**Lastenheft**

1. Funktionale Anforderungen
2. Weiche Anforderungen
3. Qualitätsanforderungen
4. Risikomanagement
5. Mögliche Erweiterungen
6. **Funktionale Anforderungen**

* Jeder Benutzer kann ein Ticket lösen.
* Jeder Benutzer kann ein Ticket zum Bezahlen aus der Ticketliste ausgewählt werden.
* Jeder Benutzer kann ein Ticket zum Bezahlen auswählen, sobald die Details angezeigt werden.
* Jeder Benutzer kann Münzen einwerfen.
* Jeder Benutzer kann eine Parkmünze erhalten.
* Jeder Benutzer kann den aktuellen Geldstand nachschauen.
* Jeder Benutzer kann den aktuelle Tickets anschauen.
* Jeder Benutzer kann die Ticketdetails anzeigen.
* Jeder Benutzer kann die Parkmünzen nachschauen.
* Jeder Benutzer kann die Parkmünzendetails nachschauen.
* Jeder Benutzer kann die Bezahlung eines Tickets abbrechen.
* Jeder Benutzer kann ein Ticket löschen.
* Jeder Benutzer kann die Einstellungen öffnen.
* Jeder Benutzer kann die App durch den „Zurück“ Knopf oder den „Home“ Knopf beenden.
* Jeder Benutzer kann in den Einstellungen Geld verdienen gehen.
* Jeder Kunde kann in den Einstellungen auf die Rolle eines Mitarbeiters wechseln.
* Jeder Mitarbeiter kann in den Einstellungen auf die Rolle des Kunden wechseln.
* Ein Mitarbeiter kann die Anzahl der Parkmünzen und Geldmünzen im Automaten einstellen.
* Ein Mitarbeiter ist sonst ein ganz normaler Benutzer.
* Ein Mitarbeiter kann zu Testzwecken Tickets ungültig machen.
* Ein Mitarbeiter muss einen Namen haben.
* Jegliche Veränderung an Münz/- oder Geldstand wird dokumentiert mit dem Namen des Mitarbeiters.
* Die Daten des Benutzers und der Mitarbeiter sollen in einer Datenbank persistiert werden.
* Es gibt nur einen Benutzer.
* Es kann beliebig viele Mitarbeiter geben.
* Falls die App verlassen wird, soll der aktuelle Zustand gespeichert werden und beim Neustarten wieder geladen werden.
* Im Falle eines Absturzes sollen so viele Daten wie möglich persistiert sein/werden.
* Der aktuelle Benutzer muss zurücksetzbar sein, dies gilt nicht für Mitarbeiter.
* Mitarbeiter können aus dem Speicher gelöscht werden.
* Alle Mitarbeit müssen angezeigt werden können.
* Es müssen Debugging Optionen für die Zeit, den Zustand des Automaten und den Zustand des Benutzers bestehen.

1. **Nicht-Funktionale Anforderungen**

* Der Benutzer soll bei keiner Aktion eine deutliche Verzögerung spüren (> 1 s).
* Der Benutzer sollte alle GUI Elemente ohne Umgreifen erreichen können.
* Die App sollte in weniger als 5 Sekunden starten.

1. **Qualitätsanforderungen**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Muss** | **Soll** | **Nicht relevant** |
| Funktionalität | x |  |  |
| Zuverlässigkeit | x |  |  |
| Benutzbarkeit | x |  |  |
| Effizienz |  | x |  |
| Individualisierung |  |  | x |

1. **Risikomanagement**

|  |  |  |  |
| --- | --- | --- | --- |
| **Risiko** | **Wahrscheinlichkeit** | **Schaden** | **Gegenmaßnahme** |
| Einstiegsprobleme mit Werkzeugen | 80% | 5% | Gegenseitige Schulung. |
| Fortgeschrittene Probleme mit Werkzeugen | 10% | 20% | Intensiveres Helfen bei den Programmen und als letzte Lösung ein Umstieg auf andere Werkzeuge. |
| Datenverlust | 5% | 90% | Wöchentliches Backup des master Branches auf eine oder zwei Festplatten. |
| Umsetzung Datenbank schlägt fehle | 5% | 20% | Einen anderen geeigneten Datenspeicher finden. Z.B. Textdatei oder Applikationsspeicher. |
| Umsetzung Zustandsautomat schlägt fehl | 40% | 60% | Umbau des Softwarekonzepts ohne einen Zustandsautomaten. |

1. **Mögliche Erweiterungen**

* Kreditkartendaten angeben. Der Benutzer muss darauf hingewiesen werden, dass er für die Angabe von Kreditkartendaten einen Keks bekommt.
* Mehr Kunde als einen zulassen.
* Kunde und Mitarbeiter durch Passwörter schützen.
* Animationen für erhalten einer Münze bzw. eines Parkscheins.
* Mehrfach Ticketlöschen möglich.
* Soundeffekte beim Ausführen von Aktionen
  + Lautstärkeregelung in den Einstellungen vornehmen.