FINAL REPORT

Table of Contents

1. Overview

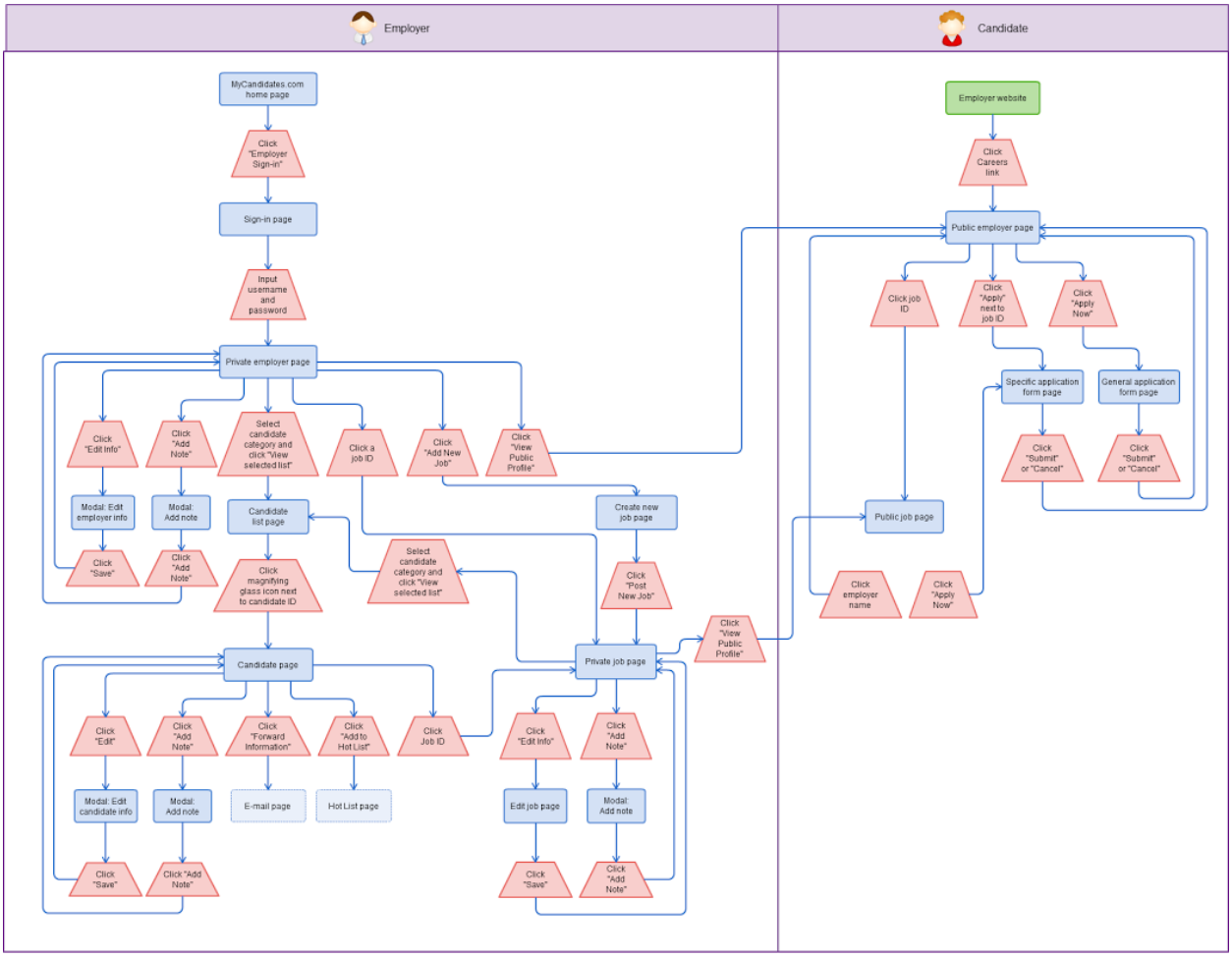
This document will cover the final report for the MyOpenJobs UTD Senior Design Project - MyCandidates. The impact, workflow, data structures, test plans and individual assessments will be discussed in this final report.

2. Impact

The interface for MyCandidates.com provides a simple and easy alternative for small- and medium-sized businesses to properly manage job postings and potential candidates. A large portion of users browse MyOpenJobs' other services using mobile devices. As a result, this interface was designed and implemented from a Mobile First approach as a Single Page Application for quick response times on both mobile and PC platforms Other application tracking systems' interfaces include too many minute details that the average small business does not need. Excess fluff and unwanted and costly features that larger applications tracking systems have implemented have been removed. MyCandidates will be a simplified tool that will greatly enhance the small- to medium-sized employer's hiring process.

