GitHub Username: tomca32

MindLeaps Sponsorships

Description

MindLeaps is a non-profit organization working with homeless, and refugee, children, by providing them with scholarships for skills that are in demand in their country's labor force. MindLeaps Sponsorships enables supporters of MindLeaps to see which students need help the most, learn about their lives, and donate towards their scholarships.

Intended User

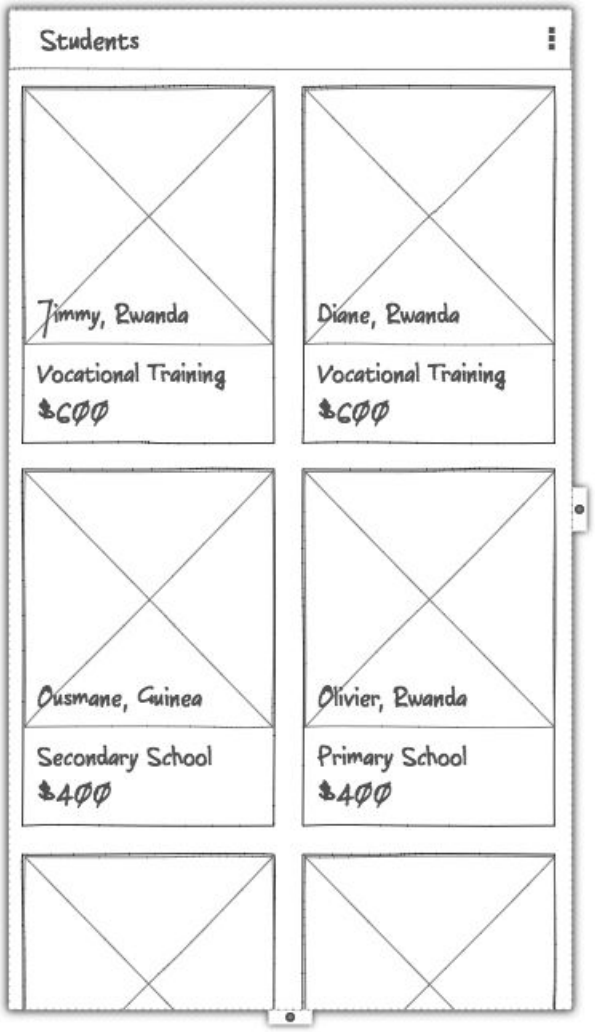
Intended users of MindLeaps Sponsorships are MindLeaps friends and donors.

Features

- Browse MindLeaps Students
- Donate to individual student's scholarships
- Filter students by Country, Type of sponsorship, and sponsorship status

