Plan de pruebas Online Marketplace for Freelancers

1.) Introducción

El sistema de gestión de perfiles de Freelancers es una plataforma diseñada para facilitar la administración y la interacción entre Freelancers y clientes en un entorno seguro y eficiente. Este sistema permite a los Freelancers crear y gestionar sus perfiles, definir niveles de experiencia, y ofrecer sus servicios a clientes que buscan talento en diversas áreas. Además, el sistema integra un módulo de mensajería que facilita la comunicación entre las partes, asegurando que las negociaciones y los proyectos se manejen de manera efectiva y transparente.

Este plan de pruebas tiene como objetivo principal garantizar la calidad del software antes de su implementación definitiva. Se enfocará en verificar que todas las funcionalidades del sistema se ejecuten correctamente, cumpliendo con los requisitos establecidos en los documentos de especificación de requisitos y asegurando que el sistema proporcione una experiencia de usuario satisfactoria y libre de errores.

2.) Objetivos

1. ***Verificar la funcionalidad del sistema:*** Asegurar que todas las funcionalidades del sistema de gestión de perfiles de freelancers, incluyendo la creación y edición de perfiles, la selección de niveles de experiencia, y la mensajería, funcionen correctamente.

2. ***Asegurar la integridad de los datos:*** Confirmar que los datos almacenados y manipulados dentro del sistema, como la información de los perfiles, niveles de experiencia y mensajes, se gestionen de manera coherente y segura.

3. ***Evaluar la interfaz de usuario:*** Confirmar que la interfaz de usuario sea intuitiva, fácil de usar y accesible para todos los tipos de usuarios, garantizando una experiencia de usuario positiva.

4. ***Gestionar errores y excepciones:*** Garantizar que el sistema maneje adecuadamente los errores y excepciones, proporcionando respuestas claras y soluciones cuando se presenten problemas, y evitando caídas o comportamientos inesperados.

5. ***Validar la seguridad del sistema:*** Verificar que el sistema asegure la privacidad y seguridad de los datos, especialmente en lo que respecta al acceso a perfiles y comunicaciones entre usuarios.

3.) Estrategia de Pruebas

Para asegurar la calidad del sistema de gestión de perfiles de freelancers, se seguirá una estrategia de pruebas integral que abarca los siguientes aspectos:

3.1) Tipos de Pruebas

- **3.1.1) Pruebas Unitarias:**
  - Realizadas por los desarrolladores para verificar que cada componente individual del sistema, como la creación de perfiles, selección de niveles de experiencia y mensajería, funcione correctamente. Se utilizarán frameworks de pruebas unitarias específicos para el lenguaje de programación en uso.
- **3.1.2) Pruebas de Integración:**
  - Verificación de la interacción entre los diferentes módulos del sistema, como la integración entre la gestión de perfiles y el sistema de mensajería. Se asegurará que los módulos funcionen correctamente cuando se combinan, y que la integración con sistemas externos, si los hay, se realice de manera fluida.
- **3.1.3) Pruebas de Sistema:**
  - Evaluación del sistema completo para garantizar que cumpla con los requisitos especificados, incluyendo pruebas funcionales y no funcionales. Se probarán casos completos desde la creación de un perfil hasta la comunicación entre usuarios, asegurando que el sistema se comporte como se espera en un entorno similar al de producción.
- **3.1.4) Pruebas de Aceptación del Usuario (UAT):**
  - Realizadas por los usuarios finales, quienes validarán que el sistema cumpla con sus expectativas y necesidades. Se enfocarán en la usabilidad del sistema, asegurando que la interfaz sea intuitiva y que las funcionalidades satisfagan los objetivos del proyecto.

3.2) Enfoque de Pruebas

- **3.2.1) Planificación y Diseño de Pruebas:**
  - Desarrollo de casos de prueba detallados basados en los requisitos especificados, incluyendo diferentes escenarios de uso del sistema. Se definirán criterios de aceptación claros para cada caso de prueba, asegurando que reflejen los objetivos del proyecto.
- **3.2.2) Preparación del Entorno de Pruebas:**
  - Configuración de entornos de prueba que simulen las condiciones de producción, incluyendo la base de datos, interfaces de usuario y cualquier sistema externo necesario. Se garantizará que todos los datos necesarios para las pruebas estén disponibles y sean precisos.
- **3.2.3) Ejecución de Pruebas:**

  - Ejecución sistemática de los casos de prueba diseñados, registrando detalladamente los resultados y cualquier defecto encontrado. Se priorizarán las pruebas que cubran los aspectos críticos del sistema.
- **3.2.4) Gestión de Defectos:**
  - Utilización de herramientas de seguimiento de defectos para registrar, priorizar y gestionar la corrección de errores. Se re-ejecutarán casos de prueba después de la corrección de defectos para asegurar que los problemas se hayan resuelto adecuadamente.
- **3.2.5) Reporte y Evaluación:**
  - Generación de informes de pruebas que resuman los resultados, proporcionando una visión general de la calidad del sistema. Se evaluará la cobertura de las pruebas y la efectividad del proceso de pruebas, recomendando mejoras si es necesario.

