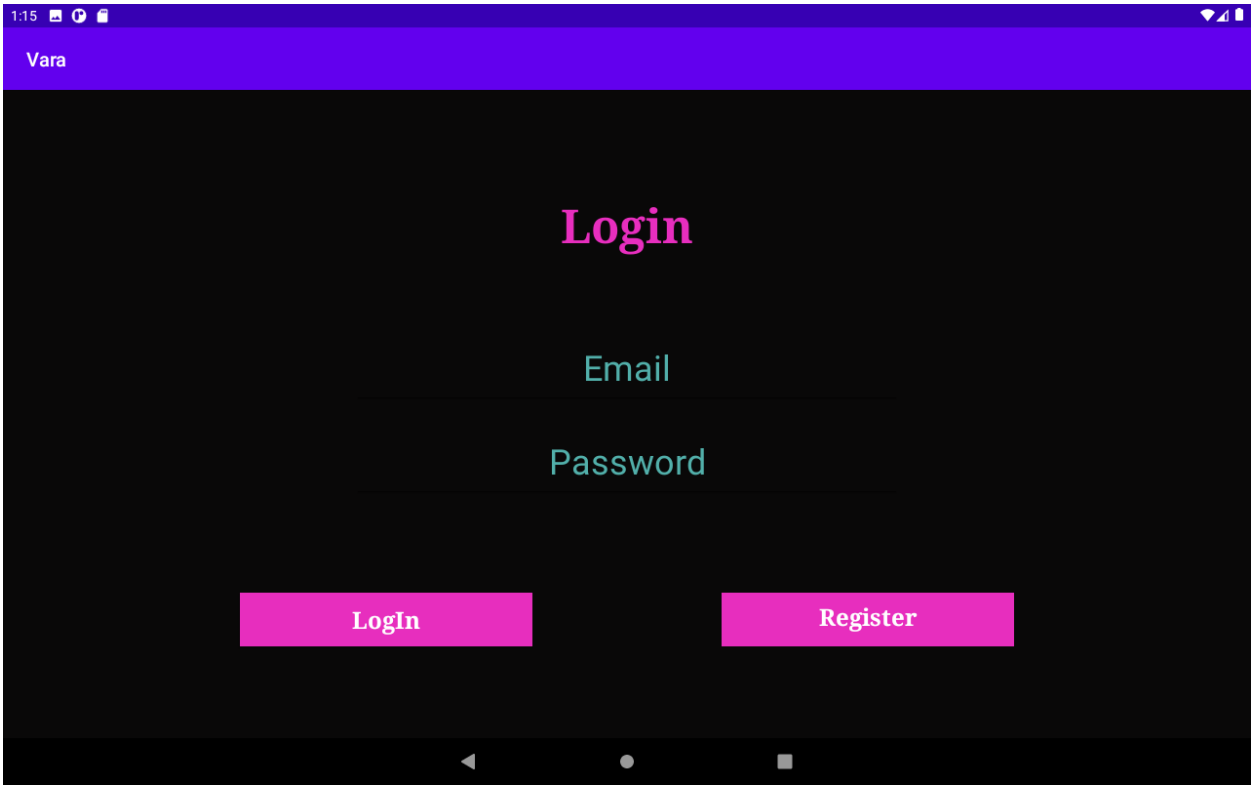
In vehicle daily routine for a VARA Platform Scenario: (Vehicle Advertising Revenue Assistance)

1) Driver logs into their Partner system (Uber / Lyft / etc) to prepare for their work shift.
2) Driver also logs into the Vara platform and initializes the VARA Platform (***Screen 1***)
3) Passenger screens wait at ***Screen 2*** until a ride stars or the Platform is signed out.
4) Upon each ride the Platform will execute the following loop:
    a. The Platform will initialize upon a wake-up transaction from the partner application.
    b. The Platform will run through the welcome page(s) ~ approx. 1 min. (***Screen 3***)
    c. The Platform will move to the adds page where the following will be displayed:
        i. Ads from Google's ad's service taking up 2/3 of the screen area
        ii. A GPS representation of the ride in progress
    d. The Platform will continue to cycle through appropriate ads (based on location and/or Google's algorithms) until the system receives a notice of ride destination / completion from the partner system. (***Screen 4***)
    e. The platform will display a farewell page & message that will allow the passenger to consent or not to the Platform sending them coupons or additional opportunities for the ads that played for them during the ride. (***Screen 5***)
    f. The platform will record the adds and rider information into the cloud database collections.
    g. The system will return to the waiting page (**Screen 2**)
5) The driver or the system will sign-out upon a sign-out of the partner application trigger event if available, otherwise it will be done manually by the driver (***Screen 1***)

