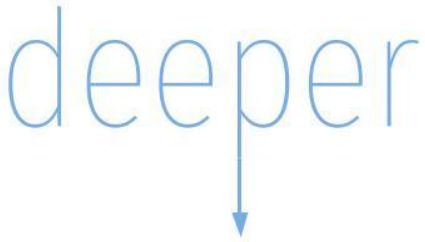
**Boston University**
**Electrical & Computer Engineering**
**EC463 Capstone Senior Design Project**

# First Prototype Testing Report

deeper




by

Team #6
deeper

Team Members

Nicole Kwon kwonn@bu.edu
Daniel Li dli0793@bu.edu
Harry Li harryli@bu.edu
Byron Mitchell byronmit@bu.edu

November 21, 2021

## Required Materials:

Software:

- Back End:
    - PyFlask backend
    - MongoDB Database
    - Firestore (Database)
    - Firebase Authentication
- Smartphone Applications on iPhone and Android device (using Expo)
    - UI Interfacing

## Pre-testing Setup Procedure:

**deeper App:**

1. Opened project in Visual Studio Code;
2. Run “expo start” in terminal under the folder of the deeper app;
3. Install Expo app on iPhone/Android device;
4. Ran app by scanning Expo barcode with iPhone and Android device.

**Login Functionality:**

1. Opened up Firebase database
2. Set up MongoDB on local machine
    a. Run “mongod” in the MongoDB folder;
    b. Run “mongo” in the MongoDB folder;
3. Ran PyFlask server on local machine
4. Open the Postman application and copy the PyFlask IP address into the Postman user interface.

## Setup:

There were two main tests done during our prototype presentation -- the deeper app demo with smooth transitions between the screens that reflect the Figma and the full login functionality.

The first test demonstrated the app and its basic functionalities. This included changing to different screens/tabs along with the functionality of buttons. The front end reflected our Figma design, which was also pulled up side-by-side with our cross-platform application to demonstrate how our visualization from our first sprint became actuality.

For the second test, we performed the full login functionality. This included going through the process of signing up and signing in. For signing up, we did not include all the required fields at first (full name, email address, and password) to demonstrate the error-checking behind our sign-up functionality. This was followed by signing in, where the user inputted the newly created email address and password for authentication, as well as displaying its existence in Firebase. There was also another login functionality shown with different backend code, locally using a PyFlask server and MongoDB.

**Measurements Taken:**

**deeper App:**

1. After opening Visual Studio, we ran Expo, and the apps were able to run successfully on both an iPhone and an Android device, demonstrating our cross-platform functionality.
2. The user was able to navigate through the different screens, from start to finish.
3. The user was able to navigate through different tabs by clicking on the buttons, represented by different icons on the bottom bar of the screen.
4. The overall UI reflected the Figma design, which was shown on the side. All of the different components from the Figma visualization were included in our frontend code.

**Login Functionality:**

1. Explain the metrics and results behind logging in and Firebase
    - To measure the functionality behind our authentication that uses Firebase, we look at the error checking in the sign-up/login page and that the correct information is stored/accessed in our database.
    a. User should be able to create an account;
        - To demonstrate the error checking in our sign-up page, we left some blank sections in our sign-up form, which then prompted the user to completely fill out the form in order to register. Also, we wrote down different passwords for the password and confirm password sections, which then prompted the user to check the password fields as they don't match in order to register. After correctly filling out the form, we could see that the user's information was safely and correctly stored in the database by accessing Firestore.
    b. User should be able to see if their login credentials are correct or not;
        - To demonstrate the error checking in our login page, we input an email address that isn't associated with any user in our database which prompts the user to use an existing account or to sign up with the provided email address. Also, by providing a correct email address, but the incorrect password prompts the user to enter the correct password associated with the account. After correctly login in, we see that the app correctly fetches the user's information as the app outputs it to the console where we can verify that it matches our database.
2. Explain about the metrics and results behind PyFlask and MongoDB
    a. User should be able to see if their login credentials are correct or not;
        - In order to measure the functionality, we saved a testing account into the database with a known password and email address. By using Postman imitating frontend passing a HTML form, we entered the email address and password. If the password didn't match with the database's record, it'll return an error message that could be seen in the Postman's terminal. If it worked, the backend will return a session cookie named "current_usr".
    b. User shouldn't need to log in again after logging in before;

- In order to test if the backend will recognize the logged in user, we will save the session cookie that returned from the backend and pass it to the same login API again. If the functionality doesn't work, the backend will require the user to use email and password to log in again. If it works, backend should return a string "Already logged in"

c. User should be able to stay logged in for 7 days;
   - Test this functionality by checking the expiration date of the session cookie. Every new login action should let the user stay logged in for 7 days. Therefore, the cookie should expire in 7 days. We checked the expiration date of the cookie through the Postman terminal.

d. User should be able to create an account;
   - By passing username, email address, and password to the API for sign up, user should be able to register a new account for the new email address. If the user entered an email address that already registered, or a username that already exists, the backend should prompt an error message saying the email/username already exists. Otherwise, the backend should return a string saying "Successfully signed up." and the database should have a new entry in its "user" collection.

**Conclusions:**

**deeper App**

I. Overall, we had a successful first prototype run of our deeper application. We were able to demonstrate that the app has cross-platform functionality: Android and IOS. The screens were able to navigate and the buttons worked as expected just like our Figma prototype. Now we need to work on adding more functionality into the different screens because it is only a front-end shell right now.

**Login Functionality:**

I. The login functionality worked as expected too. Users can sign up for a new account, and this account will be logged in our Firebase database. The app also detects faulty inputs and if the account already exists when signing up. Users can log in to an existing account or use Google sign-in. The login also detects a nonexisting account and/or an incorrect password. We have a working base login and sign-up functionality. Now we will need to add a reset password and personalized name displayed based on the current user. Additionally, we will add sessions so that users can stay logged in for a certain amount of time.

**Score Sheet**

**deeper App:**

| **Description** | **Did it work? (y/n)** |
|---|---|
| The Expo apps should be able to run successfully. | Y |

| | |
|---|---|
| The user should be able to navigate through the different screens. | Y |
| The user should be able to navigate through different tabs by clicking on the buttons. | Y |
| The UI should reflect the Figma design. | Y |

**Login Functionality:**

| **Description** | **Did it work? (y/n)** |
|---|---|
| The user can sign up for a new account. | Y |
| The user can log in to an existing account. | Y |
| The user will be denied if the password and email combo are wrong. | Y |
| The user cannot register a new account if the email address is already registered. | Y |