RTPDEMO Specs:

Functional Requirements:

Basic Functionality:

The Sample real time pricing application will allow the user to get price of a particular product based on quantity required.

Use Case – entry page:

The user interaction will start with a Home page. It will be a simple HTML page that displays a Welcome message. A "Get Price" link from the home page will take the user to a search product screen. Since the search screen is protected user will first get a login screen.

Use Case – Login page:

After user enters the user id and password – redirect to the next page. Display error if id and / or password is null.

Use Case – Search products page:

Allow the user to enter a search string based on product name. The simple search based on product name should be a wildcard search. Based on the search string - a list of products that match the search criterion will be displayed. The list will have 1 row per product along with description.

Allow use to enter requested quantity and provide a button - 'Get price'. This will take the user to the 'price details' page where price & other information will be displayed.

Use Case – get price page:

Based on the user entered product id and quantity, display the current price for that quantity.

Back End Integration:

This application will be used by 5 business units in our 'example' enterprise. Business unit is an attribute of user. For now, we will just create 4 users, 1 per business unit. Our assumption is these business units ALREADY HAVE their own back end systems for product catalog and pricing. We are just providing one common 'Front – End' that will search products and get price – from one of these 'existing' systems – integrating with them in real time. The decision – which system to search products and get price from – is dynamic. Based on the logged in user that is doing the search, we will need to go to different system of records to every time. The user does not need to know which system he/ she is searching.

IF BU = "JDO"

This BU does has a simple database table that stores products. Another table stores prices per product for a range of quantity. For example:

Productid	QTY low	QTY high	Price
abc	1	10	100
abc	11	100	98
abc	101	NULL	97

This means for abc product if quantity is less than 10 price is 100$, 11 – 100 price is 98$ and more than 100 price is 97$.

Retrieve the information from product and this other table.

IF BU = "JDBC"

This business unit has a simple database that has a product table. The product table stores id, name , description of the product as well as the price. The price is fixed regardless of the quantity.

NOTE: For the rest of the Business Units, we do not need to know how this BU stores products and price internally. As the rest of the BU s expose an API or a service to search products and get price. We will just invoke those APIs / services.

For our demo we will 'simulate' the back end system by writing a 'simulated service or API' and potentially hard coding the product as well as price values in the simulated code itself.

IF BU = "EJB"

This BU has an EJB based system and the Session bean exposes two methods GetProduct & GetPrice.

IF BU = "ERP"

This BU has an ERP system. Use Open adapter to call a 'simulated' SAP BAPI (or Xdoc interface).

IF BU = "WS"

This BU exposes a web service. We should consume the web service to get product and price details.

Authentication and Authorization:

User information (id & password) to be validated with the LDAP store. The User in LDAP will have an attribute 'userType'. In userType is buyer only then the user should be allowed to check price, else the button should not appear.

Common Look and feel:

Both the screens should have The Open Stack Logo on top and should have a left margin that provide some dummy external link (common look & feel).

Error Handling:

Search Products page:
If the product does not exist or if quantity cannot be shipped an error should be displayed. These two errors come from the back end systems.

Similarly if user enters 0 or negative quantity an error should be displayed.

Both the errors coming from the back end systems as well as the presentation layer, should be collected together and displayed at once.

Internationalization:

Demonstrate localization (text based), default to US English.

In future: Share price with distributors:

Distributor may query today's price for various products by sending an XML feed with a list of products. Listen on a queue for that feed. Return the price in a separate feed.

Guidelines and Code Conventions:

Coding and naming standards: Follow the current code – enhance if needed.
Guideline: Simplicity , simplicity & simplicity. Avoid in-necessary technical details. We are not defining infrastructure or demonstrating technical strengths.
Its just a quick start – for APPLICATION DEVELOPMENT.