UR Ticketing System

Sri Hanmantharaopet	200379655
Joel Lesko	200301973

Computer Science 476
University of Regina
Department of Computer Science

November, 2019

Table of Contents:

Problem Definition

In the modern world, the need to keep our buildings up and running is more important than ever, however; when the buildings begin to break down the time it takes for information to reach maintenance sometimes takes longer then it needs to. In order to try and fix the problem we are developing an issue tracking ticket system similar to that found in IT services except it is for building maintenance. This application could be categorized in the application domain of utilities. The main functionality of this application is to be a web based tool where a user can log on and report a problem with the facility. Then the employee can go on and check the reports and complete the task on the report if there is a job that can be done. Inspiration to develop this type of application comes from Joel's many years of being a custodial and maintenance worker. While on the job Joel wasn't entirely sure where the problem was due to vague descriptions or the issue was left too long and now was a much larger fix since the issue was not reported in a timely manner.

With this application there are two actors that will be involved. For the case of both of the actors we will be using the assumption that they already possess the credentials to use the application. First actor is the user who will add reports in the form of tickets which contain descriptions of issues in the facility. The user will have the option to perform one of three functions: create a new ticket, view the status or edit the contents of a ticket that the user submitted, and to view the tickets that others users have submitted. When the user creates a new ticket they will add a description of the work that needs to be done. With the view or edit ticket option the user can see the status of the job or provide more information if required from the employee role. Lastly, the function to view everyone's tickets lets the user see if the issue has already reported and allows the user to increase the priority of the ticket if they want to. The second role is the employee who will check the tickets that have been added to the system and perform a number of functions with the tickets, as well the employee has all of the same base function as the user. First, the employee can request more information about the task where an email will be sent to the user notifying the user that more information is needed for the submitted ticket. Another function of the employee is to choose to accept or decline a ticket where if the employee chooses to decline a ticket a comment can be sent back to the user as to why the ticket was declined. In the reverse case where the employee accepts the ticket the status of the ticket can be changed from pending to open, in progress, or resolved.

Feasibility Study

Maintenance Connection by Maintenance Connection:

https://cmms.maintenanceconnection.com/cpt/?utm_source=capterra&utm_source=GetApp&utm_medium=cpc&utm_campaign=capterra-cmms

Features for Maintenance Connection	Features that Maintenance Connection does not have, but our application does.
• Track work orders. • Record keeping about the users organization. • Equipment inventory management. • Preventative maintenance. • Predictive maintenance. • Maintenance reporting. • Securely protects the users organization maintenance data.	• Email notification is sent to the user when more information is needed or changes have been made to the ticket. • The user can return to a submitted ticket and make changes.

Maintenance Care by Maintenance Care:

https://info.maintenancecare.com/free-maintenance-work-order-software

Features for Maintenance Care	Features that Maintenance Care does not have, but our application does.
• Web request form • Work order manager • Email notification to staff when a work order has been added. • Dashboard report which uses information collected by the organization to try and save time and money on repairs. • Work order management. • Preventative maintenance scheduling. • Parts and inventory management.	• Email notification is sent to the user when more information is needed or changes have been made to the ticket. • The user can check the work order to determine the status of the work that is being done. • The user can return to a submitted ticket and make changes.

When comparing our application to Maintenance Care and Maintenance Connect there are two features that both seem to lack which our system has included. The first of these features that our program has which the other two do not is the ability to send an email to the user if more information about the work that needs to be done is required. Maintenance Care does have an email feature where the maintenance staff is emailed when a new work order is created. In our opinion that is not as effective of an email system since the maintenance staff should be checking the software on a regular basis. Our use of the email system is more effective since it will notify the user that more information is required to perform the job. The assumption here is that after the user submits the ticket they will not always check on the progress of the ticket to know if the employee needs more information, so by sending the user an email there is a higher chance that the user will realize that there is an issue with the ticket and will add the needed information. This way repairs will get done faster since communication between the user and the employee should be more stable and consistent.

An important feature that our software has that the other two appear lack is the ability for the user to return to an existing ticket and make changes. Most likely the user will only make changes to a ticket that they have submitted is if the employee requests for more information. However, in UR Ticketing System the user can go back to a ticket that they created at anytime and add additional information. This will be especially useful for the users that have a tendency to remember additional information as soon as they submit the ticket. As well, to tie back into our email system when the user makes a change to a ticket an email will be sent out to the employee and the user of the new ticket information.

Functional Requirements

User Functions:

- Create ticket - This function allows the user to make a ticket of an issue that they have found in the facility so that the maintenance staff can review the request. Information that the user can provide includes: a title for the ticket, a description of the work that needs to be done, the room number where the issue is, the location of the problem, and if they want to subscribe to email notifications to changes on the ticket.

- Tickets in Queue - This part of our application allows the user to check all of the active incidents that have been submitted. By providing this function will hopefully reduce the number of duplicate tickets in the system. If the user finds a ticket that has already been made that contains their issue, the user will have the option to increase the priority of that ticket.
- My Tickets - This allows the user to easily find their tickets and edit a ticket if the user feels an edit is needed or the user is asked to provide more information by the employee

.

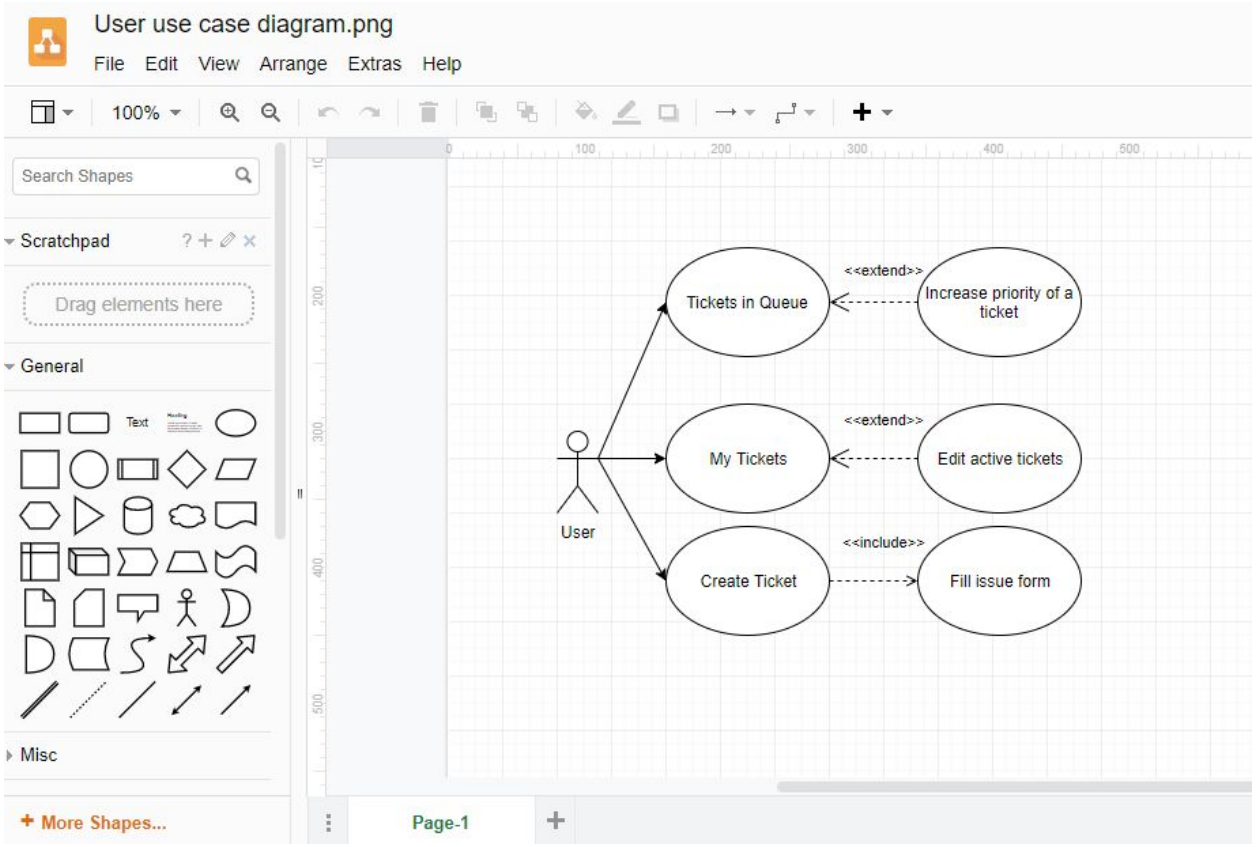
Employee Functions:

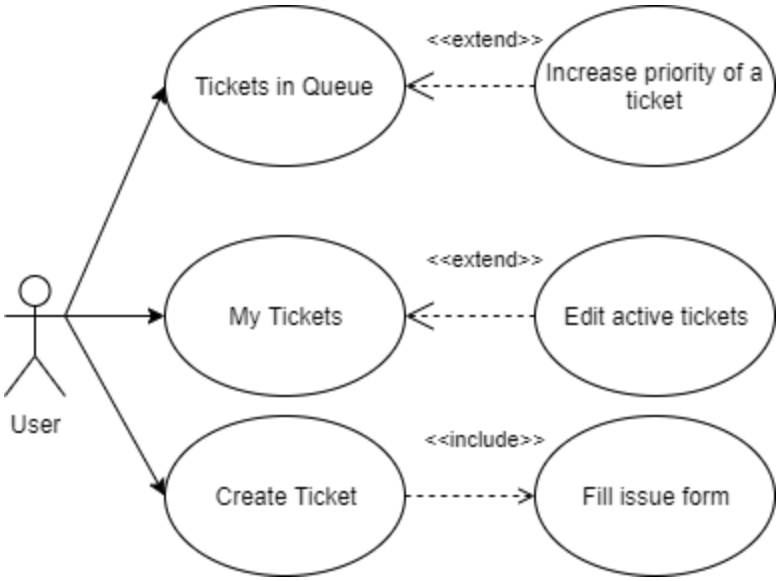
The employee has all of the functionality of the user role with some additional features.

- Tickets in Queue (Employee View) - The main functionality difference here for the employee is the ability to edit any ticket in the system.

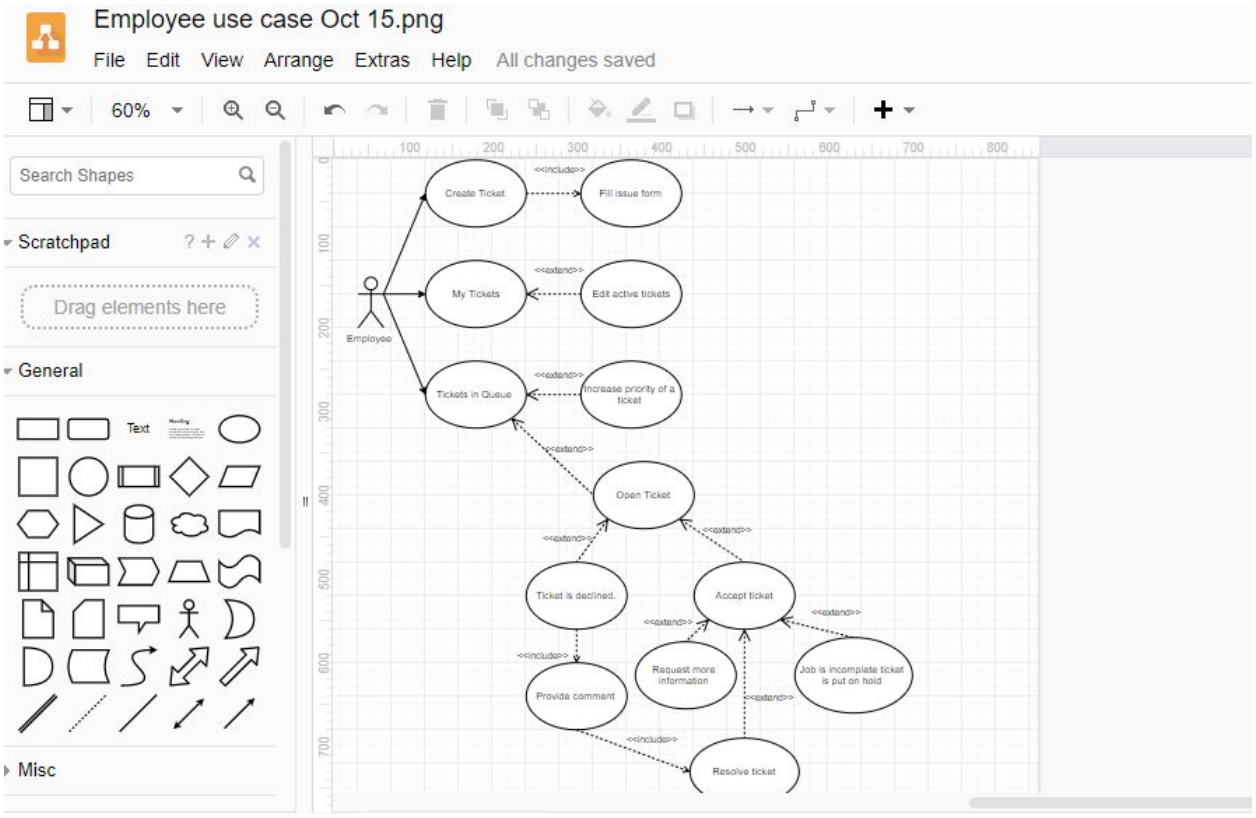
Use Cases

User use case:

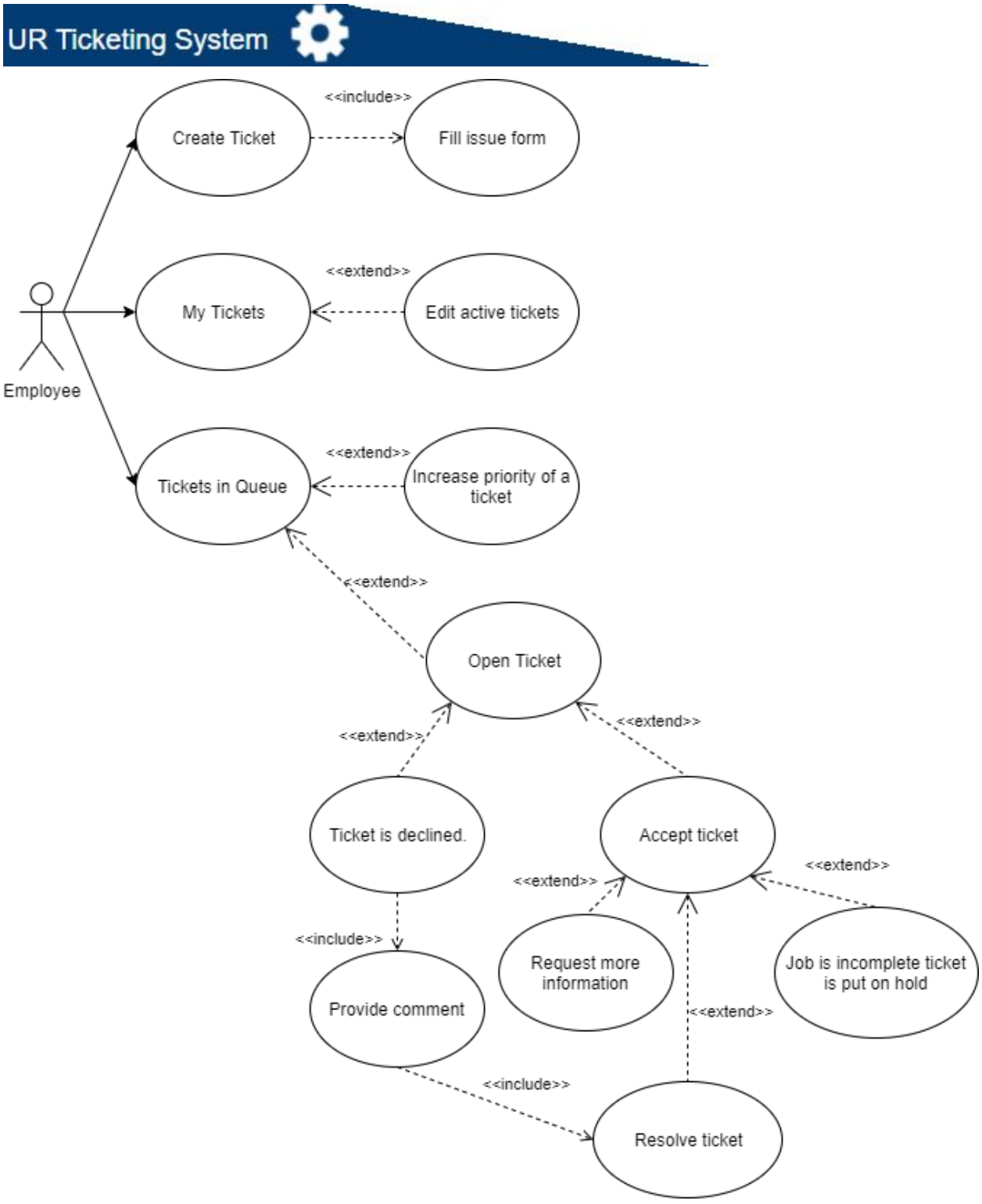




Employee use case:

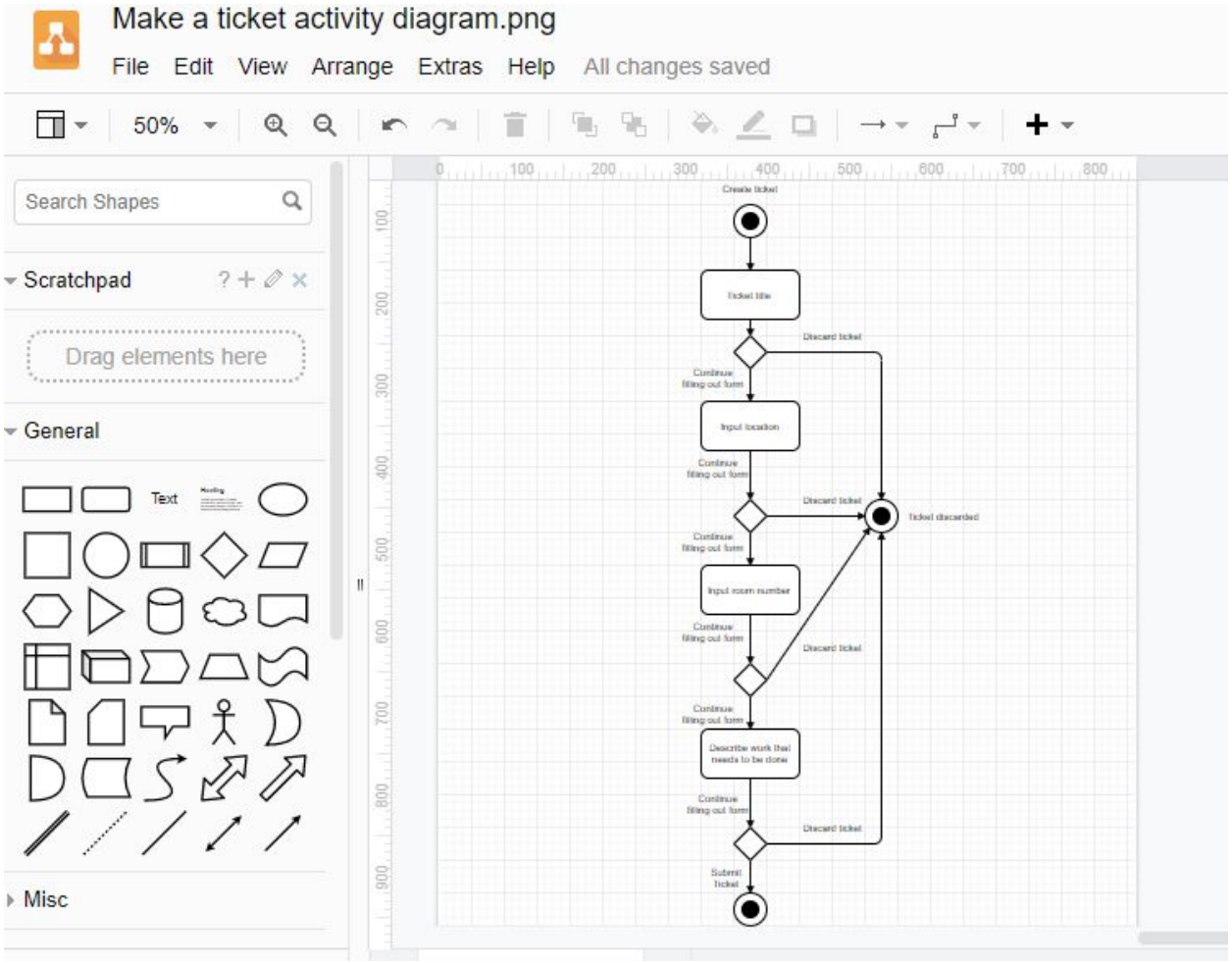


Continue to the next page for a clearer image of the Employee Use Case.

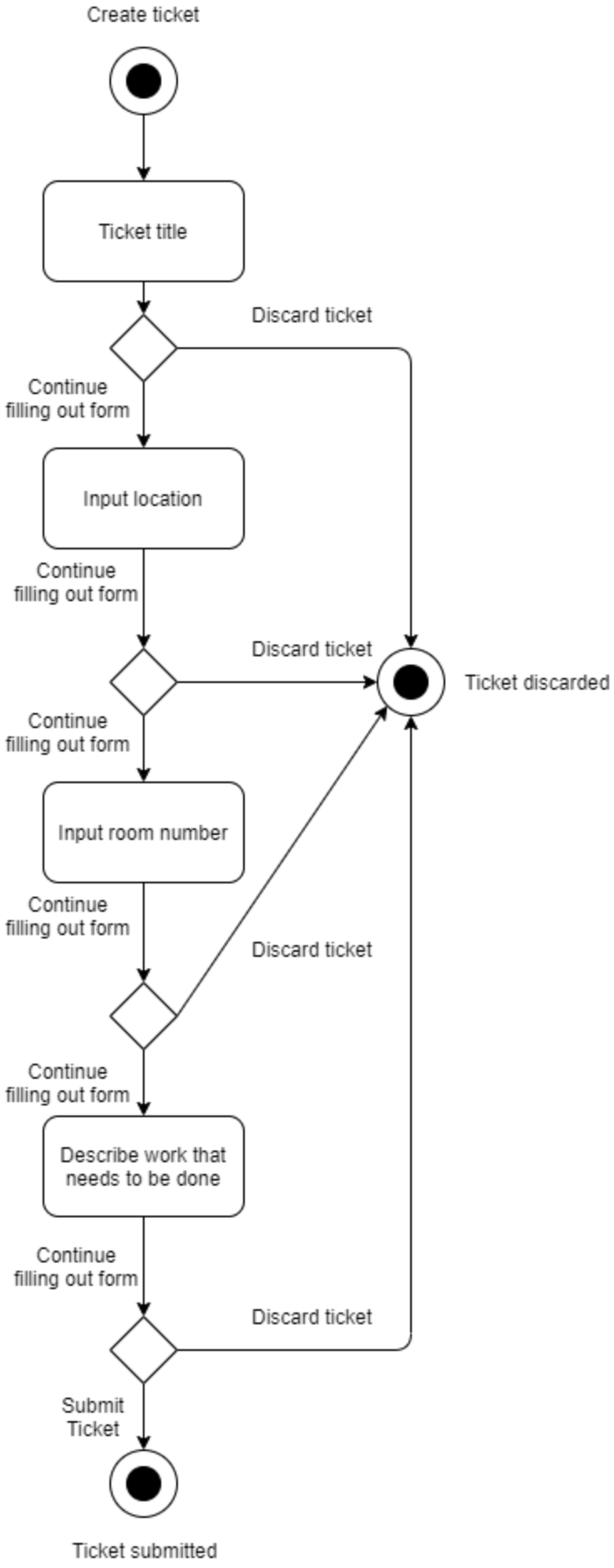

UR Ticketing System
Create Ticket
<<include>>
Fill issue form
My Tickets
<<extend>>
Edit active tickets
Employee
Tickets in Queue
<<extend>>
Increase priority of a ticket
<<extend>>
Open Ticket
<<extend>>
<<extend>>
Ticket is declined.
Accept ticket
<<extend>>
<<extend>>
<<include>>
Request more information
Job is incomplete ticket is put on hold
Provide comment
<<extend>>
<<include>>
Resolve ticket

Activity Diagrams

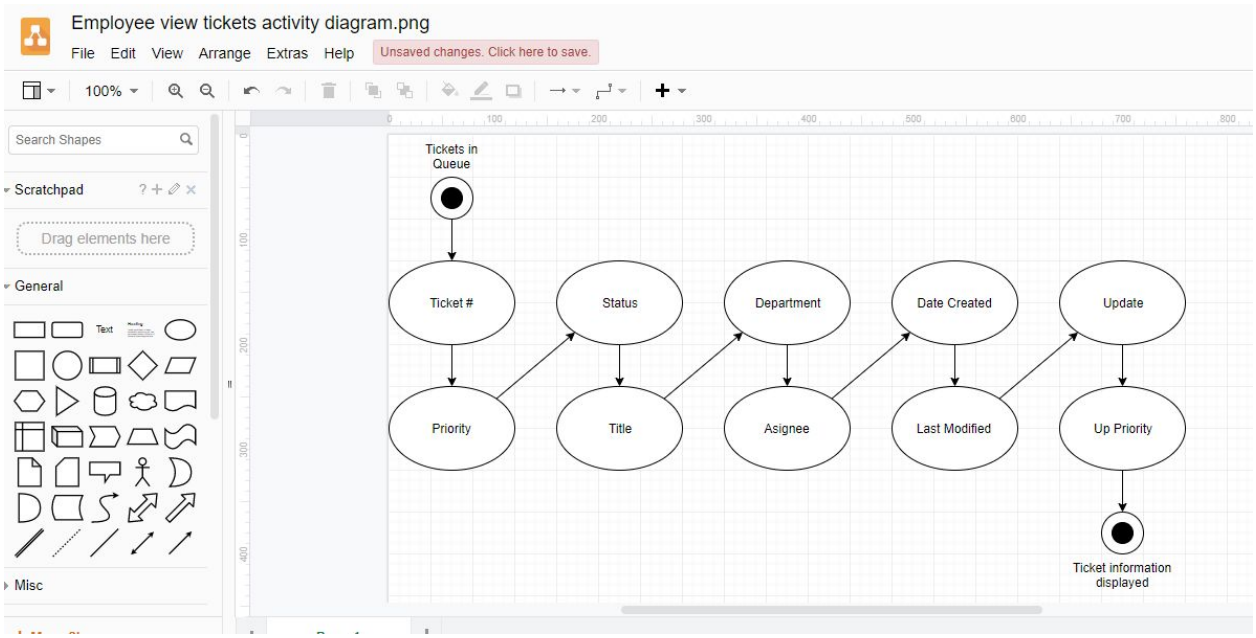
User Makes a Ticket Activity Diagram:

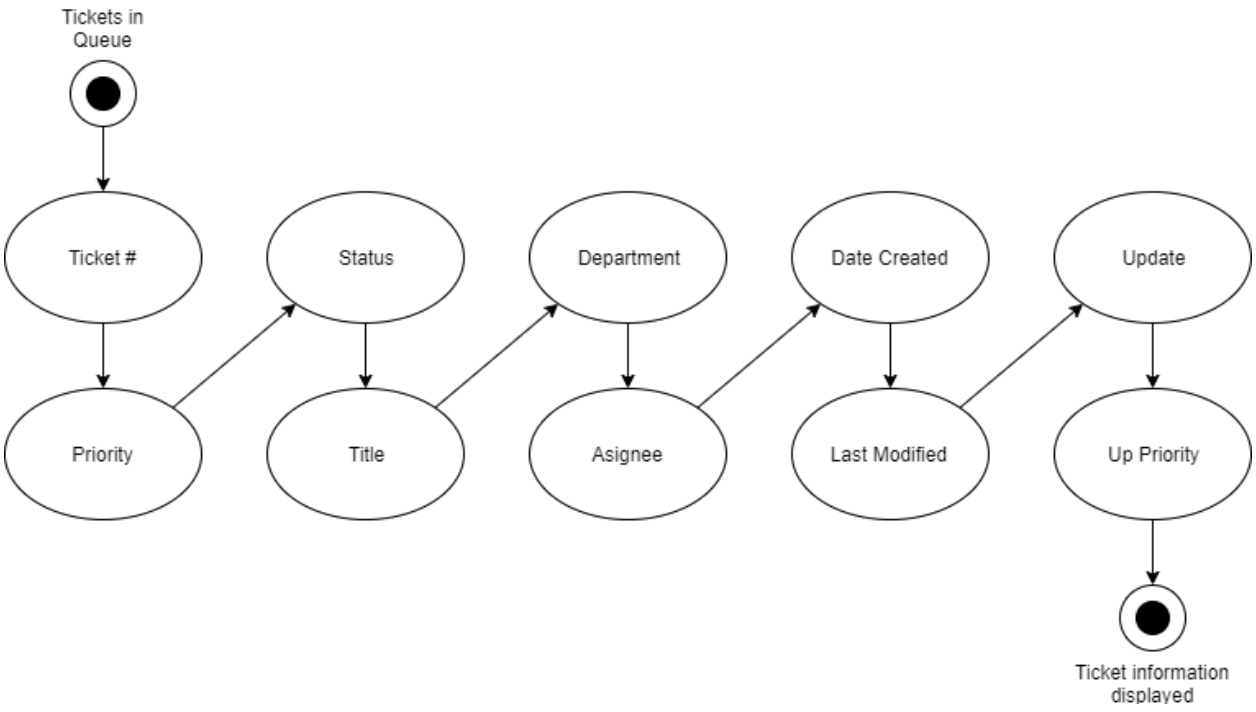


Continue to the next page for a clearer image of the User Makes a Ticket Activity Diagram.