3.3) Criterios de Aceptación

- Todos los casos de prueba deben ser ejecutados con éxito.
- Todos los defectos críticos y de alta prioridad deben ser corregidos y verificados.
- El sistema debe cumplir con los requisitos de rendimiento, seguridad y usabilidad especificados.
- Los usuarios finales deben aprobar el sistema durante las pruebas de aceptación del usuario (UAT).

3.1.) Matriz de niveles y atributos de pruebas

Matriz de Niveles vs Atributos de Calidad

La Matriz de Niveles vs Atributos de Calidad es una herramienta que permite evaluar y asegurar la calidad del sistema en diferentes niveles de pruebas. Cada nivel de pruebas se evalúa en función de distintos atributos de calidad, tales como funcionalidad, usabilidad, rendimiento, seguridad, y más. A continuación se presenta la matriz, adaptada al contexto del sistema de gestión de perfiles de freelancers.

Nivel de Pruebas	Funcionalidad	Usabilidad	Rendimiento	Seguridad	Compatibilidad	Mantenibilidad	Portabilidad
Pruebas Unitarias	Verificar que cada unidad (e.g., módulos de perfiles) cumpla con su funcionalidad	No aplicable en este nivel.	No aplicable en este nivel.	No aplicable en este nivel.	No aplicable en este nivel.	Asegurar que el código sea fácilmente comprensible y modificable.	No aplicable en este nivel.

	específica según los requisitos.						
Pruebas de Integración	Validar la correcta interacción entre los módulos del sistema, asegurando que los datos fluyen adecuadamente entre ellos.	Verificar que las interacciones entre módulos no afecten negativamente la usabilidad.	Evaluar el rendimiento de las interacciones entre módulos bajo cargas de trabajo típicas.	Comprobar que la transferencia de datos entre módulos es segura y que no se introducen vulnerabilidades.	Validar que los módulos integrados funcionen en diferentes entornos de software y hardware.	Asegurar que las interacciones entre módulos sean fáciles de modificar y mantener.	Verificar que los módulos integrados pueden ser movidos entre diferentes plataformas sin problemas.
Pruebas de Sistema	Asegurar que el sistema completo cumple con todos los requisitos funcionales.	Validar que el sistema es intuitivo, fácil de usar y que la experiencia del usuario es positiva.	Evaluar el rendimiento del sistema completo bajo diferentes escenarios de uso, incluyendo cargas de trabajo máximas.	Verificar que el sistema cumple con los requisitos de seguridad, protegiendo datos sensibles y garantizando el acceso controlado.	Comprobar que el sistema funciona correctamente en todos los entornos de destino (navegadores, dispositivos, etc.).	Asegurar que el sistema es fácil de mantener y que los cambios futuros no comprometerán su funcionamiento.	Verificar que el sistema puede ser portado a diferentes entornos sin pérdida de funcionalidad.
Pruebas de Aceptación del Usuari	Validar que el sistema cumple con las expectati	Asegurar que el sistema sea fácil de aprender,	Evaluar si el rendimiento del sistema es	Asegurar que los usuarios puedan utilizar el sistema sin	Verificar que el sistema es compatible con los entornos	Asegurar que el sistema es fácil de mantener desde la	Comprobar que el sistema puede ser utilizado

o (UAT)	vas y necesidades de los usuarios finales en términos de funcionalidad.	usar, y que satisface las expectativas de los usuarios.	aceptable desde la perspectiva del usuario final bajo condiciones reales de uso.	comprometer la seguridad y privacidad de sus datos.	de software y hardware utilizados por los usuarios finales.	perspectiva del usuario final, incluyendo actualizaciones y correcciones.	en diferentes plataformas por los usuarios finales sin problemas.

Descripción de los Atributos de Calidad

1. **Funcionalidad:** Evalúa si el sistema cumple con los requisitos funcionales especificados y si cada componente realiza su tarea correctamente.
2. **Usabilidad:** Mide cuán intuitivo y fácil de usar es el sistema, evaluando la experiencia del usuario y la facilidad de aprendizaje.
3. **Rendimiento:** Analiza la eficiencia del sistema bajo diversas condiciones de carga, incluyendo la rapidez de respuesta y el uso de recursos.
4. **Seguridad:** Verifica la protección del sistema contra amenazas y garantiza la confidencialidad, integridad y disponibilidad de los datos.
5. **Compatibilidad:** Asegura que el sistema funcione correctamente en diferentes entornos de hardware, software, y versiones de sistema operativo.
6. **Mantenibilidad:** Evalúa la facilidad con la que se pueden realizar modificaciones, correcciones, y actualizaciones al sistema sin introducir errores.
7. **Portabilidad:** Verifica que el sistema pueda ser trasladado y ejecutado en diferentes plataformas o entornos sin perder funcionalidad.

