GeoReminder

## App-Name

### Version 1.0.0, Datum: 3. Juli 2025

Hier kannst du einen Printscreen deiner App einfügen.

Abstract (Kurzbeschreibung)

An dieser Stelle kommt eine kurze Einführung, die in 5 – 10 Sätzen das Projekt beschreibt.

GeoReminder ist eine innovative App, die standortbasierte Erinnerungen ermöglicht, um das Problem vergessener Aufgaben an bestimmten Orten zu lösen. Nutzer können auf einer Karte einen Ort auswählen, einen Umkreis festlegen (z. B. 500 Meter) und eine Erinnerung mit Titel erstellen, die beim Betreten des Bereichs ausgelöst wird. Im Gegensatz zu herkömmlichen zeitbasierten Reminder-Apps, die oft zum falschen Zeitpunkt benachrichtigen, weil der genaue Aufenthaltszeitpunkt unvorhersehbar ist, bindet GeoReminder Erinnerungen direkt an den Ort. Dadurch wird sichergestellt, dass Nutzer genau in dem Moment erinnert werden, in dem sie handeln können. Die App adressiert das häufige Problem, dass ortsabhängige Aufgaben vergessen werden, weil die Erinnerung nicht mit dem tatsächlichen Aufenthaltsort verknüpft ist. GeoReminder bietet somit eine praktische Lösung für eine präzise, kontextbezogene Aufgabenverwaltung.