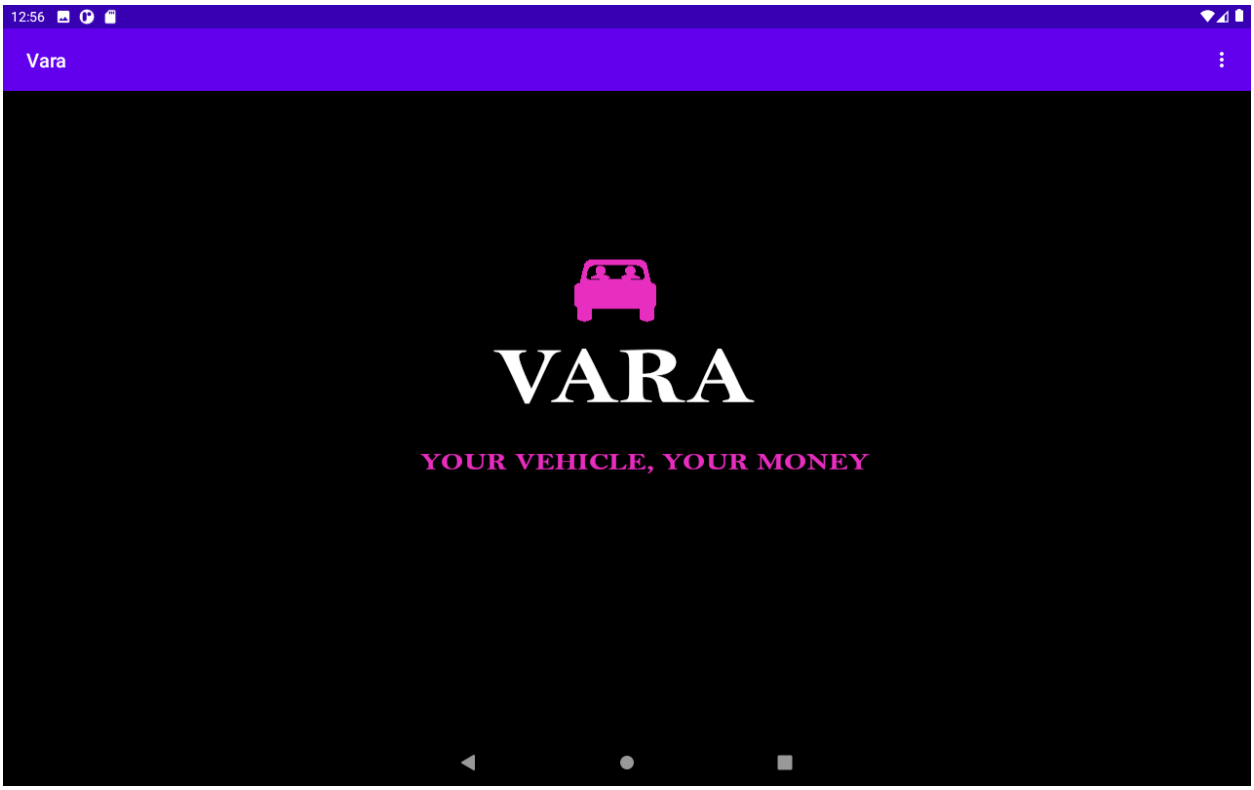
Vara Platform concept screens and flow:



Screen 1 - From this page, the user can do the following:

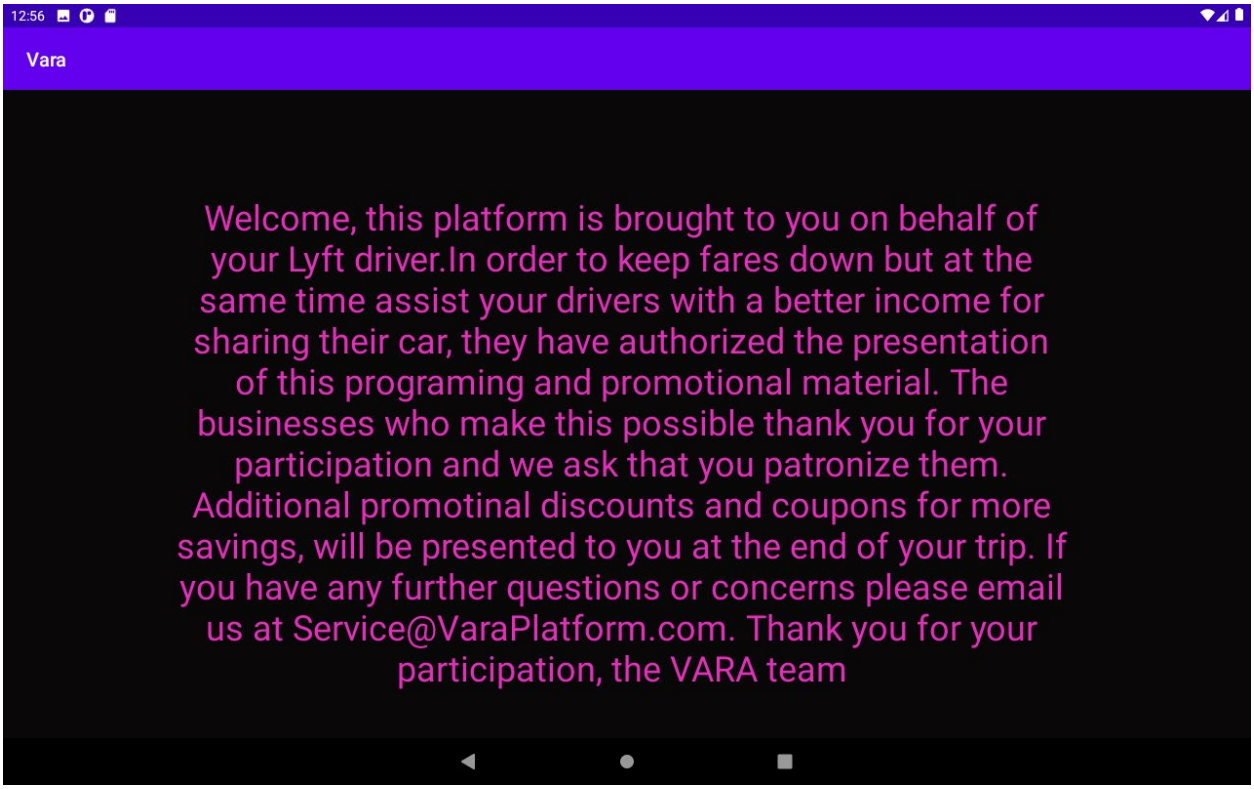
1) Login to an existing account on the VARA Platform system
    a. User Login Information is securely stored in the cloud
        i. Present back-end is firebase cloud user authentication

2) Sign-Up for a Vara Platform System account
    a. The user information entered is captured and stored in our cloud solution
        i. This is handled by our VARA Platform cloud firebase solution



Screen 2 - This page is the home page screen. From here the following could happen

1) Option to log-out of the platform
    a. Use the ellipses in the upper right corner for a drop-down menu.
    b. Options Include
        i. User admin - to update information in “UserInfo collection”
        ii. Log-Out
2) The default ‘Waiting Screen’
    a. Upon logging in to the system, the user is brought to this screen
    b. Between rides being provided to clients
        i. This screen is not capturing or doing anything
        ii. *TBD - to also capture time spent on this screen as a stat (Up/On time)
            1. Per driver / per day entry