Propósito de la Matriz

El propósito de esta matriz es garantizar que todos los aspectos clave del sistema sean cubiertos a lo largo de los distintos niveles de pruebas, asegurando que el sistema no solo funcione correctamente, sino que también sea seguro, eficiente, fácil de usar y mantener, y compatible con el entorno en el que será desplegado.

4.) Esquema de Trabajo

- **Diseño de Pruebas:** Se desarrollarán casos de prueba detallados que abarquen todos los escenarios posibles para cada funcionalidad del sistema. Se utilizarán criterios de aceptación basados en los requisitos especificados para asegurar una cobertura completa.
- **Implementación de Pruebas:** Se ejecutarán las pruebas según los casos diseñados, registrando todos los resultados y cualquier defecto encontrado. El registro se realizará en herramientas de gestión de incidencias, como JIRA, para un seguimiento efectivo.
- **Ejecución de Pruebas:** Las pruebas se realizarán tanto de forma manual como automática. Se automatizarán aquellas pruebas que sean repetitivas o de regresión, utilizando herramientas como Selenium para mejorar la eficiencia y la consistencia en los resultados.
- **Reporte de Resultados:** Se generará un informe exhaustivo que incluirá un resumen de las pruebas realizadas, los resultados obtenidos, los defectos encontrados, y las recomendaciones para la corrección de dichos defectos. Este informe será compartido con todo el equipo para facilitar la toma de decisiones.
- **Comunicación y Coordinación:** El proceso seguirá un enfoque ágil, garantizando una comunicación continua y efectiva entre todos los miembros del equipo y el cliente. Se empleará Desarrollo Dirigido por Comportamiento (BDD) para asegurar que tanto los perfiles de negocio como los técnicos colaboren eficazmente, minimizando riesgos y evitando malentendidos que puedan impactar el desarrollo del proyecto.

Roles y Responsabilidades

Titular	Tareas
Product Owner	- Proporcionar visión acerca del sistema y las reglas de negocio. - Colaborar con el equipo para definir los criterios de aceptación.
Scrum Master	- Facilitar la comunicación y colaboración dentro del equipo. - Eliminar obstáculos y garantizar que el proceso de desarrollo fluya sin problemas.
Desarrollador 1 **(Tester Principal)**	- Implementar y ejecutar casos de prueba manuales y automatizados. - Registrar defectos y asegurarse de que se aborden adecuadamente.
Desarrollador 2 **(Desarrollador de Automatización)**	- Desarrollar scripts de automatización para pruebas repetitivas y de regresión. - Mantener el entorno de pruebas automatizadas.

Desarrollador 3 **(Analista de Calidad)**	- Revisar y validar que los requisitos de calidad se cumplan en cada entrega. - Participar en la revisión de código y pruebas de integración.

Comunicación del Equipo y Coordinación con el Cliente

Reuniones Diarias

A través de reuniones diarias (dailys), el equipo de pruebas compartirá el estado del trabajo y discutirá cualquier impedimento. Se identificarán oportunidades para ajustar la estrategia de pruebas según las necesidades del sprint.

Planificación del Sprint

El equipo de pruebas trabajará de cerca con el Product Owner y el Test Manager para definir las historias de usuario, criterios de aceptación, y reglas de negocio que guiarán las pruebas.

Revisión del Sprint

En cada revisión, el equipo presentará las funcionalidades probadas al Product Owner y al equipo de desarrollo. Se invitará al cliente para que pueda ver los avances, proporcionar retroalimentación y, de ser necesario, solicitar ajustes.

Retrospectiva del Sprint

Después de cada sprint, el equipo analizará el proceso de pruebas para identificar áreas de mejora. Se evaluará la efectividad de la estrategia de trabajo y se realizarán ajustes para optimizar los futuros sprints.

Este esquema de trabajo permitirá una evaluación rigurosa y continua de la calidad del sistema, asegurando que el producto final cumpla con las expectativas del cliente y los usuarios finales

5.) Herramientas de Apoyo

Herramientas de Apoyo

En el marco de este plan de pruebas, se utilizarán diversas herramientas y artefactos para garantizar una gestión efectiva de las pruebas y el seguimiento de incidentes. A continuación, se detallan las herramientas seleccionadas y su aplicación específica en el proceso:

- ***JIRA:***
    - **Uso:** Gestión de incidencias, seguimiento de casos de prueba, planificación del trabajo y rastreo de defectos.
    - **Descripción:** JIRA será la herramienta central para la gestión del ciclo de vida de las pruebas. Permitirá a los miembros del equipo de pruebas planificar, asignar y seguir tareas, así como registrar y priorizar defectos encontrados durante la ejecución de las pruebas. Facilitará la comunicación dentro del equipo y con otros stakeholders mediante reportes y dashboards personalizados.
- ***Git/GitHub:***
    - **Uso:** Control de versiones y colaboración.
    - **Descripción:** Git y GitHub se utilizarán para gestionar el control de versiones del código fuente, asegurando que todas las modificaciones y actualizaciones sean rastreadas adecuadamente. Facilitarán la colaboración entre los desarrolladores y el equipo de pruebas, permitiendo la integración continua y la gestión eficiente de ramas y pull requests.