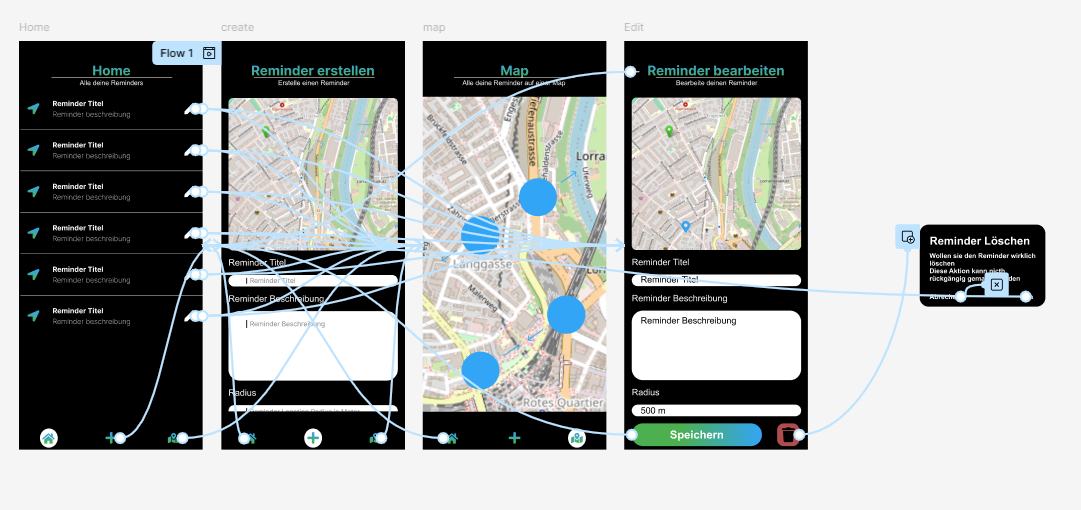
user stories

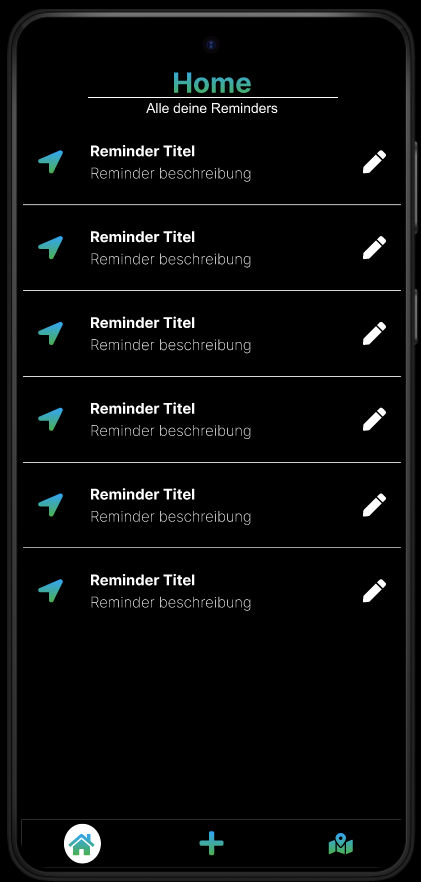
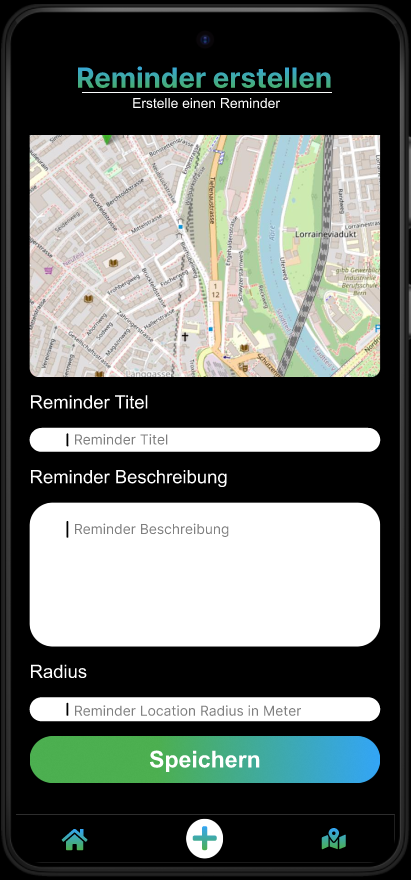
* **Als Nutzer** möchte ich eine neue Erinnerung erstellen,  
  **damit** ich Dinge genau dann erledige, wenn ich am richtigen Ort dafür bin.
* **Als Nutzer** möchte ich eine Erinnerung löschen können,  
  **damit** ich veraltete oder nicht mehr benötigte Erinnerungen entfernen kann.
* **Als Nutzer** möchte ich alle meine Erinnerungen in einer Liste sehen,  
  **damit** ich den Überblick behalte, welche Erinnerungen ich bereits erstellt habe.
* **Als Nutzer** möchte ich eine Erinnerung antippen können,  
  **damit** ich die genaue Position der Erinnerung auf einer Karte sehen kann.
* **Als Nutzer** möchte ich auf eine Edit-Icon klicken können,  
  **damit** ich direkt zur Bearbeitungsansicht gelange und sie bei Bedarf ändern oder löschen kann.
* **Als Nutzer** möchte ich eine Karte sehen, auf der alle Erinnerungen angezeigt werden,  
  **damit** ich visuell erkennen kann, wo überall Erinnerungen gesetzt wurden.
* **Als Nutzer** möchte ich, dass jede Erinnerung auf einer Karte angezeigt wird,  
  **damit** ich schnell sehe, welche Erinnerungen an welchen Orten gespeichert sind.
* **Als Nutzer** möchte ich, dass meine Erinnerungen lokal gespeichert werden,  
  **damit** sie auch nach dem Schliessen der App noch vorhanden sind.

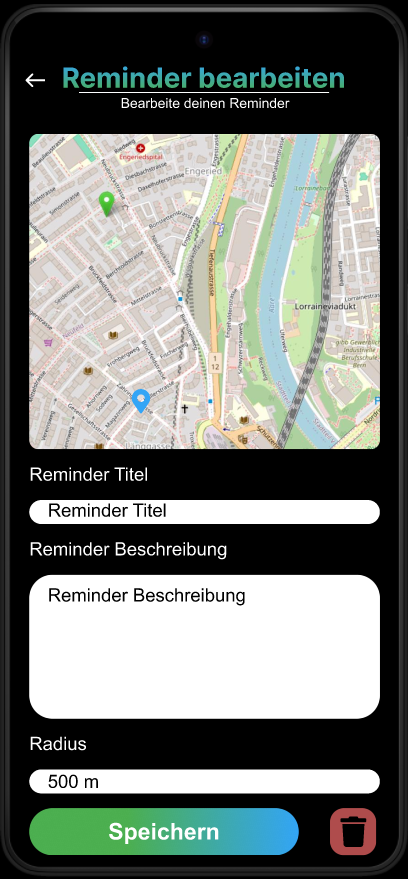
Mockups

An dieser Stelle den Prototypen deiner App beschreiben. Füge einen Printscreen deiner Screens ein und beschreibe jeden Screen einzeln.