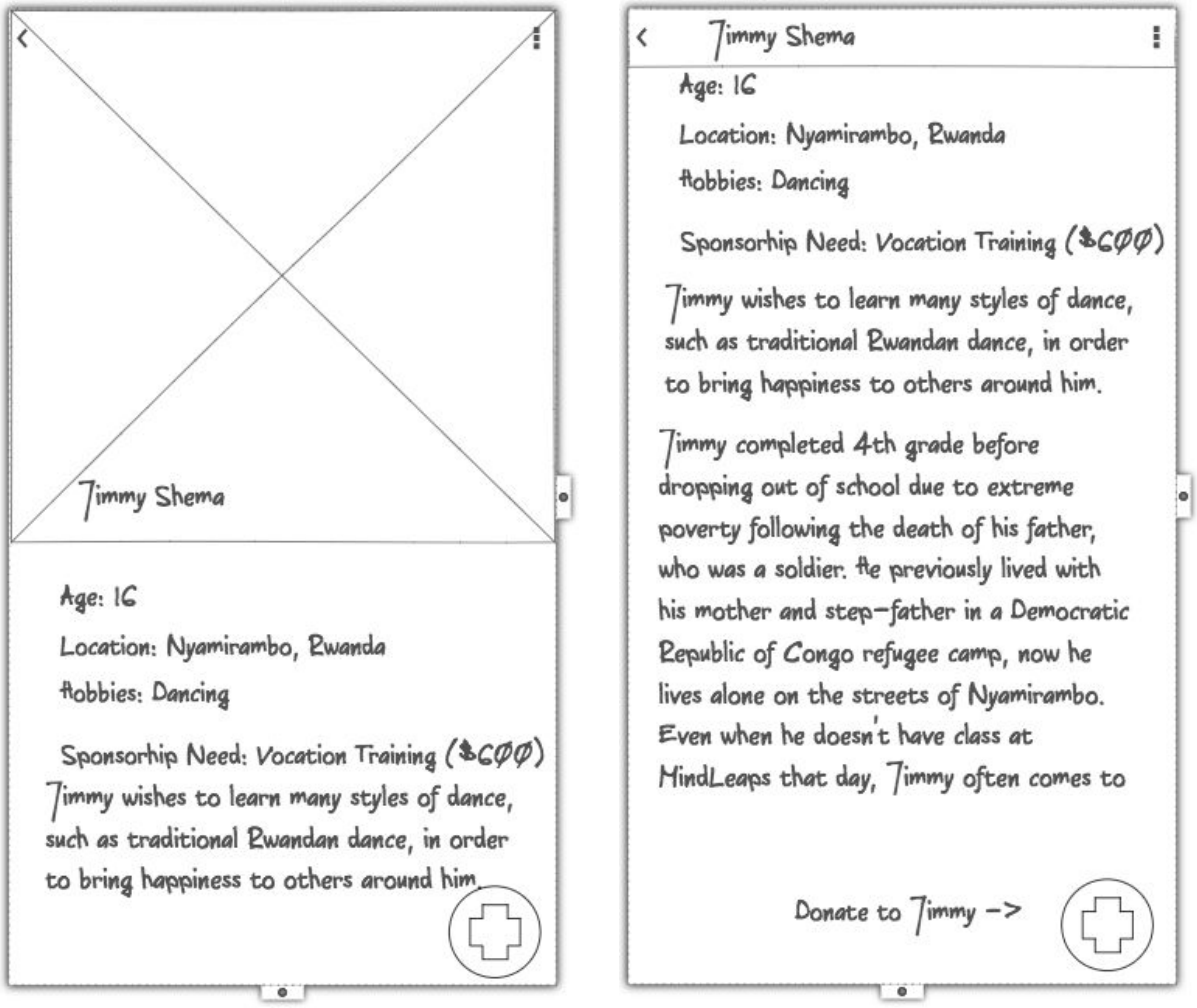
User Interface Mocks

Student List Screen



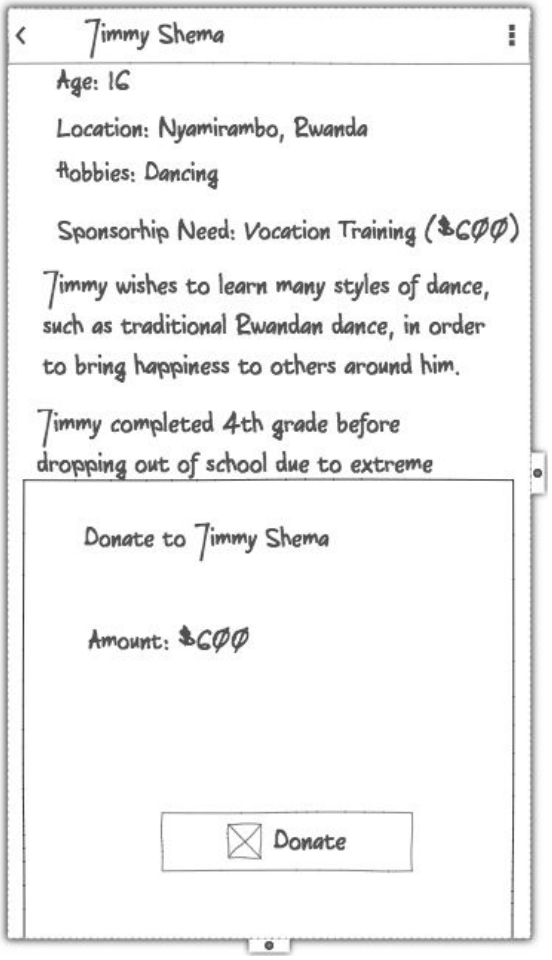
Student List Screen displays a list of student cards. Each card consists of a student image, name, country, scholarship type and amount. Tablet interface is not significantly different, it just has more grid columns. Click on an individual student's card brings the user to the student detail screen.