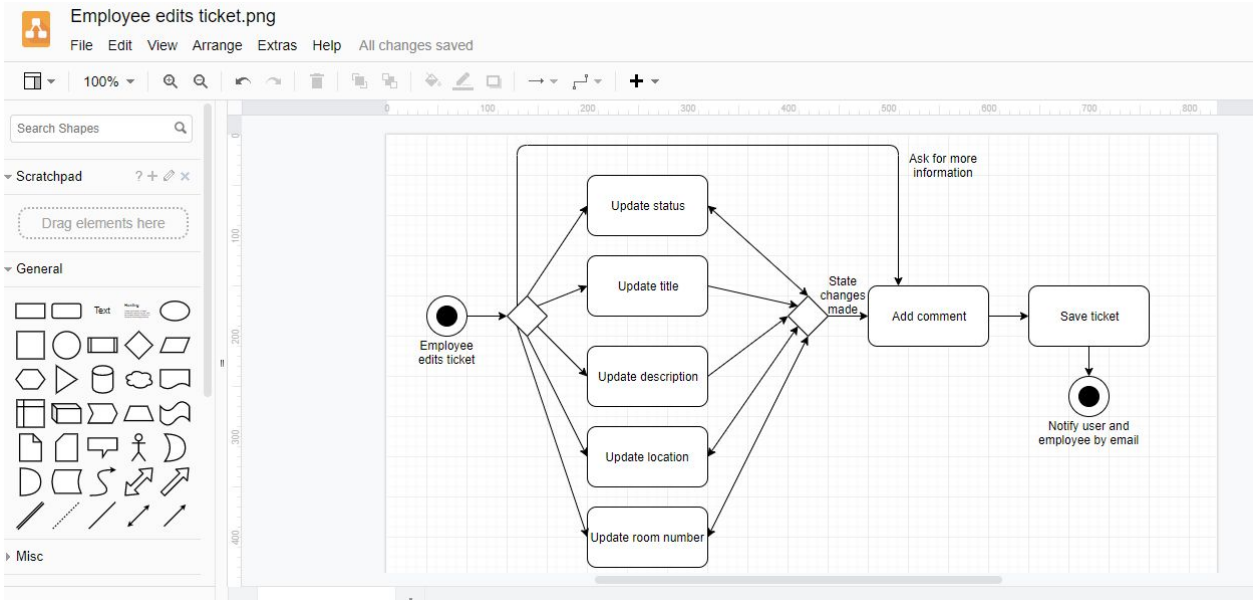

Create ticket
Ticket title
Discard ticket
Continue filling out form
Input location
Continue filling out form
Discard ticket
Ticket discarded
Continue filling out form
Input room number
Continue filling out form
Discard ticket
Continue filling out form
Describe work that needs to be done
Continue filling out form
Discard ticket
Submit Ticket
Ticket submitted

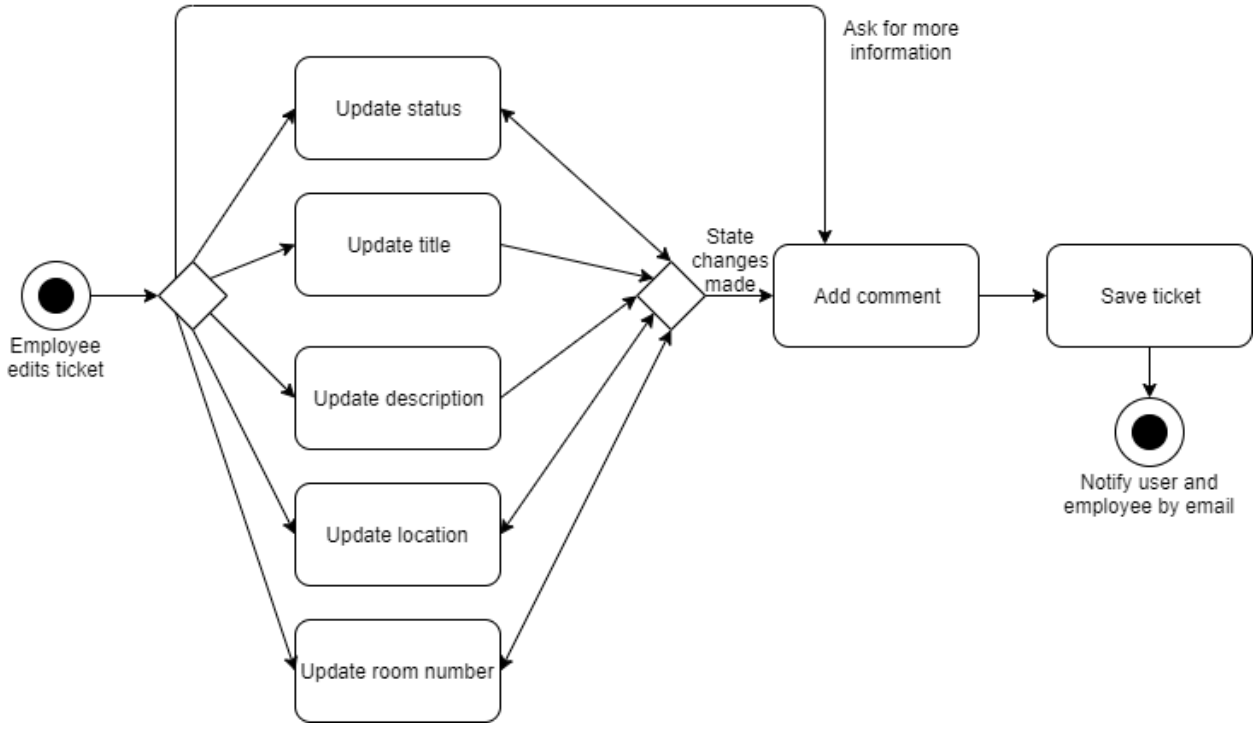
Employee Views Tickets:





Employee Edits a Ticket:

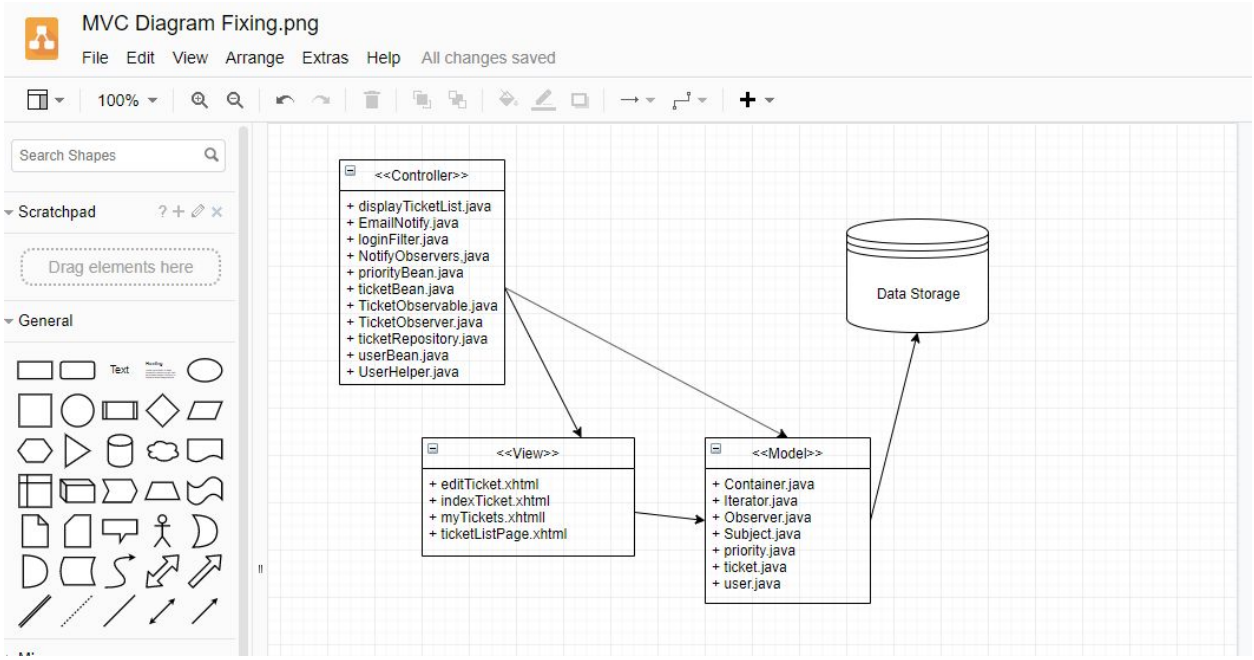




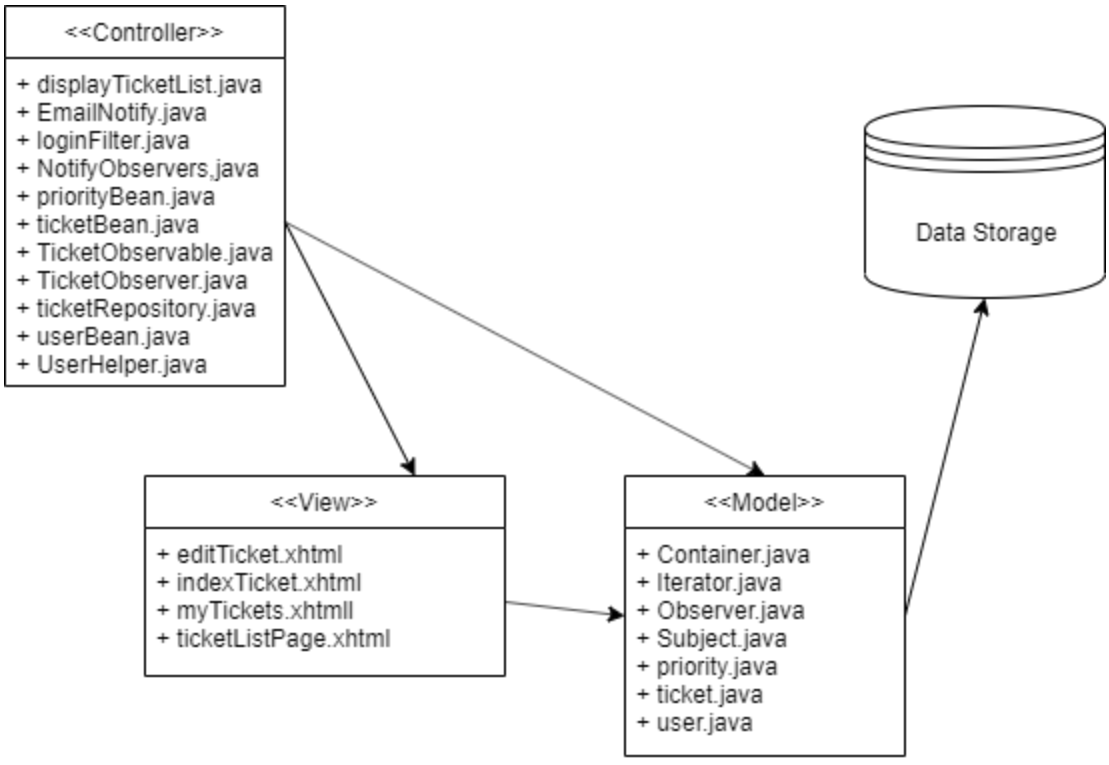
Software Qualities

For this project there are four major qualities that we will need to address to ensure that our application has been made at a quality standard. First of these qualities is robustness, since users should ideally not be visiting the web site all that often the application needs to be able to recover from improper use from users. Because users who do not use the site all that often will probably not know how to fill in information correctly, so the application needs to be able to catch those mistakes. Also, the site needs to be able to have the user connect multiple times to the database without failure in case the user keeps forgetting to add information. Second, the application needs to be correct, in the sense that the correct information is being gathered and stored correctly. As well, with the application being correct the employee then can easily access the data to view the tickets since the information was added in an acceptable format. Third, the application needs to be time efficient since the user will not want to waste time waiting for the application to load. In the case where the application takes too long to load the user not want to use it. Similarly with the employee role, time efficiency is important since the purpose of the application is to allow the employee to quickly access reports about issues around the building faster and a slow application would be defeating this purpose. Lastly, the application needs to be user friendly since no user should be a frequent visitor to the site. The application needs to be easy to understand and to follow so that a first time user can navigate the site effectively. For the employee role user friendliness is important since they will be using the web site all the time so the application needs to be friendly, therefore; the employee does not get frustrated and stop using the services.

MVC Software Architecture



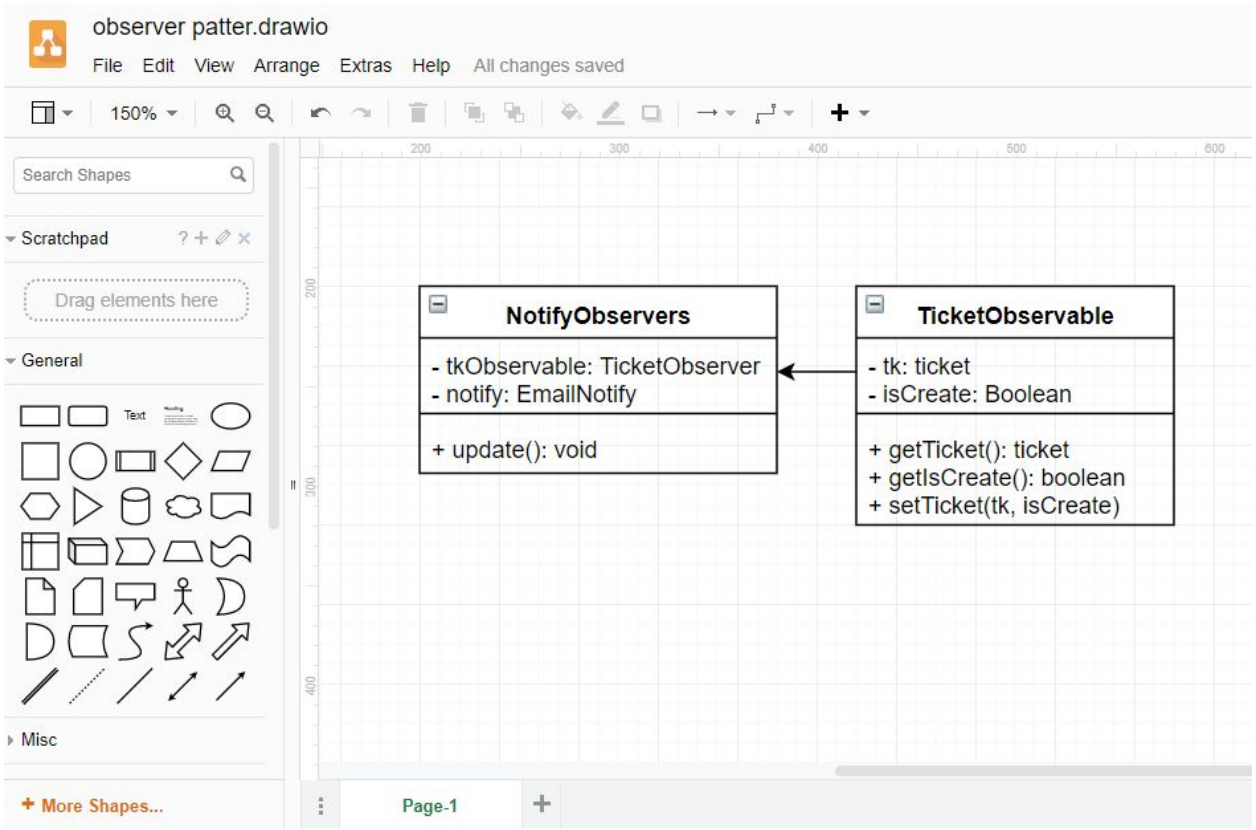
Continue to the next page for a clearer image of the MVC Software Architecture.