- ***JUnit:***
    - **Uso:** Pruebas unitarias y de integración.
    - **Descripción:** JUnit será la herramienta principal para la creación y ejecución de pruebas unitarias en el código fuente del sistema. Los desarrolladores la utilizarán para verificar el correcto funcionamiento de los

componentes individuales antes de su integración, así como para ejecutar pruebas de integración que validen la interacción entre los diferentes módulos del sistema.

- ***Selenium:***
    - **Uso:** Automatización de pruebas funcionales y de regresión.
    - **Descripción:** Selenium se utilizará para la creación y ejecución de scripts de prueba automatizados que validen la funcionalidad del sistema en distintos navegadores. Esto incluye la grabación, edición y depuración de casos de prueba. La automatización con Selenium permitirá la repetición eficiente de pruebas durante los ciclos de desarrollo, reduciendo el tiempo necesario para realizar pruebas de regresión.
- ***Servidores de Pruebas:***
    - **Uso:** Ejecución de pruebas en un entorno controlado.
    - **Descripción:** Se desplegarán uno o más servidores dedicados para realizar pruebas en un entorno que simule las condiciones de producción. Estos servidores serán configurados para replicar las configuraciones del entorno final, lo que permitirá realizar pruebas de rendimiento, carga y estrés bajo condiciones realistas.
- ***Configuración de Red:***
    - **Uso:** Simulación de entornos de red empresariales y de usuario final.
    - **Descripción:** Se establecerá una red de pruebas que incluirá tanto LAN como conexiones a Internet para simular diferentes escenarios de uso. Esto permitirá realizar pruebas que validen el comportamiento del sistema en diversas condiciones de red, garantizando su robustez y desempeño en entornos reales.

- ***GHERKIN:***
    - **Uso:** Redacción de escenarios de prueba y criterios de aceptación.
    - **Descripción:** Gherkin se utilizará para escribir pruebas en un formato legible por humanos y máquinas, que permita a todos los stakeholders,

incluidos los no técnicos, comprender los criterios de aceptación y los escenarios de prueba. Esto será fundamental para alinear las expectativas entre el equipo de desarrollo, pruebas y los clientes.

- ***Trello:***

  - **Uso:** Gestión de tareas e incidencias.
  - **Descripción:** Trello permitirá al equipo organizar y hacer seguimiento de los casos de prueba, así como de cualquier problema que surja durante el proceso, a través de tableros visuales intuitivos.

- ***Anaconda:***

  - **Uso:** Gestión de entornos y paquetes de Python.
  - **Descripción:** Facilita la creación de entornos virtuales aislados para asegurar que todas las pruebas se ejecuten en condiciones consistentes. Simplifica la instalación y actualización de paquetes necesarios para las pruebas.

- ***VS Code con Live Server:***

  - **Uso:** Editor de código y pruebas rápidas en entorno local.
  - **Descripción:** Visual Studio Code, junto con la extensión Live Server, permitirá ver los cambios en tiempo real durante el desarrollo y las pruebas, facilitando la detección rápida de errores.

- ***Django Testing Framework:***

  - **Uso:** Pruebas unitarias y de integración para aplicaciones Django.
  - **Descripción:** Herramientas integradas en Django que facilitan la creación de pruebas para modelos, vistas y formularios, garantizando que los componentes de la aplicación funcionen correctamente.

- **Jest: (USO TENTATIVO-NO CONFIRMADO)**

- **Uso:** Pruebas unitarias y de integración para JavaScript.
- **Descripción:** Ideal para equipos que desarrollan partes del frontend en JavaScript, permitiendo la creación y ejecución de pruebas de manera sencilla.

- **Pytest: (USO TENTATIVO-NO CONFIRMADO)**
  - **Uso:** Pruebas unitarias y funcionales en Python.
  - **Descripción:** Framework de pruebas simple y potente para escribir y ejecutar pruebas en Python, ofreciendo una sintaxis clara y funcionalidades avanzadas como fixtures y plugins.

- **Postman: (USO TENTATIVO-NO CONFIRMADO)**
  - **Uso:** Pruebas de API.
  - **Descripción:** Permite crear, guardar y ejecutar peticiones HTTP fácilmente, facilitando la validación de servicios web y endpoints de la API.

- **Behave: (USO TENTATIVO-NO CONFIRMADO)**
  - **Uso:** Pruebas de Desarrollo Dirigido por Comportamiento (BDD).
  - **Descripción:** Facilita la redacción de pruebas en lenguaje natural utilizando Gherkin, mejorando la comunicación entre desarrolladores y stakeholders no técnicos.

- **Selenium WebDriver (con Python): (USO TENTATIVO-NO CONFIRMADO)**
  - **Uso:** Automatización de pruebas de interfaz de usuario.
  - **Descripción:** Permite automatizar la interacción con la interfaz de usuario de la aplicación web, realizando pruebas funcionales de extremo a extremo de manera eficiente.

