**Wallpaper Generator**

**ÜK-335 25/4256**

**Mobile-Applikation realisieren**

|  |  |
| --- | --- |
| Autor | Simon Streuli |
| Erstelldatum | **20.05.2025** |
| Abgabedatum | **21.05.2025** |

Inhaltsverzeichnis

[1. Einführung 3](#_Toc198663347)

[1.1 Ziel der Applikation 3](#_Toc198663348)

[1.2 Motivation 3](#_Toc198663349)

[1.3 Zielpublikum 3](#_Toc198663350)

[2. Rahmenbedingungen 4](#_Toc198663351)

[3. Storyboard 5](#_Toc198663352)

[4. Anforderungen 6](#_Toc198663353)

[5. Testkonzept 8](#_Toc198663354)

[6. Aufbau des Systems 11](#_Toc198663355)

[7. Reflexion 11](#_Toc198663356)

# 1. Einführung

Das Projekt wurde im Rahmen der Projektarbeit im Modul 335 entwickelt.

### Ziel der Applikation

Meine App verfolgt das Ziel, individuelle Hintergrundbilder auf Basis von Nutzerprompts mithilfe künstlicher Intelligenz zu generieren. Der Nutzer gibt eine Beschreibung ein, woraufhin die App über die OpenAI API ein passendes Bild erzeugt. Dieses kann dann gespeichert oder direkt als Hintergrundbild gesetzt werden.

### 1.2 Motivation

Ich selbst habe ein paar Wallpaper-Apps probiert und eigentlich alle stecken hinter einer Paywall oder bestehen nur aus vorgefertigten Bildern. Meine Applikation hingegen, bietet eine moderne Lösung, die es ermöglicht, kreative und einzigartige Hintergrundbilder zu erzeugen und das alles gratis.

### 1.3 Zielpublikum

Die App richtet sich an Android User, die gerne personalisierte und innovative Backgrounds verwenden wollen, sowie technikaffine Personen, die gerne KI-Funktionalitäten ausprobieren.

# 2. Rahmenbedingungen

Im Rahmen der Projektarbeit nach Modul 335 müssen mindestens **10 technische Punkte** durch bestimmte Funktionen erreicht werden. Mein Projekt erfüllt diese Anforderungen wie folgt:

|  |  |  |
| --- | --- | --- |
| Kriterium | Punkte | Umsetzung |
| Mehr als eine Aktivität oder View verwenden | 1 | Startbildschirm, Prompt-Eingabe, Bildanzeige |
| Kommunikation zwischen zwei Aktivitäten oder Views | 1 | Übergabe des Nutzereingabetexts und Rückgabe des generierten Bildes |
| Verwenden einer persistenten, lokalen Datenablage | 2 | Speicherung von generierten Bildern oder Favoriten |
| Verwenden von Hintergrundoperationen (z. B. Services, Notifications, …) | 4 | Bildgenerierung erfolgt asynchron im Hintergrund per API-Aufruf |
| Externe Schnittstelle lesen | 2 | Bildgenerierung über eine KI-API wie DALL·E/OpenAI |
| Interaktion mit Systemapplikationen (z. B. Galerie, Wallpaper setzen) | 2 | Nutzer kann das Bild direkt als Hintergrundbild setzen |

**Gesamtpunktzahl: 12 Punkte ✅**

Damit ist die Grundlage für ein komplexes und praxisorientiertes Mobile-Projekt gelegt.

# 3. Storyboard

Das Storyboard veranschaulicht die zentralen Abläufe der App aus Sicht eines neuen Nutzers. Es zeigt Schritt für Schritt, wie ein User, der die App zum ersten Mal öffnet, ein personalisiertes Hintergrundbild mit Hilfe der KI generiert.

Von der Eingabe des API-Schlüssels über die Beschreibung des gewünschten Bildes bis hin zur Generierung und Speicherung des Wallpapers wird der gesamte Nutzerfluss visuell dargestellt.

Ein Bild, das Text, Screenshot, Handy, mobiles Gerät enthält.

KI-generierte Inhalte können fehlerhaft sein.

Ein Bild, das Text, Handy, Screenshot, Gerät enthält.

KI-generierte Inhalte können fehlerhaft sein.

# 4. Anforderungen

**Akteure:**

* **User**
  + **Typ**: Primärer, manueller Akteur
  + **Beschreibung**:  
    Der Nutzer ist die zentrale Person, die die App verwendet. Er interagiert direkt mit der Benutzeroberfläche, gibt seinen API-Schlüssel und eine Bildbeschreibung (Prompt) ein, startet die Generierung und verwaltet die erzeugten Hintergrundbilder.  
    Er nutzt Funktionen wie Speichern, Favorisieren oder „Als Hintergrund setzen“.
* **OpenAI**
  + **Typ:** Sekundärer, externer System-Akteur
  + **Beschreibung:**  
    Die OpenAI API ist ein externer Dienst, der die Generierung der Bilder übernimmt. Die App sendet eine Anfrage mit dem Prompt an die API, die daraufhin ein Bild erzeugt und eine URL zum Ergebnis zurückliefert. Die API ist entscheidend für die Kernfunktionalität der App.
* **Android System**
  + **Typ:** Sekundärer System-Akteur
  + **Beschreibung:**  
    Das Android-System stellt Systemfunktionen zur Verfügung, z. B. zum Speichern von Bildern, Setzen eines Bildes als Hintergrund oder für den Zugriff auf interne Verzeichnisse. Diese Aktionen werden von der App ausgelöst, aber vom System ausgeführt.

**Funktionale Anforderungen**

* Ein Bild, das Text, Diagramm, Reihe, Screenshot enthält.

