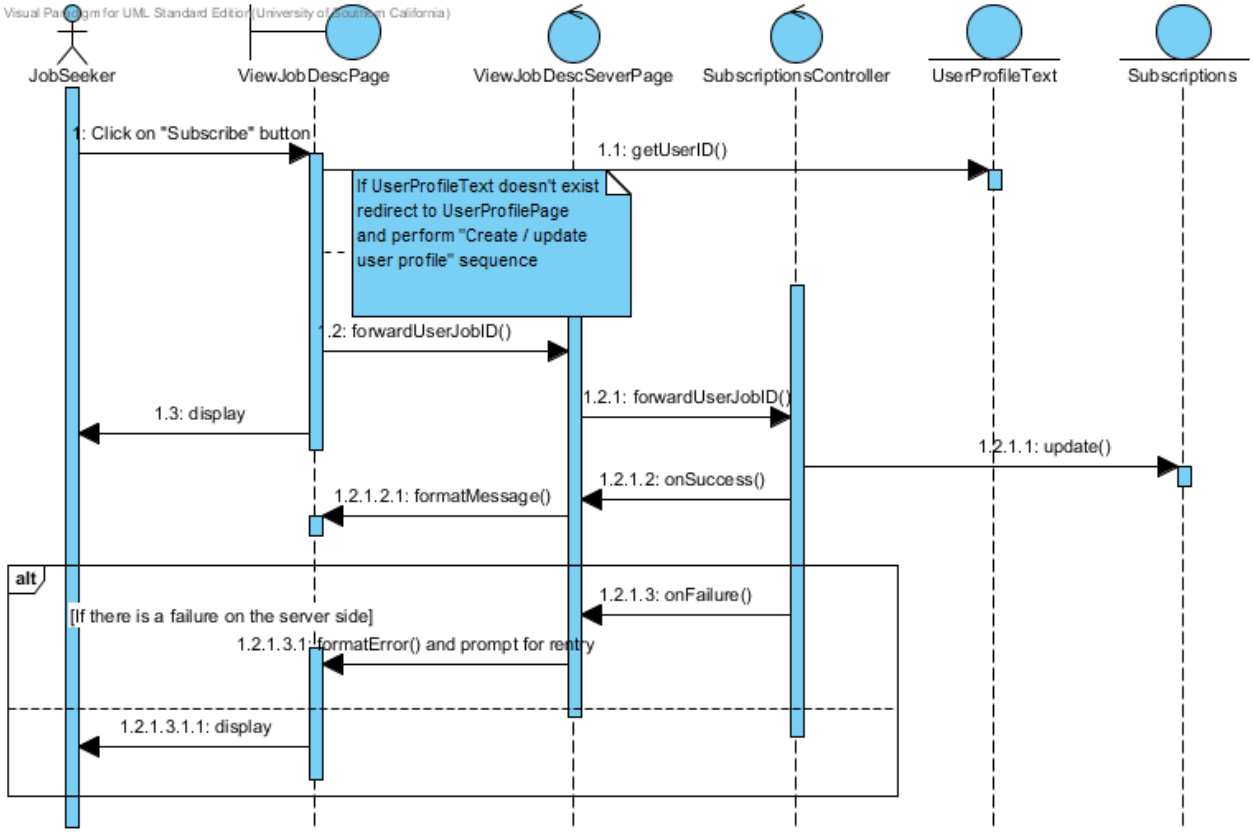

Figure 9: Process Realisation - Subscribe for notification

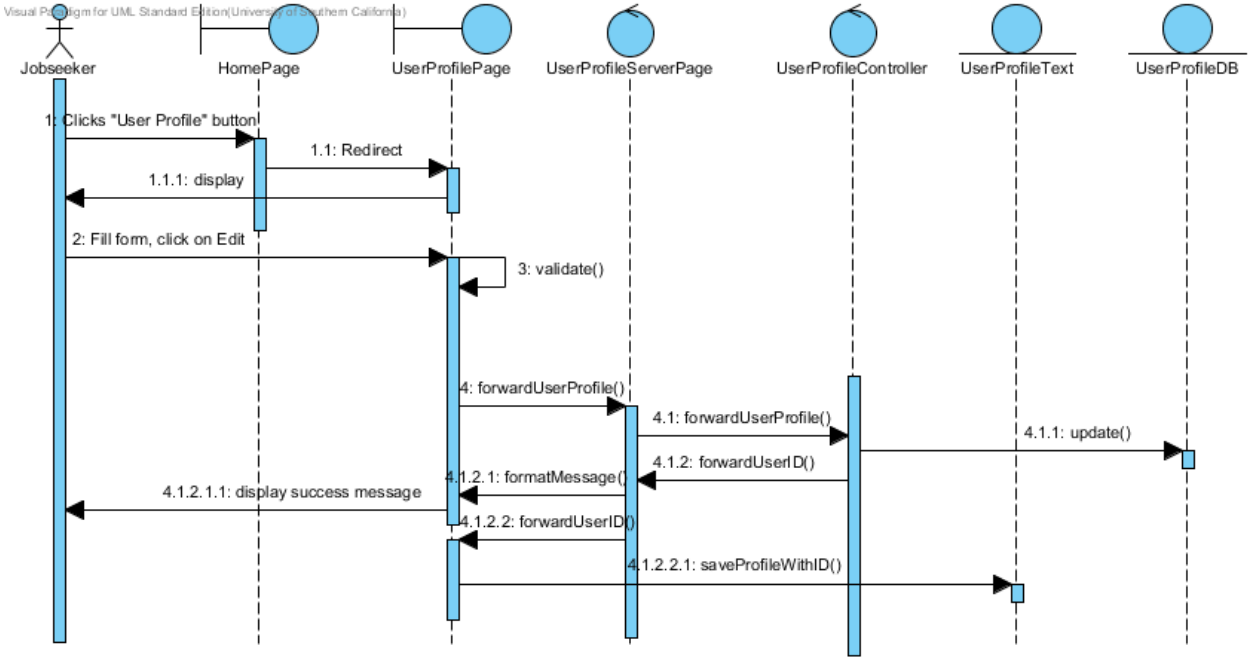

Figure 10: Process Realisation – Create / Update user profile

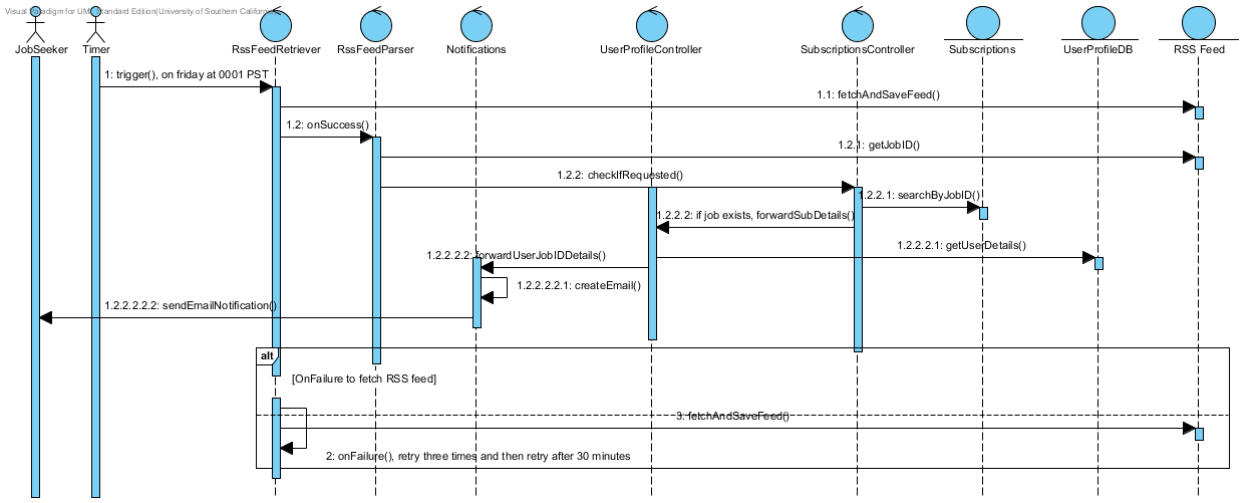

Figure 11: Process Realisation – Process RSS feed and Send Notifications

4.2 Design Rationale

We have adopted a layered architecture with a state-logic-display pattern. The 3 layers are:

- Display layer: Client side application containing the user interface.
- Logic layer: Server side java application containing the two components – subscription component and notification component.
- State layer: A DB2 database containing tables with the required information.

We have chosen this three layered architecture to clearly separate the client side of the application from the server code and the database does ensuring that the client has no direct access to the database.

To ensure that our application works on both Android and iOS we have chosen to use a COTS called Phonegap. Phonegap provides us with a means to develop for both Android and iOS simultaneously does ensuring that both the apps behave and look the same.

For the process realisation diagram we have added sequence diagrams for two of the riskiest functionalities of our application.

These are:
Subscription for Notification: This consists of two sequence diagrams, one covering the subscription for notifications and the other covering the creation / updation of the user profile used for the subscription

Processing RSS feed and sending Notifications: This consists of the server downloading and parsing the RSS feed and sending notifications to the users who have subscribed for a job which has opened.

5. Architecture Styles, Patterns and Frameworks

Name	Description	Benefits, Costs, and Limitations
Layered style (state-logic-display pattern)	The system is divided into three separate layers. Each layer communicates only with the ones immediately above and below it. The layers are: client side application (display), java server side application (logic) and DB2 database (state)	**Benefits:** Clear separation of the display, logic and data store in the application. Also ensures security as only the server side code can access the database. Clients have no direct access to the database. **Costs:** Each time the client needs to interact with the database it has to go through the server.
Client-server style	The mobile devices act as clients which access the application server to perform tasks.	**Benefits:** Any number of clients can access the server at a given point of time. Server does not need to keep track of the client's state as the client will provide all information to the server. **Costs:** Single point of failure. If the server goes down the entire application will fail.
Publish-subscribe style	The job-seeker subscribes for notifications. The server keeps a track of these subscriptions and publishes notifications when the job opens.	**Benefits:** Since some jobs may open rarely the job-seeker doesn't need to check for the job repeatedly. The server will inform the client when the job opens. **Limitations:** The server is completely responsible for sending notifications. May overload the server.

Table 7: Architectural Styles, Patterns, and Frameworks