- **Slack/Teams: (USO TENTATIVO-NO CONFIRMADO)**
  - **Uso:** Comunicación y colaboración en tiempo real.
  - **Descripción:** Herramientas de comunicación para coordinar tareas, resolver impedimentos rápidamente y mantener al equipo alineado con los objetivos de las pruebas.

- **TestRail : (USO TENTATIVO-NO CONFIRMADO)**
  - **Uso:** Gestión y documentación de casos de prueba.
  - **Descripción:** Si el equipo lo considera necesario, TestRail puede ser utilizado para organizar y documentar los casos de prueba de manera estructurada.

6.) Tipos de Prueba a Aplicar

Tipo de Prueba	Importancia para el Proyecto
Pruebas de Unidad	Verificar el correcto funcionamiento de cada componente individualmente, asegurando que las funciones y métodos operen como se espera.
Pruebas de Integración	Validar la interacción entre los diferentes módulos del sistema, asegurando que se comuniquen e integren correctamente.
Pruebas de Rendimiento	Comprobar la capacidad del sistema para manejar cargas de trabajo esperadas, maximizando el rendimiento y la eficiencia bajo condiciones de uso real.
Pruebas Funcionales	Verificar que cada funcionalidad del sistema cumple con los requisitos funcionales establecidos, garantizando que el sistema se comporte según lo esperado.
Pruebas de Aceptación del Usuario (UAT)	Involucrar a los usuarios finales para evaluar si el sistema cumple con sus necesidades y expectativas, asegurando que sea útil y relevante en un entorno real.
Pruebas de Seguridad	Asegurar que la información sensible del sistema esté protegida contra accesos no autorizados, vulnerabilidades y amenazas, garantizando la confidencialidad y la integridad de los datos.
Pruebas de Volumen	Comprobar que el sistema pueda manejar grandes volúmenes de datos y usuarios simultáneamente sin afectar el rendimiento o la estabilidad.
Pruebas de Concurrencia	Garantizar que el sistema pueda soportar múltiples usuarios accediendo simultáneamente, asegurando una experiencia consistente y estable para todos los usuarios.
Pruebas de Sistema	Verificar que todos los procesos y funciones del sistema se cumplan correctamente, asegurando que se alcanzan los requisitos generales del proyecto.

Pruebas Alpha/Beta	Identificar potenciales fallas y obtener retroalimentación de los usuarios en entornos controlados (Alpha) y en el mundo real (Beta), antes de la implementación final.

Este conjunto de pruebas cubrirá todas las áreas críticas del sistema, asegurando que se entregue un producto robusto, seguro y funcional que cumpla con las expectativas tanto de los usuarios como del cliente.

7.) Alcance Funcional

En la siguiente Matriz, se muestran las pruebas que cubrirán todos los módulos y funcionalidades del sistema:

PROCESO	SUBPROCESO	FUNCIONALIDAD
REGISTRO Y AUTENTICACIÓN DE USUARIOS	Registro de Freelancers	Verificar que los freelancers pueden registrarse con toda la información necesaria.
	Registro de Clientes	Confirmar que los clientes pueden registrarse con nombre de empresa y ID fiscal.
	Autenticación de Usuarios	Asegurar que los usuarios registrados pueden iniciar sesión correctamente.
	Validación de Contraseñas	Verificar la seguridad en la creación de contraseñas.

	Prueba de Ataques de Fuerza Bruta	Probar mecanismos de protección contra intentos de inicio de sesión repetitivos.
	Cifrado de Datos	Asegurar que los datos sensibles se manejan de forma segura.
PERFILES DE FREELANCER	Creación de Perfil	Verificar que los freelancers pueden ingresar y modificar sus datos personales.
	Gestión de Experiencia	Comprobar que la experiencia se respalda con proyectos previos y se clasifica correctamente.
	Sistema de Calificación	Asegurar que el sistema de calificación funciona correctamente.
	Interfaz de Usuario	Evaluar la facilidad de uso de la interfaz para la gestión del perfil.
	Feedback del Usuario	Verificar que los freelancers pueden responder a críticas y comentarios intuitivamente.
LISTA DE PROYECTOS	Publicación de Proyectos	Verificar que los usuarios pueden ingresar los requerimientos y presupuesto del proyecto.
	Búsqueda de Proyectos	Comprobar que el sistema permite buscar proyectos usando filtros.
	Gestión de Líneas de Tiempo	Asegurar que las líneas de tiempo se ingresan y actualizan correctamente.
	Carga de Búsqueda	Medir el tiempo de respuesta al buscar proyectos con varios filtros aplicados.

