1. Kundenauftrag

**Firma:** Musterfirma GmbH  
**Abteilung:** IT-Entwicklung  
**Kunde:** Entwicklungsabteilung / Qualitätssicherung  
**Projekt:** Smart Log & Bug Tracker

**Auftrag:**  
Die Entwicklungsabteilung benötigt ein Tool, um Logdateien automatisiert zu analysieren und Fehler, Warnungen oder Auffälligkeiten direkt in ein Bug-Tracking-System zu überführen. Ziel ist es, die Auswertung von Logs zu beschleunigen und gleichzeitig die Dokumentation von Bugs zu verbessern.

**Umsetzung:**  
Das Tool soll als C#-Desktop-Anwendung mit Datenbankanbindung entwickelt werden. Es soll sowohl für Entwickler als auch für Tester nutzbar sein.

**Zeitrahmen:** 6–8 Wochen

**Verantwortlich:** [Dein Name] – Auszubildender Fachinformatiker Anwendungsentwicklung

**2. Lastenheft (vom Kunden formuliert)**

**Zielsetzung**

* Automatisierte Analyse von Logdateien
* Vereinfachte Dokumentation und Verwaltung von Software-Fehlern
* Zentrale Übersicht über Bugs und deren Status

**Muss-Kriterien (Pflicht)**

1. **Logdateien einlesen** (.log, .txt)
2. **Filtermöglichkeiten** (Datum, Severity, Quelle)
3. **Statistik** über Fehler/Warnungen/Infos
4. **Exportfunktion** (CSV und PDF)
5. **Ticketsystem**
   * Bug anlegen mit Titel, Beschreibung, Status, Priorität
   * Zuweisung an Entwickler
   * Suche/Filterung nach Bugs
6. **Benutzerverwaltung** (Rollen: Admin, Entwickler, Tester)

**Kann-Kriterien (Optional)**

* Automatische Erkennung häufiger Fehler (Mustervergleich)
* Diagramme zur Auswertung (z. B. Fehler nach Zeit, Komponente)
* E-Mail-Benachrichtigung bei neuen Tickets

**Nicht-Ziele**

* Kein webbasiertes System (nur Desktop-App)
* Keine Anbindung an externe Systeme (z. B. Jira, GitLab)

**Rahmenbedingungen**

* Entwicklungsumgebung: Visual Studio
* Programmiersprache: C# (.NET 6 oder höher)
* Datenbank: SQLite oder SQL Server Express
* Plattform: Windows 10/11

**3. Pflichtenheft (vom Entwickler ausgearbeitet)**

**Systemarchitektur**

* **Frontend (GUI):** WPF (Windows Presentation Foundation)
* **Backend:** C# Klassenbibliothek (LogParser, TicketManager, UserManager)
* **Datenbank:** SQLite mit Tabellen (Users, Tickets, Logs)
* **Reporting:** iTextSharp (PDF), CSV-Export mit .NET-Bibliotheken

**Funktionale Umsetzung**

1. **Log-Analyzer**
   * Parser liest Logdateien zeilenweise ein
   * Regex erkennt Zeitstempel, Severity, Quelle, Nachricht
   * Ergebnisse in GUI-Tabelle anzeigen
   * Filterfunktionen (Datum, Severity, Quelle)
   * Markierte Logeinträge → „Bug erstellen“
2. **Bug-Tracking**
   * Ticketformular (Titel, Beschreibung, Status, Priorität, Zuweisung)
   * Daten in SQLite speichern
   * Tickets in Listenansicht darstellen
   * Filter und Suche (z. B. nach Status oder Entwickler)
   * Export von Buglisten (CSV/PDF)
3. **Benutzerverwaltung**
   * Login mit Benutzername/Passwort
   * Rollen: Admin (alle Rechte), Entwickler (Tickets bearbeiten), Tester (Tickets erstellen)
4. **Reporting & Statistik**
   * Balkendiagramm: Fehler pro Severity
   * Kreisdiagramm: Verteilung offener/geschlossener Bugs
   * Export als PDF oder CSV

**Nicht-funktionale Anforderungen**

* Benutzerfreundliche Oberfläche
* Performantes Einlesen auch bei großen Logdateien (10.000+ Zeilen)
* Datenbankdatei lokal verschlüsselt

**4. Soll/Haben-Vergleich**

|  |  |  |
| --- | --- | --- |
| Kriterium | Soll (Anforderung) | Haben (Umsetzung) |
| Logdateien einlesen | .log und .txt Dateien einlesen | ✔ Implementiert mit StreamReader in C# |
| Filterfunktionen | Datum, Severity, Quelle | ✔ GUI-Filter mit Linq-Ausdrücken |
| Statistik | Fehler/Warnungen/Infos zählen | ✔ Darstellung + Diagramme |
| Export | CSV + PDF | ✔ CSV via .NET, PDF via iTextSharp |
| Ticketverwaltung | Erstellen, Bearbeiten, Status, Priorität | ✔ Ticket-Modul mit SQLite |
| Ticket-Zuweisung | An Entwickler | ✔ Rollensystem implementiert |
| Benutzerverwaltung | Rollen: Admin, Entwickler, Tester | ✔ Login-System mit Passwort-Hashing |
| Erweiterte Analyse (Optional) | Mustererkennung häufiger Fehler | ✖ Nicht umgesetzt (nur Grundfunktionen) |
| E-Mail-Benachrichtigung | Versand neuer Tickets per Mail | ✖ Nicht umgesetzt |
| Plattform | Windows 10/11 Desktop-App | ✔ WPF-App |