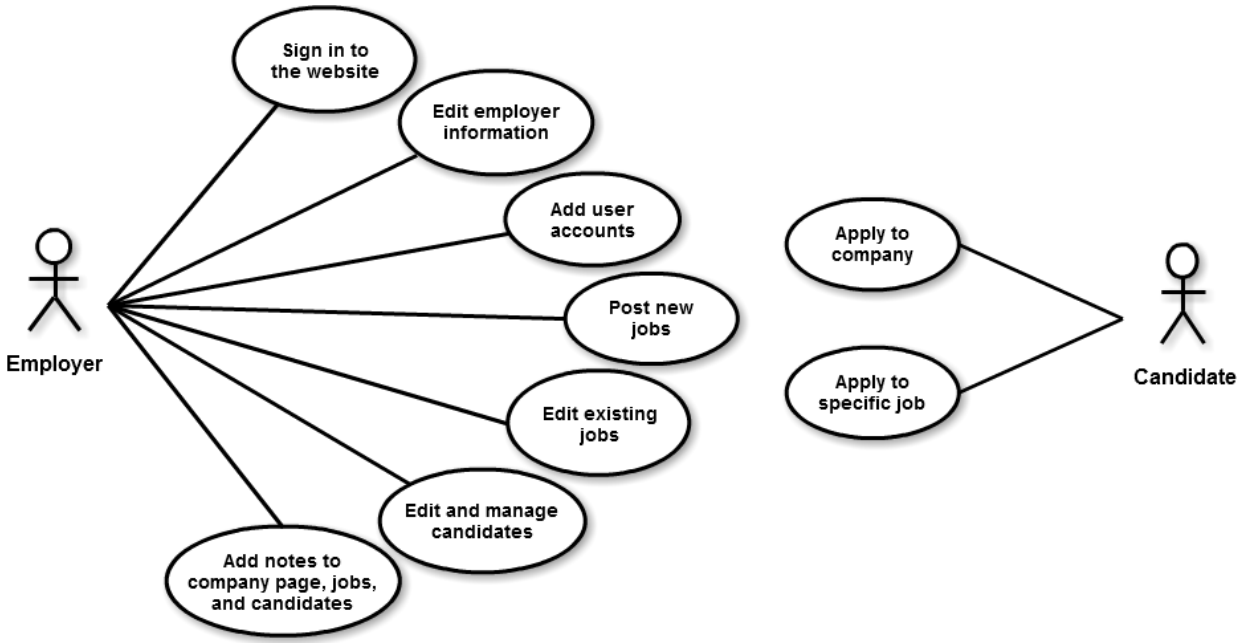
3. Workflow

The following diagram represents the potential workflows for both the Employer and the Candidate.