	Actualización en Tiempo Real	Probar que las modificaciones en la línea de tiempo se reflejan correctamente.
SISTEMA DE MENSAJERÍA	Iniciar y Recibir Chats	Probar que los usuarios pueden iniciar y recibir mensajes correctamente.
	Operaciones de Archivos	Asegurar que los usuarios pueden adjuntar, enviar, y descargar archivos sin problemas.
	Historial de Chat	Verificar que el historial de chat se almacena, busca y elimina según lo requerido.
	Cifrado de Mensajes	Asegurar que los mensajes y archivos se transmiten de forma segura.
	Gestión de Latencia	Comprobar que el sistema maneja la latencia de red de manera eficiente.
GESTIÓN DE PROYECTOS	Creación y Gestión de Proyectos	Verificar que los usuarios pueden crear y gestionar proyectos.
	Gestión de Tareas	Comprobar que las tareas pueden ser creadas y gestionadas dentro del proyecto.
	Revisiones y Aprobaciones	Asegurar que los entregables pueden ser revisados y aprobados correctamente.
	Comunicación y Colaboración	Probar la integración del sistema de mensajería con la gestión de proyectos.

	Integración de Calendarios	Verificar que el calendario compartido se actualiza y notifica correctamente.
PROCESAMIENTO DE PAGOS	Redirección a Pasarela de Pagos	Verificar que los usuarios son redirigidos correctamente a la pasarela de pagos.
	Confirmación y Actualización de Pagos	Comprobar que el sistema valida correctamente la confirmación del pago.
	Generación y Envío de Comprobantes	Asegurar que los comprobantes de pago se generan y envían automáticamente.
	Validación de Pagos	Probar la seguridad del sistema al validar transacciones de pago.
	Protección contra Fraude	Asegurar que existen medidas para prevenir y detectar transacciones fraudulentas.
NOTIFICACIONES	Envío de Notificaciones	Comprobar que las notificaciones se envían correctamente para eventos clave.
	Gestión de Preferencias	Verificar que los usuarios pueden personalizar sus preferencias de notificación.
	Configuración de Notificaciones	Asegurar que la interfaz para configurar notificaciones es fácil de usar.
PANEL DE CONTROL Y ANÁLISIS DE DATOS	Panel para Freelancers	Verificar que los freelancers pueden visualizar sus proyectos y análisis.

	Panel para Clientes	Asegurar que los clientes pueden acceder a análisis detallados de sus proyectos.
	Análisis de Datos	Comprobar que el sistema realiza correctamente el análisis de tasas de éxito y participación.
	Carga de Datos en el Panel	Medir el tiempo de carga de datos y gráficos en el panel de control.
	Actualización de Análisis	Asegurar que los análisis y métricas se actualizan en tiempo real.

8.) Esfuerzo Estimado

Esfuerzo Estimado

Diseño de Pruebas

Tareas:

- Desarrollo de casos de prueba detallados para cada funcionalidad del sistema.
- Criterios de aceptación basados en los requisitos especificados.

Actividad	Rol	Entregable	Horas

Diseño de Pruebas	Desarrollador 1 (Tester Principal)	Plan de pruebas, Casos de Prueba, Escenarios Gherkin	40
	Desarrollador 2 (Desarrollador de Automatización)		20
	Desarrollador 3 (Analista de Calidad)		20

Implementación de Pruebas

Tareas:

- Ejecución de pruebas manuales y automatizadas.
- Registro de defectos en herramientas de gestión como JIRA.

Actividad	Rol	Entregable	Horas
Implementación de Pruebas	Desarrollador 1 (Tester Principal)	Código fuente y artefactos técnicos	30
	Desarrollador 2 (Desarrollador de Automatización)		30
	Desarrollador 3 (Analista de Calidad)		20

Ejecución de Pruebas

Tareas:

- Ejecución de pruebas manuales y automatizadas.
- Registro de resultados y defectos encontrados.

Actividad	Rol	Entregable	Horas
Ejecución de Pruebas	Desarrollador 1 (Tester Principal)	Ejecución de pruebas	40
	Desarrollador 2 (Desarrollador de Automatización)		30
	Desarrollador 3 (Analista de Calidad)		20

Reporte de Resultados

Tareas:

- Generación de informes detallados sobre pruebas realizadas, resultados, defectos, y recomendaciones.

Actividad	Rol	Entregable	Horas
Reporte de Resultados	Desarrollador 1 (Tester Principal)	Informe de defectos, Informe de cobertura, Informe de rendimiento, Informe de pruebas de seguridad, Informe de pruebas de usabilidad, Informe final de pruebas	20
	Desarrollador 2 (Desarrollador de Automatización)		10
	Desarrollador 3 (Analista de Calidad)		20
	Product Owner		5

Comunicación y Coordinación

Tareas:

- Comunicación continua y efectiva dentro del equipo y con el cliente.

- Desarrollo dirigido por comportamiento (BDD) para asegurar la colaboración eficaz.