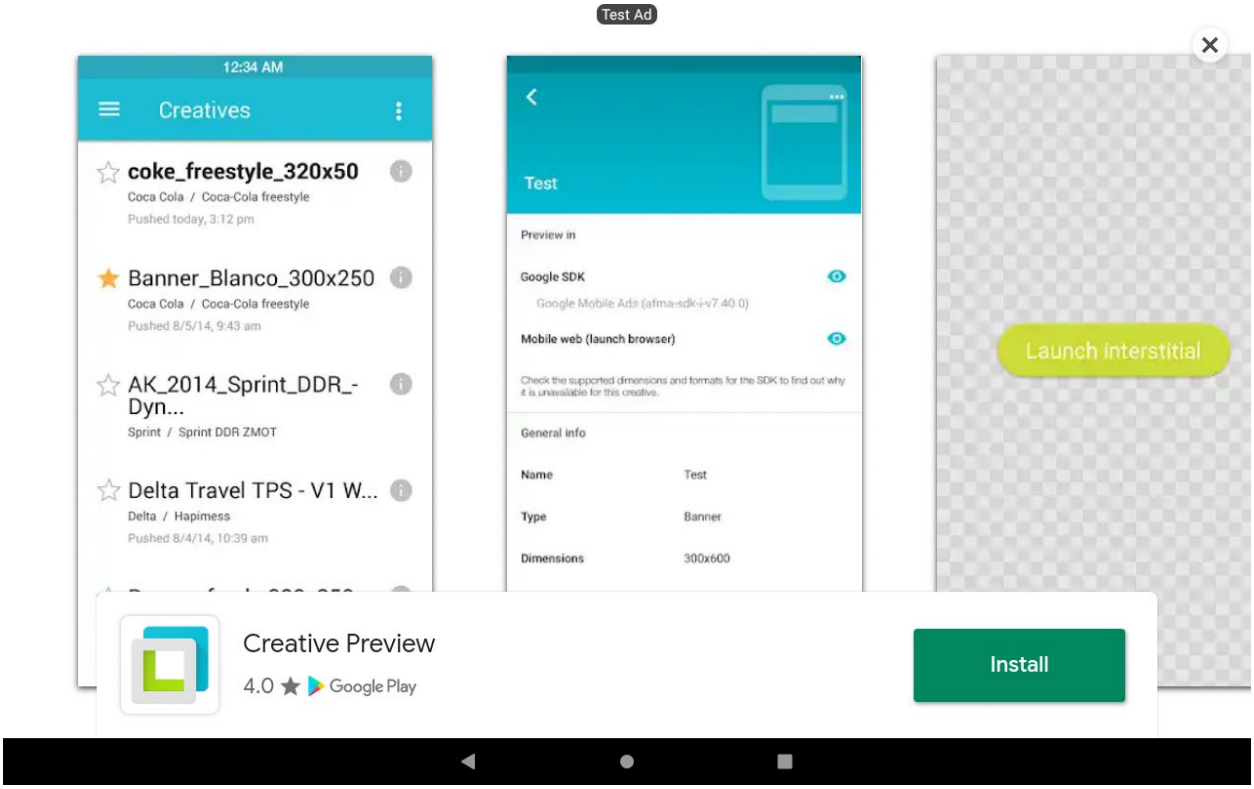


***the above is for mock up and not the final production version.**

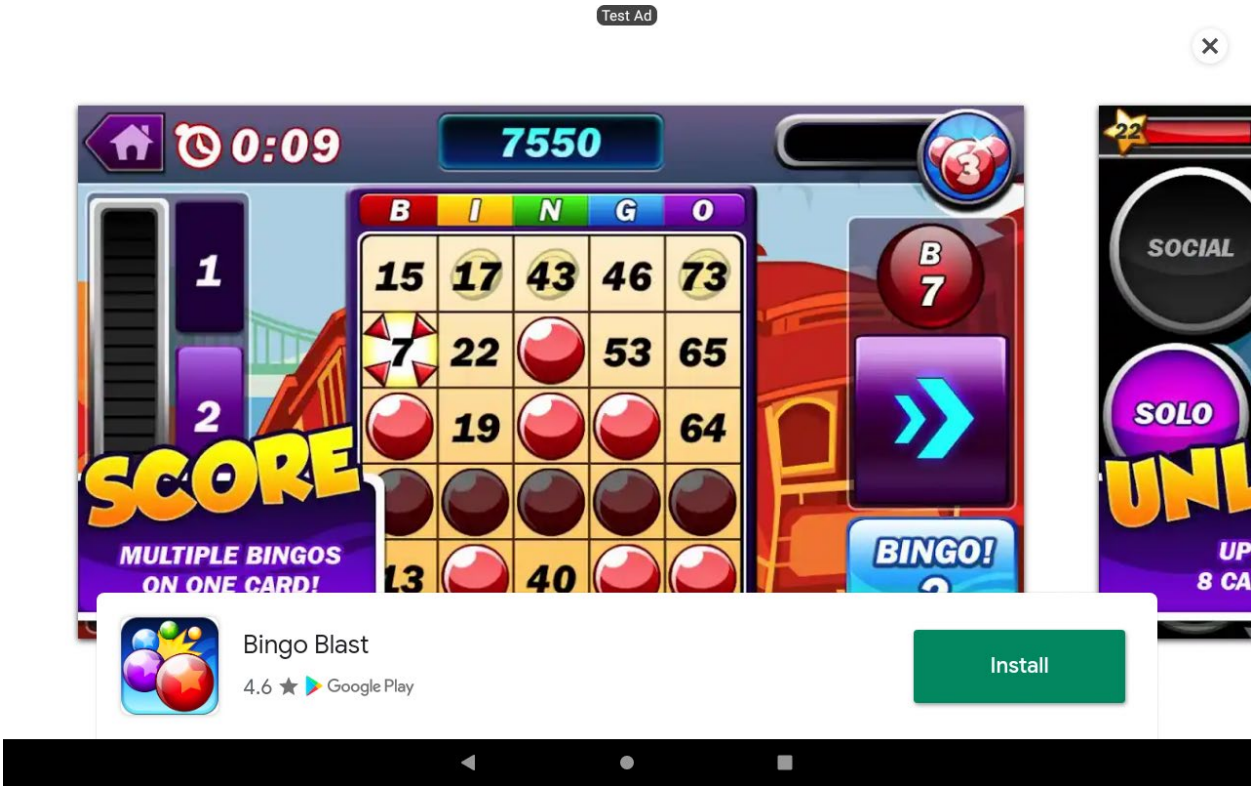
Screen 3 - This screen shows up when a ride is initiated:

Triggers for ride initiation are as follows:

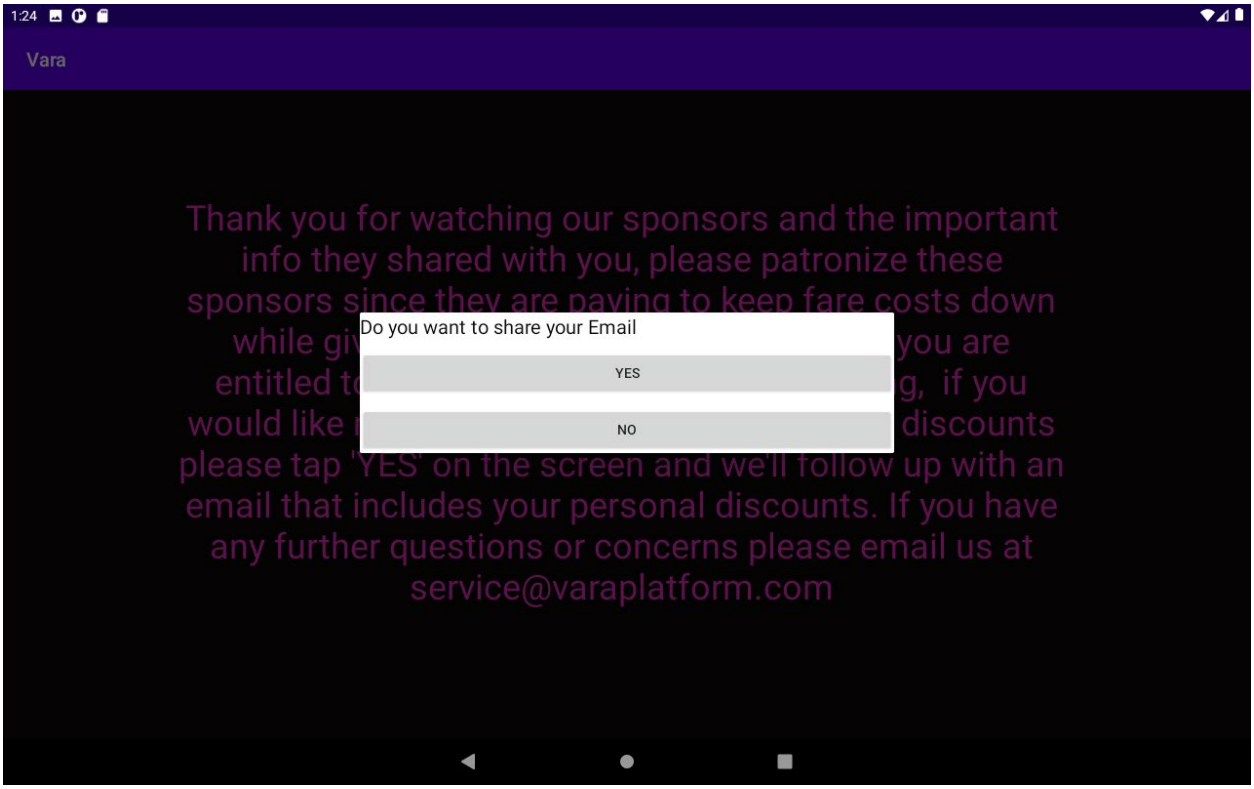
1. VARA platform user must be logged in to their account;
2. User must be connected through an authorized service partner*
3. Service Partner app will send a trigger event to the VARA Platform system that a client ride is starting.