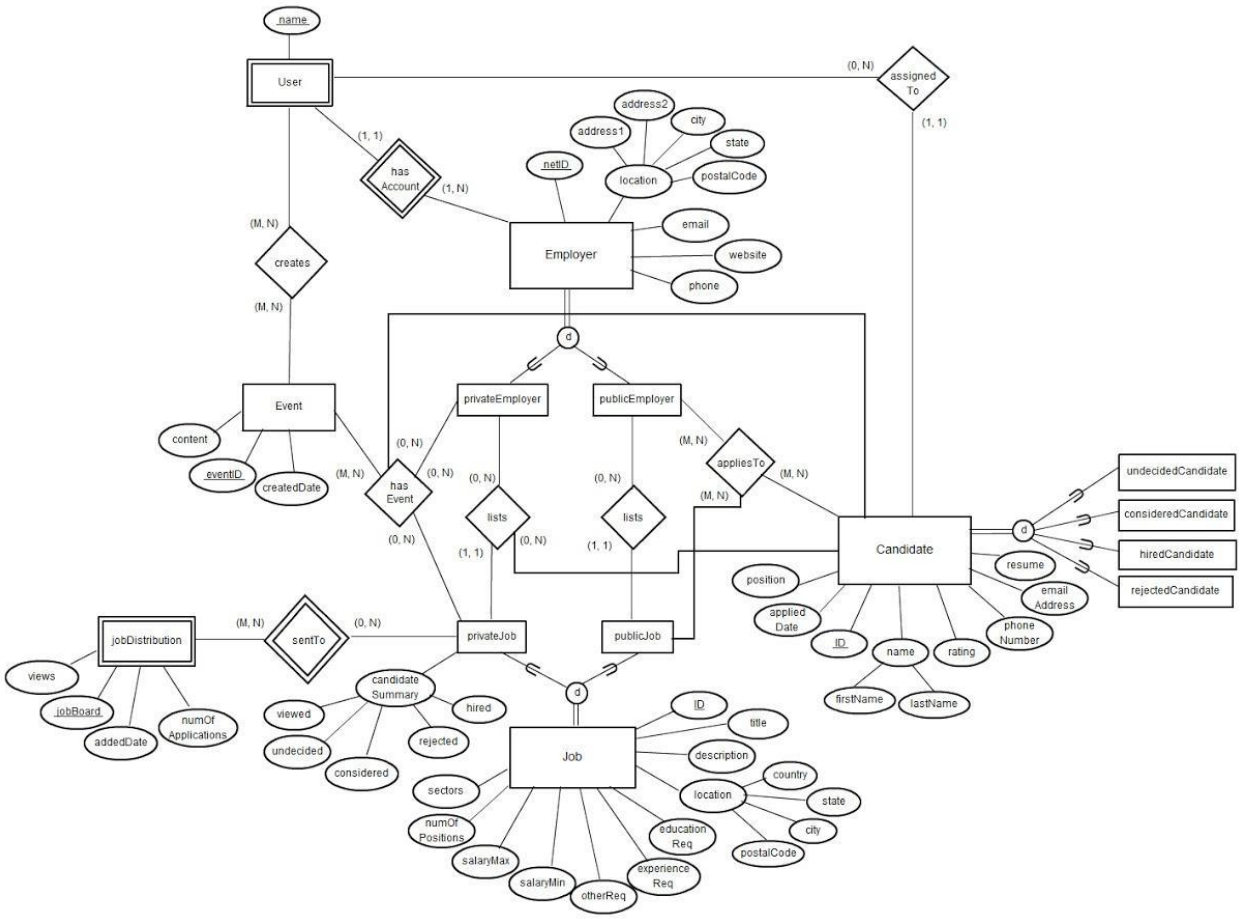


4. Use Case Diagram

The use case diagram shows all potential actions that all actors can take within the MyCandidates application. Actors include the employer and the candidate.



5. EER Diagram



6. Relation Tables

Candidate

employerNetID	**jobID**	**ID**	firstName	lastName	position
phoneNumber	emailAddress	appliedDate	**assignedTo**	resume	type
rating	**eventID**				

PRIMARY KEY (ID),
FOREIGN KEY (employerNetID) REFERENCES Employer(netID),
FOREIGN KEY (jobID) REFERENCES Job(ID),
FOREIGN KEY (assignedTo) REFERENCES User(name),
FOREIGN KEY (eventID) REFERENCES Event(eventID);
*type determines whether candidate is undecided, considering, hired, or rejected.

undecidedCandidate

candidateID

FOREIGN KEY (candidateID) REFERENCES Candidate(ID);

consideredCandidate

candidateID

FOREIGN KEY (candidateID) REFERENCES Candidate(ID);

hiredCandidate

candidateID

FOREIGN KEY (candidateID) REFERENCES Candidate(ID);

rejectedCandidate

candidateID

FOREIGN KEY (candidateID) REFERENCES Candidate(ID);

Event

eventID	createdDate	content	**userName**

PRIMARY KEY (eventID),
FOREIGN KEY (userName) REFERENCES User(name);

Job

employerNetID	**ID**	title	description	sectors	experienceReq
educationReq	salaryMin	salaryMax	otherReq	numofPositions	country
state	city	postalCode	privateFlag		

PRIMARY KEY (ID),
FOREIGN KEY (employerNetID) REFERENCES Employer(netID);

publicJob

jobID

FOREIGN KEY (jobID) REFERENCES Job(ID);

privateJob

jobID	viewed	undecided	considered	rejected	hired
eventID					

FOREIGN KEY (jobID) REFERENCES Job(ID),
FOREIGN KEY (eventID) REFERENCES Event(eventID);

Employer

netID	name	address1	address2	city	state
postalCode	website	email	phone	privateFlag	

PRIMARY KEY (netID);

publicEmployer

employerNetID	**publicJobID**

FOREIGN KEY (employerNetID) REFERENCES Employer(netID),
FOREIGN KEY (publicJobID) REFERENCES publicJob(jobID);

privateEmployer

employerNetID	eventID	privateJobID

FOREIGN KEY (employerNetID) REFERENCES Employer(netID),
FOREIGN KEY (eventID) REFERENCES Event(eventID),
FOREIGN KEY (privateJobID) REFERENCES privateJob(jobID);