Figmalink: [Figmalink](https://www.figma.com/proto/nTbbwvF9UuIDE67u59Pn5t/Geo?page-id=0%3A1&node-id=1-3&viewport=-1021%2C-395%2C1.22&t=yRuimVEkb9Oz5lol-1&scaling=scale-down&content-scaling=fixed&starting-point-node-id=1%3A6)





Technische Realisierung

An dieser Stelle beschreiben, wie die technische Realisierung der App erfolgt ist. Welche Komponenten gibt es und wie werden die Daten gespeichert resp. von wo werden diese geladen. Wenn es sich um komplizierte Abläufe handelt, werden diese am besten mit UML Diagrammen beschrieben.

Eine Fachperson, die das Kapitel liest, sollte schnell nachvollziehen können, wie die App realisiert wurde.

## Komponente:

BottomNavIcon:

Hier sind die BottomNav Icons, welche dann in der Navbar unten zu sehen sind.

Wir importieren Ionicons, MaskedView und Lineargradient.

Ich habe hier noch einen Parameter dazugetan damit man checken kann, ob das Navbar Icon gerade focused ist (also ob das es das Icon der Seite ist, auf der man sich gerade befindet.). Zuerst wird abgefragt, ob das Icon gerade focused ist und wenn dies der Fall ist, wird einfach simple ein Ionicon zurückgegeben wo die Dinge, die immer gleichbleiben, hardcoded sind, sprich Grösse und style und wo der Name von den Parametern ausgelesen wird.

Falls das Icon nicht focused ist, wird eine anderer Return code zurückgegeben. Es wird ein MaskedView zurückgegeben damit ein die Icons ein Gradient als Farbe haben können, was mit Ionicons eben sonst nicht geht. Durch MaskedView wird einfach die Form von den Icons «ausgeschnitten» und ein linear Gradient wird dahinter gelegt. Um sicherzustellen das die richtige Icon form ausgeschnitten also maskiert wird, wird bei maskElement ein View mit dem fast gleichen Ionicon zurückgegeben wie im ersten Return (nur color=’black’) ist anders.

Dann wird innerhalb vom MaskedView eine Lineargradient Komponente benutzt und dort wird dann der Gradient, den wir wollen gemacht, wodurch die Icons in der Navbar mit einem Farbverlauf zu sehen sind

Testing

In diesem Kapitel definiert ihr die manuellen Tests, die durchgeführt werden. Es müssen mind. 4 manuelle Tests definiert werden.

Hier ein Beispiel eines Tests:

|  |  |
| --- | --- |
| **Abschnitt** | **Inhalt** |
| ID | Testfallnummer (ST = Systemtest) |
| Anforderungen | Welche Anforderungen werden durch diesen Testfall abgedeckt. (User Stories) |
| Vorbedingungen | Was muss gegeben sein, damit dieser Test durchgeführt werden kann? |
| Ablauf | Welche Schritte werden bei der Durchführung des Tests durchlaufen? |
| Erwartetes Resultat | Was sollte nun passiert sein? |

Testprotokoll

In diesem Kapitel wird die Testausführung protokolliert. Welche Tests wurden wann von wem mit welchem Resultat durchgeführt.

Hier ein Beispiel eines Eintrages:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Person / Datum | Erfolgreich | Bemerkungen |
| ST-01 | D. Muster 18.06.2023 | Ja | Der Testfall war erfolgreich, der Testperson 1 ist jedoch aufgefallen, dass es in der angezeigten Fehlermeldung noch einen Rechtschreibfehler gibt. |
| … |  | … | … |
|  |  |  |  |

fazit

Hier kommt eure Reflexion zum Projekt.

* Was lief gut/schlecht?
* Wie seid ihr mit dem Endergebnis zufrieden?
* Was habt ihr gelernt?
* War alles vorhanden oder was fehlte noch?
* Usw.