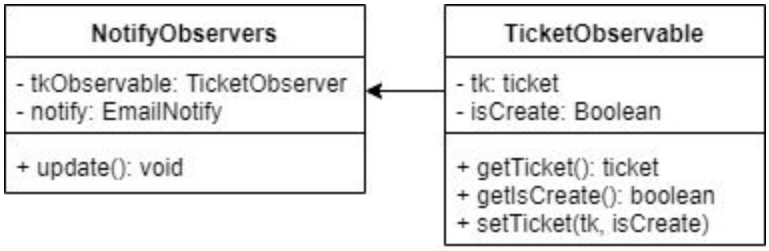


For this project we decided to use the JSF framework. Our reasoning for selecting this framework was due to our desire to develop this project in Java and this framework was recommended to us due to how easy it is to work with. The types of files that are in each part of the MVC changes based on what they have to do. In the controller folder which is named “entities” contain all of the base Java classes that the application needs. Within the model folder that is named “”jsfactivation” Java files that are called Beans can be found which will run functions on the server and can fetch or store data from the database. Lastly, the view folder named “jsf” contains all of the xhtml files that the user can view.

Design Patterns

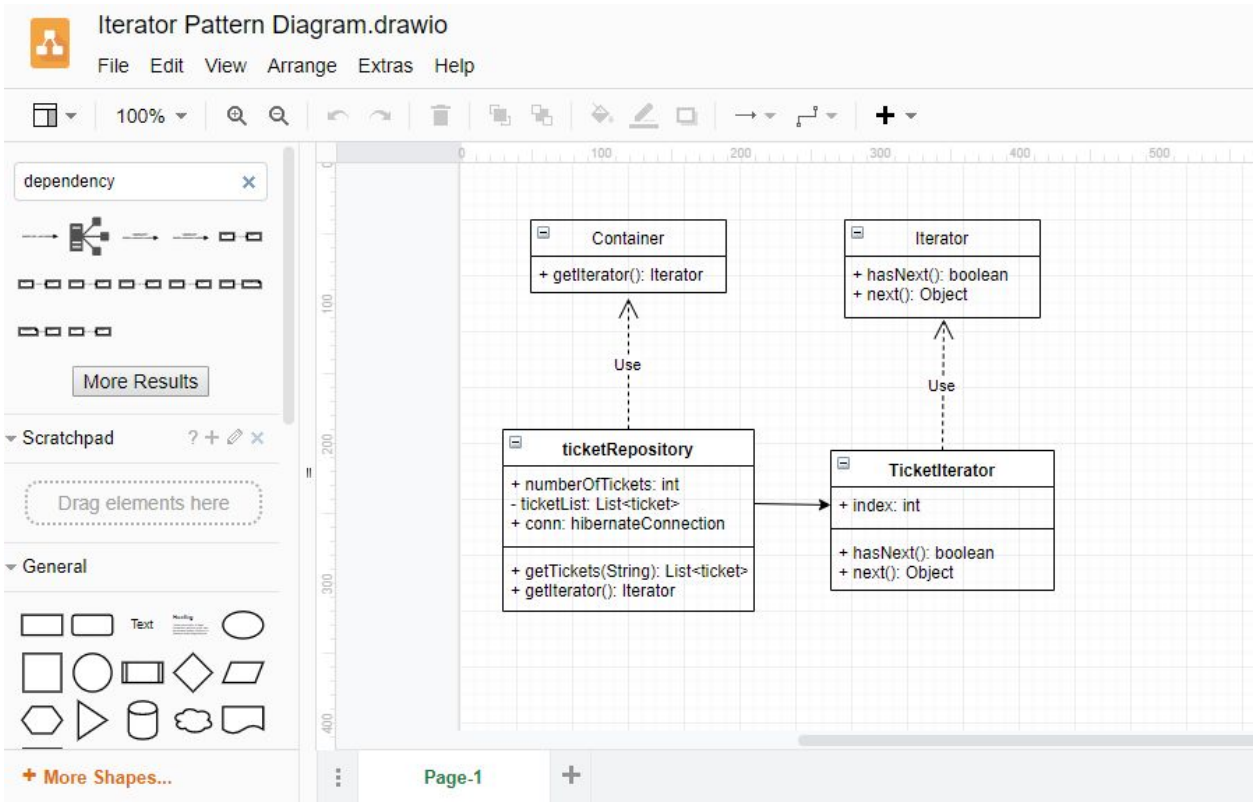
Observer Pattern:

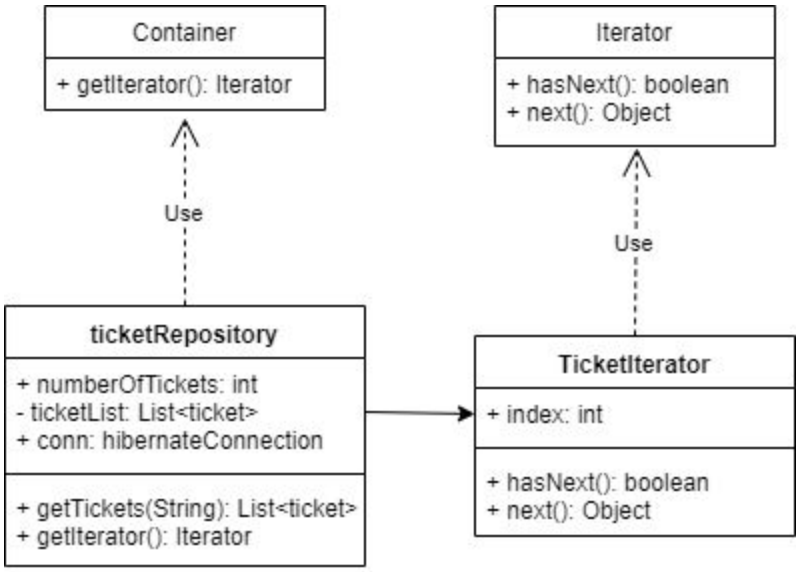




We selected the observer pattern simple due to the fact that we were already using the MVC software architecture and the observer is naturally apart of that architecture. Within our project the observer pattern is used to send notifications to both the user and the employee inorder to keep both roles notified if there are any changes with a ticket. In our system the user can choose to subscribe to be notified about changes with their tickets.

Iterator Pattern:

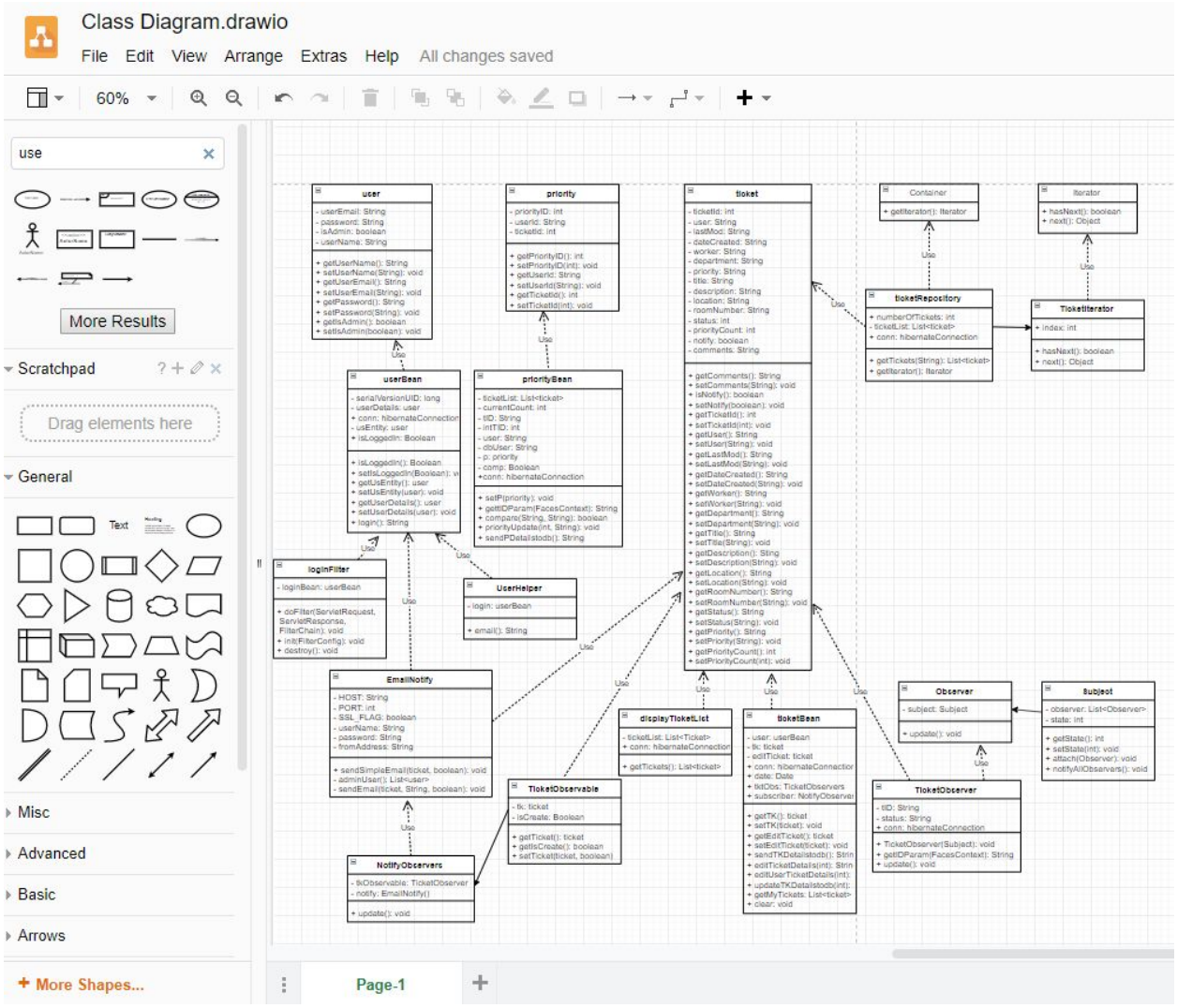




The iterator pattern was chosen since within our project the need to display a list of tickets is necessary. From doing research the iterator pattern is a powerful tool in displaying the lists, therefore; we selecter this pattern. This pattern has been

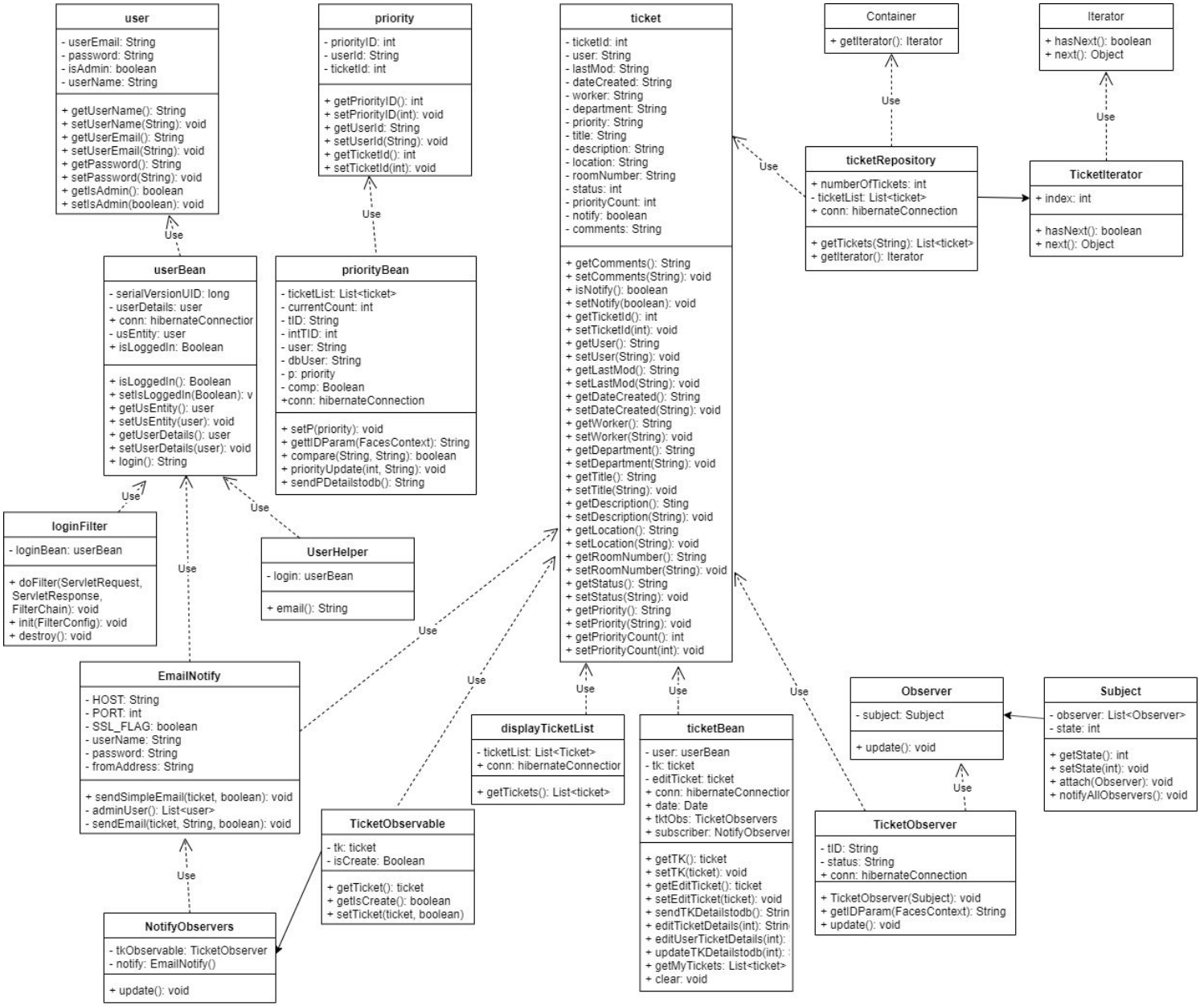
implemented on the page where the user goes to check the tickets that they have submitted.

