# [Teen Diabetes App Name] - A Sample Capstone Project

A teen with type 1 diabetes (T1D) has to deal with not only the changes brought on by adolescence, but also the complexities of managing a chronic disease. Although diabetes can be managed, it must be consistently controlled. The management process requires an effective education, motivation, self-discipline, and emotional, social, financial, family, and health care support system.

Survey findings suggest that the use of mobile tools for diabetes self-management is appealing to teens with T1D, but that they have not encountered apps that are intuitive and fit their needs. Communication with parents, clinicians, and other supportive individuals is critical for many teens with T1D, helping them troubleshoot problems and follow their medical regimen. As such, many families report using text messaging to communicate about teens’ blood glucose values. Yet, teens’ self-management could benefit from more structured mobile communication that involves sharing more nuanced behavior patterns and health outcomes.

[Teen Diabetes App Name] helps teens manage and track their management of their diabetes and share their progress with friends, family members, and healthcare providers. The app is intended to help relieve stress for teens with diabetes and their parents as they navigate through a critical and formative period in their lives.

## Basic Project Requirements

Any potential Capstone project must support multiple users and should leverage services running remotely in the cloud. Each project's specification clearly outlines the app's intended high-level behavior, yet leaves substantial room for individual creativity. Students will therefore need to flesh out many important design and implementation details. Basic requirements for all Capstone MOOC project specifications include:

1. Apps must support multiple users via individual user accounts. At least one user facing operation must be available only to authenticated users.
2. App implementations must comprise at least one instance of at least two of the following four fundamental Android components: Activity, BroadcastReceiver, Service, and ContentProvider.
3. Apps must interact with at least one remotely-hosted Java Spring-based service over the network via HTTP.
4. At runtime apps must allow users to navigate between at least three different user interface screens. For example, a hypothetical email reader app might have multiple screens, such as (1) a ListView showing all emails, (2) a detail View showing a single email, (3) a compose view for creating new emails, and (4) a Settings view for providing information about the user's email account.
5. Apps must use at least one advanced capability or API from the following list covered in the MoCCA Specialization: multimedia capture, multimedia playback, touch gestures, sensors, or animation. Experienced students are welcome to use other advanced capabilities not covered in the specialization, such as BlueTooth or Wifi-Direct networking, push notifications, or search. Moreover, projects that are specified by commercial organizations may require the use of additional organization-specific APIs or features not covered in the MoCCA Specialization. In these cases, relevant instructional material will be provided by the specifying organization.
6. Apps must support at least one operation that is performed off the UI Thread in one or more background Threads or a Thread pool. Communication between background Threads and the UI Thread should be handled by one of Android concurrency frameworks, such as the HaMeR or AsyncTask framework.

There may also be additional project-specific requirements (e.g., required use of a particular project-specific API or service).

## Basic Functional Description and App Requirements for [Teen Diabetes App Name]

1. The *Teen* is the primary user of the mobile app.  A *Teen* is represented in the app by a unit of data that contains the core set of identifying information about a diabetic adolescent including (but not necessarily limited to) a first name, a last name, a date of birth, and a medical record number.
2. The *Teen* will receive a *Reminder* in the form of alarms or notifications at patient-adjustable times, at least three (?) times per day.
3. Once the *Teen* acknowledges a *Reminder*, the app will open for a *Check-In*.  A *Check-In* is a unit of data associated with that *Teen*, a date, a time, and the user's responses to the following set of *Questions* at that date and time:
4. What was your blood sugar level at [meal time/bedtime]?
5. What time did you check your blood sugar level at [meal time]?
6. What did you eat at [meal time]?
7. Did you administer insulin?
8. Who were you with when you checked/should have checked your blood sugar?
9. Where were you when you checked/should have checked your blood sugar?
10. How was your mood when you checked/should have checked your blood sugar?
11. How was your stress level when you checked/should have checked your blood sugar?
12. How was your energy level when you checked/should have checked your blood sugar?
13. Were any of these things happening at the time you checked/should have checked your blood sugar: Rushing; feeling tired of diabetes; feeling sick; on the road; really hungry; wanting privacy; busy and didn’t want to stop; without supplies; feeling low; feeling high; having a lot of fun; tired.
14. A *Teen* is able to monitor their *Check-Ins* with data that is updated at some appropriate interval (perhaps when a *Check-In* is completed). The data can be views graphically on the mobile device.
15. A *Follower* is a different type of user (e.g., a parent, clinician, friend, etc.) who does not the ability to perform *Check-Ins*, but who can receive *Check-In* data shared from one or more *Teens*. A *Teen* can be a *Follower* for other *Teens*.
16. The *Teen* can choose what part(s) of their data to share with one or more *Followers*.
17. *FollowerTeen* data should only be disseminated to authorized/authenticated *Followers* and accessed over HTTPS to enhance privacy and security of the data.

## Implementation Considerations

* What will the user interface look like for a *Teen* so that it is quick and simple to use? How will there be at least three different user interface screens?
* How will *Reminders* be delivered to a *Teen* in a way that will help the *Teen* to use the app more frequently and consistently?
* How will the *Questions* asked to a *Teen* be stored, e.g., will they be hard-coded (and thus not extensible without changing the program) or database-driven (and thus extensible without changing the program)?
* How will the app handle missing data, e.g., *Questions* that aren’t answered by a user.
* How will a *Teen* choose which *Follower(s)* to share their *Check-In* data? Likewise, how will a *Teen* choose which elements of their *Check-In* data to share? Will the app’s sharing capabilities be integrated with social media sites, such as Facebook, Instagram, Vine, Twitter, etc.?
* How will *Check-In* data be transferred from a *Teen*'s device to other *Followers*?
* How will another *Teen* or *Follower* be notified that a *Teen* has shared a *Check-In*? What will this look like?
* How will *Teen* data be securely transferred to/from the server and shared only with the right *Followers*?
* What will a shared *Check-In* look like?
* How, when, and how often will the *Teen* or *Follower* user enter their account information? For example, will the user enter this information each time they run the app? Will they specify the information as part of a preference screen?
* What user preferences can the user set? How will the app be informed of changes to these user preferences?
* How will the app handle concurrency issues, such as how will periodic updates occur - via server push or app pull? Will the data be pulled from the server in multiple requests or all at one time?
* How will the app use at least one advanced capability or API listed above? For example, will you create an animation to explain the app? Will you allow *Teens* to take pictures of meter readings and share them to their *Followers*? Will you use push notifications from *Followers* to prompt Teens to share their data?
* Does your app really require two or more fundamental Android components? If so, which ones? For example, this app might benefit from using a ContentProvider (e.g., to store *Questions*) or from using a background Service that synchronizes local and remote data only when the device is connected to a WiFi network.