Student Detail Screen - Phone



Student detail screen displays information about a particular student. The top portion of the screen features a full bleed student image, followed up by student's details and a short personal story.

Floating Action Button (icon not final) is used for a donate action with a "Donate to Student" text inspired by Google's One Today app.

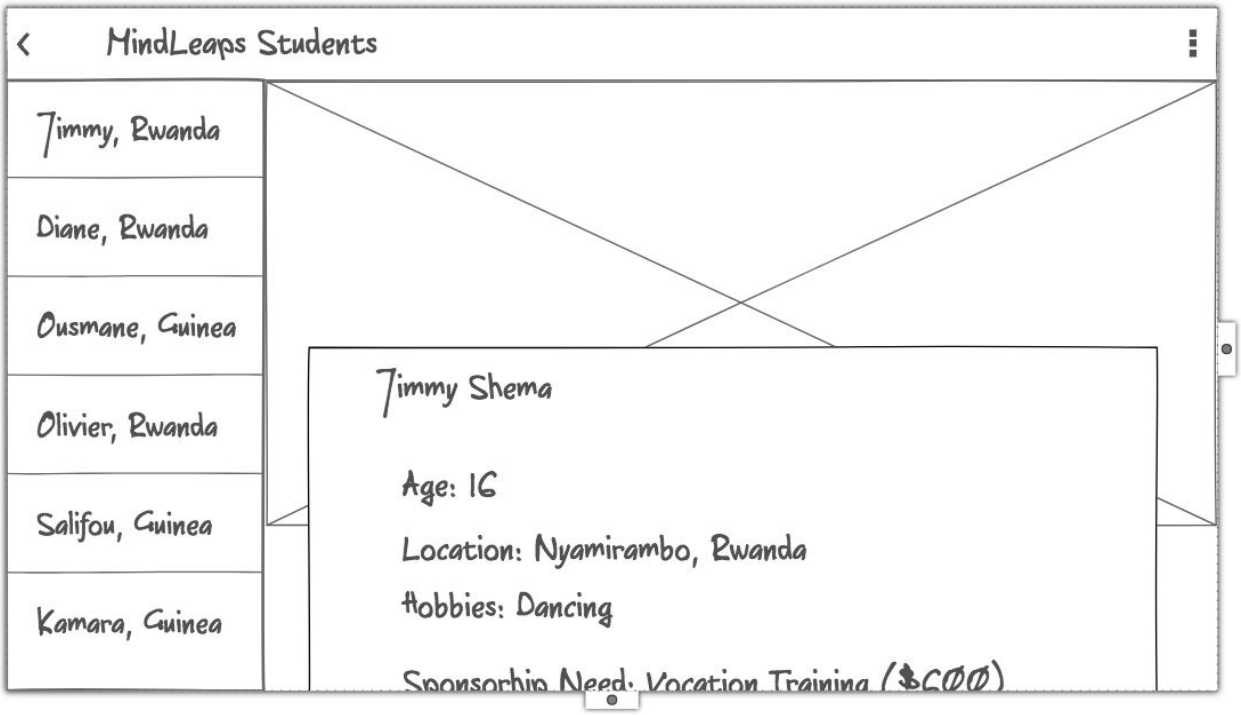


Tapping the Donate FAB brings up a separate surface with donation information and a Donate button.

Donations will be handled through Stripe and Pay with Google.

If there are no linked Android Pay accounts, this surface may contain a Credit Card form. This is not final as technical implementation concerns are not fully explored.

Student Detail Screen - Tablet



Tablet detail screen is laid out as a master list/detail layout. The detail part has identical functionality to the phone screen with small changes in layout; content is displayed on a separate surface to better indicate the scrolling intent to the user.