Class Diagram

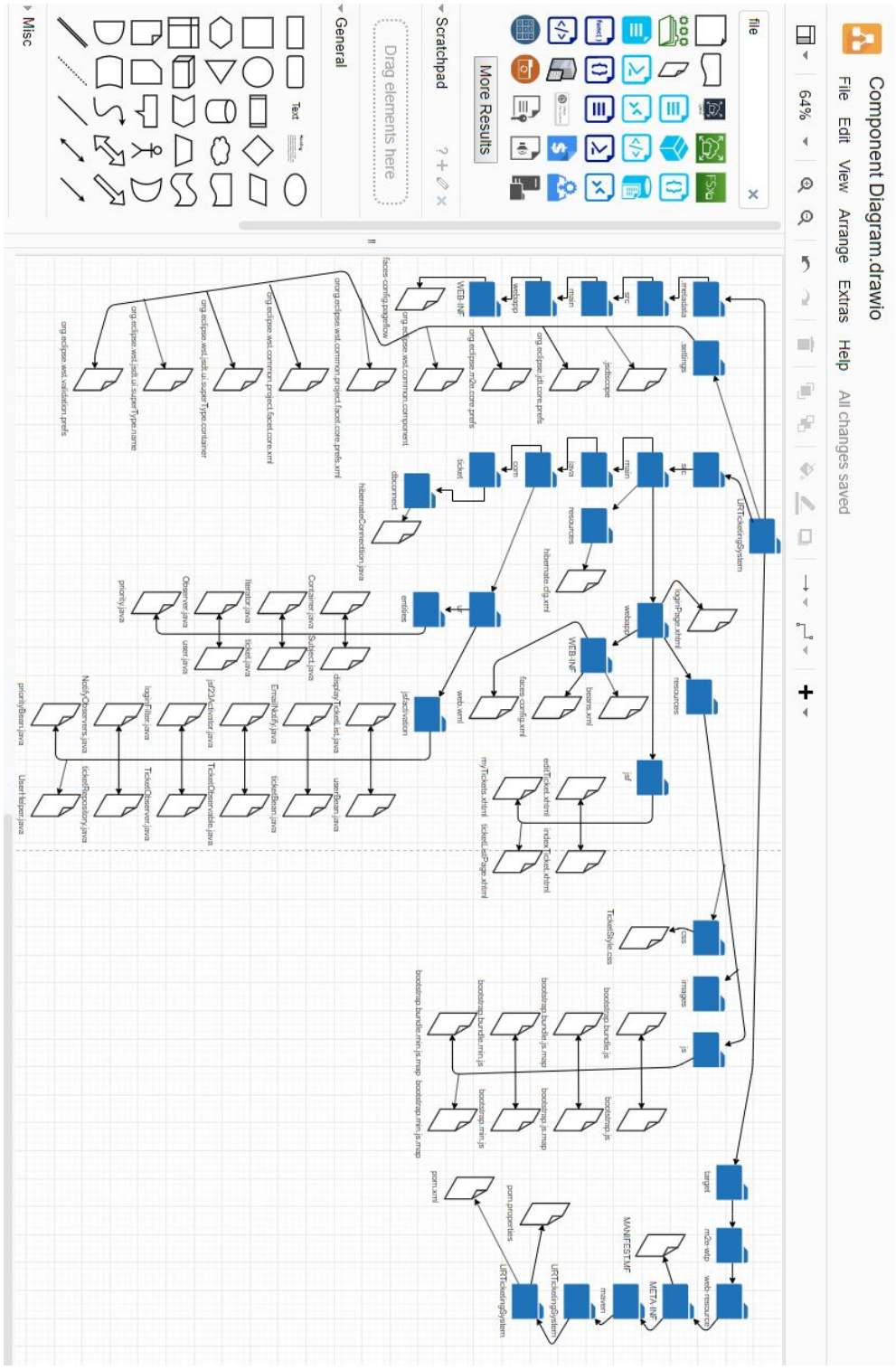


Continue to the next page for a clearer image of the Class Diagram.

