Steven Kiley
SE 452
Final Project
Readme

SETUP INSTRUCTIONS:

Provided with the download this readme came in is a directory called *csj*. The function of this web-application depends on the structure of this directory being preserved. Do not attempt to remove any files or change the structure. Inside the directory is the *web.xml* file needed to run this application.

RUN INSTRUCTIONS:

To run, open your *apache-tomcat-7.0.34* folder and place the provided *csj* directory inside the *webapps* folder. The provided file has everything it needs to load and operate.

After placing the *csj* file into the *webapps* folder, open a cmd window. Run the setup script for tomcat if needed. Navigate the command window directory to be *apache-tomcat-7.0.34\bin* then type *startup* into the console. After tomcat initializes, open a Firefox browser window and type the following into the URL bar:

localhost/csj/WideCast

You will now be on the splash page for Wide Cast cable. Hit the *User Login* button at the bottom and log in with one of the accounts (usernames and passwords below) and begin using the available features, which are detailed on the left hand sidebar.

Please note that all user accounts login using the same page. Radio buttons below the *username* and *password* fields allow user roles to be chosen. The credentials for a Customer account can only be used to login while the Customer radio button is checked (which it is by default) and the same is true for each account type.

WORKING REQUIREMENTS:

General Requirements

- Three tiers of cable TV plans available
- Two tiers of internet plans available
- Ten games hardcoded / available
- Ten PPV sports events hardcoded / available
- Twenty PPV movies hardcoded / available
- Useable login feature for all four account types / roles

Customer Requirements

- Update Record feature successfully functions
- Add PPV feature successfully functions
- Rent Game feature successfully functions
- Pay Bill feature successfully functions
- Change Plan feature successfully functions
- Cancel PPV feature successfully functions

Account Specialist Requirements

- Customer Account Creation feature successfully functions
- Update Customer Account feature successfully functions
- Create Incident Ticket feature successfully functions
- Place PPV Order feature successfully functions

Technical Support Requirements

- Schedule Ticket feature successfully functions
- Close Ticket feature successfully functions
- Cancel Ticket feature successfully functions

Manager Requirements

- Cancel Order feature successfully functions
- Update Order feature successfully functions
- Delete Order feature successfully functions

According to the instructions and my assumptions outlined below, there are no requirements that were not fully implemented.

ASSUMPTIONS AND NOTES ABOUT REQUIREMENTS:

The following are assumptions and notes I made about some of the requirements.

General Requirements

- Made use of serialized object files for the login feature, as well as some of the ticket features. This means that the changes made to Customer files, etc are persistent.

Account Specialist

- Assumed that Update Customer account referred to personal information (name, address, etc)
- Create incident ticket automatically assigns the ticket to the technical support person with the lowest number of tickets currently in their queue.

Technical Support

- Close ticket feature requires a final note about what was done to complete the ticket. The entire ticket object is then recorded in a serialized file (location: *csj/web-inf/classes/ResolvedTickets*) named for the incident ticket ID and the Tech that it was assigned to. After being saved to file it is removed from the techs queue.
- Cancel ticket simply removes the ticket without saving to file.

Manager

- All three manager features were assumed to be used on a Customer account. Therefore, all features require a valid Customer username to be entered at the appropriate screen.
- Cancel order leaves the order in the Customer's queue, but zeroes out the price.
- Delete order simply deletes it from the queue.

HOW IT WORKS & GENERAL INFORMATION:

Below is an outline of the pertinent folders and files for this project.

There are quite a few packages and subfolders that are important to the functionality of this web-application. What you'll notice right away is that there are some JSP files immediately in the *csj* folder. These are some of the more generally used JSP files that can be called up during any user role.

In the *web-inf* directory you will find the *web.xml* file needed to configure the container.

The *classes* folder contains the bulk of the code. I chose to leave all of my Servlet files in this directory, though if I had to do it again I would likely place them in the folders containing the JSPs for each login role.

The folder *accounts* contains the *.java* files defining each login role. For customers this contains lists single service items (PPV / game rentals) and recurring service items (cable plan / internet plan). The others are more generic, though the *TechnicalSupport.java* class contains a data structure to store the ticket queue. Though not explicitly named a “bean” file, any of these *.java* classes could be considered a “bean” as they contain getters and setters for each instance variable. Some of them are used like beans in the implementation.

Next we find *accountspecialistpages* which contains the JSP files for the Account Specialist login.

Beans contains *PassKeyBean.java* which is injected into the session at login and used by servlets and JSP pages to determine permission levels for certain activities.

In *customerpages* we find the JSP files for the Customer login.

Loginpages gives us the JSP files for each login role's initial “splash” page.

The folder *managerpages* contains the JSP files for the Manager login role.

PublicUtils contains quite a few files, the bulk of which comprise the backend of the application. Utility files like *Serializer.java* and *Unserializer.java* handle the writing of user accounts to file and reading them back in to the running program. Two *Enum* files (*AccountType & ServiceType)* and two *Interface* files (*Account & Service)* create a level of portability and utility for user accounts. *Ticket.java* and *CustomerData.java* are objects stored within *TechnicalSupport.java* and *Customer.java* respectively.

ResolvedTickets is a storage folder for serialized *Ticket* objects created when a Technical Support login role closes a ticket in their queue.

Similarly, *SerializedAccounts* holds the serialized *Account* objects that store users and the state of their account persistently.

The folder *services* contains three files. *RecurringService.java* is an object for monthly services like cable and internet plans. *SingleServices.java* is an object for one time services like PPV sports events and rentals of games or movies. The third file, *Services.java*, has all available services hard-coded into it and contains static methods that return HashMaps containing the respective service types.

In *technicianpages* we find the JSP files relevant to the Technical Support login role.

The general function of the project follows these steps:

1. User requests a page (like *LoginPage.jsp*)
2. User fills out form on JSP page and hits submit button
3. Form directs to a Servlet file that handles the form data and processes whatever needs to be processed.
4. Once process is complete, the Servlet uses *HttpResonse.sendRedirect(String location)* to redirect the user to the appropriate next JSP page.
5. Process repeats until user has completed all tasks needed.

USER ACCOUNT INFORMATION:

The following are accounts that come by default in *csj/web-inf/classes/SerializedAccounts*

All usernames and passwords are case sensitive.

CUSTOMERS	**Username**	**Password**
Account #1	Zeus	thundergod123
Account #2	MightyJoe	goJoe123!
Account #3	LizzyBorden	axe2grind
Account #4	Gretel345	l0stinDaW00d$

ACCOUNTSPECIALIST	**Username**	**Password**
Account #1	becky	iLoveCable
Account #2	Andy	iHateCable
Account #3	ollie	cablecable

TECH SUPPORT	**Username**	**Password**
Account #1	tech1	mrFixit
Account #2	tech2	mrsFixit
Account #3	Tech3	babyFixit

MANAGER	**Username**	**Password**
Account #1	BigBoss	superImportantGuy
Account #2	littleboss	lessImportantGuy

LINES OF CODE:

Lines of code written for project: 4648.

*Total includes JSP files, Java files, and Servlets.