Actividad	Rol	Entregable	Horas
Comunicación y Coordinación	Scrum Master	Comunicación continua	15
	Product Owner	Visión del sistema y reglas de negocio	10
	Desarrollador 1 (Tester Principal)	Información de pruebas y defectos	10
	Desarrollador 2 (Desarrollador de Automatización)	Scripts de automatización	5
	Desarrollador 3 (Analista de Calidad)	Revisión de calidad	10

Cronograma

Actividad	Inicio	Fin
Planificación Inicial de Pruebas	20/08/2024	31/08/2024
Diseño de Pruebas	04/11/2024	10/11/2024
Implementación de Pruebas	11/11/2024	18/11/2024
Ejecución de Pruebas	19/11/2024	30/11/2024
Reporte de Resultados	01/12/2024	05/12/2024

9.) Entregables del proceso

Entregables del Proceso de Pruebas

1. **Plan de Pruebas**
   - ✓ **Descripción**: Documento que detalla la estrategia global de las pruebas para el proyecto. Incluye los objetivos, alcance, criterios de entrada y salida, y los recursos necesarios. Define los métodos de prueba a seguir y los procedimientos para asegurar que todas las funcionalidades sean verificadas adecuadamente.
2. **Casos de Prueba**
   - ✓ **Descripción**: Documentos que describen los pasos específicos para probar cada funcionalidad del sistema. Incluyen los datos de entrada, las acciones a realizar y los resultados esperados para cada prueba. Estos casos guiarán a los testers en la validación de los requisitos del sistema.
3. **Ejecución de Pruebas**
   - ✓ **Descripción**: Resultados documentados de las pruebas realizadas. Incluye logs (registros de pruebas), capturas de pantalla, y cualquier evidencia generada durante la ejecución de las pruebas. Para pruebas automatizadas, se utilizarán herramientas como Selenium y JUnit para capturar estos resultados.

4. **Informe de Defectos**
    - ✓ **Descripción**: Registro de todos los defectos encontrados durante la ejecución de las pruebas. Incluye detalles sobre cada defecto, su gravedad, pasos para reproducirlo y el estado de la resolución. Facilita la comunicación de problemas encontrados al equipo de desarrollo.
5. **Informe de Cobertura**
    - ✓ **Descripción**: Documento que detalla qué partes del código fuente han sido cubiertas por las pruebas. Incluye métricas sobre la cobertura de código para asegurar que todas las funcionalidades del sistema han sido verificadas.
6. **Informe de Rendimiento**
    - ✓ **Descripción**: Resultados de las pruebas de rendimiento, incluyendo tiempos de respuesta, uso de recursos y posibles cuellos de botella identificados. Este informe ayuda a evaluar la eficiencia y capacidad del sistema bajo condiciones de carga.
7. **Informe de Pruebas de Seguridad**
    - ✓ **Descripción**: Documentación de las pruebas de seguridad realizadas, incluyendo vulnerabilidades encontradas y recomendaciones para su mitigación. Asegura que el sistema sea seguro y cumpla con los requisitos de protección de datos.
8. **Informe de Pruebas de Usabilidad**
    - ✓ **Descripción**: Resultados de las pruebas realizadas con usuarios finales. Incluye la experiencia del usuario, accesibilidad, y recomendaciones para mejorar la interfaz y la usabilidad del sistema.
9. **Informe Final de Pruebas**
    - ✓ **Descripción**: Resumen completo del proceso de pruebas, que incluye un compendio de todos los resultados obtenidos, defectos identificados, lecciones aprendidas y recomendaciones finales. Sirve como documento de cierre del ciclo de pruebas.
10. **Product Backlog**
    - ✓ **Descripción**: Lista priorizada de todas las funcionalidades, características y requisitos del producto. Este backlog se utiliza para guiar el desarrollo y las

pruebas del sistema, asegurando que se cubran todas las necesidades del proyecto.

11. **Historias de Usuario**
    - ✓ **Descripción**: Descripciones detalladas de las funcionalidades desde la perspectiva del usuario. Las historias de usuario guían el diseño y la ejecución de pruebas, asegurando que el sistema cumpla con las expectativas del usuario final.
12. **Código Fuente y Artefactos Técnicos**
    - ✓ **Descripción**: Incluye el código fuente de las pruebas desarrolladas, scripts de automatización, configuraciones y cualquier documentación técnica relevante para el proceso de pruebas.
13. **Informes de Sprint**
    - ✓ **Descripción**: Documentación de los logros de cada Sprint, incluyendo los obstáculos encontrados y las lecciones aprendidas. Proporciona una visión general del progreso del proyecto y ayuda en la planificación de futuras iteraciones.
14. **Escenarios Gherkin**
    - ✓ **Descripción**: Requisitos y funcionalidades descritas utilizando el lenguaje Gherkin. Facilita la comprensión y el desarrollo de pruebas basadas en comportamiento (BDD), asegurando que los criterios de aceptación estén claramente definidos.

Estos entregables garantizarán una cobertura completa de las pruebas, facilitando la gestión de calidad y asegurando que el sistema cumpla con los requisitos especificados.