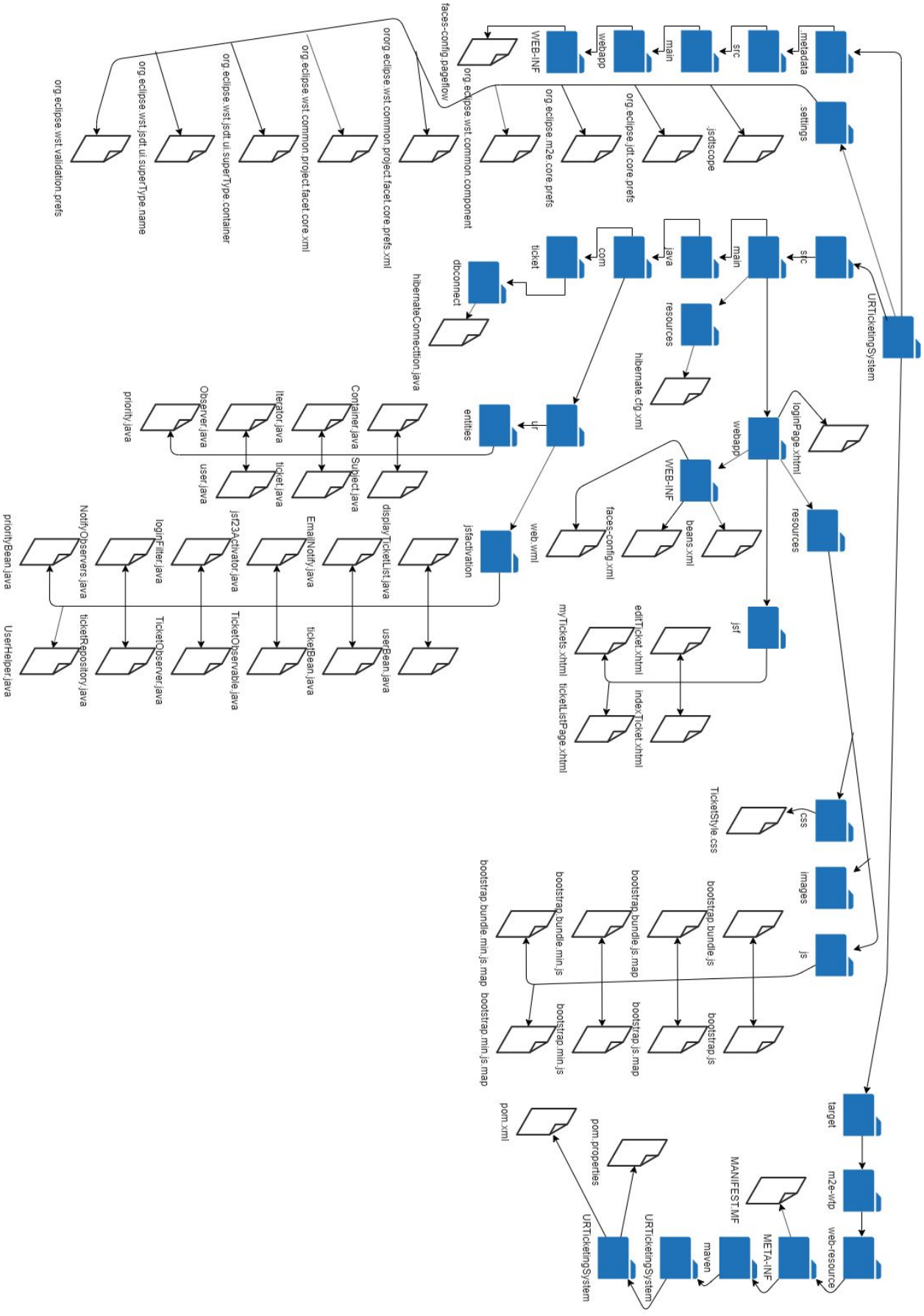
UR Ticketing System



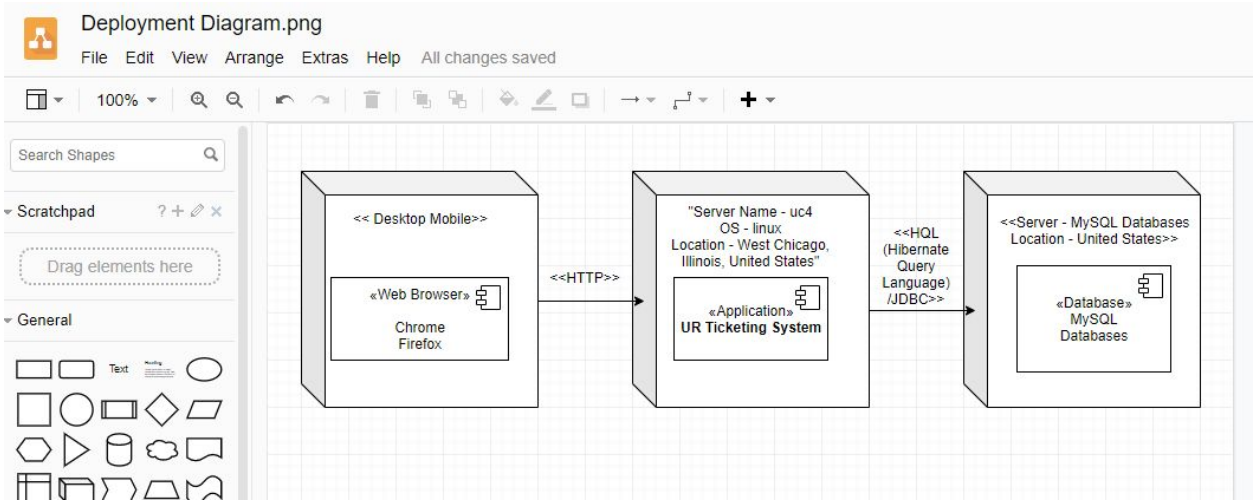
Component Diagram

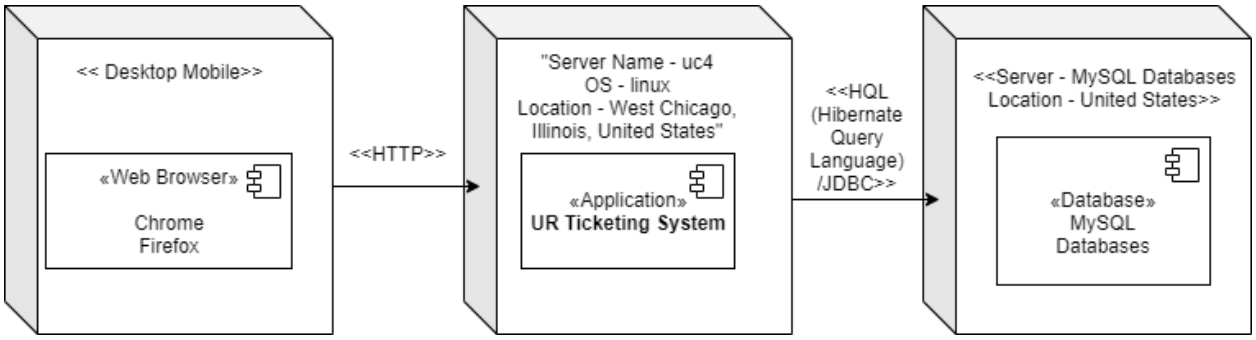


UR Ticketing System



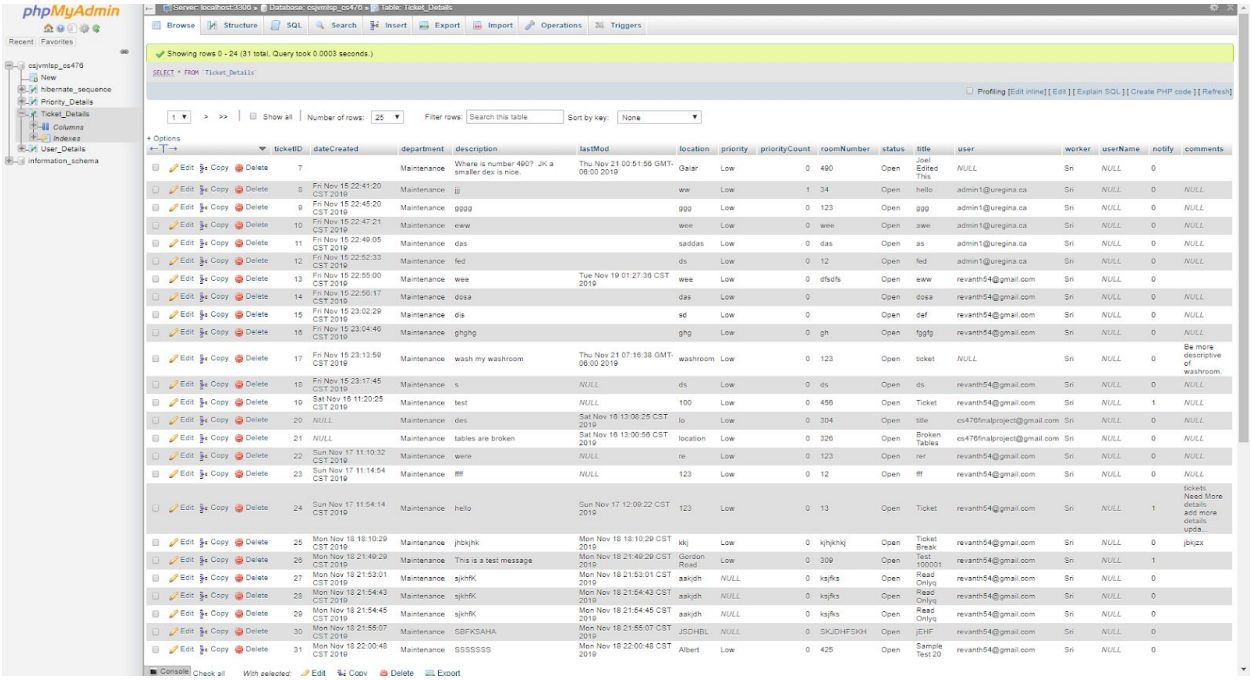
Deployment Diagram



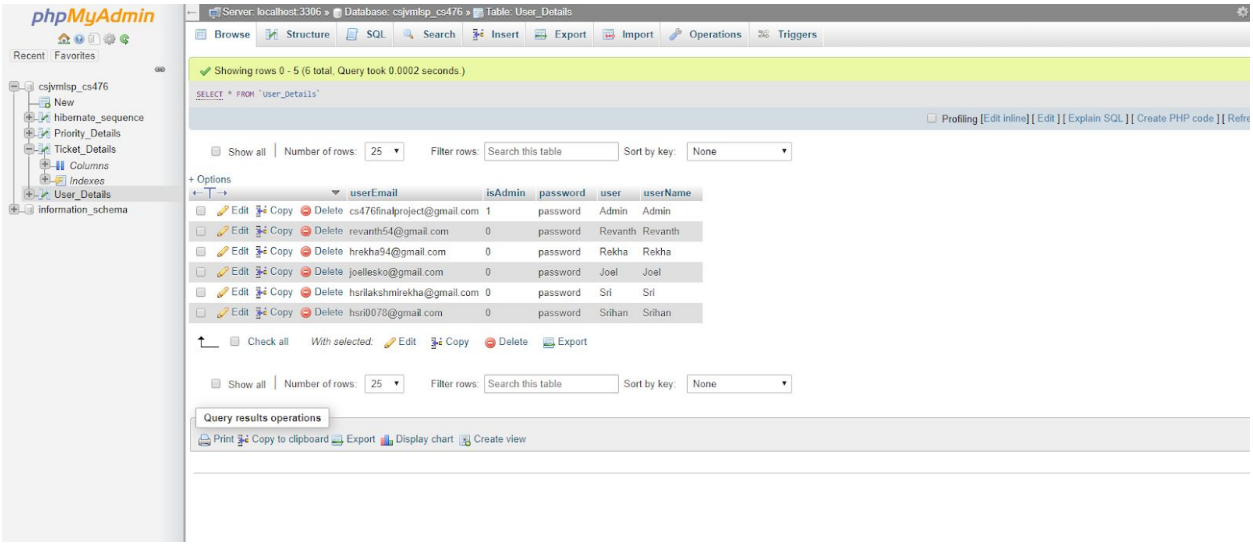


Database

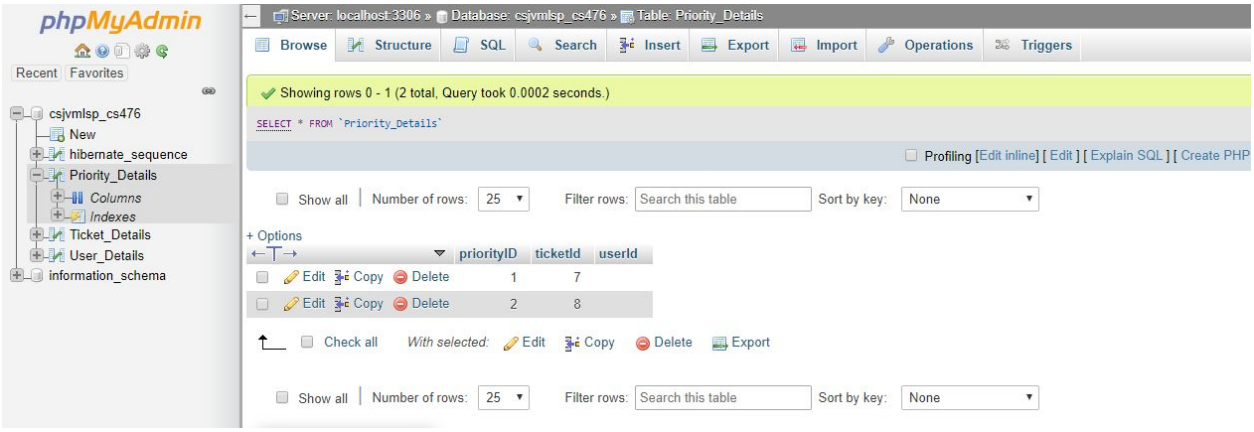
Ticket_Details:



User_Details:



Priority_Detials:



URL

http://cs476.jvmhost.net/URT/

Programing Languages

Throughout our application the following languages were used.

- Materialize and BootsFaces Frameworks: for front end development.
- JSF Framework: Extensible hypertext markup language (xhtml) to display information on the dom.
- JavaEE: for server side functions.
- CDI for dependency injection
- Hibernate for sql operations

Reused Algorithms and Programs

In order to save on time this program has made use code found online.

- The iterator pattern used in the software is a modified version of Tutorialspoint iterator pattern tutorial. https://www.tutorialspoint.com/design_pattern/iterator_pattern.htm

- The observer pattern used is a modified version of Java Code Geeks observer pattern. https://examples.javacodegeeks.com/core-java/util/observer/java-util-observer-example/
- An incorrect use of the observer pattern has been implemented and uses code from Tutorialspoint observer pattern. https://www.tutorialspoint.com/design_pattern/observer_pattern.htm
- The email system uses modified code from Quick Programming Tips. https://www.quickprogrammingtips.com/java/how-to-send-email-in-java-using-apache-commons.html
- The function to compare two string variables in Java was found from Stackoverflow. https://stackoverflow.com/questions/11271554/compare-two-objects-in-java-with-possible-null-values
- The function to take a parameter from an xhtml page and send it to the backing bean was found from MKyong. https://www.mkyong.com/jsf2/jsf-2-param-example/

Software Tools and Educational Experiences

In order to develop this application several software tools had to used which are as followed:

- JavaServer Faces (JSF) - This is the primary framework of the application. Throughout development this software was used to determine how the program should be structured. This can be seen when looking at the MVC diagram which was provided earlier in the report.
- Hibernate ORM (Hibernate) - This is a framework that assisted the project with connecting to the database. Hibernate made it very simple to create tables and columns from the variables in each class. As well, hibernate made it exceedingly easy to store objects created from the Java classes such as ticket.java into the database.
- BootFaces - This framework was used to primarily help design the front end of the application by making it easier to develop the CSS.
- Materialize - This framework was also added to assist with the design of the front end with helping with the creation of the CSS.

The selected software tools did make development easier and despite applications overall basic appearance, this project did have a lot of unforeseen challenges. The development of UR Ticketing System could be summed by a quote from one of its developers Joel Lesko, “This part should be easy! It will take no time at all…. Well after two days of research I can finally make the component.” Prior to working on this project both Joel and Sri had little to no experience working with Java. Simple lines of code such as comparing two string variables could be several hours of debugging, since all programing languages that we have used in the past achieved this with a simple comparison. However, that was not the case for many things in the project.

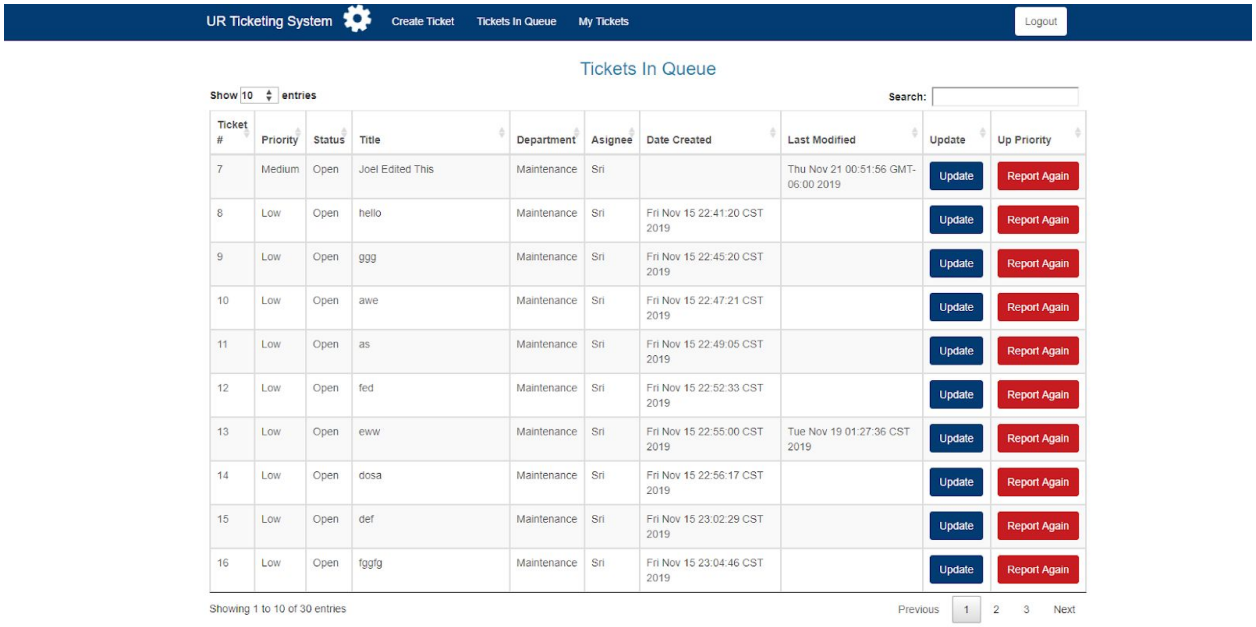
A major learning point in this project was learning how to use and stick with a framework such as JSF. In past classes that consisted of web development had all of the files in one folder and when compared to using JSF all of the files are strictly split up in order to abide by the MVC that the framework laid out. As well, it took some time for the realization that using Java as the primary language did not allow for the creation of dynamic web pages. This is due to Java code running on the server and can not be invoked with the <javascript> tag. After we grasped this concept and gained an understanding as to how to invoke the backing beans, we were able to make some workarounds to counteract the lack of dynamic programming that Java does not support.

Another learning experience was learning how to work with Hibernate. After gaining the basics of Hibernate, the creation of tables and columns was simple. This was due to Hibernate’s ability to take the variables in the class and turn them into the columns of the tables and then populate the tables. By taking the created objects in the system and map the parts of the object to the columns in the table, save on time writing code to add each item to the database. However, upon learning these basics it was soon discovered that it was difficult to use a Hibernate connection to the database to get entities from the database. Unlike using PHP to pull data from a database where a developer can query a single entity, a Hibernate connection will return objects which are not easy to extract small bits of data from such as an integer value. Instead code similar to this needs to be used, “variable = (data type) session.createSQLQuery("SELECT MySQL query'").uniqueResult();”. As well, if a single entity needs to be changed, code like this needs to be used, “org.hibernate.query.Query query = session.createQuery("UPDATE MySQL query");” this is then followed by, “int result = query.executeUpdate();”. In order to derive these lines of code required significantly more research than one would expect.

Also, BootFaces and Materialize had to be learnt in order to make well developed interfaces. Any different frameworks were studied and tested to find the ones that worked best for our application. Ultimately, this lead to the use of Materialize for the login page and the use of BootFaces for everywhere else. Once an understanding of how to use BootFaces the applications appearance increases drastically. This allowed us to see how a framework can take days worth of coding and produce the same results in a matter of minutes. Overall, we both learnt a lot about web development which could constitute as a report in itself and for the sake of not making an already long report longer we will leave our educational experiences at this.

Functional Testing

1) "Tickets in Queue" showing all tickets:



UR Ticketing System | Create Ticket | Tickets In Queue | My Tickets | Logout

Tickets In Queue

Show 10 entries | Search:

Ticket #	Priority	Status	Title	Department	Asignee	Date Created	Last Modified	Update	Up Priority
17	Low	Open	ticket	Maintenance	Sri	Fri Nov 15 23:13:59 CST 2019	Thu Nov 21 07:16:38 GMT-06:00 2019	Update	Report Again
18	Low	Open	ds	Maintenance	Sri	Fri Nov 15 23:17:45 CST 2019		Update	Report Again
19	Low	Open	Ticket	Maintenance	Sri	Sat Nov 16 11:20:25 CST 2019		Update	Report Again
20	Low	Open	title	Maintenance	Sri		Sat Nov 16 13:08:25 CST 2019	Update	Report Again
21	Low	Open	Broken Tables	Maintenance	Sri		Sat Nov 16 13:00:56 CST 2019	Update	Report Again
22	Low	Open	rer	Maintenance	Sri	Sun Nov 17 11:10:32 CST 2019		Update	Report Again
23	Low	Open	fff	Maintenance	Sri	Sun Nov 17 11:14:54 CST 2019		Update	Report Again
24	Low	Open	Ticket	Maintenance	Sri	Sun Nov 17 11:54:14 CST 2019	Sun Nov 17 12:09:22 CST 2019	Update	Report Again
25	Low	Open	Ticket Break	Maintenance	Sri	Mon Nov 18 18:10:29 CST 2019	Mon Nov 18 18:10:29 CST 2019	Update	Report Again
26	Low	Open	Test 100001	Maintenance	Sri	Mon Nov 18 21:49:29 CST 2019	Mon Nov 18 21:49:29 CST 2019	Update	Report Again

Showing 11 to 20 of 30 entries

Previous 1 2 3 Next

UR Ticketing System | Create Ticket | Tickets In Queue | My Tickets | Logout

Tickets In Queue

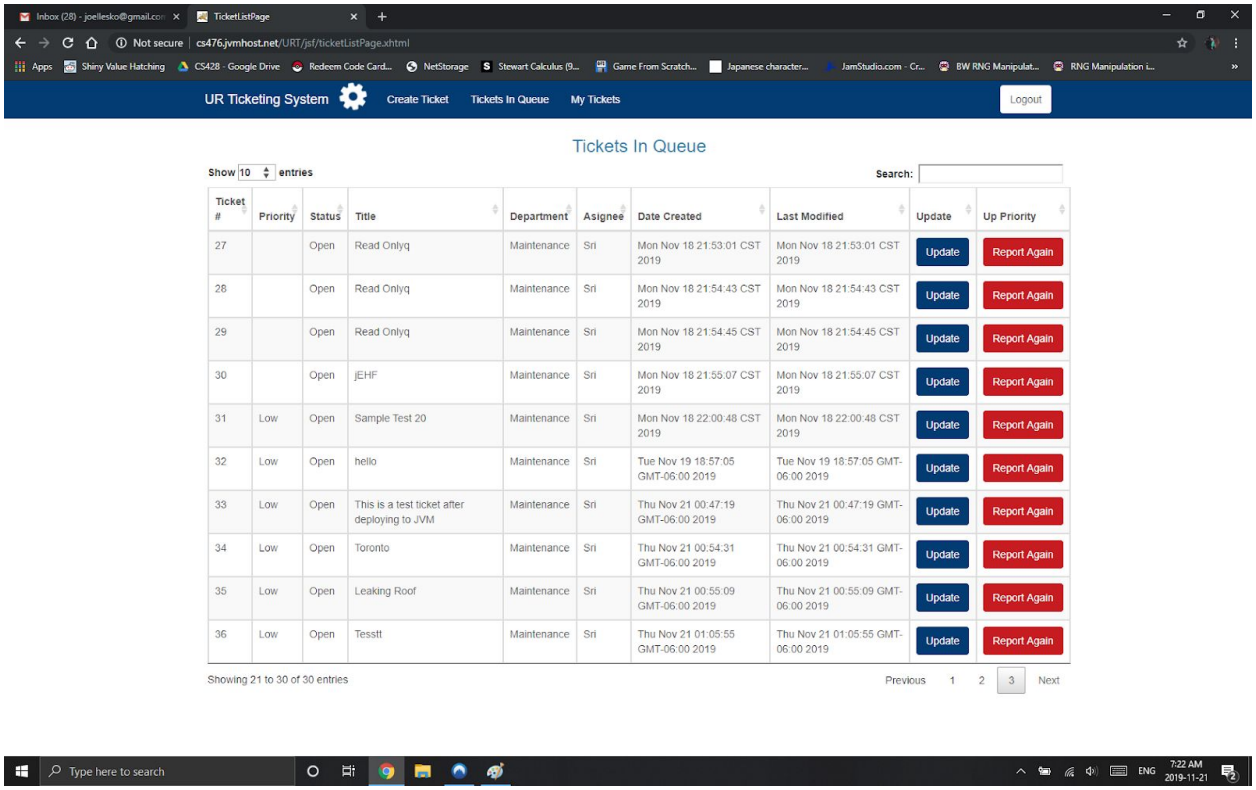
Show 10 entries | Search:

Ticket #	Priority	Status	Title	Department	Asignee	Date Created	Last Modified	Update	Up Priority
27		Open	Read Onlyq	Maintenance	Sri	Mon Nov 18 21:53:01 CST 2019	Mon Nov 18 21:53:01 CST 2019	Update	Report Again
28		Open	Read Onlyq	Maintenance	Sri	Mon Nov 18 21:54:43 CST 2019	Mon Nov 18 21:54:43 CST 2019	Update	Report Again
29		Open	Read Onlyq	Maintenance	Sri	Mon Nov 18 21:54:45 CST 2019	Mon Nov 18 21:54:45 CST 2019	Update	Report Again
30		Open	jEHF	Maintenance	Sri	Mon Nov 18 21:55:07 CST 2019	Mon Nov 18 21:55:07 CST 2019	Update	Report Again
31	Low	Open	Sample Test 20	Maintenance	Sri	Mon Nov 18 22:00:48 CST 2019	Mon Nov 18 22:00:48 CST 2019	Update	Report Again
32	Low	Open	hello	Maintenance	Sri	Tue Nov 19 18:57:05 GMT-06:00 2019	Tue Nov 19 18:57:05 GMT-06:00 2019	Update	Report Again
33	Low	Open	This is a test ticket after deploying to JVM	Maintenance	Sri	Thu Nov 21 00:47:19 GMT-06:00 2019	Thu Nov 21 00:47:19 GMT-06:00 2019	Update	Report Again
34	Low	Open	Toronto	Maintenance	Sri	Thu Nov 21 00:54:31 GMT-06:00 2019	Thu Nov 21 00:54:31 GMT-06:00 2019	Update	Report Again
35	Low	Open	Leaking Roof	Maintenance	Sri	Thu Nov 21 00:55:09 GMT-06:00 2019	Thu Nov 21 00:55:09 GMT-06:00 2019	Update	Report Again
36	Low	Open	Tesstt	Maintenance	Sri	Thu Nov 21 01:05:55 GMT-06:00 2019	Thu Nov 21 01:05:55 GMT-06:00 2019	Update	Report Again

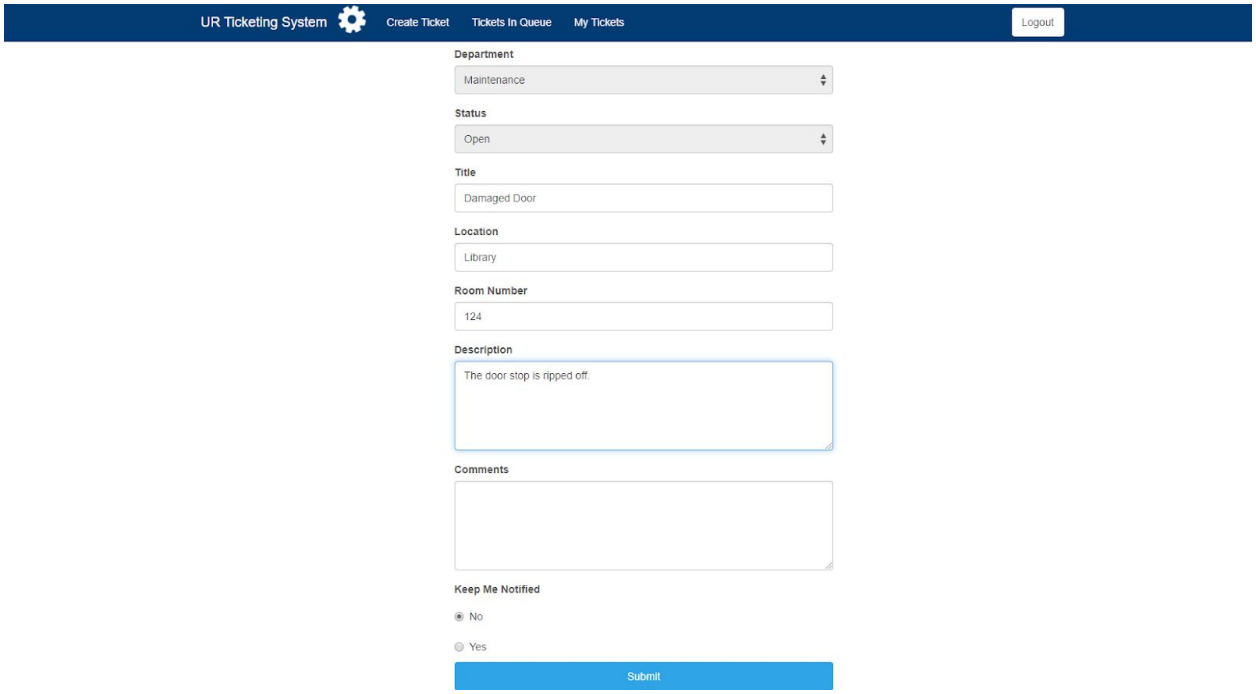
Showing 21 to 30 of 30 entries

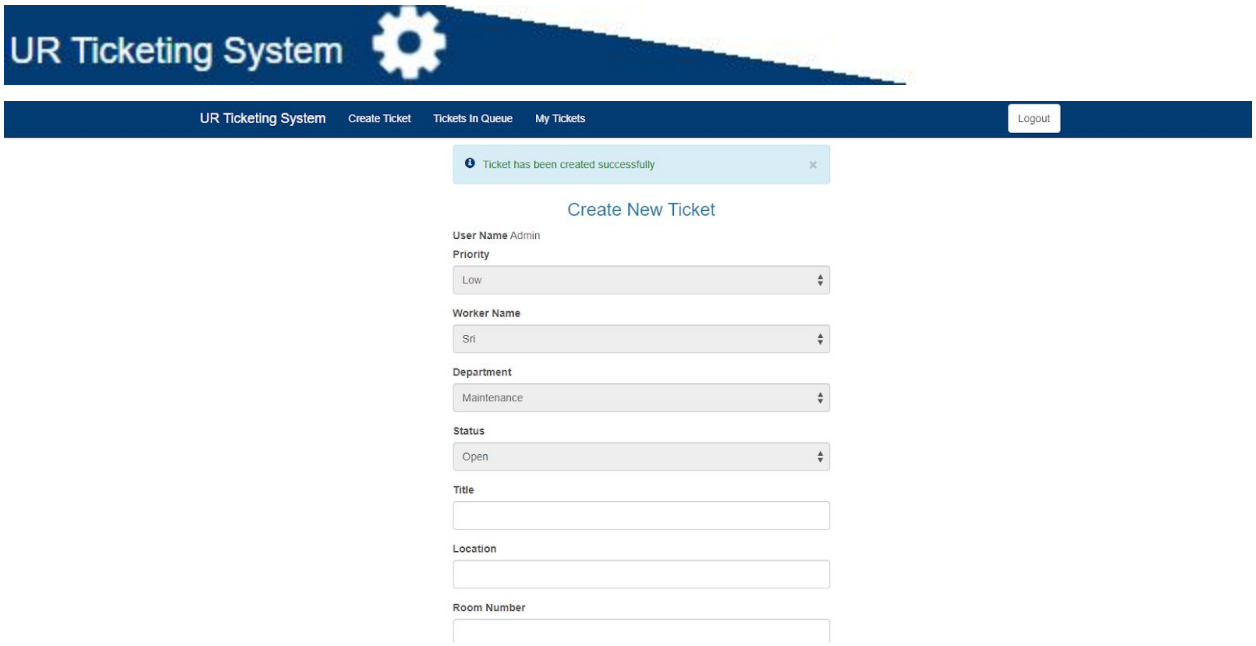
Previous 1 2 3 Next

2) Using “Create Ticket” to make a new ticket:

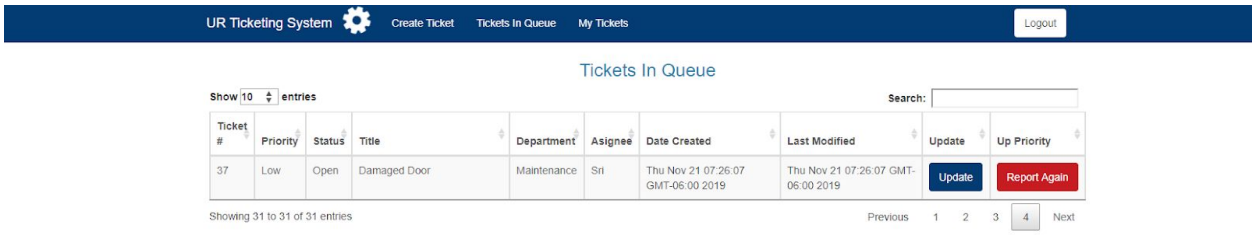


Above picture shows 3 pages of results and that there is 30 tickets.



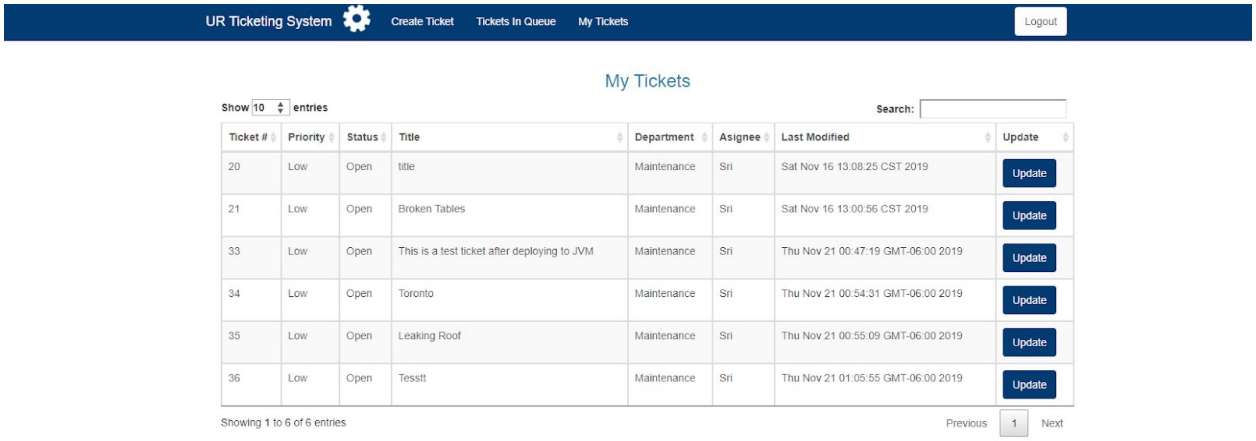


Above picture shows that the ticket was added.

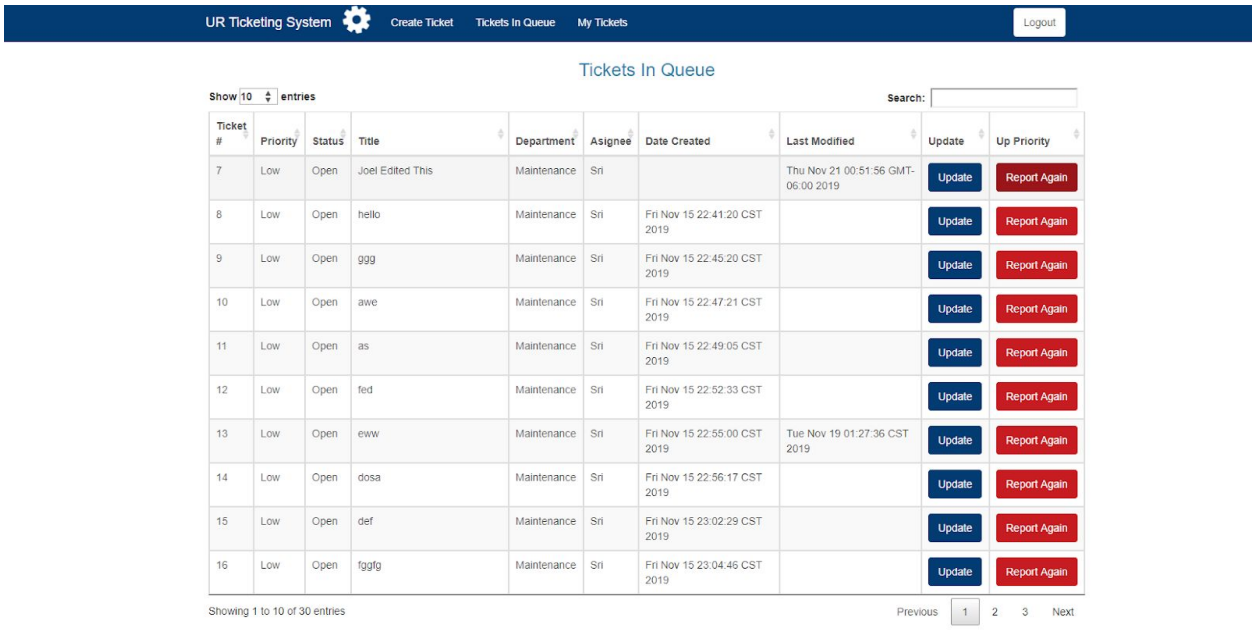


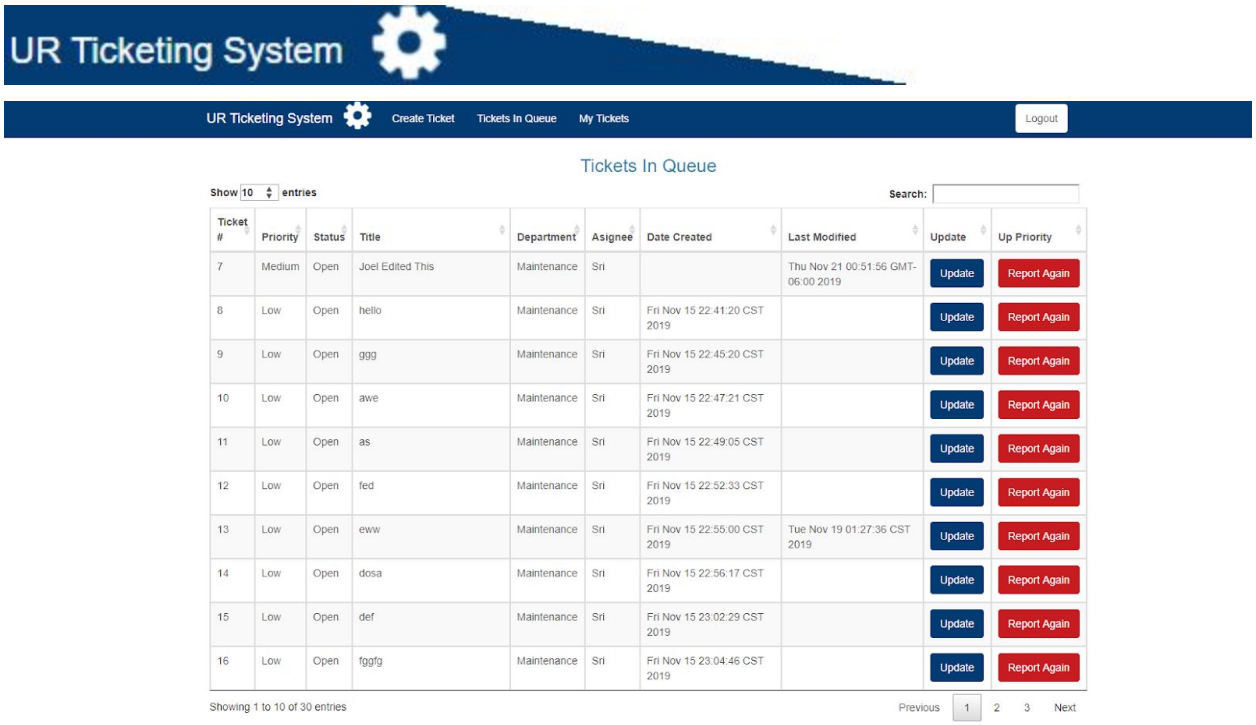
Above picture shows that there is now 31 tickets and 4 pages.