  KI-generierte Inhalte können fehlerhaft sein.Der Nutzer kann einen gültigen API-Key eingeben.
* Der Nutzer kann einen Text-Prompt zur Bildbeschreibung eingeben.
* Die App sendet den Prompt an die OpenAI API, um ein Bild zu generieren.
* Das generierte Bild wird in der App angezeigt.
* Der Nutzer kann das Bild lokal auf dem Gerät speichern.
* Der Nutzer kann das Bild als Hintergrundbild setzen.
* Der Nutzer kann generierte Bilder als Favoriten markieren.
* Der Nutzer kann eine Sammlung aller gespeicherten Bilder ansehen.

**Nicht-funktionale Anforderungen (FURPS)**

* **F - Functionality**
  + Die App muss mit einem gültigen API-Key ein funktionierendes Bild liefern.
* **U - Usability**
  + Die Benutzeroberfläche ist einfach, verständlich und auch für Erstnutzer intuitiv bedienbar.
* **R - Reliability**
  + Die App muss auf Verbindungsfehler oder ungültige API-Keys sinnvoll reagieren und Fehlermeldungen anzeigen.
* **P - Performance**
  + Die Generierung eines Bildes darf nicht länger als 15 Sekunden dauern (je nach Netzwerkverbindung).
* **S - Supportability**
  + Die App muss auf Geräten mit Android 10 oder neuer zuverlässig laufen.
  + Die Codebasis ist modular und gut wartbar (Schichtentrennung, nachvollziehbare Paketstruktur).

# 5. Testkonzept

**A. Testumfeld**

Die Tests werden vollständig auf dem Android **Emulator** innerhalb von **Android Studio** durchgeführt.

Verwendet wird ein Emulator mit:

Ein Bild, das Text, Screenshot, Handy, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.**Image**: Google Play, API-Level 36

**Gerätetyp**: Pixel 9 Pro

**Auflösung**: 1280 × 2856

**Internetverbindung**: Emulator über PC-WLAN mit stabiler Verbindung zur OpenAI-API

**Berechtigungen**: Alle systemrelevanten Berechtigungen (Speicherzugriff, Internet) sind in der Emulator-Konfiguration aktiviert.

Die App wird im Debug-Modus über Android Studio installiert und manuell gestartet.

**B. Testmethode**

Es wird manuell getestet (**Black-Box-Testing**), wobei die Anwendung ausschliesslich über ihre Benutzeroberfläche getestet wird – ohne Code-Einblick.

Alle Use-Cases aus dem Use-Case-Diagramm werden in Form von Testfällen mit festgelegten Abläufen geprüft.

Das Verhalten der App wird visuell beobachtet und mit dem erwarteten Ergebnis verglichen.

Fehler werden dokumentiert, relevante Ausgaben im Logcat werden zur Analyse herangezogen.

**C. Testfälle**

**Testfall – Bildgenerierung und Speicherung**

**Identifikation**:

Der Nutzer sollte ein eigenes Wallpaper erstellen können, indem er seine Inspiration eingibt. Danach wartet er, bis das Bild generiert wurde. Zum Schluss soll er das Bild in seiner Sammlung sehen können.

**Vorbedingungen:**

* Ein gültiger OpenAI-API-Key wurde eingegeben und akzeptiert.
* Der Nutzer befindet sich auf der Prompt-Eingabe-Seite.

**Schritt-für-Schritt-Vorgehen:**

1. Nutzer gibt im Eingabefeld den Prompt „Sonnenuntergang über Bergen“ ein.
2. Nutzer klickt auf den Button „Generieren“.
3. Die App zeigt eine Ladeanzeige an.
4. Das generierte Bild wird angezeigt.
5. Der Nutzer klickt auf „Speichern“.
6. Die App speichert das Bild und zeigt eine Bestätigung.

**Erwartetes Ergebnis:**

* Das Bild wird erfolgreich generiert und korrekt angezeigt.
* Das Bild wird in der Bildersammlung angezeigt

.

**Testfall – Gespeicherte Wallpaper in der Sammlung anzeigen**

**Identifikation:**

Der Nutzer soll seine zuvor gespeicherten Bilder in der Sammlung ansehen können.

**Vorbedingungen:**

* Mindestens ein Bild wurde erfolgreich generiert und gespeichert.
* Die App ist geöffnet.

**Schritt-für-Schritt-Vorgehen:**

1. Nutzer navigiert über das Menü oder die Navigationsleiste zur Seite Sammlungen.
2. Die App zeigt eine Grid-Ansicht der gespeicherten Bilder.
3. Der Nutzer überprüft, ob das erwartete Bild korrekt dargestellt wird.

**Erwartetes Ergebnis:**

* Die Sammlung zeigt alle zuvor gespeicherten Bilder.
* Das zuletzt generierte Bild ist sichtbar und korrekt dargestellt.

**Testfall – Ein Bild aus der Sammlung als Hintergrund festlegen**

**Identifikation:**  
Der Nutzer soll ein Bild aus der Sammlung auswählen und es als Hintergrundbild auf dem Gerät festlegen können.

**Vorbedingungen:**

* Mindestens ein Bild ist in der Sammlung gespeichert.
* Der Nutzer befindet sich auf der Collection-Seite.

**Schritt-für-Schritt-Vorgehen:**

1. Nutzer tippt auf ein Bild in der Sammlung.
2. Die App öffnet eine Detailansicht des Bildes.
3. Nutzer klickt auf den Button „Als Hintergrund festlegen“.
4. Android zeigt ggf. eine Bestätigungsnachricht oder führt die Aktion direkt aus.

**Erwartetes Ergebnis:**

* Das Bild wird erfolgreich als Hintergrundbild des Homescreens gesetzt.
* Die App gibt eine visuelle oder textliche Rückmeldung, dass der Vorgang erfolgreich war.

# 6. Aufbau des Systems

# 7. Reflexion