User

employerNetID	name

PRIMARY KEY (name),
FOREIGN KEY (employerNetID) REFERENCES Employer(netID);

jobDistribution

jobBoard	addedDate	views	numOfApplication	privateJobID

PRIMARY KEY (jobBoard),
FOREIGN KEY (privateJobID) REFERENCES privateJob(jobID);

7. Data Structure for Each Page

Public Employer {
    netId: for example 'utd-222'
    name
    address
    website
    email
    phone
    jobs: an array of objects {
        id
        employerNetId
        title
        description
        location
        datePosted
    }
}

Public Job {
    employerNetId
    id
    employerName
    title
    description

numOfPositions
experienceReq
educationReq
salaryMin
salaryMax
otherReq
sectors
location: an object {
country
city
state
postalCode
}
}

Private Employer {
netId
name
address1
address2
city
state
postalCode
website
email
phone
jobs: is an array of objects: {
id
employerNetId
title
location
candidateSummary: is an object {
viewed
undecided
considered
rejected
hired
}
}
events: is an array of objects {

createdDate
user
content
}
}

Private Job {
employerNetId
id
title
status
description
salaryMin
salaryMax
candidateSummary: is an object {
viewed
undecided
considered
rejected
hired
}
jobDistributions: is an array of objects {
jobBoard
addedDate
viewed
numOfApplication
}
events: is an array of objects {
createdDate
user
content
}
}

Job Info (Edit Job Page) {
employerNetId
id
title
description
numOfPositions

experienceReq
educationReq
salaryMin
salaryMax
otherReq
sectors
location: an object {
country
city
state
postalCode
}
}
Candidate List: an array of objects {
employerNetId
jobId: 0 for general candidates
name
id
appliedDate
rating
status
}

Candidate {
employerNetId
jobId: 0 for general candidates
id
name
position
rating: a whole number
status: such as undecided, considered...
phoneNumber
emailAddress
appliedDate
assignedTo
events: is an array of objects {
createdDate
user
content
}

}

8. Security

Security was not within the scope for this portion of the MyCandidates project. As a result, we did not do any research or testing with this aspect of the project for the final product. The only type of security that was implemented involved the user or tester logging into one of two premade accounts. A password was not required. If the user knew the entire link, they would be able to access all of the private or hidden pages that belonged to an employer. This is by no means a proper alternative for any other form of security. Our primary focus was on the website's interface and its interactions by the users. Should the MyCandidates project continue, any security or any other back end work will need to be done by a separate team in the future.

9. Test Plan

9.1 Purpose

This document describes the plan for testing the interface for MyCandidate.com web application. This Test Plan document supports the following objectives:

- Identify existing project information and the software that should be tested.
- List the recommended test requirements (high level).
- Recommend and describe the testing strategies to be employed.

9.2 Scope

This Test Plan describes the different test conducted for this web application.

The external interfaces to the following devices will be tested:

1. Mobile
2. PCs

The most critical performance measures to test are:

1. Response time for login to the private employer page.
2. Time taken to load all the graphs on private employer page..
3. Loading time for all jobs in the list on public employer page.

9.3 Requirements for Test

The listing below identifies those items (use cases, functional requirements, non-functional requirements) that have been identified as targets for testing. This list represents what will be tested.

9.3.1 Data & Database Integrity Testing

- Verify that each table in the relational schema does not have redundant data.

- Verify that a view, representative of the data required for each HTML page, can be created correctly.
- Verify that the data type of each attribute makes sense for the attribute.

9.3.2 Function Testing

- Requirements for MyCandidates, Section 2.2: "The application shall allow the employers a way to post jobs and manage candidates and applications."
- Requirements for MyCandidates, Section 2.2: "The application shall facilitates the candidates to view the company info, and apply for any jobs or just apply for the company by clicking on apply link."
- Requirements for MyCandidates, Section 2.2: "The employer will have the ability to edit the job list by either adding a new job to the list, update existing information of the job, close any particular job."
- Requirements for MyCandidates, Section 2.2: "The application shall allow the employer to manage candidate's information, their status and ratings."
- Requirements for MyCandidates, Section 2.2: "The application allows the employer to create notes for the major changes made either to job or candidate's information."

9.3.3 User Interface Testing

- Verify ease of navigation through a sample set of screens.
- Verify sample screens conform to Interface standards.
- Requirements for MyCandidates Section 3.1: "The system shall be easy to work within a web browser such as IE, Chrome, Firefox, and Safari, also iOS devices.
- Requirements for MyCandidates Section 2.3: "The application shall be appropriate for the target market of computer-literate students and the company administrators/candidates who are not familiar with up to date technology."
- Requirements for MyCandidates, Section 2.3: "The application operates on a web interface, no routines to handle specific hardware requirements are needed."

