UNIVERSIDAD DE COSTA RICA

ECCI Escuela de Ciencias de la Computación e Informática

Curso: CI-0128 I ciclo 2022
Grupo: 01

Examen final Ingeniería de software

Instrucciones: Este examen es individual, lea cuidadosamente lo solicitado. El proyecto solicitado en la parte 2 debe ser entregado el miércoles 27 de julio antes de las 11:59PM en mediación virtual. Deben entregar los artefactos solicitados en la sección, además de enviar el link del proyecto en github para ser descargado, junto con las instrucciones para poder ser ejecutado en un ambiente local

Part 1 (40%)

Selección única: Responda el formulario de selección múltiple que se encuentra en mediación virtual. Solo cuenta con un intento para responder y 40 min para completar las 30 preguntas, las preguntas se abrirán el 27 de julio a las 8am. Contarán con un intento y podrán responderlas por hasta las 10:30am. Se les mandará un correo una vez esta parte del examen esté habilitada.

Parte 2 (60%)

La segunda parte corresponde a crear un sistema web de una máquina expendedora de refresco. No debe crear una base de datos, pero debe simular una en el sentido que todos los datos deben estar en memoria. Solo debe diseñar una pantalla y no debe generar pantallas de configuraciones extra, solo una que cumpla con lo solicitado en las siguientes user stories:

1. Como consumidor quiero ver los refrescos que están disponibles en la máquina para poder elegir cuales quiero.

   AC: Se debe visualizar el estado actual de los refrescos y su precio. Inicialmente la máquina solo debe tener los siguientes refrescos:

   a. 10 latas de coca cola (precio 500 colones)
   b. 8 latas de pepsi (precio 600 colones)
   c. 10 latas de fanta (precio 550 colones)
   d. 15 latas de sprite (precio 725 colones)

   AC: Cada vez que una compra es completada debe verse el cambio reflejado en la pantalla con la cantidad nueva por tipo de refresco.

2. Cómo consumidor quiero poder ingresar la cantidad de refrescos que quiero de cada tipo para poder realizar una compra

   AC: el sistema debe tener la opción de escoger el tipo y la cantidad de refrescos que se desean. Si el usuario ingresa una cantidad mayor a la que tiene el sistema un mensaje de error debe aparecer

AC: El sistema debe enseñar el costo total de la compra en pantalla, cada vez que se selecciona otro sistema.

3. Cómo consumidor quiero poder ingresar la cantidad de dinero que quiero ingresar en el sistema.
   AC: El sistema debe admitir el pago con monedas de 500, 100, 50 y 25 colones. Además debe aceptar el ingreso de billetes de 1000 colones
   AC: Inicialmente el sistema debe contar con la siguiente cantidad de monedas para dar el vuelto:
   i. 20 monedas de 500 colones
   ii. 30 monedas de 100 colones
   iii. 50 monedas de 50 colones
   iv. 25 monedas de 25 colones
   AC: Si el sistema no tiene más monedas para realizar el vuelto, debe reflejar un mensaje de fuera de servicio.

4. Cómo consumidor quiero ser notificado del vuelto al realizar mi compra para poder saber cual es el resultado final de la compra.
   AC: El sistema debe mostrar en pantalla el monto total que debe devolverle al consumidor. Esto debe mostrarse de la siguiente manera:
   "Su vuelto es de 650 colones.
   Desglose:
   1 moneda de 500
   1 moneda de 100
   1 moneda de 50
   AC: Si el sistema no tiene suficientes monedas para el vuelto de la compra, debe mostrar un mensaje de "Fallo al realizar la compra"

Deberán entregar las siguientes artefactos:

1. Crear un mock up de la visualización de la pantalla, en la misma pantalla deben estar la cantidad de refrescos, la opción para seleccionar los refrescos, el costo total y el vuelto. Además en la misma pantalla debe mostrar los mensajes de error o confirmación. (5%)
2. Modelo del proceso para comprar un refresco (Diagrama UML) (15%)
3. Casos de prueba: Deben crear al menos 3 casos de prueba y ejecutarlos sobre el sistema final (10%).
4. Crear un repositorio y utilizar versionamiento, haciendo uso adecuado del mismo. Haga commits pequeños e individuales y bien documentados. (20%)
5. Código fuente. Se calificará que el código siga en lo posible las ideas de código limpio, además de los principios SOLID y de ser necesario principios de diseño. (30%)

6. Prueba: El código debe tener un buen nivel de cobertura por medio de pruebas unitarias. Además de contar con una prueba funcional automatizada usando selenium (20%)