***Screen 4* - Visual changes and layout updates to come and is not intended for the final production version.**



***The above are full page add samples coming from Google's ad service**

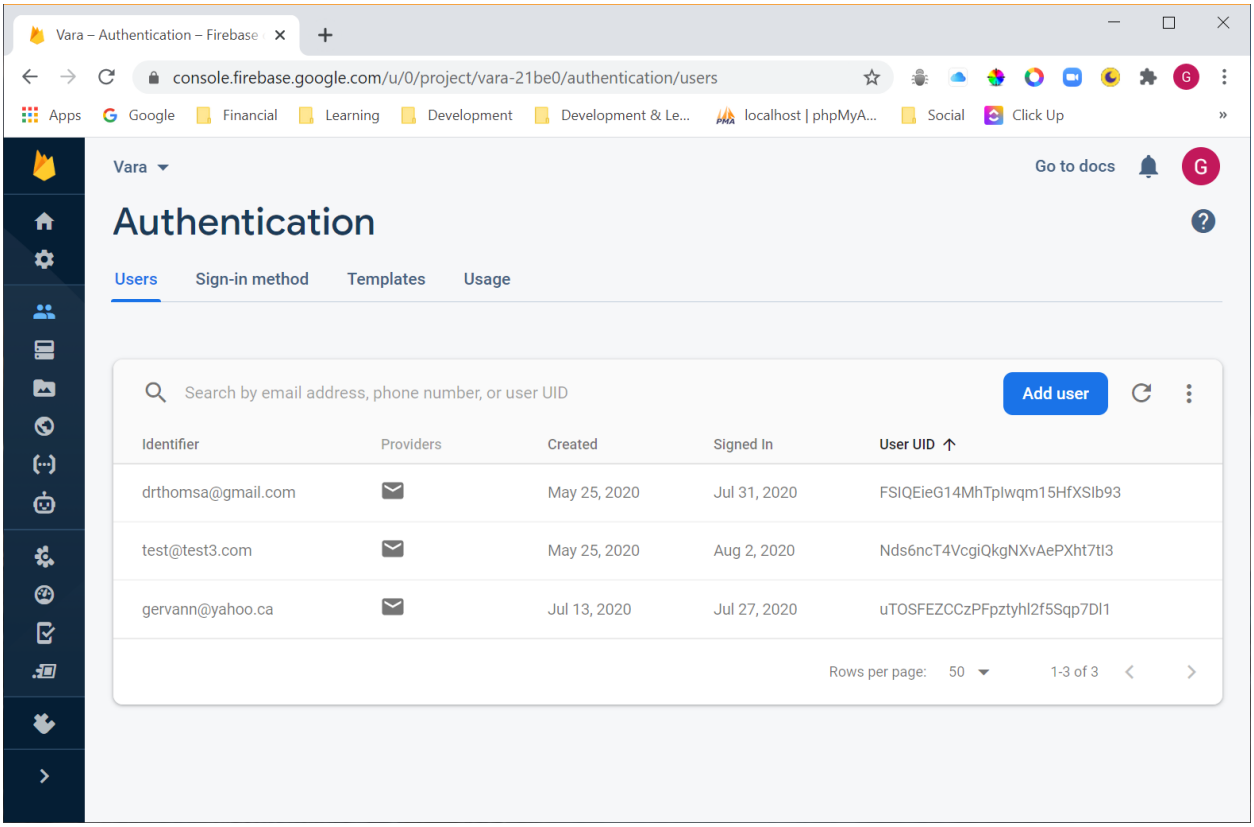


***the above is for mock up and not the final production version.**

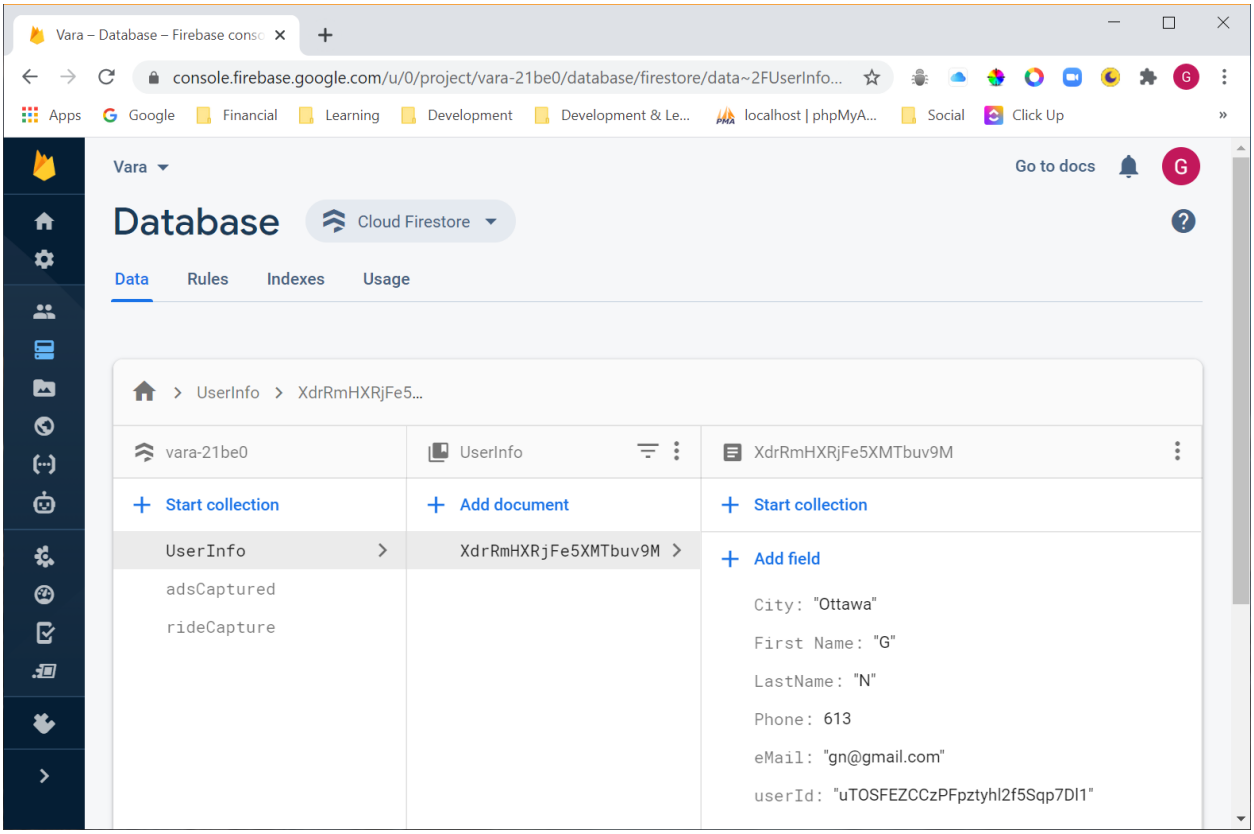
Screen 5 - This is the screen shown to the user upon the completion of the ride:

Triggers for the farewell page are as follows:

1. Service Partner app will send a trigger event to the VARA Platform system that a client ride has concluded (user has arrived at destination).
2. The screen will ask a user if they want to :
    a. Share their email with VARA Platform
    b. Do not wish to Share their information
3. User taps their option on screen by hitting 1 of buttons
4. Ride information and user options are captured from the ride
    a. User's selection is captured (Yes or No)
        i. If Yes – user's email address (provided by Partner app request event)
            1. Authorization flag set to true
        ii. If no – email identified as 'not provided'
            1. Authorization flag set to false (initial state)
    b. Ride information is captured
        i. Date & Time
        ii. How many adds were played
        iii. Ride client identification (transaction no by Partner app)

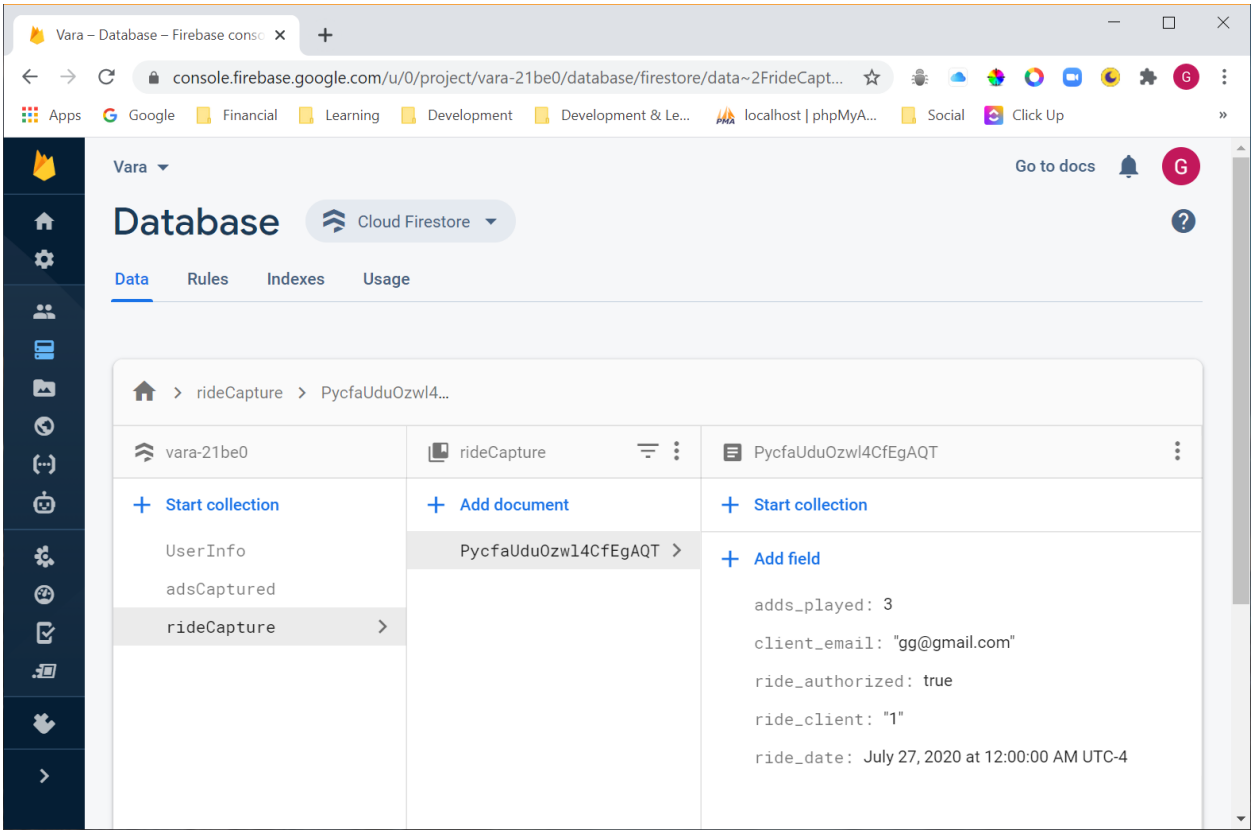


Above is the firebase cloud admin – where user log-in information is stored



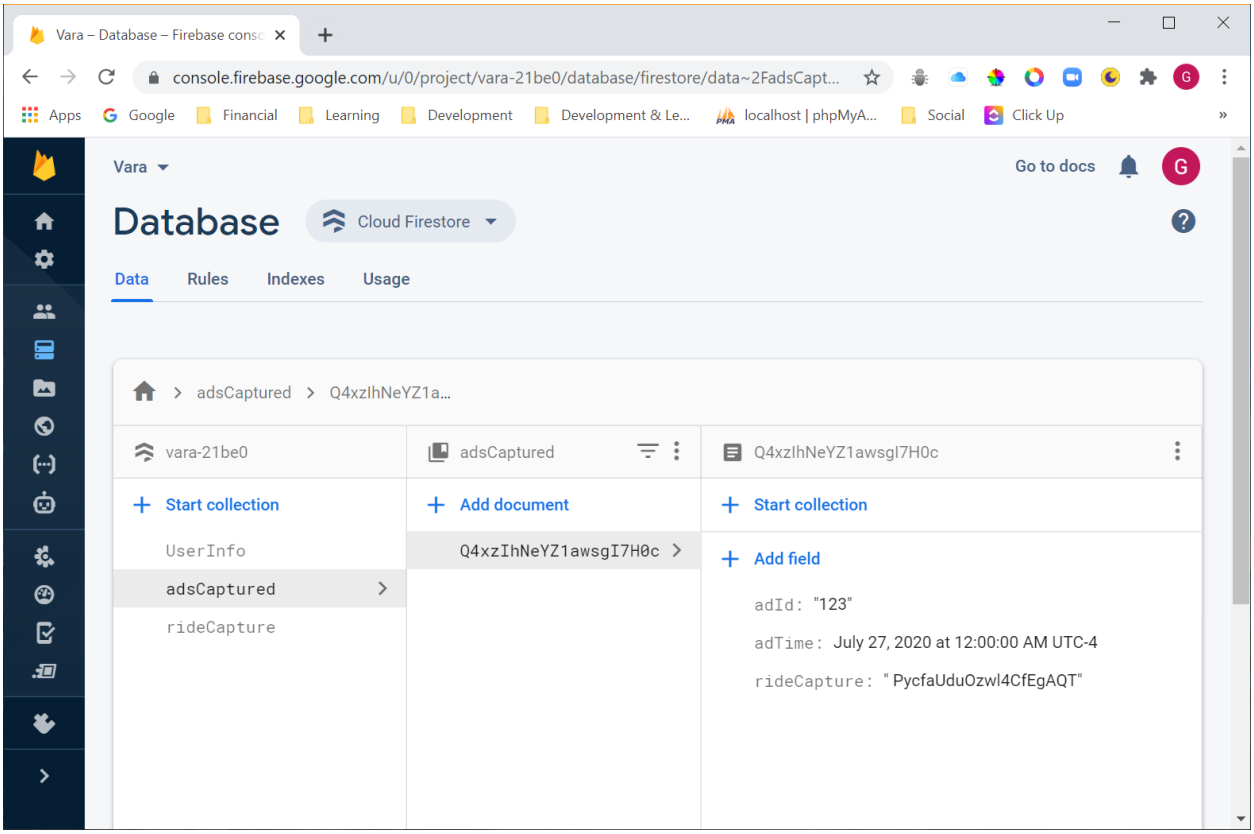
UserInfo fields:

1) City: Used to capture the city that the operator is working in for including localized targeted ads where applicable.
2) First Name: Driver's first name
3) Last Name: Driver's family name
4) Phone: Driver's phone number for text messages and as contact ID
5) eMail: Driver's email address for emailing and as contact ID
6) userID: The cross-reference to the driver's sign-in information on the system



rideCapture fields:

1) ads_played: Used to capture the number of ads played to the client during a recorded ride
2) client_email: Used to capture the client email upon client consent. This is to be provided by the partner application's API for the client being transported.
3) ride_authorized: Flag indicating client has provided consent to send them material with their email address.
4) ride_client: Identification number of clients. This is to be provided by the partner application's API for the client being transported.
5) ride_date: the date and time that the ride and advertising was provided to the client.



adsCaptured fields:

1) adId: The id or way that the system identifies the add that played. This is used as a
2) adTime: The date ad time the ad was played to the rider. This will correlate to the ride_date in the rideCapture collection.
3) rideCapture: This is the document ID to which this adCapture information belongs. The number of adsCaptured documents with matching rideCapture ids should match up with the number recorded in ads_played within each rideCapture