3) “My Tickets” only shows the tickets that the user submitted:

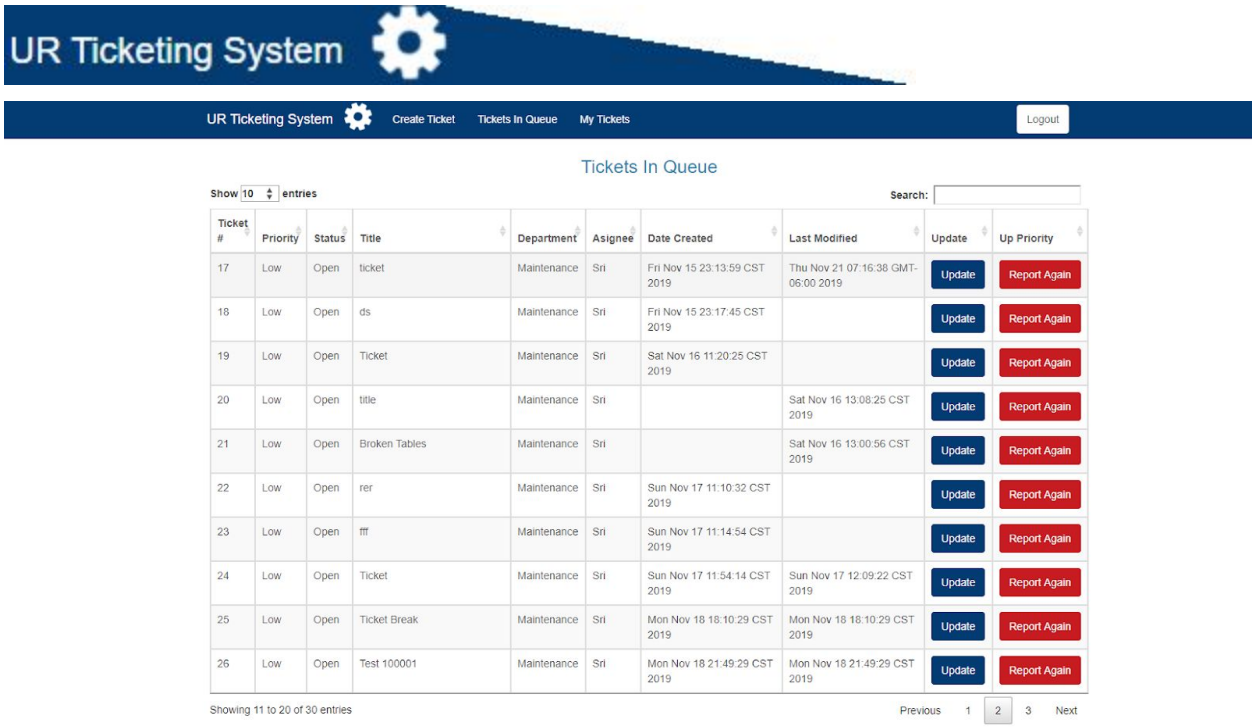


4) “Tickets in Queue” page after 5 diffrent users press “Report Again” to change ticket 7’s priority from “Low” to “Medium”:



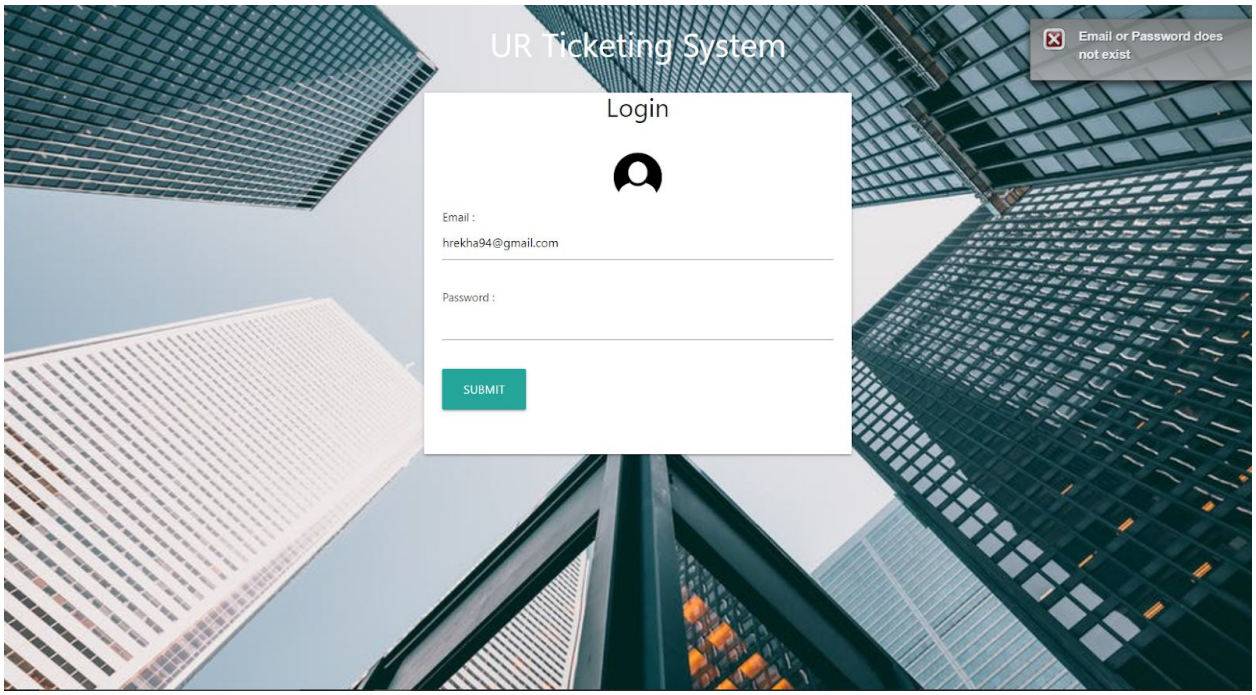


5) “Tickets in Queue” page employee edits ticket 17 and the page now shows when the ticket was last modified:

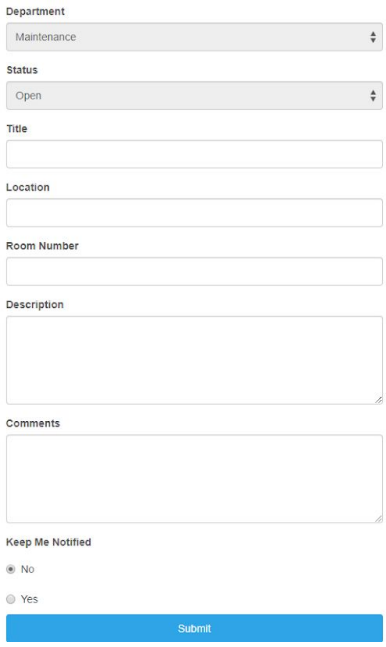


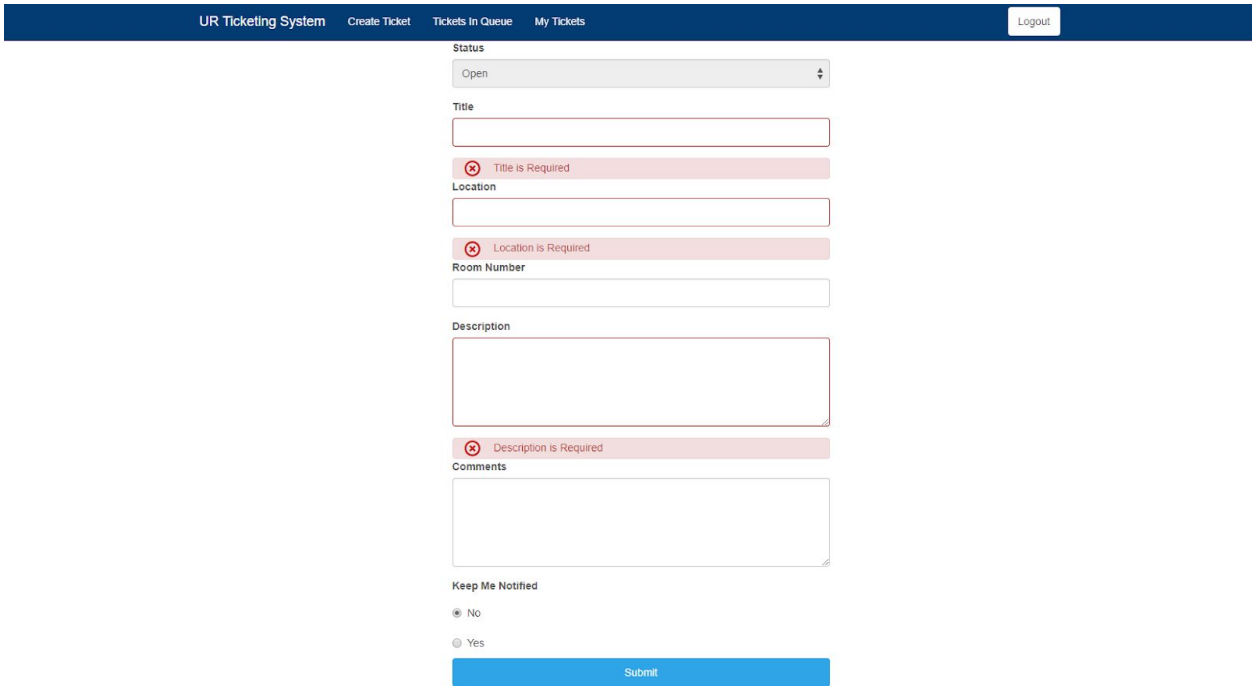
Robustness Testing

1) Incorrect credentials inputted into the login:

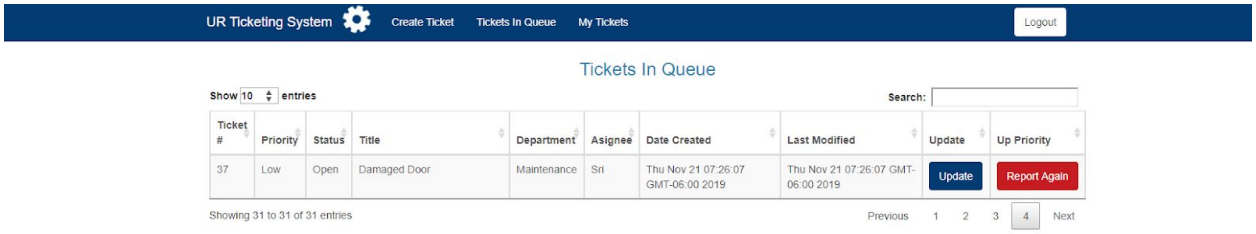


2) “Create Ticket” trying to submit ticket with empty form:

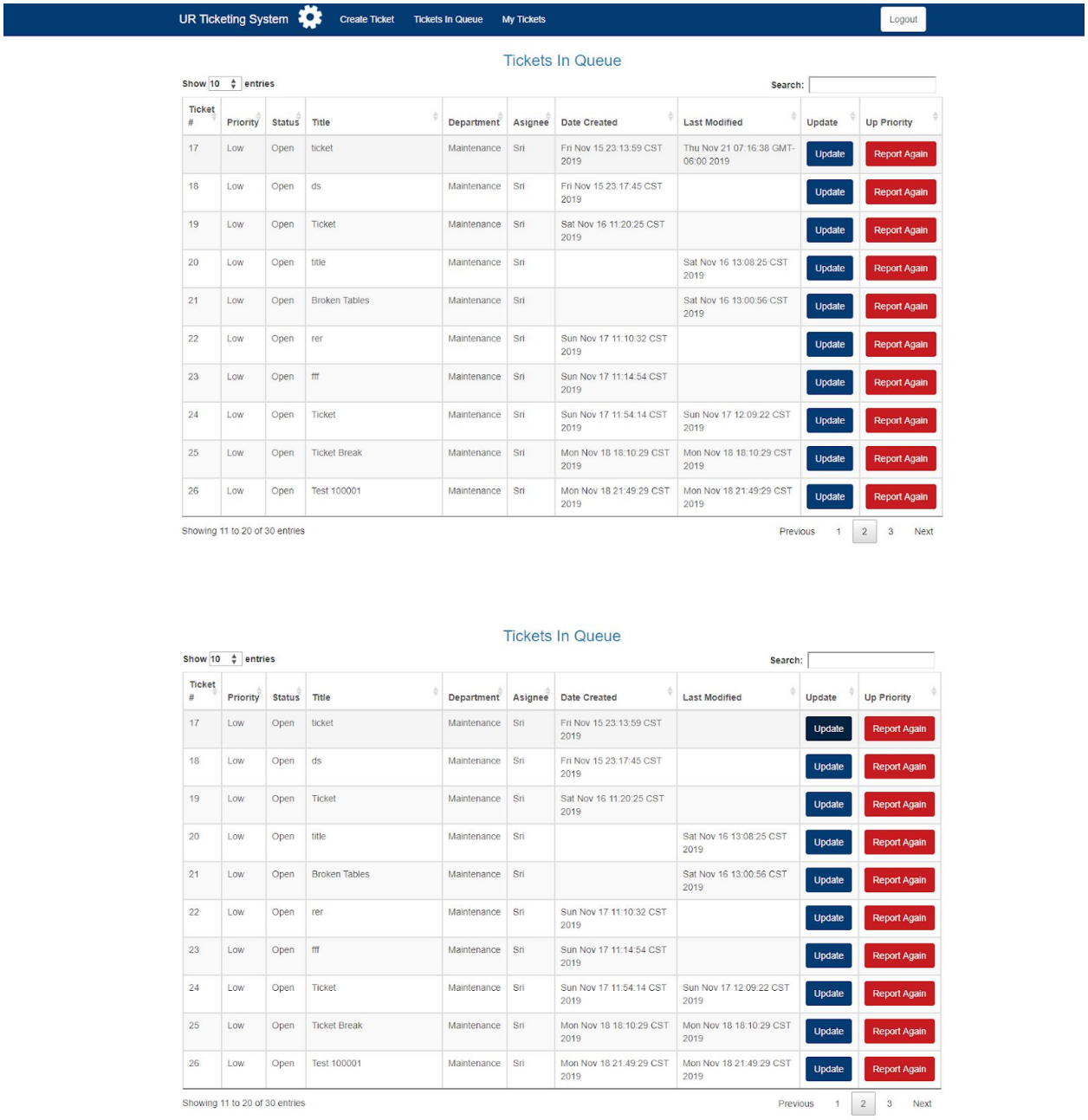




3) “Tickets in Queue” page same user pressing “Report Again” with the priority no changing:



4) “Tickets in Queue” page reloads new changes to the ticket and to the same spot in the list without crashing as seen with ticket 17:



Thank you