Key Considerations

How will your app handle data persistence?

Student information and details are handled with Firebase Realtime Database, while the images are handled with Firebase Storage.

Describe any edge or corner cases in the UX.

Collecting payment methods is handled through Android Pay; however, if a user doesn't have any payment method saved with Google, it is possible to collect payment method details through Stripe's Card Input Widget (https://stripe.com/docs/mobile/android#card-input-widget). This might complicate development significantly, so it is a "nice to have" feature and not a priority.

Describe any libraries you'll be using and share your reasoning for including them.

- Android Annotations for project organization and reducing boilerplate code
- Stripe for payment processing
- Google Payment API for accessing Android Pay payment information
- Picasso or Glide for image downloading and caching
- Firebase SDK (database and storage) for data and image store retrieval
- Support and design support libraries for backwards compatibility

Describe how you will implement Google Play Services or other external services.

- Pay with Google (play-services-wallet) is used to retrieve user's payment information that can be sent to Stripe for donation processing. When the user taps the donate FAB, app checks if the user has any payment information saved with Google via the `IsReadyToPayRequest`. If the user has payment information, the app will send it to Stripe. If not, the app will either display a credit card form or inform the user that there is no saved payment method. The implementation should follow Stripe's documentation (https://stripe.com/docs/mobile/android/pay-with-google).

Next Steps: Required Tasks

Tasks listed here are defined from the user's point of view. No work is done unless it's absolutely necessary for the current task. For example, wallet API will not be included until the work is started on payment processing.

After each completed task, app will be fully functioning, just with a reduced feature set, until all the tasks are completed.

Task 1: Project Setup

This task involves creating a new project and creating a main activity.

- Create new project in Android Studio
- Include the basic libraries: Annotations and Support
- Create an empty Main Activity
- Ensure the application compiles and runs

Task 2: Student List screen

This is probably the biggest task in the project. It involves building a list of students that are fetched from the Firebase Database and displaying their images from the Firebase Storage.

- Configure Firebase to fetch students from the remote database
- Build list layout according to the UI mock
- Fetch student images from Firebase Storage - Picasso reference: https://medium.com/@premith/how-to-use-square-s-picasso-with-firebase-storage-809bca6f81ba
- Ensure the application compiles, runs, and displays a list of students

Task 3: Student Detail screen (phone)

This task involved building the student details screen for the phone. This involves building a phone layout of the screen, fetching and displaying all of the available student details and displaying a full bleed student image. Responsiveness, and optimizing for different screen sizes is not the goal of this task. However, the application should still work on tablets. The donate FAB is not present at this point.

- Building the student detail layout according to the UI mock using coordinator layout
- Fetching and displaying individual student details
- Fetching and displaying a full bleed image

Task 4: Student Detail screen (tablet)

This task optimizes the existing student detail screen for bigger displays. No additional functionality is involved.

- Optimize the student detail screen for larger displays according to the provided UI mock for tablet

Task 5: Donate Functionality

This task starts involves integrating Stripe and Google Payments / Wallet APIs into the application. It also involved changing the student detail screen to add the FAB for donations.

- Integrate with Google Payments API
- Use TEST_ENVIRONMENT to test payments - this should be done via Gradle build flavors to ensure that environment in development while swapping it out for release
- Integrate with Stripe, following the same pattern for testing as with Google Payments
- Add FAB to existing layout and building a UI showing the donated amount and a “Donate” button

Task 6: Polish

This task focuses on taking another look at the whole app and find ways to make it more usable. This is hard to predict as many UX issues are discovered only after the app has been built and used for a while.

- Perform some real user testing by asking people unfamiliar with the application to try using it
- Collect feedback from real use testing and decide what makes sense implementing
- Implement UX improvements from real user feedback
- Think about possible enhancements such as filtering students by country or age; sorting by donate amount required
- Test and implement a payment method outside of Google Payments by collecting payment information directly in the application and sending them directly to Stripe
- Perform a security review to ensure no payment information data is ever saved on the device - the application should sends the data to payment processor and never store it locally