10.) Mecanismos de seguimiento y control

Mecanismos de Seguimiento y Control

1. **Indicadores de Desempeño**

- **Cantidad de Ejecución de Casos de Prueba Correctas y Fallidas:**
  - ✓ **Objetivo**: Evaluar la efectividad de las pruebas realizadas.
  - ✓ **Métrica**: Número de casos de prueba ejecutados con éxito frente a los que fallaron. Se medirá la tasa de éxito y los tipos de errores encontrados.
- **Información sobre Defectos:**
  - ✓ **Objetivo**: Analizar la calidad del software.
  - ✓ **Métricas**:
    - ❖ **Densidad de Errores**: Número de defectos encontrados por módulo o funcionalidad.
    - ❖ **Frecuencia de Fallos**: Número de defectos reportados por semana.
- **Clasificación de Errores por Importancia:**
  - ✓ **Objetivo**: Priorizar la corrección de defectos.
  - ✓ **Métrica**: Número de errores clasificados como alta, media y baja importancia, facilitando la gestión y resolución de los más críticos primero.
- **Requerimientos Elaborados y Aprobados:**
  - ✓ **Objetivo**: Asegurar que todos los requerimientos del sistema se han cubierto.

- ✓ **Métrica**: Número de requerimientos documentados frente a los aprobados, para verificar que todas las funcionalidades solicitadas han sido abordadas en las pruebas.

2. **Revisión y Reportes Periódicos**
   - **Informe Semanal de Progreso:**
     - ✓ **Contenido**: Resumen de actividades realizadas, casos de prueba ejecutados, defectos encontrados y corregidos, y cualquier desviación del plan.
     - ✓ **Destinatarios**: Product Owner, Scrum Master, y todo el equipo de desarrollo.
   - **Informe Diario en Reuniones de Stand-Up:**
     - ✓ **Contenido**: Actualización rápida del estado de las pruebas, problemas encontrados, y necesidades de soporte.
     - ✓ **Destinatarios**: Todo el equipo de pruebas y desarrollo.
   - **Informe de Cierre de Sprint:**
     - ✓ **Contenido**: Resultados de las pruebas del sprint, análisis de defectos, lecciones aprendidas y recomendaciones para el siguiente sprint.
     - ✓ **Destinatarios**: Todo el equipo de desarrollo y stakeholders relevantes.
3. **Gestión de Incidencias**
   - **Uso de JIRA:**
     - ✓ **Funcionalidad**: Registrar, seguir y gestionar incidencias. Cada defecto se documentará con detalles, pasos para reproducir, prioridad y asignación de responsable.
     - ✓ **Flujo de Trabajo**: Cada defecto será registrado, evaluado y asignado a los responsables correspondientes para su resolución.
   - **Revisión de Incidencias Críticas:**

- ✓ **Objetivo**: Priorizar la resolución de defectos críticos que bloquean el progreso.
- ✓ **Frecuencia**: Diaria o según sea necesario.
- ✓ **Responsables**: Tester Principal y el equipo de desarrollo.

4. **Auditorías de Calidad**
   - **Revisión de Casos de Prueba:**
     - ✓ **Objetivo**: Asegurar que los casos de prueba cubren todos los escenarios y requisitos.
     - ✓ **Frecuencia**: Antes de la ejecución de pruebas principales.
     - ✓ **Responsables**: Tester Principal y Analista de Calidad.
   - **Revisión de Documentación:**
     - ✓ **Objetivo**: Verificar la completitud y precisión de los documentos entregables.
     - ✓ **Frecuencia**: Al finalizar cada fase de pruebas.
     - ✓ **Responsables**: Tester Principal y Scrum Master.
5. **Reuniones de Seguimiento**
   - **Reunión de Seguimiento Semanal:**
     - ✓ **Contenido**: Revisión de avances, análisis de métricas e indicadores, discusión de problemas y ajustes al plan de pruebas.
     - ✓ **Participantes**: Product Owner, Scrum Master, Tester Principal, Desarrollador de Automatización y Analista de Calidad.
   - **Reunión de Revisión Post-Mortem:**
     - ✓ **Contenido**: Evaluación del proceso de pruebas, identificación de áreas de mejora, y planificación de acciones correctivas para futuros proyectos.
     - ✓ **Participantes**: Todo el equipo de pruebas y desarrollo, junto con stakeholders clave.

6. **Automatización y Monitoreo Continuo**
    - **Automatización de Pruebas Repetitivas:**
        - ✓ **Objetivo**: Incrementar la eficiencia y reducir errores manuales.
        - ✓ **Herramientas**: Selenium y JUnit.
    - **Monitoreo de Ejecuciones de Pruebas Automatizadas:**
        - ✓ **Objetivo**: Detectar fallos de manera inmediata y permitir una respuesta rápida.
        - ✓ **Herramientas**: Integración continua con herramientas como Jenkins.

Implementación de los Mecanismos

1. **Preparación:**
    - Configuración de JIRA y herramientas de automatización.
    - Definición de indicadores de desempeño y creación de plantillas de informes.
2. **Ejecución:**
    - Registro continuo de incidencias y ejecución de casos de prueba.
    - Generación de informes semanales y diarios.
3. **Evaluación:**
    - Revisión de métricas e indicadores.
    - Realización de auditorías y reuniones de seguimiento.
4. **Mejora Continua:**
    - Análisis de lecciones aprendidas y ajustes al proceso de pruebas.
    - Implementación de recomendaciones y mejoras para futuras iteraciones.