9.3.4 Performance Testing

- Verify response time for login into the system.
- Verify response time for application to be submitted.
- Verify response time for uploading the documents for application.

9.3.5 Load Testing

- Verify system response time for listing all the jobs.
- Verify system response time to load all the graphs.

9.3.6 Security and Access Control Testing

- Verify Login security through username mechanism.

9.3.7 Configuration Testing

- Requirements for MyCandidates, Section 3.1: "The web-based interface for the MyCandidates system shall run in Chrome 31.0, Firefox 26.0, Safari 5.1.7, and Internet Explorer 8.0 browsers.

9.4 Test Strategy

- The Test Strategy presents the recommended approach to the testing of the web applications. The previous section on Test Requirements described what will be tested; this describes *how* it will be tested.
- The main considerations for the test strategy are the techniques to be used and the criterion for knowing when the testing is completed.

9.5 Testing Types

9.5.1 Data and Database Integrity Testing

The databases and the database processes should be tested as separate systems. These systems should be tested without the applications.

Test Objective:	Ensure Database access methods and processes function properly and without data corruption.
Technique:	○ Invoke each database access method and process, seeding each with valid and invalid data. ○ Inspect the database to ensure the data has been populated as intended, all database events occurred properly, or review the returned data to ensure that the correct data was retrieved (for the correct reasons).
Completion Criteria:	All database access methods and processes function as designed and without any data corruption.

9.5.2 Function Testing:

Testing of the application should focus on any target requirements that can be traced directly to use cases. The goals of these tests are to verify all the functionality are working up to date. This type of testing is based upon black box techniques, that is, verifying the application (and its internal processes) by interacting with the application via the GUI and analyzing the output (results). Identified below is an outline of the testing recommended

for each application:

Test Objective:	Ensure proper application navigation, adding a new job, adding a note, and candidates, application process.
Technique:	● Execute each use case, use case flow, or function, using valid and invalid data, to verify the following: ○ The expected results occur when valid data is used. ○ The appropriate error / warning messages are displayed when invalid data is used.
Completion Criteria:	● All planned tests have been executed. ● All identified defects have been addressed.

9.5.3 User Interface Testing

User Interface testing verifies a user's interaction with the web application. The goal of User Interface Testing is to ensure that the User Interface provides the user with the appropriate access and navigation through the functions of the applications. In addition, UI Testing ensures that the objects within the UI function as expected and conform to corporate or industry standards.

Test Objective:	Verify the following: ● Navigation through the application properly reflects business functions and requirements, including window to window, field to field, and use of access methods (touch movements, mouse movements) ● Window objects and characteristics, such as menus, size, position, state, and focus conform to standards.
Technique:	● Tested on each mobile device, as well as on each browser for PCs.
Completion Criteria:	Each window successfully verified to remain consistent with benchmark version or within acceptable standard.

9.5.4 Performance Profiling

Performance testing measures response times, login process, and other time sensitive

requirements. The goal of Performance testing is to verify and validate the performance requirements have been achieved. Performance testing is usually executed several times, each using a different "background load" on the system. The initial test should be performed with a "nominal" load, similar to the normal load experienced (or anticipated) on the target system. A second performance test is run using a peak load.

- Verify response time for login into the system.
- Verify response time for application to be submitted.
- Verify response time for uploading the documents for application.

Test Objective:	Validate System Response time for graphs, candidate list, and job lists to load, under a the following two conditions: - normal anticipated volume - anticipated worse case volume
Technique:	● Tested with minimal data to check the loading time for graphs.
Completion Criteria:	● Successful completion after finding the solution.

9.5.5 Security and Access Control Testing

Security and Access Control Testing focus on two key areas of security:

- Application security, including access to the data and functions.
- Application security ensures that, based upon the desired security, users are restricted to specific functions or are limited in the data that is available to them. For example, candidates are permitted to access public job page and list of jobs, but are not authorized to edit the job, create an event, or access private page of an employee.

Test Objective:	Function / Data Security: Verify that candidate can access only those functions / data for which their candidate type is provided permissions.
Technique:	Function / Data Security: Identify and list each user type and the functions / data each type has permissions for.
Completion Criteria:	For each known user type the appropriate function / data are available.

10. Conclusion

The MyCandidates application will eventually be a tool for for small- to medium-sized employers that will assist them with their hiring process. The MyOpenJobs Senior Design Team constructed an interface for MyCandidates.com that was simple to use and efficient. The interface was contained as a Single Page Application (SPA) so that it would be efficient and easy to operate on both PC and mobile platforms. The development time lasted a single semester - September 24th, 2013, to December 17th, 2013.